A New Insight into Constructing Cryptographic Boolean Functions via Walsh Spectral Analysis

Shaozheng He[1], Jiongjiong Ren[1*], Shaozhen Chen[1], Jiaxin Yan[1], Jianhua Hou[1]

[1*]Information Engineering University, Zhengzhou, 450001, China.

*Corresponding author(s). E-mail(s): jiongjiong_fun@163.com;
Contributing authors: shawnz_he@163.com; chenshaozhen@vip.sina.com; jia_xin_yan@126.com; jianhua.hou@foxmail.com;

Abstract

Given that the Walsh spectrum directly determines key cryptographic properties of Boolean functions, the construction of such functions with desired spectral features has been a major research focus for decades. In this study, we first establish a unified framework for a class of specific Boolean function construction problems corresponding to Walsh transform, which we formally define as **Problem**. To tackle the **Problem**, we first designed the Iterative Walsh Recovery (IWR) algorithm as a framework, then added Forgetting and Greedy strategies for heuristic optimization to obtain the FG-IWR algorithm, and finally proved a necessary condition for optimization, ultimately proposing the Optimized Iterative Walsh Recovery (OIWR) algorithm. Through rigorous theoretical analysis and experimental validation, our algorithm simultaneously achieves theoretical guarantees, design flexibility, and computational efficiency. For application, we further present a novel construction method for low-weight correlation immune functions using the OIWR algorithm. Experimental results show that our method successfully addresses two fundamental constraints of Mesnager-Su's approach: limited construction capacity and power-of-two weight restrictions.

Keywords: Boolean functions, Walsh-Hadamard transform, Correlation immunity, Hamming weight, Nonlinearity

1 Introduction

1.1 Problem Formulation

In cryptography, Boolean functions serve as critical components of cryptographic systems, providing the essential nonlinearity for various cryptographic primitives, such as combiners and filters in linear feedback shift registers (LFSRs), and substitution boxes (S-boxes) in block ciphers[1]. Boolean functions possess multiple equivalent representations, among which the Walsh spectrum—derived via the *Walsh-Hadamard Transform* (see relation (1))—plays a pivotal role. The Walsh spectrum directly characterizes several crucial cryptographic properties of Boolean functions, such as balancedness [2], correlation immunity [3], and nonlinearity [4]. Therefore, the design of cryptographic Boolean functions via Walsh spectral analysis has received significant attention from cryptographers for decades.

The algebraic construction methodology based on Walsh spectra has long been established as a classical approach in the field. Its fundamental principle involves optimizing the cryptographic properties of Boolean functions through careful analysis of their Walsh-Hadamard transform characteristics. One significant optimization objective is the pursuit of high correlation immunity order (where balancedness corresponds to 0-order correlation immunity). Related studies include, for instance, the construction of low Hamming weight correlation immune functions in [5–7], and the design of resilient functions (i.e., balanced correlation immune functions) investigated in [8, 9]. Additional feasible optimization criteria based on Walsh spectral analysis include high nonlinearity[10, 11], high algebraic degree [12, 13] and optimal algebraic immunity [14, 15]. However, since most cryptographic properties exhibit inherent trade-offs in the Walsh domain, practical applications require careful balancing of these spectral-derived parameters, as discussed in [16–18].

Recent advances in Walsh-Hadamard based cryptographic Boolean function construction have increasingly employed metaheuristic algorithms (e.g., simulated annealing [19] and hill climbing [20]) and heuristic methods (e.g., greedy search), demonstrating significant potential in achieving superior cryptographic properties [21]. Techniques based on the Walsh-Hadamard transform feature prominently in this area, including its use in evolutionary algorithms to build correlation-immune functions [22, 23], in the study of plateaued functions for high nonlinearity [24–27], and in conjunction with heuristic methods for constructing cryptographic primitives [28–30].

Compared with traditional algebraic construction approaches, heuristic and metaheuristic methods offer several distinct advantages for designing cryptographic Boolean functions: (1) When simultaneously optimizing multiple cryptographic properties, heuristic/metaheuristic methods enable straightforward multi-objective optimization through flexible constraint incorporation and fitness function adjustment, whereas algebraic methods typically require sophisticated mathematical techniques. (2) Heuristic/Metaheuristic approaches demonstrate superior adaptability in parameter selection, unlike algebraic constructions that often demand specific dimensional requirements and other constraints. However, these methods also present notable limitations: (1) They generally lack rigorous mathematical proofs and theoretical

guarantees, with performance heavily dependent on experimental parameter configurations. (2) Their computational complexity imposes practical constraints, making solutions for problems with $n > 16$ variables typically intractable[21].

This study seeks to propose a novel technique for constructing cryptographic Boolean functions via Walsh spectral analysis, which simultaneously offers theoretical guarantees, design flexibility, and computational efficiency. We first attempt to consolidate the cryptographic Boolean function construction problems with special Walsh spectra into a unified framework, denoted as the **Problem**.

Problem: Let n be a positive integer. Given a finite set of pairs

$$S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m)\},$$

where each a_i is an integer with $0 \leq a_i \leq 2^n - 1$, each b_i is an integer with $-2^n \leq b_i \leq 2^n$, and $1 \leq m \leq 2^n$, find all Boolean functions on n variables $f : \{0,1\}^n \to \{0,1\}$ such that

$$\hat{\boldsymbol{f}}(a_i) = b_i, \quad i = 1, 2, \ldots, m,$$

where $\hat{\boldsymbol{f}}$ denotes the Walsh spectrum of f (see relation (1)).

If this problem can be effectively resolved, we could flexibly construct set S to obtain Boolean functions with desired cryptographic properties, e.g., adding $(0, b)$ to S enables construction of Boolean functions with specific Hamming weights. We now present a concise analysis of **Problem**.

Given the truth table of a Boolean function on n variables f, we denote the process of determining whether f satisfies S as the verification of **Problem**, which has a time complexity of $O(n2^n)$ with the *Fast Walsh-Fourier Transform* (FWT) (see 2.2). However, the time complexity of solving the **Problem** by enumerating and verifying f reaches $O(2^{2^n})$, which means solving it is more challenging than its verification. When $m = \Theta(2^n)$, which is our most common scenario, since the input size of the problem is $\Theta(2^n)$, the decision version of the **Problem** belongs to the NP complexity class [31].

1.2 Our Contributions

This paper aims to design an algorithm for solving the **Problem** and investigate its practical applications in the construction of cryptographic Boolean functions. We begin by designing the *Optimized Iterative Walsh Recovery* (OIWR) algorithm to solve the **Problem** via three key steps, supported by theoretical guarantees and experimental validations.

First, we proposed a baseline algorithm named the *Iterative Walsh Recovery* (IWR) algorithm. By defining the "error" ε and proving its monotonically non-increasing property during iterations (see Proposition 1), we provided theoretical guarantees for the convergence of IWR algorithm and identified its tendency to fall into local optima. Additionally, we provide a rough error estimation for the initial iteration (see Proposition 3) and conduct experimental verification, offering a performance guarantee for the IWR algorithm.

Second, to address the issue of the IWR algorithm's susceptibility to local optima, we introduced heuristic optimization by proposing the *Forgetting Strategy* and *Greedy Strategy* to obtain the FG-IWR algorithm, which outputs an approximate solution f_0

(see Definition 3) and a spectral position set P (see Definition 4). Through further enumeration of the set P and verification with FWT, we can derive the solutions to the **Problem**. For experimental validation, we evaluated the practical effects on 14 self-constructed instances with high nonlinearity (n ranges from 13 to 19).

Finally, to further optimize the solution process based on the FG-IWR algorithm, we first establish a necessary condition (Proposition 6) and then derive the Optimized IWR (OIWR) algorithm. Let Δ denote the collection of candidate position sets that satisfy this necessary condition (formally defined in Section 5). The OIWR algorithm reduces the time complexity of the verification phase from $O(n\,2^{n+|P|})$ to $O(|\Delta|\,n\,2^n)$. In practical validation, the OIWR algorithm successfully solved all the self-constructed instances within 400 seconds (see in Table 5), which demonstrated its high solving efficiency in constructing Boolean functions with high nonlinearity.

For cryptographic applications, we proposed a novel method to construct correlation immune functions with low Hamming weight based on the OIWR algorithm. By flexibly setting the set S, our method enables the attainment of diverse construction objectives. Experimental results show two key advancements relative to the work of Mesnager and Su [5]. On the one hand, under the same weight constraints as the Mesnager-Su approach, our method generates abundant low-weight correlation immune functions in a simple way. Specifically, we constructed 52,339 distinct instances of 12-variable, 1st-order correlation immune functions with a Hamming weight of 256. On the other hand, our method overcomes the power-of-two weight limitation to produce numerous low-weight correlation immune functions. Specifically, we constructed 141,893 instances of 12-variable, 1st-order correlation immune functions with Hamming weight ranging from 222 to 256.

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2, we describe the basic definitions, results, and tools to obtain our results. In Section 3, we introduce the baseline IWR algorithm and provide theoretical and experimental verification. In Section 4, we conduct heuristic optimization to obtain the *FG-IWR* algorithm and validate its effectiveness using self-constructed instances. In Section 5, we introduce our optimization with a necessary condition to the final step and obtain the total OIWR algorithm. In Section 6, we introduce our new framework for constructing low-weight correlation immune functions using the OIWR algorithm and experiments based on Mesnager and Su's method.

2 Preliminaries

2.1 Basic Definitions

Let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{Z}$ the set of integers, $\mathbb{R}$ the real number field, and $GF(2) = \{0, 1\}$ the binary field with addition and multiplication performed modulo 2. Let n, m be positive integers, and $\mathbb{Z}^n$, $\mathbb{R}^n$, $GF(2)^n$ respectively be the

n-dimensional vector space over $\mathbb{Z}$, $\mathbb{R}$ and $GF(2)$. Let $\mathbb{Z}^{n\times m}$, $\mathbb{R}^{n\times m}$, $GF(2)^{n\times m}$ respectively denote the set of all $n \times m$ matrices over $\mathbb{Z}$, $\mathbb{R}$ and $GF(2)$. For a set X, let $|X|$ denote its cardinality; that is, the number of elements in X if X is finite.

For any real vector $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{R}^n$, we define its *2-norm* as $\|\boldsymbol{a}\|_2 = \sqrt{\sum_{i=0}^{n-1} a_i^2}$ and denote its i-th component by $\boldsymbol{a}[i] = a_i$ for each index $i \in \{0, 1, \ldots, n-1\}$. When $\boldsymbol{a} \in GF(2)^n$, we define the support of $\boldsymbol{a}$ as the set $\mathrm{supp}(\boldsymbol{a}) = \{0 \le i \le n-1 \,|\, \boldsymbol{a}[i] = 1\}$, and its Hamming weight as the cardinality of its support, i.e., $\mathrm{wt}(\boldsymbol{a}) = |\mathrm{supp}(\boldsymbol{a})|$. For the sake of simplicity, we do not distinguish the vector $\boldsymbol{a} \in GF(2)^n$ and the integer $\sum_{i=0}^{n-1} \boldsymbol{a}[i]2^i \in \{0, 1, \cdots, 2^n - 1\}$ if the context is clear, since they are one-to-one corresponding. To avoid confusion, we denote the sum over $\mathbb{Z}$ by $+$, and the sum over $GF(2)$ by $\oplus$. The product of two matrices $\mathbf{A}$ and $\mathbf{B}$ is denoted by $\mathbf{AB}$ and the *dot product* (modulo 2) for $\boldsymbol{i}, \boldsymbol{j} \in GF(2)^n$ is denoted by $\boldsymbol{i} \cdot \boldsymbol{j}$, i.e.,

$$\boldsymbol{i} \cdot \boldsymbol{j} = \sum_{k=0}^{n-1} \boldsymbol{i}[k]\boldsymbol{j}[k] \pmod 2.$$

Let $\mathbf{I}_n$ denote the identity matrix, and a *permutation matrix* $\mathbf{P}$ [32] of size $n \times n$ is a square matrix obtained by permuting the rows of $\mathbf{I}_n$. For any $\mathbf{A} \in \mathbb{R}^{n\times m}$, left multiplication by $\mathbf{P}$ (i.e., $\mathbf{PA}$) permutes the rows of $\mathbf{A}$. $\mathbf{P}$ is an *orthogonal matrix*, i.e. $\mathbf{P}^\mathsf{T}\mathbf{P} = \mathbf{P}\mathbf{P}^\mathsf{T} = \mathbf{I}_n$, which indicates for any $\boldsymbol{a} \in \mathbb{R}^n$, we have $\|\mathbf{P}\boldsymbol{a}\|_2 = \|\boldsymbol{a}\|_2$.

