Canonical Form of Gray Codes in N-cubes

Sylvain Contassot-Vivier, Jean-François Couchot

Canonical form of Gray Codes in N-cubes

Sylvain Contassot-Vivier[1] and Jean-François Couchot[2]

[1] Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54506, France
Sylvain.Contassotvivier@loria.fr
[2] FEMTO-ST Institute, Univ. Bourgogne Franche-Comté, France

Abstract. In previous works, the idea of walking into a N-cube where a balanced Hamiltonian cycle have been removed has been proposed as the basis of a chaotic PRNG whose chaotic behavior has been proven. However, the construction and selection of the most suited balanced Hamiltonian cycles implies practical and theoretical issues. We propose in this paper a canonical form for representing isomorphic Gray codes. It provides a drastic complexity reduction of the exploration of all the Hamiltonian cycles and we discuss some criteria for the selection of the most suited cycles for use in our chaotic PRNG.

1 Introduction

The problem of designing Pseudo-Random Number Generators (PRNG) that satisfy the probabilistic properties to produce a uniform distribution is difficult. Moreover, the knowledge of the generation algorithm and any sequence of previously generated bits should not constitute a sufficient piece of information to predict the next generated bits without knowing initial conditions. In order to build such PRNG, some studies have focused on the use of chaotic systems [7,6,2].

In a previous work [4], some of the authors have proposed a PRNG based on random walk in a N-cube where a balanced Hamiltonian cycle has been removed, and its chaotic nature has been proved. Moreover, it has been shown that the removed Hamiltonian cycle should be balanced in order to produce more efficient PRNG. Balanced Hamiltonian cycles are cycles in which the numbers of occurrences of the traversed dimensions are equal or differ at most by 2. In [8], the authors have proposed an approach that provides a subset of all the Hamiltonian cycles. This approach is however undeterministic and the cardinal number of the produced subset is dramatically small compared to to the one of all the Hamiltonian cycles. In some sense, it is a partial solution of finding Hamiltonian cycles.

The undeterministic aspect of this approach has been tackled in [3] where we have proposed a particularization of it. This new procedure succeed to find balanced Hamiltonian cycles for any dimension N and solves this issue. Nevertheless, pursuing our objective to enhance the specification of the Hamiltonian cycles most suited to the use in our PRNG, we have been confronted to the fact that procedure detailed in [3] cannot produce all non-isomorphic balanced codes: it is indeed a particularization of a partial solution.

In [10], the author proposed an approach to produce all the cycles of a graph. This work may thus solve the problem of finding a large set of balanced Hamiltonian given a dimension N. However, the approach suffers from being too exhaustive and cannot be applied as soon as the dimension of the N-cube is larger than 5. One solution could be to study cycles, whose embedding into PRNG gives distinct behaviors, *i.e.* which do not belong in the same class w.r.t an equivalence relation. For that, this work proposes a canonical form dedicated to cycles and its application to the generation of a large set of balanced Hamiltonian cycles.

This paper presents these two elements. In the following section is presented the canonical form of Hamiltonian cycles, followed in Section 3 by the description of our novel algorithm. The practical interest of the algorithm is discussed in Section 4.

2 Canonical form of Gray codes

Let $S_{\mathsf{N}} = \{1, ..., \mathsf{N}\}^{2^{\mathsf{N}}}$, the set of sequences of length 2^{N} with values in $\{1, ..., \mathsf{N}\}$. Let $H_{\mathsf{N}} \subset S_{\mathsf{N}}$, the set of sequences describing Hamiltonian cycles in a N-cube. Each of these sequences gives the succession of the dimensions followed by the path. Any Hamiltonian cycle of H_{N} can be written as $h = (h_1, ..., h_{2^{\mathsf{N}}})$. Also, we remind the reader that a Hamiltonian cycle in a N-cube is a Gray code.

We call the *canonical form* of a Hamiltonian cycle, an equivalent description of the cycle that is obtained, through a specific computation process, for all its isomorphic cycles.

Before describing our computation process of the canonical forms, we provide below an overview of the different cases of isomorphism between cycles.

2.1 Isomorphic cycles

Intuitively, Hamiltonian cycles are isomorphic to each other when the paths they describe can be topologically superposed. Indeed, a same Hamiltonian cycle can be expressed in many sequences according to some simple (global) transformations of the N-cube, leading to a set of isomorphic cycles. We list below the different transformations that can be applied to a sequence to produce isomorphic cycles.

