Chapter 6
Polyhedron and Polychoron Codes for Describing Atomic Arrangements

Kengo Nishio and Takehide Miyazaki

Abstract The arrangement of atoms can be represented as a tiling of Voronoi polyhedra by using the Voronoi tessellation. We can know how an atom is surrounded by its first nearest neighbour atoms by knowing the shape of the Voronoi polyhedron associated with that atom. Furthermore, by knowing how a Voronoi polyhedron is surrounded by other Voronoi polyhedra, we can know how an atom is surrounded by its first nearest neighbours, second nearest neighbours, third nearest neighbours, …. However, there existed no methods for describing the arrangements of polyhedra, or atomic arrangements. To overcome this problem, we have recently created the polyhedron and polychoron codes [Sci. Rep. 6, 23455, Sci. Rep. 7, 40269, and Bull. Soc. Sci. Form 32, 1 (2017)]. In this chapter, we review the methods.

Keywords Voronoi polyhedron · Amorphous · Glass · Atomic structure analysis · Molecular dynamics simulation

6.1 Introduction

Since the properties of materials depend on how atoms are arranged [1, 2], understanding the arrangement of atoms is essential for studying the material properties. When we perform molecular dynamics or Monte Carlo simulations, we obtain the xyz coordinates of all the atoms. However, knowing all the atomic coordinates does not mean understanding the atomic arrangements. To understand the atomic arrangements, the essence should be extracted from the raw data of the atomic coordinates.

When studying the atomic arrangements of materials, particularly amorphous materials, the Voronoi tessellation is often used [3–10]. By using this method, the

K. Nishio (✉) · T. Miyazaki
National Institute of Advanced Industrial Science and Technology (AIST), Central 2, Umezono 1-1-1, Tsukuba, Ibaraki 305-8568, Japan
e-mail: k-nishio@aist.go.jp

I. Tanaka (ed.), *Nanoinformatics*, https://doi.org/10.1007/978-981-10-7617-6_6

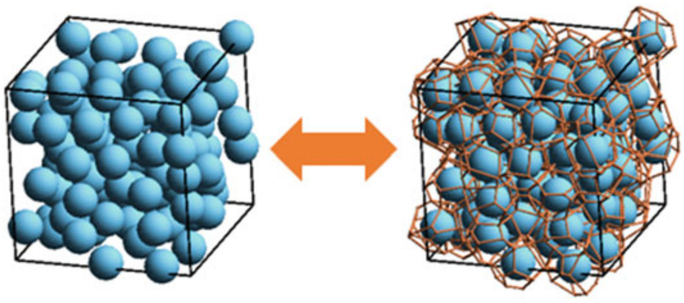


Fig. 6.1 Voronoi tessellation [11]. There is a one-to-one correspondence between the arrangement of atoms (left) and the tiling of Voronoi polyhedra (right)

arrangement of atoms can be represented as a tiling of Voronoi polyhedra (Fig. 6.1). Each Voronoi polyhedron contains one atom. We can know how an atom i is surrounded by its first nearest neighbour atoms by knowing the shape of the Voronoi polyhedron containing the atom i. For example, when the Voronoi polyhedron associated with the atom i is a dodecahedron, the atoms surrounding the atom i occupy the vertices of an icosahedron (Fig. 6.2). Therefore, we can reveal the dominant local atomic arrangements (short-range order) by identifying frequently found Voronoi polyhedra. Furthermore, by knowing how a Voronoi polyhedron is surrounded by other Voronoi polyhedra, we can know how the atom is surrounded by its first nearest neighbours, second nearest neighbours, third nearest neighbours, …. Therefore, we can reveal the long-range order by identifying frequently found assemblages of Voronoi polyhedra.

To classify Voronoi polyhedra, several methods have been proposed. For example, the Voronoi index $\langle n_3 n_4 n_5 n_6 \ldots \rangle$ [3] has often been used in studying amorphous materials. Here, n_i is the number of i-gons of a Voronoi polyhedron. However, different Voronoi polyhedra can accidentally have the same Voronoi index (Fig. 6.3). It is therefore impossible to study details of local atomic arrangements with the Voronoi index. To overcome this problem, Lazar et al. [13] used the Weinberg code [14, 15]. However, there arises a different problem. With this method, a dodecahedron, for example, is encoded as ‘1 2 3 4 5 1 5 6 7 8 1 8 9

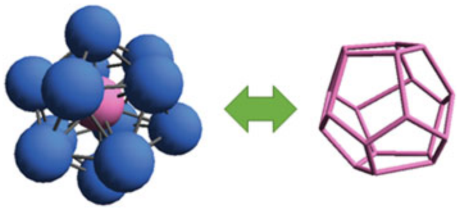


Fig. 6.2 Relation between the atomic arrangement and the shape of the Voronoi polyhedron [12]. First nearest neighbours of the pink atom are blue atoms, occupying the vertices of an icosahedron (left). The Voronoi polyhedron associated with the pink atom is a dodecahedron (right)

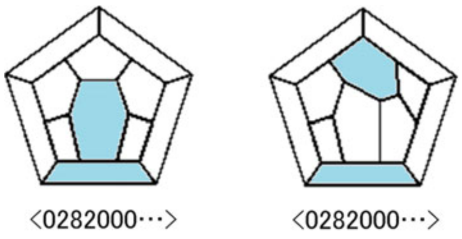


Fig. 6.3 Problem of Voronoi index [12]. Left and right polyhedra are composed of two squares, eight pentagons, and two hexagons. Therefore, both have the same Voronoi index. However, the left polyhedron is different from the right polyhedron. In fact, the hexagons of the left polyhedron adjoin each other, while the hexagons of the right polyhedron are separate from each other

10 2 10 11 12 3 12 13 14 4 14 15 6 15 16 17 7 17 18 9 18 19 11 19 20 13 20 16 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1'. Such a long codeword is difficult for human to handle. Since our brain capacity is limited, a shorter codeword is desirable. More seriously, there existed no methods for classifying assemblages of Voronoi polyhedra. Although we might not become conscious, it has severely prevented our understanding of the long-range order of amorphous materials.

Considering that the knowledge of square pyramids was used to construct the ancient pyramids of Egypt at Gaza, the study of polyhedra has a more than 4500 years history [16, 17]. However, as described above, there existed no methods for briefly representing polyhedra and assemblages of polyhedra in a unified way. To overcome this problem, we have created the polyhedron code (p_3-code) and the polychoron code (p_4-code) [11, 12, 18, 19]. The p_3-code is a method for briefly representing polyhedra. It consists of (1) an encoding algorithm for converting a way of how polygons are arranged to form a polyhedron into a sequence of numbers, which we call a polyhedron codeword (p_3-codeword, or p_3 for short) and (2) a decoding algorithm for recovering the original polyhedron from its p_3. The p_4-code is a generalization of the p_3-code for representing assemblages of polyhedra. By using the p_4-code, a way of how polyhedra are arranged to form a polyhedral assemblage can be converted into a sequence of p_3s, which we call a polychoron codeword (p_4-codeword, or p_4 for short), from which the original polyhedral assemblage can be recovered. In this chapter, we review the p_3-code and p_4-code [11, 12, 18, 19].

6.2 Polyhedron Code

6.2.1 Our Way of Viewing a Polyhedron

We regard a polyhedron as a tiling by polygons of the surface of a three-dimensional object that is topologically the same as a sphere. We are interested in the relative arrangements of polygons (which polygons are glued to which polygons), while we ignore measures such as lengths and angles.

According to the idea developed by L. Euler, A. M. Legendre, F. Möbius, and P. R. Cromwell [16], we assume that the polygons are glued such that (1) any pair of polygons meet only at their sides or corners and that (2) each side of each polygon meets exactly one other polygon along an edge. Here, we stress that parts of a polyhedron and those of the building-block polygons are clearly distinguished (Fig. 6.4). Specifically, vertices and edges are zero- and one-dimensional parts of a polyhedron, respectively. On the other hands, corners and sides are zero- and one-dimensional parts of a polygon, respectively. Since this idea plays a central role in our theory, we need a verb to briefly describe the relation between parts of a polyhedron and those of polygons. For this purpose, we use the verb '*contribute*'. For example, when we say that corners contribute to a vertex or a vertex is contributed by corners, we mean that the vertex is a point of a polyhedron at which the corners of polygons meet. We also say that a polygon (side) contributes to a vertex if one of its corners (endpoints) contributes to that vertex. When we say that an edge is contributed by sides, we mean that the edge is a line segment of a polyhedron along which the sides of polygons meet. The face of a polyhedron is a polygon. But when we call a polygon, we regard it as a building block of a polyhedron. So, we may say the edge of a face. But we cannot say the edge of a polygon.

We first describe a method for simple polyhedra. By a simple polyhedron, we mean that every vertex is degree three. Here, the degree of a vertex is the number of edges incident to that vertex. We use the property that every vertex of a simple polyhedron is contributed by three corners in the method for simple polyhedra. After describing the method for simple polyhedra, we generalize it to non-simple polyhedra.

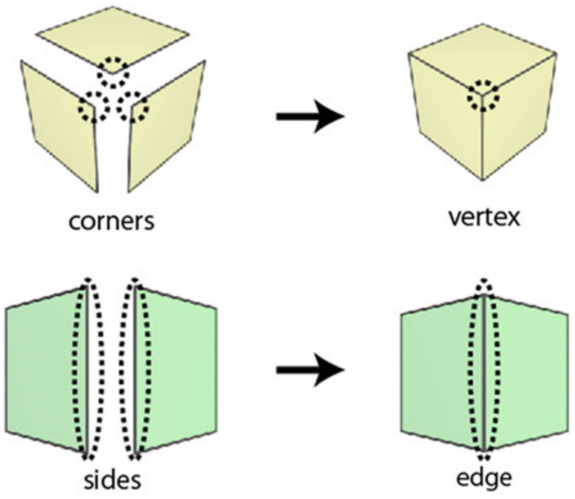


Fig. 6.4 Parts of polygons and parts of a polyhedron [18]

6.2.2 *Decoding Simple Polyhedra*

The p_3-code consists of encoding and decoding algorithms. Since the decoding algorithm is incorporated in the encoding algorithm, we first describe the decoding algorithm. To formulate the decoding algorithm, simple but a lot of new ideas must be introduced. However, the completed algorithm can be described easily. We describe the completed algorithm below. See Ref. [18] for its formulation.

6.2.2.1 How to Recover a 34443-Polyhedron

In our theory, we refer to a polyhedron illustrated in Fig. 6.5 as a 34443-polyhedron. The number sequence 34443 is not only the name of the polyhedron, but also instructs how to construct the polyhedron from its building-block polygons. Each number in 34443 indicates a building-block polygon. Specifically, since the left most number is 3, the polygon 1 is a triangle. Similarly, the polygons 2, 3, and 4 are squares, and the polygon 5 is a triangle.

