Chapter 3
Configurations in the n-Dimensional Orthogonal Pseudo-Polytopes

3.1. Introduction

The problem of enumerating the colorings of an \(n\)D hypercube with two colors has an interesting history. Mathematicians in the 1870’s were considering classifying the logical propositions that can exist. In Boolean Algebra, in the functions of \(n\) logical variables, each variable can assume a value of either True or False, leading to \(2^n\) combinations in the domain of the function. If we associate a spatial dimension in \(\mathbb{R}^n\) with each variable (True \(\equiv 1\) and False \(\equiv 0\)), then each element of the domain can be represented as a point with \(n\) coordinates of 0’s and 1’s. Attempts to enumerate equivalence classes of functions were carried out in the 1800’s, apparently without realizing that 2-colorings of the \(n\)D hypercube were also being counted. In 1871, W. S. Jevons presented an article that described the cases that arise in 2-colorings of the 2D cube and the 3D cube, but from the perspective of Boolean functions. Jevons only considered 192 of the 256 propositions corresponding to 2-colorings of the 3-cube; he found 16 of the 22 equivalence classes [Banks04]. In 1877, Clifford claimed a solution of counting cases for 2-colorings of functions. His results were showed incorrect until 1940 when Pólya demonstrated how his enumeration methods could be applied to this problem in an automated manner. He enumerated the 402 classes of 2-colorings of the 4D hypercube, and showed where Clifford’s manual tabulations had overcounted in some cases and undercounted in others [Pólya87]. In 1987 this topic was revisited, in the 2D and 3D cases, by Lorensen and Cline through their famous Marching Cubes Algorithm [Lorensen87]. Finally, the 4D case has been studied by Roberts & Hill [Roberts99], Bhaniramka et al [Bhaniramka00] and Banks & Stockmeyer [Banks06].

The problem described above has direct incidence with the problem we will deal in this chapter. In fact, it is its dual: the determination of the configurations in the \(n\)-Dimensional Orthogonal Pseudo-Polytopes. This chapter will describe some relations and strategies that could support us in the task of obtaining in a more direct way these configurations. In order to speed up the determination of the topological equivalence between a pair of configurations, we describe relations whose implementation compares any two configurations in a time which only depends on the number of hyper-octants in the space in which their hyper-boxes are embedded. We will show that our relations are in fact equivalence relations which are ‘wider’ than the classical equivalence relation based in geometrical transformations and therefore they provide an approximate solution to our problem.

3.2. Configurations for the \(n\)D-OPP’s and the Equivalence Relation \(R_f\)

A set of quasi-disjoint \(n\)D hyper-boxes determines an \(n\)D-OPP whose vertices must coincide with some of the hyper-boxes’ vertices [Aguilera02]. We consider the hyper-boxes’ vertices as the origin of an \(n\)D local coordinate system, and they may belong up to \(2^n\) hyper-boxes, one for each local hyper-octant. The \(n\)D-OPP’s vertices are determined according to the presence or absence of each of these \(2^n\) surrounding hyper-boxes. In this work, an adjacency relation (or just adjacency) between two hyper-boxes refers to the intersection of those hyper-boxes [Aguilera02]. The \(n\) possible adjacency relations between the \(2^n\) possible hyper-boxes can be of \(\Pi_0\) (vertex adjacency), \(\Pi_1\) (edge adjacency), ..., \(\Pi_{n-1}\) ((n-2)D adjacency) or \(\Pi_{n,1}\) ((n-1)D adjacency). There is only one adjacency between any two hyper-boxes (Appendix C lists some properties which are related with the adjacencies between hyper-boxes). There are \(2^{n^2}\) [Hill98] possible combinations, which according to an equivalence relation, can be grouped in equivalence classes also called configurations [Pérez-Aguila01].

Let \(f^o\) be the set of linear transformations in \(\mathbb{R}^n\) generated by all possible compositions of reflections and rotations and their inverses, in any order, with repetition of these geometric transformations allowed. The well known group \(f^o\) is called the Octahedral Symmetry Group, the 4.3.2 Symmetry Group [Mortenson99] or the Holo-Octahedral Group [Conway03]. According to [Banks04], \(f^o\) is sometimes referred in the literature as the Hyperoctahedral Group. \(f^o\) is the wreath product of a reflection with the permutations of the axes in \(\mathbb{R}^n\) and its cardinality is given by Card(\(f^o\)) = \(2^n n!\) [Cohn84]. (See the last author in order to find more details about the wreath product and the determination of the number of elements in \(f^o\)). It is well known that \(f^o\) forms a group [Coxeter63].
Let $x$ and $y$ be two combinations of $n$D hyper-boxes. Then, we can define the equivalence relation $R_f$ as $x \, R_f \, y \iff (\exists f \in F)(f(x) = y)$. Under this equivalence relation, we have that the $2^{2^n}$ possible combinations of hyper-boxes can be grouped into at least $2^{2^{n-1}}$ configurations [Ziegler94]. In the cases for the 1D-OPP’s, 2D-OPP’s (section 2.3.3.1), 3D-OPP’s (section 2.3.3.2) and 4D-OPP’s (section 2.3.3.3) we have 3, 6, 22 [Aguilera98] and 402 [Hill98] configurations respectively under equivalence relation $R_f$.

### 3.3. The Problem of Determining the Configurations for the nD-OPP’s ($n > 4$)

For the Euclidean $n$-Dimensional space we have $2^n$ possible hyper-octants (4 quadrants for 2D space, 8 octants for 3D space, and 16 hyper-octants for 4D space). This number of hyper-octants has repercussion over the possible number of combinations of vertices described through the presence or absence of hyper-boxes each one in every hyper-octant. In 4D space we have $2^{16} = 65,536$ possible combinations. In [Hill98] it is determined that there are 402 configurations for 4D-OPP’s through the relation $R_f$. However, if we want to determine the configurations for 5D Orthogonal Pseudo-Polytopes (5D-OPP’s) through exhaustive searching, we would have to consider that there are 32 hyper-octants in 5D space, and for instance, to analyze $2^{2^5} = 4,294,967,296$ combinations.

Moreover, if the number of configurations is associated with the total number of combinations, it is evident that the first one is minor than the second one. For example, in 3D space we have 22 configurations for 256 possible combinations; this can be translated as that only the 8% of the combinations can perform the role of representatives [Pérez-Aguila05]. See Table 3.1 for the application of this comparison over the configurations in 1D, 2D, 3D and 4D spaces.

<table>
<thead>
<tr>
<th>nD Space</th>
<th>Combinations</th>
<th>Configurations</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D</td>
<td>4</td>
<td>3</td>
<td>75 %</td>
</tr>
<tr>
<td>2D</td>
<td>16</td>
<td>6</td>
<td>37.5 %</td>
</tr>
<tr>
<td>3D</td>
<td>256</td>
<td>22</td>
<td>8 %</td>
</tr>
<tr>
<td>4D</td>
<td>65,536</td>
<td>402</td>
<td>0.6 %</td>
</tr>
</tbody>
</table>

Table 3.1. Percentages between the number of combinations and configurations for the nD-OPP’s.

These situations lead us to conclude that the complexity imposed by the exhaustive searching makes difficult to determine the configurations for OPP’s in spaces of 5 dimensions and beyond ([Hill98] & [Banks03]) In the following sections we will describe some tools that could support us in the task of obtaining the configurations in a more direct way. We will introduce equivalence relations that will approximate the configurations obtained respect to equivalence relation $R_f$. Moreover, we will introduce a scheme for representing combinations of hyper-boxes which is based in binary strings. The new equivalence relations together with the binary representation will allow us to compare two combinations of hyper-boxes in a time which only depends of the number of hyper-octants in the space in which the hyper-boxes are embedded. Some of the equivalence classes that are produced by our relations are also equivalence classes under the relation $R_f$. However, there exist equivalence classes in $R_f$ whose union comprises an equivalence class under our relations. This last property will allow us to conclude that the partition, induced by the proposed equivalence relations, produces an approximation of the partition induced by relation $R_f$, but with the advantage of temporal complexity reduced considerably.

### 3.4. Binary Representation for the Configurations in the nD-OPP’s

**Definition 3.1:** [Pérez-Aguila05]: *Consider set $G=\{0,1\}$. The set of vectors $B^n$, $n \geq 1$, is defined as:*

$$B^n = \underbrace{G \times \ldots \times G}_{2^n} = \{(x_1, ..., x_n) \mid x_i \in G, i = 1, ..., 2^n\}$$

An nD-OPP’s combination of hyper-boxes can be represented through a vector in the set $B^n$ [Aguilera04]. The positions of its scalars will indicate the nD space's hyper-octants. If a scalar has a value equal to one then its referred hyper-octant is occupied by a hyper-box; otherwise, the hyper-octant is empty. Since we will have $2^n$ scalars in the vector, the position of each scalar can be interpreted as a binary number with $n$ digits ($0 \ldots 0_2 \ldots 1 \ldots 1_2$). These $n$ digits will be associated with each one of the nD space's main axes by considering the most significant bit as a reference to the $X_1$ axis, the subsequent bit as a reference to the $X_2$ axis, and so forth until we consider the least significant bit as a reference to the $X_n$ axis. Moreover, if a bit is 0 then we will consider the positive part of the
corresponding axis; otherwise, we will consider its negative part. Then, through the binary representation of the position of a bit in the combination's vector we can infer its corresponding hyper-octant. For example, if the 22-th bit in a 5D configuration's vector has a value equal to one, then we infer that there is a hyper-box in the hyper-octant \( x_a \cdot x_b \cdot x_c \cdot x_d \) because \( 22_{10} = 10110_2 \).

**Appendix A** explores some properties of the set \( B^n \). In fact the set \( B^n \) is a Vector Space under given definitions of vector addition and scalar multiplication.

**Theorem 3.1**: The set \( B^n \) is a vector space over the field \((G, XOR, AND)\).
Proof: Refer to **Appendix A**.

**Definition 3.2**: Let \( A^n \subseteq B^n \) be the set of vectors that contains the \( 2^n \) permutations of \((1,0,...,0)\).

**Theorem 3.2**: The set of vectors \( A^n \) is linearly independent.
Proof: Refer to **Appendix A**.

**Theorem 3.3**: The set \( A^n \subseteq B^n \) forms a basis for \( B^n \).
Proof: Refer to **Appendix A**.

Before going any further with linear transformations in the vector space \( B^n \), let's discuss the way we can apply certain geometric transformations in \( \mathbb{R}^n \) to a combination of hyper-boxes with its corresponding vector in \( B^n \). The process, originally presented in [Aguilera04], is in fact very simple. We consider only rotation angles of 90°, 180° and 270° of main axis \( X_a \) in direction to the main axis \( X_b \) (the plane defined by axis \( X_a \) and \( X_b \) is defined as rotation plane [Hollasch91]); and reflections respect to the main axes \( X_1, ... X_n \). Each one of the hyper-boxes (i.e. occupied hyper-octants) in a combination is referred through their descriptive axes. Now, each one of the hyper-octant’s axes will be associated with a coordinate, if we are considering an axis’ negative part then the corresponding coordinate will have a value equal to –1; otherwise, the corresponding coordinate is equal to 1. By this way, a hyper-octant will be related with a point. For example, the associated point for the hyper-octant \( x_a \cdot x_b \cdot x_c \cdot x_d \) is \((-1, 1, -1, 1)\). Table 3.2 shows the hyper-octants and their corresponding points in the 4D space.

<table>
<thead>
<tr>
<th>Hyper-octant descriptive axes</th>
<th>Point</th>
<th>Hyper-octant descriptive axes</th>
<th>Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, 1, 1, 1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, 1, 1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, 1, -1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, 1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, -1, 1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, -1, -1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, -1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, 1, 1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, -1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, 1, -1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(1, -1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, 1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, 1)</td>
</tr>
<tr>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, -1)</td>
<td>( x_a \cdot x_b \cdot x_c \cdot x_d )</td>
<td>(-1, -1, -1)</td>
</tr>
</tbody>
</table>

For applying a geometric transformation we must obtain the corresponding points of each hyper-octant and transform them according to the specific rotations and/or reflections. It can be easily verified that our specific transformations will preserve the values of the coordinates of a hyper-octant’s corresponding point in –1 or 1. This indicates that, after the application of these transformations, the new coordinates for a point will have a new hyper-octant associated and, therefore, its corresponding hyper-box has been placed in a new position. Finally, a new string that represents the transformed configuration is obtained. Such new string is in fact a permutation of the scalars in the original string.

Now, we will define our working set of linear transformations in \( B^n \) by mapping them from its associated transformations in \( \mathbb{R}^n \). 

Table 3.2. The 4D space’s hyper-octants and their corresponding points.
Chapter 3 - Configurations in the n-Dimensional Orthogonal Pseudo-Polytopes

**Definition 3.3:** Consider a scalar \( s \in \{-1, 1\} \). Such scalar will be associated with a binary digit \( A(s) \) according to the following rule:

\[
A(s) = \begin{cases} 
0 & \text{iff } s = 1 \\
1 & \text{iff } s = -1 
\end{cases}
\]

**Definition 3.4:** Let \( x = (x_1, \ldots, x_n) \in \{ (\pm 1, \ldots, \pm 1) \} \subset \mathbb{R}^n \). Vector \( x \) will be associated to a binary string \( b(x) \) whose characters will determined and concatenated according to the rule:

\[
b(x) = b(x_1, \ldots, x_n) = A(x_1)A(x_2) \ldots A(x_n)
\]

The binary string \( b(x) \) describes a binary digit in the interval \([0\ldots2^n, 1\ldots2^n]_\alpha \) because \( b(-1,\ldots,-1) = A(-1)\ldots A(-1) = 1\ldots1 \)

and \( b(1,\ldots,1) = A(1)\ldots A(1) = 0\ldots0 \).

