Random Walk in a N-cube Without Hamiltonian Cycle to Chaotic Pseudorandom Number Generation: Theoretical and Practical Considerations

Sylvain Contassot-Vivier
LORIA, Université de Lorraine, Nancy, France
sylvain.contassotvivier@loria.fr

Jean-François Couchot
FEMTO-ST Institute, CNRS, Univ. Bourgogne Franche-Comté (UBFC), France
jean-francois.couchot@univ-fcomte.fr

Christophe Guyeux
FEMTO-ST Institute, CNRS, Univ. Bourgogne Franche-Comté (UBFC), France
christophe.guyeux@univ-fcomte.fr

Pierre-Cyrille Heam
FEMTO-ST Institute, CNRS, Univ. Bourgogne Franche-Comté (UBFC), France
pierre-cyrille.heam@univ-fcomte.fr

Designing a pseudorandom number generator (PRNG) is a difficult and complex task. Many recent works have considered chaotic functions as the basis of built PRNGs: the quality of the output would indeed be an obvious consequence of some chaos properties. However, there is no direct reasoning that goes from chaotic functions to uniform distribution of the output. Moreover, embedding such kind of functions into a PRNG does not necessarily allow to get a chaotic output, which could be required for simulating some chaotic behaviors.
In a previous work, some of the authors have proposed the idea of walking into a N-cube where a balanced Hamiltonian cycle has been removed as the basis of a chaotic PRNG. In this article, all the difficult issues observed in the previous work have been tackled. The chaotic behavior of the whole PRNG is proven. The construction of the balanced Hamiltonian cycle is theoretically and practically solved. An upper bound of the expected length of the walk to obtain a uniform distribution is calculated. Finally practical experiments show that the generators successfully pass the classical statistical tests.

Keywords: Pseudorandom Numbers Generator, Chaotic iterations, Random Walk

1. Introduction

The exploitation of chaotic systems to generate pseudorandom sequences is a very topical issue [Stojanovski *et al.*, 2001; Stojanovski & Kocarev, 2001; Cao *et al.*, 2009]. Such systems are fundamentally chosen because of their unpredictable character and their sensitiveness to initial conditions. In most cases, these generators simply consist in iterating a chaotic function like the logistic map [Stojanovski *et al.*, 2001; Stojanovski & Kocarev, 2001] or the Arnold's one [Cao *et al.*, 2009]... Optimal parameters of such functions remain to be found so that attractors are avoided,*e.g.*. By following this procedure, generated numbers will hopefully follow a uniform distribution. In order to check the quality of the produced outputs, PRNGs (Pseudo-Random

Number Generators) are usually tested with statistical batteries like the so-called DieHARD [Marsaglia, 1996], NIST [Barker & Roginsky, 2010], or TestU01 [L'Ecuyer & Simard, 2007] ones.

In its general understanding, the notion of chaos is often reduced to the strong sensitiveness to the initial conditions (the well known "butterfly effect"): a continuous function k defined on a metrical space is said to be *strongly sensitive to the initial conditions* if for each point x and each positive value ϵ, it is possible to find another point y as close as possible to x, and an integer t such that the distance between the t-th iterates of x and y, denoted by $k^t(x)$ and $k^t(y)$, is larger than ϵ. However, in his definition of chaos, Devaney [Devaney, 1989] imposes to the chaotic function two other properties called *transitivity* and *regularity*. The functions mentioned above have been studied according to these properties, and they have been proven as chaotic on $\mathbb{R}$. But nothing guarantees that such properties are preserved when iterating the functions on floating point numbers, which is the domain of interpretation of real numbers $\mathbb{R}$ on machines.

To avoid this lack of chaos, we have previously presented some PRNGs that iterate continuous functions G_f on a discrete domain $\{1,\ldots,n\}^{\mathbb{N}} \times \{0,1\}^n$, where f is a Boolean function (*i.e.*, $f : \{0,1\}^{\mathsf{N}} \to \{0,1\}^{\mathsf{N}}$). These generators are $CIPRNG_f^1(u)$ [Guyeux *et al.*, 2010; Bahi *et al.*, 2011a], $CIPRNG_f^2(u,v)$ [Wang *et al.*, 2010a], and $\chi_{14Secrypt}$ [Couchot *et al.*, 2014] where *CI* stands for *Chaotic Iterations*. We have firstly proven in [Bahi *et al.*, 2011a] that, to establish the chaotic nature of $CIPRNG_f^1$ algorithm, it is necessary and sufficient that the asynchronous iterations are strongly connected. We then have proven that it is necessary and sufficient that the Markov matrix associated to this graph is doubly stochastic, in order to have a uniform distribution of the outputs. We have finally established sufficient conditions to guarantee the first property of connectivity. Among the generated functions, we thus have considered for further investigations only the ones that satisfy the second property as well.

However, it cannot be directly deduced that $\chi_{14Secrypt}$ is chaotic since we do not output all the successive values of iterating G_f. This algorithm only displays a subsequence $x^{b.n}$ of a whole chaotic sequence x^n and it is indeed incorrect to say that the chaos property is preserved for any subsequence of a chaotic sequence. This article presents conditions to preserve this property.

Finding a Boolean function which provides a strongly connected iteration graph having a doubly stochastic Markov matrix is however not an easy task. We have firstly proposed in [Bahi *et al.*, 2011a] a generate-and-test based approach that solved this issue. However, this one was not efficient enough. Thus, a second scheme has been further presented in [Couchot *et al.*, 2014] by remarking that a N-cube where an Hamiltonian cycle (or equivalently a Gray code) has been removed is strongly connected and has a doubly stochastic Markov matrix.

However, the removed Hamiltonian cycle has a great influence in the quality of the output. For instance, if this one is not balanced (*i.e.*, the number of changes in different bits are completely different), some bits would be hard to switch. This article shows an effective algorithm that efficiently implements the previous scheme and thus provides functions issued from removing, in the N-cube, a *balanced* Hamiltonian cycle.

The length b of the walk to reach a distribution close to the uniform one would be dramatically long. This article theoretically and practically studies the length b until the corresponding Markov chain is close to the uniform distribution. Finally, the ability of the approach to face classical tests suite is evaluated.

This article, which is an extension of [Couchot *et al.*, 2014], is organized as follows. The next section is devoted to preliminaries, basic notations, and terminologies regarding Boolean map iterations. Then, in Section 3, Devaney's definition of chaos is recalled while the proof of chaos of our most general PRNGs is provided. This is the first major contribution. Section 4 recalls a general scheme to obtain functions with an expected behavior. Main theorems are recalled to make the article self-sufficient. The next section (Sect. 5) presents an algorithm that implements this scheme and proves that it always produces a solution. This is the second major contribution. Then, Section 6 defines the theoretical framework to study the mixing-time, *i.e.*, the sufficient amont of time until reaching an uniform distribution. It proves that this one is in the worst case quadratic in the number of elements. Experiments show that the bound is in practice significantly lower. This is the third major contribution. Section 7 gives practical results on evaluating the PRNG against the NIST suite. This research work ends with a conclusion section, where the contribution is summarized and intended future work is outlined.

2. Preliminaries

In what follows, we consider the Boolean algebra on the set $\mathbb{B} = \{0, 1\}$ with the classical operators of conjunction '.', of disjunction '+', of negation '-', and of disjunctive union $\oplus$.

Let us first introduce basic notations. Let N be a positive integer. The set $\{1, 2, \ldots, \mathsf{N}\}$ of integers belonging between 1 and N is further denoted as $[\![1, \mathsf{N}]\!]$. A *Boolean map* f is a function from $\mathbb{B}^{\mathsf{N}}$ to itself such that $x = (x_1, \ldots, x_{\mathsf{N}})$ maps to $f(x) = (f_1(x), \ldots, f_{\mathsf{N}}(x))$. In what follows, for any finite set X, $|X|$ denotes its cardinality and $\lfloor y \rfloor$ is the largest integer lower than y.

Functions are iterated as follows. At the t^{th} iteration, only the s_t-th component is said to be "iterated", where $s = (s_t)_{t\in\mathbb{N}}$ is a sequence of indices taken in $[\![1; \mathsf{N}]\!]$ called "strategy". Formally, let $F_f : \mathbb{B}^{\mathsf{N}} \times [\![1; \mathsf{N}]\!]$ to $\mathbb{B}^{\mathsf{N}}$ be defined by

$$F_f(x, i) = (x_1, \ldots, x_{i-1}, f_i(x), x_{i+1}, \ldots, x_{\mathsf{N}}).$$

Then, let $x^0 \in \mathbb{B}^{\mathsf{N}}$ be an initial configuration and $s \in [\![1; \mathsf{N}]\!]^{\mathbb{N}}$ be a strategy, the dynamics are described by the recurrence

$$x^{t+1} = F_f(x^t, s_t). \tag{1}$$

Let be given a Boolean map f. Its associated *iteration graph* $\Gamma(f)$ is the directed graph such that the set of vertices is $\mathbb{B}^{\mathsf{N}}$, and for all $x \in \mathbb{B}^{\mathsf{N}}$ and $i \in [\![1; \mathsf{N}]\!]$, the graph $\Gamma(f)$ contains an arc from x to $F_f(x, i)$. Each arc $(x, F_f(x, i))$ is labelled with i.

Running Example. Let us consider for instance $\mathsf{N} = 3$. Let $f^* : \mathbb{B}^3 \to \mathbb{B}^3$ be defined by $f^*(x_1, x_2, x_3) = (x_2 \oplus x_3, \overline{x_1}\overline{x_3} + x_1\overline{x_2}, \overline{x_1}\overline{x_3} + x_1x_2)$. The iteration graph $\Gamma(f^*)$ of this function is given in Figure 1. ■

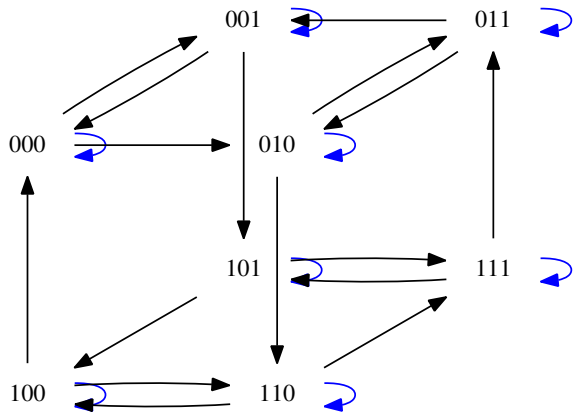

Figure 1. Iteration Graph $\Gamma(f^*)$ of the function f^*

Let us finally recall the pseudorandom number generator $\chi_{14 Secrypt}$ [Couchot *et al.*, 2014] formalized in Algorithm 1. It is based on random walks in $\Gamma(f)$. More precisely, let be given a Boolean map $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$, an input PRNG *Random*, an integer b that corresponds to a number of iterations, and an initial configuration x^0. Starting from x^0, the algorithm repeats b times a random choice of which edge to follow and traverses this edge. The final configuration is thus outputted.

Based on this setup, we can study the chaos properties of these functions. This is the aim of the next section.

