Paper

Application of Graph Theory Algorithms in Non-disjoint Functional Decomposition of Specific Boolean Functions

Tomasz Mazurkiewicz

Faculty of Cybernetics, Military University of Technology, Warsaw, Poland

https://doi.org/10.26636/jtit.2020.142520

Abstract—Functional decomposition is a technique that allows to minimize Boolean functions that cannot be optimally minimized using other methods, such as variable reduction and linear decomposition. A heuristic method for finding non-disjoint decomposition has been proposed lately. In this paper, we examine how the usage of different graph theory techniques affects the computation time and the quality of the solution obtained. In total, six different approaches were analyzed. The results presented herein prove the advantages of the proposed approaches, showing that results obtained for standard benchmark M-out-of-20 functions are better than those presented in previous publication. Results obtained for randomly generated functions prove that time complexity and scalability are significantly better when using the heuristic graph coloring algorithm. However, quality of the solution is worse, in general.

Keywords—logic synthesis, functional decomposition, non-disjoint decomposition, index generation functions.

1. Introduction

The growing use of Field Programmable Gate Arrays (FPGAs) has led to a significant increase in interest in efficient Boolean function minimization methods. Specific functions known as index generation functions (IGFs) have been gaining in popularity among researchers. Several examples of IGFs applications are presented in the literature [1], e.g. in connection with IP address tables, terminal access controllers and computer virus scanning circuits. Those functions represent the following mapping:

$$F : D^N \to \{1, 2, \dots, K\} \;, \tag{1}$$

where D^N is a set of K different binary N-bit vectors, called registered vectors. The function maps a unique integer value to each of them. If the input does not match any of the registered vectors, the function produces a zero value. The important property of index generation functions is that those functions are not fully defined. Thus, logic synthesis algorithms may be used to find a representation of the function using a number of variables that is lower than N. In particular, the optimum number of variables is:

$$\kappa = \lceil \log_2(K+1) \rceil \;. \tag{2}$$

Scientists focus mostly on two minimization techniques, i.e. reduction of variables and linear decomposition. However, those methods do not provide an optimized representation for all functions [1], [2]. Therefore, new solutions and papers concerned with these issues are emerging. Recent studies [3]–[6] investigate the application of functional decomposition in minimization of index generation functions. Functional decomposition [7] is a method, in which a Boolean function $F(X)$ with numerous input variables $X = \{x_1, x_2, \dots, x_n\}$ is divided into two functions (denoted in this paper as G and H functions) having fewer variables, i.e. $F = H(U, G(V, W))$. The decomposition scheme is shown in Fig. 1 [3]. The input variables of function G are called bound variables, while variables from set U are called free variables. The main problem is to obtain optimal sets of input variables for both functions. Memory utilization results are the best when the number of input variables of function H equals κ.

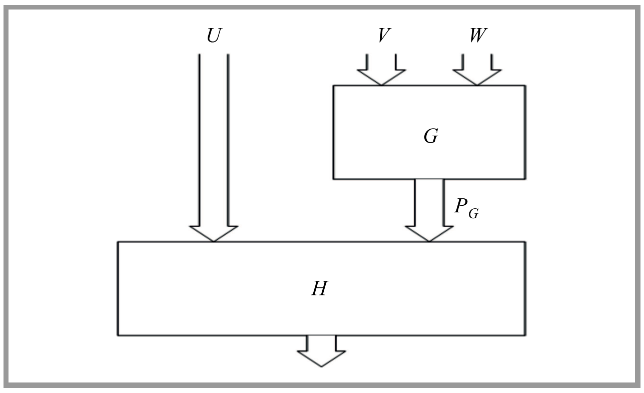


Fig. 1. Functional decomposition scheme.

The notion of partition algebra [8] and r-admissibility [9] may be used to address the problem of finding sets U and V form ($U \cup V = X$, $U \cap V = \emptyset$). However, it is often the case that the introduction of an additional set $W \subseteq U$ allows to further minimize a Boolean function, even though the number of inputs of function G was increased. Such a decomposition scheme is called non-disjoint decomposition. Therefore, proposing an efficient method for performing such a decomposition is very important. In particular, the form of set W and truth tables of functions G and H need to be found.

The exact method of finding functional decomposition consisting in using an SMT solver was proposed in [4]. However, it is a complex solution and may be used only for small values of K. Therefore, proposing a heuristic method is a very important task. The other method uses a graph coloring problem [3]. This paper examines how the usage of different techniques affects the computation time and quality of the received solution. It provides results obtained for random functions and standard benchmark IGFs, i.e. *M-out-of-N* that do not have an optimum linear decomposition [2] and cannot be minimized using variable reduction algorithms.

The rest of the paper is organized as follows. A brief overview of partition description, partition algebra, and r-admissibility is presented in Section 2. Section 3 introduces graph theory algorithms and describes how they are applied in finding functional decomposition. An evaluation of the proposed approaches is presented in Section 4. Section 5 concludes the paper.

