Cryptography and Communications
https://doi.org/10.1007/s12095-019-0351-8

An iterative method for linear decomposition of index generating functions

S. Hodžić[1] · E. Pasalic[2] · A. Chattopadhyay[3]

Received: 17 June 2018 / Accepted: 8 January 2019 / Published online: 14 January 2019

Abstract
Various methods for reducing hardware implementation cost of incompletely specified index generating functions have been proposed lately. Considering the methods based on linear decomposition, for the first time in this work, we provide necessary and sufficient conditions which describe the linear decomposition of these functions in general. These conditions are derived using the concept of functional degeneracy, and we show that the problem of linear decomposition can be translated into the problem of constructing suitable coordinate Boolean functions (which represent the generating functions) such that the linear decomposition is possible. In this context, we propose several design methods of such Boolean functions and furthermore we employ one particular design method to derive a new iterative semi-deterministic algorithm for linear decomposition. In addition, we provide a general result which describes all incompletely specified index generating functions for which the linear decomposition is (not) possible. Consequently, our results indicate that the functional degeneracy is a promising approach in derivation of new deterministic-like algorithms for linear decomposition of incompletely specified index generating functions.

Keywords (Incompletely specified) index generating functions · Linear decomposition · Linear injective mappings

Mathematics Subject Classification (2010) 06E30 · 65T50

✉ S. Hodžić
samir.hodzic@famnit.upr.si

E. Pasalic
enes.pasalic6@gmail.com

A. Chattopadhyay
anupam@ntu.edu.sg

[1] FAMNIT, University of Primorska, Koper, Slovenia

[2] FAMNIT & IAM, University of Primorska, Koper, Slovenia

[3] School of Computer Science and Engineering, College of Engineering, NTU, Singapore, Singapore

1 Introduction

Roughly speaking, an index generating function is a discrete integer-valued function defined on a (binary) vector space. Recently these functions have drawn a lot of attention due to their wide range of applications which encompasses address tables for internet routers, terminal access controller for local area networks, databases, memory patch circuits, virus scans, pattern matching, etc. [15, 18, 23]. The requirements on these functions are design dependent. For instance, in communication networks high-speed circuits (fast computation) are required along with frequent updates, which means that index generators have to be re-programmable. Thus a memory-based architecture of index generation functions is desirable.

In general, index generating functions can be implemented in different ways: by a content addressable memory (CAM) [14], a look-up table (LUT) or an index generating unit (IGU) [30] (see also [18, Section VIII-IX], [23, Chapter 12]). However, the implementation of index generating functions using some of the previously mentioned techniques may become impractical if the number of input variables is large due to increased memory costs. In many cases, an index generating function does not need to be defined on a whole domain (i.e., its values do not need to be completely specified), but rather it can be defined on a portion of the domain which leads to the notion of an *incompletely specified index generating function* (which we abbreviate as "$isig$-function"). More precisely, an $isig$-function, say f, is a mapping from $D = \{d_1, \ldots, d_k\} \subset \mathbb{F}_2^n$ to $\{1, \ldots, k\}$, where $f(d_i) = i$ and values $f(y) = c_y \geq 0$ are of no importance (commonly called *don't cares*) for $y \in \mathbb{F}_2^n \setminus D$. There exist various methods for reducing hardware implementation costs of these functions, which are briefly summarized as follows. The well-known method based on linear decomposition introduced by Nečiporuk [13] (which was later modified/extended in [8]), has been analyzed in [11, 12, 16, 18, 22, 23, 25]. The main goal of this method is to find suitable linear functions (so-called compound variables), say $y_1, \ldots, y_r$ (defined on $\mathbb{F}_2^n$) such that $f(x_1, \ldots, x_n) = \widetilde{f}(y_1, \ldots, y_r)$ $(r < n)$ for some incompletely specified ig-function $\widetilde{f}$. Another method similar to the previous one, considered in [20, 21, 24, 31], is based on finding a minimally sized set of so-called essential variables, say $\{x_{i_1}, \ldots, x_{i_r}\}$ $(1 \leq i_1 < \ldots < i_r \leq n)$, such that f can be written as $f(x_1, \ldots, x_n) = f(x_{i_1}, \ldots, x_{i_r})$. In this case, the essential variables are considered to be compound variables of weight equal to 1. There also exist somewhat different approaches such as: a gate-based decomposition method which uses suitable partitioning of the set D [9], methods based on spectral measures [6, 8, 26, 27, 35, 37], methods which utilize partially defined Boolean functions [4, Section 12] (see for instance [4, Example 12.3]), algebraic methods which employ vector spaces and polynomials [1, 2]. For further works related to index generating functions, we refer to references given in [25, Section IX].

In most of the cases the proposed algorithms are inefficient when applied to an arbitrary given $isig$-function (i.e., set D). Moreover, many of these algorithms are heuristic and perform well only for small sized D, and the optimality of output results can not be proved or guaranteed. Another drawback of (almost) all methods mentioned above is the fact that there still does not exist an algebraic treatment of these functions that would provide some explicit necessary-sufficient conditions for their linear decompositions (thus implying some efficient decomposition methods). This justifies the difficulty of the problem in general and at the same time the importance of finding an effective decomposition algorithm (heuristic or any other type) applicable to arbitrary $isig$-functions (as pointed out by Sasao et al. [12]).

In this work, we introduce and analyze the linear decomposition method based on functional degeneracy, the concept described in [38, Theorem 2.9] (or [8, Corollary 2.1]).

Considering an $isig$-function f in a general form as $f(x) = c_1\xi_1(x) + \ldots + c_s\xi_s(x)$ (ξ_i are Boolean functions, c_i are non-negative integers), for the first time (in Section 3) we derive necessary and sufficient conditions under which the function f can be linearly decomposed in less variables (regardless of whether they are compound or essential). Thus we show that the problem of finding linear decomposition of f can be reduced to the problem of finding suitable functions ξ_i (and integers c_i) which admit the transformation (decomposition) of f to an $isig$-function $\widetilde{f}$ defined in $r < n$ compound/essential variables.

Furthermore, for various cases (in Section 4) we show how one can linearly decompose a given $isig$-function (using characteristic Boolean functions), where for a sufficiently small number of linearly independent vectors in the set D (which we throughout the article define as dimension of D) we prove that the linear decomposition is always possible (Theorem 4.2). For some additional specific scenarios which regard the set D, we provide explicit construction methods of coordinate functions ξ_i which admit a linear decomposition of underlying $isig$-function. On the other hand, we describe $isig$-functions (i.e., sets D) for which the linear decomposition is not possible in general, and thus we generalize the method of proving the minimality of a given linear decomposition by SAT solver [5] (Theorem 4.3).

The previously mentioned results will lead us to a new iterative semi-deterministic algorithm (given in Section 5), which can be utilized for deducing either compound or essential variables. The analysis of our approach shows that in the former case our algorithm provides quite efficient solutions by giving nearly optimal number of compound variables with degree ≤ 3 for various values of n and relatively large k (cf. Table 6). On the other hand, for the latter case it provides somewhat weaker results which are of the similar quality as those obtained by [1] (cf. Tables 7, and 8).

To the best of our knowledge, this work is the first treatment of incompletely specified ig-functions as algebraic objects which provides general (non)existence results on linear decompositions, along with construction methods of coordinate functions ξ_i which may give rise to new iterative deterministic algorithms (with possible further improvements).

2 Preliminaries

The binary vector space of all n-tuples $x = (x_1, \ldots, x_n)$ ($x_i \in \mathbb{F}_2 = \{0, 1\}$) with the standard operations is denoted by $\mathbb{F}_2^n$. The complement of $x_i \in \mathbb{F}_2$ we denote by $\overline{x}_i = x_i \oplus 1$. For $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{F}_2^n$, the scalar (or inner) product over $\mathbb{F}_2$ is defined as $x \cdot y = x_1y_1 \oplus \cdots \oplus x_ny_n$. The Hamming weight of $x = (x_1, \ldots, x_n) \in \mathbb{F}_2^n$ is denoted and computed as $wt(x) = \sum_{i=1}^{n} x_i$. By "$\sum$" we denote the integer sum (without modulo evaluation), whereas "$\bigoplus$" denotes the sum evaluated modulo 2. We take that the set of vectors $\mathbb{F}_2^n$ is ordered lexicographically as $\mathbb{F}_2^n = \{z_0 = (0, \ldots, 0, 0), z_1 = (0, \ldots, 0, 1), \ldots, z_{2^n-1} = (1, \ldots, 1, 1)\}$. Thus, an arbitrary vector $z_j \in \mathbb{F}_2^n$ is represented as $z_j = (z_{j,1}, \ldots, z_{j,n})$ ($j \in [0, 2^n - 1]$), with $j = \sum_{t=0}^{n-1} z_{j,n-t}2^t$ being its integer representation. The all zero vector $(0, \ldots, 0) \in \mathbb{F}_2^n$ we denote by $\mathbf{0}_n$. Throughout the article, by $[i, j]$ (for $i < j$, $i, j \in \mathbb{Z}$) we denote the interval of integers between i and j, i.e., $[i, j] = \{i, i + 1, i + 2, \ldots, j\}$.

2.1 Boolean functions

The set of all Boolean functions in n variables, which is the set of mappings from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, is denoted by $\mathcal{B}_n$. Especially, the set of affine functions in n variables is given by $\mathcal{A}_n = \left\{a \cdot x \oplus b \mid a \in \mathbb{F}_2^n,\ b \in \{0, 1\}\right\}$, and similarly $\mathcal{L}_n = \left\{a \cdot x : a \in \mathbb{F}_2^n\right\} \subset \mathcal{A}_n$ denotes the

set of linear functions. The support of an arbitrary function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as $supp(g) = \{x \in \mathbb{F}_2^n : g(x) = 1\}$. For a vector $v = (v_1, \ldots, v_n) \in \mathbb{F}_2^n$, let the *minterm* function $m_v : \mathbb{F}_2^n \to \mathbb{F}_2$ [36] be defined as $m_v(x_1, \ldots, x_n) = \prod_{j=1}^n (x_j \oplus v_j \oplus 1)$. Note that $m_v(x) = 1$ if and only if $x = v$. It is well-known that any Boolean function $g : \mathbb{F}_2^n \to \mathbb{F}_2$ can be uniquely represented as

$$g(x_1, \ldots, x_n) = \bigoplus_{v \in supp(g)} m_v(x_1, \ldots, x_n). \tag{1}$$

For an arbitrary function $g \in \mathcal{B}_n$, the set of its values on $\mathbb{F}_2^n$ (*the truth table*) is defined as $T_g = (g(0, \ldots, 0, 0), g(0, \ldots, 0, 1), \ldots, g(1, \ldots, 1, 1))$. For a function g defined on $\mathbb{F}_2^n$ and a given subset $S \subseteq \mathbb{F}_2^n$, we denote by $g|_S$ the restriction of g to S (that is $g|_S = g : S \to \mathbb{F}_2$). The *Walsh transform* (WT) of $g \in \mathcal{B}_n$ at any point $\omega \in \mathbb{F}_2^n$ is defined by

$$W_g(\omega) = \sum_{x \in \mathbb{F}_2^n} g(x)(-1)^{\omega \cdot x}. \tag{2}$$

The *Walsh support* of $g \in \mathcal{B}_n$ is defined as $S_g = \{\omega \in \mathbb{F}_2^n : W_g(\omega) \neq 0\}$. A *vectorial Boolean function*, say $h : \mathbb{F}_2^n \to \mathbb{F}_2^k$, can be represented uniquely as $h(x) = (h_1(x), \ldots, h_k(x))$, where $h_i : \mathbb{F}_2^n \to \mathbb{F}_2$.

