Computing with Haar Functions

Sami Khuri
Department of Mathematics and Computer Science
San José State University
One Washington Square
San José, CA 95192-0103, USA
khuri@jupiter.sjsu.edu
Fax: (408)924-5080

Keywords: *deceptive function, fast transforms, fitness functions, Haar functions, Walsh functions*

Abstract

Walsh functions are orthogonal, rectangular functions that take values ± 1 and form a convenient basis for the expansion of genetic algorithm fitness functions. Since their introduction into genetic algorithms [2, 8], they have been used to compute the average fitness values of schemata, to decide whether functions are hard or easy for genetic algorithms, and to design deceptive functions for the genetic algorithm. In [10], Haar functions were introduced as an alternative to Walsh functions and it was shown that Haar functions are in general more computationally advantageous. This paper revisits Haar functions, albeit with a slight variation to [10]'s definition, uses the functions to construct fully deceptive functions for the genetic algorithm (as was done with Walsh functions [5]), and studies fast Haar transforms and fast Walsh-Haar transforms.

1 Introduction

Orthonormal bases have captured the attention of researchers in a variety of fields. In wavelet theory, "orthonormal wavelet bases are used as a tool to describe mathematically the 'increment in information' needed to go from a coarse approximation to a higher resolution approximation" [3]. Digital electronics brought the necessity for more practical bases than Fourier's sine/cosine functions. Orthogonal, rectangular functions are better suited for digital electronics and have also been introduced as theoretical tools for the analysis of genetic algorithms [2, 8]. One of the reasons behind the success of Walsh's rectangular functions as a theoretical tool in genetic algorithms is related to the expansion of schemata in terms of Walsh coefficients. More precisely, the shorter the order of the schema (the smaller the number of fixed positions), the fewer Walsh coefficients it needs in the expansion. In [5], Walsh functions were used to construct fully-deceptive functions. After the introduction of Haar functions, in which we use a variation to the definition found in [10], we make use of these functions to construct, in a more efficient way than with Walsh functions, similar deceptive functions.

We then study two fast transforms, the fast Haar transform, and the fast Walsh-Haar transform. Because of the interdependent nature of fast transforms, the computation of a single term is built upon the values of previous levels: many more levels for Walsh than Haar. We conclude the paper by reiterating the computational advantage that Haar functions have over their Walsh counterparts, and observe that it might be advantageous to use Haar functions rather than Walsh's for some applications. In some cases, such as with symmetrical functions and for very small problem sizes, it might be preferable to use Walsh functions.

2 Haar Functions

The Haar functions [6] form a complete set of orthogonal rectangular basis functions. They take values of 1, 0 and -1, multiplied by powers of $\sqrt{2}$ and are usually defined on an interval normalized to $[0,1)$ [12].

In the following, the Haar functions are defined on $[0, 2^\ell)$, for any positive integer ℓ, and are normalized,

$$
\begin{aligned}
H_0(x) &= 1 \quad \text{for } 0 \le x < 2^\ell \\
H_1(x) &= \begin{cases} 1 & \text{for } 0 \le x < 2^{\ell-1} \\ -1 & \text{for } 2^{\ell-1} \le x < 2^\ell \end{cases} \\
H_2(x) &= \begin{cases} \sqrt{2} & \text{for } 0 \le x < 2^{\ell-2} \\ -\sqrt{2} & \text{for } 2^{\ell-2} \le x < 2^{\ell-1} \\ 0 & \text{elsewhere in } [0, 2^\ell) \end{cases} \\
H_3(x) &= \begin{cases} \sqrt{2} & \text{for } (2)2^{\ell-2} \le x < (3)2^{\ell-2} \\ -\sqrt{2} & \text{for } (3)2^{\ell-2} \le x < (4)2^{\ell-2} \\ 0 & \text{elsewhere in } [0, 2^\ell) \end{cases} \\
&\vdots \\
H_{2^q+m}(x) &= \begin{cases} (\sqrt{2})^q & \text{for } (2m)2^{\ell-q-1} \le x \\ & \text{and } x < (2m+1)2^{\ell-q-1} \\ -(\sqrt{2})^q & \text{for } (2m+1)2^{\ell-q-1} \le x \\ & \text{and } x < (2m+2)2^{\ell-q-1} \\ 0 & \text{elsewhere in } [0, 2^\ell) \end{cases} \qquad (1) \\
&\vdots \\
H_{2^\ell-1}(x) &= \begin{cases} (\sqrt{2})^{\ell-1} & \text{for } 2(2^{\ell-1}-1) \le x \\ & \text{and } x < 2^\ell - 1 \\ -(\sqrt{2})^{\ell-1} & \text{for } 2^\ell - 1 \le x < 2^\ell \\ 0 & \text{elsewhere in } [0, 2^\ell) \end{cases}
\end{aligned}
$$