When recovering the 34443-polyhedron from 34443, we first convert each number to the building-block polygon (Fig. 6.6a). To instruct how to assemble the

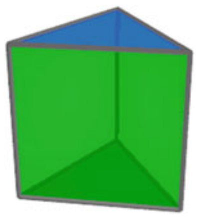


Fig. 6.5 34443-polyhedron [11]

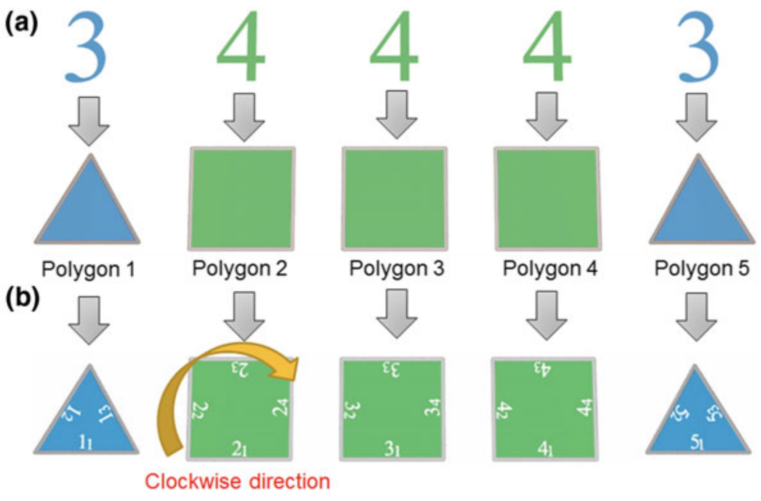


Fig. 6.6 Decoding procedures [11]

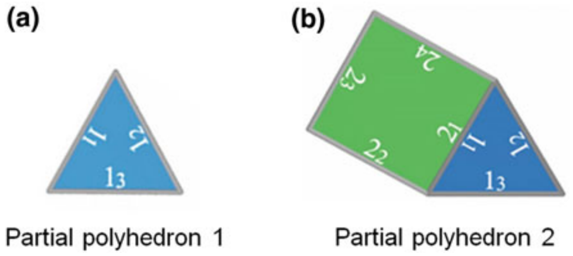


Fig. 6.7 Partial polyhedra 1 and 2 [11]

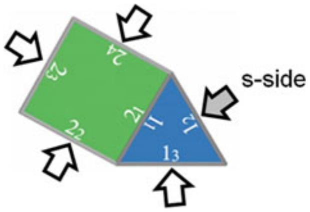


Fig. 6.8 Dangling sides [11]

building-block polygons, we assign identification numbers (IDs) $i_1, i_2, i_3, \ldots$ to the sides of the polygon i in a clockwise direction (Fig. 6.6b). Here, the side i_j means the jth side of the polygon i or the side j of the polygon i. We assume that each symbol i_j has a lexicographical number. In an example shown in Fig. 6.6, the lexicographical number increases in the order of $1_1, \ldots, 1_3, 2_1, \ldots, 2_4, 3_1, \ldots, 3_4, 4_1, \ldots, 4_4, 5_1, \ldots, 5_3$. In general, we define that $i_j < m_n$ when $i < m$, and $i_j < i_k$ when $j < k$.

We call the polygon 1 the partial polyhedron 1 (Fig. 6.7a). By glueing the side 2_1 (the side 1 of the polygon 2) to the side 1_1 of the partial polyhedron 1, we obtain a structure illustrated in Fig. 6.7b, which we call the partial polyhedron 2. Here, we introduce a term '*dangling side*'. The dangling side is a side that is not glued to another side. In the example of Fig. 6.8, the sides $1_2, 1_3, 2_2, 2_3,$ and 2_4 are the dangling sides. We call the dangling side with the smallest ID the *s-side*. In the example of Fig. 6.8, the dangling side 1_2 is the s-side.

By glueing the side 3_1 (the side 1 of the polygon 3) to the s-side 1_2 of the partial polyhedron 2, we obtain a structure illustrated in Fig. 6.9. When a vertex contributed by two dangling sides are contributed by three polyhedra, we call that vertex an illegal vertex. In Fig. 6.9, the illegal vertex is indicated by an open circle. Every vertex of a simple polyhedron is contributed by three polygons. If we proceed decoding with leaving the illegal vertex, then the number of polygons that contribute to that vertex can increase to four, five, six, …, and we cannot construct a simple polyhedron. Therefore, when an illegal vertex is generated, we *rectify* it by glueing together the dangling sides contributing to it as illustrated in Fig. 6.10. As a result, the illegal vertex is removed. We call the structure thus obtained the partial polyhedron 3.

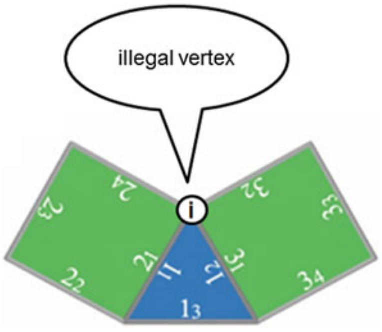


Fig. 6.9 Illegal vertex of a partial polyhedron [11]

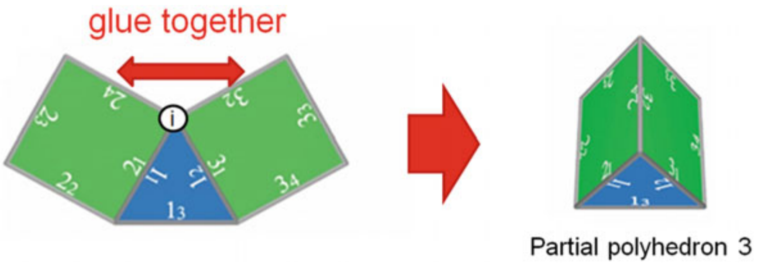


Fig. 6.10 How to rectify an illegal vertex [11]

We then repeat procedures described above. Specifically, we glue the side 4_1 (the side 1 of the polygon 4) to the s-side 1_3 of the partial polyhedron 3. As a result, two illegal vertices are generated. We, therefore, rectify them, and obtain the partial polyhedron 4. Then, we glue the side 5_1 (the side 1 of the polygon 5) to the s-side 2_3 of the partial polyhedron 4, and rectify illegal vertices. As a result, the 34443-polyhedron is completed.

In addition to the number sequence 34443, the polyhedron illustrated in Fig. 6.5 can be constructed from 43434 or 44343. Note that the number sequence 34443 is the sequence of numbers three, four, four, four and three. To give only one unique number sequence to the polyhedron, we regard the number sequences as numbers. Since the 34443, thirty-four thousand four hundred forty-three, is the smallest of three, we define it as the unique number sequence of the polyhedron. Therefore, we call the polyhedron the 34443-polyhedron.

6.2.2.2 Polyhedron Codeword

As we have seen, the polyhedron illustrated in Fig. 6.5 can be represented by 34443. We refer to the number sequence 34443 that represents the polyhedron as the p_3-codeword. The subscript 3 indicates that a polyhedron is a three-dimensional object.

The p_3 formally consists of a polygon-sequence codeword (ps_2) and a side-pairing codeword (sp), and is denoted as

$$p_3 = ps_2; sp.$$

Here, ‘;’ is a separator. The ps_2-codeword is denoted as

$$ps_2 = p_2(1)p_2(2)p_2(3)\ldots p_2(F).$$

Here, $p_2(i)$ is the number of sides of the polygon i. F is the number of faces of the polyhedron, in other words the number of polygons of the polyhedron. Although the formal form is $p_3 = ps_2; sp$, the p_3-codeword of the polyhedron illustrated in Fig. 6.5 consists of only ps_2. In other words, $p_3 = ps_2 = 34443$. There are many polyhedra whose p_3 does not have sp. However, some polyhedra need sp. For example, Tutte’s polyhedron illustrated in Fig. 6.11 is represented by $p_3 = 4555A4559554AA55555454555; E_6 9_6$. Here, A and E indicate 10 and 13, respectively. $sp = E_6 9_6$ instructs that the side E_6 should be glued to the side 9_6. The sp-codeword is formally denoted as

$$sp = y(1)x(1)y(2)x(2)y(3)x(3)\ldots y(N_{\mathrm{na}})x(N_{\mathrm{na}}).$$

Here, we refer to the pair of $y(i)$ and $x(i)$ as a *necessary additional pair* (*necessary a-pair*). Note that a necessary a-pair is identical with a non-curable a-pair of Ref. [18]. To stress that the a-pair is necessary, we call a non-curable a-pair a necessary a-pair in this chapter. The necessary a-pair $y(i)x(i)$ instructs that the sides $y(i)$ and $x(i)$ should be glued together. N_{na} is the number of the necessary a-pairs. Note that $y(i) > x(i)$ and $y(i) < y(i+1)$.

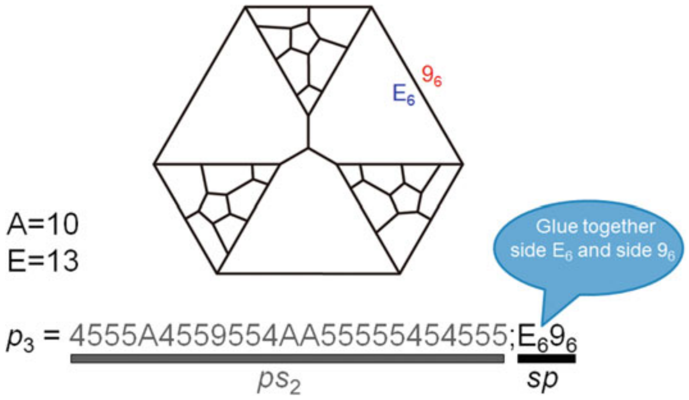

9_6
E_6
A=10
E=13
Glue together side E_6 and side 9_6
p_3 = 4555A4559554AA55555454555;$E_6 9_6$
ps_2
sp

Fig. 6.11 Tutte’s polyhedron [12]

6.2.2.3 Algorithm for Recovering the Original Polyhedron from p_3

In Sect. 6.2.2.1, we have described how to recover the 34443-polyhedron. Here, we describe how to recover the original polyhedron from its $p_3 = ps_2; sp$.

Algorithm A (Fig. 6.12)

1. $i = 1$
   (a) Polygon α is a $p_2(\alpha)$-gon$(1 \leq \alpha \leq F)$.
   (b) Assign IDs $(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{p_2(\alpha)})$ to sides of polygon α in a clockwise direction.
   (c) Polygon 1 is partial polyhedron 1.
2. $i = i + 1$
   (a) Glue side i_1 to the s-side of partial polyhedron $i - 1$.
   (b) When side $y(\beta)$ $(1 \leq \beta \leq N_{\mathrm{na}})$ is a side of polygon i, glue side $y(\beta)$ to side $x(\beta)$.
   (c) Rectify illegal vertices.
   (d) Resultant structure is partial polyhedron i.
3. Repeat procedure 2 until all the polygons are placed.

