A Proofs

In this section, we provide the proof of the propositions given in the article. For each proposition, we re-state the statement and we provide the corresponding proof.

A.1 Proof of Proposition 1.

Statement: A Boolean function f has an AI_S of 0 if and only if the restriction f_S on S is constant.

Proof. First, we show the reverse implication. We suppose that $f \in \mathcal{B}_n$ is constant on S, either $f(x) = 0$ for all $x \in S$, or $f(x) = 1$ for all $x \in S$. If f is not null everywhere but $f(x) = 0\ \forall x \in S$, then the constant function $g(x) = 1$ is not zero on all S, and is it such that $g(x)f(x) = 0$ for all $x \in S$. Similarly, if for all $x \in S$ $f(x) = 1$, the same argument can be used for the function $f + 1$ on S.

Then, we show the direct implication. We suppose that a function $g \in \mathcal{B}_n$ is a non-zero-annihilator of f restricted to S of degree 0. Then, $g \neq 0$ and $\deg(g) = 0 \Rightarrow g = 1$, therefore since g annihilates f in S, $g(x)f(x) = 0$ for all $x \in S$, resulting in $f(x)g(x) = f(x) = 0$. Similarly, if g annihilates $f + 1$ on S instead of f, we have that $g(x)(f(x) + 1) = 0$ implying that $f + 1 = 0 \Rightarrow f(x) = 1$ for all $x \in S$. □

A.2 Proof of Proposition 2.

Statement: Let $Z = \mathsf{supp}\left(f^{(o)}\right) \cap S$. If $\mathrm{rank}\left(G_{r,n}^{f_Z^{(o)},S}\right) < \mathrm{rank}(G_{r,n}^S)$, then $f^{(o)}$ admits a non-zero-annihilator restricted to S of degree at most r.

Proof. Let $D_r^n = \sum_{i=0}^{r}\binom{n}{i}$. Since $D_r^n = \mathrm{rank}(G_{r,n}^{f_S^{(o)},S}) + |\mathrm{Ker}(G_{r,n}^{f_S^{(o)},S})| = \mathrm{rank}(G_{r,n}^S) + |\mathrm{Ker}(G_{r,n}^S)|$, we have that $\mathrm{rank}(G_{r,n}^{f_S^{(o)},S}) < \mathrm{rank}(G_{r,n}^S) \iff |\mathrm{Ker}(G_{r,n}^{f_S^{(o)},S})| > |\mathrm{Ker}(G_{r,n}^S)|$. Therefore $\exists v \in \mathrm{Ker}(G_{r,n}^{f_S^{(o)},S})$ such that $v \notin \mathrm{Ker}(G_{r,n}^S)$. Hence, it exists a Boolean function g of degree r, whose truth table is given by $v \cdot G_{r,n}$, which is such that $v \cdot G_{r,n} \neq 0$ and $v \cdot G_{r,n}^S \neq 0$, but $(g \cdot f_S^{(o)})_S = v \cdot G_{r,n}^{f_S^{(o)},S} = 0$. Since $\mathrm{rank}\left(G_{r,n}^{f_S^{(o)},S}\right) = \mathrm{rank}\left(G_{r,n}^{f^{(o)},S}\right)$, $\mathrm{rank}(G_{r,n}^{f^{(o)},S}) < \mathrm{rank}(G_{r,n}^S) \Rightarrow \mathrm{rank}(G_{r,n}^{f_Z^{(o)},S}) < \mathrm{rank}(G_{r,n}^S)$ and hence the condition $\mathrm{rank}(G_{r,n}^{f_Z^{(o)},S}) < \mathrm{rank}(G_{r,n}^S)$ is enough to justify the existence of such v giving the non-zero-annihilator g of f^o restricted to S. □

A.3 Proof of Proposition 3

Statement: Let $k < \min\{(|Z|, |E_{D_n^n}|\}$ and let V_k constructed following Definition 13. Suppose $\ker(V_k) = \langle \hat{g}_k \rangle = \langle (\epsilon_1, \epsilon_2, \ldots, \epsilon_k) \rangle$ and let $g_k \in \mathcal{B}_n$ be the Boolean function having $\hat{g}_k$ as ANF. Furthermore, assume there exists a z_{k+j}, for $j \geq 1$ such that $g_k(z_{k+j}) = 1$. Then, the matrix $V_{k+1}^{'}$ constructed as:

$$V_{k+1}^{'} = \begin{pmatrix} & & & & z_{k+j}^{\alpha_1} \\ & & & & z_{k+j}^{\alpha_2} \\ & & V_k & & \vdots \\ & & & & z_{k+j}^{\alpha_k} \\ z_1^{\alpha_{k+1}} & z_2^{\alpha_{k+1}} & \cdots & z_k^{\alpha_{k+1}} & z_{k+j}^{\alpha_{k+1}} \end{pmatrix}$$

is such that $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k, 0) \notin \ker(V_{k+1}^{'})$.