For $q = 0, 1, \ldots, \ell-1$, we have $m = 0, 1, \ldots, 2^q - 1$.

The set $\{H_j(x) : j = 0, 1, 2, \ldots, 2^\ell - 1\}$ forms a basis for the fitness functions defined on the integers in $[0, 2^\ell)$. That is:

$$f(x) = \Sigma_{j=0}^{2^\ell-1} h_j \, H_j(x), \qquad (2)$$

where the h_j's, for $j = 2^q + m$, are the Haar coefficients given by:

$$h_j = \frac{1}{2^\ell} \Sigma_{x=0}^{2^\ell-1} f(x) \, H_j(x). \qquad (3)$$

The Haar function, $H_{2^q+m}(x)$, has degree q and order m. Functions with the same degree are such that each one is just a "right-shift" of the previous function. The position of the function is given by the order m. The higher the degree q, the smaller the subinterval with non-zero values for $H_j(x)$. Haar functions of degree q contain 2^q more non-zero parts than Haar functions of degree $q-1$. Consequently, each Haar coefficient depends only on the local behavior of $f(x)$. More precisely, the following two results (equivalent to Beauchamp's [1]) were obtained in [10]:

Result 1 Every Haar coefficient of degree q has at most $2^{\ell-q}$ non-zero terms. Each term corresponds to a point in an interval of the form $[(i)2^{\ell-q}, (i+1)2^{\ell-q})$. Consequently, the linear combination of each Haar coefficient h_j, where $j = 2^q + m$, has at most $2^{\ell-q}$ non-zero terms. In addition, h_0 has at most 2^ℓ non-zero terms. □

Result 2 For any fixed value $x \in [0, 2^\ell)$, $f(x)$ has at most $\ell + 1$ non-zero terms. □

In the next section, we use the Haar transform to build fully deceptive functions.

3 Designing Fully Deceptive 3-Bit Functions

In his quest to identify functions that are suitable for genetic algorithms, Goldberg, using Walsh coefficients, constructed fully deceptive 3-bit functions [5]. We use Haar functions to construct such a function. If 111 is the optimal point in the search space of a maximization problem, then all order-one schemata should lead away from it. In other words, we have:

- $f(\star\star 1) < f(\star\star 0)$, $f(\star 1\star) < f(\star 0\star)$, and $f(1\star\star) < f(0\star\star)$.

  By using Equation 2 and taking averages, the inequalities produce:
  $h_4 + h_5 + h_6 + h_7 > 0$, $h_2 + h_3 > 0$, and $h_1 > 0$, respectively.

Similarly, for order-two schemata, we want them to lead away from the best. In other words, we should have:

- $f(\star 00) > f(\star 01)$, $f(\star 00) > f(\star 10)$, and $f(\star 00) > f(\star 11)$;

  which produce the following inequalities:
  $h_4 + h_6 > 0$, $\sqrt{2}(h_2 + h_3) > -h_4 + h_5 - h_6 + h_7$, and $\sqrt{2}(h_2 + h_3) > -(h_4 + h_5 + h_6 + h_7)$.