For example, let \( x = (-1,1,1,-1,-1,1,1,1) \), then

\[
b(x) = A(-1)A(1)A(1)A(-1)A(-1)A(-1)A(1)A(1) = 10011100
\]

**Definition 3.5:** Let \( b(x) \) be the binary number associated to the vector \( x = (x_1, \ldots, x_n) \in \{ (\pm 1, \ldots, \pm 1) \} \subset \mathbb{R}^n \). The vector \( x \)

will be associated with a vector in the basis \( A^n \) of vector space \( B^n \) through the function \( m(x) \) defined in the following way:

\[
m: \ (\pm 1, \ldots, \pm 1) \subset \mathbb{R}^n \rightarrow \quad A^n \subseteq B^n
\]

\[
x \sim m(x) = (0,0,0,1,0,0,0,0,0)
\]

That is, vector \( x \) will be associated to such vector in the basis \( A^n \) of \( B^n \) such that its scalar in the position \( b(x) \) is equal to one.

For example, consider the vertices of a cube centered at the origin. The correspondence between the vertices and the vectors in the basis \( A^3 \) of \( B^3 \) is the following:

\[
\begin{array}{cccc}
(1, 1, 1) & b(1, 1, 1) = 000 & m(1, 1, 1) = (1, 0, 0, 0, 0, 0, 0, 0) \\
(1, 1, -1) & b(1, 1, -1) = 001 & m(1, 1, -1) = (0, 1, 0, 0, 0, 0, 0, 0) \\
(1, -1, 1) & b(1, -1, 1) = 010 & m(1, -1, 1) = (0, 0, 1, 0, 0, 0, 0, 0) \\
(1, -1, -1) & b(1, -1, -1) = 011 & m(1, -1, -1) = (0, 0, 0, 1, 0, 0, 0, 0) \\
(-1, 1, 1) & b(-1, 1, 1) = 100 & m(-1, 1, 1) = (0, 0, 0, 0, 1, 0, 0, 0) \\
(-1, 1, -1) & b(-1, 1, -1) = 101 & m(-1, 1, -1) = (0, 0, 0, 0, 0, 1, 0, 0) \\
(-1, -1, 1) & b(-1, -1, 1) = 110 & m(-1, -1, 1) = (0, 0, 0, 0, 0, 0, 1, 0) \\
(-1, -1, -1) & b(-1, -1, -1) = 111 & m(-1, -1, -1) = (0, 0, 0, 0, 0, 0, 0, 1)
\end{array}
\]

**Definition 3.6:** Let \( M_\alpha : \mathbb{R}^n \rightarrow \mathbb{R}^n \) be the reflection respect to \( X_\alpha \)-axis. \( M_\alpha \) has the matrix representation:

\[
M_\alpha = \begin{bmatrix}
m_{a} & m_{a} & m_{\alpha'} \\
-1 & -1 & \alpha' \\
0 & 0 & 0
\end{bmatrix}
\]

For all \( x \in \{ (\pm 1, \ldots, \pm 1) \} \subset \mathbb{R}^n \) we have that \( M_\alpha x \in \{ (\pm 1, \ldots, \pm 1) \} \). Let \( \alpha = m(x) \) and \( \alpha' = m(M_\alpha x) \). By considering to each \( \alpha, \alpha' \in A^n \subset B^n \) we can define a linear transformation \( \varphi_\alpha : B^n \rightarrow B^n \) such that \( \varphi_\alpha (\alpha) = \alpha' \). The transformation \( \varphi_\alpha \) will be called \( (a) \)-reflection in the vector space \( B^n \).

For example, consider vector space \( \mathbb{R}^3 \) and the vertices of a cube centered at the origin:

\[
\{(1,1,1), (1,1,-1), (1,-1,1), (-1,1,1), (-1,1,-1), (-1,-1,1), (-1,-1,-1)\}
\]
Consider the matrix representation of the reflection respect to $X_1$ axis

\[
M_1, x = \begin{bmatrix}
-1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

By applying such transformation to the vertices of the cube we obtain:

- $M_1(1, 1, 1) = (-1, 1, 1)$
- $M_1(-1, 1, 1) = (1, 1, 1)$
- $M_1(1, 1,-1) = (-1, 1,-1)$
- $M_1(-1, 1,-1) = (1, 1,-1)$
- $M_1(1,-1, 1) = (-1,-1, 1)$
- $M_1(-1,-1, 1) = (1,-1, 1)$
- $M_1(1,-1,-1) = (-1,-1,-1)$
- $M_1(-1,-1,-1) = (1,-1,-1)$

Let $\phi_1 : B^3 \to B^3$ the linear transformation such it satisfies:

- $\phi_1(m(1, 1, 1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(1, 1, 1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(1, 1,-1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(1, 1,-1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(1,-1, 1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(1,-1, 1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(-1, 1, 1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(-1, 1, 1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(-1, 1,-1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(-1, 1,-1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(-1,-1, 1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(-1,-1, 1) = (0,0,0,0,0,0,0,0,0)$
- $\phi_1(m(-1,-1,-1)) = \phi_1(0,0,0,0,0,0,0,0,0) = m(-1,-1,-1) = (0,0,0,0,0,0,0,0,0)$

\[\therefore \phi_1(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (x_5, x_6, x_7, x_8, x_1, x_2, x_3, x_4)\]

The reflections $\phi_2$ and $\phi_3$ in $B^3$ are defined in similar way by starting from reflections $M_2$ and $M_3$ in $R^3$ respectively.

**Definition 3.7:** Let $R_{a,b}(\theta) : R^n \to R^n$, $n \geq 2$, be the rotation of the $X_a$-axis in direction of $X_b$-axis. $R_{a,b}(\theta)$ has the matrix representation [Duffin94]:

\[
R_{a,b}(\theta) = \begin{bmatrix}
\cos \theta & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

By doing $\theta = 90^\circ$ we have the corresponding matrix representation

\[
R_{a,b}(90^\circ) = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

We will refer to $R_{a,b}(180^\circ)$ and $R_{a,b}(270^\circ)$ through $R_{a,b}(90^\circ) = R_{a,b}(90^\circ) \circ R_{a,b}(90^\circ)$ and $R_{a,b}^3(90^\circ) = R_{a,b}^2(90^\circ) \circ R_{a,b}(90^\circ)$ respectively.

**Definition 3.8:** For all $x \in \{(\pm 1, \ldots, \pm 1)\} \subseteq R^n$ we have that $R_{a,b}(90^\circ)x \in \{(\pm 1, \ldots, \pm 1)\}$. Let $\alpha = m(x)$ and $\alpha' = m(R_{a,b}(90^\circ)x)$. By considering to each $\alpha, \alpha' \in A^n \subseteq B^n$ we can define a linear transformation $\rho_{a,b} : B^n \to B^n$ such that it satisfies $\rho_{a,b}(\alpha) = \alpha'$. $\rho_{a,b}$ will be called (a,b)-rotation in the vector space $B^n$.

For example, consider vector space $R^3$ and the vertices of a cube centered at the origin:

\{(1,1,1), (1,1,-1), (1,-1,1), (1,-1,-1), (-1,1,1), (-1,1,-1), (-1,-1,1), (-1,-1,-1)\}
Consider the matrix representation of the rotation of \(X_1\)-axis in direction of the \(X_2\)-axis:

\[
R_{1,2}(90^\circ)x = \begin{bmatrix}
0 & 1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}x
\]

By applying such transformation to the vertices of the cube we obtain:

\[
R_{1,2}(90^\circ)(1, 1, 1) = (-1, -1, 1) \quad R_{1,2}(90^\circ)(-1, 1, 1) = (1, 1, 1)
\]

\[
R_{1,2}(90^\circ)(1, -1, 1) = (-1, 1, 1) \quad R_{1,2}(90^\circ)(-1, -1, 1) = (1, -1, 1)
\]

\[
R_{1,2}(90^\circ)(1, -1, -1) = (-1, 1, -1) \quad R_{1,2}(90^\circ)(-1, -1, -1) = (1, -1, -1)
\]

Let \(\rho_{1,2}: B^3 \rightarrow B^3\) the linear transformation such it satisfies:

\[
\rho_{1,2}(m(1,1,1)) = \rho_{1,2}(m(-1,-1,-1)) = (0,0,1,0,0,0,0,0,0)
\]

\[
\rho_{1,2}(m(1,1,-1)) = \rho_{1,2}(m(-1,1,-1)) = (0,0,0,0,0,0,1,0,0)
\]

\[
\rho_{1,2}(m(1,-1,1)) = \rho_{1,2}(m(-1,-1,1)) = (0,0,0,0,1,0,0,0,0)
\]

\[
\rho_{1,2}(m(1,-1,-1)) = \rho_{1,2}(m(-1,1,1)) = (0,0,0,0,0,0,1,0,0)
\]

\[
\rho_{1,2}(m(-1,1,1)) = \rho_{1,2}(m(-1,1,-1)) = (0,0,0,0,0,0,1,0,0)
\]

\[
\rho_{1,2}(m(-1,-1,1)) = \rho_{1,2}(m(-1,-1,-1)) = (0,0,0,0,0,0,1,0,0)
\]

\[
\therefore \rho_{1,2}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = (x_5, x_6, x_1, x_2, x_6, x_7, x_3, x_4)
\]

The rotations \(\rho_{1,2}^1\) and \(\rho_{1,2}^3\) in \(B^3\) are defined in similar way by starting from reflections \(R_{1,2}^1(90^\circ)\) and \(R_{1,2}^3(90^\circ)\) respectively or through \(\rho_{1,2}^1 = \rho_{1,2} \circ \rho_{1,2}\) and \(\rho_{1,2}^3 = \rho_{1,2}^3 \circ \rho_{1,2}\).

Previously we mentioned the group of transformations \(f^a\) under the context of vector space \(\mathbb{R}^n\). Now, we will redefine it formally in the context of our vector space \(B^n\). Hence, in this chapter, when we refer to the group \(f^a\) we are referring to the group specified according to the following:

**Definition 3.9:** Let \(F^a\) the group of linear transformations generated by all possible compositions of reflections and rotations in the set \(B^n\) and their inverses, in any order, with repetition of these transformations allowed. Hence \(f^a = \{ \varphi_i | \varphi_i is a (i)-reflection in \ B^n, i = 1, ..., n \} \cup \{ \rho_{ij} | \rho_{ij} is a (i,j)-rotation in \ B^n, i = 1, ..., n, i < j < n \}\)

### 3.5. Pólya’s Countings and the Number of Configurations for the nD-OPP’s

In this section we present a brief introduction to Pólya’s method and show how it can be used to compute the total number of configurations for the nD-OPP’s. A central concept in this method is the cycle index of a permutation group \(G\); in fact \(G\) is the transformations group to consider in the enumeration. This multivariable polynomial records the cycle structure of each permutation in \(G\) [Banks04]. In a precise manner, the cycle index \(Z\) of a group \(G\) is the polynomial

\[
Z(G; z_1, z_2, ..., z_d) = \frac{1}{\text{Card}(G)} \sum_{g \in G} \prod_{i=1}^{d} z_i^{j_i(g)}
\]

[Cohn84]. Where:

- \(j_i(g)\) is the number of cycles of length \(i\) in the permutation \(g \in G\)
- \(\text{Card}(G)\) is the cardinality of \(G\)
- \(d\) is the degree of \(G\)

In our case we will instantiate:

- \(G = f^a\)
- \(\text{Card}(G) = \text{Card}(f^a) = 2^n n!\)
- \(d\) is the number of elements of the set that acts as generator of the group \(f^a\): \(n\) main reflections, \(\binom{n}{2}\) main rotations and one identity transformation. Hence, \(d = \binom{n}{2} + n + 1\).

---

1. The author was introduced to Pólya’s Countings through a kindly electronic communication with William E. Lorensen (General Electric Research & Development).
In previous sections we stated that each one of the transformations in the set \( f^2 \) is in fact a permutation that acts over the scalars in a vector in the set \( B^n \). Representing our transformations as permutations will be very useful when we determine the cycle index of \( f^2 \). We will illustrate with the case \( n = 2 \) the way to compute the terms \( j(g) \) associated to each transformation in \( f^2 \). We have that \( f^2 = \{ \phi_1, \phi_2, \rho_{1,2}, \rho_{1,2}', \tau_1, \tau_2, t \} \), \( \text{Card}(f^2) = 8 \) and \( d = 4 \). Each transformation in \( f^2 \) is described in Table 3.3.

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_1(x_1, x_2, x_3, x_4) = (x_2, x_1, x_4, x_3) )</td>
<td>Reflection respect to ( x_1 )-axis</td>
</tr>
<tr>
<td>( \phi_2(x_1, x_2, x_3, x_4) = (x_3, x_4, x_1, x_2) )</td>
<td>Reflection respect to ( x_2 )-axis</td>
</tr>
<tr>
<td>( \rho_{1,2}(x_1, x_2, x_3, x_4) = (x_2, x_4, x_1, x_3) )</td>
<td>Rotation of ( x_1 )-axis in the direction of ( x_2 )-axis (90°)</td>
</tr>
<tr>
<td>( \rho_{1,2}'(x_1, x_2, x_3, x_4) = (x_4, x_3, x_2, x_1) )</td>
<td>Rotation of ( x_1 )-axis in the direction of ( x_2 )-axis (180°)</td>
</tr>
<tr>
<td>( \tau_1(x_1, x_2, x_3, x_4) = (x_4, x_2, x_3, x_1) )</td>
<td>Composition ( \tau_1 = \phi_1 \circ \rho_1 )</td>
</tr>
<tr>
<td>( \tau_2(x_1, x_2, x_3, x_4) = (x_1, x_3, x_2, x_4) )</td>
<td>Composition ( \tau_2 = \rho_1 \circ \phi_1 )</td>
</tr>
<tr>
<td>( t(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4) )</td>
<td>Identity transformation</td>
</tr>
</tbody>
</table>

Table 3.3. Description of linear transformations in the set \( f^2 \)

The Table 3.4 provides explicit details for computing each term \( z_1^{j_1(x)} z_2^{j_2(x)} z_3^{j_3(x)} z_4^{j_4(x)} \) in the cycle index of \( f^2 \).