For a subset $E \subseteq \mathbb{F}_2^n$, *the indicator function* ϕ_E will denote the Boolean function such that $\phi_E(x) = 1$ if and only if $x \in E$. Also, by $E^\perp$ we denote the set $E^\perp = \{y \in \mathbb{F}_2^n : x \cdot y = 0, \ \forall x \in E\}$. The cardinality of any set E is denoted by $\#E$. By [8, Corollary 2.2] and [7, Section IV], for a given affine subspace $b \oplus E \subseteq \mathbb{F}_2^n$ ($b \in \mathbb{F}_2^n$ is arbitrary, E is a subspace, $dim(E) = p$), we have that WT at any $u \in \mathbb{F}_2^n$ and ANF of $\phi_{b \oplus E}$ are given respectively as

$$W_{\phi_{b \oplus E}}(u) = \begin{cases} \#E \cdot (-1)^{u \cdot b}, & u \in E^\perp \\ 0, & u \notin E^\perp \end{cases}, \qquad \phi_E(x) = \prod_{i=1}^{n-p} (\tau_i \cdot x \oplus 1), \tag{3}$$

where $\{\tau_1, \ldots, \tau_{n-p}\}$ is any basis of $E^\perp$. For a subspace E and $b \in \mathbb{F}_2^n$, it holds that $\phi_{b \oplus E}(x) = \phi_E(x \oplus b)$.

2.2 Index generating functions

Let $D = \{d_1, \ldots, d_k\} \subset \mathbb{F}_2^n$ such that $\#D = k \ll 2^n$, where the (distinct) vectors $d_i \in D$ are called *registered vectors*. In general, an $isig$-function $f : D \to \{1, 2, \ldots, k\}$ can be represented by *decomposition table* (see for instance [15, 29]) or using relation (1) as

$$f(x) = \sum_{d_i \in D} i \cdot m_{d_i}(x). \tag{4}$$

Remark 2.1 Since the values $f(y) = c_y$ for $y \notin D \subset \mathbb{F}_2^n$ are of no importance (and thus can be arbitrary), in (4) we omit their specification. Although (4) means that $c_y = 0$ for all $y \in \mathbb{F}_2^n \backslash D$ (due to the fact that $m_{d_i}(x) = 1$ if and only if $x = d_i$), this representation will not mean that f is completely specified on whole $\mathbb{F}_2^n$. Thus in our work relation (4) strictly refers to the representation of an $isig$-function $f = f|_D$.

With the following example we demonstrate the representation (4) of an $isig$-function defined on $D \subset \mathbb{F}_2^4$.

Table 1 Registered vector table

$d_i \in D$	x_1	x_2	x_3	x_4	$f(d_i)$
d_1	0	0	0	1	1
d_2	1	0	1	1	2
d_3	1	1	0	0	3
d_4	0	1	1	1	4

Example 2.1 Let the set of registered vectors $D \subset \mathbb{F}_2^4$ and an $isig$-function $f : D \to \{1, 2, 3, 4\}$ be given by Table 1, where $f(d_i) = i$ for $i \in [1, 4]$. The function $f = f|_D$ (Remark 2.1) can be represented as $f(x) = 1 \cdot m_{d_1}(x) + 2 \cdot m_{d_2}(x) + 3 \cdot m_{d_3}(x) + 4 \cdot m_{d_4}(x)$. Alternatively, the function f can be represented by a decomposition table as in [15, Example 3 - Fig. 3b].

Definition 2.1 For a function $f \in \mathcal{B}_n$, we say that $f(x_1, \ldots, x_n)$ *depends on* x_i if there exists a pair of vectors $a = (a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$ and $a' = (a_1, \ldots, a_{i-1}, a_i', a_{i+1}, \ldots, a_n)$ such that f is defined for both a and a', $a_i \neq a_i'$, and $f(a) \neq f(a')$. If f depends on x_i, then x_i is said to be an *essential variable*.

2.3 The functional decomposition

A *decomposition* or *transformation* (linear or non-linear) of an $isig$-function $f : D \subset \mathbb{F}_2^n \to \{1, \ldots, k\}$ is said to be a vectorial Boolean function $h = (y_1, \ldots, y_r) : \mathbb{F}_2^n \to \mathbb{F}_2^r$ which transforms a function f into an $isig$-function $\widetilde{f} : D \to \{1, \ldots, k\}$ in r variables as

$$f(x_1, \ldots, x_n) = (\tilde{f} \circ h)(x) = \widetilde{f}(y_1(x), \ldots, y_r(x)) = \widetilde{f}(y_1, \ldots, y_r), \tag{5}$$

where $y_i = y_i(x_1, \ldots, x_n)$, $i \in [1, r]$. A linear transformation, which induces a vectorial Boolean function $h = (y_1, \ldots, y_r)$, with minimal possible value of r is called *optimal*. Recall that by [24, Theorem 1] any incompletely specified ig-function with weight k necessarily requires at least $\lceil \log_2 k \rceil$ variables to be represented. Alternatively, in terms of its linear decomposition, it holds that $r \geq \lceil \log_2 k \rceil$. By *minimal* linear decomposition we refer to a decomposition which is not necessarily optimal but for which the number of compound variables can not be reduced further by any linear transformation in general (see for instance the set $D^{(2)}$ in Example 5.2).

Assuming that the vectorial function $h = (y_1, \ldots, y_r)$ in (5) (associated to f) is linear, then the functions y_i, $i \in [1, r]$, which are called *compound variables*, can be written as $y_i(x) = \lambda_i \cdot x$, for some vectors $\lambda_i \in \mathbb{F}_2^n$. Then, the *compound degree* of these functions is defined as $wt(\lambda_i)$. Thus, one can define a binary matrix $H_{n \times r}$ as $H = [\lambda_1^T, \ldots, \lambda_r^T]$, where λ_i^T is a column vector, i.e., the transpose of λ_i. Consequently, the function f can be written as $f(x) = g(xH)$, where $g \in \mathcal{B}_r$, and if $r < n$ then the function f is said to be *algebraically degenerate* or briefly *degenerate*.[1] If there does not exist such an $n \times r$ matrix H with $r < n$ such that $f(x) = g(xH)$ holds, i.e., the minimum value of r is n, then f is said to be *algebraically nondegenerate*.

[1] From the point of cryptographic applications, this term was introduced by Mitchell [10].

3 Linear decompositions based on degeneracy

Using the results given in [38, Section 2.3], in this section we derive necessary and sufficient conditions for linear decomposition of an arbitrary $isig$-function. Since our method uses the notion of algebraic degeneration of Boolean functions, we firstly discuss a misleading result which can be related to work presented in [26] (see Remark 3.1). Let us first recall two results given in [38, Theorems 2.9] (which is [8, Corollary 2.1]) and [38, Theorems 2.10].

Theorem 3.1 [8, 38] *Let $f \in \mathcal{B}_n$. Denote by $V = \langle S_f \rangle$ the linear span of the nonzero spectrum points of f. Assume that $dim(V) = r$, and let $\mu_1, \ldots, \mu_r$ be a basis of V which defines a binary matrix $M_{n\times r}$ as $M = [\mu_1^T, \ldots, \mu_r^T]$. Then there exists a Boolean function $g(y)$ in r variables such that*

$$g(y) = g(xM) = f(x), \quad x \in \mathbb{F}_2^n.$$

Theorem 3.2 [38] *Let $f \in \mathcal{B}_n$. Then $dim\langle S_f \rangle = r$ if and only if f can be algebraically degenerated into a function in r variables.*

Based on the results given in [26, Section V] (which are given in terms of autocorrelation transform, see [26, Definition 5.1]), one can conclude that an $isig$-function f and the corresponding characteristic (Boolean) function ϕ_D can be represented in the same number of variables. However, with the following example we show that a linear decomposition (which is degeneracy) of the characteristic function ϕ_D does not imply the degeneracy of f in general, and thus Theorem 3.1 can not be (always) applied to ϕ_D in order to induce a linear decomposition of f.

Example 3.1 Let the set $D \subset \mathbb{F}_2^4$ be given as $D = \{(1, 0, 0, 0), (0, 1, 0, 0), (0, 1, 1, 0), (1, 1, 0, 1)\}$. Then, it is not difficult to check that the characteristic function ϕ_D of the incompletely specified ig-function $f : D \to \{1, 2, 3, 4\}$ has the Walsh support S_{ϕ_D} of dimension 4 (thus $dim\langle S_{\phi_D} \rangle = 4$). According to Theorem 3.2, the function ϕ_D cannot be degenerated into a function $g \in \mathcal{B}_r$ for $r < n$. However, by Example 4.2 in [31], the function f can be represented by 3 variables (which is the minimal possible number in this case), say $\{x_1, x_2, x_3\}$, $\{x_1, x_3, x_4\}$ or $\{x_2, x_3, x_4\}$.

Remark 3.1 Note that the only case when the degeneracy of ϕ_D implies the degeneracy of f is when ϕ_D has a linear decomposition in compound variables with degree equal to 1 [26].

In general, an arbitrary $isig$-function can be represented in different ways. For instance, the $isig$-function $f : D = \{d_1, d_2, d_3, d_4\} \to \{1, 2, 3, 4\}$ given in Example 3.1 can be represented by relation (4) as $f(x) = 1 \cdot m_{d_1}(x) + \ldots + 4 \cdot m_{d_4}(x)$, but it can also be represented as $f(x) = 1 + a_0(x) + 2a_1(x)$, where a_i are defined as in Table 2.

In order to set our results in a more general framework, we assume that an $isig$-function $f : D \subset \mathbb{F}_2^n \to \{1, \ldots, k\}$ can be represented in a general form as $f(x) = c_1\xi_1(x) + \ldots, c_s\xi_s(x)$, where $\xi_i \in \mathcal{B}_n$ are called coordinate functions (c_i are non-zero integers, $x \in \mathbb{F}_2^n$). However, the condition imposed on functions ξ_i, in order to properly define the $isig$-function f, is given in the following result.

Proposition 3.1 *Let $f : D = \{d_1, \ldots, d_k\} \to \{1, \ldots, k\}$ be an arbitrary incompletely specified ig-function defined as $f(d_i) = i$, $i \in [1, k]$ ($D \subset \mathbb{F}_2^n$). For integers $c_1, \ldots, c_s$*

Table 2 Definition of coordinate functions a_0 and a_1 on D from Example 3.1

$d_i \in D$	x_1	x_2	x_3	x_4	$a_0(d_i)$	$a_1(d_i)$	$f(d_i)$
d_1	1	0	0	0	0	0	1
d_2	0	1	0	0	1	0	2
d_3	0	1	1	0	0	1	3
d_4	1	1	0	1	1	1	4

($s \geq 1$) and Boolean functions $\xi_1, \ldots, \xi_s \in \mathcal{B}_n$, define matrices $\Gamma_{k\times s} = [\xi_i(d_j)]$ and $C_{s\times 1} = [c_i]$ for $i \in [1, s]$, $j \in [1, k]$. If it holds that the restrictions of ξ_i on D satisfy the system

$$\Gamma_{k\times s} C_{s\times 1} = \begin{pmatrix} \xi_1(d_1) & \ldots & \xi_s(d_1) \\ \xi_1(d_2) & \ldots & \xi_s(d_2) \\ \vdots & \vdots & \vdots \\ \xi_1(d_k) & \ldots & \xi_s(d_k) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ \vdots \\ k \end{pmatrix}, \tag{6}$$

then f can be represented as $f(x) = c_1\xi_1(x) + \ldots, c_s\xi_s(x)$, $x \in \mathbb{F}_2^n$. Equivalently, for every collection of Boolean functions $\xi_1, \ldots, \xi_s$ such that $\sum_{i=1}^{s} c_i\xi_i(d_j) = j$ ($j \in [1, k]$), it necessarily holds that $(\xi_1|_D, \ldots, \xi_s|_D)$ is an injective mapping on D.