Input: a function f, an iteration number b, an initial configuration x^0 (N bits)
Output: a configuration x (N bits)
$x \leftarrow x^0$;
for $i = 0, \ldots, b-1$ **do**
  $s \leftarrow Random(\mathsf{N})$;
  $x \leftarrow F_f(x, s)$;
end
return x;

Algorithm 1: Pseudo Code of the $\chi_{14Secrypt}$ PRNG

3. Proof of Chaos

3.1. *Motivations*

Let us us first recall the chaos theoretical context presented in [Bahi *et al.*, 2011a]. In this article, the space of interest is $\mathbb{B}^{\mathsf{N}} \times [\![1; \mathsf{N}]\!]^{\mathbb{N}}$ and the iteration function $\mathcal{H}_f$ is the map from $\mathbb{B}^{\mathsf{N}} \times [\![1; \mathsf{N}]\!]^{\mathbb{N}}$ to itself defined by

$$\mathcal{H}_f(x, s) = (F_f(x, s_0), \sigma(s)).$$

In this definition, $\sigma : [\![1; \mathsf{N}]\!]^{\mathbb{N}} \longrightarrow [\![1; \mathsf{N}]\!]^{\mathbb{N}}$ is a shift operation on sequences (*i.e.*, a function that removes the first element of the sequence) formally defined with

$$\sigma((u^k)_{k\in\mathbb{N}}) = (u^{k+1})_{k\in\mathbb{N}}.$$

We have proven [Bahi *et al.*, 2011a, Theorem 1] that $\mathcal{H}_f$ is chaotic in $\mathbb{B}^{\mathsf{N}} \times [\![1; \mathsf{N}]\!]^{\mathbb{N}}$ if and only if $\Gamma(f)$ is strongly connected. However, the corollary which would say that $\chi_{14Secrypt}$ is chaotic cannot be directly deduced since we do not output all the successive values of iterating F_f. Only a few of them are concerned and any subsequence of a chaotic sequence is not necessarily a chaotic sequence as well. This necessitates a rigorous proof, which is the aim of this section. Let us firstly recall the theoretical framework in which this research takes place.

3.2. *Devaney's Chaotic Dynamical Systems*

Consider a topological space $(\mathcal{X}, \tau)$ and a continuous function $f : \mathcal{X} \to \mathcal{X}$ [Devaney, 1989].

Definition 3.1. The function f is said to be *topologically transitive* if, for any pair of open sets $U, V \subset \mathcal{X}$, there exists $k > 0$ such that $f^k(U) \cap V \neq \varnothing$.

Definition 3.2. An element x is a *periodic point* for f of period $n \in \mathbb{N}^*$ if $f^n(x) = x$.

Definition 3.3. f is said to be *regular* on $(\mathcal{X}, \tau)$ if the set of periodic points for f is dense in $\mathcal{X}$: for any point x in $\mathcal{X}$, any neighborhood of x contains at least one periodic point (without necessarily the same period).

Definition 3.4 [Devaney's formulation of chaos [Devaney, 1989]]. *The function f is said to be* chaotic *on $(\mathcal{X}, \tau)$ if f is regular and topologically transitive.*

The chaos property is strongly linked to the notion of "sensitivity", defined on a metric space $(\mathcal{X}, d)$ by:

Definition 3.5. The function f has *sensitive dependence on initial conditions* if there exists $\delta > 0$ such that, for any $x \in \mathcal{X}$ and any neighborhood V of x, there exist $y \in V$ and $n > 0$ such that $d\left(f^n(x), f^n(y)\right) > \delta$.

The constant δ is called the *constant of sensitivity* of f.

Indeed, Banks *et al.* have proven in [Banks *et al.*, 1992] that when f is chaotic and $(\mathcal{X}, d)$ is a metric space, then f has the property of sensitive dependence on initial conditions (this property was formerly an element of the definition of chaos).

3.3. *A Metric Space for PRNG Iterations*

Let us first introduce $\mathcal{P} \subset \mathbb{N}$ a finite nonempty set having the cardinality $\mathsf{p} \in \mathbb{N}^*$. Intuitively, this is the set of authorized numbers of iterations. Denote by $p_1, p_2, \ldots, p_{\mathsf{p}}$ the ordered elements of $\mathcal{P}$: $\mathcal{P} = \{p_1, p_2, \ldots, p_{\mathsf{p}}\}$ and $p_1 < p_2 < \ldots < p_{\mathsf{p}}$.

In our Algorithm 1, p is 1 and p_1 is b. But this algorithm can be seen as b functional compositions of F_f. Obviously, it can be generalized with p_i, $p_i \in \mathcal{P}$, functional compositions of F_f. Thus, for any $p_i \in \mathcal{P}$ we introduce the function $F_{f,p_i} : \mathbb{B}^{\mathsf{N}} \times [\![1, \mathsf{N}]\!]^{p_i} \to \mathbb{B}^{\mathsf{N}}$ defined by

$$\begin{array}{l} F_{f,p_i}(x, (u^0, u^1, \ldots, u^{p_i-1})) \mapsto \\ \quad F_f(\ldots(F_f(F_f(x, u^0), u^1), \ldots), u^{p_i-1}). \end{array}$$

The considered space is $\mathcal{X}_{\mathsf{N},\mathcal{P}} = \mathbb{B}^{\mathsf{N}} \times \mathbb{S}_{\mathsf{N},\mathcal{P}}$, where $\mathbb{S}_{\mathsf{N},\mathcal{P}} = [\![1, \mathsf{N}]\!]^{\mathbb{N}} \times \mathcal{P}^{\mathbb{N}}$. Each element in this space is a pair where the first element is N-uple in $\mathbb{B}^{\mathsf{N}}$, as in the previous space. The second element is a pair $((u^k)_{k\in\mathbb{N}}, (v^k)_{k\in\mathbb{N}})$ of infinite sequences. The sequence $(v^k)_{k\in\mathbb{N}}$ defines how many iterations are executed at time k before the next output, while $(u^k)_{k\in\mathbb{N}}$ details which elements are modified.

Let us introduce the shift function Σ for any element of $\mathbb{S}_{\mathsf{N},\mathcal{P}}$.

$$\begin{array}{rccc} \Sigma : & \mathbb{S}_{\mathsf{N},\mathcal{P}} & \to & \mathbb{S}_{\mathsf{N},\mathcal{P}} \\ & \left((u^k)_{k\in\mathbb{N}}, (v^k)_{k\in\mathbb{N}}\right) & \mapsto & \begin{array}{l} \left(\sigma^{v^0}\left((u^k)_{k\in\mathbb{N}}\right),\right. \\ \qquad \left.\sigma\left((v^k)_{k\in\mathbb{N}}\right)\right). \end{array} \end{array}$$

In other words, Σ receives two sequences u and v, and it operates v^0 shifts on the first sequence and a single shift on the second one. Let us consider

$$\begin{array}{rccc} G_f : & \mathcal{X}_{\mathsf{N},\mathcal{P}} & \to & \mathcal{X}_{\mathsf{N},\mathcal{P}} \\ & (e, (u, v)) & \mapsto & \left(F_{f,v^0}\left(e, (u^0, \ldots, u^{v^0-1}\right), \Sigma(u, v)\right). \end{array} \tag{2}$$

Then the outputs $(y^0, y^1, \ldots)$ produced by the *CIPRNG*$_f^2(u, v)$ generator [Wang *et al.*, 2010b] are by definition the first components of the iterations $X^0 = (x^0, (u, v))$ and $\forall n \in \mathbb{N}, X^{n+1} = G_f(X^n)$ on $\mathcal{X}_{\mathsf{N},\mathcal{P}}$. The new obtained generator can be shown as either a post-treatment over generators u and v, or a discrete dynamical system on a set constituted by binary vectors and couple of integer sequences.

3.4. *A metric on* $\mathcal{X}_{\mathsf{N},\mathcal{P}}$

We define a distance d on $\mathcal{X}_{\mathsf{N},\mathcal{P}}$ as follows. Consider $x = (e, s)$ and $\check{x} = (\check{e}, \check{s})$ in $\mathcal{X}_{\mathsf{N},\mathcal{P}} = \mathbb{B}^{\mathsf{N}} \times \mathbb{S}_{\mathsf{N},\mathcal{P}}$, where $s = (u, v)$ and $\check{s} = (\check{u}, \check{v})$ are in $\mathbb{S}_{\mathsf{N},\mathcal{P}} = \mathcal{S}_{[\![1,\mathsf{N}]\!]} \times \mathcal{S}_{\mathcal{P}}$.

- e and $\check{e}$ are integers belonging in $[\![0, 2^{\mathsf{N}-1}]\!]$. The Hamming distance on their binary decomposition, that is, the number of dissimilar binary digits, constitutes the integral part of $d(X, \check{X})$.
- The fractional part is constituted by the differences between v^0 and $\check{v}^0$, followed by the differences between finite sequences $u^0, u^1, \ldots, u^{v^0-1}$ and $\check{u}^0, \check{u}^1, \ldots, \check{u}^{\check{v}^0-1}$, followed by differences between v^1 and $\check{v}^1$, followed by the differences between $u^{v^0}, u^{v^0+1}, \ldots, u^{v^1-1}$ and $\check{u}^{\check{v}^0}, \check{u}^{\check{v}^0+1}, \ldots, \check{u}^{\check{v}^1-1}$, etc. More precisely, let $p = \lfloor \log_{10}(\max \mathcal{P}) \rfloor + 1$ and $n = \lfloor \log_{10}(\mathsf{N}) \rfloor + 1$.

– The p first digits of $d(x, \check{x})$ are $|v^0 - \check{v}^0|$ written in decimal numeration (and with p digits: zeros are added on the left if needed).

– The next $n \times \max(\mathcal{P})$ digits aim at measuring how much $u^0, u^1, \ldots, u^{v^0-1}$ differ from $\check{u}^0, \check{u}^1, \ldots, \check{u}^{\check{v}^0-1}$. The n first digits are $|u^0 - \check{u}^0|$. They are followed by $|u^1 - \check{u}^1|$ written with n digits, etc.

∗ If $v^0 = \check{v}^0$, then the process is continued until $|u^{v^0-1} - \check{u}^{\check{v}^0-1}|$ and the fractional part of $d(X, \check{X})$ is completed by 0's until reaching $p + n \times \max(\mathcal{P})$ digits.

∗ If $v^0 < \check{v}^0$, then the $\max(\mathcal{P})$ blocs of n digits are $|u^0 - \check{u}^0|$, ..., $|u^{v^0-1} - \check{u}^{v^0-1}|$, $\check{u}^{v^0}$ (on n digits), ..., $\check{u}^{\check{v}^0-1}$ (on n digits), followed by 0's if required.

∗ The case $v^0 > \check{v}^0$ is dealt similarly.

– The next p digits are $|v^1 - \check{v}^1|$, etc.

This distance has been defined to capture all aspects of divergences between two sequences generated by the *CIPRNG*$_f^2$ method, when setting respectively (u, v) and $(\check{u}, \check{v})$ as inputted couples of generators. The integral part measures the bitwise Hamming distance between the two N-length binary vectors chosen as

seeds. The fractional part must decrease when the number of identical iterations applied by the $CIPRNG_f^2$ discrete dynamical system on these seeds, in both cases (that is, when inputting either (u, v) or $(\check{u}, \check{v})$), increases. More precisely, the fractional part will alternately measure the following elements:

- Do we iterate the same number of times between the next two outputs, when considering either (u, v) or $(\check{u}, \check{v})$?
- Then, do we iterate the same components between the next two outputs of $CIPRNG_f^2$?
- etc.

Finally, zeros are put to be able to recover what occurred at a given iteration. Such aims are illustrated in the two following examples. **Running Example.** Consider for instance that $\mathsf{N} = 13$, $\mathcal{P} = \{1, 2, 11\}$ (so $\mathsf{p} = 3$, $p = \lfloor \log_{10}(\max \mathcal{P}) \rfloor + 1 = 2$, while $n = 2$), and that $s = \begin{cases} u = \underline{6,} \ \underline{11,} 5, ... \\ v = \overline{1, 2, ...} \end{cases}$ while $\check{s} = \begin{cases} \check{u} = \underline{6,} 4 \ \underline{1,} ... \\ \check{v} = \overline{2, 1, ...} \end{cases}$.