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Associated Permutation</th>
<th>( j(g) )</th>
<th>( j(g) )</th>
<th>( j(g) )</th>
<th>( j(g) )</th>
<th>( z_1^{j_1(x)} z_2^{j_2(x)} z_3^{j_3(x)} z_4^{j_4(x)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi_1 )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 2 &amp; 1 &amp; 4 &amp; 3 \end{pmatrix} )</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>( z_1^0 z_2^0 z_3^0 z_4^0 )</td>
</tr>
<tr>
<td>( \phi_2 )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 3 &amp; 4 &amp; 1 &amp; 2 \end{pmatrix} )</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>( z_1^0 z_2^0 z_3^0 z_4^0 )</td>
</tr>
<tr>
<td>( \rho_{1,2} )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 2 &amp; 4 &amp; 1 &amp; 3 \end{pmatrix} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( z_1^0 z_2^0 z_3^0 z_4^1 )</td>
</tr>
<tr>
<td>( \rho_{1,2}' )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 4 &amp; 3 &amp; 2 &amp; 1 \end{pmatrix} )</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>( z_1^0 z_2^0 z_3^0 z_4^1 )</td>
</tr>
<tr>
<td>( \rho_{1,2}' )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 3 &amp; 1 &amp; 4 &amp; 2 \end{pmatrix} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( z_1^0 z_2^0 z_3^0 z_4^1 )</td>
</tr>
<tr>
<td>( \tau_1 )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 4 &amp; 2 &amp; 3 &amp; 1 \end{pmatrix} )</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( z_1^2 z_2^0 z_3^0 z_4^0 )</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 1 &amp; 3 &amp; 2 &amp; 4 \end{pmatrix} )</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( z_1^2 z_2^0 z_3^0 z_4^0 )</td>
</tr>
<tr>
<td>( t )</td>
<td>( \begin{pmatrix} 1 &amp; 2 &amp; 3 &amp; 4 \ 1 &amp; 2 &amp; 3 &amp; 4 \end{pmatrix} )</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( z_1^4 z_2^0 z_3^0 z_4^0 )</td>
</tr>
</tbody>
</table>

Table 3.4. Computing the terms of the cycle index polynomial of the group \( f^2 \)

Hence, the cycle index \( Z \) of the group \( f^2 \) is given by (Equation 3.1)

\[
Z(f^2, z_1, z_2, z_3, z_4) = \frac{1}{\text{Card}(f^2)} \sum_{g \in f^2} z_1^{j_1(g)} z_2^{j_2(g)} z_3^{j_3(g)} z_4^{j_4(g)}
\]

\[
= \frac{1}{8} \left( z_1^8 + z_2^8 + z_3^8 + z_4^8 + z_1^4 z_2^4 + z_1^4 z_3^4 + z_1^4 z_4^4 + z_1^2 z_2 z_3 z_4 + z_1^2 z_2 z_3 + z_1^2 z_2 z_4 + z_1^2 z_2 + z_1^2 z_3 + z_1^2 z_4 + z_1^2 + z_2^2 + z_3^2 + z_4^2 + z_2 + z_3 + z_4 + 1 \right)
\]

\[
= \frac{1}{8} \left( 3z_1^2 + 2z_2^4 + 2z_3^4 + z_4^4 \right)
\]
An elementary version of Pólya’s Theorem\(^2\) states that the number of inequivalent colorings of the hypercube, with \(k\) colors, is obtained by replacing each variable \(z_i\) in the cycle index of the group, in our case the group \(f_2^2\), with the integer \(k\) [Banks06]. Thus for \(k = 2\) colors, the number of inequivalent colorings of the square \((n = 2)\) is

\[
Z(f^2; 2, 2, 2, 2) = \frac{1}{8} (3 \cdot 2^2 + 2 \cdot 2^1 + 2 \cdot 2^2 \cdot 2^1 + 2^4) = \frac{48}{8} = 6
\]

Let us to confirm the expected results for the number of distinct colorings of the 3D cube, where we have:

- \(\text{Card}(f^3) = 3! \cdot 2^3 = 48\)
- \(d = 7\) because there are 3 reflections, 3 rotations and 1 identity transformation.

Hence the cycle index of the group \(f^3\) is given by

\[
Z(f^3; z_1, \ldots, z_7) = \frac{1}{\text{Card}(f^3)} \sum_{\alpha \in f^3} z_1^{\alpha (1)} \cdots z_7^{\alpha (7)} = \frac{1}{48} \left( z_1^6 + 6 z_1^4 z_2^2 + 13 z_1^2 z_3^2 + 51 z_2^4 z_4 + 32 z_2^2 z_6^2 + \right)
\]

(Equation 3.2)

By doing \(z_1 = \ldots = z_7 = 2\) we have the number of distinct colorings of the 3D cube:

\[
Z(f^3; 2, \ldots, 2) = \frac{1}{48} (2^8 + 6 \cdot 2^4 \cdot 2^1 + 13 \cdot 2^4 + 8 \cdot 2^2 \cdot 2^2 + 12 \cdot 2^2 + 8 \cdot 2 \cdot 2) = \frac{1,056}{48} = 22
\]

For determining the distinct colorings of the 4D hypercube we have:

- \(\text{Card}(f^4) = 4! \cdot 2^4 = 384\)
- \(d = 11\) (4 reflections, 6 rotations and 1 identity transformation)

The cycle index of the group \(f^4\) is given by

\[
Z(f^4; z_1, \ldots, z_{11}) = \frac{1}{\text{Card}(f^4)} \sum_{\alpha \in f^4} z_1^{\alpha (1)} \cdots z_{11}^{\alpha (11)} = \frac{1}{384} \left( z_1^{16} + 12 z_1^8 z_2^8 + 12 z_1^4 z_3^4 + 51 z_2^8 + 32 z_2^4 z_6^2 + \right)
\]

(Equation 3.3)

Now, by doing \(z_1 = \ldots = z_{11} = 2\), we can confirm the counting given by [Hill98] and [Banks06]:

\[
Z(f^4; 2, \ldots, 2) = \frac{1}{384} (2^{16} + 12 \cdot 2^8 \cdot 2^1 + 12 \cdot 2^4 \cdot 2^6 + 51 \cdot 2^8 + 32 \cdot 2^4 \cdot 2^2) = \frac{154,368}{384} = 402
\]

Now, we will expose the number of distinct colorings of the 5D hypercube to be expected (this case was not computed by [Roberts99] or [Banks06]):

- \(\text{Card}(f^5) = 5! \cdot 2^5 = 3,840\)
- \(d = 16\) (5 reflections, 10 rotations and 1 identity transformation)

We have the cycle index for \(f^5\) (Equation 3.4)

\[
Z(f^5; z_1, \ldots, z_{16}) = \frac{1}{\text{Card}(f^5)} \sum_{\alpha \in f^5} z_1^{\alpha (1)} \cdots z_{16}^{\alpha (16)} = \frac{1}{3840} \left( z_1^{32} + 20 z_1^{16} z_2^8 + 60 z_1^8 z_3^8 + 231 z_1^8 z_4^8 + 231 z_2^{16} + 80 z_1^4 z_5^4 + \right)
\]

Now, if \(z_1 = \ldots = z_{16} = 2\) we obtain:

\[
Z(f^5; 2, \ldots, 2) = \frac{4,716,126,720}{3,840} = 1,226,158
\]

This last result is related with another previously reported in [Aichholzer00] where were identified 1,226,525 equivalence classes of 0/1 polytopes of dimension 5D. The difference between our counting and Aichholzer’s counting arises from the fact that 0/1 polytopes with 0 to 4 vertices are not 5D polytopes but points, segments, triangles or 4D simplexes embedded in the 5D space. By considering the equivalence classes generated by these lower dimensional polytopes we get the same number of equivalence classes (for more details about 0/1 polytopes refer to [Aichholzer00] or [Pérez-Aguila03d]).

\[^2\] Pólya’s Theorem (Constant form) cited by [Palmer73]: “The number of orbits of the permutation group \(A\) acting on the set \(Y\) of functions from \(X\) to \(Y\) is obtained by replacing each variable \(z_i\) in the cycle index of the shape group \(A\) with the number \(k = \text{Card}(Y)\) of colors. That is, when the \(k\) colors are permuted but the vertices are, the number of cases (distinct colorings) is given by the reduced cycle index shown below:

\[
Z(A; k, k, \ldots, k) = \frac{1}{\text{Card}(A)} \sum_{\alpha \in A} \left( k_{\alpha (1)}^{\alpha (1)} \cdots k_{\alpha (\ell)}^{\alpha (\ell)} \right)
\]
The polynomials we have obtained characterize the number of different cases that arise in the configurations of the nD-OPP’s. Furthermore, different colorings provide us the possibility for exploding these countings in other applications such as Marching Cubes and other similar visualization algorithms.

3.6. Four Equivalence Relations in the set B^2

In this section we will describe our equivalence relations which are based in the definitions presented in previous sections. Also, we will mention the way such relations will allow to improve the determination of the geometrical and topological equivalence between two combinations of hyper-boxes.

3.6.1. The Equivalence Relation R_{adj}

Definition 3.10 [Aguilera04]: Let C(a, b) be the number of bits that do not change from binary string ‘a’ respect to binary string ‘b’.

For example:
\[
C(110, 001) = 0 \quad \text{(no bit remains unchanged)}
\]
\[
C(110, 011) = 1 \quad \text{(bit 1 does not change)}
\]
\[
C(110, 111) = 2 \quad \text{(bits 0 and 1 do not change)}
\]

By comparing the binary representations of the positions of two hyper-boxes in an n-dimensional combination we can infer the type of adjacency between them. If C(a, b) is equal to n-1, it implies that n-1 bits do not change and therefore these unchanged bits will refer to the positive or negative parts of n-1 main axes which define specifically a \( \Pi_{n-1} \) shared cell. If C(a,b)=n-2 then we have an \((n-2)\)-D adjacency (a \( \Pi_{n-2} \) shared cell); and so forth until the cases when C(a,b)=1 (edge adjacency) and C(a,b)=0 (vertex adjacency).

Definition 3.11 [Pérez-Aguila05]: Let Adj_{n}(a, b) be the function that computes the type of adjacency between two n-dimensional hyper-boxes referred through the binary digits that correspond to their respective hyper-octants. Then we will have:

\[
\begin{align*}
\Pi_{n-1} & \quad \text{((n−1)D adjacency) } \quad \text{iff } \quad \Gamma(a, b) = n-1 \\
\Pi_{n-2} & \quad \text{((n−2)D adjacency) } \quad \text{iff } \quad \Gamma(a, b) = n-2 \\
\vdots & \quad \vdots \\
\Pi_{1} & \quad \text{(edge adjacency) } \quad \text{iff } \quad \Gamma(a, b) = 1 \\
\Pi_{0} & \quad \text{(vertex adjacency) } \quad \text{iff } \quad \Gamma(a, b) = 0
\end{align*}
\]

In other words: \( \text{Adj}_{n}(a, b) = \Pi_{\Gamma(a,b)} \)

For example, consider the vector representation of the 4D combination \( c = (0,1,0,1,0,1,0,0,0,0,0,0) \). The adjacencies between its six hyper-boxes are shown in Table 3.5.

<table>
<thead>
<tr>
<th>Shared k-D cell ((0 \leq k &lt; 4))</th>
<th>Shared k-D cell ((0 \leq k &lt; 4))</th>
<th>Shared k-D cell ((0 \leq k &lt; 4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Adj}_4(0001, 0011) = \Pi_4 )</td>
<td>( \text{Adj}_4(0011, 0101) = \Pi_2 )</td>
<td>( \text{Adj}_4(0101, 1000) = \Pi_1 )</td>
</tr>
<tr>
<td>( \text{Adj}_4(0001, 0101) = \Pi_3 )</td>
<td>( \text{Adj}_4(0011, 0110) = \Pi_2 )</td>
<td>( \text{Adj}_4(0101, 0110) = \Pi_2 )</td>
</tr>
<tr>
<td>( \text{Adj}_4(0001, 0110) = \Pi_2 )</td>
<td>( \text{Adj}_4(0011, 0111) = \Pi_3 )</td>
<td>( \text{Adj}_4(0110, 0111) = \Pi_3 )</td>
</tr>
<tr>
<td>( \text{Adj}_4(0001, 0111) = \Pi_2 )</td>
<td>( \text{Adj}_4(0011, 1000) = \Pi_1 )</td>
<td>( \text{Adj}_4(0110, 1000) = \Pi_1 )</td>
</tr>
<tr>
<td>( \text{Adj}_4(0001, 1000) = \Pi_2 )</td>
<td>( \text{Adj}_4(0101, 0111) = \Pi_3 )</td>
<td>( \text{Adj}_4(0111, 0000) = \Pi_0 )</td>
</tr>
</tbody>
</table>

Table 3.5. The adjacencies between the six hyper-boxes of a 4D combination (see text for details).

Definition 3.12 [Pérez-Aguila05]: Let \( x = (x_1, \ldots, x_n) \in B^n \). \( \Gamma(x) \) will denote the number of scalars in \( x \) such that \( x_i = 1 \), \( 1 \leq i \leq 2^n \).