Proof Let us assume that $(\xi_1|_D, \ldots, \xi_s|_D)$ is not injective on D. Then there exist $d', d'' \in D$ ($d' \neq d''$) for which it holds that $(\xi_1|_D(d'), \ldots, \xi_s|_D(d'))$ and $(\xi_1|_D(d''), \ldots, \xi_s|_D(d''))$ are equal (binary) vectors, which gives that $b' = \sum_{i=1}^{s} c_i\xi_i(d') = \sum_{i=1}^{s} c_i\xi_i(d'') = b''$. However, this is in contradiction with $b' \neq b''$, which completes the proof. □

Notice that not all representations of f (referring to coefficients c_i), using some injective vectorial mapping $(\xi_1|_D, \ldots, \xi_s|_D)$, are suitable to express f correctly in this context. For instance, if $f(x) = 1\xi_1(x) + 2\xi_2(x) + 3\xi_3(x)$, $(\xi_1|_D, \xi_2|_D, \xi_3|_D)(d') = (0, 0, 1)$ and $(\xi_1|_D, \xi_2|_D, \xi_3|_D)(d'') = (1, 1, 0)$ (for some $d', d'' \in D$, $d' \neq d''$), we have that $f(d') = f(d'') = 3$, which is not allowed for an incompletely function $isig$-function f.

Remark 3.2 Since (6) is a system of linear Diophantine equations, then all results which characterize the existence of solutions for such systems can be applied here. However, the problem of finding Boolean functions ξ_i (in the representation of f) and integers c_i such that (6) holds can become rather complex due to the facts that $\Gamma_{k\times s}$ has to be a binary matrix and $C_{s\times 1}$ contains non-negative values. Note also that the problem of finding a linear vectorial function which is injective on D is proved to be NP-complete (see [34, Theorem 3.1]).

Remark 3.3 In the case when an $isig$-function f is represented by (4), then the matrix $\Gamma_{k\times k} = [\xi_i(d_j)]$ in (6) (where $s = k$) is the identity matrix.

The following result (which is based on Theorem 3.1) characterizes the linear decomposition method of an arbitrary $isig$-function f in terms of Walsh supports of its coordinate Boolean functions ξ_i.

Theorem 3.3 *Let $f : D = \{d_1, \ldots, d_k\} \to \{1, \ldots, k\}$ ($D \subset \mathbb{F}_2^n$) be an arbitrary incompletely specified ig-function defined by $f(d_i) = i$, $i \in [1, k]$ ($D \subset \mathbb{F}_2^n$), and rep-*

resented as $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$ $(c_i \geq 0)$. *For* $i \in [1, s]$, *let* B_i *be a basis of* $\langle S_{\xi_i} \rangle$. *Then the function* $f(x_1, \ldots, x_n)$ *can be degenerated (linearly decomposed) to an* $isig$*-function* $f(xM) = \widetilde{f}(y_1, \ldots, y_r) = f(x)$, $(M = [\lambda_1^T, \ldots, \lambda_r^T]$, $y_i = \lambda_i \cdot x$, $\lambda_i \in \mathbb{F}_2^n$, $\lceil \log_2 k \rceil \leq r \leq n)$ *with the minimal possible number of compound variables* r *if and only if* $\{\lambda_1, \ldots, \lambda_r\}$ *is a basis set of* $\langle \bigcup_{i=1}^{s} B_i \rangle$. *Moreover, the decomposition is optimum if* $dim \langle \bigcup_{i=1}^{s} B_i \rangle = \lceil \log_2 k \rceil$.

Proof Let $\{\lambda_1, \ldots, \lambda_r\}$ be an arbitrary basis set of $\langle \bigcup_{i=1}^{s} B_i \rangle$. By Theorem 3.2, every coordinate function ξ_i can be degenerated into the minimal possible number of compound variables $\#B_i$, where B_i is a basis set of $\langle S_{\xi_i} \rangle$. Consequently, the function f can be written in variables $\{y'_1, \ldots, y'_m\}$, where $y'_i = \lambda'_i \cdot x$ and $\{\lambda'_1, \ldots, \lambda'_m\} = \bigcup_{i=1}^{s} B_i \subseteq \langle \bigcup_{i=1}^{s} B_i \rangle$ $(r \leq m)$. Since every λ'_i can be written in terms of vectors $\lambda_1, \ldots, \lambda_r$, then f can be written in terms of compound variables $y_i = \lambda_i \cdot x$ $(i \in [1, r])$, which is clearly the minimal possible number of compound variables.

Conversely, let us assume that the minimal possible decomposition of f is given as $f(x) = \widetilde{f}(xM) = \widetilde{f}(y_1, \ldots, y_r)$, where $y_i = \lambda_i \cdot x$ $(i \in [1, r])$, for some vectors $\lambda_i \in \mathbb{F}_2^n$. Clearly, we assume that λ_i are linearly independent vectors, otherwise the existence of a basis set $\{\lambda'_1, \ldots, \lambda'_{r'}\}$ $(r' < r)$ of $\{\lambda_1, \ldots, \lambda_r\}$ would imply that f can be degenerated to $r' < r$, which contradicts the assumption. Then it must be the case that for every $i \in [1, s]$ a basis set B_i of $\langle S_{\xi_i} \rangle$ is contained in the space $\langle \lambda_1, \ldots, \lambda_r \rangle$ (where also $\langle S_{\xi_i} \rangle \subseteq \langle \lambda_1, \ldots, \lambda_r \rangle$), since all functions ξ_i have to be written in terms of compound variables y_i. Now, if $\langle \bigcup_{i=1}^{s} B_i \rangle$ is a proper subspace of $\langle \lambda_1, \ldots, \lambda_r \rangle$, then there exists a basis set $\{\lambda'_1, \ldots, \lambda'_{r'}\}$ of $\langle \bigcup_{i=1}^{s} B_i \rangle$ with the property that $r' < r$, which contradicts the assumption that f can be degenerated to r variables. Consequently, it necessarily holds that $\langle \lambda_1, \ldots, \lambda_r \rangle = \langle \bigcup_{i=1}^{s} B_i \rangle$, i.e., $\{\lambda_1, \ldots, \lambda_r\}$ is a basis set of $\langle \bigcup_{i=1}^{s} B_i \rangle$ (since λ_i are linearly independent). By [24, Theorem 1], the linear decomposition of f is optimum if $r = \lceil \log_2 k \rceil$. □

In general, Theorem 3.3 has the following important implications:

Fact 1 The problem of finding a linear decomposition of a given $isig$-function which uses compound variables $y_i = \lambda_i \cdot x$ $(i = 1, \ldots, r)$ such that $\lambda_1, \ldots, \lambda_r \in \mathbb{F}_2^n$ have minimal weights (in relation (5)), has been addressed in [29]. In the context of Theorem 3.3, such linear decompositions are precisely related to vectors with minimal weights which constitute a basis set $\{\lambda_1, \ldots, \lambda_r\}$ of $\langle \bigcup_{i=1}^{s} B_i \rangle$. Alternatively, the search of these decompositions for which $wt(\lambda_i) \leq t$ (for all $i \in [1, r]$) for some fixed value t, is mentioned as Problem 1 in [12, Section 3].

Fact 2 The set of all linear decompositions of an $isig$-function f is not less than the number of basis sets $\{\lambda_1, \ldots, \lambda_r\}$ of $\langle \bigcup_{i=1}^{s} B_i \rangle$. However, this estimation is still imprecise since the number of Boolean coordinate functions of f which satisfy the system (6) is not known.

By Theorem 3.3, the problem of finding a linear decomposition(s) (degeneration) of $isig$-functions is translated to the domain of Boolean functions, where we are looking for a suitable collection of functions ξ_i in (6) such that basis sets of $\langle S_{\xi_i} \rangle$ share common elements (which consequently reduces the number of compound variables y_i).

Therefore, the following example is given in that direction and it demonstrates the possibility of applying Theorem 3.3 to a given $isig$-function represented by (4) in order to find its

linear decomposition. Thus we illustrate the main idea of our approach, which is developed further in subsequent sections.

Example 3.2 Let the $isig$-function $f : D \subset \mathbb{F}_2^4 \to \{1, 2, 3, 4\}$ be represented as

$$f(x) = 1 \cdot b_1(x) + 2 \cdot b_2(x) + 3 \cdot b_3(x) + 4 \cdot b_4(x),$$

where the set D and coordinate Boolean functions b_i are defined as in Table 3. Notice that the restrictions of b_i on D are defined as $b_i|_D = m_{d_i}|_D$, since $m_{d_i}(x) = 1$ if and only if $x = d_i$. In order to apply Theorem 3.3, we need to define functions b_i on $\mathbb{F}_2^4 \backslash D$ due to the computation of dimensions of their Walsh supports. According to Theorem 3.2, we want to define functions b_i such that the basis sets of $\langle S_{b_i} \rangle$ share (many) common elements. For that purpose, we define the subspace $S = \{\mathbf{0}_4, d_1\}$ and $b_i = \phi_{E_i}$, where $E_i \subset \mathbb{F}_2^4$ are given as

$$E_1 = S, \quad E_2 = d_2 \oplus S, \quad E_3 = d_3 \oplus S, \quad E_4 = d_4 \oplus S.$$

The necessary condition to represent the function b_i correctly, in terms of characteristic functions, is that $E_i \cap D = \{d_i\}$, for all $i = 1, 2, 3, 4$, which is clearly satisfied. By relation (3), we have that for all $i \in [1, 4]$ it holds that

$$S_{b_i} = S_{\phi_{E_i}} = S^{\perp} = \langle \lambda_1, \lambda_2, \lambda_3 \rangle = \langle (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1) \rangle.$$

Consequently, Theorem 3.3 gives that f can be represented in variables $y_i = \lambda_i \cdot x$ ($i = 1, 2, 3$). More precisely, by ANF representation of the characteristic function given by (3) we have that b_i can be represented as $b_1(x) = \phi_{E_1}(x)$ and $b_j = \phi_{E_1}(x \oplus d_j)$ for $j = 2, 3, 4$, where

$$b_1(x) = \phi_{E_1}(x) = \prod_{i=1}^{3} (\lambda_i \cdot x \oplus 1) = \prod_{i=1}^{3} (y_i \oplus 1), \quad b_j(x) = \prod_{i=1}^{3} (y_i \oplus \lambda_i \cdot d_j \oplus 1).$$

Remark 3.4 Note that the same idea presented in the above example can be applied to f when it is represented as $f(x) = 1 + a_0(x) + 2a_1(x)$ (where a_i are defined as in Table 2).

The previous example, which regards the application of Theorem 3.3, raises the following important questions (which have not been addressed in the previous works):

I) Do there exist representations of f in terms of Boolean functions which allow an efficient construction (specification) of its coordinate Boolean functions on the whole space $\mathbb{F}_2^n$ which in turn provide an efficient determination of linear decompositions of f?

II) For a fixed representation of f, is it always possible to construct coordinate Boolean functions which provide an optimum (or minimal) linear decomposition of f (for an arbitrary fixed set $D \subset \mathbb{F}_2^n$, $\#D \ll 2^n$)?