- $f(0\star 0) > f(0\star 1)$, $f(0\star 0) > f(1\star 0)$, and $f(0\star 0) > f(1\star 1)$;

  which yield the following inequalities:
  $h_4 + h_5 > 0$, $2h_1 + (h_4 + h_5 - h_6 - h_7) > 0$, and $2h_1 + (h_4 + h_5 + h_6 + h_7) > 0$, respectively.

- $f(00\star) > f(01\star)$, $f(00\star) > f(10\star)$, and $f(00\star) > f(11\star)$;

  which yield the following inequalities:
  $h_2 > 0$, $2h_1 + \sqrt{2}(h_2 - h_3) > 0$, and $2h_1 + \sqrt{2}(h_2 + h_3) > 0$, respectively.

Finally, since 111 is the optimal point, we must have:

- $f(111) > f(000)$, $f(111) > f(001)$, $f(111) > f(010)$, and $f(111) > f(011)$,

  which yield the following inequalities:
  $2(h_1 + h_4 + h_7) + \sqrt{2}(h_2 + h_3) < 0$,
  $2(h_1 - h_4 + h_7) + \sqrt{2}(h_2 + h_3) < 0$,
  $2(h_1 + h_5 + h_7) - \sqrt{2}(h_2 + h_3) < 0$, and
  $2(h_1 - h_5 + h_7) - \sqrt{2}(h_2 - h_3) < 0$, respectively.

- $f(111) > f(100)$, $f(111) > f(101)$, and $f(111) > f(110)$,

  which produces the following inequalities:

  $h_6 + h_7 + \sqrt{2}h_3 < 0$, $-h_6 + h_7 + \sqrt{2}h_3 < 0$, and $h_7 < 0$, respectively.

Various solutions for the Haar coefficients can be obtained by solving the simultaneous set of inequalities, thus producing many fully deceptive functions. A possible solution to the inequalities is:

$h_0 = h_1 = 1, h_2 = 2\sqrt{2}, h_3 = -\sqrt{2}, h_4 = 10, h_5 = -1,$
$h_6 = 8, h_7 = -16.$

By replacing these values in Equation 2, we obtain the fully deceptive 3-bit function given in Table 1.

x	000	001	010	011	100	101	110	111
$f(x)$	26	−14	−4	0	14	−18	−30	34

Table 1: A fully deceptive 3-bit function.

The average fitness of $\star 10$, for instance, can be expressed as a linear combination of at most four non-zero Walsh coefficients:

$$f(\star 10) = w_0 + w_1 - w_2 - w_3.$$

The computation of w_i, for $i = 1, 2, 3, 4$, has 8 terms, requiring the values of $f(x)$ at all points in the interval $[0, 8)$, thus bringing the total number of computations to 32.

The average fitness of the same schema expressed as a linear combination of Haar coefficients yields:

$$f(\star 10) = h_0 - \frac{\sqrt{2}}{2}(h_2 + h_3) + h_5 + h_7.$$

According to Equation 3 and Result 1 from the previous section, the computation of h_2 and of h_3 has 2^{3-1} terms each (see Result 1, where $q = 1$ since $2 = 2^1 + 0$, and $3 = 2^1 + 1$), the computation of h_5 and h_7 have 2^{3-2} terms each (since $5 = 2^2 + 1$ and $7 = 2^2 + 3$), and h_0 requires the computation of 8 terms. Thus, the total number of computations is 20 with the Haar expansion, a substantial savings over Walsh's expansion.

To take advantage of the many repetitive computations performed with orthogonal transformations, fast Walsh and fast Haar transforms have been proposed. In the next section, we explore and compare fast Haar and Walsh transforms.