For example, in 4D combination \( c \) we have that \( \Gamma(c) = 6 \).
Definition 3.13 [Pérez-Aguila05]: Let \( x \in \mathbb{B}^n \). We define \( \text{Adj}_{n,\Pi_i}(x) \), \( 0 < i \leq (n-1) \) as the number of adjacencies in \( x \) such that \( \text{Adj}_{n,j,k} = \Pi_i \), where \( j \in \{0 \cdots 0, \ldots, 1\}_n \), \( k > j \) and \( x_j = x_k = 1 \).

In 4D combination \( c \) we have that \( \text{Adj}_{4,\Pi_2}(c) = 1 \), \( \text{Adj}_{4,\Pi_2}(c) = 4 \), \( \text{Adj}_{4,\Pi_2}(c) = 5 \) and \( \text{Adj}_{4,\Pi_2}(c) = 5 \).

Definition 3.14 [Pérez-Aguila05]: Let \( x \in \mathbb{B}^n \). We define
\[
\text{Adj}(x) = (\Gamma(x), \text{Adj}_{n,\Pi_{i-1}}(x), \ldots, \text{Adj}_{n,\Pi_1}(x), \text{Adj}_{n,\Pi_0}(x))
\]

Therefore, 4D combination \( c \) has \( \text{Adj}(c) = (6,5,5,4,1) \).

An algorithm based in Definitions 3.13 and 3.14 would determine, in a combination represented through a vector in the set \( \mathbb{B}^n \), the number of hyper-boxes and adjacencies between them. Because the maximum number of hyper-boxes which are present in a combination depends of the number of hyper-octants of the space in which they are embedded, we have that the execution time of such algorithm is related precisely with this number. Let \( h = 2^n \) be the number of hyper-octants in the \( n \)-dimensional space. If we consider the combination with \( 2^n \) hyper-boxes, that is, a hyper-box in each one of the hyper-octants, then we will have that the execution time for this case is \( O(h^2) \). Such time is an upper-bound for the times related to the different combinations with \( 0 \) to \( 2^n-1 \) hyper-boxes.

Definition 3.15 [Pérez-Aguila05]: Let \( x, y \in \mathbb{B}^n \). We will say that \( \text{Adj}(x) = \text{Adj}(y) \) if and only if
\[
(\Gamma(x) = \Gamma(y)) \land ((\text{Adj}_{n,\Pi_{i-1}}(x) = \text{Adj}_{n,\Pi_{i-1}}(y)) \land \ldots \land (\text{Adj}_{n,\Pi_1}(x) = \text{Adj}_{n,\Pi_1}(y)))
\]

Definition 3.16: Let \( x, y \) be vectors in the set \( \mathbb{B}^n \). Let the binary relation \( R_{\text{adj}} \) defined as
\[
x R_{\text{adj}} y \iff (\text{Adj}(x) = \text{Adj}(y))
\]

Theorem 3.4 [Pérez-Aguila05]: The binary relation \( R_{\text{adj}} \) in the set \( \mathbb{B}^n \) is an equivalence relation.

Proof:
Let \( x, y, z \in \mathbb{B}^n \). The following properties are satisfied:
1) Reflexivity: If \( x R_{\text{adj}} x \Rightarrow \text{Adj}(x) = \text{Adj}(x) \).
2) Symmetry: If \( x R_{\text{adj}} y \Rightarrow \text{Adj}(x) = \text{Adj}(y) \Rightarrow \text{Adj}(y) = \text{Adj}(x) \Rightarrow y R_{\text{adj}} x \).
3) Transitivity: If \( x R_{\text{adj}} y \land y R_{\text{adj}} z \Rightarrow \text{Adj}(x) = \text{Adj}(y) \land \text{Adj}(y) = \text{Adj}(z) \Rightarrow \text{Adj}(x) = \text{Adj}(z) \Rightarrow x R_{\text{adj}} z \).

Definition 3.17: The set \( \{ y \}_{\text{adj}} = \{ x \in \mathbb{B}^n : x R_{\text{adj}} y \} \) will be the equivalence class of vectors \( x \) under relation \( R_{\text{adj}} \) and vector \( y \) is their representative, i.e., \( y \) is a configuration.

Through the combination’s binary representation we have improved the representation of combinations of hyper-boxes in terms of the time and memory complexity, because an \( n \)-dimensional combination can be managed with only \( 2^n \) bits instead of the \( 2^n \) vertices for each one of the \( 2^n \) possible hyper-boxes.

3.6.2. The Equivalence Relation \( R_i \)

Now, we will introduce a third equivalence relation which is based in the isomorphism between weighted graphs. Such relation will allow us to determine that the partition of the set \( \mathbb{B}^n \), induced by relation \( R_{\text{adj}} \), produces an approximation of the partition induced by relation \( R_i \).

We will refer to an undirected weighted graph \( G \) by its sets of vertices \( V \); and edges \( E \), by notation \( G = (V, E) \) [Cormen01]. Each one of its edges \( \{u, v\} \) will have associated a non-negative number \( w(\{u, v\}) \) called the weight of edge \( \{u, v\} \).
Definition 3.18: Two graphs $G=(V,E)$ and $G'=(V',E')$ are isomorphic if there exists a bijection $f:V\rightarrow V'$ such that $\{u,v\}$ is an edge of $G$ if and only if $\{f(u),f(v)\}$ is an edge of $G'$ [Cormen01].

Definition 3.19: Let $G=(V,E)$ and $G'=(V',E')$ be two weighted graphs. $G$ and $G'$ are isomorphic weighted graphs, denoted by $G\cong G'$, if and only if $G$ and $G'$ are isomorphic by bijection $f:V\rightarrow V'$ and $(\forall \{u,v\}\in E)(w(\{u,v\})=w(\{f(u),f(v)\}))$ [Buckley90].

Definition 3.20: Let $G$ be a weighted graph. The weights sequence of vertex $v\in V(G)$, denoted by $w(v)$, is a list in decreasing order of the weights of the incident edges to $v$ [Buckley90].

Definition 3.21 [Pérez-Aguila05]: Let $x=(x_1,\ldots,x_n)\in B^n$. The adjacencies graph of $x$, denoted by $G(x)=(V(x),E(x))$, will be a weighted graph constructed in the following way:

- $V(x) = \{i \mid x_i = 1, x_i \in x; \text{i.e., the position of scalar } x_i \text{ equal to one in vector } x\}$
- $E(x) = \{(i, j) \mid i, j \in V(x), i \neq j\}$
- The weight $w((i, j))$ of each edge $(i, j)\in E(x)$ will be given by $w((i, j)) = \text{Adj}(i, j)$.

For example, $G(c)=(V(c),E(c))$ is described in Table 3.6.

<table>
<thead>
<tr>
<th>Vertices</th>
<th>$V(c) = {1, 3, 5, 6, 7, 8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges</td>
<td>$E(c) = {(1, 3), {1, 5}, {1, 6}, {1, 7}, {1, 8}, {3, 5}, {3, 6}, {3, 7}, {3, 8}, {5, 6}, {5, 7}, {5, 8}, {6, 7}, {6, 8}, {7, 8}}$</td>
</tr>
</tbody>
</table>
| Weights of edges | $w(\{1,3\})=\text{Adj}_{\{0001,0011\}}=3 \quad w(\{3,5\})=\text{Adj}_{\{0011,0101\}}=2 \quad w(\{5,7\})=\text{Adj}_{\{0101,0111\}}=3$
|          | $w(\{1,5\})=\text{Adj}_{\{0001,0101\}}=3 \quad w(\{3,6\})=\text{Adj}_{\{0011,0110\}}=2 \quad w(\{5,8\})=\text{Adj}_{\{0101,1000\}}=1$
|          | $w(\{1,6\})=\text{Adj}_{\{0001,0110\}}=1 \quad w(\{3,7\})=\text{Adj}_{\{0011,0111\}}=3 \quad w(\{6,7\})=\text{Adj}_{\{0110,0111\}}=3$
|          | $w(\{1,7\})=\text{Adj}_{\{0001,0111\}}=2 \quad w(\{3,8\})=\text{Adj}_{\{0011,1000\}}=1 \quad w(\{6,8\})=\text{Adj}_{\{0110,1000\}}=1$
|          | $w(\{1,8\})=\text{Adj}_{\{0001,1000\}}=2 \quad w(\{5,6\})=\text{Adj}_{\{0101,0110\}}=2 \quad w(\{7,8\})=\text{Adj}_{\{0111,1000\}}=0$

Table 3.6. Adjacencies graph of 4D combination $c=(0,1,0,1,0,1,1,1,1,1,0,0,0,0,0,0)$

Theorem 3.5: ($\exists x, y \in B^2)(x R_{\text{adj}} y \land G(x) \not\cong G(y))$

Proof:
We will show that there exist vectors $x, y \in B^2$ such that $\text{Adj}(x) = \text{Adj}(y)$ but their associated adjacencies graphs $G(x)$ and $G(y)$ are not isomorphic weighted graphs.

Let $x=c$ (see previous example). Let $y \in B^2$ defined by $y=(1,0,1,1,0,1,1,0,0,0,0,0,0,0)$. In Table 3.7 are shown the corresponding weights sequences of vectors $x$ and $y$.

<table>
<thead>
<tr>
<th>$V(x) = {1, 3, 5, 6, 7, 8}$</th>
<th>$V(y) = {0, 2, 3, 5, 7, 8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w(1) = (3, 3, 2, 2, 1)$</td>
<td>$w(0) = (3, 3, 2, 2, 1)$</td>
</tr>
<tr>
<td>$w(3) = (3, 3, 2, 2, 1)$</td>
<td>$w(2) = (3, 3, 2, 2, 1)$</td>
</tr>
<tr>
<td>$w(5) = (3, 3, 2, 2, 1)$</td>
<td>$w(3) = (3, 3, 2, 2, 1)$</td>
</tr>
<tr>
<td>$w(6) = (3, 2, 2, 1, 1)$</td>
<td>$w(5) = (3, 2, 2, 1, 1)$</td>
</tr>
<tr>
<td>$w(7) = (3, 3, 3, 2, 0)$</td>
<td>$w(7) = (3, 3, 2, 1, 0)$</td>
</tr>
<tr>
<td>$w(8) = (2, 1, 1, 1, 0)$</td>
<td>$w(8) = (3, 2, 1, 1, 0)$</td>
</tr>
</tbody>
</table>

Table 3.7. Weight sequences of two vectors in $B^2$ (see text for details).

Because $\text{Adj}(x) = \text{Adj}(y) = (6, 5, 5, 4, 1) \Rightarrow x R_{\text{adj}} y$.

However, the weights sequence of vertex 8 in $G(x)$, $w(8) = (2,1,1,1,0)$, does not coincide with any weights sequence of the vertices in $G(y)$ (see Table 3.7).

$\therefore G(x) \not\cong G(y) \Rightarrow (\exists x, y \in B^2)(x R_{\text{adj}} y \land G(x) \not\cong G(y))$

Definition 3.22: Let $x$ and $y$ be vectors in the set $B^2$. Let $R_{\text{adj}}$ be the relation defined by $x R_{\text{adj}} y \Leftrightarrow G(x) \equiv G(y)$.
Theorem 3.6 [Pérez-Aguila05]: The relation $R_i$ defined over the set $B^n$ is an equivalence relation.

Proof:
Let $x, y, z \in B^n$. The following properties are satisfied (‘$\circ$’ denotes the function composition operator):

1) Reflexivity: If $x R_i x$
   \[ \Rightarrow (\exists x: V(x) \rightarrow V(x), \text{t-identity mapping})(G(x) \equiv G(x)) \]
   \[ \Rightarrow (\forall x \in B^n)(x R_i x) \]

2) Symmetry: If $x R_i y \Rightarrow (\exists f: V(x) \rightarrow V(y), f\text{-bijective})(G(x) \equiv G(y))$
   \[ \Rightarrow \text{Because } f \text{ is bijective, } (\exists f^{-1}: V(y) \rightarrow V(x), f^{-1}\text{-bijective})(f \circ f^{-1} = f^{-1} \circ f = 1) \]
   \[ \Rightarrow G(y) \equiv G(x) \text{ by bijection } f^{-1} \Rightarrow y R_i x \]
   \[ \Rightarrow (\forall x, y \in B^n)(x R_i y \Rightarrow y R_i x) \]

3) Transitivity: If $x R_i y \land y R_i z$
   \[ \Rightarrow (\exists f: V(x) \rightarrow V(y), f\text{-bijective})(G(x) \equiv G(y)) \land (\exists g: V(y) \rightarrow V(z), g\text{-bijective})(G(y) \equiv G(z)) \]
   \[ \Rightarrow (\exists h: V(x) \rightarrow V(z), h\text{-bijective})(h = g \circ f) \Rightarrow (\forall x \in B^n) \text{ by bijection } h \]
   \[ \Rightarrow x R_i z \]
   \[ \Rightarrow (\forall x, y, z \in B^n)(x R_i y \land y R_i z \Rightarrow x R_i z) \]
   \[ \Rightarrow (\forall x, y \in B^n)(x R_i y) \]

\[ \therefore \text{ Relation } R_i \text{ in } B^n \text{ is an equivalence relation.} \square \]

Definition 3.23: The set $[y]_i = \{x \in B^n; x R_i y\}$ will be the equivalence class of vectors $x$ under relation $R_i$ and $y$ is their representative, i.e., $y$ is a configuration.