For any $n \in \mathbb{Z}$, the Boolean function f in n variables is a map of $GF(2)^n$ onto $GF(2)$. The set of all Boolean functions in n variables is denoted by $\mathcal{B}_n$. For a given $f \in \mathcal{B}_n$, let $\boldsymbol{f}$ be the truth table of f, which is a (0,1)-sequence defined by $(f(\mathbf{0}), f(\mathbf{1}), \ldots, f(\mathbf{2^n - 1}))^T$, where $\mathbf{0} = (0, \ldots, 0, 0), \mathbf{1} = (0, \ldots, 0, 1), \ldots, \mathbf{2^n - 1} = (1, \ldots, 1, 1)$. The matrix formed by taking the binary vector representations of all $\boldsymbol{i} \in \mathrm{supp}(\boldsymbol{f})$ as rows is defined as the *support matrix* of f, denoted by $\mathrm{Supp}(f)$ [5]. The Walsh spectrum of f, denoted by $\hat{\boldsymbol{f}} = (\hat{f}(\mathbf{0}), \hat{f}(\mathbf{1}), \ldots, \hat{f}(\mathbf{2^n - 1}))^T$, is defined via the *Walsh-Hadamard Transform* (WHT):

$$\hat{f}(\alpha) = \sum_{x \in GF(2)^n} (-1)^{\alpha\cdot x \oplus f(x)}, \alpha \in GF(2)^n. \tag{1}$$

Conversely, the truth table can be recovered from the Walsh spectrum through the inverse transform:

$$(-1)^{f(x)} = 2^{-n} \sum_{\alpha \in GF(2)^n} \hat{f}(\alpha)(-1)^{\alpha\cdot x}, x \in GF(2)^n. \tag{2}$$

Together, equations (1) and (2) establish a bijection between the truth table of f and its Walsh spectrum.

From (1), we have $\hat{\boldsymbol{f}}[0] = 2^n - 2\mathrm{wt}(\boldsymbol{f})$. A Boolean function f is balanced if $\mathrm{wt}(\boldsymbol{f}) = 2^{(n-1)}$, i.e., $\hat{\boldsymbol{f}}[0] = 0$. In terms of Walsh spectra, f is called *d-th order correlation*

immune if [33]:

$$\hat{f}(\boldsymbol{a}) = 0 \text{ for all } \boldsymbol{a} \in GF(2)^n \text{ with } 1 \leq \mathrm{wt}(\boldsymbol{a}) \leq d,$$

and the nonlinearity of f is given by [34]

$$N_f = 2^{n-1} - \frac{1}{2} \max_{\omega \in GF(2)^n} |\hat{f}(\omega)|,$$

for even n, the maximal nonlinearity $NL_{max} = 2^{n-1} - 2^{n/2-1}$ is attained for the bent functions [4]. When n is odd, the not tight upper bound of maximum nonlinearity is $2\lfloor 2^{n-2} - 2^{n/2-2} \rfloor$ [18].

The polar table of f is defined as $(-1)^{\boldsymbol{f}} = ((-1)^{f(\mathbf{0})}, (-1)^{f(\mathbf{1})}, ..., (-1)^{f(\mathbf{2^n-1})})^T$. Clearly, $(-1)^{\boldsymbol{f}}[i] = 1 - 2\boldsymbol{f}[i], i \in \{0, 1, ..., 2^n - 1\}$. It requires $O(2^n)$ operations to transform between the truth table and the polar table, which is significantly small compared with other operations in $\mathcal{B}_n$; therefore, for convenience of description, we overlook the difference between them if the context is clear.

Let $\mathbf{A}[i, j]$ denote the element at i-th row and j-th column of matrix $\mathbf{A}$. The *Walsh-Hadamard matrix* of size $2^n \times 2^n$ is defined as

$$\mathbf{H}_{2^n}[i, j] = (-1)^{\boldsymbol{i} \cdot \boldsymbol{j}}, \text{ where } i, j \in \{0, 1, ..., 2^n - 1\},$$

with $\mathbf{H}_{2^n}^T \mathbf{H}_{2^n} = \mathbf{H}_{2^n} \mathbf{H}_{2^n}^T = 2^n \mathbf{I}_{2^n}$. Therefore, for $\boldsymbol{a} \in \mathbb{R}^{2^n}$, we have

$$||\mathbf{H}_{2^n} \boldsymbol{a}||_2^2 = 2^n ||\boldsymbol{a}||_2^2.$$

With the Walsh-Hadamard matrix, the relation (1) and (2) can be written as

$$\begin{cases} \hat{\boldsymbol{f}} = \mathbf{H}_{2^n} (-1)^{\boldsymbol{f}} \\ (-1)^{\boldsymbol{f}} = \frac{\mathbf{H}_{2^n} \hat{\boldsymbol{f}}}{2^n} \end{cases}. \tag{3}$$

The Walsh spectrum follows the Parseval Identity:

$$||\hat{\boldsymbol{f}}||_2^2 = 2^n ||(-1)^{\boldsymbol{f}}||_2^2 = 2^{2n}.$$

Definition 1 [35] Let $\mathcal{X}$ be a normed vector space (typically $\mathbb{R}^n$) equipped with a norm $\|\cdot\|$. An iterative algorithm $\mathcal{F}$ is said to be *convergent* if, for any admissible initial point $x_0 \in \mathcal{X}$, the sequence $\{x_k\}_{k=0}^{\infty}$ generated by $\mathcal{F}$ satisfies the Cauchy criterion: for every $\epsilon > 0$, there exists a positive integer $N(\epsilon)$ such that

$$\|x_m - x_n\| < \epsilon \quad \text{for all } m, n \geq N(\epsilon).$$

2.2 FWT (Fast Walsh-Fourier Transform) Algorithm

Computing the Walsh transform using relation (3) has a time complexity of 2^{2n} additions/subtractions. A faster method exists to obtain the Walsh spectrum using the *Fast*

Walsh-Fourier transform (FWT)[36], which is a discrete version of the *Fast Fourier transform*. It requires only $n2^n$ additions/subtractions. The pseudocode for the FWT is presented in Algorithm 1.

Algorithm 1 FWT algorithm

Input: n —Variable Size of f
$\boldsymbol{v}$ —Polar Table of f
Output: $\boldsymbol{w}$ —Walsh Spectrum of f
Function Name: Polar_Walsh()

Initialization: $\boldsymbol{w} \Leftarrow \boldsymbol{v}$
for $i = 1, 2, ..., n$ **do**
  for $h = 0, 1, ..., 2^{n-i} - 1$ **do**
    for $j = 0, 1, ..., 2^{i-1} - 1$ **do**
      $\boldsymbol{w}[j + h \cdot 2^i] \Leftarrow \boldsymbol{w}[j + h \cdot 2^i] + \boldsymbol{w}[j + 2^{i-1} + h \cdot 2^i]$
      $\boldsymbol{w}[j + 2^{i-1} + h \cdot 2^i] \Leftarrow \boldsymbol{w}[j + h \cdot 2^i] - \boldsymbol{w}[j + 2^{i-1} + h \cdot 2^i]$
    end for
  end for
end for
return $\boldsymbol{w}$

In addition, according to equation (3), the inverse of the FWT is shown in Algorithm 2.

Algorithm 2 Inverse of the FWT algorithm

Input: n —Variable Size of f
$\boldsymbol{w}$ —Walsh Spectrum of f
Output: $\boldsymbol{v}$ —Polar Table of f
Function Name: Walsh_Polar()

Initialization: $\boldsymbol{v} \Leftarrow \boldsymbol{w}$
for $i = 1, 2, ..., n$ **do**
  for $h = 0, 1, ..., 2^{n-i} - 1$ **do**
    for $j = 0, 1, ..., 2^{i-1} - 1$ **do**
      $\boldsymbol{v}[j + h \cdot 2^i] \Leftarrow \boldsymbol{v}[j + h \cdot 2^i] + \boldsymbol{v}[j + 2^{i-1} + h \cdot 2^i]$
      $\boldsymbol{v}[j + 2^{i-1} + h \cdot 2^i] \Leftarrow \boldsymbol{v}[j + h \cdot 2^i] - \boldsymbol{v}[j + 2^{i-1} + h \cdot 2^i]$
    end for
  end for
end for
for $i = 0, 1, ...2^n - 1$ **do**
  $\boldsymbol{v}[i] \Leftarrow \boldsymbol{v}[i]/2^n$
end for
return $\boldsymbol{v}$

Algorithm 3 IWR algorithm

Input: $S = \{(a_i, b_i), 1 \leq i \leq m\}$ —Part-Walsh Spectrum
$\boldsymbol{w}^{(1)} \in \{-2^n, -2^n + 2, ..., 0, ..., 2^n + 2, 2^n\}^{2^n}$ —Initialization Vector
Output: $\boldsymbol{w}^\dagger, \boldsymbol{v}^\dagger$

$\boldsymbol{w}^{(1)}[a_k] \Leftarrow b_k$ for (a_k, b_k) in S, $i \Leftarrow 1$
while *True* **do**
  $\tilde{\boldsymbol{v}}^{(i)} \Leftarrow Walsh_Polar(\boldsymbol{w}^{(i)})$
  $\boldsymbol{v}^{(i)} \Leftarrow sgn(\tilde{\boldsymbol{v}}^{(i)})$ ▷ Correct $\boldsymbol{v}$ to a right form
  $\boldsymbol{w}^{(i+1)} \Leftarrow Polar_Walsh(\boldsymbol{v}^{(i)})$
  if $\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)}$ **then**
    break
  end if
  for $j = 1, 2, ..., m$ **do** $\boldsymbol{w}^{(i+1)}[a_j] = b_j$ ▷ Correct $\boldsymbol{w}$ with S
  end for
  $i \Leftarrow i + 1$
end while
$\boldsymbol{w}^\dagger \Leftarrow \boldsymbol{w}^{(i+1)}$
$\boldsymbol{v}^\dagger \Leftarrow \boldsymbol{v}^{(i)}$
return $\boldsymbol{w}^\dagger, \boldsymbol{v}^\dagger$

3 IWR Algorithm: The Baseline Algorithm

3.1 Algorithm Design

For clarity of presentation, we denote the truth table of a Boolean function $f \in \mathcal{B}_n$ a *solution* to **Problem** (or, when the context is clear, a solution of the set S) if its Walsh spectrum satisfies $\hat{\boldsymbol{f}}(a_i) = b_i$ for all $(a_i, b_i) \in S$. The goal of our algorithmic design is to determine as many solutions as possible. Building upon the FWT algorithm's capability for efficient conversion between complete Walsh spectra and polar tables, we design the *Iterative Walsh Recovery* (IWR) algorithm (Algorithm 3). To ensure a smooth iterative process, we introduce a sign function $sgn()$ to transform the vector $\boldsymbol{v}$ into the correct form of a polar table at each iteration:

Definition 2 Given any vector $\boldsymbol{h} \in \mathbb{R}^l$, we define function $sgn()$ as

$$sgn(\boldsymbol{h}) = (sign(\boldsymbol{h}[0]), sign(\boldsymbol{h}[1]), ..., sign(\boldsymbol{h}[l])) \text{ ,where}$$
$$sign(\boldsymbol{h}[i]) = \begin{cases} -1, \boldsymbol{h}[i] \leq 0, \\ 1, \boldsymbol{h}[i] > 0, \end{cases} \quad 0 \leq i \leq l. \tag{4}$$

Next, we analyze Algorithm 3 through both theoretical and experimental approaches.

3.2 Convergence of IWR Algorithm

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ be the set of indices in S. Let $\mathbf{P}$ be a $2^n \times 2^n$ permutation matrix that reorders the rows of $\mathbf{H}_{2^n}$ such that the rows indexed by $\mathcal{A}$ are placed in

the first m positions, i.e.,

$$\mathbf{P}\mathbf{H}_{2^n} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix},$$

where $\mathbf{A}$ is the $m \times 2^n$ submatrix corresponding to the known spectral positions, and $\mathbf{B}$ is the $(2^n - m) \times 2^n$ submatrix of the remaining rows.

In Algorithm 3, for $t \in \mathbb{N}$, we have $\mathbf{P}\boldsymbol{w}^{(t+1)} = \begin{pmatrix} \boldsymbol{b} \\ \mathbf{B}\boldsymbol{v}^{(t)} \end{pmatrix}$.