Table 3 Definition of coordinate functions b_i on D

$d_i \in D$	b_1	b_2	b_3	b_4	$f(d_i)$
$d_1 = (0, 0, 0, 1)$	1	0	0	0	1
$d_2 = (1, 0, 1, 1)$	0	1	0	0	2
$d_3 = (1, 1, 0, 0)$	0	0	1	0	3
$d_4 = (0, 1, 1, 1)$	0	0	0	1	4

III) More generally, under which conditions with respect to an arbitrary set $D \subset \mathbb{F}_2^n$ ($\#D \ll 2^n$) there exists a representation of f which allows an optimal (or minimal) linear decomposition for some coordinate Boolean functions?

In the rest of the paper, we address the above questions by deriving and analyzing some general results which regard both f and D.

4 Construction of suitable coordinate functions

Although the questions (I)-(III) posed in Section 3 are hard to answer in general, in this section we provide some general/specific results which describe the linear decomposition of an $isig$-function f given as in Proposition 3.1. More precisely, in Section 4.1 we propose some methods to design suitable (affine) subspaces E_i, which in turn gives us a possibility to define ξ_i as $\xi_i = \phi_{E_i}$ that correctly represent f and allows its decomposition into less compound variables (by applying Theorem 3.3). While Section 4.1 mainly concerns the construction of suitable coordinate functions ξ_i, in Section 4.2 we discuss the cases for which a linear decomposition of a given $isig$-function is not possible with respect to a given set D in general.

4.1 Employing characteristic functions

Recall that in Example 3.2 we have defined coordinate functions of f as indicator functions on suitable (affine) subspaces E_i. In general, the selection of coordinate functions ξ_i in terms of characteristic functions ϕ_{E_i} regardless of whether E_i is a linear or affine subspace (compare to Example 3.2), is described as follows.

Proposition 4.1 *Let $f : D \subset \mathbb{F}_2^n \to \{1, \ldots, k\}$ be an arbitrary $isig$-function defined as $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$, and for $i \in [1, s]$ let $\Lambda_i = supp(\xi_i|_D) = \{d \in D : \xi_i|_D(d) = 1\}$. Then:*

i) *Assume that for $i \in [1, s]$ it holds that $\langle \Lambda_i \rangle \cap D = \Lambda_i$, and let $E_i \subset \mathbb{F}_2^n$ be an arbitrary (affine) subspace such that $\Lambda_i \subset E_i$ and $E_i \cap D = \Lambda_i$. Then, the assignment $\xi_i = \phi_{E_i}$ (for $i \in [1, s]$) gives a correct definition of f in terms of relation (6).*
ii) *Alternatively, assume that $\langle D \setminus \Lambda_i \rangle \cap D = D \setminus \Lambda_i$, and let $E_i \subset \mathbb{F}_2^n$ be an arbitrary subspace such that $D \setminus \Lambda_i \subset E_i$ and $E_i \cap D = D \setminus \Lambda_i$. Then $\xi_i = \phi_{E_i} \oplus 1$ (for $i \in [1, s]$) correctly defines f in terms of relation (6).*

Remark 4.1 By Theorem 3.3, the function f in Proposition 4.1 can be degenerated to less than n variables if $dim(\bigcup_{i=1}^{s} E_i) < n$.

Let us now assume that there exists (fixed) $i \in [1, s]$ for which it holds that $\xi_i|_D(d' \oplus d'') = 0$, for some $d', d'' \in D$ with $d', d'' \in \Lambda_i$. In this case, we are not able to define a subspace E_i such that $d', d'' \in E_i$, since $d' \oplus d'' \in E_i$ would imply that $\phi_{E_i}(d' \oplus d'') = 1 \neq \xi_i|_D(d' \oplus d'')$. The following result establishes the impossibility of setting $\xi_i = \phi_{E_i}$ ($\Lambda_i \subseteq E_i$) for arbitrary sets D with respect to a given representation $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$.

Proposition 4.2 *Let $f : D \to \{1, \ldots, k\}$ be an arbitrary $isig$-function defined as $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$ ($\xi_i \in \mathcal{B}_n$), and for $i \in [1, s]$ define sets $\Lambda_i = supp(\xi_i|_D) = \{d \in D : \xi_i|_D(d) = 1\}$. Assume that for a fixed (arbitrary) $i \in [1, s]$ there exist vectors $d_{i_1}, \ldots, d_{i_t} \in$*

Λ_i such that $d_{i_1} \oplus \ldots \oplus d_{i_t} \notin \Lambda_i$. Then there does not exist a subspace $E_i \subset \mathbb{F}_2^n$ such that $E_i \cap D = \Lambda_i$ and $d_{i_1} \oplus \ldots \oplus d_{i_t} \notin E_i$.

Proof Let us assume that there exists a subspace $E_i \subset \mathbb{F}_2^n$ such that $E_i \cap D = \Lambda_i$ and $d_{i_1} \oplus \ldots \oplus d_{i_t} \notin E_i$. Since $d_{i_1}, \ldots, d_{i_t} \in \Lambda_i$ and $\Lambda_i \subseteq E_i$, it implies that $d_{i_1}, \ldots, d_{i_t} \in E_i$, which is a contradiction. □

Combining Theorem 3.2 and Proposition 4.1 we obtain the following result which describes the decomposition method of an $isig$-function with coordinate functions ξ_i defined as indicator functions of suitable (affine) subspaces.

Theorem 4.1 *Let $f : D = \{d_1, \ldots, d_k\} \to \{1, \ldots, k\}$ ($D \subset \mathbb{F}_2^n$) be an arbitrary incompletely specified ig-function given as $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$, and for $i \in [1, s]$ define sets $\Lambda_i = supp(\xi_i|_D) = \{d \in D : \xi_i|_D(d) = 1\}$. Assume that $\langle \Lambda_i \rangle \cap D = \Lambda_i$, and let ξ_i be defined as $\xi_i = \phi_{E_i}$ for (affine) subspaces $E_i \subset \mathbb{F}_2^n$ which satisfy that $E_i \cap D = \Lambda_i$ ($\Lambda_i \subset E_i$). If B_i is a basis of $\langle S_{\xi_i} \rangle = E_i^{\perp}$ and $B = \{\lambda_1, \ldots, \lambda_r\}$ is a basis of $\langle \bigcup_{i=1}^{s} B_i \rangle$, then f can be written in compound variables $y_1, \ldots, y_r$, where $y_i = \lambda_i \cdot x$, $x \in \mathbb{F}_2^n$. The decomposition of f is optimum if $r = \lceil \log_2 k \rceil$.*

We know simplify the conditions in Proposition 4.1 (that is the idea presented in Example 3.2), thus making the application of Theorem 3.2 much more efficient. Let the set Δ_D be defined as

$$\Delta_D = \{d_i \oplus d_j : d_i, d_j \in D,\ i \neq j\}, \tag{7}$$

where we consider Δ_D not to be a multi-set by removing the vectors that appear multiple times. Notice that Δ_D (considered as a multi-set) was called a difference matrix (considering its vectors as rows) and used in [31, 34] for derivation of algorithms for linear decomposition.

Theorem 4.2 *Let $f : D = \{d_1, \ldots, d_k\} \to \{1, \ldots, k\}$ ($f(d_i) = i$, $D \subset \mathbb{F}_2^n$) be an arbitrary $isig$-function given as $f(x) = \sum_{i=1}^{s} c_i \xi_i(x)$. Assume that there exists a subspace $S \subset \mathbb{F}_2^n \setminus \Delta_D$ (Δ_D defined by (7)) with $dim(S) = t$ ($\leq n - \lceil \log_2 k \rceil$), and let $E_i = d_i \oplus S$ for $i \in [1, k]$. Then:*

i) *For every $d \in D$ it holds that $(d \oplus S) \cap D = \{d\}$.*
ii) *For $\xi_i(x) = \phi_{E_i}(x)$, $c_i = i$ ($s = k$, $i \in [1, k]$), it holds that f can be represented as $f(x) = \sum_{i=1}^{k} i \cdot \phi_{E_i}(x)$.*
ii) *The function f can be degenerated to $r = n - t \geq \lceil \log_2 k \rceil$ variables $y_1, \ldots, y_r$, where $y_i = \lambda_i \cdot x$ ($x \in \mathbb{F}_2^n$), and $\{\lambda_1, \ldots, \lambda_r\}$ is a basis of $S^{\perp}$. The function f is given as*

$$f(x) = \sum_{i=1}^{k} i \cdot \prod_{j=1}^{r} (y_j \oplus \lambda_j \cdot d_i \oplus 1).$$

iv) *Consequently, if $wt(\lambda_i) = 1$, then y_i are essential variables of f.*

Proof i) Let $d \in D$ be an arbitrary vector. Since $\mathbf{0}_n \in S$, it holds that $d \in d \oplus S$. If we assume that there exists $\widetilde{d} \in D$ such that $\widetilde{d} \neq d$ and $\widetilde{d} \in (d \oplus S) \cap D$, then for some vector $z \in S$ it holds that $d \oplus z = \widetilde{d}$, which is equivalent to $z = d \oplus \widetilde{d}$. Consequently, $z \in \Delta_D$, which is a contradiction (since $S \subset \mathbb{F}_2^n \setminus \Delta_D$).

ii) By Proposition 3.1, the function f can be represented as $f(x) = \sum_{i=1}^{k} i \cdot \phi_{E_i}(x)$ if and only if $\xi_i|_D = \phi_{E_i}|_D = m_{d_i}|_D$, which is satisfied by the previous part due to the fact that $(d_i \oplus S) \cap D = E_i \cap D = \{d_i\}$, $i \in [1, k]$.

iii) By relation (3) we have that for all $i \in [1, k]$ it holds that $S_{\xi_i} = S^{\perp}$, which by Theorem 3.1 means that all ξ_i can be degenerated to r compound variables $y_1, \ldots, y_r$, where $y_i = \lambda_i \cdot x$ ($x \in \mathbb{F}_2^n$) with $\{\lambda_1, \ldots, \lambda_r\}$ being a basis of $S^{\perp}$. Consequently, the ANF of $\xi_i = \phi_{E_i}$ in terms of variables y_i is given as $\phi_{E_i}(x) = \prod_{j=1}^{r}(y_j \oplus \lambda_j \cdot d_i \oplus 1)$.

iv) Using the previous part, [38, Theorem 2.1] and $wt(\lambda_i) = 1$ imply that f can be written in terms of compound variables y_i, i.e., y_i are essential variables of f.

□

Remark 4.2 Theorem 4.2 is especially efficient for sets D of sufficiently small dimension (i.e., the number of linearly independent vectors), since the corresponding set Δ_D will contain less vectors. Clearly, in the case when $r = n - t = \lceil \log_2 k \rceil$ ($\#D = k$), then Theorem 4.2 explicitly provides (and thus guarantees) an optimal decomposition of f.

To illustrate Theorem 4.2, we provide the following example.

Example 4.1 Consider an incompletely specified ig-function $f : D = \{d_1, \ldots, d_6\} \subset \mathbb{F}_2^4 \to \{1, \ldots, 6\}$ ($f(d_i) = i$, for $i \in [1, 6]$), where the set D is given as

$$D = \{(0, 0, 1, 0), (0, 1, 0, 1), (0, 1, 1, 1), (1, 0, 1, 0), (1, 0, 0, 0), (1, 1, 0, 1)\}.$$

We have that $dim\langle D\rangle = 3$ and $\Delta_D = D \cup \{(1, 1, 1, 1)\}$ ($\Delta_D \subset \langle D\rangle$), and let us define the one-dimensional space $S = \{\mathbf{0}_4, (0, 0, 1, 1)\} \subset \mathbb{F}_2^4 \setminus \Delta_D$. Defining affine subspaces E_i as $E_i = d_i \oplus S$, we have that $E_i \cap D = \{d_i\}$, for all $i \in [1, 6]$. Consequently, by Theorem 4.2 we may define f as $f(x) = \sum_{d_i \in D} i \cdot \phi_{E_i}(x)$ (which means that $\xi_i = \phi_{E_i}$), and all functions ϕ_{E_i} can be degenerated to $dim(S^{\perp}) = 3$ compound variables $y_i = \lambda_i \cdot x$, $i = 1, 2, 3$ (due to relation (3)), where λ_i constitute the basis of $S^{\perp} = \langle \lambda_1, \lambda_2, \lambda_3\rangle = \langle (0, 0, 1, 1), (0, 1, 0, 0), (1, 0, 0, 0)\rangle$.