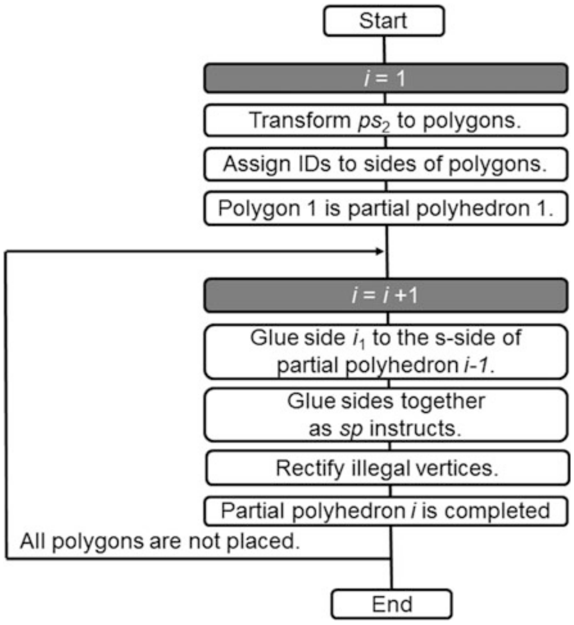

Start
i = 1
Transform ps2 to polygons.
Assign IDs to sides of polygons.
Polygon 1 is partial polyhedron 1.
i = i +1
Glue side i1 to the s-side of partial polyhedron i-1.
Glue sides together as sp instructs.
Rectify illegal vertices.
Partial polyhedron i is completed
All polygons are not placed.
End

Fig. 6.12 Decoding algorithm [12]

6.2.3 Encoding Simple Polyhedra

6.2.3.1 Schlegel Diagram

So far, we have dealt with three-dimensional polyhedra. For convenience, we use Schlegel diagrams [17, 20] to illustrate polyhedra from now on. The Schlegel diagram is the projection of a polyhedron onto a plane. The Schlegel diagram of the 34443-polyhedron is illustrated in Fig. 6.13a. Here, there are two things we should note. First, the outside polygon *abc* of the Schlegel diagram corresponds to the interior of the polygon *abc* of the polyhedron. Second, counterclockwise directions around inside polygons of a Schlegel diagram correspond to clockwise directions around the corresponding polygons of the polyhedron, while a clockwise direction around the outside polygon of a Schlegel diagram corresponds to a clockwise direction around the corresponding polygon of the polyhedron. For example, a travel $z \rightarrow x \rightarrow a \rightarrow c$ in the Schlegel diagram is in a counterclockwise direction. However, the corresponding travel in the polyhedron is in a clockwise direction. On the other hand, a travel $a \rightarrow b \rightarrow c$ around the outside polygon of the Schlegel diagram and the corresponding travel in the polyhedron are both in clockwise directions. Figure 6.13b illustrates the decoding process of the 34443-polyhedron by using Schlegel diagrams.

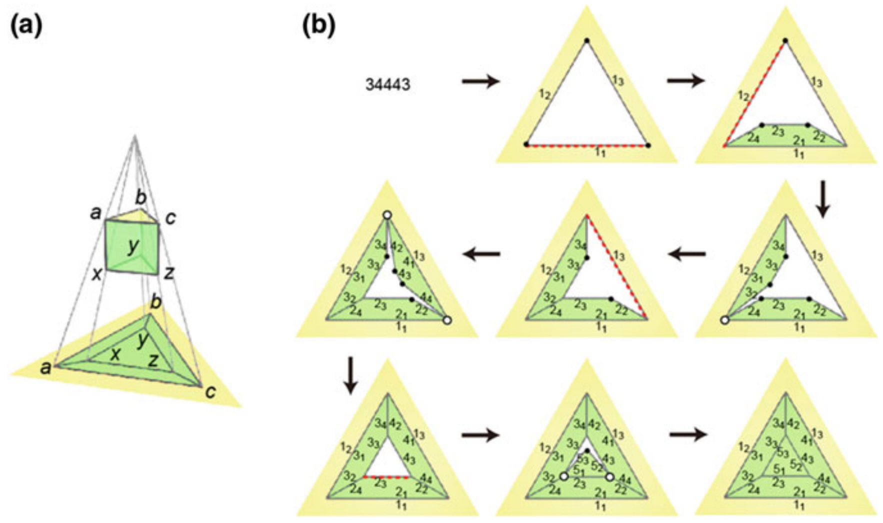


Fig. 6.13 Schlegel diagram [18]. **a** Relation between a polyhedron and its Schlegel diagram. **b** Decoding process illustrated by using Schlegel diagrams. Open circles are illegal vertices. Filled circles are degree two vertices

6.2.3.2 Polygon-Sequence Codeword

When encoding a polyhedron, we first choose a polygon and its side as a seed. Different seeds yield different p_3s. To assign one unique p_3 to that polyhedron, we introduce the lexicographical number $\mathrm{Lex}(p_3)$. We define p_3 with the smallest lexicographical number as the unique p_3 of that polyhedron. We have described the lexicographical numbers of 34443, 43434, and 44343 in Sect. 6.2.2.1. We will describe how to deal with $p_3 = ps_2; sp$ in Sect. 6.2.3.7.

The p_3-codeword consists of ps_2 and sp. We first describe how to generate ps_2. The ps_2-codeword is the sequence of $p_2(i)$ s. Generating ps_2 is, therefore, assigning IDs to polygons. To distinguish between polygons to which IDs have already been assigned and polygons to which IDs will be assigned later, we use colours. We first colour all the polygons. When an ID is assigned, we make the polygon transparent.

We explain how to generate ps_2 by using the 34443-polyhedron as an example. In Fig. 6.14, polyhedra are expressed by using Schlegel diagrams. We choose the outside polygon and its side indicated by the arrow as a seed (Fig. 6.14a). The polygon chosen as the seed is the polygon 1. Since the outside polygon is a triangle, $p_2(1) = 3$. We then assign IDs to the sides of the polygon 1 from the side chosen as the seed in a clockwise direction, and make the polygon 1 transparent (Fig. 6.14b).

When decoding, we have defined a dangling side as a side that is not glued to another side. In encoding, we define a dangling side as a side of a transparent polygon that is glued to a coloured polygon. In Fig. 6.14b, the sides $1_1, 1_2,$ and 1_3 are dangling sides. Since 1_1 is the lexicographically smallest, the side 1_1 is the

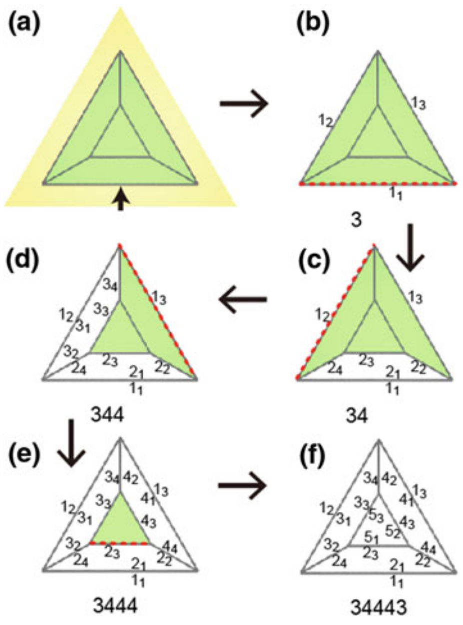


Fig. 6.14 Encoding process [18]. The dashed lines are s-sides

s-side. The polygon 2 is the one that is glued to the s-side 1_1. Since the polygon 2 is a square, $p_2(2) = 4$. We assign IDs to the sides of the polygon 2 from the side glued to the s-side 1_1 in a clockwise direction, and then make the polygon 2 transparent (Fig. 6.14c). We note that the side IDs of the polygon 2 are assigned in a counterclockwise direction in Fig. 6.14c. This is because a counterclockwise direction around any polygon of the Schlegel diagram other than the outside polygon corresponds to a clockwise direction around the corresponding polygon of a polyhedron.

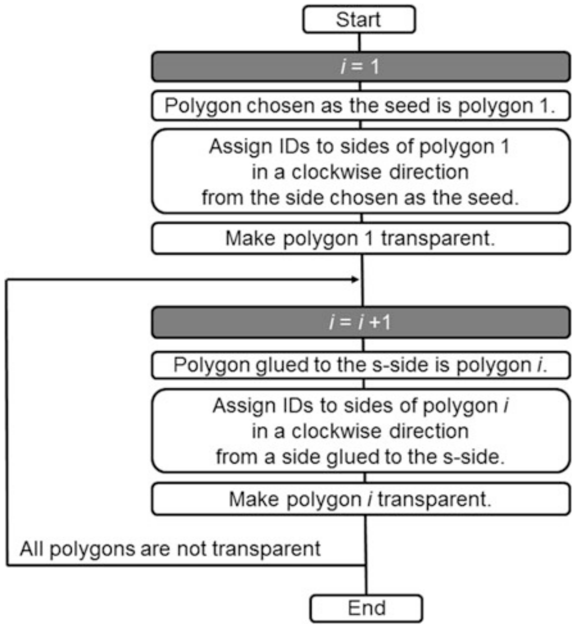
All that is left now is to repeat above described procedures. Specifically, since the polygon that is glued to the s-side 1_2 is the polygon 3, $p_2(3) = 4$. After assigning IDs to its sides, we make the polygon 3 transparent (Fig. 6.14d). The polygon 4 is the one that is glued to the s-side 1_3, and $p_2(4) = 4$. After assigning IDs to its sides, we make the polygon 4 transparent (Fig. 6.14e). The polygon 5 is the one that is glued to the s-side 2_3, and $p_2(5) = 3$. After assigning IDs to its sides, we make the polygon 5 transparent (Fig. 6.14f). All the polygons have become transparent, and $ps_2 = 34443$ has been completed.

To summarize, for a given seed, ps_2 can be generated as follows.

Algorithm B (Fig. 6.15)

1. $i = 1$
   (a) Polygon chosen as a seed is polygon 1.
   (b) Assign IDs $(1_1, 1_2, 1_3, \ldots, 1_{p_2(1)})$ to its sides in a clockwise direction from the side chosen as a seed.

Fig. 6.15 Encoding algorithm [12]


Start
i = 1
Polygon chosen as the seed is polygon 1.
Assign IDs to sides of polygon 1 in a clockwise direction from the side chosen as the seed.
Make polygon 1 transparent.
i = i +1
Polygon glued to the s-side is polygon i.
Assign IDs to sides of polygon i in a clockwise direction from a side glued to the s-side.
Make polygon i transparent.
All polygons are not transparent
End

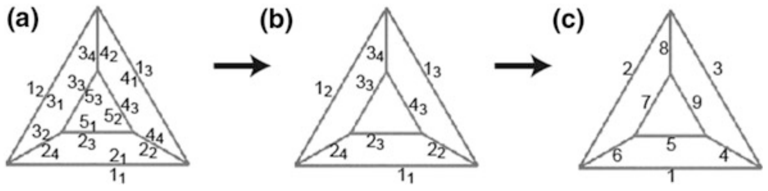


Fig. 6.16 How to assign edge IDs [18]

(c) Make polygon 1 transparent.

2. $i = i + 1$

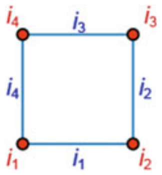
   a. Polygon glued to the s-side is polygon i.
   b. Assign IDs $(i_1, i_2, i_3, \ldots, i_{p_2(i)})$ to its sides in a clockwise direction from the side that is glued to the s-side.
   c. Make polygon i transparent.

3. Repeat the procedure 2 until all the polygons get transparent.

Face and side IDs are assigned by generating ps_2. By using the side IDs, we can assign edge and vertex IDs. The edge IDs will be used when generalizing the p_3-code to the p_4-code for polyhedral assemblages, while vertex IDs will be used when dealing with non-simple polyhedra.

We first describe how to assign edge IDs (Fig. 6.16). Since two side IDs are associated with every edge, we tentatively assign the smaller side ID to the edge (Fig. 6.16b). Since the tentative IDs thus assigned are not in a sequential order, we relabel the IDs so that the edge i is the one with the ith smallest tentative ID (Fig. 6.16c). To assign vertex IDs, we first assign IDs to corners as illustrated in Fig. 6.17. We note that i_1 is assigned to the corner shared by the sides i_1 and $i_{p_2(i)}$. For $1 < j \leq p_2(i)$, i_j is assigned to the corner shared by the sides i_{j-1} and i_j. Since three corner IDs are associated with every vertex, we tentatively assign the smallest corner ID to the vertex, and then relabel the IDs so that the vertex i is the one with the ith smallest tentative ID.

Fig. 6.17 Corner and edge IDs [18]



6.2.3.3 Outline of How to Generate *sp*

To describe how to generate *sp*, we need to introduce simple but a lot of new ideas. One of them is the zeroth tentative *sp* ($tsp^{(0)}$). Although details are given in Ref. [18], we note that $tsp^{(0)}$ is defined so that it has following properties;

1. The original polyhedron can be recovered from $ps_2; tsp^{(0)}$ by using Algorithm A (Sect. 6.2.2.3),
2. but $tsp^{(0)}$ can contain information that is not needed for recovering the original polyhedron.