2. Partition Algebra

The notion of partition algebra [8] and r-admissibility [9] is shortly introduced in this part of the paper. Let S be a finite set. Consider a collection of subsets of S. If their union is S, then we call it a cover of S. A blanket is a cover $\mathbb{B} = \{B_1, B_2, \ldots, B_k\}$ of a non-empty and distinct subset of S, called blocks, whose union is S. A set system is a blanket in which the blocks satisfy that $B_i \subseteq B_j \Rightarrow i = j$. A partition is a set system, where the blocks are disjoint, i.e. $B_i \cap B_j = \emptyset$ $(i \neq j)$.

Notice that for IGFs, the $F(X)$ is an isomorphic function between domain D^N and the set $T = \{1, 2, \ldots, K\}$. Each block of partition P_a includes these elements of the set T which have the same value at the x_a position.

Let P_a and P_b be two partitions on the same set. We define the following relation $\leq$ for partitions:

$$P_a \leq P_b \Leftrightarrow \forall B_i \in P_a : (\exists B_j \in P_b : B_i \subseteq B_j) \ . \tag{3}$$

Multiplication of the partitions may be defined, i.e. $P = P_aP_b$ if P is the partition with the largest blocks such that $P \leq P_a$ and $P \leq P_b$. This partition may be easily calculated by finding intersections of all blocks of P_a and P_b. The following theorem can be formulated using the theory introduced above.

Theorem 1: A function F has a functional decomposition $F = H(U, G(V))$ if and only if $\exists P_G \geq P_V : P_U P_G \leq P_F$ [8].

The quotient partition may be used to determine sets U and V. This is a partition of P_a over P_b whose blocks are those of P_a and elements are the elements of the product P_aP_b. We denote this partition $P_a|P_aP_b$. Using the quotient partition, we define r-admissibility of the set $\{P_1, P_2, \ldots, P_k\}$ in relation to the partition P_F in the following manner:

$$r = k + \lceil \log_2 \left(\gamma(P_1P_2 \ldots P_k | P_1P_2 \ldots P_kP_F) \right) \rceil \ , \tag{4}$$

where $\gamma(P_1P_2 \ldots P_k | P_1P_2 \ldots P_kP_F)$ is the number of elements in the largest block of the quotient partition. P_F corresponds to the characteristic partition of function F, i.e. the partition whose blocks include these elements that map onto the same value of $F(X)$. Notice that for any IGF, the characteristic partition equals $P_F = \{\overline{1}, \overline{2}, \ldots, \overline{K}\}$.

The concept of r-admissibility may be interpreted as follows: if a set of partitions is r-admissible, then there exists a decomposition in which function G has $r - k$ outputs, function H has r inputs and $W \subseteq U$ is also an input to function G. In the best case scenario, $W = \emptyset$. By choosing a set U in which r is as small as possible ($V = X \backslash U$), we may easily determine the number of input variables of function H. In that case, $k = |U|$ and function G has $r - |U|$ outputs. A necessary condition for r-admissibility is presented in lemma 1 [8].

Lemma 1: Set P is r-admissible if and only if all subsets of P are s-admissible, where $s \leq r$.

This lemma may be used to easily form set U. To demonstrate the application of r-admissibility, let us consider example 1.

Table 1
Example function

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

Example 1: Consider an IGF shown in Table 1 and notice that $K = 6$ and $N = 4$. The following partitions are calculated:

- $P_1 = \{\overline{1,3,5,6}; \overline{2,4}\} \Rightarrow r = 1 + \lceil \log_2(4) \rceil = 3$,
- $P_2 = \{\overline{1,2,4,6}; \overline{3,5}\} \Rightarrow r = 1 + \lceil \log_2(4) \rceil = 3$,
- $P_3 = \{\overline{1,4}; \overline{2,3,5,6}\} \Rightarrow r = 1 + \lceil \log_2(4) \rceil = 3$,
- $P_4 = \{\overline{2,5}; \overline{1,3,4,6}\} \Rightarrow r = 1 + \lceil \log_2(4) \rceil = 3$.

All partitions are 3-admissible. Thus, we may use partition multiplication to further search for functional decomposition of an input function. The following partitions are obtained:

- $P_1P_2 = \{\overline{1,6}; \overline{3,5}; \overline{2,4}\} \Rightarrow r = 2 + \lceil \log_2(2) \rceil = 3$,
- $P_1P_3 = \{\overline{1}; \overline{3,5,6}; \overline{4}; \overline{2}\} \Rightarrow r = 2 + \lceil \log_2(3) \rceil = 4$,
- $P_1P_4 = \{\overline{5}; \overline{1,3,6}; \overline{2}; \overline{4}\} \Rightarrow r = 2 + \lceil \log_2(3) \rceil = 4$.
- $P_2P_3 = \{\overline{1,4}; \overline{2,6}; \overline{3,5}\} \Rightarrow r = 2 + \lceil \log_2(2) \rceil = 3$,
- $P_2P_4 = \{\overline{2}; \overline{5}; \overline{1,3,4,6}\} \Rightarrow r = 2 + \lceil \log_2(3) \rceil = 4$,
- $P_3P_4 = \{\overline{1,4}; \overline{2,5}; \overline{3,6}\} \Rightarrow r = 2 + \lceil \log_2(2) \rceil = 3$.