4.2 On non-existence of linear decompositions

The main question which arises in Theorem 4.2, which is further investigated in rest of the work, is whether there always exists a subspace $S \subset \mathbb{F}_2^n \setminus D$ (for arbitrary given set D) for which Theorem 4.2-(i) holds. The answer is (partially) given by the following result which describes all sets D, i.e., $isig$-functions defined on D, which can not be linearly decomposed in general.

Theorem 4.3 *Let $f : D \to \{1, \ldots, k\}$ ($D \subset \mathbb{F}_2^n$, $\#D = k$) be an arbitrary $isig$-function. If for every $z \in \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$ it holds that $(z \oplus D) \cap D \neq \emptyset$, then there does not exist a linear vectorial mapping $h(x) = (\lambda_1 \cdot x, \ldots, \lambda_p \cdot x)$ ($x \in \mathbb{F}_2^n$) with $p \leq n - 1$ which is injective on D. Equivalently, it holds that $\Delta_D = \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$.*

Proof Suppose that $h(x) = (\lambda_1 \cdot x, \ldots, \lambda_p \cdot x)$ is an injective mapping on $D \subset \mathbb{F}_2^n$, such that $p \leq n - 1$. Let us consider a vector $z \neq \mathbf{0}_n$ which is orthogonal on all vectors λ_i (for $i \in [1, p]$), i.e., consider a non-zero vector $z \in \langle \lambda_1, \ldots, \lambda_p\rangle^{\perp}$. Clearly, such a vector exists since $dim\langle \lambda_1, \ldots, \lambda_p\rangle \leq n - 1$. Since for every $z \in \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$ and $d \in D$ it holds that

$z \oplus d \in D$, then there exists $d' \in D$ ($d' \neq d$) such that $z = d \oplus d'$, i.e., $z \in \Delta_D$ (where Δ_D is defined by (7)). Recall from [34, Section II] that h is injective on D if and only if $h(v) \neq \mathbf{0}_p$, for every $v \in \Delta_D$. However, we have that $z \in \Delta_D$ and $h(z) = \mathbf{0}_p$ since $z \perp \lambda_i$ for all $i \in [1, p]$, which means that h is not injective.

In order to prove the equality $\Delta_D = \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$, it is clear from the previous part (due to the given assumption) that for every $u \in \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$ it holds that $u = d' \oplus d''$, for some $d' \oplus d'' \in \Delta_D$, which means that $\mathbb{F}_2^n \setminus \{\mathbf{0}_n\} \subseteq D_f \subset \mathbb{F}_2^n$. However, by definition (7) we have that $\mathbf{0}_n \notin \Delta_D$, which proves the statement. □

Remark 4.3 In the context of Theorem 4.2-(i), the inequality $\Delta_D \neq \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$ means that we can find at least 1-dimensional subspace $S = \{\mathbf{0}_n, \widetilde{d}\}$ ($\widetilde{d} \notin \Delta_D$, $\widetilde{d} \neq \mathbf{0}_n$) for which $(d \oplus S) \cap D = \{d\}$ (for all $d \in D$) holds.

It is important to note that Theorem 4.3 provides an alternative solution for establishing the minimality of the number of compound variables of a given degenerated incompletely specified $isig$-function $f(x) = \widetilde{f}(y_1, \ldots, y_r)$. A description of the SAT-based method, which also reduces the number of compound variables and establishes the minimality of solution, by verifying the fact that any further reduction of a given solution is impossible, is given in [28] (see also [25, Section VII]). In fact, there is no guarantee that either Theorem 4.3 or SAT-based method achieve the optimality of solution and these results can be rather used in any algorithm for verifying whether the obtained number of compound variables is minimal or not, cf. Step 3 in Example 5.1.

As noted by T. Sasao in [25], the SAT-based method is both time and memory consuming and thus only applicable to sets D of relatively small size. Regarding the analysis of the minimality of the solution, Theorem 4.3 (in comparison to SAT-based method) can be used even for relatively large sets D and thus can be used in any iterative algorithm which provides a linear decomposition (for instance, it is used in Algorithm 1 in Section 5.1). The only condition that need to be checked in Theorem 4.3 is that $\Delta_D = \mathbb{F}_2^n \setminus \{\mathbf{0}_n\}$ and then D can not be further decomposed.

5 An iterative method for linear decomposition

In this section we derive a new iterative algorithm for linear decomposition of a given $isig$-function, which is essentially based on Theorem 4.2. While the basic mode of the new algorithm is given in Section 5.1, in Section 5.2 we provide additional analyses along with comparison to other known methods.

5.1 A basic mode of the new algorithm

The main idea behind the recursive employment of Theorem 4.2 is illustrated with the following example, where we firstly recall the following result which is known as Theorem 7.1 in [25].

Theorem 5.1 [25] *Let $f : D \to \{1, \ldots, k\}$ ($D \subset \mathbb{F}_2^n$, $\#D = k$) be an arbitrary $isig$-function. Then f can be degenerated to variables $y_1, \ldots, y_r$, that is $f(x) = \widetilde{f}(y_1, \ldots, y_r)$, where $y_i = \lambda_i \cdot x$ for some vectors $\lambda_i \in \mathbb{F}_2^n$, if and only if $h(x) = (y_1, \ldots, y_r) = (\lambda_1 \cdot x, \ldots, \lambda_r \cdot x)$ is an injective mapping on D.*

Example 5.1 Let $f : D = \{d_1, \ldots, d_7\} \to \{1, \ldots, 7\}$ ($D \subset \mathbb{F}_2^6$) be an incompletely specified ig-function, where the set D is given as

$$D = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{pmatrix} = \begin{pmatrix} 0\ 0\ 0\ 0\ 1\ 0 \\ 0\ 1\ 0\ 0\ 1\ 0 \\ 0\ 0\ 1\ 0\ 1\ 0 \\ 0\ 0\ 1\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1 \\ 1\ 1\ 1\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1\ 1\ 1 \end{pmatrix}.$$

Step 1: Let us consider the vector $\sigma_1 = (0, 0, 0, 0, 0, 1) \in \mathbb{F}_2^6 \setminus \Delta_D$ and define $S_1 = \{\mathbf{0}_6, \sigma_1\}$. Now, if we define the characteristic functions ϕ_{E_i} ($i \in [1, 7]$), where $E_i = d_i \oplus S_1$, then by Theorem 4.2-(i) it holds that $E_i \cap D = \{d_i\}$. Consequently, the function f can be represented as $f(x) = \sum_{i=1}^{7} i \cdot \phi_{E_i}(x)$. Furthermore, Theorem 4.2-(iii) implies that ϕ_{E_i} can be degenerated to $dim(S_1^{\perp}) = 5$ compound variables $y_i^{(1)} = \lambda_i^{(1)} \cdot x$ ($i \in [1, 5]$), where $\lambda_i^{(1)}$ are basis vectors of $S_1^{\perp} = \{\mathbf{0}_6, \sigma_1\}^{\perp}$, which we shortly denote as $S_1^{\perp} = \sigma_1^{\perp}$. Considering the canonical basis $\{\lambda_1^{(1)}, \ldots, \lambda_5^{(1)}\} = \{e_1^{(1)}, \ldots, e_5^{(1)}\}$ ($e_i^{(1)}$ has a non-zero coordinate at the i-th position) of $S_1^{\perp} \subset \mathbb{F}_2^6$, we have that $y_i^{(1)} = x_i$ for $i \in [1, 5]$. Now, due to Theorem 5.1, we can construct a vectorial linear injective function $h_1 : D \subset \mathbb{F}_2^6 \to D^{(1)} \subset \mathbb{F}_2^5$ as

$$h_1(x) = \left(y_5^{(1)}, \ldots, y_1^{(1)}\right) = (x_5, \ldots, x_1) = xM_1, \quad x \in \mathbb{F}_2^6,$$

where the matrix M_1 is given as $M_1 = (\lambda_5^{(1)}, \ldots, \lambda_1^{(1)})_{6\times 5}$, and $\lambda_j^{(1)}$ ($j \in [1, 5]$) are written as column vectors.

Step 2: Using the same reasoning as in **Step 1**, we proceed further with transforming $D^{(1)}$ to some set $D^{(2)} \subset \mathbb{F}_2^4$. Hence, taking $\sigma_2 = (0, 0, 0, 0, 1) \in \mathbb{F}_2^5 \setminus \Delta_{D^{(1)}}$ ($\Delta_{D^{(1)}}$ defined by (7) with respect to $D^{(1)}$), we define $S_2 = \{\mathbf{0}_5, \sigma_2\}$, and consequently for the second transformation we may take the set $\{\lambda_1^{(2)}, \ldots, \lambda_4^{(2)}\} = \{e_1^{(2)}, e_2^{(2)}, e_3^{(2)}, e_4^{(2)}\}$ which is the canonical basis of $S_2^{\perp} = \sigma_2^{\perp} \subset \mathbb{F}_2^5$. Thus, the second transformation of $D^{(1)}$ is the mapping $h_2 : D^{(1)} \to D^{(2)} \subset \mathbb{F}_2^4$ defined as

$$h_2(x) = \left(y_4^{(2)}, \ldots, y_1^{(2)}\right) = (x_4, \ldots, x_1) = xM_2, \quad x \in \mathbb{F}_2^5,$$

where the matrix M_2 is given as $M_2 = \left(\lambda_4^{(2)}, \ldots, \lambda_1^{(2)}\right)_{5\times 4}$.

Step 3: Let $\sigma_3 = (0, 0, 1, 1) \in \mathbb{F}_2^4 \setminus \Delta_{D^{(2)}}$, and define $S_3 = \{\mathbf{0}_4, \sigma_3\}$. For the third transformation we may take the basis set $\left\{\lambda_1^{(3)}, \lambda_2^{(3)}, \lambda_3^{(3)}\right\} = \{(0, 0, 1, 1), (0, 1, 0, 0), (1, 0, 0, 0)\}$ of $S_3^{\perp} = \sigma_3^{\perp} \subset \mathbb{F}_2^4$. Hence, for $y_i^{(3)} = \lambda_i^{(3)}$ ($i = 1, 2, 3$), the linear mapping $h_3 : D^{(2)} \to D^{(3)} \subseteq \mathbb{F}_2^3$ we may define as

$$h_3(x) = \left(y_1^{(3)}, y_2^{(3)}, y_3^{(3)}\right) = (x_3 \oplus x_4, x_2, x_1) = xM_3, \quad x \in \mathbb{F}_2^4,$$

where $M_3 = (\lambda_3^{(3)}, \lambda_2^{(3)}, \lambda_1^{(3)})_{4\times 3}$.