First of all, it can be noticed that describing a cycle by the sequence of the traversed dimensions in the N-cube does not specify any starting vertex. So, a sequence does not represent only a single cycle but the 2^{N} cycles that are isomorphic up to the starting position in the N-cube.

In a similar way, applying a cyclic shift to a sequence, in any direction, is equivalent to change only its starting vertex, but this does not change the path topology. So, shifted sequences are also isomorphic cycles.

Moreover, as the N-cubes considered in the scope of this paper are not oriented, the direction of the cycle is not significant and then, an isomorphic cycle is obtained by inverting the order of a sequence.

Finally, cycles can also be isomorphic up to rotations/symmetries, which are obtained by renumbering the dimensions of the N-cube. For example, exchanging dimensions 2 and 3 in a 3-cube is similar to performing a 90 degrees rotation around dimension 1. In the following, that operation may also be referred to as the relabeling of a sequence since it only changes the dimensions labels. It is worth noticing that some dimensions relabelings are equivalent to the sequence inversion combined with a cyclic shift.

In order to define the canonical form of Hamiltonian cycle, we need to introduce some functions over H_{N}.

2.2 Preliminary tools

Let $R : H_{\mathsf{N}} \to H_{\mathsf{N}}$, the function that renumbers a Hamiltonian cycle h to a sequence $R(h)$ by mapping the successive distinct values (dimensions) of h to the ordered values from 1 to N. So, the first value h_1 of h is necessarily mapped to 1, then the first distinct value in the remaining of h (that is $(h_2, \ldots, h_{2^{\mathsf{N}}})$) is mapped to 2, and so on. As function R applies a renumbering, it follows that $\forall i, j \in \{1, \ldots, 2^{\mathsf{N}}\}, h_i = h_j \Leftrightarrow R(h)_i = R(h)_j$.

The effect of function R is to apply rotations/symmetries to a sequence, by relabeling the dimensions of the N-cube, in order to express it in a specific order of the traversed dimensions, without modifying topology of the path. So, this function is an automorphism on H_{N}.

As an example, if we have $\mathsf{N} = 3$ and the sequence $h = (2, 3, 1, 3, 2, 3, 1, 3)$, then $R(h) = (1, 2, 3, 2, 1, 2, 3, 2)$. So, the dimensions 1, 2 and 3 are respectively replaced by (relabeled) 3, 1 and 2 (as shown in Fig. 1), where the three dimensions labels and the starting vertex are fixed. It can be seen that both sequences are isomorphic up to a rotation around dimension 1 and an orientation inversion.

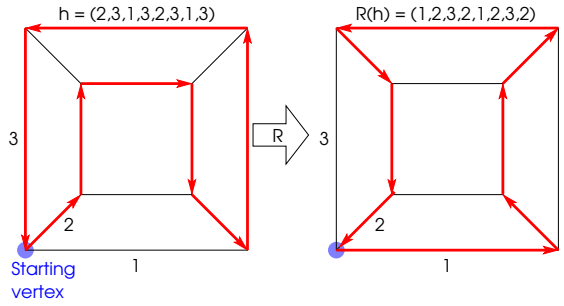

Fig. 1: Application of function R on sequence $h = (2, 3, 1, 3, 2, 3, 1, 3)$.

As the lexicographic order over sequences of length N provides a total order on H_{N}, the results of R are totally ordered. So, for any subset X of H_{N} there exists a unique minimal value of the results of R applied to any $h \in X$. This property is used in the computation of our canonical form.

Let $D : H_{\mathsf{N}} \times \{1, \ldots, 2^{\mathsf{N}}\} \to H_{\mathsf{N}}$, the function that associates to a sequence $h = (h_1, \ldots, h_{2^{\mathsf{N}}})$ and an integer k, the sequence

$D(h,k) = (h_k, h_{k+1}, \ldots, h_{2^N}, h_1, h_2, \ldots, h_{k-1})$, which is h after $k-1$ successive left cyclic shifts, so that h_k becomes the first value of the sequence.

The effect of function D is simply to change the starting point of the sequence, without modifying the cycle itself, as can be seen on Fig. 2. As well as function R, this function is also an automorphism on H_N and it is also used to compute our canonical form of isomorphic cycles.

Going back to our previous example sequence $h = (2,3,1,3,2,3,1,3)$ and choosing $k = 3$, we obtain $D(h,3) = (1,3,2,3,1,3,2,3)$.