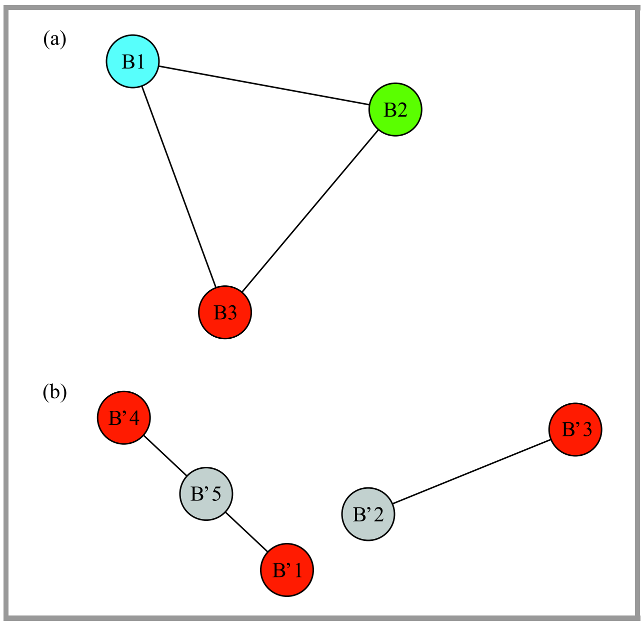


Fig. 2. Graphs obtained in two iterations: (a) graph representing an example function, (b) regenerated graph.

Based on lemma 1, it is known that there is no partition with $k = 3$ that is 3-admissible. Therefore, the following input variables may be used to form set U: x_1 and x_2. Since $V = X \setminus U$, we get the following set V: $\{x_3, x_4\}$. Moreover, the number of outputs from function G equals $r - |U| = 3 - 2 = 1$. Two different forms of set U could be used as well: $\{x_2, x_3\}$ and $\{x_3, x_4\}$. Both these pairs are also 3-admissible. A question regarding the existence of a disjoint decomposition of this function, i.e. whether $W = \emptyset$, is discussed in next section.

3. Graph Theory and Its Application

As mentioned in the previous section, r-admissibility may be used to determine the input variables to form U and V sets. Additionally, the minimal number of input variables to function H is found. The graph theory may be used to address two more problems:

1. which variables should form a set W?
2. what values are mapped to function G outputs?

Let $\Gamma = (N_\Gamma, E_\Gamma)$ be an undirected graph, where each node (called also a vertex) $v \in N_\Gamma$ represents a block of P_V and each edge $e = (B_i, B_j) \in E_\Gamma$ represents an unmergeable pair of $P_U|P_U P_F$ partition. Two blocks B_i and B_j are unmergeable if partition P_{ij} obtained from P_V by merging those two blocks satisfies the following condition: $P_U P_{ij} \nleq P_F$. In that case, calculating P_G (the characteristic partition of function G) consists in finding the coloring of the graph. The problem of graph coloring consists in assigning a color to each node, so that no two adjacent nodes have the same color assigned. The same value of function G is assigned to the nodes colored using the same color.

In order to find non-disjoint functional decomposition by using the graph coloring problem [3], the following steps are required:

1. calculate r-admissibility of an input function,
2. choose set U such that r is at its minimum,
3. $V = X \backslash U$,
4. generate graph Γ based on $P_U|P_U P_F$ and P_V partitions,
5. while the chromatic number of the graph $\chi(\Gamma)$ is larger than $2^{r-|U|}$ add a variable from set U to set W,
6. unless $W = U$, regenerate the graph and go back to the previous step.

It should be remembered that $r - |U|$ is the number of outputs of function G. Thus, the number of different values specified with so many variables is $\beta = 2^{r-|U|}$ at the most. Therefore, the chromatic number of graph Γ cannot be larger than β to represent a function using functional decomposition scheme.
The key part of the presented procedure is choosing the right variable to be added to set W. In the original paper [3], a node with the maximum degree is chosen. If there are several nodes with the same degree, we choose the first one found. Furthermore, this node must correspond to the block of P_V partition, which contains at least two elements p, q that $\exists_{e_1,e_2 \in E_\Gamma} : \{p\} \in e_1 \wedge \{q\} \in e_2$ and belongs to different blocks in $P_V P_i$ partition. As a result, $x_i \in U$ is added to set W. Different approaches to the choice of the node are analyzed further on in this section.
We use partition multiplication to calculate $P_{V'}$ partition and regenerate the graph, i.e. $P_{V'} = P_V P_i$. This new partition contains more blocks than a partition P_V. Thus, the regenerated graph has more nodes. On the other hand, the number of edges remains unchanged.