So

$$d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = 0.01\ 0004000000000000000\ 01\ 1005...$$

Indeed, the $p = 2$ first digits are 01, as $|v^0 - \check{v}^0| = 1$, and we use p digits to code this difference ($\mathcal{P}$ being $\{1, 2, 11\}$, this difference can be equal to 10). We then take the $v^0 = 1$ first terms of u, each term being coded in $n = 2$ digits, that is, 06. As we can iterate at most $\max(\mathcal{P})$ times, we must complete this value by some 0's in such a way that the obtained result has $n \times \max(\mathcal{P}) = 22$ digits, that is: 0600000000000000000000. Similarly, the first $\check{v}^0 = 2$ terms in $\check{u}$ are represented by 0604000000000000000000, and the value of their digit per digit absolute difference is equal to 0004000000000000000000. These digits are concatenated to 01, and we start again with the remainder of the sequences. ■

Running Example. Consider now that $\mathsf{N} = 9$ ($n = 1$), $\mathcal{P} = \{2, 7\}$ ($\mathsf{p} = 2, p = 1$), and that $s = \begin{cases} u = \underline{6, 7,} \ \underline{4, 2,}... \\ v = \overline{2, 2, ...} \end{cases}$ while $\check{s} = \begin{cases} \check{u} = \underline{4, 9, 6, 3, 6, 6, 7,} \ \underline{9, 8,} ... \\ \check{v} = \overline{7, 2, ...} \end{cases}$

So: $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = 0.5\ 2263667\ 1\ 5600000....$ ■

d can be more rigorously written as follows:

$$d(x, \check{x}) = d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) + d_{\mathbb{B}^{\mathsf{N}}}(e, \check{e}),$$

where:

- $d_{\mathbb{B}^{\mathsf{N}}}$ is the Hamming distance,
- $\forall s = (u, v), \check{s} = (\check{u}, \check{v}) \in \mathcal{S}_{\mathsf{N},\mathcal{P}}$,

$$d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = \sum_{k=0}^{\infty} \frac{1}{10^{(k+1)p + kn \max(\mathcal{P})}} \left(|v^k - \check{v}^k| + \left| \sum_{l=0}^{v^k - 1} \frac{u^{\sum_{m=0}^{k-1} v^m + l}}{10^{(l+1)n}} - \sum_{l=0}^{\check{v}^k - 1} \frac{\check{u}^{\sum_{m=0}^{k-1} \check{v}^m + l}}{10^{(l+1)n}} \right| \right)$$

Let us show that,

Proposition 1. *d is a distance on $\mathcal{X}_{\mathsf{N},\mathcal{P}}$.*

Proof. $d_{\mathbb{B}^{\mathsf{N}}}$ is the Hamming distance. We will prove that $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}$ is a distance too, thus d will also be a distance, being the sum of two distances.

- Obviously, $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) \geqslant 0$, and if $s = \check{s}$, then $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = 0$. Conversely, if $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = 0$, then $\forall k \in \mathbb{N}, v^k = \check{v}^k$ due to the definition of d. Then, as digits between positions $p + 1$ and $p + n$ are null and correspond to $|u^0 - \check{u}^0|$, we can conclude that $u^0 = \check{u}^0$. An extension of this result to the whole first $n \times \max(\mathcal{P})$ blocs leads to $u^i = \check{u}^i$, $\forall i \leqslant v^0 = \check{v}^0$, and by checking all the $n \times \max(\mathcal{P})$ blocs, $u = \check{u}$.

- $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}$ is clearly symmetric ($d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s,\check{s}) = d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(\check{s},s)$).
- The triangle inequality is obtained because the absolute value satisfies it as well.

■

Before being able to study the topological behavior of the general chaotic iterations, we must first establish that:

Proposition 2. *For all $f : \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$, the function G_f is continuous on $(\mathcal{X}, d)$.*

Proof. We will show this result by using the sequential continuity. Consider a sequence $x^n = (e^n, (u^n, v^n)) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}^{\mathbb{N}}$ such that $d(x^n, x) \longrightarrow 0$, for some $x = (e, (u, v)) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}$. We will show that $d\left(G_f(x^n), G_f(x)\right) \longrightarrow 0$. Remark that u and v are sequences of sequences.

As $d(x^n, x) \longrightarrow 0$, there exists $n_0 \in \mathbb{N}$ such that $d(x^n, x) < 10^{-(p+n\max(\mathcal{P}))}$ (its $p + n\max(\mathcal{P})$ first digits are null). In particular, $\forall n \geqslant n_0, e^n = e$, as the Hamming distance between the integral parts of x and $\check{x}$ is 0. Similarly, due to the nullity of the $p + n\max(\mathcal{P})$ first digits of $d(x^n, x)$, we can conclude that $\forall n \geqslant n_0$, $(v^n)^0 = v^0$, and that $\forall n \geqslant n_0$, $(u^n)^0 = u^0$, $(u^n)^1 = u^1$, ..., $(u^n)^{v^0-1} = u^{v^0-1}$. This implies that:

- $G_f(x^n)_1 = G_f(x)_1$: they have the same Boolean vector as first coordinate.
- $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(\Sigma(u^n, v^n); \Sigma(u, v)) = 10^{p+n\max(\mathcal{P})} d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u^n, v^n); (u, v))$. As the right part of the equality tends to 0, we can deduce that it is also the case for the left part of the equality, and so $G_f(x^n)_2$ is convergent to $G_f(x)_2$.

■

3.5. $\Gamma_{\mathcal{P}}(f)$ *as an extension of* $\Gamma(f)$

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{\mathsf{p}}\}$. We define the directed graph $\Gamma_{\mathcal{P}}(f)$ as follows.

- Its vertices are the 2^{N} elements of $\mathbb{B}^{\mathsf{N}}$.
- Each vertex has $\sum_{i=1}^{\mathsf{p}} \mathsf{N}^{p_i}$ arrows, namely all the $p_1, p_2, \ldots, p_{\mathsf{p}}$ tuples having their elements in $[\![1, \mathsf{N}]\!]$.
- There is an arc labeled $u_0, \ldots, u_{p_i-1}$, $i \in [\![1, \mathsf{p}]\!]$ between vertices x and y if and only if $y = F_{f,p_i}(x, (u_0, \ldots, u_{p_i-1}))$.

It is not hard to see that the graph $\Gamma_{\{1\}}(f)$ is $\Gamma(f)$ formerly introduced in [Bahi *et al.*, 2011a] for the $CIPRNG_f^1(u)$ generator, which is indeed $CIPRNG_f^2(u, (1)_{n\in\mathbb{N}})$.

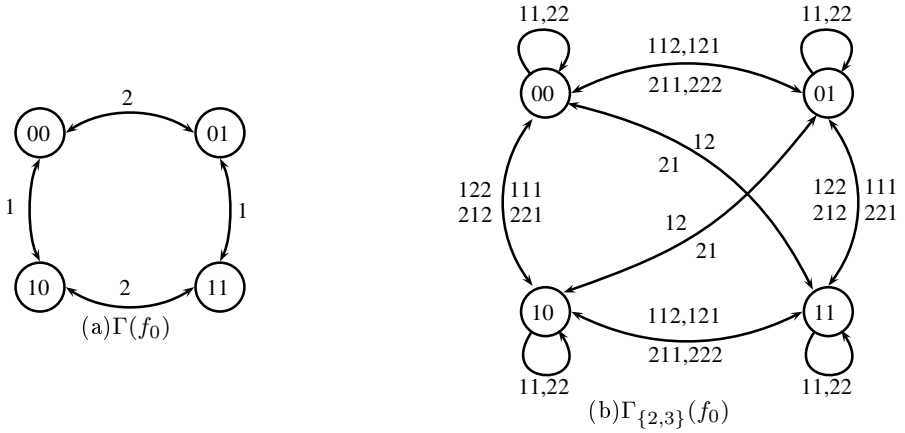

Figure 2. Iterating $f_0 : (x_1, x_2) \mapsto (\overline{x_1}, \overline{x_2})$

Running Example. Consider for instance $\mathsf{N} = 2$, Let $f_0 : \mathbb{B}^2 \longrightarrow \mathbb{B}^2$ be the negation function, *i.e.*, $f_0(x_1, x_2) = (\overline{x_1}, \overline{x_2})$, and consider $\mathcal{P} = \{2, 3\}$. The graphs of iterations are given in Figure 2. Figure 2(a)

shows what happens when each iteration result is displayed . On the contrary, Figure 2(b) illustrates what happens when 2 or 3 modifications are systematically applied before results are generated. Notice that here, the orientations of arcs are not necessary since the function f_0 is equal to its inverse f_0^{-1}. ■

3.6. *Proofs of chaos*

We will show that,

Proposition 3. *$\Gamma_{\mathcal{P}}(f)$ is strongly connected if and only if G_f is topologically transitive on $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$.*

Proof. Suppose that $\Gamma_{\mathcal{P}}(f)$ is strongly connected. Let $x = (e,(u,v)), \check{x} = (\check{e},(\check{u},\check{v})) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}$ and $\varepsilon > 0$. We will find a point y in the open ball $\mathcal{B}(x,\varepsilon)$ and $n_0 \in \mathbb{N}$ such that $G_f^{n_0}(y) = \check{x}$: this strong transitivity will imply the transitivity property. We can suppose that $\varepsilon < 1$ without loss of generality.

Let us denote by $(E,(U,V))$ the elements of y. As y must be in $\mathcal{B}(x,\varepsilon)$ and $\varepsilon < 1$, E must be equal to e. Let $k = \lfloor \log_{10}(\varepsilon) \rfloor + 1$. $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V))$ must be lower than ε, so the k first digits of the fractional part of $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V))$ are null. Let k_1 be the smallest integer such that, if $V^0 = v^0$, ..., $V^{k_1} = v^{k_1}$, $U^0 = u^0$, ..., $U^{\sum_{l=0}^{k_1} V^l - 1} = u^{\sum_{l=0}^{k_1} v^l - 1}$. Then $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V)) < \varepsilon$. In other words, any y of the form $(e,((u^0,...,u^{\sum_{l=0}^{k_1} v^l - 1}),(v^0,...,v^{k_1}))$ is in $\mathcal{B}(x,\varepsilon)$.

Let y^0 such a point and $z = G_f^{k_1}(y^0) = (e',(u',v'))$. $\Gamma_{\mathcal{P}}(f)$ being strongly connected, there is a path between e' and $\check{e}$. Denote by $a_0,\ldots,a_{k_2}$ the edges visited by this path. We denote by $V^{k_1} = |a_0|$ (number of terms in the finite sequence a_1), $V^{k_1+1} = |a_1|$, ..., $V^{k_1+k_2} = |a_{k_2}|$, and by $U^{k_1} = a_0^0$, $U^{k_1+1} = a_0^1$, ..., $U^{k_1+V_{k_1}-1} = a_0^{V_{k_1}-1}$, $U^{k_1+V_{k_1}} = a_1^0$, $U^{k_1+V_{k_1}+1} = a_1^1$,...

Let

$$y = (e,((u^0,\ldots,u^{\sum_{l=0}^{k_1} v^l - 1}, a_0^0,\ldots,a_0^{|a_0|}, a_1^0,\ldots,a_1^{|a_1|},\ldots,a_{k_2}^0,\ldots,a_{k_2}^{|a_{k_2}|}, \check{u}^0,\check{u}^1,\ldots),$$
$$(v^0,\ldots,v^{k_1},|a_0|,\ldots,|a_{k_2}|,\check{v}^0,\check{v}^1,\ldots))).$$

So $y \in \mathcal{B}(x,\varepsilon)$ and $G_f^{k_1+k_2}(y) = \check{x}$.