Theorem 3.7: $(\forall[x]_i \subseteq B^n)(\exists[y]_{adj})(|[x]_i \subseteq [y]_{adj})$

Proof:
Let $[x]_i$ be any class in $B^n$ under relation $R_i$.
   \[ \Rightarrow (\forall z \in [x]_i)(G(z) \equiv G(x)) \]
   \[ \Rightarrow (\forall z \in [x]_i)(\text{Adj}(z) = \text{Adj}(x)) \]
   \[ \Rightarrow (\forall [y]_{adj})(\forall z \in [y]_{adj}) \]
   \[ \Rightarrow (\forall z \in [x]_i)(\text{Adj}(z) = \text{Adj}(x) \land \text{Adj}(x) = \text{Adj}(y)) \]
   \[ \Rightarrow (\forall z \in [x]_i)(\text{Adj}(z) = \text{Adj}(y)) \]
   \[ \Rightarrow (\forall z \in [x]_i)(z \in [y]_{adj}) \]
   \[ \therefore [x]_i \subseteq [y]_{adj} \]

According to Theorem 3.5, we determined the existence of combinations $x$ and $y$ such that $\text{Adj}(x) = \text{Adj}(y)$, that is, both belong to the same equivalence class under relation $R_{adj}$, however $x \notin [y]_i$, and $y \notin [x]_i$. By Theorem 3.7 we proved the existence of equivalence classes under relation $R_i$ such that they can be characterized as subsets of equivalences classes under $R_{adj}$. Then, we can conclude that there exist combinations $x, y \in B^n$ such that $([x]_i \subseteq [y]_{adj} \land [y]_i \subseteq [x]_{adj})$ where $[x]_i \cap [y]_i = \emptyset$. Therefore, the equivalence classes induced by $R_{adj}$ can be seen as an approximation of the equivalence classes generated by relation $R_i$.

Definition 3.24: We will say that a partition of the set $B^n$ induced by an equivalence relation $R_1$ is coarser than another partition induced by equivalence relation $R_2$, denoted by $R_1 \geq R_2$, if and only if

$(\forall[x]_1 \subseteq B^n, [x]_1$ equivalence relation under $R_1, [y]_1$ equivalence relation under $R_2)([x]_1 \subseteq [y]_1)$

Corollary 3.1: The partition induced by equivalence relation $R_{adj}$ is coarser than the partition induced by equivalence relation $R_i$, i.e., $R_{adj} \geq R_i$. 

Proof: By Theorem 3.7 we have that $(\forall[x]_i \subseteq B^n)(\exists[y]_{adj} \subseteq B^n([x]_i \subseteq [y]_{adj})$
\[ \therefore \text{ by Definition 3.24, } R_{adj} \geq R_i. \square \]

It can be asked if the partition induced by equivalence relation $R_i$ is the same than the partition induced by equivalence relation $R_{adj}$. This hypothesis can be verified experimentally for 1D, 2D and 3D spaces and the answer in these cases is yes. However, and surprisingly, we have 401 configurations (or equivalence classes) induced by $R_i$ while there are 402 configurations induced by $R_{adj}$. See in Table 3.8 the comparison of the distribution of the 65,536 possible combinations of 4D hyper-boxes according to each relation.
According to Table 3.8, there is a difference between the number of configurations with 4 hyper-boxes: 18 configurations induced by $R_i$ and 19 configurations induced by $R_f$. This observation lead us to hypothesize that there exist combinations of four 4D hyper-boxes with the same adjacencies graphs, however, the bijective function, that determined these graphs were isomorphic, does not represent a geometric transformation. Therefore these combinations, in the same equivalence class under $R_i$, are not geometrically equivalent and hence they are in different equivalence classes under $R_f$. We will show that $R_i \geq R_f$.

**Definition 3.25:** Let $c$ be a combination of hyper-boxes in the $n$-Dimensional space. An **Odd Adjacency Edge** of $c$, or just **Odd Edge**, will be an edge with an odd number of incident hyper-boxes of $c$. Conversely, if an edge has an even number of incident hyper-boxes of $c$, it will be called **Even Adjacency Edge**, or just **Even Edge**.

**Definition 3.26:** Let $x_i, 1 \leq i \leq n$, be an axis in the $n$-Dimensional space. $x_i^+$ will denote to the positive part of $x_i$-axis while $x_i^-$ will denote to the negative part of $x_i$-axis.

**Definition 3.27:** Let $c$ be a combination of hyper-boxes in the $n$-Dimensional space. For $1 \leq i \leq n$ we define to $\vartheta^+(c)$ and $\vartheta^-(c)$ as

$$
\vartheta^+(c) = \begin{cases} 
1 & \text{iff there is an Odd Edge on } x_i^+ \\
0 & \text{otherwise}
\end{cases}
\quad \vartheta^-(c) = \begin{cases} 
1 & \text{iff there is an Odd Edge on } x_i^- \\
0 & \text{otherwise}
\end{cases}
$$

The process for identifying an odd edge in a combination with binary representation is straightforward. A hyper-box will be incident to an edge on $x_i^+$ if its $i$-th bit in the binary representation of its position in the string is equal to 0. On the other hand, a hyper-box will be incident to an edge on $x_i^-$ if its $i$-th bit in the binary representation of its position in the string is equal to 1. For example, the hyper-box in the hyper-octant $x_1^+ x_2^- x_3^+ x_4^- x_5$ (2210 = 101102) is incident to the edges on $x_1^+$, $x_2^+$, $x_3^+$, $x_4^-$ and $x_5^+$. By counting the number of times that each hyper-box is incident to an edge we can infer if it is an odd edge or an even edge. An algorithm that computes the counting of odd edges in a combination of hyper-boxes with binary representation will have a temporal complexity of $O(h)$ (remember that $h$ is the number of hyper-octants in the $n$-dimensional space).

**Definition 3.28:** Let $c$ be a vector in the set $B^n$. $\vartheta(c)$ will denote to the vector whose first position contains to $\Gamma(c)$ and its subsequent $2n$ elements are $\vartheta^+(c)$ and $\vartheta^-(c)$, for $1 \leq i \leq n$, listed in increasing order.
Consider for example the 3D configuration $c=(0,1,0,1,0,1,1,1,1)$. According to Definition 3.27: $\vartheta_i^c(c)=0$, $\vartheta_1^c(c)=1$, $\vartheta_2^c(c)=1$, $\vartheta_3^c(c)=0$, $\vartheta_4^c(c)=1$. By applying Definition 3.28 we have $\vartheta(c)=(4,0,0,1,1,1,1)$. As can be seen, $\vartheta(c)$ is in fact a vector that shows the presence or absence of odd edges in a configuration.

**Definition 3.29:** Let $x$ and $y$ be vectors in the set $B^n$ with their corresponding vectors $\vartheta(x)=(\Gamma(x),x_1,x_2,...x_{2n})$ and $\vartheta(y)=(\Gamma(y),y_1,y_2,...y_{2n})$. We will say that $\vartheta(x)=\vartheta(y)$ if and only if $(\Gamma(x)=\Gamma(y)) \land (x_1=y_1) \land ... \land (x_{2n}=y_{2n})$

**Theorem 3.8:** $(\exists x, y \in B^n)(x R, y \land \vartheta(x) \neq \vartheta(y))$

Proof:

We will show that there exist vectors $x, y \in B^n$ such that $G(x) \equiv G(y)$ but their associated vectors $\vartheta(x) \neq \vartheta(y)$. Let $x_0=(1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0)$ and $y_0=(1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0)$ be vectors in the set $B^i$. The Tables 3.9 and 3.10 show the adjacencies graphs for these combinations.

<table>
<thead>
<tr>
<th>Vertices</th>
<th>$V(x_0) = {0, 3, 5, 6}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges</td>
<td>$E(x_0) = {(0,3), (0,5), (0,6), (3,5), (3,6), (5,6)}$</td>
</tr>
<tr>
<td>Weights of edges</td>
<td>$w(0,3)=\text{Adj}_4(0000,0011)=2$</td>
</tr>
<tr>
<td></td>
<td>$w(0,5)=\text{Adj}_4(0000,0101)=2$</td>
</tr>
<tr>
<td></td>
<td>$w(0,6)=\text{Adj}_4(0000,0110)=2$</td>
</tr>
</tbody>
</table>

Table 3.9. Adjacencies graph of 4D combination $x_0=(1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0)$

<table>
<thead>
<tr>
<th>Vertices</th>
<th>$V(y_0) = {0, 3, 5, 9}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edges</td>
<td>$E(y_0) = {(0,3), (0,5), (0,9), (3,5), (3,9), (5,9)}$</td>
</tr>
<tr>
<td>Weights of edges</td>
<td>$w(0,3)=\text{Adj}_4(0000,0011)=2$</td>
</tr>
<tr>
<td></td>
<td>$w(0,5)=\text{Adj}_4(0000,0101)=2$</td>
</tr>
<tr>
<td></td>
<td>$w(0,9)=\text{Adj}_4(0000,0110)=2$</td>
</tr>
</tbody>
</table>

Table 3.10. Adjacencies graph of 4D combination $y_0=(1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0)$

It can be verified that there exists an isomorphism between the combinations such that $x_0 \sim y_0$, hence $y_0 \in [x_0]$. However, their distributions of odd edges differs completely: $\vartheta(x_0) \neq \vartheta(y_0)$, that is, $\vartheta(x_0) \neq \vartheta(y_0)$.

Through the above Theorem we have seen that combinations $x_0$ and $y_0$ are in the same equivalence class $[x_0]$, however they are not equivalent combinations from a geometrical and topological point of view because their counting of odd edges. Therefore, $y_0 \notin [x_0]$ and $x_0 \notin [y_0]$, that is, to each combination corresponds a different equivalence class under relation $R_e$. Furthermore, the function that allowed the isomorphism between $G(x)$ and $G(y)$ can not be considered a geometric transformation because it modified the topology in the combinations by transforming some odd edges to even edges and vice versa through its application.

**Theorem 3.9:** $(\forall x, y \subseteq B^n)(\exists x, y \subseteq B^n)(x \subseteq y)\land (y \subseteq x)\land (x \neq y)$

Proof:

Let $x, y$ be any class in $B^n$ under relation $R_e$.

$(\forall x, y \subseteq B^n)(x \subseteq y)\land (y \subseteq x)\land (x \neq y)$

Through Theorem 3.8 we have proved the existence of combinations $x$ and $y$ of hyper-boxes such that $G(x) \equiv G(y)$ but $x \neq y$ and $x \notin [y]$. By Theorem 3.9 we have formalized the existence of equivalence classes under relation $R_e$ that can be characterized as subsets of equivalence classes under $R_e$. By this way we have the following...
Corollary 3.2: The partition induced by equivalence relation \( R_i \) is coarser than the partition induced by equivalence relation \( R_{	ext{e}} \), i.e., \( R_i \geq R_{	ext{e}} \).

Proof:
By Theorem 3.9 we have that \((\forall |x| \subseteq B^n)(\exists |y| \subseteq B^n)(|x| \subseteq |y|)\)
\( \therefore \) by Definition 3.24, \( R_i \geq R_{	ext{e}} \).

3.6.3. The Equivalence Relation \( R_{	ext{E}} \)

Now, we will define the equivalence relation \( R_{	ext{E}} \) which is based in the previously presented odd edges counting in a combination of hyper-boxes.

**Definition 3.30:** Let \( x \) and \( y \) be vectors in the set \( B^n \). Let \( R_{	ext{E}} \) be the binary relation defined by
\( x R_{	ext{E}} y \iff \vartheta(x) = \vartheta(y) \).

**Theorem 3.10:** The binary relation \( R_{	ext{E}} \) in the set \( B^n \) is an equivalence relation.

Proof:
Let \( x, y \) and \( z \) be vectors in the set \( B^n \). There are satisfied the following properties:
1) Reflexivity: If \( x R_{	ext{E}} x \Rightarrow \vartheta(x) = \vartheta(x) \iff (\forall x \in B^n)(x R_{	ext{E}} x) \)
2) Symmetry: If \( x R_{	ext{E}} y \Rightarrow \vartheta(x) = \vartheta(y) \Rightarrow \vartheta(y) = \vartheta(x) \Rightarrow y R_{	ext{E}} x \iff (\forall x, y \in B^n)(x R_{	ext{E}} y \Rightarrow y R_{	ext{E}} x) \)
3) Transitivity: If \( x R_{	ext{E}} y \land y R_{	ext{E}} z \Rightarrow \vartheta(x) = \vartheta(y) \land \vartheta(y) = \vartheta(z) \Rightarrow \vartheta(x) = \vartheta(z) \Rightarrow x R_{	ext{E}} z \)
\( \therefore \) Relation \( R_{	ext{E}} \) in \( B^n \) is an equivalence relation.

**Definition 3.31:** The set \( \{ x \in B^n : x R_{	ext{E}} y \} \) will be the equivalence class of vector \( x \) under relation \( R_{	ext{E}} \) and \( y \) is their representative, i.e., \( y \) is a configuration.

We will show that \( R_{	ext{E}} \geq R_{	ext{adj}} \). The following results will be useful.

**Definition 3.32:** \( \mathbb{R}_{	ext{adj}}^+ \) will denote to the subspace defined by the positive part of \( x_i \)-axis and the remaining axes of the \( n \text{D} \) space. \( \mathbb{R}_{	ext{adj}}^- \) will denote to the subspace defined by the negative part of \( x_i \)-axis and the remaining axes of the \( n \text{D} \) space.

**Theorem 3.11:** Let \( c \) be a combination of hyper-boxes in the \( n \text{D} \) space. In \( c \) there are exactly \( n \) linearly independent odd edges, which are incident to the origin of the coordinate system in the combination, if and only if combination \( c \) has an odd number of hyper-boxes.