Example 2: Consider the function shown in Table 1. As reported in the previous section, we have $U = \{x_1, x_2\}$, $V = \{x_3, x_4\}$, and $r = 3$ through the application of r-admissibility. By using partition algebra, the following partitions are calculated:

- $P_U = P_1 P_2 = \{\overline{1,6}; \overline{3,5}; \overline{2,4}\}$,
- $P_U|P_U P_F = \{\overline{(1)(6)}; \overline{(3)(5)}; \overline{(2)(4)}\}$,
- $P_V = \{\overline{1,4}; \overline{2,5}; \overline{3,6}\} = \{B_1, B_2, B_3\}$.

The process of creating P_G is presented in Fig. 3. It starts with moving blocks B_1 and B_2 to the first block of P_G and a block B_3 to the second block. However, based on the $P_U|P_U P_F$ partition, element 4 should be separated from 2. Thus, the miniterm $\overline{1,4}$ needs to be divided. Since vectors 1 and 4 differ at position x_1, this variable is added to set W. To automate the process, we construct graph Γ. Notice that we get three blocks B_i (based on partition P_V), which will be represented by nodes in the graph. Furthermore, we get three unmergeable pairs: $(1,6)$, $(3,5)$ and $(2,4)$. Therefore, B_1 is unmergeable with B_3, since $1 \in B_1$ and $6 \in B_3$.

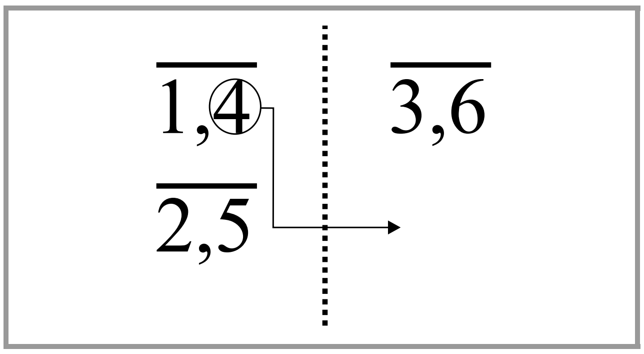


Fig. 3. Creation of P_G partition.

Similarly, B_3 is unmergeable with B_2 and B_2 is unmergeable with B_1. Thus, we get graph Γ presented in Fig. 2a. This graph is 3-colorable. However, $\chi(\Gamma) = 3 > 2^{r-|U|} = 2$. Thus, disjoint decomposition of this function does not exist, i.e. $W \neq \emptyset$.

In order to find non-disjoint decomposition, we need to add a variable to set W. We look for a variable that separates $1 \in B_1$ from $4 \in B_1$. Therefore, the node that represents block B_1 will be split into two nodes, which might lead to decreasing the chromatic number of the graph. It may be noticed in Table 1 that vectors one and two differ at position x_1. Therefore, we add this variable to set W. Having $W = \{x_1\}$, a new partition $P_{V'}$is calculated:

$$P_{V'} = P_V P_1 = \{\overline{1};\overline{4};\overline{2};\overline{5};\overline{3,6}\} = \{B^{'}_1, B^{'}_2, B^{'}_3, B^{'}_4, B^{'}_5\}.$$

Partition P_U remains the same. Graph Γ is now 2-colorable (see Fig. 2b). Thus, non-disjoint decomposition is found. The same output value from function G is assigned to the nodes colored using the same color. The truth tables obtained are presented in Table 2.

Table 2
Decomposed function

Function G			
x_1	x_3	x_4	G
0	1	0	0
1	0	1	0
0	0	0	1
1	1	0	1
0	0	1	0
0	0	0	1

Function F			
x_1	x_2	G	F
0	1	0	1
1	1	0	2
0	0	1	3
1	1	1	4
0	0	0	5
0	1	1	6

In the presented procedure, the node with the maximum degree is used to determine the variable to be added to set W. This method was based on the observation that $\chi(\Gamma) \leq \Delta(\Gamma) + 1$, where $\Delta(\Gamma)$ is the maximum node degree. Thus, reduction of the upper bound on the chromatic number by splitting the node with the maximum degree may lead to a lower value of $\chi(\Gamma)$. In this paper, we examine a different approach, i.e. the application of the maximum clique problem. This proposal is based on the fact that $\chi(\Gamma) \geq \omega(\Gamma)$, where $\omega(\Gamma)$ is the size of the maximum clique. Thus, by choosing a node from that clique, we may decrease the lower bound on the chromatic number.

Clique C of Γ is a subset of N_Γ such that every two vertices of C are adjacent, i.e. $\forall_{u,v \in C}\{u,v\} \in E_\Gamma$. The maximum clique problem (MCP) consists in finding a clique that is not contained in any other clique and its cardinality is the largest among all cliques. For example, in the graph presented in Fig. 2a, we have clique $C = \{B_1, B_2, B_3\}$ whose cardinality is equal to 3. This clique is the maximum. On the other hand, $C' = \{B_1, B_2\}$ is a clique as well, but $C' \subset C$. MCP is a well-studied problem. Various approaches have been proposed in the literature [10], both exact and heuristic, to solve this problem. The SageMaths [11] *clique_maximum()* function can be used to find the maximum clique in an undirected graph. It uses the Cliquer software [12] to solve the MCP problem by default, which implements an exact branch-and-bound algorithm.