Proof. Suppose $\ker(V_k) = \langle \hat{g}_k \rangle = \langle (\epsilon_1, \epsilon_2, \ldots, \epsilon_k) \rangle$, and that there exists $j \geq 1$ such that $g_k(z_{k+j}) = 1$, but it is also such that $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k, 0) \in \ker(V'_{k+1})$.

If $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k, 0) \in \ker(V'_{k+1})$, then $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k, 0) \cdot V'_{k+1} = \hat{0}$. Specifically, the last equation in the system of equations described by $(\epsilon_1, \epsilon_2, \ldots, \epsilon_k, 0) \cdot V'_{k+1} = \hat{0}$ is:

$$\epsilon_1 z_{k+j}^{\alpha_1} + \cdots + \epsilon_k z_{k+j}^{\alpha_k} + 0 \cdot z_{k+j}^{\alpha_{k+1}} = \epsilon_1 z_{k+j}^{\alpha_1} + \cdots + \epsilon_k z_{k+j}^{\alpha_k} = 0.$$

However, from our hypothesis, we have:

$$g_k(z_{k+j}) = 1 \iff g_k(z_{k+j}) = (\epsilon_1, \epsilon_2, \ldots, \epsilon_k) \begin{pmatrix} z_{k+j}^{\alpha_1} \\ z_{k+j}^{\alpha_2} \\ \vdots \\ z_{k+j}^{\alpha_k} \end{pmatrix} = \epsilon_1 z_{k+j}^{\alpha_1} + \cdots + \epsilon_k z_{k+j}^{\alpha_k} = 1.$$

This contradiction shows that it cannot be the case that $(\epsilon_1, \ldots, \epsilon_k, 0)$ is in the kernel of V'_{k+1}.

$\square$

A.4 Proof of Proposition 4.

Statement: The sequence of matrices $(V_k)_k$, constructed as described in Proposition 3, ensures that:

$$\dim(\ker(V_k)) \leq 1, \quad \forall k \in [1, |Z|].$$

Proof. We prove the result by induction:

- Base case. The initial matrix $V_1 = (1)$ has full rank, so $\dim(\ker(V_1)) = 0 \leq 1$.
- Induction step. We suppose that $\dim(\ker(V_k)) \leq 1$. We show that $\dim(\ker(V_{k+1})) \leq 1$.
  - If $\dim(\ker(V_k)) = 0$, adding a new row and column to form V_{k+1} can increase the kernel by at most 1. Hence, $\dim(\ker(V_{k+1})) \leq 1$.
  - If $\dim(\ker(V_k)) = 1$, let $\hat{z}$ be an element of $\ker(V_{k+1})$. Then either:

* $\hat{z} = (\hat{\epsilon}, 0)$:

$$\hat{z} \cdot V_{k+1} = 0 \implies \hat{\epsilon} \cdot V_k = 0 \implies \hat{\epsilon} \in \ker(V_k).$$

By Proposition 3, $\hat{\epsilon}$ can only be a trivial solution.

* $\hat{z} = (\hat{y}, 1)$, where $\hat{y} \in \mathbb{F}_2^k$: In this case $\hat{z} \cdot V_{k+1} = 0$ implies:

$$\hat{y} \cdot V_k + (z_1^{\alpha_{k+1}}, \cdots, z_k^{\alpha_{k+1}}) = 0 \iff \hat{y} \cdot V_k = (z_1^{\alpha_{k+1}}, \cdots, z_k^{\alpha_{k+1}}), \quad (1)$$

and

$$\hat{y} \cdot (z_{k+1}^{\alpha_1}, \cdots, z_{k+1}^{\alpha_k})^T + z_{k+1}^{\alpha_{k+1}} = 0 \iff \hat{y} \cdot (z_{k+1}^{\alpha_1}, \cdots, z_{k+1}^{\alpha_k})^T = z_{k+1}^{\alpha_{k+1}}. \quad (2)$$

Since $\dim(\ker(V_k)) = 1$, $\mathrm{rank}(V_k) = k - 1$. Hence, there exists at most one particular solution $\hat{y}_p$ for Equation 1. If $\hat{y}_p$ also solves Equation 2, then $\hat{y}_p$ is in $\ker(V_{k+1})$.