Proof:
\( \Rightarrow \) Consider \( x_i \)-axis, \( 1 \leq i \leq n \), in which one of the odd edges is embedded. The odd edge is on \( x_i^+ \) or \( x_i^- \) and by Definition 3.25, an odd number \( n_i^+ \) of \( n \text{D} \) hyper-boxes are incident to it. Such hyper-boxes will be embedded in \( \mathbb{R}_{	ext{adj}}^+ \) or \( \mathbb{R}_{	ext{adj}}^- \) according to the referred odd edge. Let \( n_i^0 \) the even number of hyper-boxes that are incident to the even edge which is collinear to the selected odd edge. Hence, \( n_i^+ + n_i^0 = \Gamma(c) \) is an odd number. By applying the same procedure to the remaining \( n-1 \) axes we obtain the same result.
\( \therefore \) \( n_i^+ + n_i^0 = \Gamma(c) \) is an odd number \( \forall i, 1 \leq i \leq n \).

\( \Leftarrow \) Consider \( x_i \)-axis, \( 1 \leq i \leq n \). Because \( \Gamma(c) \) is an odd number, the hyper-boxes in the combination \( c \) are distributed in such way that in \( \mathbb{R}_{	ext{adj}}^+ \) (or \( \mathbb{R}_{	ext{adj}}^- \)) there are an odd number of hyper-boxes while in \( \mathbb{R}_{	ext{adj}}^+ \) (or \( \mathbb{R}_{	ext{adj}}^- \)) there are an even number of hyper-boxes. Therefore, there is an odd edge incident to the origin in \( \mathbb{R}_{	ext{adj}}^+ \) (or \( \mathbb{R}_{	ext{adj}}^- \)). By applying this reasoning to the remaining \( n-1 \) axes we identify in each one an odd edge incident to the origin. By this way we have identified \( n \) linearly independent odd edges incident to the origin in the combination \( c \).
Chapter 3 - Configurations in the n-Dimensional Orthogonal Pseudo-Polytopes

**Corollary 3.3:** Let \( c \) be a combination of hyper-boxes in the nD space. In \( c \) there are \( n \) pairs of collinear odd edges or collinear even edges, incident to the origin, if and only if combination \( c \) has an even number of hyper-boxes.

Proof:
This proposition is the counterreciprocal of **Theorem 3.11** \( (p \equiv q \Leftrightarrow \neg p \Leftrightarrow \neg q) \).

**Theorem 3.12:** For all \( x, y \in B^n \) if \( \Gamma(x) = \Gamma(y) \) is an odd number then \( \vartheta(x) = \vartheta(y) \).

Proof:
By **Theorem 3.11** combinations \( x \) and \( y \) will have \( n \) linearly independent odd edges. Therefore, by **Definition 3.29** \( \vartheta(x) = (\Gamma(x), \underbrace{0, \ldots, 0}_{x}, \underbrace{1, \ldots, 1}_{x}) = \vartheta(y) \).

The implication of **Theorem 3.12** rises in the fact that all the combinations with the same odd number of hyper-boxes will be in the same equivalence class, i.e., \( (\forall x, y \in B^n)(\Gamma(x) = \Gamma(y) \text{ is an odd number}) (y \in [x]_E) \). Experimentally we can verify that the partition induced by relation \( R_E \) is equal to the partition induced by relation \( R_{adj} \) only in the 1D and 2D spaces. In the case related to 3D space we have 16 configurations under \( R_E \) while there are 22 configurations under \( R_{adj} \). Consider for example 3D combinations \( a = (1,1,1,0,0,0,0,0), b = (1,1,0,0,0,0,1,0) \) and \( c = (1,0,0,1,0,0,1,0) \). We have, \( \vartheta(a) = \vartheta(b) = \vartheta(c) = (3,0,0,0,1,1,1) \), hence \( b, c \in [a]_E \). However, \( Adj(a) = (3,2,1,0) \), \( Adj(b) = (3,1,1,1) \) and \( Adj(c) = (3,0,3,0) \), therefore the three combinations are in three different equivalence classes under \( R_{adj} \). We have then that equivalence relation \( R_E \) is coarser than equivalence relation \( R_{adj} \):
\[
R_E \geq R_{adj}
\]
We have the following ‘sorting’ between our equivalence relations [Pérez-Aguila06a]:
\[
R_E \geq R_{adj} \geq R_\Gamma \geq R_f
\]

### 3.6.4. The Equivalence Relation \( R_\Gamma \)

It is well known that a hyper-box, from the combinatorial and topological point of view, can be seen as kD hyper-boxes (including the nD hyper-box itself) with \( k = 0, 1, 2, \ldots, n \). In fact, these kD hyper-boxes constitute the closure of the nD hyper-box [Takala91]. We can say, appealing to combinatorial topology terminology, that a combination of nD hyper-boxes is called a k-chain. A chain’s boundary consists of those (k-1)D hyper-boxes that are incident to an odd number of kD hyper-boxes (including the nD hyper-box itself) with \( k = 0, 1, 2, \ldots, n \). If every (k-1)D hyper-box is shared by an even number of kD hyper-box, the chain has no boundary [Henle94]. The sum of two chains consists of those hyper-boxes appearing in either chain but not in both (Due to the practical purpose of this chapter, the presented background and the following definitions are sufficient for our objectives. In **Chapter 4** we will aboard with more detail the fundaments behind k-chains from Spivak’s point of view).

**Definition 3.33:** Let \( c \) be a combination of n-Dimensional hyper-boxes. \( \beta(c) \) will be the (n-1)-chain that consists of the (n-1)D hyper-boxes resulting from the sum of the \( \Gamma(c) \) (n-1)-chains, each one composed by all the (n-1)D hyper-boxes, of each one of the nD hyper-boxes in \( c \).

**Definition 3.34:** Let \( c \) be a combination of n-Dimensional hyper-boxes. \( H(c) \) will denote to the vector whose first element is \( \Gamma(c) \) and whose remaining \( n \) elements constitute a list, in increasing order, of the number of (n-1)D hyper-boxes in \( \beta(c) \) which are embedded in each one of the \( n \) (n-1)D main hyperplanes defined by the space of the configuration.

**Property 3.1:** Let \( c \) be a combination of nD hyper-boxes. Each (n-1)D hyper-box in \( \beta(c) \) belongs to only one nD hyper-box of \( c \).

If a combination \( c \) is represented through binary representation, then the above Property establishes the way we can determine the (n-1)D hyper-boxes included in \( H(c) \). The binary representation of the position in the string of a hyper-box describes the hyper-octant in which that hyper-box is embedded. By considering the descriptive axes of such hyper-octant as a string of length \( n \), we can obtain \( n \) strings of length \( n-1 \) whose characters are taken from the original string. For example, from the 5D hyper-box in the hyper-octant \( \overline{x,x,x,x,x} \) (2210 = 101102) we obtain five strings of length four: \( \overline{x,x,x,x}, \overline{x,x,x,x}, \overline{x,x,x,x}, \overline{x,x,x,x} \) and \( \overline{x,x,x,x} \). Such strings
contain the descriptive axes of the 4D hyper-boxes which are incident to the origin of the combination. Moreover, such descriptive axes determine the main hyperplane in which each 4D hyper-box is embedded. In this way we can determine the (n-1)D hyper-boxes incident to the origin for each nD hyper-box in a combination. According to Property 3.1, if a given (n-1)D hyper-box is present in two hyper-boxes then it cannot be included in H(e) and hence it is discarded. An algorithm that implements this procedure will have a temporal complexity of O(h^n).

For example, consider 4D combination e=(1,0,0,0,0,1,0,1,0,0,0,0,0,0). The 4D hyper-box in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$ has 3D hyper-boxes incident to the origin in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$, $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4$, $\mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$; 4D hyper-box in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$ has 3D hyper-boxes in $\mathbf{x}_1 \mathbf{x}_3 \mathbf{x}_4$, $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4$, $\mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, and $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$; and finally 4D hyper-box in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, has 3D hyper-boxes in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4$, $\mathbf{x}_1 \mathbf{x}_3 \mathbf{x}_4$, $\mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, and $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$. Hence, we can define the 3-chain (each one of its elements in labeled according to the descriptive axes that define each 3D cell, in Chapter 4 we will show formal procedures for labeling elements in a k-chain):

$$(\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4, \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4, \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4) + \left(\mathbf{x}_1 \mathbf{x}_3 \mathbf{x}_4, \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4, \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4\right) + \left(\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3, \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4, \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4\right) + \left(\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3, \mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4, \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4\right)$$

Because 3D hyper-box $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$ is present in 4D hyper-boxes in $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, and $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, it is not included the evaluated 3-chain and $\beta(e)$, $\beta(e)$ is the set composed by the remaining ten 3D hyper-boxes embedded each one in the hyperplanes defined by the axes $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3$, $\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_4$, $\mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4$, and $\mathbf{x}_1 \mathbf{x}_3 \mathbf{x}_4$. Hence, $H(e) = (3,1,3,3,3)$.

**Definition 3.35:** Let $a, b \in B^n$ with their respective vectors $H(a) = (\Gamma(a), a_1, \ldots, a_n)$ and $H(b) = (\Gamma(b), b_1, \ldots, b_n)$. We will say $H(a) = H(b)$ if and only if

$$(\Gamma(a) = \Gamma(b)) \land (a_1 = b_1) \land \ldots \land (a_n = b_n).$$

**Definition 3.36:** Let $a, b \in B^n$. We define the relation $R_{H}$ in the set $B^n$ as $a R_{H} b \iff H(a) = H(b)$.

**Theorem 3.13:** The relation $R_{H}$ is an equivalence relation.

**Proof:**
Let $x, y$ and $z$ be vectors in the set $B^n$. There are satisfied the following properties:

1) Reflexivity: If $x R_{H} x \Rightarrow H(x) = H(x) : (\forall x \in B^n)(x R_{H} x)$
2) Symmetry: If $x R_{H} y \Rightarrow H(x) = H(y) \Rightarrow y R_{H} x : (\forall x, y \in B^n)(x R_{H} y \Rightarrow y R_{H} x)$
3) Transitivity: If $x R_{H} y \land y R_{H} z \Rightarrow H(x) = H(y) \land H(y) = H(z) \Rightarrow H(x) = H(z) \Rightarrow x R_{H} z$

$\therefore (\forall x, y, z \in B^n)(x R_{H} y \land y R_{H} z \Rightarrow x R_{H} z)$

$\therefore$ Relation $R_{H}$ in $B^n$ is an equivalence relation.

**Definition 3.37:** The set $[y]_{H} = \{x \in B^n : x R_{H} y\}$ will be the equivalence class of vector $x$ under relation $R_{H}$ and $y$ is their representative, i.e., $y$ is a configuration.

As we have proceeded with our previous equivalence relations, to show that $R_{E} \geq R_{H} \geq R_{adj}$ we must identify two combinations of hyper-boxes $x_0$ and $y_0$ such that $\vartheta(x_0) = \vartheta(y_0)$ but $H(x_0) \neq H(y_0)$; and two combinations of hyper-boxes $x_1$ and $y_1$ such that $H(x_1) = H(y_1)$ but $\text{Adj}(x_1) \neq \text{Adj}(y_1)$. In the first case, the partitions induced by relations $R_{E}$ and $R_{H}$ are the same for 1D and 2D spaces (3 and 6 configurations respectively). In the 3D space we have 16 configurations under $R_{E}$ and 19 configurations under $R_{H}$. Consider 3D combinations $x_0=(1,1,1,0,0,0,0,0,0)$ and $y_0=(1,1,0,0,0,0,1,0,0)$. $\vartheta(x_0)=(3,0,0,0,1,1,1)=\vartheta(y_0)$, however, $H(x_0)=(3,1,1,3)$ and $H(y_0)=(3,1,2,3)$, i.e., $H(x_0) \neq H(y_0)$. Hence

$$R_{E} \geq R_{H}$$

The partitions induced by relation $R_{H}$ and $R_{adj}$ are the same for 1D and 2D spaces (3 and 6 configurations respectively). There are 19 configurations under $R_{E}$ and 22 configurations under $R_{H}$ which correspond to the case in the 3D space. Let $x_1=(1,0,0,0,1,0,0)$ and $y_1=(1,0,0,0,0,0,1,0)$. $H(x_1) = (2,2,2,2) = H(x_2)$ but $\text{Adj}(x_1) \neq \text{Adj}(y_1)$ because $\text{Adj}(x_1)=(2,0,1,0)$ and $\text{Adj}(y_1)=(2,0,0,1)$. Therefore

$$R_{E} \geq R_{adj}$$
3.7. Fast Comparison of Combinations of nD Hyper-Boxes through Relations \( R_{\text{adj}}, R_H, R_E \)

Based on results presented in previous sections, we can state the following

**Theorem 3.14** [Pérez-Aguila06a]: \( R_E \geq R_H \geq R_{\text{adj}} \geq R_i \geq R_f \)

At this point, it is important to consider that

If \( f(x) = y \) for \( x, y \in B^n \) and \( f \in \mathcal{f}_n \) then \( (x R_i y) \land (x R_{\text{adj}} y) \land (x R_H y) \land (x R_E y) \)

but the reciprocal is not necessarily true [Pérez-Aguila06a]. For example, let \( x_0 = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \) and \( y_0 = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \) be combinations in the set \( B^5 \). It can be verified that

\[
\begin{align*}
\text{Adj}(x_0) &= (5,1,6,3,0,0) = \text{Adj}(y_0), \\
\vartheta(x_0) &= (5,0,0,0,0,0,1,1,1,1,1) = \vartheta(y_0), \\
H(x_0) &= (5,2,5,5,5,5) = H(y_0) \\
G(x_0) &\cong G(y_0)
\end{align*}
\]

Thus, \( (x_0 R_i y_0) \land (x_0 R_{\text{adj}} y_0) \land (x_0 R_H y_0) \land (x_0 R_E y_0) \) but, through exhaustive searching, there is no function \( f \) in the set \( \mathcal{f}_5 \) such that \( f(x_0) = y_0 \).