Conversely, if $\Gamma_{\mathcal{P}}(f)$ is not strongly connected, then there are 2 vertices e_1 and e_2 such that there is no path between e_1 and e_2. Thus, it is impossible to find $(u,v) \in \mathbb{S}_{\mathsf{N},\mathcal{P}}$ and $n \in \mathbb{N}$ such that $G_f^n(e,(u,v))_1 = e_2$. The open ball $\mathcal{B}(e_2,1/2)$ cannot be reached from any neighborhood of e_1, and thus G_f is not transitive. ■

We now show that,

Proposition 4. *If $\Gamma_{\mathcal{P}}(f)$ is strongly connected, then G_f is regular on $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$.*

Proof. Let $x = (e,(u,v)) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}$ and $\varepsilon > 0$. As in the proofs of Prop. 3, let $k_1 \in \mathbb{N}$ such that

$$\Big\{(e,((u^0,\ldots,u^{v^{k_1}-1},U^0,U^1,\ldots),(v^0,\ldots,v^{k_1},V^0,V^1,\ldots)) \mid$$

$$\forall i,j \in \mathbb{N}, U^i \in [\![1,\mathsf{N}]\!], V^j \in \mathcal{P}\Big\} \subset \mathcal{B}(x,\varepsilon),$$

and $y = G_f^{k_1}(e,(u,v))$. $\Gamma_{\mathcal{P}}(f)$ being strongly connected, there is at least a path from the Boolean state y_1 of y to e. Denote by $a_0,\ldots,a_{k_2}$ the edges of such a path. Then the point:
$(e,((u^0,\ldots,u^{v^{k_1}-1},a_0^0,\ldots,a_0^{|a_0|},a_1^0,\ldots,a_1^{|a_1|},\ldots,a_{k_2}^0,\ldots,$
$a_{k_2}^{|a_{k_2}|},u^0,\ldots,u^{v^{k_1}-1},a_0^0,\ldots,a_{k_2}^{|a_{k_2}|}\ldots),$
$(v^0,\ldots,v^{k_1},|a_0|,\ldots,|a_{k_2}|,v^0,\ldots,v^{k_1},|a_0|,\ldots,|a_{k_2}|,\ldots))$ is a periodic point in the neighborhood $\mathcal{B}(x,\varepsilon)$ of x. ■

G_f being topologically transitive and regular, we can thus conclude that

PRNG	LCG	MRG	AWC	SWB	SWC	GFSR	INV
NIST	11	14	15	15	14	14	14
DieHARD	16	16	15	16	18	16	16

PRNG	LCG	MRG	AWC	SWB	SWC	GFSR	INV
NIST	15	15	15	15	15	15	15
DieHARD	18	18	18	18	18	18	18

Theorem 1. *Function G_f is chaotic on $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$ if and only if its iteration graph $\Gamma_{\mathcal{P}}(f)$ is strongly connected.*

Corollary 1. *The pseudorandom number generator $\chi_{14Secrypt}$ is not chaotic on $(\mathcal{X}_{\mathsf{N},\{b\}}, d)$ for the negation function.*

Proof. In this context, $\mathcal{P}$ is the singleton $\{b\}$. If b is even, no vertex e of $\Gamma_{\{b\}}(f_0)$ can reach its neighborhood and thus $\Gamma_{\{b\}}(f_0)$ is not strongly connected. If b is odd, no vertex e of $\Gamma_{\{b\}}(f_0)$ can reach itself and thus $\Gamma_{\{b\}}(f_0)$ is not strongly connected. ■

3.7. *Comparison with other well-known generators*

The objective of this section is to evaluate the statistical performance of the proposed CIPRNG method, by comparing the effects of its application on well-known but defective generators. We considered during the experiments the following PRNGs: linear congruential generator (LCG), multiple recursive generators (MRG) add-with-carry (AWC), subtract-with-borrow (SWB), shift-with-carry (SWC) Generalized Feedback Shift Register (GFSR), and nonlinear inversive generator. A general overview and a reminder of these generators can be found, for instance, in the documentation of the TestU01 statistical battery of tests [L'Ecuyer & Simard, 2007]. For each studied generator, we have compared their scores according to both NIST [Barker & Roginsky, 2010] and DieHARD [Marsaglia, 1996] statistical batteries of tests, by launching them alone or inside the $CIPRNG_f^2(v, v)$ dynamical system, where v is the considered PRNG set with most usual parameters, and f is the vectorial negation.

Obtained results are reproduced in Tables 1 and 2. As can be seen, all these generators considered alone failed to pass either the 15 NIST tests or the 18 DieHARD ones, while both batteries of tests are always passed when applying the $CIPRNG_f^2$ post-treatment. Other results in the same direction, which can be found in [Bahi *et al.*, 2011b], illustrate the fact that operating a provable chaotic post-treatment on defective generators tends to improve their statistical profile.

Such post-treatment depending on the properties of the inputted function f, we need to recall a general scheme to produce functions and an iteration number b such that $\Gamma_{\{b\}}$ is strongly connected.

4. Functions with Strongly Connected $\Gamma_{\{b\}}(f)$

First of all, let $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$. It has been shown [Bahi *et al.*, 2011a, Theorem 4] that if its iteration graph $\Gamma(f)$ is strongly connected, then the output of $\chi_{14Secrypt}$ follows a law that tends to the uniform distribution if and only if its Markov matrix is a doubly stochastic one. In [Couchot *et al.*, 2014, Section 4], we have presented a general scheme which generates function with strongly connected iteration graph $\Gamma(f)$ and with doubly stochastic Markov probability matrix.

Basically, let us consider the N-cube. Let us next remove one Hamiltonian cycle in this one. When an edge (x, y) is removed, an edge (x, x) is added.

Running Example. For instance, the iteration graph $\Gamma(f^*)$ (given in Figure 1) is the 3-cube in which the Hamiltonian cycle $000, 100, 101, 001, 011, 111, 110, 010, 000$ has been removed. ■

We have first proven the following result, which states that the N-cube without one Hamiltonian cycle has the awaited property with regard to the connectivity.

Theorem 2. *The iteration graph $\Gamma(f)$ issued from the N-cube where an Hamiltonian cycle is removed, is strongly connected.*

Moreover, when all the transitions have the same probability ($\frac{1}{n}$), we have proven the following results:

Theorem 3. *The Markov Matrix M resulting from the N-cube in which an Hamiltonian cycle is removed, is doubly stochastic.*

Let us consider now a N-cube where an Hamiltonian cycle is removed. Let f be the corresponding function. The question which remains to be solved is: *can we always find b such that $\Gamma_{\{b\}}(f)$ is strongly connected?*

The answer is indeed positive. Furthermore, we have the following results which are stronger than previous ones.

Theorem 4. *There exists $b \in \mathbb{N}$ such that $\Gamma_{\{b\}}(f)$ is complete.*

Proof. There is an arc (x, y) in the graph $\Gamma_{\{b\}}(f)$ if and only if M^b_{xy} is positive where M is the Markov matrix of $\Gamma(f)$. It has been shown in [Bahi *et al.*, 2011a, Lemma 3] that M is regular. Thus, there exists b such that there is an arc between any x and y. ■

This section ends with the idea of removing a Hamiltonian cycle in the N-cube. In such a context, the Hamiltonian cycle is equivalent to a Gray code. Many approaches have been proposed as a way to build such codes, for instance the Reflected Binary Code. In this one and for a N-length cycle, one of the bits is exactly switched $2^{\mathsf{N}-1}$ times whereas the other bits are modified at most $\left\lfloor \frac{2^{\mathsf{N}-1}}{\mathsf{N}-1} \right\rfloor$ times. It is clear that the function that is built from such a code would not provide a uniform output.

The next section presents how to build balanced Hamiltonian cycles in the N-cube with the objective to embed them into the pseudorandom number generator.

5. Balanced Hamiltonian Cycle

Many approaches have been developed to solve the problem of building a Gray code in a N-cube [Robinson & Cohn, 1981; Bhat & Savage, 1996; Suparta & Zanten, 2004; Bykov, 2016], according to properties the produced code has to verify. For instance, [Bhat & Savage, 1996; Suparta & Zanten, 2004] focus on balanced Gray codes. In the transition sequence of these codes, the number of transitions of each element must differ at most by 2. This uniformity is a global property on the cycle, *i.e.*, a property that is established while traversing the whole cycle. On the other hand, when the objective is to follow a subpart of the Gray code and to switch each element approximately the same amount of times, local properties are wished. For instance, the locally balanced property is studied in [Bykov, 2016] and an algorithm that establishes locally balanced Gray codes is given.

The current context is to provide a function $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ by removing an Hamiltonian cycle in the N-cube. Such a function is going to be iterated b times to produce a pseudorandom number, *i.e.*, a vertex in the N-cube. Obviously, the number of iterations b has to be sufficiently large to provide a uniform output distribution. To reduce the number of iterations, it can be claimed that the provided Gray code should ideally possess both balanced and locally balanced properties. However, both algorithms are incompatible with the second one: balanced Gray codes that are generated by state of the art works [Suparta & Zanten, 2004; Bhat & Savage, 1996] are not locally balanced. Conversely, locally balanced Gray codes yielded by Igor Bykov approach [Bykov, 2016] are not globally balanced. This section thus shows how the non deterministic approach presented in [Suparta & Zanten, 2004] has been automatized to provide balanced Hamiltonian paths such that, for each subpart, the number of switches of each element is as uniform as possible.

5.1. *Analysis of the Robinson-Cohn extension algorithm*

As far as we know three works, namely [Robinson & Cohn, 1981], [Bhat & Savage, 1996], and [Suparta & Zanten, 2004] have addressed the problem of providing an approach to produce balanced gray code. The

authors of [Robinson & Cohn, 1981] introduced an inductive approach aiming at producing balanced Gray codes, assuming the user gives a special subsequence of the transition sequence at each induction step. This work has been strengthened in [Bhat & Savage, 1996] where the authors have explicitly shown how to build such a subsequence. Finally the authors of [Suparta & Zanten, 2004] have presented the *Robinson-Cohn extension* algorithm. Their rigorous presentation of this algorithm has mainly allowed them to prove two properties. The former states that if N is a 2-power, a balanced Gray code is always totally balanced. The latter states that for every N there exists a Gray code such that all transition count numbers are 2-powers whose exponents are either equal or differ from each other by 1. However, the authors do not prove that the approach allows to build (totally balanced) Gray codes. What follows shows that this fact is established and first recalls the approach.

Let be given a $\mathsf{N}-2$-bit Gray code whose transition sequence is $S_{\mathsf{N}-2}$. What follows is the *Robinson-Cohn extension* method [Suparta & Zanten, 2004] which produces a N-bits Gray code.

(1) Let l be an even positive integer. Find $u_1, u_2, \ldots, u_{l-2}, v$ (maybe empty) subsequences of $S_{\mathsf{N}-2}$ such that $S_{\mathsf{N}-2}$ is the concatenation of

$$s_{i_1}, u_0, s_{i_2}, u_1, s_{i_3}, u_2, \ldots, s_{i_l-1}, u_{l-2}, s_{i_l}, v$$

where $i_1 = 1$, $i_2 = 2$, and $u_0 = \emptyset$ (the empty sequence).

(2) Replace in $S_{\mathsf{N}-2}$ the sequences $u_0, u_1, u_2, \ldots, u_{l-2}$ by $\mathsf{N}-1, u'(u_1, \mathsf{N}-1, \mathsf{N}), u'(u_2, \mathsf{N}, \mathsf{N}-1), u'(u_3, \mathsf{N}-1, \mathsf{N}), \ldots, u'(u_{l-2}, \mathsf{N}, \mathsf{N}-1)$ respectively, where $u'(u, x, y)$ is the sequence u, x, u^R, y, u such that u^R is u in reversed order. The obtained sequence is further denoted as U.

(3) Construct the sequences $V = v^R, \mathsf{N}, v$, $W = \mathsf{N}-1, S_{\mathsf{N}-2}, \mathsf{N}$, and let W' be W where the first two elements have been exchanged.

(4) The transition sequence S_{N} is thus the concatenation U^R, V, W'.

It has been proven in [Suparta & Zanten, 2004] that S_{N} is the transition sequence of a cyclic N-bits Gray code if $S_{\mathsf{N}-2}$ is. However, step (1) is not a constructive step that precises how to select the subsequences which ensure that yielded Gray code is balanced. Following sections show how to choose the sequence l to have the balance property.

5.2. *Balanced Codes*

Let us first recall how to formalize the balance property of a Gray code. Let $L = w_1, w_2, \ldots, w_{2^{\mathsf{N}}}$ be the sequence of a N-bits cyclic Gray code. The transition sequence $S = s_1, s_2, \ldots, s_{2^n}$, s_i, $1 \leq i \leq 2^{\mathsf{N}}$, indicates which bit position changes between codewords at index i and $i+1$ modulo 2^{N}. The *transition count* function $TC_{\mathsf{N}} : \{1, \ldots, \mathsf{N}\} \to \{0, \ldots, 2^{\mathsf{N}}\}$ gives the number of times i occurs in S, *i.e.*, the number of times the bit i has been switched in L.

The Gray code is *totally balanced* if TC_{N} is constant (and equal to $\frac{2^{\mathsf{N}}}{\mathsf{N}}$). It is *balanced* if for any two bit indices i and j, $|TC_{\mathsf{N}}(i) - TC_{\mathsf{N}}(j)| \leq 2$.