Since $(\hat{\epsilon}, 0)$ cannot be in $\ker(V_{k+1})$, the only possible element in $\ker(V_{k+1})$ is $\hat{y}_p$, if it exists. Therefore, $\dim(\ker(V_{k+1})) \leq 1$.

□

A.5 Proof of Proposition 5.

Statement: Let $\ker(V_{E_k,Z_k}) = \langle \hat{g}_k \rangle = \langle (\epsilon_1, \ldots, \epsilon_k) \rangle$, where the corresponding Boolean function $g_k \in \mathcal{B}_n$ satisfies $g_k(x) = 0,\ \forall x \in Z$.

Define $E^* = (\alpha_1, \alpha_2, \ldots, \alpha_{k-1}, \alpha_{k+1})$, a vector constructed from E_k by replacing the last element α_k with α_{k+1}. Let $V_{k+1} = V_{E^*_{k+1},Z_k}$ be the matrix defined by replacing the monomial α_k in V_{E_k,Z_k}, with the monomial α_{k+1}:

$$V_{E^*_{k+1},Z_k} = \begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & E_{k-1}, V_{Z_{k-1}} & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ z_1^{\alpha_{k+1}} & z_2^{\alpha_{k+1}} & \cdots & z_{k-1}^{\alpha_{k+1}} & z_k^{\alpha_{k+1}} \end{pmatrix}.$$

Then, either $\hat{g}_k \notin \ker(V_{E^*_{k+1},Z_k})$, or $\hat{g}_k$ is the only element (aside from $\hat{0}$) in the kernel of $V_{E^*_{k+1},Z_k}$. Hence:

$$\dim\Big(\ker(V_{E^*_{k+1},Z_k})\Big) \leq 1.$$

Proof. Proposition 3 ensures that the kernel of $V_{E^*_{k+1},Z_k}$ does not contain elements derived from the kernel of $V_{E_{k-1},Z_{k-1}}$. Therefore, $V_{E_{k-1},Z_{k-1}}$ can be treated as a full-rank matrix. Modifying the last row of a matrix that contains a full-rank sub-matrix does not increase the size of the kernel. This is because the first k entries of the elements in the kernel of V_{E_k,Z_k} and $V_{E^*_{k+1},Z_k}$ are the same, with the only difference arising from the last row. Since this row is the only one that can introduce linear dependence, the dimension of the kernel of $V_{E^*_{k+1},Z_k}$ is bounded by 1. □

A.6 Proof of Proposition 6.

Statement: Let k be such that $1 < k < \min\left(|Z|, |E_{D_n^n}|\right)$ it holds that

$$EF\left(V_k\right) =_p EF\left(\begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & EF\left(V_{k-1}\right) & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ (z_1^{\alpha_k} & z_2^{\alpha_k} & \cdots & z_{k-1}^{\alpha_k})P_{k-1} & z_k^{\alpha_k} \end{pmatrix}\right).$$

where P_{k-1} is such that $U_{k-1}^{-1} = P_{k-1}$, where $V_{k-1} = L_{k-1}U_{k-1}$ with $EF(V_{k-1}) =_p L_{k-1}$

Proof. Let k be such that $1 < k < \min\left(|Z|, |E_{D_n^n}|\right)$. The LU-decomposition of V_{k-1} is $V_{k-1} = L_{k-1}U_{k-1}$, where U_{k-1} is an upper triangular matrix and $L_{k-1} = EF\left(V_{k-1}\right)$ is the lower triangular matrix.