4 Fast Transforms

Fast transforms are represented by layered flowcharts where an intermediate result at a certain stage is obtained by adding (or subtracting) two intermediate results from the previous layer. Thus, when dealing with fast transforms, it is more appropriate to count the number of operations (additions or subtractions), which is equivalent to counting the number of terms [13]. These are in-place algorithms, and so memory storage for intermediate-stage computations is not needed. The calculated pair of output values can replace the corresponding pair of data in the preceding stage.

We study the fast Haar and the fast Walsh-Haar transforms, which are modified Cooley and Tukey algorithms, modeled after the fast Fourier transform.

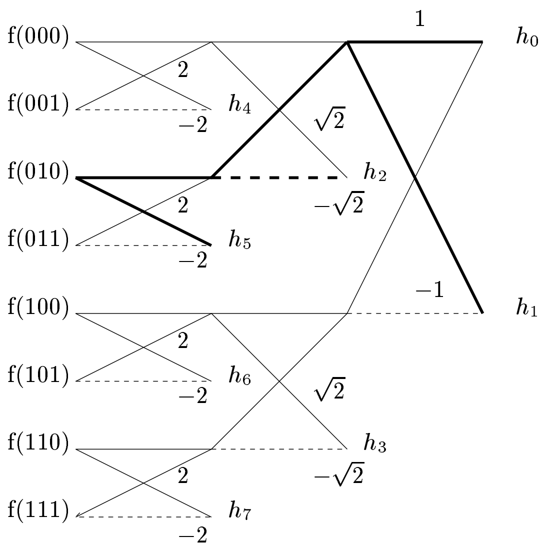

Figure 1: Fast Haar transform.

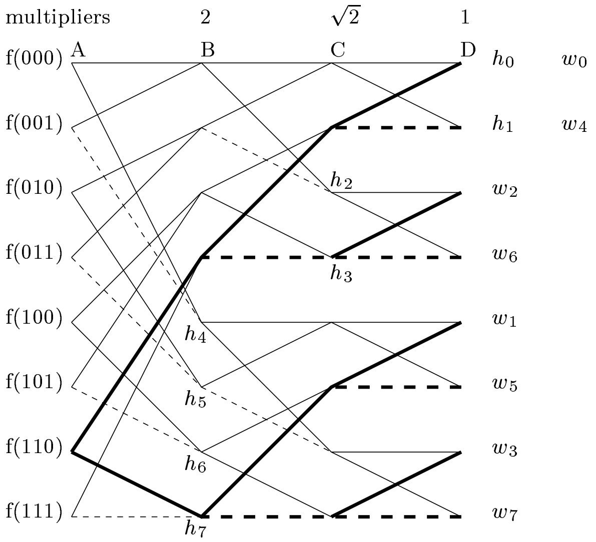

Figure 2: Fast Walsh-Haar transform.

4.1 Fast Haar Transform

Fast Haar transforms can be implemented with on the order of at most 2^ℓ computations. More precisely, Roeser and Jernigan's fast Haar transform requires $2(N-1)$ operations [13], where $N = 2^\ell$. A flow diagram of their algorithm, with $\ell = 3$, is shown in Figure 1. Each term from the leftmost column is either added to (solid line) or subtracted from (dashed line) its adjacent term to produce a new term for a subsequent stage of the algorithm.

Some operations have multiplying factors greater than one and their values can be seen over the corresponding joining lines. The flow diagram going from left to right produces the Haar coefficients (after multiplying by $\frac{1}{2^\ell}$, according to Equation 3), while the opposite direction produces the value of the function for any point $x \in [0, 2^\ell - 1]$.

For example, to compute h_2, we locate it in the figure, and read from right to left to obtain:

$$h_2 = \frac{1}{8}[\sqrt{2}(f(000) + f(001)) - \sqrt{2}(f(010) + f(011))].$$

To compute $f(010)$, we read from left to right (thick line):

$$f(010) = h_0 + h_1 - \sqrt{2}h_2 + 2h_5.$$

Note that the numbers of terms in the expansions of h_2 and $f(010)$ are consistent with Results 1 and 2, respectively, from the previous section.