The main weakness of this approach is that no polynomial-time (exact) algorithm is known for the MCP, i.e. it is classified as NP-complete. The fastest algorithm solves the problem in time $O(2^{0.249 \cdot |N_\Gamma|})$ [13]. Thus, it was decided that a random variable addition to set W should be analyzed. This method was chosen because it is a less time consuming approach. However, the solution quality might be low. Furthermore, each application of this method may lead to different results.

In paper [3], the exact coloring algorithm using mixed integer linear programming (MILP) was relied upon to find the coloring. However, usage of the exact coloring algorithm does not guarantee that the optimum result will be found [4]. Moreover, its complexity is high, i.e. it is an NP-complete problem. Thus, we examine the application of the heuristic coloring algorithm and its influence on the received results. In particular, the Welsh-Powell [14] algorithm is considered in this paper.

One of the recent studies [15] shows that it is a very efficient algorithm, but the solution quality is sometimes bad. Time complexity of this algorithm is $O(|N_\Gamma|^2)$. We will learn how relevant this is in the next section. Any other heuristic or meta-heuristic methods may be used to provide better time-efficiency than the exact coloring algorithm.

4. Evaluation

In this section, we evaluate six different approaches:

1. A1 – application of the maximum clique problem and the exact coloring algorithm,

Table 3
Linear decomposition results

K	κ	P_{avg}	P_{opt}
20	5	5.21	787
30	5	6.16	0
40	6	6.98	20
50	6	7.48	0

Table 4
Results for different values of K

Results for $K = 20$						
	A1	A2	A3	A4	A5	A6
$\|W\| = 0$	16	15	16	15	16	15
$\|W\| = 1$	103	78	103	67	99	83
$\|W\| = 2$	80	96	80	102	75	77
$\|W\| = 3$	14	23	14	29	23	38
$\|W\| = 4$	0	0	0	0	0	0
$\|W\|_{avg}$	1.43	1.60	1.43	1.68	1.49	1.65
$\|W\|_{rel}$	1.00	1.12	1.00	1.17	1.04	1.15
T_{rel}	2.49	1.59	2.39	1.67	1.64	1.00

Results for $K = 30$						
	A1	A2	A3	A4	A5	A6
$\|W\| = 0$	590	406	590	406	590	406
$\|W\| = 1$	259	379	247	373	261	387
$\|W\| = 2$	133	196	145	201	131	189
$\|W\| = 3$	3	3	3	3	3	3
$\|W\| = 4$	0	0	0	0	0	0
$\|W\|_{avg}$	0.54	0.79	0.55	0.80	0.54	0.79
$\|W\|_{rel}$	1.00	1.47	1.03	1.48	1.00	1.45
T_{rel}	2.96	1.45	2.77	1.19	2.78	1.00

Results for $K = 40$						
	A1	A2	A3	A4	A5	A6
$\|W\| = 0$	81	52	81	51	81	51
$\|W\| = 1$	210	151	216	165	216	156
$\|W\| = 2$	326	338	297	305	278	325
$\|W\| = 3$	294	319	302	337	313	301
$\|W\| = 4$	69	109	81	124	92	147
$\|W\|_{avg}$	2.06	2.29	2.09	2.34	2.12	2.34
$\|W\|_{rel}$	1.00	1.11	1.01	1.13	1.03	1.14
T_{rel}	2.37	1.25	2.06	1.15	1.94	1.00

Results for $K = 50$						
	A1	A2	A3	A4	A5	A6
$\|W\| = 0$	540	388	540	388	540	388
$\|W\| = 1$	270	201	275	208	263	207
$\|W\| = 2$	141	260	124	246	149	275
$\|W\| = 3$	40	138	48	142	42	125
$\|W\| = 4$	4	4	1	1	2	1
$\|W\|_{avg}$	0.69	1.16	0.68	1.15	0.70	1.14
$\|W\|_{rel}$	1.02	1.71	1.00	1.69	1.03	1.68
T_{rel}	21.82	1.53	22.10	1.25	22.87	1.00

2. A2 – application of the maximum clique problem and a heuristic coloring algorithm,

3. A3 – application of finding a node with maximum order and the exact coloring algorithm,

4. A4 – application of finding a node with maximum order and a heuristic coloring algorithm.

5. A5 – application of random variable selection and the exact coloring algorithm,

6. A6 – application of random variable selection and the heuristic coloring algorithm.

In this experiment we generated random IGFs with $N = 32$ and $K = \{20, 30, 40, 50\}$. For each value of K, 1000 functions were generated. All functions were firstly minimized using the linear decomposition algorithm [2]. The results obtained were presented in Table 3. P_{avg} denotes the average number of variables after linear decomposition and P_{opt} denotes the number of functions that have an optimum linear decomposition. For those functions that do not have an optimum linear decomposition, we looked for functional decomposition. Table 4 summarizes the results obtained. The following parameters are presented:

- $|W| = 0, 1, \ldots, 4$ – the number of functions with a particular size of set W,
- $|W|_{avg}$ – the average size of set W,
- $|W|_{rel}$ – the value calculated using the following equation: W_{avg}/W^{*}_{avg}, where W^{*}_{avg} is the minimum average size among all approaches,
- T_{rel} – the average computation time for a particular approach divided by the minimum average computation time of all approaches.