$$V_k = \begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & L_{k-1}U_{k-1} & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ z_1^{\alpha_k} & z_2^{\alpha_k} & \cdots & z_{k-1}^{\alpha_k} & z_k^{\alpha_k} \end{pmatrix} = \begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & L_{k-1} & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ (z_1^{\alpha_k} & z_2^{\alpha_k} & \cdots & z_{k-1}^{\alpha_k})U_{k-1}^{-1} & z_k^{\alpha_k} \end{pmatrix} \begin{pmatrix} & & & & 0 \\ & & & & 0 \\ & & U_{k-1} & & \vdots \\ & & & & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since U_{k-1} is an upper triangular matrix, $\begin{pmatrix} & & & & 0 \\ & & & & 0 \\ & & U_{k-1} & & \vdots \\ & & & & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$ is also an upper triangular matrix, and therefore

$$EF\left(V_k\right) =_p EF\left(\begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & L_{k-1} & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ (z_1^{\alpha_k} & z_2^{\alpha_k} & \cdots & z_{k-1}^{\alpha_k})U_{k-1}^{-1} & z_k^{\alpha_k} \end{pmatrix}\right).$$

Taking $L_{k-1} =_p EF\left(V_{k-1}\right)$ and $P_{k-1} = U_{k-1}^{-1}$:

$$EF\left(V_k\right) =_p EF\left(\begin{pmatrix} & & & & z_k^{\alpha_1} \\ & & & & z_k^{\alpha_2} \\ & & EF\left(V_{k-1}\right) & & \vdots \\ & & & & z_k^{\alpha_{k-1}} \\ (z_1^{\alpha_k} & z_2^{\alpha_k} & \cdots & z_{k-1}^{\alpha_k})P_{k-1} & z_k^{\alpha_k} \end{pmatrix}\right).$$

$\square$

B Algorithms

In this section, we provide the pseudo-codes of the algorithms described in Section 3 and Section 4.2.

Algorithm 2 Algebraic immunity of f restricted to the set S, Reed-Muller method.

Input: $x \in \mathbb{F}_2^{2^n}$ evaluation of the truth table of $f \in \mathcal{B}_n$ and the restricted set $S \subseteq \mathbb{F}_2^n$.
Output: $\mathsf{AI}_S(f)$

function FIND_DEG_SMALLEST_S_ANNIHILATOR($G_{n,n}^{f_Z^{(o)},S}, G_{d,n}^S, D^n$)
  $r \leftarrow 0$;
  $i \leftarrow 0$;
  $ef_f, ef_S \leftarrow null, null$;
  $previousIndex \leftarrow 0$;
  while $r \leq |S|/2$ **do**
    $D_i^n \leftarrow D^n[i]$;
    $ef_f \leftarrow$ ECHELON_FORM_LAST_D_i^n_ROWS $\left(ef_f \mid\mid G_{n,n}^{f_Z^{(o)},S}[previousIndex : D_i^n]\right)$[1];
    $ef_S \leftarrow$ ECHELON_FORM_LAST_D_i^n_ROWS $\left(ef_S \mid\mid G_{n,n}^{S}[previousIndex : D_i^n]\right)$;
    $r \leftarrow rank\,(ef_S)$;
    if $rank\,(ef_f) < r$ **then**
      return i;
    end if
    $previousIndex \leftarrow D_i^n$;
    $i \leftarrow i + 1$;
  end while
end function

$Z = supp(x) \cap S$;
$Z_c = supp(x+1) \cap S$;
if $Z = \emptyset$ of $Z_c = \emptyset$ **then return** 0;
end if
$m_{max}, D^{\leq n_{max}} \leftarrow$ READ_VALUES_FROM_FILE();
$G_{n,n} \leftarrow m_{max}[n]$;
$D^n \leftarrow D^{\leq n_{max}}[n]$;
$G_{n,n}^{f_Z,S}, G_{n,n}^{(f+1)_Z,S}, G_{n,n}^S \leftarrow$ CONSTRUCT_RESTRICTED_GENERATOR_MATRICES$(G_{n,n}, x, S)$[2]
$immunity_f, immunity_{f+1} \leftarrow$ (
 FIND_DEG_SMALLEST_S_ANNIHILATOR($G_{n,n}^{f_Z,S}, G_{n,n}^S, D^n$)
 | FIND_DEG_SMALLEST_S_ANNIHILATOR($G_{n,n}^{(f+1)_Z,S}, G_{n,n}^S, D^n$)
);
return MIN$(immunity_f, immunity_{f+1})$[3];

Algorithm 3 Algorithm to compute the algebraic immunity of a function $f \in \mathcal{B}_n$ restricted to S