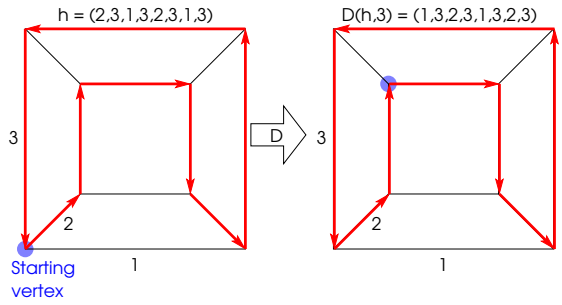

Fig. 2: Application of function D on sequence $h = (2,3,1,3,2,3,1,3)$ and $k = 3$.

Let $W : H_n \times \{1, \ldots, 2^N\} \to \{1, \ldots, 2^N\}$, the function that associates to a sequence h and an integer k, the length of the minimal sub-sequence of cycle h starting at h_k and containing all the values in $\{1, \ldots, n\}$.

Getting back to our example h and choosing $k = 4$, we have $W((2,3,1,3,2,3,1,3),4) = 4$ as the smallest sub-sequence containing $\{1,2,3\}$ starting from $h_4 = 3$ in h is $(2,3,1,\underline{3,2,3,1},3)$, that is to say $(3,2,3,1)$, whose length is 4. In the same way, we have $W((2,3,1,3,2,3,1,3),7) = 3$ as the minimal sub-sequence from $h_7 = 1$ is $(1,3,2)$. However, we can notice too that $W((2,3,1,3,2,3,1,3),8) = 4$.

In [1], Bykov uses this notion of minimal sub-sequence containing all the dimensions of the N-cube to define the *window width* of a sequence h. It corresponds to the maximal value of function W over all the possible starting points in the sequence. It provides information about the *local balance* between the traversed dimensions along the cycle. This window width can be defined by $M(h)$, for $h \in H_N$ as :

$$M(h) = \max_{k \in \{1,\ldots,2^N\}} W(h,k) \tag{1}$$

2.3 Canonical form

The function $C : H_N \to H_N$, defined by:

$$C(h) = \min \{R(D(h,k)) |\, k \text{ is s.t. } W(h,k) = M(h)\} \tag{2}$$

produces the canonical form of any sequence from H_N. Notice that this set is ordered according to the aforementioned lexical ordering, which is total.

The role of the C function is to provide a unique representative of for each class of Hamiltonian cycle. By class of Hamiltonian cycle, we mean the set of isomorphic Hamiltonian cycles according to translations (changing the starting point of the sequence) and rotations/symmetries (changing the dimensions labels). So, we have the following theorem.

Theorem 1. *For any cycles a and b in H_{N}, $C(a) = C(b)$ if and only if a and b are isomorphic cycles.*

Proof. As both functions R and D are automorphisms on H_{N}, the composite function $R \circ D$ also is an automorphism on H_{N}. Thus, for any integer $k \in \{1, \ldots, 2^{\mathsf{N}}\}$, sequence $R(D(h, k))$ is isomorphic to sequence h, and so is $C(h)$. Also, this implies that for any two non-isomorphic sequences h and g in H_{N}, there does not exist any couples of integers i and j in $\{1, \ldots, 2^{\mathsf{N}}\}$ such that $R(D(h, i)) = R(D(g, j))$. Thus, the results of $C(h)$ and $C(g)$ are necessarily different when h and g are not in the same classes of isomorphic cycles.

However, there remains the question of uniqueness of the result of C for all sequences in a same class of H_{N}. That property induces that for any two isomorphic sequences h and g in H_{N}, there exist two integers i and j in $\{1, \ldots, 2^{\mathsf{N}}\}$ such that $R(D(h, i)) = R(D(g, j))$. From the previous observations, it is obvious that $R(D(h, k))$ is isomorphic to $R(D(g, j))$, but we have to show that for some adequately chosen i and j, they are *identical* sequences.

As a first step, let us consider two sequences $h = (h_1, \ldots, h_l)$ and $g = (g_1, \ldots, g_l)$ that are isomorphic only up to rotations/symmetries. As such transformations can be expressed by dimensions relabeling, it follows that g and h are mutual relabelings of each others:

$$h_1 \leftrightarrow g_1, \quad h_2 \leftrightarrow g_2, \quad \ldots, \quad h_l \leftrightarrow g_l \tag{3}$$

and

$$\forall i, j \in \{1, \ldots, l\} \quad h_i = h_j \Leftrightarrow g_i = g_j \tag{4}$$