Running Example. Let $L^* = 000, 100, 101, 001, 011, 111, 110, 010$ be the Gray code that corresponds to the Hamiltonian cycle that has been removed in f^*. Its transition sequence is $S = 3, 1, 3, 2, 3, 1, 3, 2$ and its transition count function is $TC_3(1) = TC_3(2) = 2$ and $TC_3(3) = 4$. Such a Gray code is balanced.

Let $L^4 = 0000, 0010, 0110, 1110, 1111, 0111, 0011, 0001, 0101, 0100, 1100, 1101, 1001, 1011, 1010, 1000$ be a cyclic Gray code. Since $S = 2, 3, 4, 1, 4, 3, 2, 3, 1, 4, 1, 3, 2, 1, 2, 4$, TC_4 is equal to 4 everywhere, this code is thus totally balanced.

On the contrary, for the standard 4-bits Gray code $L^{st} = 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000$, we have $TC_4(1) = 8$ $TC_4(2) = 4$ $TC_4(3) = TC_4(4) = 2$ and the code is neither balanced nor totally balanced. ■

Theorem 5. *Let* N *in* $\mathbb{N}^*$*, and* a_{N} *be defined by* $a_{\mathsf{N}} = 2\left\lfloor \frac{2^{\mathsf{N}}}{2\mathsf{N}} \right\rfloor$*. There exists then a sequence* l *in step (1) of the* Robinson-Cohn extension *algorithm such that all the transition counts* $TC_{\mathsf{N}}(i)$ *are* a_{N} *or* $a_{\mathsf{N}} + 2$ *for any* i*,* $1 \leq i \leq \mathsf{N}$.

The proof is done by induction on N. Let us immediately verify that it is established for both odd and even smallest values, *i.e.*, 3 and 4. For the initial case where $\mathsf{N} = 3$, *i.e.*, $\mathsf{N} - 2 = 1$ we successively have: $S_1 = 1, 1$, $l = 2$, $u_0 = \emptyset$, and $v = \emptyset$. Thus again the algorithm successively produces $U = 1, 2, 1$, $V = 3$, $W = 2, 1, 1, 3$, and $W' = 1, 2, 1, 3$. Finally, S_3 is $1, 2, 1, 3, 1, 2, 1, 3$ which obviously verifies the theorem. For the initial case where $\mathsf{N} = 4$, *i.e.*, $\mathsf{N} - 2 = 2$ we successively have: $S_1 = 1, 2, 1, 2$, $l = 4$, $u_0, u_1, u_2 = \emptyset, \emptyset, \emptyset$, and $v = \emptyset$. Thus again the algorithm successively produces $U = 1, 3, 2, 3, 4, 1, 4, 3, 2$, $V = 4$, $W = 3, 1, 2, 1, 2, 4$, and $W' = 1, 3, 2, 1, 2, 4$. Finally, S_4 is $2, 3, 4, 1, 4, 3, 2, 3, 1, 4, 1, 3, 2, 1, 2, 4$ such that $TC_4(i) = 4$ and the theorem is established for odd and even initial values.

For the inductive case, let us first define some variables. Let c_N (resp. d_N) be the number of elements whose transition count is exactly a_N (resp $a_\mathsf{N} + 2$). Both of these variables are defined by the system

$$\begin{cases} c_\mathsf{N} + d_\mathsf{N} = \mathsf{N} \\ c_\mathsf{N} a_\mathsf{N} + d_\mathsf{N}(a_\mathsf{N} + 2) = 2^\mathsf{N} \end{cases} \Leftrightarrow \begin{cases} d_\mathsf{N} = \dfrac{2^\mathsf{N} - \mathsf{N}.a_\mathsf{N}}{2} \\ c_\mathsf{N} = \mathsf{N} - d_\mathsf{N} \end{cases}$$

Since a_N is even, d_N is an integer. Let us first prove that both c_N and d_N are positive integers. Let q_N and r_N, respectively, be the quotient and the remainder in the Euclidean division of 2^N by $2\mathsf{N}$, *i.e.*, $2^\mathsf{N} = q_\mathsf{N}.2\mathsf{N} + r_\mathsf{N}$, with $0 \leq r_\mathsf{N} < 2\mathsf{N}$. First of all, the integer r is even since $r_\mathsf{N} = 2^\mathsf{N} - q_\mathsf{N}.2\mathsf{N} = 2(2^{\mathsf{N}-1} - q_\mathsf{N}.\mathsf{N})$. Next, a_N is $\frac{2^\mathsf{N} - r_\mathsf{N}}{\mathsf{N}}$. Consequently d_N is $r_\mathsf{N}/2$ and is thus a positive integer s.t. $0 \leq d_\mathsf{N} < \mathsf{N}$. The proof for c_N is obvious.

For any i, $1 \leq i \leq \mathsf{N}$, let zi_N (resp. ti_N and bi_N) be the occurrence number of element i in the sequence $u_0, \ldots, u_{l-2}$ (resp. in the sequences $s_{i_1}, \ldots, s_{i_l}$ and v) in step (1) of the algorithm.

Due to the definition of u' in step (2), $3.zi_\mathsf{N} + ti_\mathsf{N}$ is the number of element i in the sequence U. It is clear that the number of element i in the sequence V is $2bi_\mathsf{N}$ due to step (3). We thus have the following system:

$$\begin{cases} 3.zi_\mathsf{N} + ti_\mathsf{N} + 2.bi_\mathsf{N} + TC_{\mathsf{N}-2}(i) = TC_\mathsf{N}(i) \\ zi_\mathsf{N} + ti_\mathsf{N} + bi_\mathsf{N} = TC_{\mathsf{N}-2}(i) \end{cases} \Leftrightarrow$$

$$\begin{cases} zi_\mathsf{N} = \dfrac{TC_\mathsf{N}(i) - 2.TC_{\mathsf{N}-2}(i) - bi_\mathsf{N}}{2} \\ ti_\mathsf{N} = TC_{\mathsf{N}-2}(i) - zi_\mathsf{N} - bi_\mathsf{N} \end{cases} \tag{3}$$

In this set of 2 equations with 3 unknown variables, let b_i be set with 0. In this case, since TC_N is even (equal to a_N or to $a_\mathsf{N}+2$), the variable zi_N is thus an integer. Let us now prove that the resulting system has always positive integer solutions z_i, t_i, $0 \leq z_i, t_i \leq TC_{\mathsf{N}-2}(i)$ and s.t. their sum is equal to $TC_{\mathsf{N}-2}(i)$. This latter constraint is obviously established if the system has a solution. We thus have the following system.

$$\begin{cases} zi_\mathsf{N} = \dfrac{TC_\mathsf{N}(i) - 2.TC_{\mathsf{N}-2}(i)}{2} \\ ti_\mathsf{N} = TC_{\mathsf{N}-2}(i) - zi_\mathsf{N} \end{cases} \tag{4}$$

The definition of $TC_\mathsf{N}(i)$ depends on the value of N. When $3 \leq N \leq 7$, values are defined as follows:

$$TC_3 = [2, 2, 4]$$
$$TC_5 = [6, 6, 8, 6, 6]$$
$$TC_7 = [18, 18, 20, 18, 18, 18, 18]$$

$$TC_4 = [4, 4, 4, 4]$$
$$TC_6 = [10, 10, 10, 10, 12, 12]$$

It is not difficult to check that all these instanciations verify the aforementioned constraints.

When $N \geq 8$, $TC_{\mathsf{N}}(i)$ is defined as follows:

$$TC_{\mathsf{N}}(i) = \begin{cases} a_{\mathsf{N}} \text{ if } 1 \leq i \leq c_{\mathsf{N}} \\ a_{\mathsf{N}} + 2 \text{ if } c_{\mathsf{N}} + 1 \leq i \leq c_{\mathsf{N}} + d_{\mathsf{N}} \end{cases} \tag{5}$$

We thus have

$$\begin{aligned} TC_{\mathsf{N}}(i) - 2.TC_{\mathsf{N}-2}(i) &\geq a_{\mathsf{N}} - 2(a_{\mathsf{N}-2} + 2) \\ &\geq \tfrac{2^{\mathsf{N}} - r_{\mathsf{N}}}{\mathsf{N}} - 2\left(\tfrac{2^{\mathsf{N}-2} - r_{\mathsf{N}-2}}{\mathsf{N}-2} + 2\right) \\ &\geq \tfrac{2^{\mathsf{N}} - 2N}{\mathsf{N}} - 2\left(\tfrac{2^{\mathsf{N}-2}}{\mathsf{N}-2} + 2\right) \\ &\geq \tfrac{(\mathsf{N}-2).2^{\mathsf{N}} - 2N.2^{\mathsf{N}-2} - 6N(N-2)}{\mathsf{N}.(\mathsf{N}-2)} \end{aligned}$$

A simple variation study of the function $t : \mathbb{R} \to \mathbb{R}$ such that $x \mapsto t(x) = (x-2).2^x - 2x.2^{x-2} - 6x(x-2)$ shows that its derivative is strictly positive if $x \geq 6$ and $t(8) = 224$. The integer $TC_{\mathsf{N}}(i) - 2.TC_{\mathsf{N}-2}(i)$ is thus positive for any $\mathsf{N} \geq 8$ and the proof is established.

For each element i, we are then left to choose zi_{N} positions among $TC_{\mathsf{N}}(i)$, which leads to $\binom{TC_{\mathsf{N}}(i)}{zi_{\mathsf{N}}}$ possibilities. Notice that all such choices lead to an Hamiltonian path.

6. Mixing Time

This section considers functions $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ issued from an hypercube where an Hamiltonian path has been removed as described in the previous section. Notice that the iteration graph is always a subgraph of N-cube augmented with all the self-loop, *i.e.*, all the edges (v, v) for any $v \in \mathbb{B}^{\mathsf{N}}$. Next, if we add probabilities on the transition graph, iterations can be interpreted as Markov chains.

Running Example. Let us consider for instance the graph $\Gamma(f)$ defined in Figure 1 and the probability function p defined on the set of edges as follows:

$$p(e) \begin{cases} = \frac{2}{3} \text{ if } e = (v, v) \text{ with } v \in \mathbb{B}^3, \\ = \frac{1}{6} \text{ otherwise.} \end{cases}$$

The matrix P of the Markov chain associated to the function f^* and to its probability function p is

$$P = \frac{1}{6} \begin{pmatrix} 4&1&1&0&0&0&0&0 \\ 1&4&0&0&0&1&0&0 \\ 0&0&4&1&0&0&1&0 \\ 0&1&1&4&0&0&0&0 \\ 1&0&0&0&4&0&1&0 \\ 0&0&0&0&1&4&0&1 \\ 0&0&0&0&1&0&4&1 \\ 0&0&0&1&0&1&0&4 \end{pmatrix}.$$

■

A specific random walk in this modified hypercube is first introduced (see Section 6.1). We further study this random walk in a theoretical way to provide an upper bound of fair sequences (see Section 6.2). We finally complete this study with experimental results that reduce this bound (Sec. 6.3). For a general reference on Markov chains, see [Levin *et al.*, 2006], and particularly Chapter 5 on stopping times.

6.1. *Formalizing the Random Walk*

First of all, let π, μ be two distributions on $\mathbb{B}^{\mathsf{N}}$. The total variation distance between π and μ is denoted $\|\pi - \mu\|_{\mathrm{TV}}$ and is defined by

$$\|\pi - \mu\|_{\mathrm{TV}} = \max_{A \subset \mathbb{B}^{\mathsf{N}}} |\pi(A) - \mu(A)|.$$

It is known that

$$\|\pi - \mu\|_{\mathrm{TV}} = \frac{1}{2} \sum_{X \in \mathbb{B}^{\mathsf{N}}} |\pi(X) - \mu(X)|.$$

Moreover, if ν is a distribution on $\mathbb{B}^{\mathsf{N}}$, one has

$$\|\pi - \mu\|_{\mathrm{TV}} \leq \|\pi - \nu\|_{\mathrm{TV}} + \|\nu - \mu\|_{\mathrm{TV}}$$

Let P be the matrix of a Markov chain on $\mathbb{B}^{\mathsf{N}}$. For any $X \in \mathbb{B}^{\mathsf{N}}$, let $P(X, \cdot)$ be the distribution induced by the $\mathrm{bin}(X)$-th row of P, where $\mathrm{bin}(X)$ is the integer whose binary encoding is X. If the Markov chain induced by P has a stationary distribution π, then we define

$$d(t) = \max_{X \in \mathbb{B}^{\mathsf{N}}} \|P^t(X, \cdot) - \pi\|_{\mathrm{TV}}.$$

and

$$t_{\mathrm{mix}}(\varepsilon) = \min\{t \mid d(t) \leq \varepsilon\}.$$

Intuitively speaking, $t_{\mathrm{mix}}(\varepsilon)$ is the time/steps required to be sure to be ε-close to the stationary distribution, wherever the chain starts.