Lemma 1 *For any $\boldsymbol{v} \in \{-1, 1\}^{2^n}$, $t \in \{1, 2, ...\}$, we have*

$$\|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \mathbf{P}\boldsymbol{w}^{(t)}\|_2^2 \leq \|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v} - \mathbf{P}\boldsymbol{w}^{(t)}\|_2^2.$$

Equivalently, since $\mathbf{P}^T\mathbf{P} = \mathbf{I}_{2^n}$, we have

$$\|\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \boldsymbol{w}^{(t)}\|_2^2 \leq \|\mathbf{H}_{2^n}\boldsymbol{v} - \boldsymbol{w}^{(t)}\|_2^2.$$

Proof From the definition of the sgn function, we have

$$|\boldsymbol{v}^{(t)}[i] - \tilde{\boldsymbol{v}}^{(t)}[i]| \leq |\boldsymbol{v}[i] - \tilde{\boldsymbol{v}}^{(t)}[i]|, \forall i \in \{0, 1, 2, ..., 2^n - 1\}.$$

Therefore,

$$\|\boldsymbol{v}^{(t)} - \tilde{\boldsymbol{v}}^{(t)}\|_2^2 \leq \|\boldsymbol{v} - \tilde{\boldsymbol{v}}^{(t)}\|_2^2.$$

Since $\mathbf{H}_{2^n}\tilde{\boldsymbol{v}}^{(t)} = \boldsymbol{w}^{(t)}$, we have

$$\mathbf{P}\mathbf{H}_{2^n}\tilde{\boldsymbol{v}}^{(t)} = \mathbf{P}\boldsymbol{w}^{(t)}.$$

Now, we compute:

$$\begin{aligned}
\|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \mathbf{P}\boldsymbol{w}^{(t)}\|_2^2 &= \|\mathbf{P}(\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \boldsymbol{w}^{(t)})\|_2^2 \\
&= \|\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \boldsymbol{w}^{(t)}\|_2^2 \\
&= \|\mathbf{H}_{2^n}(\boldsymbol{v}^{(t)} - \tilde{\boldsymbol{v}}^{(t)})\|_2^2 \\
&= 2^n\|\boldsymbol{v}^{(t)} - \tilde{\boldsymbol{v}}^{(t)}\|_2^2 \\
&\leq 2^n\|\boldsymbol{v} - \tilde{\boldsymbol{v}}^{(t)}\|_2^2 \\
&= \|\mathbf{H}_{2^n}(\boldsymbol{v} - \tilde{\boldsymbol{v}}^{(t)})\|_2^2 \\
&= \|\mathbf{H}_{2^n}\boldsymbol{v} - \boldsymbol{w}^{(t)}\|_2^2 \\
&= \|\mathbf{P}(\mathbf{H}_{2^n}\boldsymbol{v} - \boldsymbol{w}^{(t)})\|_2^2 \\
&= \|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v} - \mathbf{P}\boldsymbol{w}^{(t)}\|_2^2.
\end{aligned}$$

□

Proposition 1 *In Algorithm 3, for $t \in \{1, 2, ...\}$, define the error at the t-th round as*

$$\varepsilon(\boldsymbol{v}^{(t)}) = \|\mathbf{A}\boldsymbol{v}^{(t)} - \boldsymbol{b}\|_2^2.$$

Then

$$\varepsilon(\boldsymbol{v}^{(t+1)}) \leq \varepsilon(\boldsymbol{v}^{(t)}) - \|\mathbf{B}\boldsymbol{v}^{(t+1)} - \mathbf{B}\boldsymbol{v}^{(t)}\|_2^2.$$

Proof Recall that $\mathbf{P}\mathbf{H}_{2^n} = \begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}$ and $\mathbf{P}\boldsymbol{w}^{(t+1)} = \begin{pmatrix}\boldsymbol{b}\\ \mathbf{B}\boldsymbol{v}^{(t)}\end{pmatrix}$.

By Lemma 1, we have

$$\|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v}^{(t+1)} - \mathbf{P}\boldsymbol{w}^{(t+1)}\|_2^2 \le \|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v}^{(t)} - \mathbf{P}\boldsymbol{w}^{(t+1)}\|_2^2.$$

Substituting the expressions for $\mathbf{P}\mathbf{H}_{2^n}$ and $\mathbf{P}\boldsymbol{w}^{(t+1)}$, we obtain

$$\left\|\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\boldsymbol{v}^{(t+1)} - \begin{pmatrix}\boldsymbol{b}\\ \mathbf{B}\boldsymbol{v}^{(t)}\end{pmatrix}\right\|_2^2 \le \left\|\begin{pmatrix}\mathbf{A}\\ \mathbf{B}\end{pmatrix}\boldsymbol{v}^{(t)} - \begin{pmatrix}\boldsymbol{b}\\ \mathbf{B}\boldsymbol{v}^{(t)}\end{pmatrix}\right\|_2^2.$$

Expanding the squared norms, we have

$$\|\mathbf{A}\boldsymbol{v}^{(t+1)} - \boldsymbol{b}\|_2^2 + \|\mathbf{B}\boldsymbol{v}^{(t+1)} - \mathbf{B}\boldsymbol{v}^{(t)}\|_2^2 \le \|\mathbf{A}\boldsymbol{v}^{(t)} - \boldsymbol{b}\|_2^2.$$

Therefore,

$$\varepsilon(\boldsymbol{v}^{(t+1)}) = \|\mathbf{A}\boldsymbol{v}^{(t+1)} - \boldsymbol{b}\|_2^2 \le \|\mathbf{A}\boldsymbol{v}^{(t)} - \boldsymbol{b}\|_2^2 - \|\mathbf{B}\boldsymbol{v}^{(t+1)} - \mathbf{B}\boldsymbol{v}^{(t)}\|_2^2 = \varepsilon(\boldsymbol{v}^{(t)}) - \|\mathbf{B}\boldsymbol{v}^{(t+1)} - \mathbf{B}\boldsymbol{v}^{(t)}\|_2^2.$$

□

Proposition 2 *Algorithm 3 converges within a finite number of rounds. Moreover, the algorithm converges at round T if and only if*

$$\mathbf{B}\boldsymbol{v}^{(T+1)} = \mathbf{B}\boldsymbol{v}^{(T)}.$$

Proof From Proposition 1, $\varepsilon(\boldsymbol{v}^{(t)})$ is monotonically non-increasing. Since the state space of $\boldsymbol{w}^{(t)}$ is finite (each component lies in $\{-2^n, -2^n+2, \ldots, 2^n\}$), there must exist $T > 0$ such that

$$\varepsilon(\boldsymbol{v}^{(T+1)}) = \varepsilon(\boldsymbol{v}^{(T)}).$$

By Proposition 1, this equality holds if and only if $\|\mathbf{B}\boldsymbol{v}^{(T+1)} - \mathbf{B}\boldsymbol{v}^{(T)}\|_2^2 = 0$, i.e.,

$$\mathbf{B}\boldsymbol{v}^{(T+1)} = \mathbf{B}\boldsymbol{v}^{(T)}.$$

Now, using the update rule for $\boldsymbol{w}^{(t+1)}$ in the permuted system:

$$\mathbf{P}\boldsymbol{w}^{(T+2)} = \begin{pmatrix}\boldsymbol{b}\\ \mathbf{B}\boldsymbol{v}^{(T+1)}\end{pmatrix} = \begin{pmatrix}\boldsymbol{b}\\ \mathbf{B}\boldsymbol{v}^{(T)}\end{pmatrix} = \mathbf{P}\boldsymbol{w}^{(T+1)}.$$

Since $\mathbf{P}$ is invertible, we have $\boldsymbol{w}^{(T+2)} = \boldsymbol{w}^{(T+1)}$. This implies that the input to Algorithm 3 in round $T+2$ is identical to that in round $T+1$. Hence, the computed $\boldsymbol{v}^{(T+2)}$ will equal $\boldsymbol{v}^{(T+1)}$. By induction, we obtain

$$\boldsymbol{v}^{(T+1)} = \boldsymbol{v}^{(T+2)} = \cdots.$$

By Definition 1, Algorithm 3 converges within a finite number of rounds. □

Proposition 2 indicates that Algorithm 3 iteratively approaches the part-Walsh spectral constraints in S, guaranteeing convergence to a local optimum within finite steps. However, the algorithm lacks the capability to escape local optima once encountered. We experimentally verified this on an instance with $n = 16$ and $|S| = 31,955$ where a solution exists. We set $\boldsymbol{w}^{(1)} = \mathbf{P}^{-1}\begin{pmatrix}\boldsymbol{b}\\ \mathbf{0}\end{pmatrix}$, and conducted the IWR algorithm. Denote the number of spectral points in S satisfied by $\boldsymbol{v}^{(t)}$ as $cnt(t)$ and results are shown in Table 1.

3.3 Error Upper Bound of the Solution Obtained by the IWR Algorithm

Through Proposition 2, we can obtain the theoretical guarantee for the convergence of Algorithm 3. Next, we will discuss the quality of the solution obtained by the algorithm.

Proposition 3 *Assume the existence of a solution* $\boldsymbol{v}^*$ *and define* $\boldsymbol{w}^* = \mathbf{H}_{2^n}\boldsymbol{v}^*$. *Let the initial vector be*

$$\boldsymbol{w}^{(1)} = \mathbf{P}^T \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{d}^{(1)} \end{pmatrix},$$

and denote

$$\mathbf{P}\boldsymbol{w}^{(1)} = \mathbf{P}\boldsymbol{w}^* + \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix}.$$

Let $\boldsymbol{\eta} = \tilde{\boldsymbol{v}}^{(1)} - \boldsymbol{v}^*$. *Then*

$$\varepsilon(\boldsymbol{v}^{(1)}) = \|\mathbf{A}\boldsymbol{v}^{(1)} - \boldsymbol{b}\|_2^2 \le 4\|\boldsymbol{\delta}\|_2^2.$$

Proof Since $\tilde{\boldsymbol{v}}^{(1)} = \frac{1}{2^n}\mathbf{H}_{2^n}\boldsymbol{w}^{(1)}$, we have

$$\boldsymbol{\eta} = \tilde{\boldsymbol{v}}^{(1)} - \boldsymbol{v}^* = \frac{1}{2^n}\mathbf{H}_{2^n}(\boldsymbol{w}^{(1)} - \boldsymbol{w}^*).$$

Note that $\mathbf{P}(\boldsymbol{w}^{(1)} - \boldsymbol{w}^*) = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix}$. Since $\mathbf{P}$ is orthogonal, $\boldsymbol{w}^{(1)} - \boldsymbol{w}^* = \mathbf{P}^T \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix}$. Hence,

$$\boldsymbol{\eta} = \frac{1}{2^n}\mathbf{H}_{2^n}\mathbf{P}^T \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix}.$$

Since $\mathbf{H}_{2^n}\mathbf{P}^T = (\mathbf{P}\mathbf{H}_{2^n})^T$, we obtain

$$\|\boldsymbol{\eta}\|_2^2 = \frac{1}{4^n} \cdot 2^n \left\| \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix} \right\|_2^2 = \frac{1}{2^n}\|\boldsymbol{\delta}\|_2^2.$$

Since $\boldsymbol{v}^{(1)} = \operatorname{sgn}(\tilde{\boldsymbol{v}}^{(1)})$ and $\tilde{\boldsymbol{v}}^{(1)} = \boldsymbol{\eta} + \boldsymbol{v}^*$, we have

$$|\boldsymbol{v}^{(1)}[j] - \boldsymbol{v}^*[j]| = \begin{cases} 0, & \operatorname{sgn}(\boldsymbol{v}^*[j] + \boldsymbol{\eta}[j]) = \boldsymbol{v}^*[j], \\ 2, & \text{otherwise.} \end{cases}$$

Therefore,

$$\|\boldsymbol{v}^{(1)} - \boldsymbol{v}^*\|_2^2 = 4\big|\{j \mid \operatorname{sgn}(\boldsymbol{v}^*[j] + \boldsymbol{\eta}[j]) \neq \boldsymbol{v}^*[j]\}\big| \le 4\big|\{j \mid |\boldsymbol{\eta}[j]| \ge 1\}\big| \le 4\|\boldsymbol{\eta}\|_2^2.$$

Finally, using the relation $\mathbf{P}\mathbf{H}_{2^n} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \end{pmatrix}$ and the orthogonality of $\mathbf{P}$, we have

$$\begin{aligned} \varepsilon(\boldsymbol{v}^{(1)}) &= \|\mathbf{A}\boldsymbol{v}^{(1)} - \boldsymbol{b}\|_2^2 \\ &\le \|\mathbf{P}\mathbf{H}_{2^n}\boldsymbol{v}^{(1)} - \mathbf{P}\boldsymbol{w}^*\|_2^2 \\ &= \|\mathbf{H}_{2^n}\boldsymbol{v}^{(1)} - \boldsymbol{w}^*\|_2^2 \\ &= 2^n\|\boldsymbol{v}^{(1)} - \boldsymbol{v}^*\|_2^2 \\ &\le 2^n \cdot 4\|\boldsymbol{\eta}\|_2^2 \end{aligned}$$

$$= 2^n \cdot 4 \cdot \frac{1}{2^n} \|\boldsymbol{\delta}\|_2^2$$
$$= 4\|\boldsymbol{\delta}\|_2^2.$$

□

Corollary 1 *If initialized with* $\boldsymbol{d}^{(1)} = \mathbf{0}$*, then by Parseval's Identity,*

$$\varepsilon(\boldsymbol{v}^{(1)}) \leq 4\big(2^{2n} - \|\boldsymbol{b}\|_2^2\big).$$

Proposition 3 and Corollary 1 estimate the error after one iteration without prior knowledge of the $\boldsymbol{v}^*$. Although the bounding techniques yield conservative estimates, the results provide valuable insights into the IWR algorithm: for fixed dimension n with solution existence, the quality of solution given by Algorithm 3 depends critically by the initial error $||\boldsymbol{\delta}||_2^2$ ($||\boldsymbol{b}||_2^2$ when $\boldsymbol{d}^{(1)} = \mathbf{0}$).