Moreover, $R(h)$ and $R(g)$ are also respective relabelings of h and g. The fact that $R(h) = R(g)$ is ensured by the ordered relabeling over $\{1, \ldots, n\}$. Indeed, as the relabeling follows the numerical order of integers, it produces the same sequence for h and g according to the total lexicographic order over H_{N}:

$$\begin{array}{l} h_1 \to 1, g_1 \to 1 \\ h_2 \to 2, g_2 \to 2 \end{array} \tag{5}$$

and due to (4), we have:

$$\forall i \in \{3, \ldots, l\}, k \in \{1, \ldots, n\}, \quad h_i \to k \Leftrightarrow g_i \to k \tag{6}$$

Thus, function R produces the same result for sequences that are isomorphic up to rotations/symmetries.

The next step consists in taking into account cyclic shifts between sequences. Solving this problem is similar to finding a way to re-align all isomorphic cycles

according to a common starting vertex. Fortunately, this is possible according to the notion of window width, previously introduced and expressed by functions W and M. Indeed, the window width discriminates the positions in a sequence, by identifying the ones with the highest local balance, that is to say the ones from which starts the longest minimal sub-sequence containing all values in $\{1, .., n\}$. Obviously, the window width is the same for all isomorphic cycles, as they have the same sequence of local balances up to a cyclic shift, whatever the labels of the dimensions. For any class of cycles, there is at least one position corresponding to the window width and we use it as the reference starting position to force the alignment of all cycles in the class.

When there is exactly one such position in a class, there is no ambiguity and every cycle of the class if shifted to begin at this position. However, for some classes, there might exist several positions corresponding to the window width. Thus, an additional deterministic selection must be applied to those possibilities. This is where the total lexicographic order is exploited, by selecting the position whose ordered relabeling produces the smallest sequence relatively to that order. This is what is expressed by the *min* operator in function C. As the result is a minimal value over a totally ordered space, it is unique and it ensures the common re-alignment of all the cycles in a same class.

So, function C re-aligns isomorphic cycles to a common starting position and relabels their dimensions in an ordered way that ensures a unique result for isomorphic cycles. □

Finally, it is worth noticing that it is the use of the window width notion combined to cyclic shifts, the total lexicographic order over H_{N} and the dimensions relabeling that allows us to compute a unique class representative.

So, the binary relation E induced by function C:

$$\forall a, b \in H_{\mathsf{N}}, \quad E(a, b) = \begin{cases} 1 \text{ if } C(a) = C(b) \\ 0 \text{ otherwise} \end{cases} \tag{7}$$

is an equivalence relation over H_{N} since C is a function.

2.4 Examples of application of C

Applying function C to our example sequence $h = (2, 3, 1, 3, 2, 3, 1, 3)$ yields $C(h) = (1, 2, 1, 3, 1, 2, 1, 3)$ and $M(h) = 4$. Moreover, that maximal value is obtained for the positions: 2, 4, 6 and 8. So, it is necessary to compute the sequences $R(D(h, 2))$, $R(D(h, 4))$, $R(D(h, 6))$, and $R(D(h, 8))$ in order to get the minimal one according to the lexicographic order over H_{N}, leading to results in Table 1.

Finally, as the four results are identical, the minimal sequence is this unique result, leading to $C(h) = (1, 2, 1, 3, 1, 2, 1, 3)$.

In fact, it can be checked that all the sequences in H_3 lead to that unique result of function C. This comes from the fact that there is only one Gray code up to isomorphism in H_3.

k	$D(h,k)$	$R(D(h,k))$
2	$(3,1,3,2,3,1,3,2)$	$(1,2,1,3,1,2,1,3)$
4	$(3,2,3,1,3,2,3,1)$	$(1,2,1,3,1,2,1,3)$
6	$(3,1,3,2,3,1,3,2)$	$(1,2,1,3,1,2,1,3)$
8	$(3,2,3,1,3,2,3,1)$	$(1,2,1,3,1,2,1,3)$

Table 1: Application of $R \circ D$ to $h = (2,3,1,3,2,3,1,3)$.

In order to provide a more representative example, let us apply function C to some cycles from a N-cube providing several classes of Hamiltonian cycles. If we consider $h = (1,2,1,3,4,3,2,1,2,3,4,2,4,1,4,3)$ in H_4, then $M(h) = 6$ and there are five possible starting positions: 6, 9, 11, 13 and 16. Fig. 3(a) gives the results of $R \circ D$ for those possibilities. The minimal vector is $(1,2,1,3,1,4,3,2,3,4,1,4,2,3,2,4)$, and so is $C(h)$.