One can prove that

$$t_{\mathrm{mix}}(\varepsilon) \leq \lceil \log_2(\varepsilon^{-1}) \rceil t_{\mathrm{mix}}(\frac{1}{4})$$

Let $(X_t)_{t \in \mathbb{N}}$ be a sequence of $\mathbb{B}^{\mathsf{N}}$ valued random variables. A $\mathbb{N}$-valued random variable τ is a *stopping time* for the sequence (X_i) if for each t there exists $B_t \subseteq (\mathbb{B}^{\mathsf{N}})^{t+1}$ such that $\{\tau = t\} = \{(X_0, X_1, \ldots, X_t) \in B_t\}$. In other words, the event $\{\tau = t\}$ only depends on the values of $(X_0, X_1, \ldots, X_t)$, not on X_k with $k > t$.

Let $(X_t)_{t \in \mathbb{N}}$ be a Markov chain and $f(X_{t-1}, Z_t)$ a random mapping representation of the Markov chain. A *randomized stopping time* for the Markov chain is a stopping time for $(Z_t)_{t \in \mathbb{N}}$. If the Markov chain is irreducible and has π as stationary distribution, then a *stationary time* τ is a randomized stopping time (possibly depending on the starting position X), such that the distribution of X_τ is π:

$$\mathbb{P}_X(X_\tau = Y) = \pi(Y).$$

6.2. *Upper bound of Stopping Time*

A stopping time τ is a *strong* stationary time if X_τ is independent of τ. The following result will be useful [Levin *et al.*, 2006, Proposition 6.10],

Theorem 6. *If τ is a strong stationary time, then $d(t) \leq \max_{X \in \mathbb{B}^{\mathsf{N}}} \mathbb{P}_X(\tau > t)$.*

Let $E = \{(X, Y) \mid X \in \mathbb{B}^{\mathsf{N}}, Y \in \mathbb{B}^{\mathsf{N}},\ X = Y \text{ or } X \oplus Y \in 0^*10^*\}$. In other words, E is the set of all the edges in the classical N-cube. Let h be a function from $\mathbb{B}^{\mathsf{N}}$ into $[\![1, \mathsf{N}]\!]$. Intuitively speaking h aims at memorizing for each node $X \in \mathbb{B}^{\mathsf{N}}$ whose edge is removed in the Hamiltonian cycle, *i.e.*, which bit in $[\![1, \mathsf{N}]\!]$ cannot be switched.

We denote by E_h the set $E \setminus \{(X, Y) \mid X \oplus Y = 0^{\mathsf{N}-h(X)}10^{h(X)-1}\}$. This is the set of the modified hypercube, *i.e.*, the N-cube where the Hamiltonian cycle h has been removed.

We define the Markov matrix P_h for each line X and each column Y as follows:

$$\begin{cases} P_h(X, X) = \frac{1}{2} + \frac{1}{2\mathsf{N}} & \\ P_h(X, Y) = 0 & \text{if } (X, Y) \notin E_h \\ P_h(X, Y) = \frac{1}{2\mathsf{N}} & \text{if } X \neq Y \text{ and } (X, Y) \in E_h \end{cases} \tag{6}$$

We denote by $\overline{h} : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ the function such that for any $X \in \mathbb{B}^{\mathsf{N}}$, $(X, \overline{h}(X)) \in E$ and $X \oplus \overline{h}(X) = 0^{\mathsf{N}-h(X)}10^{h(X)-1}$. The function $\overline{h}$ is said to be *square-free* if for every $X \in \mathbb{B}^{\mathsf{N}}$, $\overline{h}(\overline{h}(X)) \neq X$.

Lemma 1. *If $\overline{h}$ is bijective and square-free, then $h(\overline{h}^{-1}(X)) \neq h(X)$.*

Proof. Let $\overline{h}$ be bijective. Let $k \in [\![1, \mathsf{N}]\!]$ s.t. $h(\overline{h}^{-1}(X)) = k$. Then $(\overline{h}^{-1}(X), X)$ belongs to E and $\overline{h}^{-1}(X) \oplus X = 0^{\mathsf{N}-k}10^{k-1}$. Let us suppose $h(X) = h(\overline{h}^{-1}(X))$. In such a case, $h(X) = k$. By definition of $\overline{h}$, $(X, \overline{h}(X)) \in E$ and $X \oplus \overline{h}(X) = 0^{\mathsf{N}-h(X)}10^{h(X)-1} = 0^{\mathsf{N}-k}10^{k-1}$. Thus $\overline{h}(X) = \overline{h}^{-1}(X)$, which leads to $\overline{h}(\overline{h}(X)) = X$. This contradicts the square-freeness of $\overline{h}$. ■

Let Z be a random variable that is uniformly distributed over $[\![1, \mathsf{N}]\!] \times \mathbb{B}$. For $X \in \mathbb{B}^{\mathsf{N}}$, we define, with $Z = (i, b)$,

$$\begin{cases} f(X, Z) = X \oplus (0^{\mathsf{N}-i}10^{i-1}) & \text{if } b = 1 \text{ and } i \neq h(X), \\ f(X, Z) = X & \text{otherwise.} \end{cases}$$

The Markov chain is thus defined as

$$X_t = f(X_{t-1}, Z_t)$$

An integer $\ell \in [\![1, \mathsf{N}]\!]$ is said *fair* at time t if there exists $0 \leq j < t$ such that $Z_{j+1} = (\ell, \cdot)$ and $h(X_j) \neq \ell$. In other words, there exists a date j before t where the first element of the random variable Z is exactly l (*i.e.*, l is the strategy at date j) and where the configuration X_j allows to cross the edge l.

Let τ_{stop} be the first time all the elements of $[\![1, \mathsf{N}]\!]$ are fair. The integer τ_{stop} is a randomized stopping time for the Markov chain (X_t).

Lemma 2. *The integer τ_{stop} is a strong stationary time.*

Proof. Let τ_ℓ be the first time that ℓ is fair. The random variable Z_{τ_ℓ} is of the form (ℓ, b) such that $b = 1$ with probability $\frac{1}{2}$ and $b = 0$ with probability $\frac{1}{2}$. Since $h(X_{\tau_\ell - 1}) \neq \ell$ the value of the ℓ-th bit of X_{τ_ℓ} is 0 or 1 with the same probability $(\frac{1}{2})$. This probability is independent of the value of the other bits.

Moving next in the chain, at each step, the l-th bit is switched from 0 to 1 or from 1 to 0 each time with the same probability. Therefore, for $t \geq \tau_\ell$, the ℓ-th bit of X_t is 0 or 1 with the same probability, and independently of the value of the other bits, proving the lemma. ■

Theorem 7. *If $\overline{h}$ is bijective and square-free, then $E[\tau_{\text{stop}}] \leq 8\mathsf{N}^2 + 4\mathsf{N}\ln(\mathsf{N}+1)$.*

For each $X \in \mathbb{B}^{\mathsf{N}}$ and $\ell \in [\![1, \mathsf{N}]\!]$, let $S_{X,\ell}$ be the random variable that counts the number of steps from X until we reach a configuration where ℓ is fair. More formally

$$S_{X,\ell} = \min\{t \geq 1 \mid h(X_{t-1}) \neq \ell \text{ and } Z_t = (\ell, .) \text{ and } X_0 = X\}.$$

Lemma 3. *Let $\overline{h}$ is a square-free bijective function. Then for all X and all ℓ, the inequality $E[S_{X,\ell}] \leq 8\mathsf{N}^2$ is established.*

Proof. For every X, every ℓ, one has $\mathbb{P}(S_{X,\ell} \leq 2) \geq \frac{1}{4\mathsf{N}^2}$. Let $X_0 = X$. Indeed,

- if $h(X) \neq \ell$, then $\mathbb{P}(S_{X,\ell} = 1) = \frac{1}{2\mathsf{N}} \geq \frac{1}{4\mathsf{N}^2}$.
- otherwise, $h(X) = \ell$, then $\mathbb{P}(S_{X,\ell} = 1) = 0$. But in this case, intuitively, it is possible to move from X to $\overline{h}^{-1}(X)$ (with probability $\frac{1}{2N}$). And in $\overline{h}^{-1}(X)$ the l-th bit can be switched. More formally, since $\overline{h}$ is square-free, $\overline{h}(X) = \overline{h}(\overline{h}(\overline{h}^{-1}(X))) \neq \overline{h}^{-1}(X)$. It follows that $(X, \overline{h}^{-1}(X)) \in E_h$. We thus have $P(X_1 = \overline{h}^{-1}(X)) = \frac{1}{2\mathsf{N}}$. Now, by Lemma 1, $h(\overline{h}^{-1}(X)) \neq h(X)$. Therefore $\mathbb{P}(S_{x,\ell} = 2 \mid X_1 = \overline{h}^{-1}(X)) = \frac{1}{2\mathsf{N}}$, proving that $\mathbb{P}(S_{x,\ell} \leq 2) \geq \frac{1}{4\mathsf{N}^2}$.

Therefore, $\mathbb{P}(S_{X,\ell} \geq 3) \leq 1 - \frac{1}{4\mathsf{N}^2}$. By induction, one has, for every i, $\mathbb{P}(S_{X,\ell} \geq 2i) \leq \left(1 - \frac{1}{4\mathsf{N}^2}\right)^i$. Moreover, since $S_{X,\ell}$ is positive, it is known [Mitzenmacher & Upfal, 2005, lemma 2.9], that

$$E[S_{X,\ell}] = \sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq i).$$

Since $\mathbb{P}(S_{X,\ell} \geq i) \geq \mathbb{P}(S_{X,\ell} \geq i+1)$, one has

$$\begin{aligned} E[S_{X,\ell}] &= \textstyle\sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq i) \\ &\leq \mathbb{P}(S_{X,\ell} \geq 1) + \mathbb{P}(S_{X,\ell} \geq 2) \\ &\quad +2\textstyle\sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq 2i). \end{aligned}$$

Consequently,

$$E[S_{X,\ell}] \leq 1 + 1 + 2\sum_{i=1}^{+\infty} \left(1 - \frac{1}{4\mathsf{N}^2}\right)^i = 2 + 2(4\mathsf{N}^2 - 1) = 8\mathsf{N}^2,$$

which concludes the proof. ■

Let τ'_{stop} be the time used to get all the bits but one fair.

Lemma 4. *One has* $E[\tau'_{\text{stop}}] \leq 4\mathsf{N}\ln(\mathsf{N}+1)$.

Proof. This is a classical Coupon Collector's like problem. Let W_i be the random variable counting the number of moves done in the Markov chain while we had exactly $i-1$ fair bits. One has $\tau'_{\text{stop}} = \sum_{i=1}^{\mathsf{N}-1} W_i$. But when we are at position X with $i-1$ fair bits, the probability of obtaining a new fair bit is either $1 - \frac{i-1}{\mathsf{N}}$ if $h(X)$ is fair, or $1 - \frac{i-2}{\mathsf{N}}$ if $h(X)$ is not fair.