To further validate the convergence rate and solution quality of Algorithm 3, we then conducted a series of experiments as follows. First, we randomly selected 3 Boolean functions respectively in $\mathcal{B}_{13}, \mathcal{B}_{16}, \mathcal{B}_{19}$. For each function, we constructed S by randomly sampling spectral points from their Walsh spectra until $\frac{\|\mathbf{b}\|_2^2}{2^{2n}} = p$ was met, where $p \in \{0.3, 0.6, 0.9\}$. Then, we executed Algorithm 3, initializing with $\boldsymbol{d}^{(1)} = \mathbf{0}$. Let k denote the number of convergence rounds. By normalizing the parameters through division by 2^{2n}, the experimental results for $\varepsilon(\boldsymbol{v}^{(1)}), \varepsilon(\boldsymbol{v}^*)$ and $4||\boldsymbol{\delta}||_2^2$ are shown in Figure 1. Based on the experimental results, we derive the following properties of the IWR algorithm:

- It typically converges within very few iterations.
- The first iteration typically achieves the most significant error reduction (also shown in Table 1).
- After normalization, the convergence error $\frac{\varepsilon(\boldsymbol{v}^*)}{2^{2n}}$ is greatly influenced by the initial error $\frac{\|\boldsymbol{\delta}\|_2^2}{2^{2n}}$. Moreover, when the initial error is sufficiently small (e.g., instances when $p = 0.9$ in our experiments), the convergence error reaches 0, indicating that the IWR algorithm successfully identifies a solution.

Table 1 Experimental results for $n = 16$, $|S| = 31{,}955$ with IWR algorithm

Iteration rounds	$cnt(t)$	$\varepsilon(\boldsymbol{v}^{(t)})$	$\lvert S\rvert$
$t = 0$	\	4,179,318,832	31,955
$t = 1$	6,170	1,785,504	31,955
$t = 2$	6,170	1,785,504	31,955
$t = 50$	6,170	1,785,504	31,955
$t = 500$	6,170	1,785,504	31,955

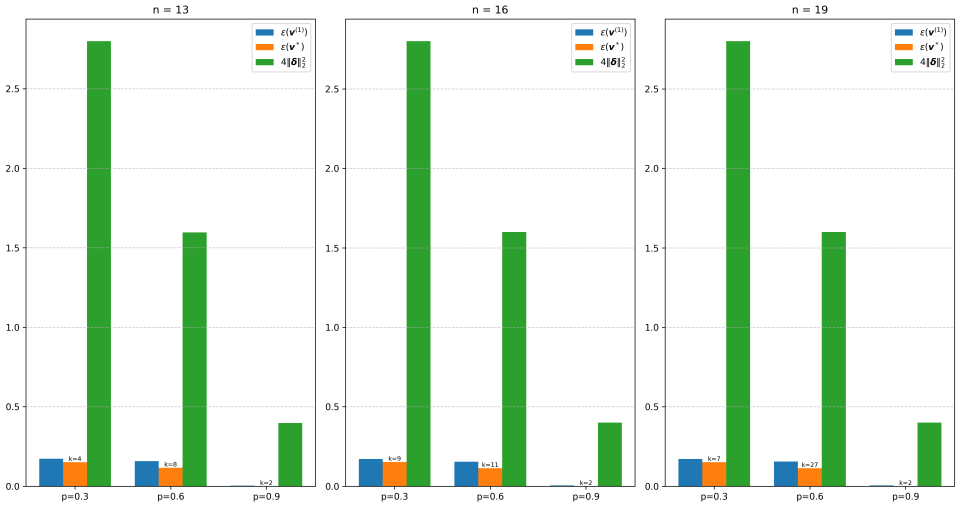


Fig. 1 Practical properties of IWR algorithm.

Table 2 Experimental results for $n = 16$, $|S| = 31{,}955$ with IWR algorithm optimized by the Forgetting Strategy

Iterations rounds	$cnt(t)$	$\varepsilon(\boldsymbol{v}^{(t)})$	$\lvert S\rvert$
$t = 100(\lambda_1 = \frac{1}{2})$	6,677	1,789,408	31,955
$t = 200(\lambda_2 = \frac{1}{3})$	0	127,948	31,955
$t = 300(\lambda_3 = \frac{1}{4})$	31,953	128	31,955
$t = 400(\lambda_4 = \frac{1}{5})$	31,953	128	31,955
$t = 500(\lambda_5 = \frac{1}{6})$	31,953	128	31,955

4 FG-IWR Algorithm: Add Two Strategies to the IWR Algorithm for Heuristic Optimization

4.1 Forgetting Strategy

Proposition 2 demonstrates that Algorithm 3 converges to local optima when $\mathbf{B}\boldsymbol{v}^{(T+1)} = \mathbf{B}\boldsymbol{v}^{(T)}$. To overcome this limitation, we develop a *Forgetting Strategy* that progressively zeros out decreasing proportions ($\lambda_1 \geq \lambda_2 \geq \cdots \to 0$) of $\mathbf{B}\boldsymbol{v}^{(T)}$ components while maintaining $\mathbf{A}\boldsymbol{v}^{(T)}$, with each modified solution serving as new input for Algorithm 3. This controlled decay process, where solution components are selectively "forgotten", ensures convergence while escaping local optima.

Significant progress is made after the addition of the *Forgetting Strategy*. Frequently, this strategy enables Algorithm 3 to output a batch of Boolean functions with significantly reduced errors compared to previous iterations, indicating they are substantially "closer" to the solutions as Table 2 shows.

4.2 Greedy Strategy

Based on the numerous "approximate solutions" provided by the Forgetting Strategy, a natural question arises: how can we leverage them to obtain the desired solutions? This is precisely what the *Greedy Strategy* aims to address.

Definition 3 Given $cnt0 \in \mathbb{N}$, define $f_0 \in \mathcal{B}_n$ as an approximate solution if $\varepsilon(\boldsymbol{v}_0) \leq cnt0$, where $\boldsymbol{v}_0 = (-1)^{\boldsymbol{f}_0}$.

Proposition 4 *Given (the truth tables) of two Boolean functions $f_1, f_2 \in \mathcal{B}_n$, if $||\hat{\boldsymbol{f}}_1 - \hat{\boldsymbol{f}}_2||_2^2 < M$, then*

$$\mathrm{wt}\left(\boldsymbol{f}_1 + \boldsymbol{f}_2\right) < \frac{M}{2^{n+2}}.$$

Proof Let $\boldsymbol{v}_1 = (-1)^{\boldsymbol{f}_1}, \boldsymbol{v}_2 = (-1)^{\boldsymbol{f}_2}$, we have

$$\hat{\boldsymbol{f}}_1 - \hat{\boldsymbol{f}}_2 = \mathbf{H}_{2^n}(\boldsymbol{v}_1 - \boldsymbol{v}_2). \tag{5}$$

Both sides of equation (5) are multiplied by their transposes, we have

$$||\hat{\boldsymbol{f}}_1 - \hat{\boldsymbol{f}}_2||_2^2 = 2^n ||\boldsymbol{v}_1 - \boldsymbol{v}_2||_2^2. \tag{6}$$

As $(-1)^{\boldsymbol{f}} = \mathbf{1} - 2\boldsymbol{f}$, we have

$$|\{i \in \{0, 1, 2, ..., 2^n - 1\} | \boldsymbol{v}_1[i] \neq \boldsymbol{v}_2[i]\}| = \mathrm{wt}\left(\boldsymbol{f}_1 + \boldsymbol{f}_2\right).$$

For $i \in \{0, 1, 2, ..., 2^n - 1\}$,

$$(\boldsymbol{v}_1[i] - \boldsymbol{v}_2[i])^2 = \begin{cases} 0, \boldsymbol{v}_1[i] = \boldsymbol{v}_2[i] \\ 4, \boldsymbol{v}_1[i] \neq \boldsymbol{v}_2[i] \end{cases}.$$

Therefore, we have

$$||\boldsymbol{v}_1 - \boldsymbol{v}_2||_2^2 = 4\mathrm{wt}\left(\boldsymbol{f}_1 + \boldsymbol{f}_2\right). \tag{7}$$

From equations (6) and (7), we obtain

$$\mathrm{wt}\left(\boldsymbol{f}_1 + \boldsymbol{f}_2\right) = \frac{||\hat{\boldsymbol{f}}_1 - \hat{\boldsymbol{f}}_2||_2^2}{2^{n+2}} < \frac{M}{2^{n+2}}.$$

□

Corollary 2 *For any approximate solution $f_0 \in \mathcal{B}_n$, if there exists a solution f^* satisfying $||\hat{\boldsymbol{f}}_0 - \hat{\boldsymbol{f}}^*||_2^2 \leq k\varepsilon(\boldsymbol{v}_0)$, where $k \in \mathbb{N}, \boldsymbol{v}_0 = (-1)^{\boldsymbol{f}_0}$, we have:*

$$\mathrm{wt}(\boldsymbol{f}_0 + \boldsymbol{f}^*) < \frac{k \cdot cnt0}{2^{(n+2)}}.$$

Proposition 4 and Corollary 2 reveal that when solutions to **Problem** exist and the approximate solutions are "close" enough to the solutions in their Walsh spectra, their truth tables are necessarily similar. Motivated by this insight, we try to obtain the desired solutions through the truth tables of the approximate solutions and the positions where the values differ between these truth tables. We define these special positions as:

Definition 4 Given all the approximate solutions $f_1, f_2, ..., f_t$, define

$$P \triangleq \bigcup_{1 \le k_1 < k_2 \le t} \left\{ x_i \mid \boldsymbol{f}_{k_1}[x_i] \neq \boldsymbol{f}_{k_2}[x_i] \right\}.$$

Proposition 5 *Given all approximate solutions $f_1, f_2, \ldots, f_t$ and the set P, we have:*

$$P = \bigcup_{1 \le k \le t} \{ x_i \mid \boldsymbol{f}_1[x_i] \neq \boldsymbol{f}_k[x_i] \}$$

Proof Let $P' = \bigcup_{1 \le k \le t} \{ x_i \mid \boldsymbol{f}_1[x_i] \neq \boldsymbol{f}_k[x_i] \}$. By Definition 4, it is clear that $P' \subseteq P$.

$\forall a \in P$, by Definition 4, there exist k_1, k_2 with $1 \le k_1 < k_2 \le t$ such that $\boldsymbol{f}_{k_1}[a] \neq \boldsymbol{f}_{k_2}[a]$.

- If $\boldsymbol{f}_1[a] \neq \boldsymbol{f}_{k_1}[a]$, then by the definition of P', we have $a \in P'$.
- If $\boldsymbol{f}_1[a] = \boldsymbol{f}_{k_1}[a]$, then since $\boldsymbol{f}_{k_1}[a] \neq \boldsymbol{f}_{k_2}[a]$, it follows that $\boldsymbol{f}_1[a] \neq \boldsymbol{f}_{k_2}[a]$, and thus $a \in P'$.

Therefore, $P \subseteq P'$, which implies $P = P'$. □

We use set P to record the differences in the truth tables between the approximate solutions. In practical implementation, Proposition 5 demonstrates that a single traversal of the approximate solutions suffices to derive the set P, thus greatly reducing the computational complexity when a large number of approximate solutions are involved.

Figure 2 shows the main concept of the *Greedy Strategy*. Let f_0 be a solution and f_1, f_2 be approximate solutions. We argue that the transformation from $\boldsymbol{f}_1$ to $\boldsymbol{f}_2$ can be interpreted as first obtaining $\boldsymbol{f}_0$ and then flipping some bits to reach $\boldsymbol{f}_2$. Therefore, when the approximate solutions are adequate, $\{x_i | \boldsymbol{f}_0[x_i] \neq \boldsymbol{f}_1[x_i]\} \subseteq P$. Thus, based on any approximate solution f_1, we enumerate all subsets P' of P, flip the bits of $\boldsymbol{f}_1$ at positions in P' (i.e., $\boldsymbol{f}'_1[a] \Leftarrow \boldsymbol{f}_1[a] \oplus 1$, $a \in P'$), and then individually check whether $\boldsymbol{f}'_1$ is a solution to S using the FWT. The time complexity of this verification process is $O(n \cdot 2^{n+|P|})$.

The core of the aforementioned *Greedy Strategy* lies in determining the set P. According to Definition 4, P is uniquely determined by the complete set of approximate solutions $\{f_1, f_2, \ldots, f_t\}$. Consequently, by Definition 1, the selection of the parameter $cnt0$ fully specifies P.

In practice, the choice of the set P—equivalently, the choice of $cnt0$—involves a trade-off between solution completeness and computational efficiency. Without considering specially constructed the set P, a larger $cnt0$ leads to a larger $|P|$. This, in turn, allows the algorithm to find more solutions satisfying S, but at the cost of an exponential increase in computational complexity. In the extreme case where $cnt0 = 2^{2n}$, we have $|P| = 2^n$, and the greedy algorithm essentially reduces to an exhaustive search over $\mathcal{B}_n$.