The most remarkable result to emerge from the data is that the application of MILP has led to the lowest average size of set W. No significant difference in the obtained results was observed between A1, A3, and A5 approaches. However, the approach proposed in the original paper [3] provided the best results only for $K = 20$ and $K = 50$. On average, the application of MCP has led to the best results, i.e. value of $|W|_{rel}$ obtained using this approach is the lowest for three different values of K.

Even though solution quality is very good, all these approaches are very time consuming. The average computation time for different values of K is presented in Fig. 4. The results presented confirm usefulness of the Welsh-Powell algorithm. Approaches relying on this algorithm guarantee the lowest computation time. Moreover, the computation time does not increase significantly depending on the value of K. This study shows that selection of the graph coloring algorithm is much more important than the approach adopted to select the variables. The use of a heuristic graph coloring algorithm leads to slightly worse solution quality. However, time complexity is much better, e.g. approaches relying on the Welsh-Powell algorithm were over 20 times faster for $K = 50$.

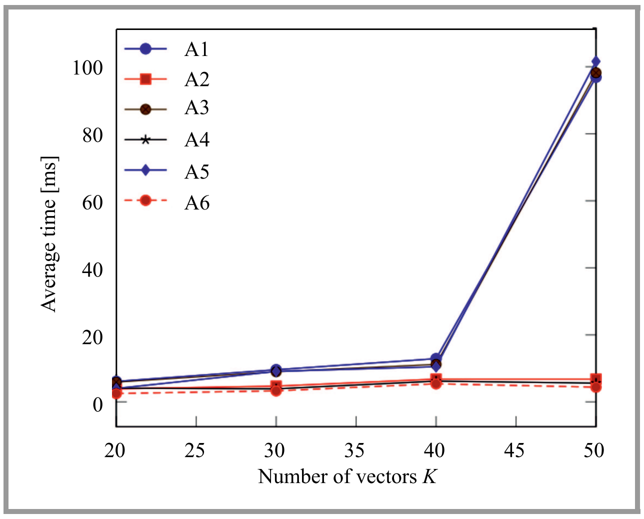


Fig. 4. Average computation time for different values of K.

Surprisingly, the number of functions whose decomposition led to $|W| = 4$ is large for $K = 40$. However, that number is close to zero for other values of K. The number of functions with $|W| = 2$ and $W = |3|$ is also quite high. Thus, the average size of set W is significantly higher. The reason for this rather extraordinary result is still unclear.

Table 5
Number of minimized functions using functional decomposition

K	A1	A2	A3	A4	A5	A6
20	213	212	213	213	213	213
30	985	984	985	983	985	985
40	980	969	977	973	980	980
50	995	991	988	985	996	996

The number of minimized functions using each approach is presented in Table 5. For $K = 20$, most functions have the optimum linear decomposition. Thus, we did not search for the functional decomposition of those functions. Notice that the second approach has led to one less successful functional decomposition, i.e. decomposition was not found because $W = U$. For higher values of K, most of the functions were minimized using the approaches described in this paper. Notice that the use of the exact coloring algorithm typically leads to a higher number of minimized functions. The single most significant observation to emerge from the results is that approaches A5 and A6 approaches result in the highest number.

Interestingly, for $K = 20$ and $K = 40$, we were able to find an optimum linear (Table 3) or a functional decomposition of every input function using approaches A1, A5, or A6 approaches.

The experiments confirm the assumption that random variable selection is time-efficient. The fastest approach for all values of K was A6. Contrary to expectations, quality of the solution obtained while using this approach was relatively good. However, this approach leads to different results in each execution of the algorithm.

Consider *M-out-of-N* functions. They are often chosen as benchmarks functions to evaluate the efficiency of IGF decomposition algorithms. They consist of $K = \binom{N}{M}$ vectors, whose length is N and Hamming weight is M. It was proved [1] that the optimum linear decomposition of a *2-out-of-20* coder does not exist. To the best of our knowledge, the optimum decomposition of a *4-out-of-20* function was not found either [2]. We use the described approaches to further minimize both functions. The inputs to the algorithm are truth tables of both functions after linear decomposition. The results obtained were presented in Table 6, where "–" means that the algorithm did not stop within the allocated time, i.e. the period of 60 minutes. Notice that the use of MILP for $M = 4$ has led to timeouts. Thus, the presented results prove that IGFs with many vectors should be minimized using a heuristic graph coloring algorithm.