On the other hand, equivalence relation \( R_i \) provided tools in order to determine that \( R_{\text{adj}} \geq R_f \). However, it should not be considered from a practical point of view, because it is based on the isomorphism between graphs. It is well known that an algorithm that determines if two graphs are isomorphic or not is in the complexity class NP [Cormen01]. The algorithms based in equivalence relations \( R_{\text{adj}}, R_H, R_E \) under binary representation for combinations of hyper-boxes have their complexity in \( \mathcal{O}(h^2) \) where \( h \) is the number of hyper-octants in the nD space. By ordering the equivalence relations according to their ‘thickness’ we have established the way two combinations \( x, y \) of hyper-boxes could be compared. See **Algorithm 3.1**.

1. If \( (\vartheta(x) = \vartheta(y)) \) then // Verifying if \( x R_E y \)
2. If \( (H(x) = H(y)) \) then // Verifying if \( x R_H y \)
3. If \( (\text{Adj}(x) = \text{Adj}(y)) \) then // Verifying if \( x R_{\text{adj}} y \)
4. Return ‘\( x \) and \( y \) are possibly equivalent combinations’
5. Return ‘\( x \) and \( y \) are not equivalent combinations’

**Algorithm 3.1.** Determining if two combinations of hyper-boxes \( x \) and \( y \) are ‘possibly equivalent combinations’ through relations \( R_E, R_H \) and \( R_{\text{adj}} \).

These equivalent relations working together provide an algorithm whose execution time is \( \mathcal{O}(h^2) \) and they at the same time induce a partition of \( B^n \) which can be seen as an approximation of the partition induced by \( R_i \). In **Table 3.11** are shown the partitions induced by \( R_E, R_H, R_{\text{adj}}, R_E+R_H+R_{\text{adj}} \) (previous algorithm) and \( R_i \) over the sets \( B^1, B^2, B^3 \) and \( B^4 \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>( R_E )</th>
<th>( R_H )</th>
<th>( R_{\text{adj}} )</th>
<th>( R_E+R_H+R_{\text{adj}} )</th>
<th>( R_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>19</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>147</td>
<td>253</td>
<td>340</td>
<td>402</td>
</tr>
</tbody>
</table>

**Table 3.11.** Partitions of set \( B^2 \) induced by equivalence relations \( R_E, R_H, R_{\text{adj}}, R_E+R_H+R_{\text{adj}} \) and \( R_i \).
We can modify the Algorithm 3.1 in order to use relations \( R_e, R_h \) and \( R_{adj} \) for discarding in a very fast way combinations of hyper-boxes that clearly are not topologically and geometrically equivalent; and only combinations \( x, y \) such that \( (R_e(x)=R_e(y)) \land (R_{adj}(x)=R_{adj}(y)) \land (R_h(x)=R_h(y)) \) are exhaustively evaluated in order to find function \( f \in \mathbb{F}^n \) and according to the result determine in a precise way if they are or not equivalent. See Algorithm 3.2.

1. If \( \emptyset(x) = \emptyset(y) \) then // Verifying if \( xR_e y \)
2. If \( H(x) = H(y) \) then // Verifying if \( xR_h y \)
3. If \( \text{Adj}(x) = \text{Adj}(y) \) then // Verifying if \( xR_{adj} y \)
4. If exists function \( f \in \mathbb{F}^n \) such that \( f(x) = y \) then // Verifying if \( xR_e y \)
5. Return ‘\( x \) and \( y \) are equivalent combinations’
6. Return ‘\( x \) and \( y \) are not equivalent combinations’

Algorithm 3.2. Determining if two combinations of hyper-boxes \( x \) and \( y \) are ‘equivalent combinations’ through relations \( R_e, R_h, R_{adj} \) and \( R \).

The best case for this algorithm occurs when combinations \( x \) and \( y \) are not equivalent combinations and that output instance was generated by equivalence relations \( R_e, R_h \) or \( R_{adj} \). In this situation the execution time is, as mentioned before, \( O(h^5) \). The worst case occurs when combinations \( x \) and \( y \) are not equivalent combinations because there is no function \( f \in \mathbb{F}^n \) such that \( f(x) = y \). This implies that all the functions in the set \( f' \) where evaluated. The cardinality of the set \( f' \) is given by \( \text{Card}(f') = 2^n \). Therefore, execution time for the worst case is given by \( O(2^n n!) \).

According to Table 3.1, which was presented in the introduction of this chapter, only 0.6% of the combinations in the set \( B^5 \) can perform the role of configurations (or representatives of equivalence classes). By considering our proposed algorithm, we can hypothesize that, at least 402 combinations (that 0.6%) could have an execution time of \( O(2^n n!) \) which yields to \( 2^n \cdot 4! = 384 \) steps while the remaining combinations could have the best case execution time \( O(h^5) \) which yields to \( (2^n)^2 \cdot 256 \) steps. By considering only equivalence relation \( R_e \), the evaluation of each one of the 65,536 combinations would require an execution time of 384 steps. In the case for \( B^5 \) we have more important differences:

- Worst case execution time: \( O(2^n n!) \) which yields to \( 2^n \cdot 5! = 3840 \) steps.
- Best case execution time: \( O(h^5) \) which yields to \( (2^n)^2 = 1024 \) steps.

While the execution times for comparing combinations in \( B^6 \) are given by:

- Worst case execution time: \( O(2^n n!) \) which yields to \( 2^n \cdot 6! = 4680 \) steps.
- Best case execution time: \( O(h^5) \) which yields to \( (2^n)^2 = 4096 \) steps.

3.8. The ‘Test-Box’ Algorithm

In this section we will describe the “Test-Box” Algorithm which will allows us to determine the configurations for the nD-OPP’s in a more direct way. We will combine this procedure with our equivalence relations in order to speed up the determination of the configurations.

3.8.1. The Iterative ‘Test-Box’ Algorithm

The first version of the ‘Test-Box’ Algorithm determines the configurations for the nD-OPP’s starting from the configuration with zero hyper-boxes. The remaining configurations are determined by using a 'Test-Box' (a rectangle, cube, hypercube, etc.). For each configuration, we will add to it a 'Test-Box' in one of its empty hyper-octants. This adding will produce a new combination which must be analyzed with the set of the configurations (before combinations) yet processed. If the combination is not equivalent, according to one or some equivalence relations, to any combination in the set of configurations then we have found a new configuration. This process is repeated until all the configuration’s empty hyper-octants have been evaluated with a 'Test-Box' [Pérez-Aguila03e]. In Table 3.12 are shown the 3D combinations obtained from a 3D configuration with three boxes and by applying a 'Test-Box' in all its empty octants.

Table 3.12. Obtaining new configurations through a 3D configuration and a “Test-Box” (shown as wireframe model).
If we suppose that the combinations of hyper-boxes are represented through a vector in the set $B^n$, then adding a 'Test-Box' is straightforward because empty hyper-octants are zeros in the vector, then, a 'Test-Box' is added in these positions by changing a zero to one.

Algorithm 3.3 implements the above procedures.

Input: The number $n$ of dimensions for the configurations to obtain, $n > 0$
Output: The set of configurations for the specified space. Each configuration is represented by a vector in set $B^n$.

Procedure getConfigurationsForSpaceUsingIterativeTestBox($n$)

// We assume the combinations are represented through vectors in the set $B^n$.  
// The configuration with zero hyper-boxes is added to set 'configurations'.
configurations.add($\langle 0, \ldots, 0 \rangle$)

/* Starts the loop for generating new combinations from the set 'configurations' using a 'Test-Box' (rectangle, cube, hypercube, etc.) whose position (hyper-octant to occupy) is indicated by variable 'testBoxPosition'. */
hyperOctants = $2^n$

For each configuration $c$ in the set configurations

testBoxPosition = 0

/* Starts the loop for generating new combinations from configuration 'c' using a 'Test-Box'. */
while (testBoxPosition < hyperOctants)

if ($c[testBoxPosition] = 0$)

/* It is obtained the combination 'newC' from configuration 'c' and the 'Test-Box' added in the hyper-octant specified by 'testBoxPosition'. */
newC = getNewCombination($c$, testBoxPosition)

/* It is verified if combination 'newC' is equivalent, according to one or some equivalence relations, to a configuration in the set 'configurations'. */
if (configurations.existsEquivalentConfiguration(newC) = false)

/* A new configuration has been found and it is added to the set 'configurations'. */
configurations.add(newC)

end-of-if

end-of-while

testBoxPosition++

end-of-for

// The possible configurations have been found. Set 'configurations' is returned.
return configurations

end-of-Procedure

Algorithm 3.3. The Iterative "Test-Box" Algorithm for Determining the Configurations for the nD-OPP's.

3.8.2. The Recursive ‘Test-Box’ Algorithm

The extrusion of an $(n-1)$D configuration to an nD space implies that each one of its hyper-boxes will be translated in a direction that is perpendicular to the $(n-1)$D space in which is embedded [Pérez-Aguila03e]. The translation of each box will describe then a hyper-box. It is important to consider that an nD combination obtained through the extrusion of an $(n-1)$D configuration is not unique, because there are 2 possible translation directions for each hyper-box. For example, in Table 3.13 it is presented the extrusion of a 2D configuration for obtaining three 3D combinations.

Table 3.13. Extrusion of a 2D configuration and the obtained 3D combinations (the arrows indicate the extrusion direction).
Through extruding configurations it is possible to obtain some configurations from nD space by using the configurations from (n-1)D space and so on. By this way, we obtain then a recursive process whose basic case is the configurations for 1D-OPP’s (see Table 3.14).

Table 3.14. The possible configurations (a-c) in 1D space.

The Recursive “Test-Box” algorithm works with the following principle [Pérez-Aguila03e]: to have access to (n-1)D configurations for obtaining the nD configurations. Each (n-1)D configuration is extruded just one time and in just one direction, this means that, the boxes that compose the (n-1)D configuration are extruded towards the same perpendicular direction from space in which are embedded. Once this process is applied, the (n-1)D configuration is not required again. For example, some configurations for 2D-OPP’s are extruded just one time and towards the same direction for obtaining configurations for 3D-OPP’s (Table 3.15).

Table 3.15. Extruding a 2D configuration in the same direction and obtaining their 3D analogous.

For extruding an (n-1)D configuration that is represented as a vector in the set $B^{n-1}$, we must add a zero at the end of each one of the corresponding binary representations of the positions whose value is equal to one. The values of the original positions are in $[0, ..., 0, 1, ..., 1]_{n-1}$, but by the adding we described they are now in $[0, ..., 0, 1, ..., 1]_n$. These new positions are now set to one in the new vector in $B^n$. This completes the extrusion. Our procedure is assuming that the extrusion direction is towards $X_n$-axis and the hyper-boxes are finally embedded in $\mathbb{R}^+_{n}$ (remember that $\mathbb{R}^+_{n}$ denotes to the subspace defined by the positive part of $X_n$-axis and the remaining axes of the nD space). For example, consider 2D combination $x = (1,0,1,1)$. Hence the binary representations, of the positions whose value is equal to one, are $00_2=0, 10_2=2$ and $11_2=3$. By concatenating a zero at the end we have $000_2=0, 100_2=4$ and $110_2=6$. These new positions are set to one in a new vector in $B^3$, hence we have the extruded combination $x' = (1,0,0,1,0,1,0)$.

Once the configurations from (n-1)D space have been extruded, we have now the same number of nD configurations. The next step is the use of each nD configuration for obtaining the remaining configurations. We will use a ‘Test-Box’. For each configuration, we will add to it a ‘Test-Box’ in one of its empty hyper-octants. This adding will produce a new combination which must be analyzed with the set of the configurations (before combinations) yet processed. If the combination is not equivalent, according to one or some equivalence relations, to any combination in the set of configurations then we have found a new configuration. This process is repeated until all the configuration’s empty hyper-octants have been evaluated with a ‘Test-Box’.

We have now the elements to propose a recursive algorithm applying extrusions and a “Test-Box”. The algorithm is resumed with the following main steps [Pérez-Aguila03e]:

1. For a number $n$ of dimensions we obtain the (n-1)D configurations. If $n=1$ then we have the basic case which return 1D configurations from table 2.14.
2. The (n-1)D configurations are extruded in nD configurations.
3. It is added a “Test-Box” to each nD configuration in their empty hyper-octants, this operation will produce new combinations.
4. Each new produced combination will be evaluated with the set of identified configurations. If it is a new configuration then it will be added to the set of identified configurations and considered to be evaluated with a “Test-Box”, because it could produce new configurations.
Algorithm 3.4 implements the above procedures.

**Input:** The number $n$ of dimensions for the configurations to obtain, $n > 0$

**Output:** The set of configurations for the specified space. Each configuration is represented by a vector in set $B^n$.