If the set S carries sufficient information, the condition $\|\hat{\boldsymbol{f}}_0 - \hat{\boldsymbol{f}}^*\|_2^2 \le k\varepsilon(\boldsymbol{v}_0)$ in Corollary 2 is often satisfied. Accordingly, we suggest $cnt0 = 2^{(n+2)+l}$ so that $\mathrm{wt}(\boldsymbol{f}_1 - \boldsymbol{f}^*) < k2^l$, with $l \in \{0, 1, 2, 3, 4\}$. The value of l is adaptive: increase l if no

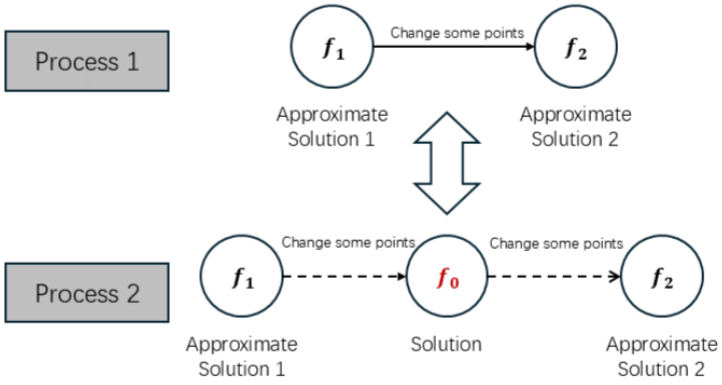


Fig. 2 The main concept of Greedy Strategy

approximate solution is found after multiple attempts; decrease it if the size $|P|$ grows beyond computationally tractable limits.

In summary, the *Forgetting Strategy* contributes to obtaining sufficient approximate solutions and the *Greedy Strategy* converts them into solutions. After adding these to the IWR algorithm, we presented the *FG-IWR* algorithm, that is, Algorithm 4. The total time complexity is $O(|T| \cdot r \cdot n \cdot 2^n + n \cdot 2^{n+|P|})$, where $|T|, r, |P|$ are constants.

Algorithm 4 FG-IWR algorithm

Input: $S = \{(a_i, b_i), 1 \le i \le m\}$ —Part-Walsh Spectrum
T —Numbers of Points to Forget, Sorted in Descending Order
r —Maximum Number of Iteration Rounds for the same $t \in T$
$cnt0$ —Upper Bound for Approximate Solutions
Output: C —Set of solutions

Initialization: $\boldsymbol{w}^{(1)} \Leftarrow \mathbf{0}, P \Leftarrow \emptyset, flag \Leftarrow 0$
$\boldsymbol{w}^{(1)}[a_k] \Leftarrow b_k$ for (a_k, b_k) in S
for $t \in T$ **do**
  for $i = 1, 2, ..., r$ **do**
    $\tilde{\boldsymbol{v}}^{(i)} \Leftarrow Walsh_Polar(\boldsymbol{w}^{(i)})$
    $\boldsymbol{v}^{(i)} \Leftarrow sgn(\boldsymbol{v}^{(i)})$ ▷ Correct $\boldsymbol{v}$ to a right form
    $\boldsymbol{w}^{(i+1)} \Leftarrow Polar_Walsh(\boldsymbol{v}^{(i)})$
    if $\varepsilon(\boldsymbol{v}^{(i)}) \le cnt0$ **then** ▷ One approximate solution is found
      if $flag = 0$ **then** $\boldsymbol{v}_0 \Leftarrow \boldsymbol{v}^{(i)}, flag \Leftarrow 1$ ▷ Polar table of the first approximate solution
      end if
      if $flag = 1$ **then** $P \Leftarrow P \cup \{a | \boldsymbol{v}^{(i)}[a] \ne \boldsymbol{v}_0[a]\}$
      end if
    end if
    if $\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)}$ **then**
      Randomly generate $s_1, s_2, ..., s_t \in \{0, 1, 2, ..., 2^n - 1\} \backslash \{a_i | 1 \le i \le m\}$
      for $k = 1, 2, ..., t$ **do** $\boldsymbol{w}^{(i+1)}[s_k] \Leftarrow 0$
      end for
    end if
    for $j = 1, 2, ..., m$ **do** $\boldsymbol{w}^{(i+1)}[a_j] = b_j$ ▷ Correct $\boldsymbol{w}$ with S
    end for
  end for
end for
$C \Leftarrow \emptyset$
for $P' \subseteq P$ **do**
  $\boldsymbol{v}'_0 \Leftarrow \boldsymbol{v}_0$
  for $p \in P'$ **do** $\boldsymbol{v}'_0[p] \Leftarrow -\boldsymbol{v}_0[p]$
  end for
  if $Polar_Walsh(\boldsymbol{v}'_0)[a_k] = b_k$ for (a_k, b_k) in S **then** $C \Leftarrow C \cup \{\frac{1-\boldsymbol{v}'_0}{2}\}$
  end if
end for
return C

Notes: The set $T = \{t_1, t_2, \ldots\}$ must satisfy $2^n - m > t_1 > t_2 > \cdots > 0$ with $\lim_{i \to \infty} t_i = 0$. We recommend constructing T as: $T = \{(2^n - m)\lambda_1, (2^n - m)\lambda_2, \ldots\}$ where $\lambda_i = f(i)$, and $f(i)$ is a monotonically decreasing function defined on $(0, +\infty)$ with range containing $(0, 1]$. A typical example is $f(x) = \frac{1}{(x+1)}$.

Table 3 Overview of Our Self-constructed Instances

n	Problem Name	$\|S\|$	p	N_f	NL_{max}	$\frac{N_f}{NL_{max}}$
13	13.1	4405	0.959	3874	4064	95.3%
	13.2	4701	0.917	3848	4064	94.7%
14	14.1	7986	0.973	7842	8128	96.5%
	14.2	7709	0.941	7854	8128	96.6%
15	15.1	17609	0.954	15784	16320	96.7%
	15.2	18713	0.925	15854	16320	97.1%
16	16.1	31955	0.973	31964	32640	97.9%
	16.2	30756	0.942	31918	32640	97.8%
17	17.1	58991	0.902	64372	65408	98.4%
	17.2	55716	0.849	64418	65408	98.5%
18	18.1	123044	0.939	129440	130816	98.9%
	18.2	119757	0.915	129464	130816	99.0%
19	19.1	281177	0.957	259952	261888	99.3%
	19.2	363009	0.919	259902	261888	99.2%

Next, we use a series of self-constructed instances to verify the solution effect of *FG-IWR* algorithm. From the perspective of experimental efficiency, we varied n from 13 to 19. The instances were generated as follows. For each n, we randomly generated a large number of Boolean functions and retained the two functions with the highest nonlinearity. We then computed their Walsh spectra and iteratively selected spectral values at random until the parameter $p = \frac{||\boldsymbol{b}||_2^2}{2^{2n}}$ met a predefined threshold. This procedure yielded 14 instances of the **Problem**, with detailed information presented in Table 3. The complete dataset is available in [37].

We then executed the *FG-IWR* algorithm on these instances. Parameter configurations and experimental outcomes are presented in Table 4. Results show that Algorithm 4 achieves rapid convergence and produces expected outcomes in these test instances, but the time complexity of verification step exhibits is prohibitive when $|P|$ is quite large. Nevertheless, the solutions required are scarce in the search space (e.g., for the 16.1 problem, only 11 solutions exist in 2^{28} searches, with a ratio of 4.1×10^{-8}). Therefore, the exhaustion and verification process has considerable room for optimization.

Table 4 Performance of FG-IWR Algorithm with Varied Parameters

Problem Name	Algorithm Parameters			$\|P\|$	Execution Time of Algorithm 4	Time Complexity of the Verification Step
	$cnt0(2^{n+5})$	$\|T\|$	r			
13.1	2^{18}	10	100	7	5s	$O(2^{23.7})$
13.2	2^{18}	10	100	8	6s	$O(2^{24.7})$
14.1	2^{19}	10	100	24	3s	$O(2^{41.8})$
14.2	2^{19}	10	100	28	3s	$O(2^{45.8})$
15.1	2^{20}	10	200	8	58s	$O(2^{26.9})$
15.2	2^{20}	10	200	20	15s	$O(2^{38.9})$
16.1	2^{21}	10	300	28	45s	$O(2^{48.0})$
16.2	2^{21}	10	200	26	29s	$O(2^{27.3})$
17.1	2^{22}	10	300	29	94s	$O(2^{50.1})$
17.2	2^{22}	10	300	133	97s	$O(2^{154.1})$
18.1	2^{23}	15	500	30	159s	$O(2^{52.2})$
18.2	2^{23}	15	500	76	128s	$O(2^{98.2})$
19.1	2^{24}	15	500	8	278s	$O(2^{31.2})$
19.2	2^{24}	15	500	36	270s	$O(2^{59.2})$

5 OIWR Algorithm: Optimizing with a necessary condition

We considered compressing the search space under a certain necessary condition before verifying it using the FWT. Definition 5 to Example 1 describe the necessary condition.

Definition 5 Given $\boldsymbol{f}$ the truth table of an arbitrary Boolean function $f \in \mathcal{B}_n$, define

$$\mathbf{H}_{2^n}^{\boldsymbol{f}} \triangleq \{(-1)^{\alpha\cdot\beta\oplus\boldsymbol{f}(\beta)}\}_{(\alpha,\beta\in GF(2)^n)}.$$

Denote the element at i-th row and j-th column of $\mathbf{H}_{2^n}^{\boldsymbol{f}}$ as $\mathbf{H}_{2^n}^{\boldsymbol{f}}[i,j]$, and the submatrix composed of $\mathbf{H}_{2^n}^{\boldsymbol{f}}[a,b], a \in A, b \in B$ as $\mathbf{H}_{2^n}^{\boldsymbol{f}}[A,B]$.

The matrix $\mathbf{H}_{2^n}^{\boldsymbol{f}}$ is actually a decomposition of $\hat{\boldsymbol{f}}$, which implies that $\sum_{j=0}^{2^n-1} \mathbf{H}_{2^n}^{\boldsymbol{f}}[i,j] = \hat{\boldsymbol{f}}[i]$.

Lemma 2 *Given two Boolean functions $f_1, f_2 \in \mathcal{B}_n$, define the set $J = \{j | \boldsymbol{f}_1[j] \neq \boldsymbol{f}_2[j]\}$, then we have*

$$\hat{\boldsymbol{f}}_1[i] = \hat{\boldsymbol{f}}_2[i] + \sum_{j \in J} [(-2) \cdot (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2(j)}], 0 \leq i \leq 2^n - 1,$$

where $\boldsymbol{i}$ and $\boldsymbol{j}$ denote the n-bit binary vector representations of i and j, respectively.

Proof For any $j \in J$, the value of $\boldsymbol{f}_1[j]$ differs from $\boldsymbol{f}_2[j]$. We analyze the contribution of each such j to the difference $\hat{\boldsymbol{f}}_1[i] - \hat{\boldsymbol{f}}_2[i]$.

Case 1: If $(-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j]} = -1$, then flipping $\boldsymbol{f}_2[j]$ to $\boldsymbol{f}_1[j]$ changes the sign:

$$(-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_1[j]} = (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j] \oplus 1} = 1.$$

Thus, the difference in the Walsh spectrum due to j is:

$$\hat{\boldsymbol{f}}_1[i] - \hat{\boldsymbol{f}}_2[i] = 1 - (-1) = 2 = -2 \cdot (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j]}.$$

Case 2: If $(-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j]} = 1$, then flipping $\boldsymbol{f}_2[j]$ to $\boldsymbol{f}_1[j]$ gives:

$$(-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_1[j]} = (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j] \oplus 1} = -1.$$

The difference in the Walsh spectrum due to j is:

$$\hat{\boldsymbol{f}}_1[i] - \hat{\boldsymbol{f}}_2[i] = -1 - 1 = -2 = -2 \cdot (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j]}.$$

Since the contributions from all $j \in J$ are additive, the total difference between $\hat{\boldsymbol{f}}_1[i]$ and $\hat{\boldsymbol{f}}_2[i]$ is:

$$\hat{\boldsymbol{f}}_1[i] = \hat{\boldsymbol{f}}_2[i] + \sum_{j \in J} \left[(-2) \cdot (-1)^{\boldsymbol{i} \cdot \boldsymbol{j} \oplus \boldsymbol{f}_2[j]} \right].$$

□

Given any Boolean functions $f_1, f_2 \in \mathcal{B}_n$, define

$$X = \{x_i | \hat{\boldsymbol{f}}_1[x_i] = \hat{\boldsymbol{f}}_2[x_i]\} \triangleq \{x_1, x_2, ..., x_m\},$$

$$P_1 = \{y_{k_i} | \boldsymbol{f}_1[y_{k_i}] \neq \boldsymbol{f}_2[y_{k_i}]\} \triangleq \{y_{k_1}, y_{k_2}, ..., y_{k_t}\}.$$

Based on these definitions, we propose Proposition 6.

Proposition 6 (Necessary Condition) *For any set $X_1 = \{x_{k_1}, x_{k_2}, ..., x_{k_l}\} \subset X$, s.t.*

$$\hat{\boldsymbol{f}}_1[x_{k_i}] = \hat{\boldsymbol{f}}_2[x_{k_i}], i = 1, 2, ..., l,$$

the sum of each row in $\mathbf{H}_{2^n}^{\boldsymbol{f}_2}[X_1, P_1]$ equals 0.

