Mathematical and Computational Applications

MDPI

Review

Convergence versus Divergence Behaviors of Asynchronous Iterations, and Their Applications in Concrete Situations

Christophe Guyeux

Equipe Algorithmique Numérique Distribuée (AND), Departement Informatique des Systèmes Complexes (DISC), FEMTO-ST Institute, UMR 6174 CNRS, University Bourgogne Franche-Comté, F90000 Belfort, France; cguyeux@femto-st.fr

Received: 8 September 2020; Accepted: 12 October 2020; Published: 16 October 2020

Abstract: Asynchronous iterations have long been used in distributed computing algorithms to produce calculation methods that are potentially faster than a serial or parallel approach, but whose convergence is more difficult to demonstrate. Conversely, over the past decade, the study of the complex dynamics of asynchronous iterations has been initiated and deepened, as well as their use in computer security and bioinformatics. The first work of these studies focused on chaotic discrete dynamical systems, and links were established between these dynamics on the one hand, and between random or complex behaviours in the sense of the theory of the same name. Computer security applications have focused on pseudo-random number generation, hash functions, hidden information, and various security aspects of wireless sensor networks. At the bioinformatics level, this study of complex systems has allowed an original approach to understanding the evolution of genomes and protein folding. These various contributions are detailed in this review article, which is an extension of the paper "An update on the topological properties of asynchronous iterations" presented during the Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing (Pareng 2019).

Keywords: distributed computing; asynchronous iterations; theoretical modelling; chaos theory and applications

1. Introduction

In the context of distributed computing, serial and parallel approaches have very quickly shown their limitations, especially in terms of speed [1–3]. Various approaches have been proposed for almost half a century now, to avoid waiting times within processes, to distribute calculations more intelligently and on demand, to take into account the heterogeneity of resources, delais, etc. These modern approaches to distributing calculations in a collection of computation units often prove, in practice, to be able to solve a problem faster than more traditional approaches; but in theory, evidence of convergence (and speed of convergence) is generally more difficult to obtain. In this context, the mathematical formulation of the so-called asynchronous iterations has made it possible to establish numerous mathematical proofs of convergence, thus justifying the interest in these asynchronous approaches.

In parallel, during the last decade, a certain restricted class of asynchronous iterations has been studied in the context of the mathematical theory of chaos. Its topological behaviour has been examined, both theoretically and for its potential applications. At the theoretical level, the research was not limited to a particular class of discrete dynamic systems based on the mathematical theory of chaos. The "complexity" of the dynamics is generated, as a whole, with tools derived from probabilities and statistics, to evaluate its randomness, from the mathematical theory of chaos (to study, precisely,

its chaos, entropy, ergodicity, etc.), or even from that of complexity. The studied dynamical systems, for their part, can be variations of asynchronous iterations but also systems from computer science (sensor networks, neural networks, pseudorandom number generators, etc.) and from biology (protein folding, genomes evolution, etc.). The approach has then consisted of tracking down, modeling, and studying theoretically these complex dynamics that occur in biology and computer science, and to take benefits at the application level. These theoretical studies have had many practical implications, including information security, wireless sensor networks, and bioinformatics.

The purpose of this article is precisely to take up and make known the various advances, both theoretical and practical, concerning asynchronous iterations. The history of the research, both in terms of convergence and divergence, will be presented through significant events in the literature, and the richness of the disorderly nature of such iterations will be highlighted. Various applications previously proposed will then be discussed, in various fields, to illustrate the richness and potential of such tools. This article is an extension of an article accepted at the Pareng 2019 conference, entitled "An update on the topological properties of asynchronous iterations" [4]. Compared to the latter, we, first of all, enriched the theoretical part. Above all, a relatively substantial part has been added concerning the various applications of this theoretical work in various disciplines related to computer science and biology. This new part makes this article a complete one, containing the various interesting aspects related to asynchronous iterations.

This article is structured as follows. In the next section, various historical approaches, among the most significant in the theoretical study of the convergence of asynchronous iterations, will be recalled. These theoretical frameworks and convergence results will be put into perspective in Section 3, in which the opposite (disorder and divergence of such iterations) will be followed. An application section (Section 4) completes these theoretical presentations of the order and disorder of asynchronous iterations. This article will conclude with a discussion, in which the applications of such an approach to disorder will be discussed, and avenues for theoretical exploration will be proposed.