The *sp*-codeword is obtained by reducing the redundancy in $tsp^{(0)}$.

To describe a little bit more about the relation between *sp* and $tsp^{(0)}$, we introduce the partial polyhedron $D(i)$, polyhedron $P(i)$, and partial polyhedron $E(i)$. When decoding, polygons are glued together one by one. $D(i)$ is the assemblage of polygons obtained when the polygon i is attached. On the other hand, when generating ps_2, the polygons get transparent one by one. $P(i)$ is the polyhedron obtained when the polygon i becomes transparent. We note that encoded polygons of $P(i)$ are transparent, but the others are coloured. $E(i)$ is the assemblage of polygons obtained by removing the coloured polygons from $P(i)$. $P(4)$ of Fig. 6.14e is reproduced in Fig. 6.18a. $E(4)$ is obtained from $P(4)$ by removing the coloured polygon (Fig. 6.18b). Since recognizing transparent polygons is difficult, we coloured polygons of $E(4)$ (Fig. 6.18c). As for partial polyhedra, we do not distinguish between transparent polygons and coloured polygons. We therefore consider that coloured $E(4)$ is identical with $E(4)$.

Now we look at the sequence of partial polyhedra obtained in encoding $E(1)E(2)E(3)\ldots E(F)$ and the sequence of partial polyhedra obtained in decoding $D(1)D(2)D(3)\ldots D(F)$. When decoding from $ps_2; tsp^{(0)}$, $E(i) = D(i)$ for $1 \le i \le F$ (Fig. 6.19a). But, what we need is $E(F) = D(F)$. We therefore admit $E(i) \ne D(i)$ for $i < F$, and reduce redundancy from $tsp^{(0)}$ to obtain *sp* (Fig. 6.19b).

To describe details of $tsp^{(0)}$, we need to describe a-pairs. For this purpose, we first describe a term '*plot*'.

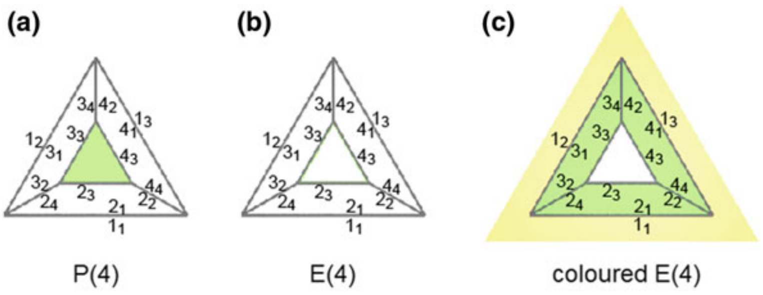


Fig. 6.18 Relation between polyhedron $P(i)$ and partial polyhedron $E(i)$ [18]

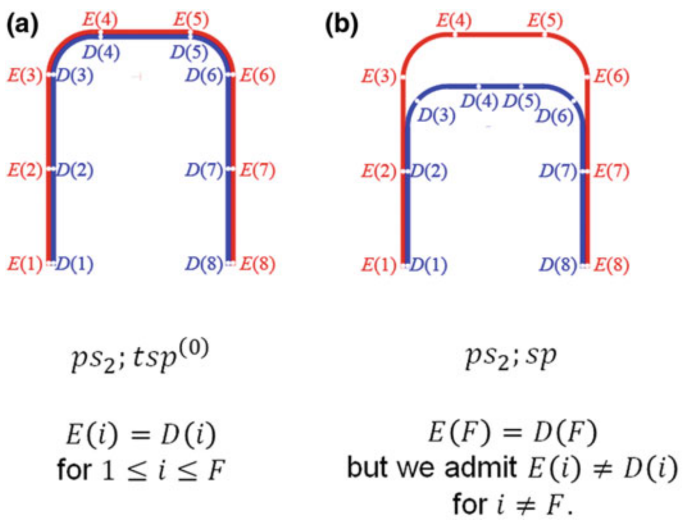

(a)
E(4) E(5) D(4) D(5) E(3) D(3) D(6) E(6) E(2) D(2) D(7) E(7) E(1) D(1) D(8) E(8)
$ps_2; tsp^{(0)}$
$E(i) = D(i)$ for $1 \le i \le F$
(b)
E(4) E(5) E(3) E(6) D(3) D(4) D(5) D(6) E(2) D(2) D(7) E(7) E(1) D(1) D(8) E(8)
$ps_2; sp$
$E(F) = D(F)$ but we admit $E(i) \neq D(i)$ for $i \neq F$.

Fig. 6.19 Comparison between $tsp^{(0)}$ and sp [12]

6.2.3.4 Plot

When two dangling sides of different polygons adjoin each other, we consider that they are *chained*. We call the chain of dangling sides the plot. We also call a separate dangling side the plot. We assign the smallest ID of the dangling sides constituting a plot to that plot. In an example shown in Fig. 6.20, the dangling sides 1_2 and 2_4 of different polygons adjoin each other, so that they are chained. On the other hand, since the dangling sides 1_2 and 1_3 belong to the same polygon, they are not chained. Similarly, the dangling sides 2_3 and 2_4 are not chained. Therefore, the dangling sides 1_2 and 2_4 constitute the plot 1_2. Similarly, the dangling sides 1_4 and 2_2 constitute the plot 1_4. The separate dangling side 1_3 constitutes the plot 1_3 by itself. Similarly, the separate dangling side 2_3 froms the plot 2_3. Here, we point out that the dangling sides of the same plot are all glued to the same polygon.

In Ref. [18], we defined ‘chained’ as follows: two dangling sides are chained when they contribute to the same vertex contributed by two transparent polygons.

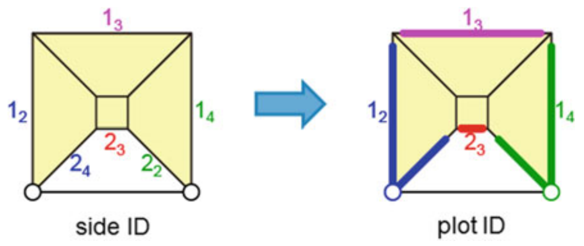


Fig. 6.20 Plot. The pair of Dangling sides contributing to a circle are chained [18]

However, the definition is complicated. In Ref. [12], we have found that the same term ‘chained’ can be defined more briefly, and we use this brief definition as described above.

6.2.3.5 How to Generate $tsp^{(0)}$

The $tsp^{(0)}$-codeword consists of a-pairs. We therefore describe a-pairs. In generating ps_2, polygons of the polyhedron get transparent one by one. The process can be represented as $P(1)P(2)P(3)\ldots P(F)$. The a-pairs relate to how the polygon i is glued to the other polygons in $P(i-1)$. To explain this, we generate ps_2 of a polyhedron illustrated in Fig. 6.21a twice with choosing the outside polygon and the side indicated by the arrow as a seed. When the first generation is finished, IDs are assigned to all the polygons and sides. Therefore, we can perform the second generation with knowing all the IDs in advance. Figure 6.21b illustrates $P(1)$, which is obtained when polygon 1 becomes transparent. In $P(1)$, the dangling sides 1_1, 1_2, 1_3 and 1_4 constitute the plots 1_1, 1_2, 1_3 and 1_4, respectively. The smallest ID plot (*s-plot* for short) is the plot 1_1, to which the polygon 2 is glued. In general, the polygon i is glued to the s-plot of $P(i-1)$. This is because, by definition, the polygon that is glued to the s-side of $P(i-1)$ is the polygon i. Figure 6.21c illustrates $P(7)$, where the polygon 8 is glued to the s-plot 3_4. In addition to the s-plot, the polygon 8 is glued to the plot 5_6, which we call an additional plot. In general, the additional plots of $P(i-1)$ are plots other than the s-plot to which the polygon i is glued. By definition, the smallest ID of dangling sides constituting the additional plot 5_6 is 5_6. The side 5_6 is glued to the side 8_5 of the polygon 8, and we refer to the pair of the sides 8_5 and 5_6 as the a-pair 8_55_6. Note that the lexicographically larger 8_5 proceeds 5_6. As is illustrated in Fig. 6.21d, 10_45_4 is also an a-pair. When generating ps_2 of the polyhedron illustrated in Fig. 6.21a, 8_55_6 and 10_45_4 are a-pairs. By collecting the a-pairs, $tsp^{(0)} = 8_55_610_45_4$.

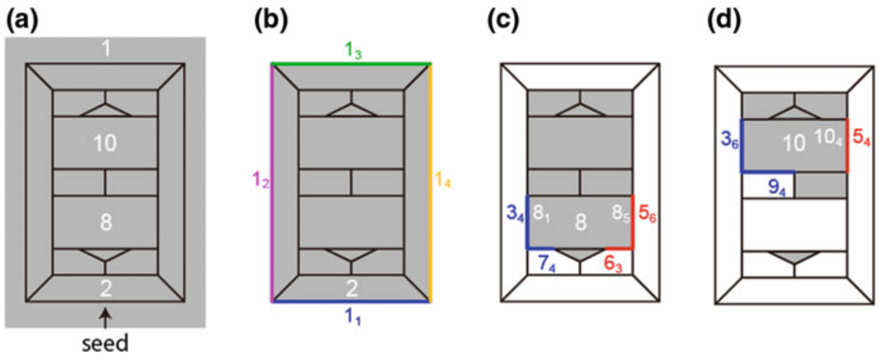


Fig. 6.21 Explanation of $tsp^{(0)}$ [18]

The $tsp^{(0)}$-codeword is formally denoted as

$$tsp^{(0)} = y_a(1)x_a(1)y_a(2)x_a(2)y_a(3)x_a(3)\ldots y_a(N_a)x_a(N_a).$$

Here, $y_a(i)x_a(i)$ is the ith a-pair, where $y_a(i) > x_a(i)$ and $y_a(i) < y_a(i+1)$. N_a is the number of a-pairs.

6.2.3.6 How to Generate *sp*

As described above, when we encode the polyhedron illustrated in Fig. 6.21a, we obtain $ps_2; tsp^{(0)} = 458585574755433; 8_5 5_6 10_4 5_4$. The original polyhedron can be recovered from $ps_2; tsp^{(0)}$ using Algorithm A described in Sect. 6.2.2.3 (See Ref. [18] for its proof). But $tsp^{(0)}$ can contain information that is not needed for recovering the original polyhedron.

We now examine whether $10_4 5_4$ is necessary or not. To do so, we try to decode from $458585574755433; 8_5 5_6$ which is obtained by removing $10_4 5_4$ from $ps_2; tsp^{(0)}$ (Fig. 6.22). Since it does not have the information of the a-pair $10_4 5_4$, the sides 10_4 and 5_4 are not glued together in $D(10)$, which is obtained when the polygon 10 is placed. And then the polygon 13 is glued to the side 5_4 in $D(13)$. This means that we cannot recover the original polyhedron without $10_4 5_4$. On the other hand, when we decode $458585574755433; 10_4 5_4$ which is obtained by removing $8_5 5_6$ from $ps_2; tsp^{(0)}$, the sides 8_5 and 5_6 are separate in $D(8)$, but they are glued together in $D(13)$, and the original polyhedron can be recovered (Fig. 6.23). This means that $8_5 5_6$ is not necessary. By removing the unnecessary $8_5 5_6$ from $tsp^{(0)}$, *sp* is obtained as $10_4 5_4$.