In this paper, we do not analyze other standard benchmark functions, i.e. *M-out-of-16*, *1-out-of-20*, and *3-out-of-20* functions. Those functions have optimum linear [1], [2] or disjoint functional decomposition [4].

Table 6
Results for *M-out-of-20* functions

Approach	$M = 2,\ K = 190$		$M = 4,\ K = 4845$	
	Time [s]	$\lvert W\rvert$	Time [s]	$\lvert W\rvert$
A1	0.17	1	–	–
A2	0.05	2	19.33	3
A3	0.25	2	–	–
A4	0.04	2	17.92	3
A5	0.17	1–3	–	–
A6	0.02	2–3	7.98	2–4

For each approach, the computation time and the result obtained were presented. We analyze only the computation time of steps 3–5 from the procedure described in Section 3. We run the algorithm ten times, applying approaches A5 and A6. In the table, we present the minimum and the maximum number of variables in set W. Moreover, we present the time needed to find the solution with the minimum size of set W. Notice that for $M = 4$, the size of W might be equal to 2, 3 or 4. Similarly, for $M = 2$, quality of the solution varies. Thus, the algorithm relying on random variable selection needs to be executed multiple times in order to find the best solution.

The table is interesting in several ways. As far as we know, no other authors have found non-disjoint functional decomposition with $|W| = 1$ for a *2-out-of-20* function using a heuristic approach. Moreover, this solution is optimal. It was proved using the SMT-based exact method [4]. We also believe that the result obtained for a *4-out-of-20* function, i.e. $|W| = 2$, is the one that is best-known.

In order to highlight the significance of the results obtained, we analyze the total memory size using the scheme presented in Fig. 1. If both functions G and H are implemented using memories, the total size in bits equals:

$$\mathbb{S} = 2^{|V \cup W|} \cdot (r - |U|) + 2^r \cdot \kappa \ . \tag{5}$$

The results obtained are presented in Table 7. On average, the memory size was reduced by 10% using the approach presented in this paper due to the minimization of set W size. The results presented highlight the usefulness of the approach proposed in this paper.

Table 7
Memory size ($\mathbb{S}$) for *M-out-of-20* functions

M	Paper	$\|U\|$	$\|V\|$	$\|W\|$	r	$\mathbb{S}$
2	[4]	4	5	2	8	2560
	This paper	4	5	1	8	2304
4	[4]	6	9	3	13	135168
	This paper	6	9	2	13	120832

On the other hand, the memory size using linear decomposition only, equals:

$$\mathbb{S}_L = 2^p \cdot \kappa \ , \tag{6}$$

where p denotes the number of variables after linear decomposition, i.e. $K = 20 \Rightarrow p = 9$ and $K = 40 \Rightarrow p = 15$ [2]. The memory size for $K = 20$ is $\mathbb{S}_L = 4096$ bits. Functional decomposition has led to minimizing memory usage by 43.75%. Moreover, for $K = 40$, minimization by 71.63% has been achieved, i.e. $\mathbb{S}_L = 425984$. These results prove how important functional decomposition is.

During the experiment, we also tested whether the first-fit approach to variable selection leads to good results, i.e. variables were added in order of occurrence in set U. Our results were below expectations in terms of solution quality. For example, we obtained $|W| = 4$ for both *2-out-of-20* and *4-out-of-20* functions using the Welsh-Powell algorithm. On the other hand, this method was on average faster than other approaches. Since the solution quality achieved is bad, we decided to omit the results.

5. Conclusion

In this paper, application of the graph theory in non-disjoint decomposition of index generation functions was investigated. Several possible modifications of the original method were described [3] and the results obtained were analyzed. The research has highlighted the importance of selection of the graph coloring algorithm selection. In general, the results prove that methods using the Welsh-Powell algorithm are time-efficient and provide relatively good solution quality. The method using random variable selection to set W was the fastest one. On the other hand, approaches using the exact coloring algorithm led to the best results in terms of solution quality. However, they are much more time-consuming and led to timeouts for functions with a large set of input vectors.

Novel solutions were found for non-disjoint decomposition of standard benchmark functions, i.e. *2-out-of-20* and *4-out-of-20* functions. They guarantee minimization of memory usage by approximately 10% compared to previous results [4]. These results have further strengthened confidence in the usefulness of functional decomposition. A significant minimization of memory requirement is achieved compared to a scenario in which linear decomposition is applied only.

The approach relying on random variable selection is very fast and may lead to very good solutions. This method has some limitations stemming from the fact that each execution of the algorithm might lead to a different result. Therefore, this approach needs to be applied multiple times in order to assess the quality of the provided solution.

References

[1] T. Sasao, "Index generation functions: Minimization methods", in *Proc. 47th IEEE Int. Symp. on Multiple-Valued Logic ISMVL 2017*, Novi Sad, Serbia, 2017, pp. 197–206 (DOI: 10.1109/ISMVL.2017.22).