We also note that the total number of operations (additions and subtractions) in Figure 1, is (going from left to right), 8 on the first layer, 4 on the second, and 2 on the last, i.e., 14, which is equal to $2(2^\ell - 1)$ for $\ell = 3$.

Note that the coefficients do not occur in sequential order. When $\ell = 3$, reading from top to bottom, the order is: $h_0, h_4, h_2, h_5,\ h_1, h_6, h_3, h_7$. To locate a coefficient, the following equation is used [13]:

$$2^{(\ell - q)}(2^{-1} + m)$$

for $0 < m < 2^q - 1$ and $0 < q < \ell$ where q is the degree, and m the member to be determined. For instance, if $\ell = 3$, then the location of h_5, i.e., h_{2^2+1}, is $2^{(3-2)}(2^{-1} + 1) = 3$. In the next section, we show an algorithm that gives both Walsh and Haar transforms.

Strings		*Fitness*
1111	0000	0
1110	0001	16
1101	0010	31
1011	0100	29
1010	0101	25
1001	0110	39
1000	0111	23

Table 2: Binary strings with their fitness values.

4.2 The Fast Walsh-Haar Transform

Either the Walsh or the Haar expansion can be obtained from Fino's composite Walsh-Haar transform [4]. The transform is presented by the flow diagram of Figure 2 for $\ell = 3$. The multiplying constants employed in the routine for the Haar transform are shown in the diagram as the constants

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
w_j	398	0	0	-46	0	-82	-86	0	0	-2	-118	0	-66	0	0	2
h_j	398	0	$-33\sqrt{2}$	$33\sqrt{2}$	-102	-16	16	102	$-32\sqrt{2}$	$-10\sqrt{2}$	$8\sqrt{2}$	$32\sqrt{2}$	$-32\sqrt{2}$	$-8\sqrt{2}$	$10\sqrt{2}$	$32\sqrt{2}$

Table 3: Binary strings with their fitness values.

associated with the levels. At level B, the multiplier is 2, at level C the multiplier is $\sqrt{2}$ and at level D, the multiplier is 1. Each term from the leftmost column is either added to (solid line) or subtracted from (dashed line) its adjacent term to produce a new term of a subsequent stage of the algorithm. The flow diagram going from left to right produces the Haar coefficients (after multiplying by $\frac{1}{2^\ell}$ and the appropriate level multiplier) or the Walsh coefficients (after multiplying by $\frac{1}{2^\ell}$). Going in the opposite direction produces the value of the function for any point $x \in [0, 2^\ell - 1]$. For instance, going from right to left:

$$h_7 = 2[f(110) - f(111)]\frac{1}{8}.$$

And

$$h_2 = \sqrt{2}[f(000) + f(001) - f(010) - f(011)]\frac{1}{8}.$$

Going from left to right, we can compute $f(110)$ in terms of Haar coefficients:

$$f(110) = h_0 - h_1 - \sqrt{2}h_3 + 2h_7,$$

or as a linear combination of Walsh coefficients:

$$f(110) = w_0 - w_4 - w_2 + w_6 + w_1 - w_5 - w_3 + w_7.$$

We note that the total number of operations to get Walsh coefficients, in Figure 2, (additions and subtractions), is 2^3 in each of the 3 layers. i.e., 24, which is equal to $3(2^\ell)$ for $\ell = 3$. The Haar coefficients are obtained much faster than the Walsh. Indeed, Haar requires $2(2^\ell - 1)$ operations compared to Walsh's $\ell(2^\ell)$. Thus, $\ell 2^\ell - 2^{\ell+1} + 2$ more operations are needed with Walsh than with Haar functions. For instance, for $\ell = 10$, we need to perform $8,192$ more operations when the Walsh fast transform is used instead of the Haar transform.

It is generally more advantageous to use the Haar transform rather than its Walsh counterpart. However, the next section examines a special case for which the latter might be preferable.