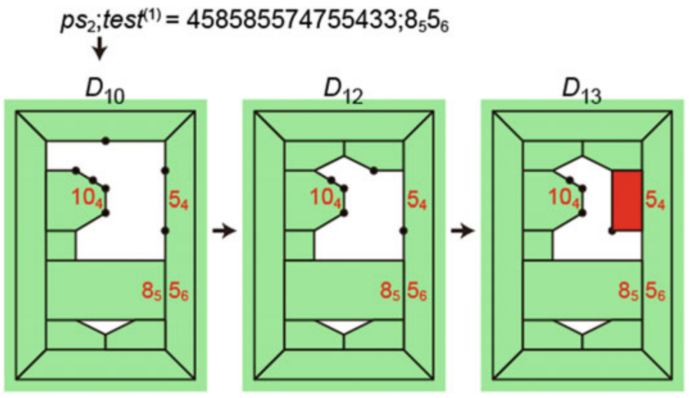


Fig. 6.22 Necessary a-pair [18]

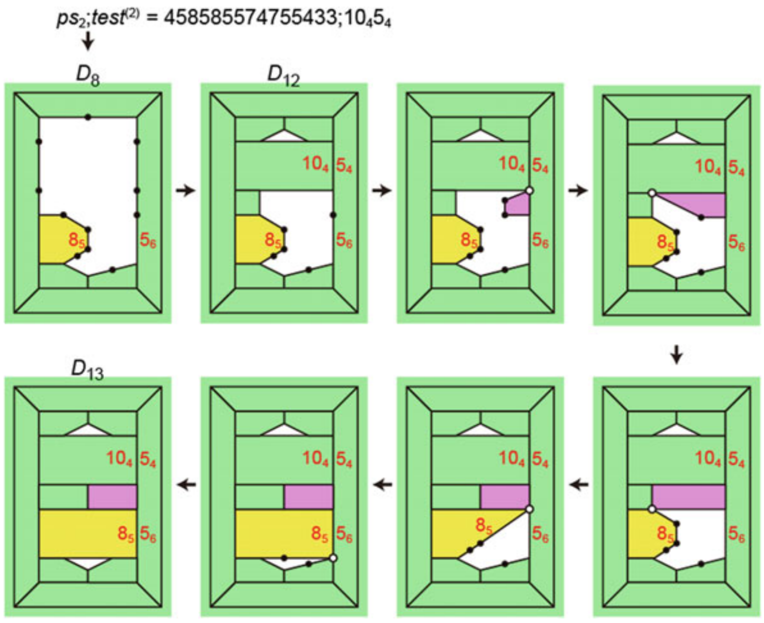


Fig. 6.23 Unnecessary a-pair [18]

For a given $tsp^{(0)}$, sp can be generated as follows.
Algorithm C

1. $i = 0$
   (a) $tsp^{(0)} = y_a(1)x_a(1)y_a(2)x_a(2)y_a(3)x_a(3)\ldots y_a(N_a)x_a(N_a)$.
2. $i = i + 1$
   (a) Construct $test^{(i)}$ from $tsp^{(i-1)}$ by removing $y_a(N_a - i + 1)x_a(N_a - i + 1)$.
   (b) Decode from $ps_2; test^{(i)}$.
      ① If the original polyhedron is recovered, then $tsp^{(i)} = test^{(i)}$.
      ② Otherwise, $tsp^{(i)} = tsp^{(i-1)}$.
3. Repeat the procedure 2, until we obtain $tsp^{(N_a)} = sp$.

The sp-codeword is obtained from $tsp^{(0)}$ by removing unnecessary a-pairs. In other words, sp consists of necessary a-pairs.

6.2.3.7 Lexicographical Number of p_3

Different seeds yield different p_3s. To assign one unique p_3 to a polyhedron, we describe the lexicographical number $\mathrm{Lex}(p_3)$. We regard $\mathrm{Lex}(p_3)$ as a base-n number, where n is any sufficiently large number as described below. Since p_3 consists of ps_2 and sp, $\mathrm{Lex}(p_3)$ consists of $\mathrm{Lex}(ps_2)$ and $\mathrm{Lex}(sp)$. We first describe $\mathrm{Lex}(ps_2)$. Since ps_2 is the sequence of F numbers, we define $\mathrm{Lex}(ps_2)$ as a F-digit base-n number $\mathrm{Lex}(ps_2) = p_2(1)p_2(2)p_2(3)\ldots p_2(F)$, where $p_2(i)$ is the value of the $(F - i + 1)$th digit. Note that $p_2(i)$ in the number sequence $ps_2 = p_2(1)p_2(2)p_2(3)\ldots p_2(F)$, is the ith number. Similarly, $\mathrm{Lex}(sp)$ is a $2N_{\mathrm{na}}$-digit base-n number $y(1)x(1)y(2)x(2)y(3)x(3)\ldots y(N_{\mathrm{na}})x(N_{\mathrm{na}})$. $\mathrm{Lex}(p_3)$ is the concatenation of $\mathrm{Lex}(ps_2)$ and $\mathrm{Lex}(sp)$. For reference, the concatenation of 24 expressed in base-10 (twenty-four) and 5 expressed in base-10 (five) is 245 (two hundred forty-five).

Note that, to regard $\mathrm{Lex}(p_3)$ as a base-n number, n should be larger than $p_2(i)$, $y(i)$ and $x(i)$. Since the number of sides of a polyhedron is $2E = 6(F - 2)$, n should be larger than $2E$. Here, E is the number of edges of a polyhedron.

Since there are $2E$ different selections of seeds, $2E$ different p_3s can be generated from a polyhedron. If we consider that the polyhedron is identical with its mirror image, additional $2E$ different p_3s can be generated from the mirror image. By selecting the smallest of $4E$ different p_3s, we can assign one unique p_3 to the polyhedron. By considering that a polyhedron is identical with its mirror image, the unique p_3s can be used to determine the isomorphism of polyhedral graphs. A polyhedral graph is a planer triply connected graph that has no multiple edges. If we regard a region enclosed by two edges as a 2-gon, p_3s can be used to determine the isomorphism of planer triply connected graphs. When we want to distinguish the polyhedron from its mirror image, we may select the smallest of $2E$ different p_3s. But the unique p_3s thus generated cannot be used to determine the isomorphism of polyhedral graphs.

6.2.3.8 Solving the Problem of Voronoi Index

As is shown in Fig. 6.24, two different polyhedra have the same Voronoi index $\langle 0282000\ldots\rangle$. By using p_3s, we can say that 455665555455- and

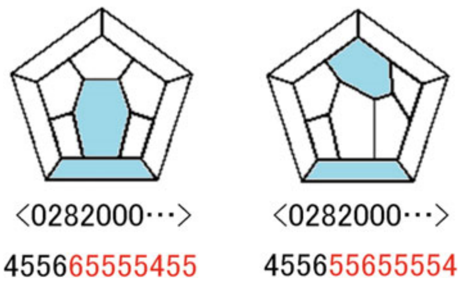


Fig. 6.24 Solving the problem of Voronoi index [12]

455655655554-polyhedra have the same Voronoi index $\langle 0282000\ldots\rangle$. In other words, different polyhedra can be distinguished by using our method.

6.2.4 Non-simple Polyhedron

6.2.4.1 Cut-and-Dot Method

So far, we have assumed that polyhedra are simple. The theory for simple polyhedra can be easily generalized to non-simple polyhedra that have one or more vertices of degree more than three. Figure 6.25 illustrates a pentagonal pyramid. Since the apex is degree five, the pentagonal pyramid is a non-simple polyhedron. But when we cut the apex, a simple polyhedron can be obtained. By distinguishing the cross section from other faces, we can establish a one-to-one correspondence between a non-simple polyhedron and a simple polyhedron with a cross section. Using this relation, the non-simple polyhedron can be represented by $544444\dot{5}$. The dot over '5' indicates that the pentagon is a cross section which should be shrunk to a vertex. Note that this approach was inspired by Kempe's patch method for the four colour problem [17].

When dealing with polyhedra without cross sections, we have defined the s-side as the smallest ID dangling side. As for polyhedra with cross sections, we define the s-side as the smallest ID dangling side of polygons that are not cross sections. The reason is given later. A non-simple polyhedron can be encoded as follows:

1. Cut vertices of degree more than three.
2. Choose a polygon that is not a cross section and its side that does not contribute to any edge of the cross sections as a seed.
3. Generate p_3 using Algorithm B.
4. Put dots over numbers of ps_2 corresponding to the cross sections.

By encoding, IDs can be assigned to faces, edges and vertices of a non-simple polyhedron as follows (Fig. 6.26):

1. Assign IDs to a polyhedron with cross sections.
2. Shrink cross sections to vertices. As a result, some IDs disappear.

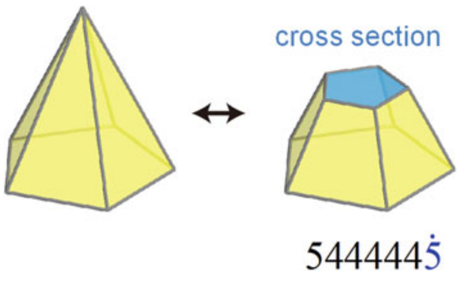


Fig. 6.25 One-to-one corresponding between a non-simple polyhedron and a simple polyhedron with a cross section [18]

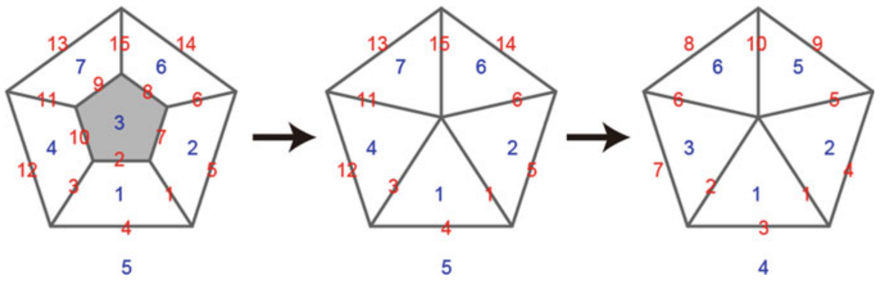


Fig. 6.26 How to assign IDs to a non-simple polyhedron [18]

3. Relabel IDs so that they become sequential orders.

We have modified the definition of the s-side. As a result, IDs assigned using the method described above conform to IDs assigned by directly applying Algorithm B to the non-simple polyhedron. Note that a codeword can be generated by directly applying Algorithm B to a non-simple polyhedron. However, the original polyhedron cannot be recovered from the codeword using Algorithm A. For example, a pentagonal pyramid can be encoded as 533333. But it cannot be recovered from 533333.

To assign one unique codeword to a non-simple polyhedron, we define $\mathrm{Lex}(p_3)$ for p_3 with dots. For this purpose, we define $\mathrm{Lex}(ps_2)$ as the concatenation of $\mathrm{Lex}\left(ps_2^{(1)}\right)$ and $\mathrm{Lex}\left(ps_2^{(2)}\right)$. The $ps_2^{(1)}$-codeword is obtained from ps_2 by replacing every number without a dot by 0 and then removing all dots, while $ps_2^{(2)}$ is obtained by removing all dots from ps_2. For example, when $ps_2 = 544444\dot{5}$, $ps_2^{(1)} = 0000005$ and $ps_2^{(1)} = 5444445$. Therefore, $\mathrm{Lex}(ps_2) = 00000055444445$.