Proof According to Lemma 2, for any $x_{k_i} \in X_1$,

$$\begin{aligned} \hat{\boldsymbol{f}}_1[x_{k_i}] - \hat{\boldsymbol{f}}_2[x_{k_i}] &= \sum_{j=1}^{t} (-2) \cdot (-1)^{\boldsymbol{x_{k_i}} \cdot \boldsymbol{y_{k_j}} \oplus \boldsymbol{f}_2[y_{k_j}]} \\ &= \sum_{j=1}^{t} (-2\mathbf{H}_{2^n}^{\boldsymbol{f}_2}[x_{k_i}, y_{k_j}]). \end{aligned}$$

So, the sum of each row of $-2\mathbf{H}_{2^n}^{\boldsymbol{f}_2}[X_1, P_1]$ is 0. Thus, the sum of each row of $\mathbf{H}_{2^n}^{\boldsymbol{f}_2}[X_1, P_1]$ is 0. □

To illustrate Proposition 6 clearly, we give Example 1.

Example 1 For $f_1, f_2 \in \mathcal{B}_2$, we have $\boldsymbol{f}_1 = (0,0,1,1), \hat{\boldsymbol{f}}_1 = (0,0,4,0), \boldsymbol{f}_2 = (0,1,1,0), \hat{\boldsymbol{f}}_{\boldsymbol{2}} = (0,0,0,4)$. So,

$$\mathbf{H}_{2^n}^{\boldsymbol{f}_2} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

According to Proposition 6, $X = \{0,1\}, P_1 = \{1,3\}$, then, let $X_1 = X$, we have:

$$\mathbf{H}_{2^n}^{\boldsymbol{f}_1}[X_1, P_1] = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}.$$

It is clear that the sum of each row of $\mathbf{H}_{2^n}^{\boldsymbol{f}_1}[X_1, P_1]$ is 0.

In Proposition 6, we set f_2 as an approximate solution found in Algorithm 4 and f_1 as the required solution. Then, X is the set of positions in the part-Walsh spectrum S, and the matrix $\mathbf{H}_{2^n}^{\boldsymbol{f}_2}$ can be obtained through preprocessing. Set X_1 can be selected flexibly based on the computational power. Furthermore, if P_1 is determined, the solution f_1 can be easily determined from f_2, and P_1 and f_1 are in one-to-one correspondence when f_2 is fixed. Therefore, our goal is shifted to finding all P_1 that satisfy Proposition 6 first and verifying them with S using the FWT algorithm.

The process of identifying P_1 can be transformed into solving a constraints-only *0-1 ILP (Integer Linear Programming)* problem [38]:

$$\mathbf{H}_{2^n}^{\boldsymbol{f}_1}[X_1, P] \cdot \boldsymbol{Y}^T = \mathbf{0}, \boldsymbol{Y} \in GF(2)^{|P|} \text{is the variable.} \tag{8}$$

Let $\mathcal{Y}$ denote the solution set of equation system (8). An injective mapping from $\mathcal{Y}$ to the power set of P can be naturally constructed: each vector $\boldsymbol{y} \in \mathcal{Y}$ is mapped to the set $\{P[i] \mid y[i] \neq 0,\ i = 0, 1, \ldots, |P| - 1\}$, where $P[i]$ represents the i-th element of the set P under an arbitrary but fixed ordering. Denote the image of this injection by Δ. Subsequently, by flipping the values of an arbitrary approximate solution f_2 at the positions in each subset $P_1 \in \Delta$, we generate candidate functions and then verify each of them using the FWT. This process yields all solutions for the set S.

In practical implementation, since (1) the optimization problem (8) can be solved efficiently using *Gurobi Optimizer* [39] (in our experiments, for cases where $|P| \leq 500, |X_1| \geq 100$, Gurobi typically solves (8) in less than 100 seconds), (2) the time complexity of solving (8) is adaptively controllable by adjusting the parameters X_1 and P, and (3) the number of solutions of (8) admits an upper bound under Gurobi's branch-and-cut framework, the complexity of the verification step is much lower than the process of finding P. Let $\eta = 1 - \frac{num_1}{num_2}$, where num_1 denotes the number of approximate solutions satisfying Proposition 6, and $num_2 = 2^{|P|}$, which is the size of verification space. In experiments, we set X_1 as the 500 points in S with maximal $|b_i|, i \in \{1, 2, ..., m\}$ (this strategy was inspired by the results in Appendix A), and applied the optimization method described in this section to the aforementioned self-constructed instances. Results are shown in Table 5. All the solutions and their corresponding sets S are available in [37].

Table 5 The Solving Performance after Optimizing with the Necessary Condition

Problem Name	num_1	η	Number of solutions	Time of Running Gurobi	Total Time*
13.1	1	99.2188%	1	<1s	11s
13.2	1	99.9985%	1	<1s	11s
14.1	4096	99.9756%	10	<1s	10s
14.2	16384	99.9939%	30	2s	23s
15.1	1	99.6094%	1	<1s	62s
15.2	1	99.9999%	1	<1s	18s
16.1	16384	99.9939%	11	3s	90s
16.2	4096	99.9756%	15	3s	72s
17.1	16384	99.9969%	119	5s	231s
17.2	4096	99.9999%	60	21s	136s
18.1	16384	99.9985%	22	11s	359s
18.2	1024	99.9023%	2	15s	153s
19.1	1	99.6094%	1	<1s	287s
19.2	1	99.9999%	1	<1s	274s

* Total time represents the sum of runtime under our experimental environment (see Section 7), including data processing, Algorithm 5 execution, and associated computational overhead.

In summary, the necessary condition in Proposition 6 meets our expectations for the following reasons:

1. If a Boolean function does not satisfy Proposition 6, it must not be a solution.
2. The time complexity of verifying the necessary condition with Gurobi is low.
3. With this necessary condition, we can screen out most of wrong conditions. In our experiments, η is usually above 99.9%.

Thus far, we have presented the complete OIWR algorithm shown in Algorithm 5. Its time complexity is $O((|T| \cdot r + |\Delta|) \cdot n \cdot 2^n)$, which is approximate to the time complexity of verifying the **Problem**. Experimental results on our self-constructed instances (see in Table 5) demonstrate the algorithm's capability to generate a batch of Boolean functions with high nonlinearity.

Algorithm 5 The Complete OIWR Algorithm

Input: $S = \{(a_i, b_i), 1 \le i \le m\}$ —Part-Walsh Spectrum
T —Numbers of Points to Forget, Sorted in Descending Order
r —Maximum Number of Iteration Rounds for the same $t \in T$
$cnt0$ —Upper Bound for Approximate Solutions
Output: C —Set of solutions
Function Name: OIWR()

Initialization: $\boldsymbol{w}^{(1)} \Leftarrow \mathbf{0}, P \Leftarrow \emptyset, flag \Leftarrow 0$
$\boldsymbol{w}^{(1)}[a_k] \Leftarrow b_k$ for (a_k, b_k) in S
for $t \in T$ **do**
  for $i = 1, 2, ..., r$ **do**
    $\tilde{\boldsymbol{v}}^{(i)} \Leftarrow Walsh_Polar(\boldsymbol{w}^{(i)})$
    $\boldsymbol{v}^{(i)} \Leftarrow sgn(\boldsymbol{v}^{(i)})$ ▷ Correct $\boldsymbol{v}$ to a right form
    $\boldsymbol{w}^{(i+1)} \Leftarrow Polar_Walsh(\boldsymbol{v}^{(i)})$
    if $\varepsilon(\boldsymbol{v}^{(i)}) \le cnt0$ **then** ▷ One approximate solution is found
      if $flag = 0$ **then** $\boldsymbol{v}_0 \Leftarrow \boldsymbol{v}^{(i)}, flag \Leftarrow 1$ ▷ Polar table of the first approximate solution
      end if
      if $flag = 1$ **then** $P \Leftarrow P \cup \{a | \boldsymbol{v}^{(i)}[a] \ne \boldsymbol{v}_0[a]\}$
      end if
    end if
    if $\boldsymbol{w}^{(i+1)} = \boldsymbol{w}^{(i)}$ **then**
      Randomly generate $s_1, s_2, ..., s_t \in \{0, 1, 2, ..., 2^n - 1\} \backslash \{a_i | 1 \le i \le m\}$
      for $k = 1, 2, ..., t$ **do** $\boldsymbol{w}^{(i+1)}[s_k] \Leftarrow 0$
      end for
    end if
    for $j = 1, 2, ..., m$ **do** $\boldsymbol{w}^{(i+1)}[a_j] = b_j$ ▷ Correct $\boldsymbol{w}$ with S
    end for
  end for
end for
$C \Leftarrow \emptyset$
Construct $\mathbf{H}_{2^n}^{\boldsymbol{f}_0}$ using $\boldsymbol{v}_0$ according to Definition 5
Select the set $X_1 \subseteq \{a_1, a_2, ..., a_m\}$ according to the construction requirements
Solve $\mathbf{H}_{2^n}^{\boldsymbol{f}_1}[X_1, P] \cdot \boldsymbol{Y}^{\mathsf{T}} = \mathbf{0}$, where $\boldsymbol{Y} \in GF(2)^{|P|}$, and denote the solution set by $\mathcal{Y}$.
$\Delta \Leftarrow \emptyset$
for $y \in \mathcal{Y}$ **do** $\Delta \cup \{P[i] \mid y[i] \ne 0,\ i \in \{0, 1, \ldots, |P| - 1\}\}$
end for
for $P_1 \in \Delta$ **do**
  for $p \in P_1$ **do** $\boldsymbol{v}_0'[p] \Leftarrow -\boldsymbol{v}_0[p]$
  end for
  if $Polar_Walsh(\boldsymbol{v}_0')[a_k] = b_k$ for (a_k, b_k) in S **then** $C \Leftarrow C \cup \{\frac{1 - \boldsymbol{v}_0'}{2}\}$
  end if
end for
return C

6 Application of OIWR for Constructing Low Hamming Weight Correlation Immune Functions

The concept of correlation immune functions was first introduced by Siegenthaler (1984) [3] in the context of stream cipher design within symmetric cryptography, with further theoretical developments provided by Camion et al. at CRYPTO'91 [40]. Since their inception, the construction of such functions has attracted significant research attention. In recent years, correlation immune Boolean functions have demonstrated novel applications in countering side-channel attacks (SCAs). These attacks target implementations of block ciphers in embedded systems like smart cards, FPGA (stands for Field Programmable Gate Arrays), and ASIC (stands for Application-Specific Integrated Circuit). Unlike traditional cryptanalytic methods, SCAs employ distinct attack models and exhibit remarkable practical effectiveness. Carlet in his book[18] discussed physical attacks and related problems on functions, and explained in Section 12.1 (p. 460) that low Hamming weight correlation immune functions are particularly critical for SCA protection. Despite their importance, there is still a lack of research on the construction of such functions. Notably, Mesnager and Su's 2021 work [5] is considered as a pivotal contribution to this emerging research direction. They employed an algebraic approach to construct specialized support matrices, enabling the construction of n-variable, dth-order correlation immune functions with weights of 2^m.While effective, this approach carries two inherent limitations: (1) The deterministic nature of their constructing process makes it difficult to efficiently generate multiple Boolean functions with identical parameters; (2) The construction cannot produce Boolean functions with Hamming weights that are not the form of power of 2, a limitation that becomes particularly pronounced when n and m take large values.

Building upon the Walsh-spectral definition of correlation immune functions and Hamming weight (see in Section 2), we observe that by adding the necessary spectral points to the set S, the construction of correlation immune functions with low Hamming weight can be fully reduced to a class of instances of the **Problem**. However, since these necessary spectral points typically constitute only a small fraction of the complete Walsh spectrum, only adding them to the set S and executing the OIWR algorithm would result in excessive initial error, preventing convergence to an approximate solution. Therefore, we consider using the existing low Hamming weight correlation immune function to construct a set S by continuously selecting its Walsh spectral points until an approximate solution can be obtained. We formally summarize this construction framework as follows:

1. Select an n-variable, low-Hamming-weight, d-th order correlation immune function f and compute its Walsh spectrum $\hat{\boldsymbol{f}}$ via FWT (see Algorithm 1).
2. Construct the set S by strategically selecting spectral points from $\hat{\boldsymbol{f}}$.
3. Execute the OIWR algorithm for S. If no approximate solution is obtained, return to Step 2).

In this solving framework, different construction strategies for the set S significantly affect the properties of the final solutions. As a concrete application of this

framework, we selected a 12-variable, 1st-order correlation immune function with Hamming weight 256 (2^8), constructed using the Mesnager-Su method.

6.1 Construction of multiple 1st-order correlation immune functions with 12 variables and weight 256

Algorithm 6 Construction of f_1

Output: Supp(f_1) —Support Matrix of f_1
Function Name: GetF1()

1: **for** $i = 0, 1, ..., 2^8 - 1$ **do**
2: binary $\leftarrow$ binary representation of i with length 8
3: $\mathcal{M}_8[i, :] \leftarrow$ binary ▷ Generate all vectors in $GF(2)^8$
4: **end for**
5: $\mathbf{E} \leftarrow \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \\ 1&1&0&0 \\ 0&1&1&0 \\ 0&0&1&1 \\ 1&0&0&1 \end{bmatrix}$ ▷ Extension matrix, satisfying the condition of relation (6) in [5]
6: $\mathbf{E}' \leftarrow \mathcal{M}_8 \cdot \mathbf{E} \mod 2$
7: Supp(f_1) $\leftarrow [\mathcal{M}_8, \mathbf{E}']$ ▷ Matrix concatenation
8: **return** Supp(f_1)

First, we construct a 1st-order correlation immune function f_1 with 12 variables and weight 256 using Mesnager and Su's method, as described in Algorithm 6. The Walsh spectrum of f_1 takes only 3 distinct values: $|\{w \mid \hat{\boldsymbol{f}}_1(w) = 3584\}| = 1$, $|\{w \mid \hat{\boldsymbol{f}}_1(w) = -512\}| = 15$ and $|\{w \mid \hat{\boldsymbol{f}}_1(w) = 0\}| = 4080$. This demonstrates that the non-zero values in $\hat{\boldsymbol{f}}_1$ exhibit extremely sparse distribution, which is a direct consequence of the linear algebraic construction method employed.