Now, let us build a cycle g that is isomorphic to h by applying to h the following operations:

1- invert the sequence order $\rightarrow (3,4,1,4,2,4,3,2,1,2,3,4,3,1,2,1)$
2- apply 4 left cyclic shifts $\rightarrow (2,4,3,2,1,2,3,4,3,1,2,1,3,4,1,4)$
3- exchange dimensions 2 and 3 $\rightarrow (3,4,2,3,1,3,2,4,2,1,3,1,2,4,1,4)$

We thus have $g = (3,4,2,3,1,3,2,4,2,1,3,1,2,4,1,4)$ and $M(g) = 6$ with five corresponding starting positions: 3, 9, 12, 14 and 16. From the application of $R \circ D$ on those instances, depicted in Fig. 3(b), we deduce that $C(g) = (1,2,1,3,1,4,3,2,3,4,1,4,2,3,2,4)$ and that $C(g) = C(h)$.

As a last example, let us consider another cycle in H_4 that is not isomorphic to g and h. This is the case for $f = (3,1,4,1,2,1,4,1,3,1,4,1,2,1,4,1)$ because the numbers of occurrences of the dimensions are not equal in f, whereas they are for g and h. For the computation of $C(f)$, we have $M(f) = 8$ and four corresponding starting positions: 2, 6, 10 and 14. All four positions produce the same result by $R \circ D$, shown in Fig. 3(c), and then $C(f) = (1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,4)$, which is different from $C(g)$ and $C(h)$.

This illustrates the class separation realized by function C when there are several classes in the considered H_{N} space, as non-isomorphic cycles lead to distinct results whereas isomorphic cycles lead to the same one.

2.5 Discussion over the interest of the canonical form

This work provides an efficient way to partition the H_{N} space up to isomorphisms by computing unique representatives of the classes. Such partitions are very useful as soon as one wants to study properties of Gray codes in dimensions larger than 3, as it is possible to focus only on classes representatives. This lead to more efficient algorithms as the number of classes increases much slowly than the number of instances. Moreover, the total order over the class representatives can also be exploited to implement efficient storage and classification algorithms when exploring a given H_{N} space.

k	$D(h,k)$	$R(D(h,k))$
6	$(3,2,1,2,3,4,2,4,1,4,3,1,2,1,3,4)$	$(1,2,3,2,1,4,1,3,2,3,1,4,3,4,2,4)$
9	$(2,3,4,2,4,1,4,3,1,2,1,3,4,3,2,1)$	$(1,2,3,1,3,4,3,2,4,1,4,2,3,2,1,4)$
11	$(4,2,4,1,4,3,1,2,1,3,4,3,2,1,2,3)$	$(1,2,1,3,1,4,3,2,3,4,1,4,2,3,2,4)$
13	$(4,1,4,3,1,2,1,3,4,3,2,1,2,3,4,2)$	$(1,2,1,3,2,4,2,3,1,3,4,2,4,3,1,4)$
16	$(3,1,2,1,3,4,3,2,1,2,3,4,2,4,1,4)$	$(1,2,3,2,1,4,1,3,2,3,1,4,3,4,2,4)$

(a) $h = (1,2,1,3,4,3,2,1,2,3,4,2,4,1,4,3)$.

k	$D(g,k)$	$R(D(g,k))$
3	$(2,3,1,3,2,4,2,1,3,1,2,4,1,4,3,4)$	$(1,2,3,2,1,4,1,3,2,3,1,4,3,4,2,4)$
9	$(2,1,3,1,2,4,1,4,3,4,2,3,1,3,2,4)$	$(1,2,3,2,1,4,2,4,3,4,1,3,2,3,1,4)$
12	$(1,2,4,1,4,3,4,2,3,1,3,2,4,2,1,3)$	$(1,2,3,1,3,4,3,2,4,1,4,2,3,2,1,4)$
14	$(4,1,4,3,4,2,3,1,3,2,4,2,1,3,1,2)$	$(1,2,1,3,1,4,3,2,3,4,1,4,2,3,2,4)$
16	$(4,3,4,2,3,1,3,2,4,2,1,3,1,2,4,1)$	$(1,2,1,3,2,4,2,3,1,3,4,2,4,3,1,4)$

(b) $g = (3,4,2,3,1,3,2,4,2,1,3,1,2,4,1,4)$.

k	$D(f,k)$	$R(D(f,k))$
2	$(1,4,1,2,1,4,1,3,1,4,1,2,1,4,1,3)$	$(1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,4)$
6	$(1,4,1,3,1,4,1,2,1,4,1,3,1,4,1,2)$	$(1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,4)$
10	$(1,4,1,2,1,4,1,3,1,4,1,2,1,4,1,3)$	$(1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,4)$
14	$(1,4,1,3,1,4,1,2,1,4,1,3,1,4,1,2)$	$(1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,4)$

(c) $f = (3,1,4,1,2,1,4,1,3,1,4,1,2,1,4,1)$

Fig. 3: Application of $R \circ D$ to cycles from H_4.