6.2.4.2 Using Duality

The cut-and-dot method described in the previous section is applicable to all non-simple polyhedra, but is sometimes inefficient. For example, the octahedron is encoded as $66\dot{4}6\dot{4}6\dot{4}6\dot{4}6\dot{4}6\dot{4}6$. Since 6 is repeated twice and then $\dot{4}6$ is repeated six times, $66\dot{4}6\dot{4}6\dot{4}6\dot{4}6\dot{4}6\dot{4}6$ can be shortened to $6^2(\dot{4}6)^6$. However, the representation of the octahedron with beautiful symmetries is not beautiful. We think that it is a problem. To overcome this problem, we use the duality of polyhedra [17, 21]. Since the octahedron is the dual of the hexahedron, we represent the octahedron as $\bigstar 4^6$. Here, $\bigstar$ represents the dual, 4^6 is p_3 of the hexahedron, and $\bigstar 4^6$ means the dual of the 4^6-polyhedron. We describe the details of this method below.

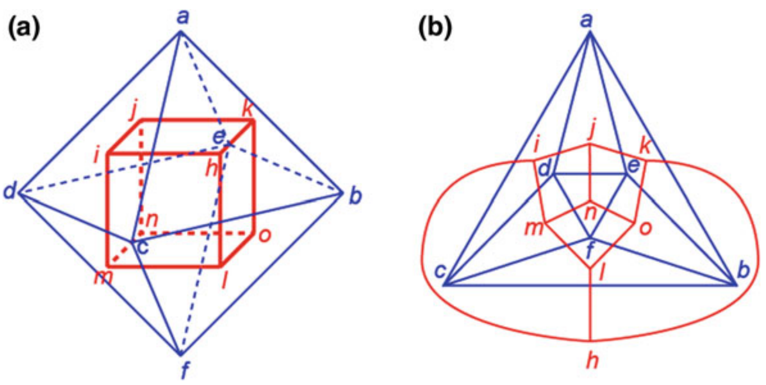


Fig. 6.27 Duality. **a** The dual of the octahedron is the hexahedron. **b** Graph representation of (**a**). The octahedron and hexahedron are dual to each other [18]

For any polyhedron, its dual is constructed as follows (Fig. 6.27):

1. Put a vertex v_i^* of the dual polyhedron on the face f_i of the given polyhedron.
2. Link v_i^* and v_j^* by the edge e_{ij}^*, when f_i and f_j share an edge e.

The dual of the octahedron is the hexahedron, and the dual of the hexahedron is the octahedron. Thus, the octahedron and hexahedron are dual to each other. By using ★, a polyhedron composed of triangles only can be briefly represented, for its dual is a simple polyhedron.

Since there is a one-to-one correspondence between an original polyhedron and its dual, we can determine the edge and face IDs of the original from those of its dual. For example, in an example shown in Fig. 6.27b, the face *dcf* of the octahedron corresponds to the vertex *m* of the hexahedron. We, therefore, assign the ID of the vertex *m* to the face *dcf*. Similarly, the edge *dc* of the octahedron corresponds to the edge *mi*. We, therefore, assign the ID of the edge *mi* to the edge *dc*.

When we encode a simple polyhedron, we first choose a seed, and then generate p_3 from the seed. The side chosen as a seed contributes to the edge 1, and the polygon chosen as a seed becomes the polygon 1. Therefore, choosing a seed is determining the edge 1 and face 1. When we encode a non-simple polyhedron using the duality, we also choose a side and polygon of the non-simple polyhedron as a seed. We then choose a seed for its dual so that the edge of the dual corresponding to the edge of the original contributed by the side chosen as a seed becomes the edge 1 and that the vertex of the dual corresponding to the polygon of the original chosen as a seed becomes the vertex 1. For example, when we encode the octahedron with choosing the polygon *dcf* and its side *dc* as a seed, the polygon *mihl* and its side *mi* is a seed for its dual. Then 1 is assigned to the vertex *m* of the dual, and therefore to the polygon *dcf* of the original. Similarly, 1 is assigned to the edge *mi* of the dual, and therefore to the edge *dc* of the original contributed by the side *dc*.

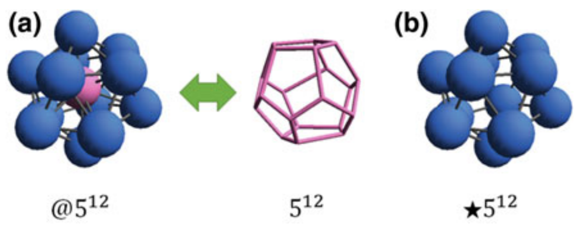


Fig. 6.28 Relation between a local atomic arrangement and a Voronoi polyhedron [12]. **a** The Voronoi polyhedron associated with the pink atom is 5^{12}. The pink and its neighbouring atoms form a @5^{12} cluster. **b** The atoms adjacent to the pink atom occupy the vertices of a ★5^{12}-polyhedron

Note that in Ref. [18], we determine the seed for the dual in a different way. But for simplicity, we have modified the method in Ref. [12], and the modified version is described above.

To assign one unique p_3 to any non-simple polyhedron, we assume Lex(★) = 1 and define Lex(★p_3) as the concatenation of Lex(★) and Lex(p_3). By doing so, Lex(★4^6) = 1444444, while $\mathrm{Lex}\left(6^2(\dot{4}6)^6\right) = 00404040404040664646464646464$. Therefore, the unique p_3 of the octahedron is ★4^6.

6.2.5 *Relation Between an Atomic Arrangement and a Voronoi Polyhedron*

To represent atomic arrangements, we introduce the symbol @ that relates a Voronoi polyhedron and its corresponding atomic arrangement as follows. We refer to the Voronoi polyhedron associated with the atom i as the Voronoi polyhedron i. In other words, the atom i and its nearest neighbour atoms define the Voronoi polyhedra i. When the Voronoi polyhedron i is a p_3-polyhedron, we represent the arrangement of atoms defining the Voronoi polyhedron, namely the atom i and its first nearest neighbour atoms, as an @p_3-cluster (Fig. 6.28a). Note that, in the @p_3-cluster, first nearest neighbour atoms of the atom i occupy the vertices of a ★p_3-polyhedron and the atom i locates at the centre of the ★p_3-polyhedron (Fig. 6.28b).

6.3 Polychoron Code

We can study the short-range order of amorphous materials by classifying the Voronoi polyhedra with the p_3-code. We can study the long-range order by classifying assemblages of Voronoi polyhedra. A polyhedral assemblage can be regarded as a part of a polychoron (four-dimensional polytope). The p_3-code for polyhedra can be easily generalized to deal with polychora, for it is based on the hierarchy of structures of polytopes: a polyhedron (three-dimensional polytope) is an assemblage of polygons (two-dimensional polytopes). In this section, we generalize the p_3-code for polyhedra to the p_4-code for polychora. The p_4-code consists of the encoding algorithm for converting a polychoron into p_4 and the decoding algorithm for recovering the original polychoron from its p_4. The p_4-code can be used to study the long-range order of amorphous materials.

Since we are living in the three-dimensional world, understanding four-dimensional objects is not easy. But understanding Schlegel diagrams of polychora is not difficult. As shown in Fig. 6.13a, a polyhedron can be represented as a two-dimensional object by using a Schlegel diagram. Similarly, a polychoron can be represented as a three-dimensional object by using a Schlegel diagram. The Schlegel diagram of a polychoron *abcdefgh* is illustrated in Fig. 6.29. We can see that the polychoron is an assemblage of two 3333-polyhedra and four 34443-polyhedra, and polyhedra are glued face to face. We note that the outside of the polyhedron *abcd* of the Schlegel diagram corresponds to the inside of the polyhedron *abcd* of the polychoron. By using the p_4-code, the polychoron *abcdefgh* is represented by p_4 = 3333 34443 34443 34443 34443 3333. The p_4-codeword is the sequence of p_3s and instructs how to construct the polychoron from its building-block polyhedra. Since the left most p_3 is 3333, the polyhedron 1 is a 3333-polyhedron. Similarly, the polyhedra 2, 3, 4, and 5 are 34443-polyhedra, and the polyhedron 6 is a 3333-polyhedra. To describe the p_4-code, we first describe the relations between parts of polyhedra and parts of a polychoron.

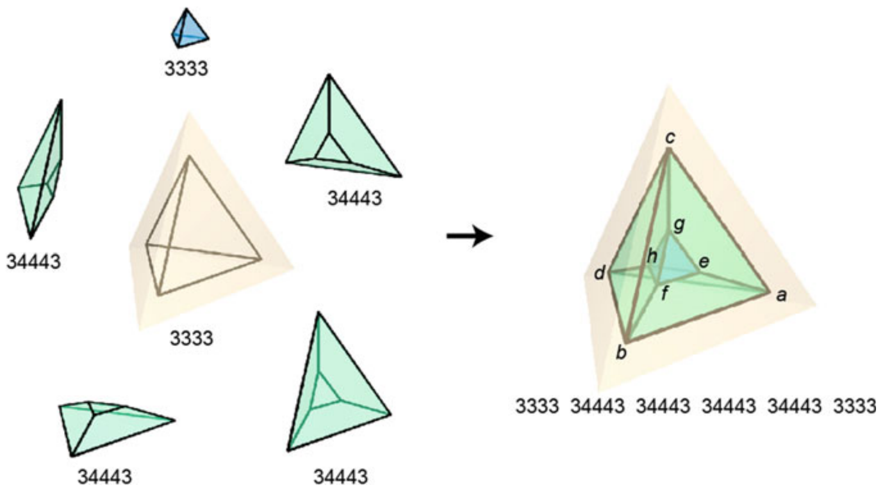


Fig. 6.29 Polychoron represented by using a Schlegel diagram [19]

6.3.1 Our Way of Viewing a Polychoron

We regard a polychoron as a tiling by polyhedra of the surface of a four-dimensional object that is topologically the same as a 3-sphere. We assume that the polyhedra are glued together such that (1) any pair of polyhedra meet only at their faces, edges, or vertices and that (2) each face of each polyhedron meets exactly one other polyhedron along a ridge. We distinguish parts of a polychoron and parts of its building-block polyhedra (Fig. 6.30). The 0-face is a point of a polychoron where the vertices of polyhedra meet; the peak is a line segment of a polychoron where the edges of polyhedra meet; the ridge is an area of a polychoron where the faces of polyhedra meet. The cell of a polychoron is a polyhedron.

6.3.2 1-Simple Polychoron

Polyhedra can be classified into simple and non-simple polyhedra according to the degrees of the vertices. As described above, we have first created the method for simple polyhedra, and then generalized it for non-simple polyhedra. Polychora can be classified into simple and non-simple polychora according to the types of the

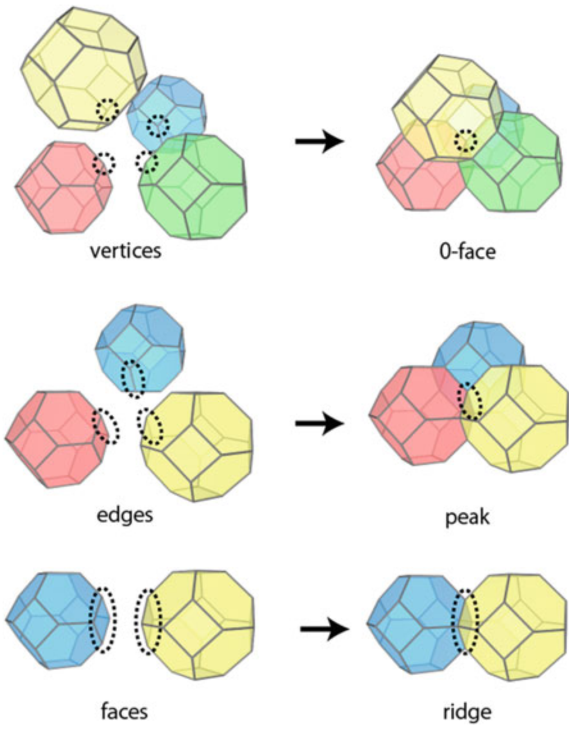


Fig. 6.30 Parts of polyhedra and parts of a polychoron [18]

0-faces as well. A polychoron whose 0-faces are all degree four is called a simple polychoron. Here, the degree of a 0-face is the number of peaks incident to that 0-face. However, to describe the p_4-code, we need to classify polychora according to the types of the peaks. We, therefore, generalize the concept of ‘simple’. For this purpose, we first define the degree of a peak as the number of ridges incident to that peak. We then call a polychoron whose peaks are all degree three a *1-simple* polychoron. The ‘1’ indicates the simplicity regarding one-dimensional parts of a polychoron, namely peaks or 1-faces. We first describe the method for 1-simple polychora, and then briefly describe the generalization of the method for non-1-simple polychora. In the method for 1-simple polychora, we use the property that every peak of a 1-simple polychoron is contributed by three edges.