5 Symmetrical Fitness Functions

If $f(x)$ is a symmetrical function, then many Walsh coefficients will vanish while the Haar coefficients will have non-zero values. This is due to the symmetry properties of Walsh transforms. More precisely, the proof of the following result [14] can be found in the Appendix.

Result 3 Let $f(x)$ be a symmetric function, i.e., $f(x) = f(\bar{x})$, where $\bar{x}_i = 1 - x_i$. Then every Walsh coefficients w_j that contains an odd number of ones in its index j, is zero.

The maximum cut problem illustrates this situation. Given a weighted graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ is the set of vertices, E the set of edges, and w_{ij} the weight of the edge between vertices i and j, the problem consists in partitioning V into two disjoint sets V_0 and V_1, such that the sum of the weights of the edges from E that have one endpoint in V_0 and the other in V_1 is maximized.

Example 1 Consider the instance of the maximum cut problem with four vertices, and five edges with the following weights: $w_{12} = w_{21} = 6$, $w_{13} = w_{31} = 10$, $w_{23} = w_{32} = 11$, $w_{24} = w_{42} = 15$, $w_{34} = w_{43} = 8$, and $w_{ij} = 0$ for all other values of i and j. If we let $x_i = 0$ when vertex i is in V_0 and $x_i = 1$ when i is in V_1, then each string in $\{0,1\}^4$ will represent a possible partition of the vertices. Since $w_{ij} = w_{ji}$ for $1 \le i, j \le 4$, the fitness function to maximize is given by ([11]):

$$f(x) = \Sigma_{i=1}^{n-1} \Sigma_{j=i+1}^{n} \left[x_i(1 - x_j) + x_j(1 - x_i)\right] w_{ij}$$

with $n = 4$.

The function is symmetrical and every binary string has the same fitness value as its complement. For instance, $f(1010) = f(0101) = 25$. Table 2 gives all possible solutions with their corresponding fitness values.

The Walsh and Haar coefficients tabulated in Table 3 have to be multiplied by $\frac{1}{16}$.

We note that although it is more efficient to compute the Haar coefficients, it might still be preferable to use the Walsh transform in this problem, since half of the coefficients drop out while all but one of the Haar coefficients have non-zero value, and since, more importantly, ℓ is very small.

6 Conclusion

The aim of this article is to bring to the attention of researchers involved in theoretical aspects of genetic algorithms the existence and some of the properties of Haar functions. More precisely, these functions can be used as a tool, to compute fitness averages, to compute schemata, and to construct deceptive functions. Because of the computational advantages, Haar functions might be used for certain applications instead of Walsh functions.

Using Haar functions a basis, the expansion of the fitness function requires substantially fewer terms than with Walsh functions. The Haar coefficients too can be written as linear combinations of very few points that are "close together" and do not require the computation of all points, as is the case with Walsh coefficients. The computational difference between Walsh and Haar functions can also be seen by considering fast transforms. With these implementations, modeled after the fast Fourier transform, Haar coefficients require $O(N)$ transformations, whereas $O(N\, log_2 N)$ transformations are required for Walsh, where $N = 2^\ell$. We showed that the difference between the total number of terms required for the computation of the expansions of Walsh and of Haar remains exponential in ℓ (of order $\ell 2^\ell$).

Acknowledgments

The author would like to thank Dr. Frederick Stern for many discussions on the topic, and one of the reviewers' suggestions that have improved the clarity and style of this work. He would also like to thank Stefan Bischof and Helko Lehmann for getting the paper in camera–ready form.

References

[1] Beauchamp K. G., (1984), *Applications of Walsh and Related Functions*, Academic Press.

[2] Bethke , A. D., (1980), *Genetic Algorithms as Function Optimizers*, Doctoral dissertation, University of Michigan.

[3] Daubechies, I., (1992), *Ten Lectures on Wavelets*, Philadelphia: SIAM, 1992.