3 Balanced Gray codes generation algorithm

We remind the reader that in balanced Gray codes, the dimensions of the N-cube are used a same number of times or at most with a difference of two occurrences. When all the dimensions are used exactly the same number of times, we speak of totally balanced Gray codes. This is only possible for N-cubes whose dimension is a power of 2.

In order to generate the complete subset of balanced Gray codes in a given H_{N} space, we have adapted the $(d,g)-$algorithm proposed by Wild [10] to generate all the Hamiltonian cycles. As this algorithm produces more cycles than the ones we are interested in, we had to insert an additional selection phase during the generation process in order to discard branches that would lead to imbalanced Hamiltonian cycles.

That additional selection can be placed before the other treatments (coherency, small cycles elimination,...) applied to each generation node (in the generation tree). By this way, it cuts any unproductive branch as soon as possible, thus avoiding useless computations.

That selection consists in checking that the occurrences of the dimensions already used in the partial construction of the cycle are compatible with a balanced cycle. When this is not the case, the candidate is discarded. To check this, we compute two values that are respectively, the maximal number of occurrences allowed per dimension in a balanced code ($\overline{O}$), and the maximal number of dimensions ($\overline{D}$) with that specific number of occurrences.

Those two numbers can be directly deduced from the dimension N of the N-cube:

$$\overline{O} = \left\lfloor \frac{2^{\mathsf{N}}}{\mathsf{N}} \right\rfloor + 2\left(\left\lceil \frac{2^{\mathsf{N}}}{\mathsf{N}} \right\rceil - \left\lfloor \frac{2^{\mathsf{N}}}{\mathsf{N}} \right\rfloor \right) \text{ and } \overline{D} = \frac{2^{\mathsf{N}} \bmod \mathsf{N}}{2} \tag{8}$$

The imbalance detection algorithm is given in Alg. 1.1.

Algorithm 1.1: Imbalance detection algorithm

```
Input: a partially built path p
Output: a boolean indicating True if the path is imbalanced and False otherwise

Initialize array od[] of size n with 0
nbD ← 0        // Number of dimensions with max occurrences
imb ← False    // We start with balanced path assumption

for each valid move in p do
  get the dimension d along which the move is done
  od[d] ← od[d] + 1    // move added to occurrences of d
  if od[d]> Ō then     // too much moves on dimension d
    imb ← True         // imbalance
  else
    if od[d]= Ō        // dim d reaches max occurrences
      if nbD= D̄ then   // too much dims with max occs
        imb ← True     // imbalance
      else             // new dim with max occs added
        nbD ← nbD + 1
      endif
    endif
  endif
endfor
return imb
```

The imbalance is detected as soon as the number of occurrences of one dimension exceeds $\overline{O}$ or the number of dimensions having reached $\overline{O}$ exceeds $\overline{D}$.

Two other algorithmic enhancements may be added to the process. The former is a treatment of the nodes in the generation tree that aims at speeding up the descent towards the leafs, by jumping several levels in the tree in a same iteration. The latter is quite an extension of the former as it consists in starting the generation process not at the root of the tree but several levels deeper. However, experiments show that such additions do not systematically reduce the cost of the algorithm. A deeper study is necessary to precisely determine the impact of those additions.

Finally, all the paths that are totally specified within the generation process (the leaves of the generation tree) are transformed into their canonical form. That form is added to the lexicographically ordered list of balanced Gray codes if not already present.

So, we obtain an algorithm that generates all the non-isomorphic balanced Gray codes in a given H_{N} space.

4 Application

The first series of experiments is dedicated to the validation of the canonical form previously presented. Then, the second part is dedicated to the balanced Gray code generation algorithm.

4.1 Validation of the canonical form and the generation algorithm

The first set of experiments consists in checking the completeness of the obtained generation algorithm described in Section 3. So, this algorithm is used to experimentally retrieve *all* the classes in N-cubes up to dimension five. For larger dimensions, the number of distinct cycles is too large to be exhaustively computed (777739016577752714 for H_6).