Note that, using our generalized notation, a simple polychoron is called a 0-simple polychoron. Here, ‘0’ means the simplicity regarding zero-dimensional parts of a polychoron, namely 0-faces. Similarly, a simple polyhedron is a 0-simple polyhedron. If a polychoron is 0-simple, then it is also 1-simple, for its peaks are all degree three. However, a 1-simple polychoron is not always 0-simple. In other words, a set of 0-simple polychora is a subset of 1-simple polychora. For example, the 24-cell composed of 24 octahedra is 1-simple, but is not 0-simple. The polyhedra of a 0-simple polychoron are all 0-simple. On the other hand, the polyhedra of a 1-simple polychoron can be non-0-simple. For example, the octahedra of the 1-simple 24-cell are non-0-simple.

We also note that the definition of the peak degree described above is different from the definition given in Ref. [18]: the degree of a peak is the number of polyhedra contributing to that peak. In the previous definition, the polyhedra (building blocks of a polychoron) are focused. The previous definition, therefore, does not match the definition of the vertex degree where the edges (parts of a polyhedron) are focused: the degree of a vertex is the number of edges incident to that vertex. To match the definition of the vertex degree, namely to focus on the parts of a polychoron, we have introduced the new definition. In addition, in Ref. [18], we used the term ‘a non-affected polychoron’ to mean a 1-simple polychoron. However, since our new terminology is suitable for the systematic description of fundamental characteristics of polytopes, we use ‘1-simple’ instead of ‘non-affected’.

6.3.3 Polychoron Codeword

The p_4-code is a method for converting a way of how polyhedra are arranged to form a polychoron into p_4 from which the original structure can be recovered. The p_4-codeword consists of polyhedron-sequence codeword (ps_3) and face-pairing codeword (fp), and is denoted as

$$p_4 = ps_3;fp.$$

Here, ps_3 is denoted as

$$ps_3 = p_3(1)p_3(2)p_3(3)\dots p_3(C),$$

where $p_3(i)$ is p_3 of polyhedron i, C is the number of cells of a polychoron, in other words the number of polyhedra of a polychoron. The *fp*-codeword is denoted as

$$fp = w(1)z(1)v(1)w(2)z(2)v(2)w(3)z(3)v(3)\dots w(N_{\mathrm{na}})z(N_{\mathrm{na}})v(N_{\mathrm{na}}).$$

Here, $w(i)z(i)v(i)$ is the necessary a-pair for the polychoron, instructing that the face $w(i)$ and face $v(i)$ should be glued together in such a way that the edge of the face $w(i)$ contributed by the side $z(i)$ is glued to the smallest ID edge of the face $v(i)$. N_{na} is the number of necessary a-pairs. Note that $w(i) > v(i)$ and $w(i) < w(i+1)$.

6.3.4 How to Generate ps_3

To encode a polychoron, we first chose a polyhedron, face of the polyhedron, and edge of the face as a seed. Depending on how we choose the seed, p_4 changes. By using lexicographical numbers described in Sect. 6.3.8, one unique p_4 can be assigned to each polychoron. The p_4-codeword consists of ps_3 and *fp*. We first describe how to generate ps_3.

Generating ps_3 is assigning IDs to polyhedra. We use colours to distinguish between polyhedra to which IDs have already been assigned and polyhedra to which IDs will be assigned later. We first colour all the polyhedra. When an ID is assigned, we make the polyhedron transparent (Fig. 6.31). We refer to the face of a transparent polyhedron glued to a coloured polyhedron as a dangling face. To instruct how to assign IDs to polyhedra, we assign IDs to the faces (edges) of every polyhedron by encoding the polyhedron. Specifically, we assign i_j to the jth face (edge) of the polyhedron i. The dangling face with the smallest ID is called the *s-face*.

For a given seed, ps_3 can be generated as follows.

Algorithm D

```
1. $i = 1$
    a. Polyhedron chosen as the seed is polyhedron 1.
    b. Generate $p_3$ of polyhedron 1 and assign IDs to its faces (edges) by encoding
       polyhedron 1 in such a way that $1_1$ is assigned to the face (edge) chosen as
       the seed.
    c. Make polyhedron 1 transparent.
```

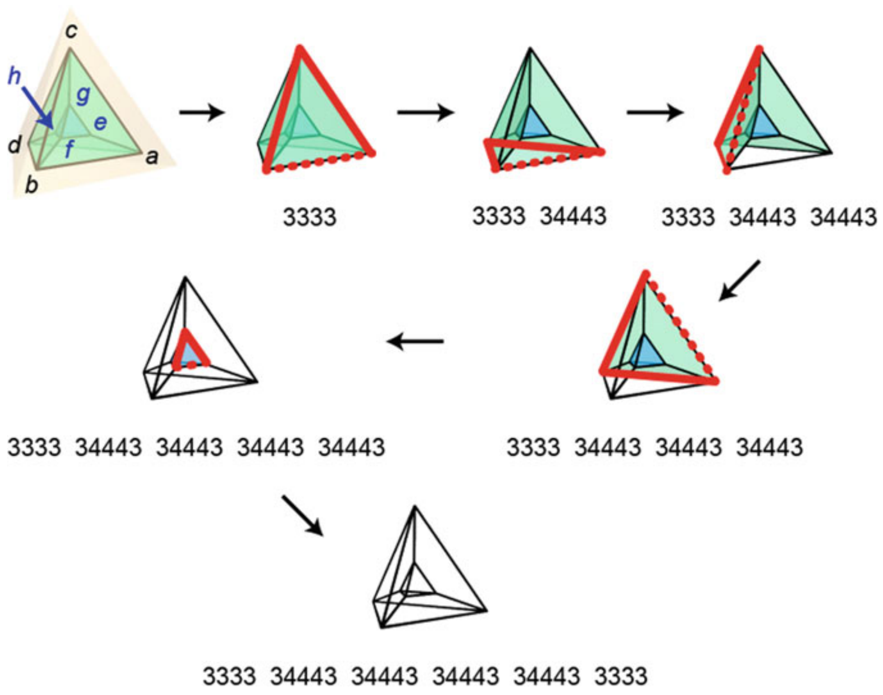


Fig. 6.31 Encoding a polychoron. The edges of each s-face are indicated by red lines. The smallest ID edge of each s-face is indicated by a dotted line [18]

2. $i = i + 1$

   a. Polyhedron glued to the s-face is polyhedron i.
   a. Generate p_3 of polyhedron i and assign IDs to its faces (edges) by encoding polyhedron i in such a way that i_1 is assigned to the face glued to the s-face (the edge glued to the smallest ID edge of the s-face).
   b. Make polyhedron i transparent.

3. Repeat the procedure 2 until all the polyhedra get transparent.

6.3.5 How to Generate $tfp^{(0)}$

To describe *fp*, we introduce a zeroth tentative *fp* ($tfp^{(0)}$). For this purpose, we introduce some ideas. When two dangling faces of different polyhedra adjoin each other, we consider that they are chained. The chained dangling faces constitute a plot. A separate dangling face also constitutes a plot by itself. We assign the smallest face ID of all the dangling faces constituting a plot to that plot.

In generating ps_3, polyhedra of the polychoron get transparent one by one. We consider a polychoron $P_4(i-1)$ obtained when the $(i-1)$th polyhedron gets transparent. The polyhedron i is glued to the s-plot of $P_4(i-1)$. When the polyhedron i is glued to plots other than the s-plot, those plots are the a-plots for the polychoron. Suppose that the face w_a of the polyhedron i is glued to the face v_a of the a-plot v_a in such a way that the edge contributed by the side z_a of the polygon w_a is glued to the smallest ID edge of the face v_a. Then the pair of $w_a z_a v_a$ is the a-pair for the polychoron. The $tfp^{(0)}$-codeword is obtained by collecting the a-pairs in such a way that $w_a(i) < w_a(i-1)$;

$$tfp^{(0)} = w_a(1)z_a(1)v_a(1)\ldots w_a(N_\mathrm{a})z_a(N_\mathrm{a})v_a(N_\mathrm{a}).$$

Here, N_a is the number of a-pairs.

6.3.6 How to Recover a Polychoron from $ps_3; tsp^{(0)}$

To describe how to recover a polychoron from its $ps_3; tsp^{(0)}$, we first describe the dangling face for decoding and an illegal peak. In decoding, we call a face that is not glued to another polyhedron a dangling face. When a peak contributed by two dangling faces is also contributed by three polyhedra, we call that peak an illegal peak. The illegal peak can be rectified by glueing the two dangling faces together.

The polychoron can be recovered from its $ps_3; tsp^{(0)}$ as follows:

Algorithm E

1. $i = 1$
   a. Polyhedron α is a $p_3(\alpha)$-polyhedron $(1 \le \alpha \le C)$.
   b. Assign α_j to the jth face (edge) of polyhedron α.
   c. Polyhedron 1 is partial polychoron 1.
2. $i = i + 1$
   a. Glue face i_i (face 1 of polyhedron i) to the s-face of partial polychoron $i-1$.
   c. When $w_a(\beta)$ is the face ID of polyhedron i $(1 \le \beta \le N_\mathrm{a})$, glue together faces $w_a(\beta)$ and $v_a(\beta)$ in such a way that the edge contributed by side $z_a(\beta)$ is glued to the smallest ID edge of the face $v_a(\beta)$.
   d. Rectify illegal peaks.
   e. Structure thus obtained is partial polychoron i.
3. Repeat the procedure 2 until all the polyhedra are placed.

For reference, recovering a polychoron from 3333 34443 34443 34443 34443 3333 is illustrated in Supplemental Information of Ref. [18].