Therefore, $\mathbb{P}(W_i = k) \leq \left(\frac{i-1}{\mathsf{N}}\right)^{k-1} \frac{\mathsf{N}-i+2}{\mathsf{N}}$. Consequently, we have $\mathbb{P}(W_i \geq k) \leq \left(\frac{i-1}{\mathsf{N}}\right)^{k-1} \frac{\mathsf{N}-i+2}{\mathsf{N}-i+1}$. It follows that $E[W_i] = \sum_{k=1}^{+\infty} \mathbb{P}(W_i \geq k) \leq \mathsf{N}\frac{\mathsf{N}-i+2}{(\mathsf{N}-i+1)^2} \leq \frac{4\mathsf{N}}{\mathsf{N}-i+2}$.

It follows that $E[W_i] \leq \frac{4\mathsf{N}}{\mathsf{N}-i+2}$. Therefore

$$E[\tau'_{\text{stop}}] = \sum_{i=1}^{\mathsf{N}-1} E[W_i] \leq 4\mathsf{N}\sum_{i=1}^{\mathsf{N}-1} \frac{1}{\mathsf{N}-i+2} = 4\mathsf{N}\sum_{i=3}^{\mathsf{N}+1} \frac{1}{i}.$$

But $\sum_{i=1}^{\mathsf{N}+1} \frac{1}{i} \leq 1 + \ln(\mathsf{N}+1)$. It follows that $1 + \frac{1}{2} + \sum_{i=3}^{\mathsf{N}+1} \frac{1}{i} \leq 1 + \ln(\mathsf{N}+1)$. Consequently, $E[\tau'_{\text{stop}}] \leq 4\mathsf{N}(-\frac{1}{2} + \ln(\mathsf{N}+1)) \leq 4\mathsf{N}\ln(\mathsf{N}+1)$. ■

One can now prove Theorem 7.

Proof. Since τ'_{stop} is the time used to obtain $\mathsf{N}-1$ fair bits. Assume that the last unfair bit is ℓ. One has $\tau_{\text{stop}} = \tau'_{\text{stop}} + S_{X_\tau,\ell}$, and therefore $E[\tau_{\text{stop}}] = E[\tau'_{\text{stop}}] + E[S_{X_\tau,\ell}]$. Therefore, Theorem 7 is a direct application of Lemma 3 and 4. ■

Now using Markov Inequality, one has $\mathbb{P}_X(\tau > t) \leq \frac{E[\tau]}{t}$. With $t_n = 32N^2 + 16N\ln(N+1)$, one obtains: $\mathbb{P}_X(\tau > t_n) \leq \frac{1}{4}$. Therefore, using the definition of t_{mix} and Theorem 6, it follows that $t_{\text{mix}} \leq 32N^2 + 16N\ln(N+1) = O(N^2)$.

Notice that the calculus of the stationary time upper bound is obtained under the following constraint: for each vertex in the N-cube there are one ongoing arc and one outgoing arc that are removed. The calculus doesn't consider (balanced) Hamiltonian cycles, which are more regular and more binding than this constraint. Moreover, the bound is obtained using the coarse Markov Inequality. For the classical (lazy) random walk the N-cube, without removing any Hamiltonian cycle, the mixing time is in $\Theta(N\ln N)$. We conjecture that in our context, the mixing time is also in $\Theta(N\ln N)$.

In this latter context, we claim that the upper bound for the stopping time should be reduced. This fact is studied in the next section.

6.3. *Practical Evaluation of Stopping Times*

Let be given a function $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ and an initial seed x^0. The pseudo code given in Algorithm 2 returns the smallest number of iterations such that all elements $\ell \in [\![1, \mathsf{N}]\!]$ are fair. It allows to deduce an approximation of $E[\tau_{\text{stop}}]$ by calling this code many times with many instances of function and many seeds.

Input: a function f, an initial configuration x^0 (N bits)
Output: a number of iterations $nbit$
$nbit \leftarrow 0$;
$x \leftarrow x^0$;
$fair \leftarrow \emptyset$;
while $|fair| < \mathsf{N}$ **do**
$s \leftarrow Random(\mathsf{N})$;
$image \leftarrow f(x)$;
if $Random(1) \neq 0$ *and* $x[s] \neq image[s]$ **then**
$fair \leftarrow fair \cup \{s\}$;
$x[s] \leftarrow image[s]$;
end
$nbit \leftarrow nbit + 1$;
end
return $nbit$;

Algorithm 2: Pseudo Code of stopping time computation

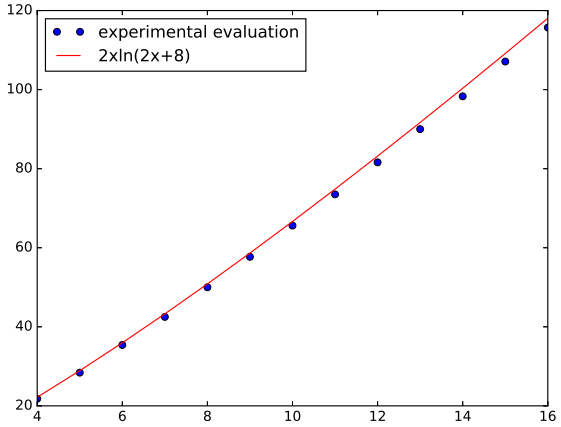

Figure 3. Average Stopping Time Approximation

Practically speaking, for each number N, $3 \leq \mathsf{N} \leq 16$, 10 functions have been generated according to the method presented in Section 5. For each of them, the calculus of the approximation of $E[\tau_{\text{stop}}]$ is executed 10000 times with a random seed. Figure 3 summarizes these results. A circle represents the approximation of $E[\tau_{\text{stop}}]$ for a given N. The line is the graph of the function $x \mapsto 2x\ln(2x+8)$. It can firstly be observed that the approximation is largely smaller than the upper bound given in Theorem 7. It can be further deduced that the conjecture of the previous section is realistic according to the graph of $x \mapsto 2x\ln(2x+8)$.

7. Experiments

Let us finally present the pseudorandom number generator χ_{16HamG}, which is based on random walks in $\Gamma_{\{b\}}(f)$. More precisely, let be given a Boolean map $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$, a PRNG *Random*, an integer b that corresponds to an iteration number (*i.e.*, the length of the walk), and an initial configuration x^0. Starting from x^0, the algorithm repeats b times a random choice of which edge to follow, and crosses this edge provided it is allowed to do so, *i.e.*, when $Random(1)$ is not null. The final configuration is thus outputted. This PRNG is formalized in Algorithm 3.

This PRNG is slightly different from $\chi_{14Secrypt}$ recalled in Algorithm 1. As this latter, the length of the random walk of our algorithm is always constant (and is equal to b). However, in the current version,

Input: a function f, an iteration number b, an initial configuration x^0 (N bits)
Output: a configuration x (N bits)
$x \leftarrow x^0$;
for $i = 0, \ldots, b-1$ **do**
if $Random(1) \neq 0$ **then**
$s^0 \leftarrow Random(\mathsf{N})$;
$x \leftarrow F_f(x, s^0)$;
end
end
return x;

Algorithm 3: Pseudo Code of the χ_{16HamG} PRNG

Function f	$f(x)$, for x in $(0, 1, 2, \ldots, 2^n - 1)$	N	b
ⓐ	[13,10,9,14,3,11,1,12,15,4,7,5,2,6,0,8]	4	64
ⓑ	[29, 22, 25, 30, 19, 27, 24, 16, 21, 6, 5, 28, 23, 26, 1, 17, 31, 12, 15, 8, 10, 14, 13, 9, 3, 2, 7, 20, 11, 18, 0, 4]	5	78
ⓒ	[55, 60, 45, 44, 58, 62, 61, 48, 53, 50, 52, 36, 59, 34, 33, 49, 15, 42, 47, 46, 35, 10, 57, 56, 7, 54, 39, 37, 51, 2, 1, 40, 63, 26, 25, 30, 19, 27, 17, 28, 31, 20, 23, 21, 18, 22, 16, 24, 13, 12, 29, 8, 43, 14, 41, 0, 5, 38, 4, 6, 11, 3, 9, 32]	6	88
ⓓ	[111, 124, 93, 120, 122, 114, 89, 121, 87, 126, 125, 84, 123, 82, 112, 80, 79, 106, 105, 110, 75, 107, 73, 108, 119, 100, 117, 116, 103, 102, 101, 97, 31, 86, 95, 94, 83, 26, 88, 24, 71, 118, 69, 68, 115, 90, 113, 16, 15, 76, 109, 72, 74, 10, 9, 104, 7, 6, 65, 70, 99, 98, 64, 96, 127, 54, 53, 62, 51, 59, 56, 60, 39, 52, 37, 36, 55, 58, 57, 49, 63, 44, 47, 40, 42, 46, 45, 41, 35, 34, 33, 38, 43, 50, 32, 48, 29, 28, 61, 92, 91, 18, 17, 25, 19, 30, 85, 22, 27, 2, 81, 0, 13, 78, 77, 14, 3, 11, 8, 12, 23, 4, 21, 20, 67, 66, 5, 1]	7	99
ⓔ	[223, 238, 249, 254, 243, 251, 233, 252, 183, 244, 229, 245, 227, 246, 240, 176, 175, 174, 253, 204, 203, 170, 169, 248, 247, 226, 228, 164, 163, 162, 161, 192, 215, 220, 205, 216, 155, 222, 221, 208, 213, 150, 212, 214, 219, 211, 145, 209, 239, 202, 207, 140, 195, 234, 193, 136, 231, 230, 199, 197, 131, 198, 225, 200, 63, 188, 173, 184, 186, 250, 57, 168, 191, 178, 180, 52, 187, 242, 241, 48, 143, 46, 237, 236, 235, 138, 185, 232, 135, 38, 181, 165, 35, 166, 33, 224, 31, 30, 153, 158, 147, 218, 217, 156, 159, 148, 151, 149, 19, 210, 144, 152, 141, 206, 13, 12, 171, 10, 201, 128, 133, 130, 132, 196, 3, 194, 137, 0, 255, 124, 109, 120, 122, 106, 125, 104, 103, 114, 116, 118, 123, 98, 97, 113, 79, 126, 111, 110, 99, 74, 121, 72, 71, 70, 117, 101, 115, 102, 65, 112, 127, 90, 89, 94, 83, 91, 81, 92, 95, 84, 87, 85, 82, 86, 80, 88, 77, 76, 93, 108, 107, 78, 105, 64, 69, 66, 68, 100, 75, 67, 73, 96, 55, 190, 189, 62, 51, 59, 41, 60, 119, 182, 37, 53, 179, 54, 177, 32, 45, 44, 61, 172, 11, 58, 9, 56, 167, 34, 36, 4, 43, 50, 49, 160, 23, 28, 157, 24, 26, 154, 29, 16, 21, 18, 20, 22, 27, 146, 25, 17, 47, 142, 15, 14, 139, 42, 1, 40, 39, 134, 7, 5, 2, 6, 129, 8]	8	109

we add the constraint that the probability to execute the function F_f is equal to 0.5 since the output of *Random(1)* is uniform in $\{0, 1\}$. This constraint is added to match the theoretical framework of Sect. 6.

Notice that the chaos property of G_f given in Sect.3 only requires the graph $\Gamma_{\{b\}}(f)$ to be strongly connected. Since the χ_{16HamG} algorithm only adds probability constraints on existing edges, it preserves this property.

For each number $\mathsf{N} = 4, 5, 6, 7, 8$ of bits, we have generated the functions according to the method given in Sect. 4 and 5. For each N, we have then restricted this evaluation to the function whose Markov Matrix (issued from Eq. (6)) has the smallest practical mixing time. Such functions are given in Table 3. In this table, let us consider, for instance, the function ⓐ from $\mathbb{B}^4$ to $\mathbb{B}^4$ defined by the following images : $[13, 10, 9, 14, 3, 11, 1, 12, 15, 4, 7, 5, 2, 6, 0, 8]$. In other words, the image of 3 (0011) by ⓐ is 14 (1110): it is obtained as the binary value of the fourth element in the second list (namely 14).

In this table the column that is labeled with b gives the practical mixing time where the deviation to the standard distribution is inferior than 10^{-6}.