Input: $x \in \mathbb{F}_2^{2^n}$ evaluation of the truth table of $f \in \mathcal{B}_n$ and the restricted set $S \subseteq \mathbb{F}_2^n$.
Output: $\mathsf{AI}_S(f)$

$Z \leftarrow supp(x) \cap S$;
$Z_c \leftarrow supp(x+1) \cap S$;
if $Z = \emptyset$ or $Z_c = \emptyset$ **then**
    return 0;
end if

$E_{D_n^n} \leftarrow$ CONSTRUCT_MONOMIALS$(n = n, d = n)$[4];
$r \leftarrow \frac{|Z|}{|Z_c|}$ **if** $|Z| < |Z_c|$ **else** $\frac{|Z_c|}{|Z|}$;
if $r \geq \frac{1}{4}$ **then**
    $immunity_f, immunity_{f+1} \leftarrow ($
FIND_DEG_SMALLEST_S_ANNIHILATOR$(Z = Z, E = E_{D_n^n}, S = S)$
$|$ FIND_DEG_SMALLEST_S_ANNIHILATOR$(Z = Z_c, E = E_{D_n^n}, S = S)$
$)$;
    return MIN$(immunity_f, immunity_{f+1})$;
else
    return FIND_DEG_SMALLEST_S_ANNIHILATOR_SEQ$(Z = Z, Z_c, E = E_{D_n^n}, S = S))$[5];
end if

C Comparison between Algorithm 2 and Algorithm 3 on WAPB functions

In this part, we compare the execution times of Algorithm 2 and Algorithm 3 by running both algorithms on multiple WAPB functions restricted to the largest set $\mathsf{E}_{\lceil n/2 \rceil, n}$, with different numbers of variables. Specifically, the experiment is done by running the

[2] The symbol $||$ is used as row-concatenation of matrices, meaning that $C = A||B$ is the matrix constructed by appending all the row of the matrix B to the matrix A.

[3] CONSTRUCT_RESTRICTED_GENERATOR_MATRICES is a function extracting $G_{n,n}^{f_{supp(f) \cap S}, S}$, $G_{n,n}^{f_{supp(f+1) \cap S}, S}$ and $G_{n,n}^{S}$ from the full Reed-Muller code generator matrix $G_{n,n}$ following Definition 11.

[4] The function MIN takes care of $null$ values: MIN$(immunity_f, null) = immunity_f$ and MIN$(null, immunity_{f+1}) = immunity_{f+1}$.

[9] CONSTRUCT_MONOMIALS generates all monomials up to the degree d with n variables. $E_{D_n^n}$ may also be pre-computed and the function CONSTRUCT_MONOMIALS removed from the algorithm.

[10] The implementation of the function FIND_DEG_SMALLEST_S_ANNIHILATOR_SEQ is similar to the function FIND_DEG_SMALLEST_S_ANNIHILATOR described in Algorithm 1 with the difference that the degree of the minimal-degree non-zero annihilator of f or of $f+1$ restricted on S, is found sequentially instead of in parallel.

algorithms with the number of variables n going from 1 to 12. For both algorithms, if $n \leq 4$, we calculate the average over the $\mathsf{AI}_{\lceil n/2 \rceil}$ of all functions in $\mathcal{WAPB}_n$, whereas for $n > 4$, we calculate the average over the $\mathsf{AI}_{\lceil n/2 \rceil}$ of a random sample of 10^4 $\mathcal{WAPB}_n$ functions.

Figure 1 shows the mean execution time per value of n of the two algorithms. We denote by $T\left(G^{\leq n_{max}}\right) = T\left(G^{\leq 12}\right)$ the necessary time to pre-compute all the necessary Reed-Muller codes's generator matrices following Equation (2): we recall that it is the necessary time to compute the sequence $(G_{n,n})_{n \in [1, 12 = n_{max}]}$, not only the generator matrix $G_{n,n}$ for the current iteration n. In fact, $T\left(G^{\leq 12}\right)$ is constant, that is because the computation occurs before beginning the experiment since it is expected that Algorithm 2 is able to run for any number of variable n up to an upper bound n_{max}, without having to compute the Reed-Muller code for it.