At this point, noticing that $\Delta_{D^{(3)}} = D^{(3)} = \mathbb{F}_2^3 \setminus \{\mathbf{0}_3\}$ implies, by Theorem 4.3, that $D^{(3)}$ can not be further transformed by any injective linear mapping with less than 3 coordinate

functions, i.e., f can not be degenerated to less than 3 variables. These three transformations of the space D are given by

$$D \xrightarrow{h_1} D^{(1)} = \begin{pmatrix} 1&0&0&0&0\\ 1&0&0&1&0\\ 1&0&1&0&0\\ 1&1&1&0&0\\ 0&0&0&0&0\\ 1&0&1&1&1\\ 1&1&0&1&0 \end{pmatrix} \xrightarrow{h_2} D^{(2)} = \begin{pmatrix} 0&0&0&1\\ 1&0&0&1\\ 0&1&0&1\\ 0&1&1&1\\ 0&0&0&0\\ 1&1&0&1\\ 1&0&1&1 \end{pmatrix} \xrightarrow{h_3} D^{(3)} = \begin{pmatrix} 1&0&0\\ 1&0&1\\ 1&1&0\\ 0&1&0\\ 0&0&0\\ 1&1&1\\ 0&0&1 \end{pmatrix}.$$

Finally, the linear transformation, say h, which maps D to $D^{(3)} = \mathbb{F}_2^3$ is given as a composition $h(x) = (h_3 \circ h_2 \circ h_1)(x)$, i.e., $h(x) = (y_1, y_2, y_3) = xM_1M_2M_3 = (x_4 \oplus x_5, x_3, x_2)$ $(x \in \mathbb{F}_2^6)$, and h is an optimal decomposition of D. Also, the function f can be easily written in terms of compound variables y_1, y_2, y_3 from $D^{(3)}$ as $f(x) = \sum_{i=1}^{7} i \cdot m_{h(d_i)}(y_1, y_2, y_3)$.

Before we formally provide an algorithm which is based on the previously presented idea, we firstly discuss the underlying ideas behind its main steps, as illustrated in Example 5.1. Namely, Theorem 4.2 and Theorem 4.3 ensure that the sequence of injective transformations reduces the number of compound variables by 1 in each step. Clearly, at each step one can proceed further if and only if there exists a one-dimensional space S_i for which the condition from Theorem 4.2-(i) is satisfied.

However, with the following example we show that not all choices of (non-zero) vectors σ_i in each step lead to the same number of compound variables at the end (and thus an optimal decomposition is not always guaranteed).

Example 5.2 Consider the set D given as in Example 5.1 and let the first two vectors of transformations σ_1, σ_2 be given as $\sigma_1 = (1, 1, 0, 1, 1, 1) \in \mathbb{F}_2^6 \backslash \Delta_D$, $\sigma_2 = (1, 0, 1, 0, 1) \in \mathbb{F}_2^5 \backslash \Delta_{D^{(1)}}$. Let the corresponding basis sets $\{\lambda_1^{(1)}, \ldots, \lambda_5^{(1)}\}$ of $\sigma_1^{\perp}$ and $\{\lambda_1^{(2)}, \ldots, \lambda_4^{(2)}\}$ of $\sigma_2^{\perp}$ be given as

$$\begin{pmatrix} \lambda_1^{(1)}\\ \lambda_2^{(1)}\\ \lambda_3^{(1)}\\ \lambda_4^{(1)}\\ \lambda_5^{(1)} \end{pmatrix} = \begin{pmatrix} 0&0&1&0&0&0\\ 1&0&0&0&0&1\\ 1&0&0&0&1&0\\ 1&0&0&1&0&0\\ 1&1&0&0&0&0 \end{pmatrix}, \quad \begin{pmatrix} \lambda_1^{(2)}\\ \lambda_2^{(2)}\\ \lambda_3^{(2)}\\ \lambda_4^{(2)} \end{pmatrix} = \begin{pmatrix} 0&0&0&1&0\\ 0&1&0&0&0\\ 1&0&0&0&1\\ 1&0&1&0&0 \end{pmatrix}.$$

Now, if we define $h_1(x) = (\lambda_1^{(1)} \cdot x, \ldots, \lambda_5^{(1)} \cdot x)$ $(x \in \mathbb{F}_2^6)$ and $h_2(x) = (\lambda_1^{(2)} \cdot x, \ldots, \lambda_4^{(2)} \cdot x)$ $(x \in \mathbb{F}_2^5)$, then the linear mapping $h_2 \circ h_1$ injectively transforms D to $D^{(2)} = \{(0, 0, 0, 1), (0, 0, 1, 1), (0, 0, 1, 0), (1, 0, 1, 0), (0, 1, 0, 0), (1, 0, 1, 1), (1, 1, 1, 1)\}$ so that $\Delta_{D^{(2)}} = \mathbb{F}_2^4 \setminus \{\mathbf{0}_4\}$, which means that $D^{(2)}$ can not be further transformed (Theorem 4.3).

Remark 5.1 It is worth mentioning that for an arbitrary set D there does not always exist an optimal decomposition (see for instance [25, Example 7.2]), and therefore the optimality/minimality of a decomposition (in terms of definitions given in Section 2) is not simple to prove in general.

The iterative method presented in Example 5.1 (which is based on Theorem 4.2) is formally described as follows.

Algorithm 1

Input: Set $D = \{d_1, \ldots, d_k\} \subset \mathbb{F}_2^n$.
Output: Linear vectorial injective mapping $h(x) = (\lambda_1 \cdot x, \ldots, \lambda_r \cdot x)$ $(\lambda_j \in \mathbb{F}_2^n)$, where $h : D \to D^{(n-r)} \subset \mathbb{F}_2^r$ for some parameter r such that $\lceil \log_2 k \rceil \leq r \leq n$.

Set $D = D^{(0)}$, the identity matrix $M_0 = I_{n\times n}$, and for $i = 1, 2, \ldots, n - r + 1$ $(\lceil \log_2 k \rceil \leq r \leq n)$, do the following:
Step 1: Search for a first non-zero vector σ_i in lexicographically ordered space $\mathbb{F}_2^{n-(i-1)}$, starting with $\sigma_i = (0, \ldots, 0, 1)$, such that $(\sigma_i \oplus D^{(i-1)}) \cap D^{(i-1)} = \emptyset$.
Step 2: Choose a basis $\{\lambda_1^{(i)}, \ldots, \lambda_{n-i}^{(i)}\}$ of $\sigma_i^{\perp} \subset \mathbb{F}_2^{n-(i-1)}$ and define the matrix $M_i = [\lambda_1^{(i)}, \ldots, \lambda_{n-i}^{(i)}]_{(n-i+1)\times(n-i)}$, where $\lambda_j^{(i)}$ are written as column vectors.
Step 3: Update and save $M_i \longrightarrow M_{i-1}M_i$ and $D^{(i)} \longrightarrow D^{(i-1)}M_i = \{dM_i : d \in D^{(i-1)}\}$ (replace M_{i-1} and $D^{(i-1)}$ with M_i and $D^{(i)}$, respectively).

Step 4 Repeat **Steps 1-3** until for some $i = n - r + 1$ it holds that there does not exist σ_{n-r+1} such that $(\sigma_{n-r+1} \oplus D^{(n-r)}) \cap D^{(n-r)} = \emptyset$.

Step 5 Compute and return the function $h(x) = xM_{n-r}$.

Remark 5.2 Note that in **Step 1** if for some σ_i it holds that $(\sigma_i \oplus D^{(i-1)}) \cap D^{(i-1)} = \emptyset$, then $\sigma_i \notin \Delta_{D^{(i-1)}}$, i.e., $D^{(i-1)}$ can be transformed further since $D^{(i-1)} \neq \mathbb{F}_2^{n-(i-1)} \setminus \{\mathbf{0}_{n-(i-1)}\}$ (Theorem 4.3). Thus, by Theorems 4.2 and 4.3, we have that Algorithm 1 provides a sequence of linear injective transformations of D, given as $h_i(x) = x\widehat{M}_i$, where $\widehat{M}_i = [\lambda_1^{(i)}, \ldots, \lambda_{n-i}^{(i)}]_{(n-i+1)\times(n-i)}$ and the output injective mapping $h : D \to D^{(n-r)} \subset \mathbb{F}_2^r$ is given as $h(x) = (h_{n-r} \circ \cdots \circ h_1)(x) = x\widehat{M}_1\widehat{M}_2 \cdots \widehat{M}_{n-r} = xM_{n-r}$ $(x \in \mathbb{F}_2^n)$, where $M_{n-r} = [\lambda_1, \ldots, \lambda_r]$ is of the size $n \times r$.

Remark 5.3 If we in **Step 1** add the condition that $wt(\sigma_i) = 1$, then consequently $\sigma_i^{\perp}$ always contains a basis whose vectors are of weight 1. Thus, Algorithm 1 may provide compound variables of degree equal to 1, which are actually the essential variables of f (Theorem 4.2-(iv)).

Remark 5.4 In general, Algorithm 1 can produce a transformation of D which reduces the number of compound variables more than 1 per step. This is achieved if in **Step 1** instead of σ_i we consider a subspace $S_i \subset \mathbb{F}_2^{n-(i-1)} \setminus \Delta_{D^{(i-1)}}$ with $dim(S_i) \geq 2$ (due to Theorem 4.2). Clearly, in such a case we are reducing the number of compound variables by $dim(S_i^{\perp})$. However, the problem which we encounter in this case is an efficient search for a subspace S_i outside of $\Delta_{D^{(i-1)}}$.

An efficient method of selecting a basis of a given vector $\sigma \in \mathbb{F}_2^k$ (which is needed in **Step 2**) is given in the following simple result.

Proposition 5.1 *Let $\sigma = (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}_2^k \setminus \{\mathbf{0}_k\}$ be an arbitrary vector, and let $e_1, \ldots, e_k \in \mathbb{F}_2^k$ be the canonical basis of $\mathbb{F}_2^k$. Suppose that $p \in [1, k]$ is the minimal integer*

for which $\alpha_p = 1$ (i.e., $\alpha_j = 0$ for $j < p$), and denote by $\widehat{e}_j$ the vectors obtained by modifying e_i as follows:

$$\widehat{e}_j = \begin{cases} e_j, & \alpha_j = 0 \\ e_p \oplus e_j, & \alpha_j = 1 \end{cases}, \quad j \in \{1, \ldots, k\} \setminus \{p\}.$$

Then the set $\{\widehat{e}_j : j \in \{1, \ldots, k\} \setminus \{p\}\}$ is basis of $\sigma^{\perp} = \{z \in \mathbb{F}_2^k : \sigma \cdot z = 0\}$.

Proof Let $S = \{1, \ldots, k\} \setminus \{p\}$ and define $A = \{j \in S : \alpha_j = 0\}$ and $B = \{t \in S : \alpha_t = 1\}$. Then, clearly for all indices $j \in A$ it holds that $\sigma \cdot \widehat{e}_j = \sigma \cdot e_j = \alpha_j \cdot 1 = 0$, since $\alpha_j = 0$. Also, for indices $t \in B = S \setminus A$ (which correspond to $\alpha_t = 1$) we have that $\sigma \cdot \widehat{e}_t = \sigma \cdot (e_p \oplus e_t) = \alpha_p \cdot 1 \oplus \alpha_t \cdot 1 = 0$, since $\alpha_t = 1$ and $\alpha_p = 1$. Consequently, all $\widehat{e}_j$ are orthogonal to σ. Moreover, due to the definition of $\widehat{e}_j$ ($j \in \{1, \ldots, k\} \setminus \{p\}$) it is not difficult to see that they are linearly independent, which completes the proof. □

An example which illustrates Proposition 5.1 is given in Appendix. Note that the efficiency of Proposition 5.1 lies in the fact that it keeps the weights of basis vectors at minimum, and later our experimental results will show that Algorithm 1 in combination with Proposition 5.1 produces linear decompositions with very low weights (thus suitable for efficient hardware implementation). Now, the representation of an incompletely specified ig-function f decomposed by Algorithm 1 is given as follows.