[2] T. Mazurkiewicz and T. Łuba, "Linear and non-linear decomposition of index generation functions", in *Proc. 26th Int. Conf. on Mixed Design of Integr. Circ. and Syst. MIXDES 2019*, Rzeszów, Poland, 2019, pp. 246-251 (DOI: 10.23919/MIXDES.2019.8787031).

[3] T. Mazurkiewicz and T. Łuba, "Non-disjoint decomposition using r-admissibility and graph coloring and its application in index generation functions minimization", in *Proc. 26th Int. Conf. on Mixed Design of Integr. Circ. and Syst. MIXDES 2019*, Rzeszów, Poland, 2019, pp. 252–256 (DOI: 10.23919/MIXDES.2019.8787118).

[4] T. Mazurkiewicz, "Non-disjoint functional decomposition of index generation functions", in *Proc. 50th IEEE Int. Symp. on Multiple-Valued Logic ISMVL 2020*, Miyazaki, Japan, 2020, pp. 137–142 (DOI: 10.1109/ISMVL49045.2020.00-16).

[5] T. Sasao, K. Matsuura, and Y. Iguchi, "A heuristic decomposition of index generation functions with many variables", in *Proc. of the 20th Worksh. on Synth. and Syst. Integr. of Mixed Inform. Technol. SASIMI 2016*, Kyoto, Japan, 2016 [Online]. Available: http://www.lsi-cad.com/sasao/Papers/files/SASIMI2016.pdf

[6] T. Sasao, K. Matsuura, and Y. Iguchi, "An algorithm to find optimum support-reducing decompositions for index generation functions", in *Proc. of Design, Autom. & Test in Europe Conf. & Exhibition DATE 2017*, Lausanne, Switzerland, 2017, pp. 812–817 (DOI: 10.23919/DATE.2017.7927100).

[7] H. A. Curtis, *New Approach to Design of Switching Circuits*, 1st ed. D. Van Nostrand, 1962 (ISBN: 9780442017941).

[8] J. A. Brzozowski and T. Łuba, "Decomposition of Boolean functions specified by cubes", *J. of Multi-Valued Logic & Soft Comput.*, vol. 9, pp. 377–417, 2003 [Online]. Available: http://maveric.uwaterloo.ca/reports/2003_JMVLSC_BrzozowskiLuba.pdf

[9] G. Borowik, T. Łuba, and P. Tomaszewicz, "A notion of r-admissibility and its application in logic synthesis", *IFAC Proc. Vol.*, vol. 42, no. 21, pp. 172–177, 2009 (DOI: 10.3182/20091006-3-ES-4010.00032).

[10] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems", *Eur. J. of Operat. Res.*, vol. 242, no. 3, pp. 693–709, 2015 (DOI: 10.1016/j.ejor.2014.09.064).

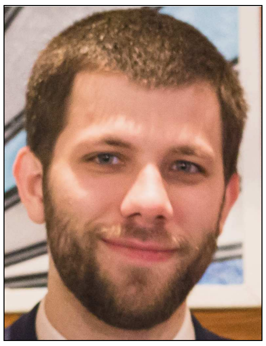
[11] The Sage Developers, "SageMath, the Sage Mathematics Software System (Version 8.3)" [Online]. Available: https://www.sagemath.org

[12] S. Niskanen and P. R. J. Ostergård, "Cliquer User's Guide, version 1.0", Tech. Rep. T48, Helsinki University of Technology, Department of Electrical and Communications Engineering, Communications Laboratory, Espoo, Finland, 2003 [Online]. Available: https://users.aalto.fi/~pat/cliquer/cliquer_fm.pdf

[13] J. M. Robson, "Finding a maximum independent set in time $O(2^{n/4})$", Tech. Rep., LaBRI, Université Bordeaux, 2001 [Online]. Available: https://www.labri.fr/perso/robson/mis/techrep.html

[14] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems", *The Computer J.*, vol. 10, no. 1, pp. 85–86, 1967 (DOI: 10.1093/comjnl/10.1.85).

[15] M. Aslan and N. A. Baykan, "A performance comparison of graph coloring algorithms", in *Proc. Int. Conf. on Adv. Technol. & Sci. ICAT'16*, Konya, Turkey, 2016, pp. 266–273 (DOI: 10.18201/ijisae.273053).



Tomasz Mazurkiewicz received his M.Sc. degree in Information Technology (specialty field – cryptology) from the Military University of Technology, Warsaw, Poland in 2014. He is currently enrolled in a Ph.D. program at the same University. His research interests are in logic synthesis and its application in FPGA-based electronic devices. His recent activities cover minimization methods of specific Boolean functions called index generation functions. He is the author and co-author of several papers dealing with this topic.

https://orcid.org/0000-0001-7305-2379

E-mail: tomasz.mazurkiewicz@wat.edu.pl

Faculty of Cybernetics
Military University of Technology
Kaliskiego 2
00-909 Warsaw, Poland