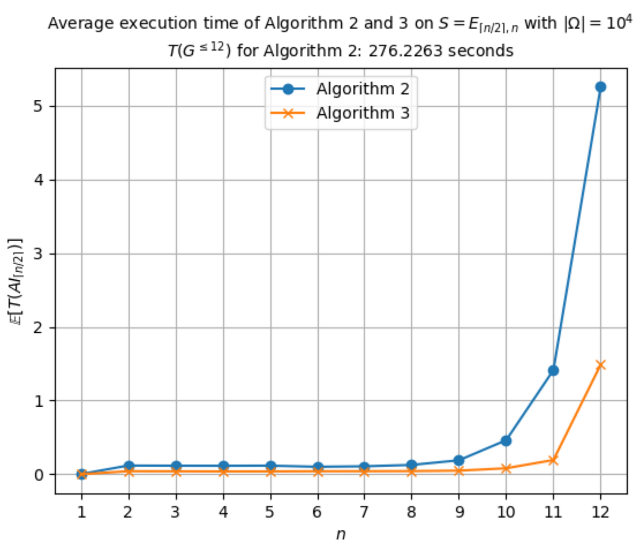

Fig. 1: Average time execution per number of variables of Algorithms 2 and 3 over 10^4 random $\mathcal{WAPB}_n$ functions on the set $E_{\lceil n/2 \rceil, n}$. $T\left(G^{\leq 12}\right)$ gives the time of the pre-computation of the sequence of all Reed-Muller matrices up to $G_{12,12}$.

The pre-computation times of $G^{\leq n_{max}}$ for Algorithm 2 are given in the Table 1. From Figure 1, we observe that Algorithm 3 outperforms Algorithm 2 for $n \leq 12$, even without considering the pre-computation time required for $G^{\leq n_{max}}$ and $D^{\leq n_{max}}$ in Algorithm 2. Furthermore, Algorithm 2 becomes impractical for $n > 12$ due to the excessive complexity of pre-computation. In contrast, Algorithm 3 remains feasible for larger values of n.

$G^{\leq 1}$	$G^{\leq 2}$	$G^{\leq 3}$	$G^{\leq 4}$	$G^{\leq 5}$	$G^{\leq 6}$	$G^{\leq 7}$	$G^{\leq 8}$	$G^{\leq 9}$	$G^{\leq 10}$	$G^{\leq 11}$	$G^{\leq 12}$
0.0126	0.0042	0.0017	0.004	0.0149	0.0470	0.1938	0.806	3.5456	14.9168	65.5014	276.2263

Table 1: Pre-computation time (in seconds) for $G^{\leq n_{max}}$ for n_{max} from 1 to 12.

Both implementations of the algorithms used to generate the plot in Figure 1 have been developed in *Python*, with the following enhancements:

– The linear algebra operations in Algorithm 2 are performed using the *SageMath* library, while those in Algorithm 3 are executed using an ad-hoc custom *Python* package implemented in *Rust*. This setup ensures comparability between the two implementations, as the *SageMath* library is highly optimized for linear algebra operations.
– The computation of $G^{\leq n_{max}}$ for Algorithm 2 is performed using the *generator_matrix* method of the *SageMath* object *BinaryReedMullerCodes*. While this computation could be more efficient if implemented in *Rust*, the current overall implementation still faces scalability issues. Specifically, for $n \geq 16$, the algorithm becomes impractical because $G_{16,16}$ cannot be stored in a native *Python* list or a *SageMath* matrix object. However, if the entire algorithm were implemented in *Rust*, it could potentially be feasible. Consequently, for larger values of n, Algorithm 2 is not usable, while Algorithm 3 remains viable.

D Plots of AIk probability distributions of WPB functions and average execution times.

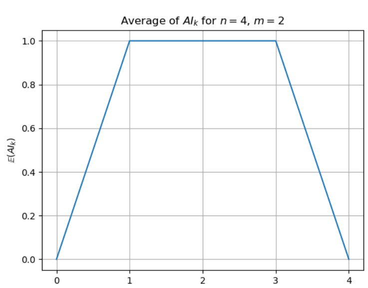

Fig. 2: Mean of AI_k over $\mathcal{WPB}_2$.