Let us first discuss about results against the NIST test suite. In our experiments, 100 sequences (s = 100) of 1,000,000 bits are generated and tested. If the value $\mathbb{P}_T$ of any test is smaller than 0.0001, the

Test	MT_4	MT_5	MT_6	MT_7	MT_8
Frequency (Monobit)	0.924 (1.0)	0.678 (0.98)	0.102 (0.97)	0.213 (0.98)	0.719 (0.99)
Frequency within a Block	0.514 (1.0)	0.419 (0.98)	0.129 (0.98)	0.275 (0.99)	0.455 (0.99)
Cumulative Sums (Cusum) *	0.668 (1.0)	0.568 (0.99)	0.881 (0.98)	0.529 (0.98)	0.657 (0.995)
Runs	0.494 (0.99)	0.595 (0.97)	0.071 (0.97)	0.017 (1.0)	0.834 (1.0)
Longest Run of Ones in a Block	0.366 (0.99)	0.554 (1.0)	0.042 (0.99)	0.051 (0.99)	0.897 (0.97)
Binary Matrix Rank	0.275 (0.98)	0.494 (0.99)	0.719 (1.0)	0.334 (0.98)	0.637 (0.99)
Discrete Fourier Transform (Spectral)	0.122 (0.98)	0.108 (0.99)	0.108 (1.0)	0.514 (0.99)	0.534 (0.98)
Non-overlapping Template Matching*	0.483 (0.990)	0.507 (0.990)	0.520 (0.988)	0.494 (0.988)	0.515 (0.989)
Overlapping Template Matching	0.595 (0.99)	0.759 (1.0)	0.637 (1.0)	0.554 (0.99)	0.236 (1.0)
Maurer's "Universal Statistical"	0.202 (0.99)	0.000 (0.99)	0.514 (0.98)	0.883 (0.97)	0.366 (0.99)
Approximate Entropy (m=10)	0.616 (0.99)	0.145 (0.99)	0.455 (0.99)	0.262 (0.97)	0.494 (1.0)
Random Excursions *	0.275 (1.0)	0.495 (0.975)	0.465 (0.979)	0.452 (0.991)	0.260 (0.989)
Random Excursions Variant *	0.382 (0.995)	0.400 (0.994)	0.417 (0.984)	0.456 (0.991)	0.389 (0.991)
Serial* (m=10)	0.629 (0.99)	0.963 (0.99)	0.366 (0.995)	0.537 (0.985)	0.253 (0.995)
Linear Complexity	0.494 (0.99)	0.514 (0.98)	0.145 (1.0)	0.657 (0.98)	0.145 (0.99)
Test	ⓐ	ⓑ	ⓒ	ⓓ	ⓔ
Frequency (Monobit)	0.129 (1.0)	0.181 (1.0)	0.637 (0.99)	0.935 (1.0)	0.978 (1.0)
Frequency within a Block	0.275 (1.0)	0.534 (0.98)	0.066 (1.0)	0.719 (1.0)	0.366 (1.0)
Cumulative Sums (Cusum) *	0.695 (1.0)	0.540 (1.0)	0.514 (0.985)	0.773 (0.995)	0.506 (0.99)
Runs	0.897 (0.99)	0.051 (1.0)	0.102 (0.98)	0.616 (0.99)	0.191 (1.0)
Longest Run of Ones in a Block	0.851 (1.0)	0.595 (0.99)	0.419 (0.98)	0.616 (0.98)	0.897 (1.0)
Binary Matrix Rank	0.419 (1.0)	0.946 (0.99)	0.319 (0.99)	0.739 (0.97)	0.366 (1.0)
Discrete Fourier Transform (Spectral)	0.867 (1.0)	0.514 (1.0)	0.145 (1.0)	0.224 (0.99)	0.304 (1.0)
Non-overlapping Template Matching*	0.542 (0.990)	0.512 (0.989)	0.505 (0.990)	0.494 (0.989)	0.493 (0.991)
Overlapping Template Matching	0.275 (0.99)	0.080 (0.99)	0.574 (0.98)	0.798 (0.99)	0.834 (0.99)
Maurer's "Universal Statistical"	0.383 (0.99)	0.991 (0.98)	0.851 (1.0)	0.595 (0.98)	0.514 (1.0)
Approximate Entropy (m=10)	0.935 (1.0)	0.719 (1.0)	0.883 (1.0)	0.719 (0.97)	0.366 (0.99)
Random Excursions *	0.396 (0.991)	0.217 (0.989)	0.445 (0.975)	0.743 (0.993)	0.380 (0.990)
Random Excursions Variant *	0.486 (0.997)	0.373 (0.981)	0.415 (0.994)	0.424 (0.991)	0.380 (0.991)
Serial* (m=10)	0.350 (1.0)	0.678 (0.995)	0.287 (0.995)	0.740 (0.99)	0.301 (0.98)
Linear Complexity	0.455 (0.99)	0.867 (1.0)	0.401 (0.99)	0.191 (0.97)	0.699 (1.0)

sequences are considered to be not good enough and the generator is unsuitable.

Table 4 shows $\mathbb{P}_T$ of sequences based on χ_{16HamG} using different functions, namely ⓐ,..., ⓔ. In this algorithm implementation, the embedded PRNG *Random* is the default Python PRNG, *i.e.*, the Mersenne Twister algorithm [Matsumoto & Nishimura, 1998]. Implementations for $\mathsf{N} = 4, \ldots, 8$ of this algorithm is evaluated through the NIST test suite and results are given in columns MT_4, ..., MT_8. If there are at least two statistical values in a test, this test is marked with an asterisk and the average value is computed to characterize the statistics.

We first can see in Table 4 that all the rates are greater than 97/100, *i.e.*, all the generators achieve to pass the NIST battery of tests. It can be noticed that adding chaos properties for Mersenne Twister algorithm does not reduce its security against this statistical tests.

8. Conclusion

This work has assumed a Boolean map f which is embedded into a discrete-time dynamical system G_f. This one is supposed to be iterated a fixed number p_1 or p_2,..., or p times before its output is considered. This work has first shown that iterations of G_f are chaotic if and only if its iteration graph $\Gamma_{\mathcal{P}}(f)$ is strongly connected where $\mathcal{P}$ is $\{p_1, \ldots, p\}$. It can be deduced that in such a situation a PRNG, which iterates G_f, satisfies the property of chaos and can be used in simulating chaos phenomena.

We then have shown that a previously presented approach can be directly applied here to generate function f with strongly connected $\Gamma_{\mathcal{P}}(f)$. The iterated map inside the generator is built by first removing from a N-cube a balanced Hamiltonian cycle and next by adding a self loop to each vertex. The PRNG can thus be seen as a random walk of length in $\mathcal{P}$ into this new N-cube. We have presented an efficient method to compute such a balanced Hamiltonian cycle. This method is an algebraic solution of an undeterministic approach [Suparta & Zanten, 2004] and has a low complexity. To the best of the authors knowledge, this is the first time a full automatic method to provide chaotic PRNGs is given. Practically speaking, this approach preserves the security properties of the embedded PRNG, even if it remains quite cost expensive.

We furthermore have presented an upper bound on the number of iterations that is sufficient to obtain an uniform distribution of the output. Such an upper bound is quadratic on the number of bits to output. Experiments have however shown that such a bound is in $\mathsf{N}.\log(\mathsf{N})$ in practice. Finally, experiments through the NIST battery have shown that the statistical properties are almost established for $\mathsf{N} = 4, 5, 6, 7, 8$ and should be observed for any positive integer N.

In future work, we intend to understand the link between statistical tests and the properties of chaos for the associated iterations. By doing so, relations between desired statistically unbiased behaviors and

topological properties will be understood, leading to better choices in iteration functions. Conditions allowing the reduction of the stopping-time will be investigated too, while other modifications of the hypercube will be regarded in order to enlarge the set of known chaotic and random iterations.

Acknowledgements

This work is partially funded by the Labex ACTION program (contract ANR-11-LABX-01-01). Computations presented in this article were realised on the supercomputing facilities provided by the Mésocentre de calcul de Franche-Comté.

References

Bahi, J., Couchot, J.-F., Guyeux, C. & Richard, A. [2011a] "On the link between strongly connected iteration graphs and chaotic boolean discrete-time dynamical systems," *FCT'11, 18th Int. Symp. on Fundamentals of Computation Theory* (Oslo, Norway), pp. 126–137.

Bahi, J., Fang, X., Guyeux, C. & Wang, Q. [2011b] "On the design of a family of CI pseudo-random number generators," *WICOM'11, 7th Int. IEEE Conf. on Wireless Communications, Networking and Mobile Computing* (Wuhan, China), pp. 1–4.

Banks, J., Brooks, J., Cairns, G. & Stacey, P. [1992] "On Devaney's definition of chaos," *Amer. Math. Monthly* **99**, 332–334.

Barker, E. & Roginsky, A. [2010] "Draft NIST special publication 800-131 recommendation for the transitioning of cryptographic algorithms and key sizes," .

Bhat, G. S. & Savage, C. D. [1996] "Balanced gray codes," *Electr. J. Comb.* **3**, URL http://www.combinatorics.org/Volume_3/Abstracts/v3i1r25.html.

Bykov, I. S. [2016] "On locally balanced gray codes," *Journal of Applied and Industrial Mathematics* **10**, 78–85.

Cao, L., Min, L. & Zang, H. [2009] "A chaos-based pseudorandom number generator and performance analysis," *Computational Intelligence and Security, 2009. CIS '09. International Conference on* (IEEE), pp. 494–498.

Couchot, J., Héam, P., Guyeux, C., Wang, Q. & Bahi, J. M. [2014] "Pseudorandom number generators with balanced gray codes," *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, eds. Obaidat, M. S., Holzinger, A. & Samarati, P. (SciTePress), ISBN 978-989-758-045-1, pp. 469–475.

Devaney, R. L. [1989] *An Introduction to Chaotic Dynamical Systems*, 2nd ed. (Addison-Wesley, Redwood City, CA).

Guyeux, C., Wang, Q. & Bahi, J. [2010] "Improving random number generators by chaotic iterations application in data hiding," *Computer Application and System Modeling (ICCASM), 2010 International Conference on* (IEEE), pp. V13–643–V13–647.

L'Ecuyer, P. & Simard, R. J. [2007] "TestU01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw* **33**.

Levin, D. A., Peres, Y. & Wilmer, E. L. [2006] *Markov chains and mixing times* (American Mathematical Society), URL http://scholar.google.com/scholar.bib?q=info:3wf9IU94tyMJ:scholar.google.com/&output=citation&hl=en&as_sdt=2000&ct=citation&cd=0.

Marsaglia, G. [1996] "Diehard: a battery of tests of randomness," *http://stat.fsu.edu/ geo/diehard.html* .

Matsumoto, M. & Nishimura, T. [1998] "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)* **8**, 3–30.

Mitzenmacher, M. & Upfal, E. [2005] *Probability and Computing* (Cambridge University Press).

Robinson, J. P. & Cohn, M. [1981] "Counting sequences," *IEEE Trans. Comput.* **30**, 17–23, URL http://dl.acm.org/citation.cfm?id=1963620.1963622.

Stojanovski, T. & Kocarev, L. [2001] "Chaos-based random number generators-part i: analysis [cryptography]," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on* **48**, 281–288.

Stojanovski, T., Pihl, J. & Kocarev, L. [2001] "Chaos-based random number generators. part ii: practical realization," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on* **48**, 382–385.

Suparta, I. & Zanten, A. v. [2004] "Totally balanced and exponentially balanced gray codes," *Discrete Analysis and Operation Research (Russia)* **11**, 81–98.

Wang, Q., Bahi, J., Guyeux, C. & Fang, X. [2010a] "Randomness quality of CI chaotic generators. application to internet security," *INTERNET'2010. The 2nd Int. Conf. on Evolving Internet* (IEEE Computer Society Press, Valencia, Spain), pp. 125–130, best Paper award.

Wang, Q., Bahi, J., Guyeux, C. & Fang, X. [2010b] "Randomness quality of CI chaotic generators. application to internet security," *INTERNET'2010. The 2nd Int. Conf. on Evolving Internet* (IEEE Computer Society Press, Valencia, Spain), pp. 125–130, best Paper award.