For each set H_{N}, all Hamiltonian cycles are generated by the algorithm *without* activating the balance selection. Then, canonical forms of the cycles are computed according to C in order to deduce the distinct classes in the space.

The numbers of resulting classes have been compared to the references provided in [1] and initially coming from [5]. Our algorithm has successfully found a unique class for dimensions 2, and 3. It found 9 classes for dimension 4 and 237675 classes for dimension 5. These results confirm the completeness of the generation algorithm.

4.2 Application of the balanced Gray code generation algorithm

In theory, the presented algorithm can generate all the balanced cycles for a given dimension of N-cube. However, this is not pertinent in practice due to the exponential increase of the number of cycles. In such case, any algorithm would be confronted to two limitations: memory and execution time. For example, our algorithm can generate all the balanced Gray codes for dimensions up to 5 in a few seconds whereas it would take non reasonable time to generate all the cycles for dimensions 6 and above.

Indeed, in our application context of PRNGs, we need only to generate some particular balanced cycles, according to the regarded properties. It is then possible to restrict the search to some particular cycles. So, it should be possible to obtain a fast algorithm for generating specific balanced Gray codes.

Moreover, compared to other methods to generate balanced Gray codes, like the extended Robinson-Cohn (further denoted as e-RB) algorithm [9] or the Bykov's one [1], our approach presents the advantage of being more complete, and thus more flexible. It is able to find any balanced cycle that has some specific properties, namely which is locally balanced and whose mixing time (time until the Markov chain is ε close to the uniform distribution) is reduced.

As a first example, if we consider dimension 5, the e-RB method can only generate 2 balanced cycles (modulo cycle isomorphism), given in Table 2. The cycles are given in canonical form and the numbers in the left column correspond to their positions in the totally ordered set of all balanced cycles for dimension 5 (26155). Both cycles have a local balance of 12 and a mixing time of 31 where ε is

10^{-6}. However, for this dimension, the minimal local balance is 7 (only one cycle) and the best mixing time is 29 (several cycles with different local balances). All those cycles are listed in Table 3, together with their local balance and mixing time. So, it is clear that our method is better suited to find cycles of interest for the construction of PRNGs.

Num	sequence of traversed dimensions of the N-cube	local balance	mixing time
19708	1 2 3 1 4 1 3 2 3 4 1 5 4 5 3 5 1 2 3 2 4 2 1 4 3 5 4 5 1 2 1 5	12	31
20904	1 2 3 2 1 2 3 4 3 2 1 5 1 4 1 2 4 5 3 5 4 2 4 5 1 5 2 3 4 3 2 5	12	31

Table 2: The 2 balanced cycles generated by e-RB method in dimension 5 and their corresponding mixing time when ε is 10^{-6}.

Num	sequence of traversed dimensions of the N-cube	local balance	mixing time
22534	1 2 3 2 1 4 5 4 1 3 2 3 1 5 4 5 1 2 3 2 1 4 5 4 1 3 2 3 1 5 4 5	7	34
962	1 2 1 3 1 2 4 1 4 5 2 4 5 3 2 1 5 4 3 2 3 1 4 1 5 3 5 2 1 3 4 5	10	29
983	1 2 1 3 1 2 4 1 4 5 4 3 2 1 5 3 5 2 4 2 3 1 3 4 5 3 2 1 5 4 1 5	10	29
8962	1 2 1 3 2 4 2 1 4 5 3 2 4 1 5 1 3 4 5 4 1 2 1 3 5 1 5 4 3 2 3 5	10	29
22624	1 2 3 2 4 1 2 3 4 5 3 5 2 5 1 4 1 2 3 2 5 1 2 3 5 4 3 4 2 4 1 5	10	29
24059	1 2 3 2 4 3 2 1 4 5 3 5 2 5 1 4 1 2 3 2 5 3 2 1 5 4 3 4 2 4 1 5	10	29
11087	1 2 1 3 4 1 2 3 2 1 5 4 3 1 5 2 5 4 1 2 1 3 4 2 4 1 5 3 4 5 3 5	11	29
18407	1 2 3 1 3 4 2 3 2 1 5 3 5 2 5 4 2 1 3 1 2 4 5 4 1 5 4 2 3 2 4 5	11	29
772	1 2 1 3 1 2 1 4 3 2 3 5 2 4 2 3 4 1 5 3 5 4 3 2 3 4 5 3 5 1 4 5	12	29
6759	1 2 1 3 2 1 4 1 3 2 3 5 4 3 4 1 5 2 5 3 2 5 4 5 3 2 3 1 4 3 4 5	12	29
14967	1 2 1 3 4 3 2 1 3 1 4 5 2 1 5 2 5 4 2 5 3 5 4 1 4 3 2 3 1 4 1 5	12	29
16317	1 2 3 1 2 1 4 3 1 3 2 5 1 5 3 5 4 2 4 3 4 2 1 3 5 4 5 2 3 4 3 5	12	29
17396	1 2 3 1 3 4 1 2 1 3 4 5 2 3 4 2 3 2 5 1 5 3 4 3 2 5 4 5 3 4 1 5	12	29