Next, we construct the part-Walsh spectrum set S_1, the process is:

1. Initialize $S_1 \leftarrow \{(0, 3584)\}$;
2. $\forall \boldsymbol{w} \in GF(2)^n$ with $\mathrm{wt}(\boldsymbol{w}) = 1$, insert $(w, 0)$ into S_1;
3. While $\|\boldsymbol{b}\|_2^2 \leq 2^{2n}\theta$, insert $(i, \hat{\boldsymbol{f}}_1(i))$ for $i = 0, 1, ..., 2^n - 1$ and $\hat{\boldsymbol{f}}_1(i) \neq 0$ (parameters: $n = 6$, $\theta = 0.8$);

Additionally, in our experiments, we observed that our baseline algorithm typically converges at the first iteration round for problems exhibiting highly concentrated Walsh spectral distributions. In this case, to reduce the error introduced by a single

iteration, we modified the $sgn()$ function in Definition 2 as follows:

$$sgn(h(i)) = \begin{cases} -1, h(i) \leq -0.3 \\ a, -0.3 < h(i) \leq 0 \\ b, 0 \leq h(i) < 0.3 \\ 1, h(i) \geq 0.3 \end{cases}, 0 \leq i \leq l, \tag{9}$$
$$P\{a = -1\} = P\{b = 1\} = \frac{2}{3},$$
$$P\{a = 1\} = P\{b = -1\} = \frac{1}{3}.$$

Next, we use the OIWR algorithm to solve S_1. The process is almost the same as the test example in Section 4. The only difference is that when using Gurobi for solving and subsequent verification, only the positions necessary for the 1st-order correlation immunity, namely $X_1 = \{w|\text{wt}(\boldsymbol{w}) = 1, \boldsymbol{w} \in GF(2)^n\}$, $S_1' = \{(w, 0)|\text{wt}(\boldsymbol{w}) = 1, \boldsymbol{w} \in GF(2)^n\}$. The algorithm parameters are set to $r = 1000, cnt0 = 2^{17}, |T| = 15$. In this experiment, 52339 1st-order correlation immune functions with 12 variables and a weight of 256 were obtained in less than 400s. The truth tables of these Boolean functions are available in [41].

6.2 Construction of 1st-Order Correlation Immune Functions in 12 Variables whose Weight is no longer limited to the power of 2

Algorithm 7 Construction of f_2

Output: Supp(f_2)—Support Matrix of f_2

Supp(f_2) ← GETF1() ▷ 256×12 matrix
Randomly select 128 indices $I \subset \{1, ..., 256\}$
for $i \in I$ **do**
    Supp(f_2)$[i, 12]$ ← Supp(f_2)$[i, 12] \oplus 1$ ▷ Flip a part of elements in the last column
end for
$c \leftarrow \sum_{i=1}^{256}$ Supp(f_2)$[i, 12]$
if $c \neq 128$ **then** ▷ Revise the last column to maintain correlation immunity
    Select $|128 - c|$ positions to set K where Supp(f_2)$[k, 12] = \begin{cases} 0, \text{if } c > 128 \\ 1, \text{if } c < 128 \end{cases}$
    for $k \in K$ **do**
        Supp(f_2)$[k, 12]$ ← Supp(f_2)$[k, 12] \oplus 1$
    end for
end if
return Supp(f_2)

Table 6 Value of $\hat{\boldsymbol{f}_2}$

Value	Count	Value	Count
0	2232	3584	1
88	16	-88	24
56	32	-56	56
48	80	-48	72
40	72	-40	40
32	128	-32	152
24	152	-24	192
16	168	-16	184
8	168	-8	248
64	32	-96	8
-72	24	-80	8
-512	7		

First, building upon Algorithm 6, we introduce a random flipping method in the last column of the support matrix to develop Algorithm 7. This enhanced algorithm preserves the original construction constraints while incorporating nonlinear operations, thereby achieving improved uniformity in the Walsh spectral distribution for the constructed functions. Let $f_2 \in \mathcal{B}_{12}$ denote the resulting 1st-order correlation immune Boolean function with Hamming weight 256, constructed via Algorithm 7. Table 6 summarizes the statistics of $\hat{\boldsymbol{f}}_2$. Building upon f_2, we construct the set S_2 through the following procedure:

1. Initialize $S_2 \leftarrow \{(0, 3584)\}$;
2. $\forall \boldsymbol{w} \in GF(2)^n$ with $\mathrm{wt}(\boldsymbol{w}) = 1$, insert $(w, 0)$ into S_2;
3. Insert $(i, \hat{\boldsymbol{f}}_{\boldsymbol{2}}(i))$ for $i = 0, 1, ..., 2^n - 1$ and $|\hat{\boldsymbol{f}}_{\boldsymbol{2}}(i)| \leq 80$ into S_2.

Next, we apply the OIWR algorithm to solve S_2, following the same procedure described in Section 6.1. The computation yields 141,893 low-weight 1st-order correlation immune functions within 1000 seconds. Their weight distribution is presented in Table 7. The results demonstrate the successful construction of 141,892 new 1st-order correlation immune functions with non-power-of-two weights (i.e., weights distinct from $2^8 = 256$). The truth tables of these Boolean functions are available in [41].

In summary, for the aforementioned specific case, our OIWR-based construction method for correlation immune functions with low Hamming weight achieves two key improvements over the Mesnager-Su method: (1) generating numerous Boolean functions satisfying the original constraints, and (2) overcoming the power-of-two weight restriction to produce abundant correlation immune functions with low Hamming

Table 7 Weight of new constructed 1st-order correlation immune functions

Weight	Count	Weight	Count
222	40	232	19,943
226	1,257	236	28,852
230	10,947	228	4,246
234	27,668	224	258
238	22,914	240	14,717
242	6,478	220	5
244	3,352	246	717
248	432	252	43
250	23	256	1

weight. The proposed framework can be extended to other parameter configurations (n variables, 2^m weight, d-th-order correlation immunity). It should be noted, however, that for cases with $d \geq 2$, Algorithm 7 requires redesign to ensure Boolean functions with optimally uniform spectral distributions.

7 Conclusion

This study is devoted to provide a new insight into constructing cryptographic Boolean functions via Walsh spectral analysis. We begin by reformulating the problem of constructing Boolean functions with desired cryptographic properties as **Problem**: the reconstruction of Boolean functions from partial Walsh spectrum. To tackle the **Problem**, we design the IWR algorithm and optimized it using multiple theoretical and practical techniques to obtain the OIWR algorithm. Our algorithm achieves a reduced time complexity of $O((|T|\cdot r+|\Delta|)\cdot n\cdot 2^n)$ for solving the **Problem**, while demonstrating both efficiency and solution accuracy across 14 test instances derived from highly nonlinear Boolean functions. Furthermore, for practical implementation, we propose a novel indirect construction method for correlation immune functions with low Hamming weight based on the OIWR algorithm. Through distinct constructing strategies of the set S, our framework enables flexible realization of diverse construction objectives. Experimental validation using Mesnager and Su's framework demonstrates that our approach overcomes two key limitations of their method: (1) restricted construction quantity, and (2) power-of-two weight constraints.

In future research, studying the counting problem for the **Problem**, i.e. determining the number of Boolean functions satisfying a given S, would be interesting. Such analysis would reveal the effects of Walsh spectrum points on the solutions, providing guidance for set S construction. We have conducted preliminary investigations on simplified cases of this problem, as detailed in Appendix A. It would be exciting also

to further investigate the construction of other cryptographic Boolean functions (e.g., bent functions) using the OIWR algorithm.

Acknowledgements. This study is supported by the National Natural Science Foundation of China No. 62206312.

Declarations

Data Availability. The datasets generated/supporting this study are available in the IEEE DataPort repository, and are cited in the manuscript as [37] and [41].

Experimental Setup. All experiments in this study are conducted on a workstation equipped with an Intel(R) Core(TM) i7-9700 CPU @ 3.0GHz and 16GB RAM, and Gurobi Optimizer 10.0.1 (free academic license).

Appendix A Discussion on the Counting Problem

Definition 6 For an arbitrary $n \in \{1, 2, 3, \ldots\}$, define the set:

$$\begin{aligned} S \triangleq \{(a_1, b_1), (a_2, b_2), \ldots, (a_m, b_m) \mid & 1 \le m \le 2^n,\ 0 \le a_i \le 2^n - 1, \\ & -2^n \le b_i \le 2^n,\ m, a_i, b_i \in \mathbb{Z}, \\ & i = 1, 2, \ldots, m\}, \end{aligned}$$

determine the number of possible Boolean functions $f \in \mathcal{B}_n$ satisfying S, i.e. $\hat{\boldsymbol{f}}(a_i) = b_i, i = 1, 2, \ldots, m$.

Definition 6 presents the counting problem for **Problem**, which we believe is difficult for an arbitrary n and S. However, in some special cases, for example, when one or two spectral points are known, we provide theoretical results.

A.1 Case A: $S = \{(\alpha_0, w_0)\}$, i.e., one spectral point is known

Proposition 7 *Given an arbitrary $f \in \mathcal{B}_n$ and a spectral point (α_0, w_0) in the Walsh spectrum of f, let $S = \{(\alpha_0, w_0)\}$ and define the number of solutions as M. Then,*

$$M = \binom{2^n}{2^{n-1} + \frac{w_0}{2}}.$$

Proof Let $\boldsymbol{L}_{\alpha_0}$ be the truth table of the linear function $L_{\alpha_0}(x) = \alpha_0 \cdot x$. From the Walsh transform definition in (1), we have

$$\hat{\boldsymbol{f}}[\alpha_0] = \sum_{x=0}^{2^n-1} (-1)^{\boldsymbol{f}[x] \oplus \boldsymbol{L}_{\alpha_0}[x]}.$$

Since $\hat{\boldsymbol{f}}[\alpha_0] = w_0$ and $(-1)^{\boldsymbol{f}[x] \oplus \boldsymbol{L}_{\alpha_0}[x]} = 1 - 2(\boldsymbol{f}[x] \oplus \boldsymbol{L}_{\alpha_0}[x])$, we obtain

$$w_0 = 2^n - 2\,\mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_0}).$$

Let $d = \mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_0})$ denote the Hamming distance between f and L_{α_0}. Then

$$d = \frac{2^n - w_0}{2}.$$

The number of Boolean functions f at a fixed Hamming distance d from a fixed linear function equals the number of ways to choose d positions out of 2^n where $\boldsymbol{f}$ differs from $\boldsymbol{L}_{\alpha_0}$, i.e., $\binom{2^n}{d}$. Hence,

$$M = \binom{2^n}{(2^n - w_0)/2}.$$

Using the symmetry $\binom{2^n}{k} = \binom{2^n}{2^n-k}$, this can be rewritten as $\binom{2^n}{2^{n-1}+w_0/2}$. □

Remark 1 (1) Uyan et al.[42] obtained the same result as in Proposition 7 by consecutive bit-flips in the truth tables of the affine Boolean functions.

(2) From Proposition 7, we can conclude that the number of solutions is related only to the *2-norm* of the known spectral point, that is, $\sqrt{w_0^2} = |w_0|$, regardless of its position.

(3) When n is determined, the larger $||\boldsymbol{w}||_2$, the smaller the value of M, which indicates that the spectral point with a larger absolute value contains more information about f.

A.2 Case B: $S = \{(\alpha_1, w_1), (\alpha_2, w_2)\}$, i.e., two spectral points are known

Proposition 8 *Given an arbitrary $f \in \mathcal{B}_n$ and two spectral points in its Walsh spectrum $S = \{(\alpha_1, w_1), (\alpha_2, w_2)\}$, define the number of solutions as M. Then,*

$$M = \binom{2^{n-1}}{2^{n-2} - \frac{w_1+w_2}{4}} \cdot \binom{2^{n-1}}{2^{n-2} - \frac{w_1-w_2}{4}}.$$

Proof Let $\boldsymbol{L}_{\alpha_1}$ and $\boldsymbol{L}_{\alpha_2}$ be the truth tables of the linear functions $L_{\alpha_1}(x) = \alpha_1 \cdot x$ and $L_{\alpha_2}(x) = \alpha_2 \cdot x$, respectively.