Theorem 5.2 *Let $f : D = \{d_1, \ldots, d_k\} \to \{1, \ldots, k\}$ ($D \subset \mathbb{F}_2^n$) be an arbitrary $isig$-function. Assuming that a linear injective transformation $h : D = \{d_1, \ldots, d_k\} \to D^{(n-r)} \subset \mathbb{F}_2^r$ is obtained by **Algorithm 1** (or any other algorithm), where $h(x) = (y_1, \ldots, y_r) = (\lambda_1 \cdot x, \ldots, \lambda_r \cdot x)$ ($\lambda_i \in \mathbb{F}_2^n$), then the function f is given as*

$$f(x) = \widetilde{f}(y_1, \ldots, y_r) = \sum_{i=1}^{k} i \cdot m_{h(d_i)}(y_1, \ldots, y_r).$$

It is important to notice that Algorithm 1 actually efficiently uses the construction results given in Section 4 (related to the design of Boolean coordinate functions) in an iterative manner in order to obtain a decomposition of a given $isig$-function. Using the same idea, one may derive various new algorithms (or modifications of Algorithm 1) by considering other approaches that provide the degeneration of f.

5.2 Analysis of Algorithm 1 and comparisons

In this section we analyze the running time complexity of Algorithm 1 and give a performance comparison to other methods.

5.2.1 Complexity analysis

The most time-consuming part in Algorithm 1 regards the decision which σ_i does not belong to $\Delta_{D^{(i-1)}}$ (Remark 5.2), due to (plausibly) large cardinality of $\#\Delta_{D^{(i-1)}}$. The problem of finding such σ_i is included in the well-known $(t, k, v) - TargetSum_q$ ($q = 2$) problem for vector spaces, which has been considered in [3] (see also [33, Section 3.1]).

The idea of checking whether $(\sigma_i \oplus D^{(i-1)}) \cap D^{(i-1)} = \emptyset$ rather than generating $\Delta_{D^{(i-1)}}$ utilize the property that $\#D \ll 2^n$ and is similar to the approach taken in [3, Section 4].

Table 4 Complexities of existing methods

Methods	[19]	[24]	[31]	[17]	[12]	[22]	[1]
Time	$O(n^5k)$	$O(n^tk\log(k))$	NA (generates Δ_D)	$O(n^sk\log(k))$	$O(ntk\log(k))$	Too long	$O\left(\binom{n}{s^*}^2 n2^m\right)$
Space	$O(nk)$	$O(n^tk)$	$O(nk^2)$	$O(nk)$	$O(nk)$	$O(k^22^n)$	NA

This implies that the time complexity of deciding whether σ_i outside of $\Delta_{D^{(i-1)}}$ (for fixed $i \geq 1$) in **Step 1** is at most $k \cdot O(n-i+1)$ to compute $\sigma_i \oplus d$, for $d \in \Delta_{D^{(i-1)}}$. A loose upper bound on this complexity is kn since the verification that $(\sigma_i \oplus D^{(i-1)}) \cap D^{(i-1)} = \emptyset$ may be terminated before computing $\sigma_i \oplus d$ for all $d \in \Delta_{D^{(i-1)}}$ and the length of the vectors σ_i and d is in general $n-i-1 < n$. In average, also confirmed by computer simulations, one needs to check approximately less than k many vectors σ_i until a suitable σ_i is found which satisfies $(\sigma_i \oplus D^{(i-1)}) \cap D^{(i-1)} = \emptyset$. Thus, an upper bound on finding σ_i is therefore $O(k^2n)$.

In addition, using that the complexity of computing a product of two quadratic matrices of size $n \times n$ is $O(n^{2.34})$ and taking into account that our matrices $n \times (n-i+1)$ and $(n-i+1)\times(n-i)$ a good estimate for this operation is $O(n^2)$ since the sizes are smaller than $n \times n$. This implies that the complexity in each step (upper bound) is $O(k^2n+n^2)$. Notice that, for a standard size of parameters k and n, the complexity of computing the orthogonal basis using the method given in Proposition 5.1 is neglected. Thus, the time complexity of Algorithm 1 is then $(n-r+1)O(k^2n+n^2)$ taking into account that there are in total $n-r+1$ steps. The memory complexity refers to storing the matrix M_i and the set $D^{(i)} \subset \mathbb{F}_2^{n-i}$ so that it can be upper bounded by $O(n^2+kn)$. The above complexity estimates are very conservative but they anyway provide a rough upper bound for the purpose of a performance comparison to other algorithms. In most cases $n \ll k$ and therefore the time and memory complexity are $(n-r+1)O(k^2n)$ and $O(kn)$, respectively.

Remark 5.5 Note that further complexity reduction of **Step 1** can be achieved by implementing the algorithm described in [3, Section 4], which is based on Fast Fourier Transform and convolution (for more details see [3, Facts 6-7]). In that case, the time of finding σ_i in **Step 1** can be solved in deterministic $2^{O(2)} \cdot k \cdot O(n)$ time (cf. [3, Theorem 13]).

In Tables 4 and 5 we compare the running complexities between various methods which provide either compound variables or essential variables in the decomposition process. Here, n is the number of variables, k is the number of indices ($\#D = k$), t is the maximum compound degree, and s is the number of initial variables taken out of n variables to initialize the method. Note that [1] is tested on s^*-out-of-n binary code with m variables.

Table 5 Complexity of our method with respect to different modes

Methods	Ours (basic mode)	Ours ([3, Theorem 13] mode)
Time	$(n-r+1)O(k^2n)$	$2^{O(2)}(n-r+1)O(kn)$
Space	$O(kn)$	$O(kn)$

Table 6 The performance of Algorithm 1 on arbitrary sets $D \subset \mathbb{F}_2^n$: $\lceil \log_2 k \rceil$ is the optimal number of compound variables, $L = 2\lceil \log_2(k+1) \rceil - 4$

n	$\#D = k$	Number of random sets D	Average value of $\#\Delta_D$	Average number of compound variables derived by Algorithm 1 (in brackets are given lower and upper bounds over all D)	Maximal compund degree over all D	$\lceil \log_2 k \rceil$	L
5	5	50	$2^{3.2}$	3	Mostly 1 (Over all ≤ 2)	3	2
	10	50	$2^{4.7}$	4	≤ 2	4	4
	10	50	$2^{5.3}$	4	Mostly $1-2$ (Over all ≤ 3)	4	4
7	20	50	$2^{6.7}$	5.5 (5-6)	Mostly $1-2$ (Over all ≤ 3)	5	6
	30	50	$2^{6.9}$	6.2 (6-7)	≤ 2	5	6
	20	50	$2^{7.3}$	5.5 (5-6)	≤ 3	5	6
9	35	50	$2^{8.5}$	6.8 (6-7)	Mostly $1-2$ (Over all ≤ 4)	6	8
	50	50	$2^{8.9}$	7.4 (7-8)	≤ 3	6	8
	20	50	$2^{7.5}$	5.5 (5-6)	≤ 3	5	6
	50	50	$2^{9.8}$	7.6 (7-8)	≤ 3	6	8
11	80	30	$2^{10.6}$	8.9 (8-9)	Mostly $1-2$ (Over all ≤ 3)	7	10
	100	20	$2^{10.8}$	9	Mostly $1-2$ (Over all ≤ 3)	7	10
	200	20	$2^{10.9}$	11	–	8	12
	100	10	$2^{11.8}$	9	≤ 4	7	10
13	300	10	$2^{12.9}$	12	≤ 2	9	14
	500	5	$2^{12.9}$	13	–	9	14
	700	5	$2^{12.9}$	13	–	10	16
	100	10	$2^{12.3}$	9	≤ 4	7	10
17	200	10	$2^{14.2}$	11	Mostly $1-2$ (Over all ≤ 3)	8	12
	300	5	$2^{15.2}$	12	≤ 3	9	14
25	50	50	–	7.6 (7-8)	≤ 3	6	8
	100	50	–	9	≤ 3	7	10

5.2.2 Comparisons to other methods

In Tables 6 and 7 we show the performance of Algorithm 1[2] for randomly selected sets D, where in **Step 1** of the algorithm a basis of $\sigma_i^{\perp}$ is selected as described in Proposition 5.1 (which appears to be quite efficient approach). Also, the performance of Algorithm 1 in providing essential (and compound) variables, along with comparison to other methods, is given by Tables 7 and 8.

Note that Tables 6, 7 and 8 illustrate the performance of our method on randomly selected sets D exclusively. We find certain methods that justify the algorithm performance on m-out-of-n sets (codes) D, as slightly misleading since such sets possess a structure which

[2]Algorithm 1 was implemented in Wolfram Mathematica 9, on the machine with the following characteristics: Lenovo ThinkPad E540, OS Windows 7 (64-bit), Intel Core i5-4200M-2.5GHz, RAM 4Gb.

Table 7 Comparison of Algorithm 1 with methods [1, 21] on arbitrary sets $D \subset \mathbb{F}_2^n$: Results for [1, 21] are recalled from [25, Table 8.1], where t is the **compound degree**, Ceiling of M_{50} is the lower bound for $t = 1$ given by [32, Property 4.1], $L = 2\lceil \log_2(k+1) \rceil - 4$

n	$\#D = k$	[21] ($t = 1$ for all D)	[1] (Average t for all D)	Average number of compound variables derived by Algorithm 1 (t is upper bounded for all D)	Average number of essential variables derived by Algorithm 1 ($t = 1$ for all D)	$\lceil \log_2 k \rceil$	M_{50}	$L - 1$
20	20	5.5	6.2 ($t = 7.8$)	5.5 ($t \leq 4$)	6.5	5	4.42	5
	60	8.1	8.9 ($t = 6.9$)	8 ($t \leq 3$)	9.3	6	7.26	7
	100	9.5	10.1 ($t = 7.4$)	9 ($t \leq 3$)	10.5	7	8.67	9
40	20	5.1	6.1 ($t = 16.1$)	5.5 ($t \leq 3$)	6.4	5	4.07	5
	60	8	8.9 ($t = 15.7$)	8 (Mostly $1-2$, Over all ≤ 4)	9.2	6	6.75	7
	100	9.4	10 ($t = 17.1$)	9 (Mostly $1-3$, Over all ≤ 5)	10.6	7	8.07	9
60	20	5	6.2 ($t = 23.9$)	5.6 ($t \leq 3$)	6.4	5	3.92	5
	60	8	8.9 ($t = 24.8$)	8 (Mostly $1-2$, Over all ≤ 3)	9.2	6	6.55	7
	100	9	10 ($t = 26.4$)	9 ($t \leq 4$)	10.6	7	7.85	9

The presented results are obtained by considering 100 random sets $D \subset \mathbb{F}_2^n$

Table 8 The performance of Algorithm 1 on arbitrary sets $D \subset \mathbb{F}_2^n$ in providing **essential variables**: Ceiling of M_{50} is the lower bound given by [32, Property 4.1], $L = 2\lceil \log_2(k+1) \rceil - 4$

n	$\#D = k$	Number of random sets D	Average number of compound variables derived by Algorithm 1 (in brackets are given lower and upper bounds over all D)	$\lceil \log_2 k \rceil$	M_{50}	L
5	5	100	3.1 (3-4)	3	2.01	2
	10	100	4.5 (4-5)	4	3.69	4
10	30	100	7.4 (7-8)	5	6.22	6
	50	100	8.7 (8-10)	6	7.80	8
	100	100	9.9 (9-10)	7	9.61	10
15	80	100	9.9 (9-10)	7	8.45	10
	150	100	11.6 (11-13)	8	10.38	12
	250	50	13.1 (12-14)	8	12.05	12
17	80	50	10 (9-10)	7	8.25	10
	200	30	12.4 (12-13)	8	10.97	12
	350	20	14 (13-16)	9	12.76	14
20	100	50	10.5 (9-12)	7	8.67	10
	300	20	13.6 (13-14)	9	11.85	14
	400	20	14.2 (13-15)	9	12.72	14

Note: Algorithm 1 stops when in **Step 1** there does not exist a vector σ_i with $wt(\sigma_i) = 1$

can be utilized in order to obtain claimed efficiency (see for instance [24, Section V], [15, Section 7.1], [9, Section 3]). Recall that, by definition, the set D is not generated by any rule and therefore the assumption on randomness is well motivated.