Table 3: Excerpt of the 26155 non isomorphic Hamiltonian cycles generated by our method with either the smallest local balance or the smallest mixing time with $\varepsilon = 10^{-6}$ for dimension $\mathsf{N} = 5$.

A second example is related to the Bykov's construction of locally balanced cycles. The proposed algorithm builds a family of Hamiltonian cycles in a N-cube with a specific local balance of at most $n + 3.log_2(n)$. However, Table 3 shows us two facts. The former is that only two cycles among the 7403 with this particular local balance (11 for dimension 5) obtain the minimal mixing time. The latter is that this minimum is reached also by some cycles with other local balances (10 and 12). Thus, a more exhaustive algorithm, like the one we propose, is useful to get all the cycles better suited to the inclusion in a PRNG and to provide a wider choice.

5 Conclusion

A canonical form has been proposed to provide unique representations of Hamiltonian cycles in N-cubes. All the properties of an equivalence relation over the

set H_{N} have been proved. Based on this form and the Wild's algorithm that generates cycles in graphs, a new algorithm has been designed to generate all the balanced Hamiltonian cycles in any N-cube. Restrictions to specific cycles can be used to limit the generation and to avoid the combinatorial explosion on the number of cycles for dimensions greater than 6.

In the application context of PRNG construction, we have shown that our algorithm is better suited than other existing methods that generate only specific cycles, like the extended Robinson-Cohn and the Bykov ones.

Hence, our algorithm provides a useful tool to study the cycles properties that are relevant to the inclusion in a PRNG. This study is planned as our next work, together with performance optimization of our generation algorithm.

Acknowledgments

This work is partially funded by the Labex ACTION program (contract ANR-11-LABX-01-01).

References

1. Bykov, I.S.: On locally balanced gray codes. Journal of Applied and Industrial Mathematics 10(1), 78–85 (2016), http://dx.doi.org/10.1134/S1990478916010099
2. Cao, L., Min, L., Zang, H.: A chaos-based pseudorandom number generator and performance analysis. In: Computational Intelligence and Security, 2009. CIS '09. International Conference on. vol. 1, pp. 494–498. IEEE (Dec 2009)
3. Couchot, J.F., Contassot-Vivier, S., Héam, P.C., Guyeux, C.: Random walk in a n-cube without hamiltonian cycle to chaotic pseudorandom number generation: Theoretical and practical considerations. International Journal of Bifurcation and Chaos (2016), accepted on Oct 2016
4. Couchot, J., Héam, P., Guyeux, C., Wang, Q., Bahi, J.M.: Pseudorandom number generators with balanced gray codes. In: SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014. pp. 469–475 (2014)
5. Sloane, N.: On-line encyclopedia of integer sequences, http://oeis.org
6. Stojanovski, T., Kocarev, L.: Chaos-based random number generators-part i: analysis [cryptography]. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on 48(3), 281–288 (Mar 2001)
7. Stojanovski, T., Pihl, J., Kocarev, L.: Chaos-based random number generators. part ii: practical realization. Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on 48(3), 382–385 (Mar 2001)
8. Suparta, I., Zanten, A.v.: Totally balanced and exponentially balanced gray codes. Discrete Analysis and Operation Research (Russia) 11(4), 81–98 (2004)
9. Suparta, I., Zanten, A.v.: A construction of gray codes inducing complete graphs. Discrete Mathematics 308(18), 4124–4132 (2008)
10. Wild, M.: Generating all cycles, chordless cycles, and hamiltonian cycles with the principle of exclusion. Journal of Discrete Algorithms 6(1), 93 – 102 (2008), http://www.sciencedirect.com/science/article/pii/S1570866707000020, selected papers from {AWOCA} 2005Sixteenth Australasian Workshop on Combinatorial Algorithms