**Procedure** `getConfigurationsForSpaceUsingRecursiveTestBox(n)` 

// We assume the combinations are represented through vectors in the set $B^n$.

if ($n = 1$) then

// Basic Case: Return the 3 configurations for 1D space.

return get1DSpaceConfigurations()

else /* Recursive call: the (n-1)D configurations are obtained and added to the set 'previousConfigurations'. */

previousConfigurations = getConfigurationsForSpaceUsingRecursiveTestBox($n - 1$)

For each configuration $c$ in the set previousConfigurations

// (n-1)D configuration 'c' is extruded and obtain nD configuration 'newC'.

newC = extrudeConfiguration(c)

// The configuration 'newC' is added to the set 'configurations'.

configurations.add(newC)

end-of-for /* Starts the loop for generating new combinations from the set 'configurations' using a 'Test-Box' (rectangle, cube, hypercube, etc.) whose position (hyper-octant to occupy) is indicated by variable 'testBoxPosition'. */

hyperOctants = 2

For each configuration $c$ in the set configurations

\[ \text{testBoxPosition} = 0 \]

/* Modify to testBoxPosition = 1 if the 'Test-Box' will not be added in empty hyper-octants in $\mathbb{R}^n_+$ (See text for details) */

/* Starts the loop for generating new combinations from configuration 'c' using a 'Test-Box'. */

while (testBoxPosition < hyperOctants)

if ($c[\text{testBoxPosition}] = 0$) then

/* It is obtained the combination 'newC' from configuration 'c' and the 'Test-Box' added in the hyper-octant specified by 'testBoxPosition'. */

newC = getNewCombination(c, testBoxPosition)

/* It is verified if combination 'newC' is equivalent, according to one or some equivalence relations, to a configuration in the set 'configurations'. */

if (configurations.containsEquivalentConfiguration(newC) = false) then

/* A new configuration has been found and it is added to the set 'configurations'. */

configurations.add(newC)

end-of-if

end-of-if

end-of-while

end-of-for

/* The possible configurations have been found. Set 'configurations' is returned. */

return configurations

end-of-Procedure

**Algorithm 3.4.** The Recursive “Test-Box” Algorithm for Determining the Configurations for the nD-OPP’s.

Consider the following reasoning: Because the extrusion of an (n-1)D configuration to an nD configuration will locate all their hyper-boxes in $\mathbb{R}^n_+$ then, we can expect that, a ‘Test-Box’ could not be positioned in the empty hyper-octants in $\mathbb{R}^n_+$ because it would be produced an nD combination that corresponds to an extrusion of an (n-1)D configuration. Consider for example the following case: The 3D configuration shown in Figure 3.1.b is the extrusion
of 2D configuration shown in Figure 3.1.a. By adding a ‘Test-Box’ according to Figures 3.1.c to 3.1.f we get 3D combinations that do not correspond to extrusions of 2D configurations. But, by adding a ‘Test-Box’ according to Figure 3.1.g we get a 3D combination that corresponds to the extrusion of the 2D configuration shown in Figure 3.1.h.

The last reasoning suggest us that is not necessary to consider the adding of a ‘Test-Box’ in the empty hyper-octants in $\mathbb{R}^n_+$. Modifying the Recursive ‘Test-Box’ Algorithm in order to consider this reasoning is straightforward: the counter $testBoxPosition$ must be initialized to one and incremented by two instead of the original initialization to zero and increments by one (see Algorithm 3.4). By this way, and assuming that the combinations are represented through vectors in $B^n$, the counter will point only to those positions whose binary representation have their last bit equal to one, that is, the hyper-octants embedded in $\mathbb{R}^n_-$.

### 3.8.3. Experimental Results

For determining the number of considered $nD$ combinations to obtain the $nD$ configurations identified through the ‘Test-Box’ algorithms it is necessary to analyze the output’s size, i.e., the number of identified configurations. Due to we will know the number of configurations until the algorithms finished, we have then an output-sensitive complexity analysis [deBerg97]. Remember that $2^n$ is the number of hyper-octants in the $nD$ space. We must consider that configurations with 1 hyper-box have $2^n-1$ empty hyper-octants, configurations with 2 hyper-boxes have $2^n-2$ empty hyper-octants and so on. Let $CTB_{n,R,i}$ ($nD$ Configurations-by-Test-Box) be the number of those $nD$ configurations with $i$ hyper-boxes under equivalence relation $R$. Then we have that the number of combinations to analyze by the Iterative ‘Test-Box’ algorithm is

$$C(n,R)=O\left(\sum_{i=0}^{n} CTB_{n,R,i} \cdot (2^n-i)\right)$$  \hfill (Equation 3.5)

Table 3.16 shows the number of identified configurations and the number of analyzed combinations, between the exhaustive algorithm (evaluating all the possible $2^{2n}$ combinations) and the Iterative ‘Test-Box’ Algorithm, under equivalence relation $R_{\text{E}} + R_{\text{H}} + R_{\text{adj}} + R_{\text{f}}$ for the cases where $n \in \{1, 2, 3, 4\}$. We include in our analysis the counting provided by Equation 3.5 in order to show that in the referred cases our formula provides a tight bound for the counting of analyzed combinations.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Configurations: Exhaustive Algorithm</th>
<th>Analyzed Combinations</th>
<th>Configurations: Iterative ‘Test-Box’</th>
<th>Analyzed Combinations</th>
<th>$\sum_{i=0}^{n} CTB_{n,(R_{\text{E}}+R_{\text{H}}+R_{\text{adj}}+R_{\text{f}}),i} \cdot (2^n-i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>16</td>
<td>6</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>256</td>
<td>22</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>4</td>
<td>402</td>
<td>65,336</td>
<td>402</td>
<td>3,216</td>
<td>3,216</td>
</tr>
</tbody>
</table>

| Table 3.16. Exhaustive Algorithm Countings Vs. Iterative ‘Test-Box’ Countings under Equivalence Relation $R_{\text{E}} + R_{\text{H}} + R_{\text{adj}} + R_{\text{f}}$. |
In the case related to the Recursive ‘Test-Box’ Algorithm, when a ‘Test-Box’ is added to each available empty hyper-octant in the combinations, we a bound by summing the number of kD combinations analyzed in each one of the recursive calls with \( k = 1, \ldots, n \). The counting for each \( k \) can be achieved by using the previously described sum in Equation 3.5:

\[
\sum_{i=1}^{k} CTB_{k,i} \cdot (2^i - i)
\]

for each \( k \) in \( \{1, \ldots, n\} \). Hence, the number of combinations to analyze by the Recursive ‘Test-Box’ algorithm, under equivalence relation \( R \), is given by

\[
C(n,R) = O\left(\sum_{k=1}^{n} \left(\sum_{i=0}^{k} CTB_{k,i} \cdot (2^i - i)\right)\right) \quad \text{(Equation 3.6)}
\]

Table 3.17 shows the number of identified configurations and the number of analyzed combinations, between the exhaustive algorithm and the Recursive ‘Test-Box’ Algorithm, in the case where a ‘Test-Box’ is added to each one of the empty hyper-octants of a combination. The analysis is given under equivalence relation \( R_E + R_H + R_{adj} + R_f \) for the cases where \( n \in \{1, 2, 3, 4\} \). We include in our analysis the counting provided by Equation 3.6 in order to show that in the referred cases our formula provides a tight bound for the counting of analyzed combinations.

<table>
<thead>
<tr>
<th>( n )</th>
<th>Configurations: Exhaustive Algorithm</th>
<th>Analyzed Combinations</th>
<th>Configurations: Recursive ‘Test-Box’ Case 1</th>
<th>Analyzed Combinations</th>
<th>( \sum_{k=1}^{n} \left(\sum_{i=0}^{k} CTB_{k,i} \cdot (2^i - i)\right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>16</td>
<td>6</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>256</td>
<td>22</td>
<td>103</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>402</td>
<td>65,336</td>
<td>402</td>
<td>3,319</td>
<td>3,319</td>
</tr>
</tbody>
</table>

Table 3.17. Exhaustive Algorithm Countings Vs. Recursive ‘Test-Box’ Countings (Case 1) under Equivalence Relation \( R_E + R_H + R_{adj} + R_f \)

For determining the number of combinations analyzed by the Recursive ‘Test-Box’ Algorithm, when a ‘Test-Box’ is added only to the empty hyper-octants embedded in \( \mathbb{R}^+_n \), we assume that the combinations have \( 2^{k-1} \) empty hyper-octants in \( \mathbb{R}^+_n \), for each \( k \) in \( \{1, \ldots, n\} \). Under this assumption, we will consider that a configuration with \( i \) hyper-boxes can generate at most \( 2^{k-1} \) combinations with \( i+1 \) hyper-boxes. Hence, an upper bound for the number of combinations to analyze, in this case of the Recursive ‘Test-Box’ Algorithm, is provided through

\[
C(n,R) = O\left(\sum_{k=1}^{n} 2^{k-1} \left(\sum_{i=0}^{k} CTB_{k,i} \right)\right) \quad \text{(Equation 3.7)}
\]

Finally, Table 3.18 shows the number of identified configurations and the number of analyzed combinations, between the exhaustive algorithm and the Recursive ‘Test-Box’ Algorithm, in the case where a ‘Test-Box’ is added to each one of the empty hyper-octants in \( \mathbb{R}^+_n \). The analysis is given under equivalence relation \( R_E + R_H + R_{adj} + R_f \) for the cases where \( n \in \{1, 2, 3, 4\} \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>Configurations: Exhaustive Algorithm</th>
<th>Analyzed Combinations</th>
<th>Configurations: Recursive ‘Test-Box’ Case 2</th>
<th>Analyzed Combinations</th>
<th>( \sum_{k=1}^{n} 2^{k-1} \left(\sum_{i=0}^{k} CTB_{k,i} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>16</td>
<td>6</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>256</td>
<td>22</td>
<td>70</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>402</td>
<td>65,336</td>
<td>402</td>
<td>2,218</td>
<td>3,319</td>
</tr>
</tbody>
</table>

Table 3.18. Exhaustive Algorithm Countings Vs. Recursive ‘Test-Box’ Countings (Case 2) under Equivalence Relation \( R_E + R_H + R_{adj} + R_f \)
Appendix B contains a complete analysis of the ‘Test-Box’ algorithms under each one of our equivalence relations for the cases where $n \in \{1, 2, 3, 4\}$.

3.9. Conclusions

According to the results presented in Tables 3.16 to 3.18, the ‘Test-Box’ Algorithms perform the identification of the configurations for the nD-OPP by inspecting a number of combinations which is significantly minor than the total number of possible combinations (see Table 3.19), while the complexity imposed for determining if two combinations are topologically and geometrically equivalent is reduced by applying equivalence relations $R_E$, $R_H$ and $R_{adj}$.

<table>
<thead>
<tr>
<th>nD Space</th>
<th>Configurations</th>
<th>Analyzed Combinations: Exhaustive Algorithm</th>
<th>Percentage</th>
<th>Analyzed Combinations: Iterative ‘Test-Box’</th>
<th>Percentage</th>
<th>Analyzed Combinations: Recursive ‘Test-Box’ Case 1</th>
<th>Percentage</th>
<th>Analyzed Combinations: Recursive ‘Test-Box’ Case 2</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1D</td>
<td>3</td>
<td>4</td>
<td>75 %</td>
<td>3</td>
<td>100 %</td>
<td>3</td>
<td>100 %</td>
<td>3</td>
<td>100 %</td>
</tr>
<tr>
<td>2D</td>
<td>6</td>
<td>16</td>
<td>37.5 %</td>
<td>12</td>
<td>50 %</td>
<td>15</td>
<td>40 %</td>
<td>11</td>
<td>54.5 %</td>
</tr>
<tr>
<td>3D</td>
<td>22</td>
<td>256</td>
<td>8 %</td>
<td>88</td>
<td>25 %</td>
<td>103</td>
<td>21.3 %</td>
<td>70</td>
<td>31.4 %</td>
</tr>
<tr>
<td>4D</td>
<td>402</td>
<td>65,536</td>
<td>0.6 %</td>
<td>3,216</td>
<td>1.25 %</td>
<td>3,319</td>
<td>12.1 %</td>
<td>2,218</td>
<td>18.1 %</td>
</tr>
</tbody>
</table>

Table 3.19. Percentages between the number of combinations and configurations for the nD-OPP’s for the algorithms: Exhaustive, Iterative ‘Test-Box’, Recursive ‘Test-Box’ (Case 1) and Recursive ‘Test-Box’ (Case 2)

The Table 3.19 shows the analysis previously presented in Section 3.3, that is, we associated the number of configurations with the total number of analyzed combinations for each algorithm in 1D, 2D, 3D and 4D spaces. According to our results, the best option case of the ‘Test-Box’ Algorithms corresponds to the recursive definition when a ‘Test-Box’ is added in the empty hyper-octants embedded in $\mathbb{R}^n$. The second best option corresponds to the iterative definition and in third place, according to the number of analyzed combinations, we have the recursive definition when a ‘Test-Box” is added in all the empty hyper-octants of a combination.

It is essential to determine the configurations for the $n$D-OPP’s because they represent a finite subset which can be used to determine geometric and topologic properties for these $n$D-OPP’s. For example, in [Aguilera02] there are used only the configurations to determine some properties for 4D-OPP’s (See also Section 2.3 in this work). In other contexts, finding configurations allows precise and formal design, developing and debugging of new Substitope Algorithms (such as Marching Cubes) in higher dimensions [Banks06].

This Chapter has described some relations that support us in the task for obtaining in a more direct way the configurations in the n-Dimensional Orthogonal Pseudo-Polytopes. In order to speed up the determination of the topological equivalence between a pair of configurations, we have described relations whose implementation compares any two configurations in a time which only depends of the number of hyper-octants in the space in which their hyper-boxes are embedded. We have showed that our relations are in fact equivalence relations which are “wider” than the equivalence relation based in geometrical transformations and therefore they themselves provides an approximate solution to our problem, but when they are combined precisely with relation $R_f$ we speed up the comparison between combinations of hyper-boxes. Moreover, through the configuration’s binary representation, both the comparison of combinations and the ‘Test-Box’ Algorithms have been improved in terms of the time and memory complexity, because an n-dimensional configuration can be managed with only $2^n$ bits instead of the $2^n$ vertices for each one of the $2^n$ possible hyper-boxes.