The main properties of Algorithm 1 can be summarized as follows:

- From presented results (Tables 6, 7 and 8) we conclude that Algorithm 1 can be seen as a semi-deterministic algorithm since it provides compound variables whose degree in most of the cases is upper bounded by 3 for various parameters n and k (set D is random). In addition, it provides a linear decomposition of a given $isig$-function which is nearly optimal (Table 6) in almost all cases for which the decomposition is not considered to be difficult (in terms of [32, Property 4.2]).
- The presented results indicate that the number of compound variables agrees with Properties 4.1, 4.2 and 4.3 given in [32]. In this context, Algorithm 1 shows somewhat weaker performance in derivation of essential variables (Tables 7 and 8) than the derivation of compound variables whose degree is not necessarily equal to 1 (Table 6). Regarding [32, Property 4.1], the difference $|r - \lfloor M_{50} \rfloor|$ in Table 7 for $t = 1$ ranges from 2.4 to 3.6, while in Table 8 is at maximum 2.

Remark 5.6 Notice in Table 6 that for sets D with large $\#\Delta_D$ Algorithm 1 does not provide a significant reduction of compound variables, which actually agrees to the upper bound $n \leq L$, which is Property 4.2 in [32]. In other words, for n relatively close to L, with $L \leq n$, the reduction of variables is considered to be difficult.

6 Conclusions

In this work we have shown that the linear decomposition of a given $isig$-function (defined on a set $D \subset \mathbb{F}_2^n$) is closely related to Walsh supports of its coordinate Boolean functions and the structure of D. It turns out that the representation of an $isig$-function, which refers to defining its coordinate functions on whole space $\mathbb{F}_2^n$, plays an important role. In this context, we have provided several general/specific construction methods of coordinate functions for which the linear decomposition is possible, along with the general test for checking whether the linear decomposition is possible at all (that is Theorem 4.3). In addition, we showed that the recursive employment of these results, in order to derive a linear decomposition of a given $isig$-function, is a promising approach which may lead to various new (semi)deterministic algorithms such as our Algorithm 1 given in Section 5.1.

Regarding the questions posed in Section 3, using the representation of an $isig$-function as in (4), by Theorem 4.2 and Algorithm 1 we have partially answered the question (I). While the question (II) is completely answered by Theorem 3.3, the question (III) is to a certain extend addressed by Theorem 4.2 in the case when Δ_D is not too large, since it provides an explicit method of linear decomposition of an $isig$-function.

Acknowledgments Samir Hodžić is supported in part by the Slovenian Research Agency (research program P3-0384 and Young Researchers Grant). Enes Pasalic is partly supported by the Slovenian Research Agency (research program P3-0384 and research project J1-9108). Also, the first two authors gratefully acknowledge the European Commission for funding the InnoRenew CoE project (Grant Agreement no. 739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European regional Development Fund). Anupam Chattopadhyay will like to acknowledge Tsutomu Sasao for helpful discussions related to the complexity of index generation functions.

Appendix

The performance of Algorithm 1 shown in Tables 6, 7 and 8 uses Proposition 5.1 for finding a basis of $\sigma_i^{\perp}$ in **Step 1**.

Example A.1 Let us consider $\sigma_i = (\alpha_1, \ldots, \alpha_4) = (1, 0, 1, 1) \in \mathbb{F}_2^4$ $(n - (i - 1) = 4)$. We have that $p = 1$ is the minimal index in $\{1, \ldots, 4\}$ such that $\alpha_p = 1$. Note that one may consider other indices p as well (not necessary the minimal one). Using the vectors $e_1, \ldots, e_4 \in \mathbb{F}_2^4$, we have that $\widehat{e}_j$ (for $j \in \{2, 3, 4\} = \{1, 2, 3, 4\} \setminus \{p\}$), defined as in Proposition 5.1, are given as

$$\begin{pmatrix} \widehat{e}_2 \\ \widehat{e}_3 \\ \widehat{e}_4 \end{pmatrix} = \begin{pmatrix} e_2 \\ e_1 \oplus e_3 \\ e_1 \oplus e_4 \end{pmatrix} = \begin{pmatrix} 0\ 1\ 0\ 0 \\ 1\ 0\ 1\ 0 \\ 1\ 0\ 0\ 1 \end{pmatrix}.$$

Thus the set $\{\widehat{e}_2, \widehat{e}_3, \widehat{e}_4\}$ is basis of $\sigma^{\perp}$.

References

1. Astola, J., Astola, P., Stanković, R., Tabus, I.: An algebraic approach to reducing the number of variables of incompletely defined discrete functions. In: 46th International Symposium on Multiple-Valued Logic, pp. 107–112 (2016)
2. Astola, J., Astola, P., Stanković, R., Tabus, I.: Index generation functions based on linear and polynomial transformations. In: 46th International Symposium on Multiple-Valued Logic, pp. 102–106 (2016)
3. Bhattacharyya, A., Indyk, P., Woodruff, D.P., Xie, N.: The complexity of linear dependence problems in vector spaces. Innovations Comput. Sci., 496–508. ISBN 978-7-302-24517-9 (2011)
4. Crama, Y., Hammer, P.L.: Boolean Functions: Theory, Algorithms, and Applications. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge (2011)
5. Een, N., Sorensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing. Lecture Notes in Computer Science, SAT 2003, vol. 2919, pp. 502–518. Springer, Berlin (2003)
6. Karpovsky, M.G., Stankovic, R.S., Astola, J.T.: Spectral Logic and Its Applications for the Design of Digital Devices. Wiley, Hoboken (2007)
7. Kolomeec, N., Pavlov, A.: Bent functions on the minimal distance. IEEE Region 8, SIBIRCON, Irkutsk Listvyanka, Russia (2010)
8. Lechner, R.J.: Harmonic analysis of switching functions. Recent Developments in Switching Theory, New York, Edited by: Amar Mukhopadhyay, pp. 121–228 (1971)
9. Luba, T., Borowik, G., Jankowski, C.: Gate-based decomposition of index generation functions. Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments, vol. 10031 (2016)
10. Mitchell, C.: Enumerating Boolean functions of cryptographic significance. J. Cryptol. **2**(3), 155–170 (1990)
11. Nagayama, S., Sasao, T., Butler, J.T.: An efficient heuristic for linear decomposition of index generation functions. In: 46nd International Symposium on Multiple-Valued Logic, pp. 96–101. Japan (2016)
12. Nagayama, S., Sasao, T., Butler, J.T.: A balanced decision tree based heuristic for linear decomposition of index generation functions. IEICE Trans. Inf. Syst. **E100.D**(8), 1583–1591 (2017)
13. Nečiporuk, E.I.: Network synthesis by using linear transformation of variables. Dokladi Akademii Nauk SSSR **123**(4), 610–612 (1958)
14. Pagiamtzis, K., Sheikholeslami, A.: Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. IEEE J. Solid State Circuits **41**(3), 712–727 (2006)
15. Sasao, T.: Index generation functions: tutorial. J. Multiple-Valued Logic Soft Comput. **23**(3–4), 235–263 (2014)
16. Sasao, T.: Multiple-valued input index generation functions: optimization by linear transformation. In: 42nd International Symposium on Multiple-Valued Logic, pp. 185–190. Canada (2012)
17. Sasao, T.: A reduction method for the number of variables to represent index generation functions: s-Min method. In: 45nd International Symposium on Multiple-Valued Logic, pp. 164–169. Canada (2015)
18. Sasao, T.: Index generation functions: theory and applications. In: International Symposium on Communications and Information Technologies, pp. 585–590. Japan (2010)
19. Sasao, T.: Linear transformations for variable reduction. ReedMuller Workshop (2011)
20. Sasao, T.: On the numbers of variables to represent multi-valued incompletely specified functions. In: 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools. France (2010)
21. Sasao, T.: On the numbers of variables to represent sparse logic functions. In: IEEE/ACM International Conference on Computer-Aided Design, pp. 45–51. San Jose, USA (2008)
22. Sasao, T.: Index generation functions: recent developments. In: 41st IEEE International Symposium on Multiple-Valued Logic, pp. 235–263. Finland (2011)
23. Sasao, T.: Memory-Based Logic Synthesis. Springer-Verlag, New York (2011)
24. Sasao, T.: Linear decomposition of index generation functions. In: 17th Asia and South Pacific Design Automation Conference, pp. 781–788 (2012)
25. Sasao, T.: Index generation functions: Minimization methods. In: 47th International Symposium on Multiple-Valued Logic, pp. 197–206 (2017)
26. Sasao, T.: An application of autocorrelation functions to find linear decompositions for incompletely specified index generation functions. In: 43rd International Symposium on Multiple-Valued Logic, pp. 96–102 (2013)
27. Sasao, T.: A linear decomposition of index generation functions: optimization using autocorrelation functions. J. Multiple-Valued Logic Soft Comput. **28**(1), 105–127 (2017)

28. Sasao, T., Fumishi, I., Iguch, Y.: A method to minimize variables for incompletely specified index generation functions using a SAT solver. In: International Workshop on Logic and Synthesis, Mountain View, pp. 161–167 (2015)
29. Sasao, T., Nakamura, T., Matsuura, M.: Representation of incompletely specified index generation functions using minimal number of compound variables. In: 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools, pp. 765–772 (2009)
30. Sasao, T., Matsuura, M., Nakahara, H.: A realization of Index generation functions using modules of uniform sizes. In: International Workshop on Logic and Synthesis, pp. 201–208. California (2010)
31. Sasao, T., Urano, Y., Iguchi, Y.: A method to find linear decompositions for incompletely specified index generation functions using difference matrix. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E97.A**(12), 2427–2433 (2014)
32. Sasao, T., Urano, Y., Iguchi, Y.: A lower bound on the number of variables to represent incompletely specified index generation functions. In: 44th International Symposium on Multiple-Valued Logic, pp. 7–12. Germany (2014)
33. Abboud, A., Lewi, K., Williams, R.: Losing weight by gaining edges. In: 22th Annual European Symposium on Algorithms, Lecture Notes in Computer Science, vol. 8737, pp. 1–12. Poland (2014)
34. Simovici, D.A., Zimand, M., Pletea, D.: Several remarks on index generation functions. In: 42nd IEEE International Symposium on Multiple-Valued Logic, pp. 179–184. Canada (2012)
35. Stanković, M., Stanković, R.: Variable reduction of index generating functions in Walsh-Hadamard domain. In: Proceedings Reed-Muller Workshop, pp. 80–85. Japan (2013)
36. Troy Nagle, H., Carroll, B.D., David Irwin, J.: An Introduction to Computer Logic (Prentice-Hall computer applications in electrical engineering series), 1st edn. Prentice-Hall, Englewood Cliffs (1975)
37. Varma, D., Trachtenberg, E.: Design automation tools for efficient implementation of logic functions by decomposition. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **8**(8), 901–916 (1989)
38. Wu, C.-K., Feng, D.: Boolean functions and their applications in cryptography. Advances in Computer Science and Technology (2016)