From the Walsh transform definition, we have

$$\hat{\boldsymbol{f}}[\alpha_1] = 2^n - 2\,\mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_1}), \quad \hat{\boldsymbol{f}}[\alpha_2] = 2^n - 2\,\mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_2}).$$

Let $d_1 = \mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_1})$ and $d_2 = \mathrm{wt}(\boldsymbol{f} \oplus \boldsymbol{L}_{\alpha_2})$. Then

$$d_1 = \frac{2^n - w_1}{2}, \quad d_2 = \frac{2^n - w_2}{2}.$$

Partition the input space $\{0, 1, \ldots, 2^n - 1\}$ into two sets:

$$S = \{x \mid \boldsymbol{L}_{\alpha_1}[x] = \boldsymbol{L}_{\alpha_2}[x]\}, \quad T = \{x \mid \boldsymbol{L}_{\alpha_1}[x] \neq \boldsymbol{L}_{\alpha_2}[x]\},$$

we obtain $|S| = |T| = 2^{n-1}$.

Let $a = |\{x \in S \mid \boldsymbol{f}[x] \neq \boldsymbol{L}_{\alpha_1}[x]\}|$ and $b = |\{x \in T \mid \boldsymbol{f}[x] \neq \boldsymbol{L}_{\alpha_1}[x]\}|$. Then,

$$d_1 = a + b.$$

For $x \in T$, since $\boldsymbol{L}_{\alpha_1}[x] \neq \boldsymbol{L}_{\alpha_2}[x]$, we have $\boldsymbol{f}[x] \neq \boldsymbol{L}_{\alpha_2}[x]$ if and only if $\boldsymbol{f}[x] = \boldsymbol{L}_{\alpha_1}[x]$. Hence, the number of $x \in T$ with $\boldsymbol{f}[x] \neq \boldsymbol{L}_{\alpha_2}[x]$ is $|T| - b = 2^{n-1} - b$, and thus

$$d_2 = a + (2^{n-1} - b).$$

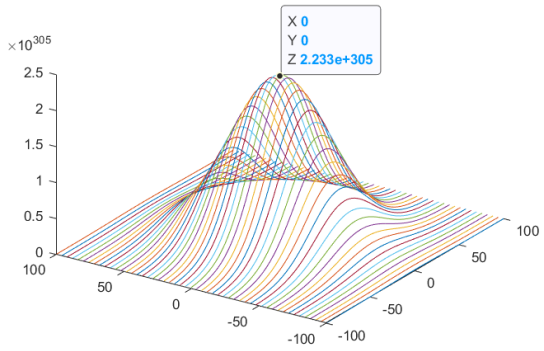


Fig. A1 Relationship between M and (w_1, w_2) given two known spectral points for $n = 10$.

Solving these equations yields

$$a = \frac{d_1 + d_2 - 2^{n-1}}{2} = 2^{n-2} - \frac{w_1 + w_2}{4}, \quad b = \frac{d_1 - d_2 + 2^{n-1}}{2} = 2^{n-2} - \frac{w_1 - w_2}{4}.$$

The number of ways to choose the set of points in S where f differs from L_{α_1} is $\binom{2^{n-1}}{a}$, and similarly for T it is $\binom{2^{n-1}}{b}$. Since the choices in S and T are independent, the total number of functions f satisfying both conditions is

$$M = \binom{2^{n-1}}{a} \cdot \binom{2^{n-1}}{b} = \binom{2^{n-1}}{2^{n-2} - \frac{w_1+w_2}{4}} \cdot \binom{2^{n-1}}{2^{n-2} - \frac{w_1-w_2}{4}}.$$

□

Remark 2 (1) Proposition 8 shows that when two points are known in the Walsh spectrum, the number of solutions is still only related to the spectral values, regardless of their positions.

(2) The larger $||\boldsymbol{w}||_2$, the smaller M, which still implies that spectral points with larger absolute values contain more information regarding f. Since the form of M in Proposition 8 cannot intuitively demonstrate this point, we employ MATLAB to generate a 3D plot with w_1, w_2, and M mapped to the X-, Y-, and Z-axes respectively for the case $n = 10$, as shown in Figure A1.

References

[1] Carlet, C., Crama, Y., Hammer, P.L.: Boolean functions for cryptography and error-correcting codes. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, pp. 257–397. Cambridge Univ. Press, Cambridge (2010). https://doi.org/10.1017/CBO9780511780448.011

[2] Chakrabarty, K., Hayes, J.: Balanced boolean functions. IEE Proc.-Comput. Digit. Tech. **145**(1), 52–62 (1998) https://doi.org/10.1049/ip-cdt:19981769

[3] Siegenthaler, T.: Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans. Inf. Theory **30**(5), 776–780 (1984) https://doi.org/10.1109/TIT.1984.1056949

[4] Rothaus, O.S.: On "bent" functions. J. Comb. Theory A **20**(3), 300–305 (1976) https://doi.org/10.1016/0097-3165(76)90024-8

[5] Mesnager, S., Su, S.: On correlation immune boolean functions with minimum hamming weight power of 2. IEEE Trans. Inf. Theory **67**(11), 7501–7517 (2021) https://doi.org/10.1109/TIT.2021.3109946

[6] Kudin, S., Pasalic, E.: Efficient design methods of low-weight correlation-immune functions and revisiting their basic characterization. Discrete Appl. Math. **284**, 150–157 (2020) https://doi.org/10.1016/j.dam.2020.03.031

[7] Carlet, C., Chen, X.: Constructing low-weight d th-order correlation-immune boolean functions through the fourier-hadamard transform. IEEE Trans. Inf. Theory **64**(4), 2969–2978 (2018) https://doi.org/10.1109/TIT.2017.2785775

[8] Pang, S., Wang, J., Wang, X., Wang, X.: Application of orthogonal array and walsh transform in resilient function. Chin. J. Electron. **27**(2), 281–286 (2018) https://doi.org/10.1049/cje.2017.09.011

[9] Su, S., Wang, B., Li, J.: On the constructions of resilient boolean functions with five-valued walsh spectra and resilient semi-bent functions. Discrete Appl. Math. **309**, 1–12 (2022) https://doi.org/10.1016/j.dam.2021.11.012

[10] Dobbertin, H.: Construction of bent functions and balanced boolean functions with high nonlinearity. In: Preneel, B. (ed.) Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. Lecture Notes in Computer Science, vol. 1008, pp. 61–74. Springer, Leuven (1994). https://doi.org/10.1007/3-540-60590-8_5

[11] Zhang, W.: High-meets-low: Construction of strictly almost optimal resilient boolean functions via fragmentary walsh spectra. IEEE Trans. Inf. Theory **65**(9), 5856–5864 (2019) https://doi.org/10.1109/TIT.2019.2899397

[12] Jin, W., Du, X., Wu, W., Chen, Z.: Generic construction of boolean functions with a few walsh transform values of any possible algebraic degree. Adv. Math. Commun. **16**(4, SI), 811–832 (2022) https://doi.org/10.3934/amc.2022050

[13] Su, S.: A new construction of rotation symmetric bent functions with maximal algebraic degree. Adv. Math. Commun. **13**(2), 253–265 (2019) https://doi.org/10.3934/amc.2019017

[14] Zhang, H., Su, S.: A new construction of rotation symmetric boolean functions with optimal algebraic immunity and higher nonlinearity. Discrete Appl. Math.

262, 13–28 (2019) https://doi.org/10.1016/j.dam.2019.02.030

[15] Su, S., Li, J., Wang, B.: A new construction of odd-variable rotation symmetric boolean functions with optimal algebraic immunity and higher nonlinearity. Theor. Comput. Sci. **887**, 152–163 (2021) https://doi.org/10.1016/j.tcs.2021.07.018

[16] Qu, L., Zhang, Q., Li, K.: Parametric construction approach of balanced boolean functions from two-to-one mappings. J. Cryptol. **38**(3) (2025) https://doi.org/10.1007/s00145-025-09545-6

[17] Budaghyan, L.: Construction and Analysis of Cryptographic Functions. Springer, Bergen (2014)

[18] Carlet, C. (ed.): Boolean Functions for Cryptography and Coding Theory. Cambridge University Press, Cambridge (2020)

[19] Laarhoven, P.J.M., Aarts, E.H.L.: Simulated Annealing: Theory and Applications. Springer, Eindhoven (1987)

[20] Izbenko, Y.I., Kovtun, V., Kuznetsov, A.: The design of boolean functions by modified hill climbing method. In: Sixth International Conference on Information Technology: New Generations, ITNG 2009, Las Vegas, Nevada, USA, 27-29 April 2009, pp. 356–361. IEEE Computer Society, Las Vegas (2009). https://doi.org/10.1109/ITNG.2009.102

[21] Djurasevic, M., Jakobovic, D., Mariot, L., Picek, S.: A survey of metaheuristic algorithms for the design of cryptographic boolean functions. Cryptogr. Commun. **15**(6), 1171–1197 (2023) https://doi.org/10.1007/s12095-023-00662-2

[22] Knezevic, K., Picek, S., Mariot, L., Jakobovic, D., Leporati, A.: The design of (almost) disjunct matrices by evolutionary algorithms. In: International Conference on Theory and Practice of Natural Computing, pp. 152–163 (2018). https://doi.org/10.1007/978-3-030-04070-3_12 . Springer

[23] Picek, S., Guilley, S., Carlet, C., Jakobovic, D., Miller, J.F.: Evolutionary approach for finding correlation immune boolean functions of order t with minimal hamming weight. In: International Conference on Theory and Practice of Natural Computing, pp. 71–82 (2015). https://doi.org/10.1007/978-3-319-26841-5_6 . Springer

[24] Zheng, Y., Zhang, X.-M.: On plateaued functions. IEEE Trans. Inf. Theory **47**(3), 1215–1223 (2001) https://doi.org/10.1109/18.915690

[25] Clark, J.A., Jacob, J.L., Maitra, S., Stănică, P.: Almost boolean functions: The design of boolean functions by spectral inversion. Comput. Intell. **20**(3), 450–462 (2004)

[26] Mariot, L., Leporati, A.: A genetic algorithm for evolving plateaued cryptographic boolean functions. In: Proc. Int. Conf. Theory Pract. Nat. Comput. (TPNC 2015), pp. 33–45. Springer, Prague, Czech Republic (2015)

[27] Saber, Z., Uddin, M.F., Youssef, A.: On the existence of (9, 3, 5,240) resilient functions. IEEE Trans. Inf. Theory **52**(5), 2269–2270 (2006)

[28] Arshad, S., Khan, M.: Construction of nonlinear component based on bent boolean functions. Comput. Appl. Math. **43**(1) (2024) https://doi.org/10.1007/s40314-023-02545-x

[29] Kavut, S., Tutdere, S.: Highly nonlinear (vectorial) boolean functions that are symmetric under some permutations. Adv. Math. Commun. **14**(1), 127–136 (2020) https://doi.org/10.3934/amc.2020010

[30] Jakobovic, D., Picek, S., Martins, M.S.R., Wagner, M.: Toward more efficient heuristic construction of boolean functions. Appl. Soft Comput. **107** (2021) https://doi.org/10.1016/j.asoc.2021.107327

[31] Sipser, M.: Introduction to the Theory of Computation. PWS Publishing Company, Cambridge (1996). https://doi.org/10.1145/230514.571645

[32] Strang, G.: Introduction to Linear Algebra. SIAM, Cambridge (2022)

[33] Xiao, G., Massey, J.L.: A spectral characterization of correlation-immune combining functions. IEEE Trans. Inf. Theory **34**(3), 569–571 (1988) https://doi.org/10.1109/18.6037

[34] Meier, W., Staffelbach, O.: Nonlinearity criteria for cryptographic functions. In: Quisquater, J., Vandewalle, J. (eds.) Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings. Lecture Notes in Computer Science, vol. 434, pp. 549–562. Springer, Belgium (1989). https://doi.org/10.1007/3-540-46885-4_53

[35] Burden, R., Faires, J.D.: Numerical Analysis, 9th Edition. Cengage Learning, Cambridge (2010)

[36] Shanks, J.L.: Computation of the fast walsh-fourier transform. IEEE Trans. Computers **18**(5), 457–459 (1969) https://doi.org/10.1109/T-C.1969.222685

[37] He, S.: Test Instances for OIWR. IEEE Dataport (2025). https://doi.org/10.21227/7904-ns52 . https://dx.doi.org/10.21227/7904-ns52

[38] Senju, S., Toyoda, Y.: An approach to linear programming with 0-1 variables. Manage. Sci., 196–207 (1968)

[39] Optimization, G.: Gurobi. Accessed 9 November 2024. (2008). https://www.

gurobi.com/

[40] Camion, P., Carlet, C., Charpin, P., Sendrier, N.: On correlation-immune functions. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 86–100. Springer, California (1991). https://doi.org/10.1007/3-540-46766-1_6

[41] He, S.: Truth tables of low Hamming weight 1st-order correlation immune Boolean functions with 12 variables. IEEE Dataport (2025). https://doi.org/10.21227/e17v-rf27 . https://dx.doi.org/10.21227/e17v-rf27

[42] Uyan, E., Çalık, Ç., Doğanaksoy, A.: Counting boolean functions with specified values in their walsh spectrum. J. Comput. Appl. Math. **259**, 522–528 (2014) https://doi.org/10.1016/J.CAM.2013.06.035