6.3.7 How to Generate fp

The original polychoron can be recovered from $ps_3; tfp^{(0)}$ by using Algorithm E. However, $tfp^{(0)}$ can contain information that is not necessary for recovering the original polychoron. The *fp*-codeword is obtained by reducing the redundancy in $tfp^{(0)}$ as follows;

Algorithm F

```
1. $i = 0$
   a. $tfp^{(0)} = w_a(1)z_a(1)v_a(1)\ldots w_a(N_{\mathrm{a}})z_a(N_{\mathrm{a}})v_a(N_{\mathrm{a}})$.
2. $i = i + 1$
   a. Construct $test^{(i)}$ by removing $w_a(N_a - i + 1)z_a(N_a - i + 1)v_a(N_a - i + 1)$ from $tfp^{(i-1)}$.
   b. Decode from $ps_3; test^{(i)}$.
      i. If the original polychoron is recovered, then $tfp^{(i)} = test^{(i)}$.
      ii. Otherwise, $tfp^{(i)} = tfp^{(i-1)}$.
3. Repeat the procedure 2 until $fp = tfp^{(N_{\mathrm{a}})}$ is obtained.
```

6.3.8 Lexicographical Number of p_4

Different p_4s are generated from different seeds. To determine one unique p_4, we define $\mathrm{Lex}(p_4)$ as follows. Since $p_4 = ps_3; fp$, $\mathrm{Lex}(p_4)$ is the concatenation of $\mathrm{Lex}(ps_3)$ and $\mathrm{Lex}(fp)$. Since ps_3 is the sequence of $p_3(i)$s, $\mathrm{Lex}(ps_3)$ is a C-digit number $\mathrm{Lex}(p_3(1))\mathrm{Lex}(p_3(2))\mathrm{Lex}(p_3(3))\ldots\mathrm{Lex}(p_3(C))$, where $\mathrm{Lex}(p_3(i))$ is the value of the $(C - i + 1)$th digit. Similarly, $\mathrm{Lex}(fp)$ is a $3N_{\mathrm{a}}$-digit number $\mathrm{Lex}(w(1))\mathrm{Lex}(z(1))\mathrm{Lex}(v(1))\ldots\mathrm{Lex}(w(N_{\mathrm{na}}))\mathrm{Lex}(z(N_{\mathrm{na}}))\mathrm{Lex}(v(N_{\mathrm{na}}))$. A total of $12P$ p_4s are obtained from a polychoron and its mirror image, where P is the number of peaks. By choosing the lexicographically smallest one, we can assign one unique p_4 to a polychoron.

6.3.9 Non-1-Simple Polychora

A 1-simple polychoron has one or more peaks of degree more than three. By cutting such peaks and distinguishing cross-section polyhedra from other polyhedra, a one-to-one correspondence can be established between a non-1-simple polychoron and a 1-simple polychoron with cross-section polyhedra. By using this correspondence, the p_4-code can be generalized to deal with non-1-simple polychora (See [18] for details). However, this approach is not always efficient. So, we also use the duality of polychora. By using the duality, the 5-cell can be represented by T^4, the 8-cell by H^8, the 16-cell by $\bigstar H^8$, the 24-cell by O^{24}, the 120-cell by D^{120}, and the 600-cell by $\bigstar D^{120}$. Here, $T = 3333 = 3^4$ represents the tetrahedron, $H = 444444 = 4^6$ represents the hexahedron, $O = \bigstar H$ represents the octahedron, and $D = 555555555555 = 5^{12}$ represents the dodecahedron.

6.3.10 Ridge-Sequence Codeword

A polychoron relating to an amorphous material is 0-simple. Since they are all 0-simple, the polyhedra of a 0-simple polychoron can be represented without '$\cdot$' and '$\bigstar$'. In other words, all the numbers of sides of polygons of a 0-simple polychoron are recorded in p_4. We introduce a ridge-sequence codeword (rs) to briefly represent a 0-simple polychoron below.

We first describe tentative ridge IDs and ridge IDs. Since two face IDs are associated with every ridge, we tentatively assign the smaller face ID to the ridge. Since the tentative IDs thus assigned are not in a sequential order, we relabel the IDs so that the ridge i is the one with the ith smallest tentative ID.

In a polychoron, polyhedra are glued together face to face. For example, in the polychoron illustrated in Fig. 6.29, the first face of the polyhedron 2 (34443-polyhedron) is glued to the first face of the polyhedron 1 (3333-polyhedron). Since the face 2_1 is glued to the face 1_1, $p_2(2_1) = p_2(1_1) = 3$. Here, $p_2(i_j)$ is the value of p_2 of jth number of ps_2 of the polyhedron i. In general, when the face a_b is glued to the face x_y, $p_2(a_b) = p_2(x_y)$. The ridge is a part of a polychoron where two faces of polyhedra meet. Suppose that $a < x$ and $r_t(a_b)$ is the number of peaks of the ridge with a tentative ID a_b. Then $p_2(a_b) = p_2(x_y) = r_t(a_b)$. Since $r_t(a_b)$ is recorded twice, p_4 is redundant. This originates in that we regard a polychoron as an assemblage of polyhedra. If we regard a polychoron as an assemblage of ridges and use rs, we can reduce the redundancy in p_4 (See Ref. [19] for details). The rs-codeword is denoted as,

$$rs = r(1)r(2)r(3)\ldots r(R).$$

Here, $r(i)$ is the number of peaks of the ridge i. R is the number of ridges of a polychoron. By using rs, we can represent a polychoron whose p_4 is 3333 34443 34443 34443 34443 3333 by a briefer codeword $p_4^{(rs)} = rs = 33334443443433$ (Fig. 6.32).

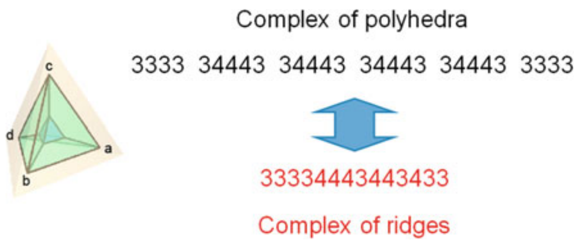


Fig. 6.32 Ridge-sequence codeword [19]

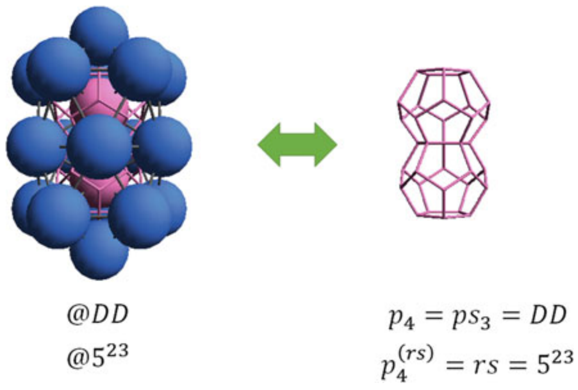


Fig. 6.33 Relation between a local atomic arrangement and an assemblage of Voronoi polyhedra. Here, $D = 5^{12}$ is p_3 of the dodecahedron

6.3.11 *Relation Between a Local Atomic Arrangement and an Assemblage of Voronoi Polyhedra*

An assemblage of polyhedra can be regarded as a partial polychoron, and is represented by p_4. For example, as is illustrated in Fig. 6.33, an assemblage of two dodecahedra is represented by $p_4 = ps_3 = DD$ ($p_4^{(rs)} = rs = 5^{23}$). The arrangement of atoms that define the polyhedral assemblage is represented by $@DD$ ($@\ 5^{23}$). The $@DD$-cluster ($@\ 5^{23}$-cluster) can be regarded as overlapping two $@D$-clusters.

6.4 Summary

We have reviewed the p_4- and p_3-codes, which we have created as a method for studying the structures of amorphous materials [11, 12, 18, 19]. The p_3-code is a method for briefly representing polyhedra. It consists of (1) an encoding algorithm for converting a way of how polygons are arranged to form a polyhedron into p_3 and (2) a decoding algorithm for recovering the original polyhedron from its p_3. We can study the short-range order of amorphous materials by classifying Voronoi polyhedra according to their p_3s. The p_4-code is a generalization of the p_3-code for representing assemblages of polyhedra. By using the p_4-code, a way of how polyhedra are arranged to form a polyhedral assemblage can be converted into p_4, from which the original polyhedral assemblage can be recovered. We can study the long-range order of amorphous materials by classifying assemblages of Voronoi polyhedra according to their p_4s.

References

1. K. Nishio, J. Kōga, T. Yamaguchi, F. Yonezawa, Theoretical study of light-emission properties of amorphous silicon quantum dots. Phys. Rev. B **67**, 195304 (2003). https://doi.org/10.1103/PhysRevB.67.195304
2. K. Nishio, A.K.A. Lu, G. Pourtois, Low-strain Si/O superlattices with tunable electronic properties: ab initio calculations. Phys. Rev. B **91**, 165303 (2015). https://doi.org/10.1103/PhysRevB.91.165303
3. J. Bernal, Bakerian lecture 1962—structure of liquids. Proc. R. Soc. Lond. A-Math. Phys. Sci. **280**, 299+ (1964). https://doi.org/10.1098/rspa.1964.0147
4. J. Finney, Random packings and structure of simple liquids. 1. Geometry of random close packing. Proc. R. Soc. Lond. A **319**, 479–493 (1970). https://doi.org/10.1098/rspa.1970.0189
5. J. Finney, Random packings and structure of simple liquids. 2. Molecular geometry. Proc. R. Soc. Lond. A **319**, 495–507 (1970). https://doi.org/10.1098/rspa.1970.0190
6. F. Yonezawa, Glass transition and relaxation of disordered structures. Solid State Phys. **45**, 179–254 (1991)
7. Y.Q. Cheng, E. Ma, Atomic-level structure and structure–property relationship in metallic glasses. Prog. Mater. Sci. **56**, 379–473 (2011). https://doi.org/10.1016/j.pmatsci.2010.12.002
8. K. Nishio, J. Kōga, T. Yamaguchi, F. Yonezawa, Confinement-induced stable amorphous solid of Lennard-Jones Argon. J. Phys. Soc. Jpn. **73**, 627–633 (2004). https://doi.org/10.1143/JPSJ.73.627
9. K. Nishio, T. Miyazaki, H. Nakamura, Universal medium-range order of amorphous metal oxides. Phys. Rev. Lett. **111**, 155502 (2013). https://doi.org/10.1103/PhysRevLett.111.155502
10. A. Hirata, L.J. Kang, T. Fujita et al., Geometric frustration of icosahedron in metallic glasses. Science **341**, 376–379 (2013). https://doi.org/10.1126/science.1232450
11. Creation of a mathematical method to express irregular atomic arrangements of amorphous materials, http://www.aist.go.jp/aist_e/list/latest_research/2017/20170201/en20170201.html. Accessed 28 Aug 2017
12. K. Nishio, T. Miyazaki, Polyhedron code—rule hidden in polyhedra (in Japanese). Bull. Soc. Sci. Form **32**, 1 (2017)
13. E.A. Lazar, J.K. Mason, R.D. MacPherson, D.J. Srolovitz, Complete topology of cells, grains, and bubbles in three-dimensional microstructures. Phys. Rev. Lett. **109**, 095505 (2012). https://doi.org/10.1103/PhysRevLett.109.095505
14. L. Weinberg, On the maximum order of the automorphism group of a planar triply connected graph. SIAM J. Appl. Math. **14**, 729–738 (1966). https://doi.org/10.1137/0114062
15. L. Weinberg, A simple and efficient algorithm for determining isomorphism of planar triply connected graphs. IEEE Trans. Circuit Theory **13**, 142–148 (1966). https://doi.org/10.1109/TCT.1966.1082573
16. P.R. Cromwell, *Polyhedra* (Cambridge University Press, 1999)
17. I. Stewart, R. Wilson, *Four Colors Suffice: How the Map Problem Was Solved*, Color edn. (Princeton University Press, Princeton, New Jersey, 2013)

18. K. Nishio, T. Miyazaki, How to describe disordered structures. Sci. Rep. **6**, 23455 (2016). https://doi.org/10.1038/srep23455
19. K. Nishio, T. Miyazaki, Describing polyhedral tilings and higher dimensional polytopes by sequence of their two-dimensional components. Sci. Rep. **7**, 40269 (2017). https://doi.org/10.1038/srep40269
20. G.M. Ziegler, *Lectures on Polytopes* (Springer, New York, 2012)
21. R.J. Wilson, *Introduction to Graph Theory* (Pearson, Harlow, England; New York, 2012)