[4] Fino, B.J., (1972), *Relations Between Haar and Walsh-Hadamard Transforms*, Proc. IEEE (Letters), vol 60, pp. 647-648.

[5] Goldberg, D.E., (1989), *Genetic Algorithms and Walsh Functions Part II: Deception and its Analysis*, Complex Systems 3, pp. 153-171.

[6] Haar, A., (1910), *Zur Theorie der Orthogonalen Funktionensysteme*, Math. Annalen 69:331-371.

[7] Harmuth, H. F., (1968), *A Generalized Concept of Frequency and Some Applications*, IEEE Transactions on Information Theory, IT-14:375-382.

[8] Holland, J.H., (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.

[9] Karpovsky, M.G., (1985), *Spectral Techniques and Fault Detection*, Academic Press.

[10] Khuri S (1994) Walsh and Haar Functions in Genetic Algorithms. *Proceedings of the 1994 ACM Symposium of Applied Computing* (New York: ACM Press) 201–5.

[11] Khuri, S., Bäck, T., and Heitkötter, J., (1994), *An Evolutionary Approach to Combinatorial Optimization Problems*, Proc. of the 22nd. ACM Computer Science Conf., Phoenix, Arizona.

[12] Kremer, H., (1973), *On Theory of Fast Walsh Transform Algorithms*, Colloquium on the Theory and Applications of Walsh and Other Non-Sinusoidal Functions, Hatfield, U.K.

[13] Roeser, P. R., and Jernigan, M. E., (1982), *Fast Haar Transform Algorithms*, IEEE Transactions on Computers C-31 no 2:175-177.

[14] Rudolf, G., (1994), Private communication reproduced in the Appendix.

Appendix

Result 3
Let $f(x)$ be a symmetric function, i.e., $f(x) = f(\bar{x})$, where $\bar{x}_i = 1 - x_i$. Then every Walsh coefficients w_j that contains an odd number of ones in its index j, is zero.
Note that $1 - 2\,\bar{x}_i = 1 - 2\,(1 - x_i) = (-1) \cdot (1 - 2\,x_i)$.
The Walsh coefficient w_j is calculated by

$$
\begin{aligned}
w_j &= \sum_{x=0}^{2^l-1} f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \\
&= \sum_{x=0}^{2^{l-1}-1} f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \\
&\quad + \sum_{x=2^{l-1}}^{2^l-1} f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \\
&= \sum_{x=0}^{2^{l-1}-1} f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \\
&\quad + \sum_{x=0}^{2^{l-1}-1} f(\bar{x}) \prod_{i=1}^{l} (1 - 2\,\bar{x}_i)^{j_i} \\
&= \sum_{x=0}^{2^{l-1}-1} \left[f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} + f(\bar{x}) \prod_{i=1}^{l} (1 - 2\,\bar{x}_i)^{j_i} \right] \\
&= \sum_{x=0}^{2^{l-1}-1} f(x) \left[\prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \right. \\
&\qquad \left. + \prod_{i=1}^{l} (-1)^{j_i} \cdot (1 - 2\,x_i)^{j_i} \right] \\
&= \sum_{x=0}^{2^{l-1}-1} f(x) \left[\left(\prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \right) \cdot \left(1 + \prod_{i=1}^{l} (-1)^{j_i} \right) \right] \\
&= 2 \sum_{x=0}^{2^{l-1}-1} f(x) \left[\left(\prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} \right) (1 - \|j\|_1 \bmod 2) \right] \\
&= \begin{cases} 2 \cdot \sum_{x=0}^{2^{l-1}-1} f(x) \prod_{i=1}^{l} (1 - 2\,x_i)^{j_i} & \text{if } \|j\|_1 \text{ even} \\ 0 & \text{if } \|j\|_1 \text{ odd} \end{cases}
\end{aligned}
$$

where $\|j\|_1 = \sum_{i=1}^{l} j_i$ denotes the l_1–norm of j in binary notation. □