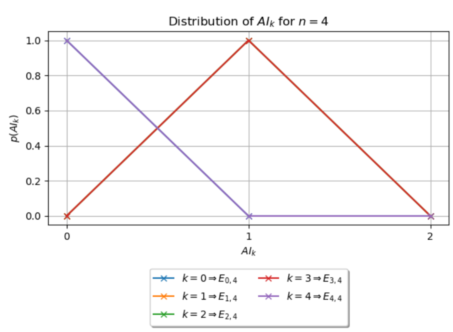

Fig. 3: Probability distribution of AI_k over $\mathcal{WPB}_2$.

Fig. 4: Mean of AI_k and probability distribution of AI_k from Definition 16 of 4-variable WPB functions.

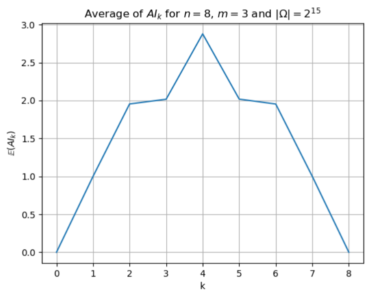

Fig. 5: Estimated average of AI_k over $\mathcal{WPB}_3$.

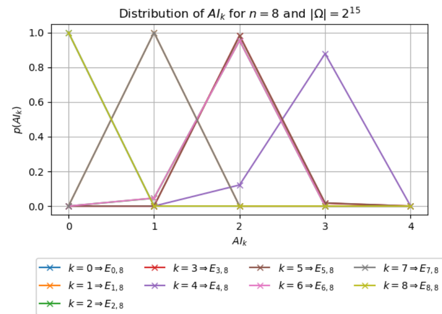

Fig. 6: Estimated Probability distribution of AI_k over $\mathcal{WPB}_3$.

Fig. 7: Estimated average of AI_k and probability distribution of AI_k over 8-variable WPB functions, with 2^{15} samples.

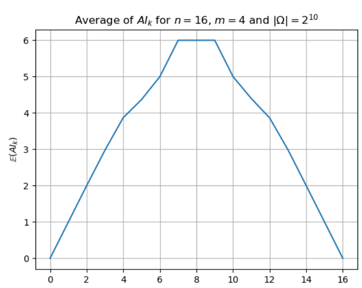

Fig. 8: Estimated average of AI_k over $\mathcal{WPB}_4$.

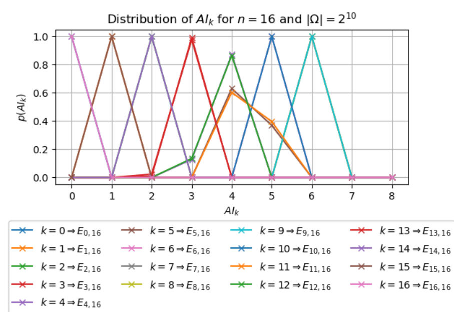

Fig. 9: Estimated Probability distribution of AI_k over $\mathcal{WPB}_4$.

Fig. 10: Estimated average of AI_k and probability distribution of AI_k over 8-variable WPB functions, with 2^{10} samples.

Table 2: Averages execution times and $\mathsf{AI}k$ of $\mathcal{WPB}_2$, $\mathcal{WPB}_3$ and $\mathcal{WPB}_4$ functions.

(a) Average execution time and AI_k for $n = 4$, $m = 2$.

k	0	1	2	3	4
$\mathbb{E}[T(AI_k)]$	0.53×10^{-4}	0.0526	0.0487	0.0586	0.6×10^{-5}
$\mathbb{E}[AI_k]$	0.00	1.00	1.00	1.00	0.00

(b) Average execution time and AI_k for $n = 8$, $m = 3$ and sampleSize $= 2^{15}$.

k	1	2	3	4	5	6	7
$\mathbb{E}[T(AI_k)]$	0.0756	0.086	0.094	0.090	0.102	0.092	0.0938
$\mathbb{E}[AI_k]$	1.00	1.95	2.02	2.88	2.02	1.95	1.00

(c) Average execution time and AI_k for $n = 16$, $m = 4$ and sampleSize $= 2^{10}$.

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathbb{E}[T(AI_k)]$	0.334	0.342	0.41	4.01	91.81	636.37	2838.92	6384.38	3111.12	732.50	118.96	5.23	0.43	0.35	0.35
$\mathbb{E}[AI_k]$	1.00	2.00	2.99	3.89	4.37	5.00	6.00	6.00	6.00	5.00	4.40	3.87	2.99	2.00	1.00