2. An Historical Perspective of the Convergence Study

2.1. Introduction the Asynchronous Iterations

Given $f : (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}} \to (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}}$, the asynchronous iterations [5] are defined as follow: $x^0 \in (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}}$, and

$$\forall n \in \mathbb{N}, \forall i \in [\![1, \mathsf{N}]\!], x_i^{n+1} \begin{cases} f(x^n)_i & \text{if } i \in s^n \\ x_i^n & \text{else.} \end{cases} \quad (1)$$

where $(s^n)_{n\in\mathbb{N}} \in \mathcal{P}([\![1, \mathsf{N}]\!])$ is a sequence of subsets of $\{1, 2, ..., \mathsf{N}\}$: $\mathcal{P}(X)$ here refers to all the parts of a set X, when $[\![a, b]\!]$ is the set of integers ranging from a to b; finally, the n-th term of a sequence u is written in exponent notation u^n, as this is a vector with N coordinates: $u_1^n, \ldots, u_{\mathsf{N}}^n$. Asynchronous iterations have provided, for decades, a mathematical framework for finding advanced algorithm schemas in distributed computing [6]: roughly speaking, the coordinates of the vector x^n correspond to the calculation units, the function f is the equivalent of the calculation to be distributed, and the sequence s is the way to distribute these calculations: the sequence $\{1, 2, \ldots, \mathsf{N}\}, \{1, 2, \ldots, \mathsf{N}\}, \{1, 2, \ldots, \mathsf{N}\}, \ldots$ corresponding to a parallel calculation, when $\{1\}, \{2\}, \ldots \{\mathsf{N}\}, \{1\}, \{2\}, \ldots$ is a serial calculation, for example.

This mathematical formulation of asynchronous distributed algorithms has made it possible to establish various theoretical frameworks, in which proof of convergence and convergence rate calculations could be established. As an illustrative example, three special cases of Definition 1 will be recalled in the next section, which have a significant theoretical and practical interest. These are Chazan and Miranker's historical approach, asynchronous memory iterations, and Bertsekas' model (Note that in the last 20 years, largely influenced by the problems of the theory of linear asynchronous systems with discrete time, the so-called joint spectral radius theory has been developed. Within the framework of this theory, many questions of the convergence of linear asynchronous iterations and the

stability of asynchronous systems have received theoretical development, see, e.g., [7]). Situations for which we can be sure of the convergence of these models will be presented, before discussing the problem of the termination of algorithms. We will recall how some of these iterations were studied in applied mathematics. The purpose of these studies is above all to seek situations of convergence of iterations on a given system, by considering additional assumptions specific to computer models, i.e., mainly finiteness constraints.

Taking a direction diametrically opposed to these convergence efforts, we will then discuss that the study of chaos of such asynchronous iterations, described as discrete dynamic systems, has been initiated this last decade, and studied in more depth in various publications that followed the founding article of [8].

2.2. Three Historical Models

The iterations considered in this manuscript have been studied for more than fifty years, both in terms of their convergence and their applications [9,10]. In this section, rather than being exhaustive, we have chosen to arbitrarily present three models that have had a significant historical impact.

2.2.1. The Chazan and Miranker Model

The first theoretical work on asynchronous algorithms dates back to 1969, it focused on linear system resolutions.

The interest of asynchronism when resolving such systems, using several computers that can communicate with each other, is as follows. In an asynchronous iterative algorithm, the components of the iterated vector are updated in parallel, without any a priori order or synchronization between the machines. The asynchronous implementation allows better recovery of communications by calculations (no downtime due to synchronizations) and allows failures (temporary or permanent). Asynchronism also has the advantage of better adaptability to changes in the computing infrastructure, such as changes in topology: changes in the number of computers, or in the possible communications between them.

This model, based on the work of Chazan and Miranker [11], Miellou [12], and Baudet [13], has been formalized on the following manner:

Definition 1 (Chazan and Miranker model)**.** *Let $\mathcal{X} = \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_\alpha}$. The asynchronous iterations associated to $F : \mathcal{X} \to \mathcal{X}$, with initial condition $x^0 \in \mathcal{X}$, correspond to the sequence $x \in \mathcal{X}^{\mathbb{N}}$ defined by:*

$$x_i^t = \begin{cases} F_i\left(x_1^{I_1^t}, \ldots, x_\alpha^{I_\alpha^t}\right) & \text{if } i \in S^t \\ x_i^{t-1} & \text{if } i \notin S^t \end{cases}$$

where x_i is the sub-vector of x on $\mathbb{R}^{n_i}$, F_i is the i-th block-component of F, and $\forall t \in \mathbb{N}^*$, $S^t \subset [\![1;\alpha]\!]$ and $I^t \in \mathbb{N}^\alpha$.

In other words, at iteration t, the value of the i-th block-component x_i is either the value of x_i at iteration $t-1$, or the mapping $F_i\left(x_1^{I_1^t}, \ldots, x_\alpha^{I_\alpha^t}\right)$, in which the current component blocks x_i^t are not taken into account, but one of their previous values $x_i^{I_i^t}$, i.e., the component block that the system had at the time I_i^t. This approach allows for very general transmission delays to be taken into account. Finally, the S^t sequence indicates which cell blocks should be updated at time t. It can be seen that this model of delay iterations is a special case of Definition 1.

In addition, it is assumed in this model that the S and I sequences test the following assumptions:

Hypothesis 1 (H1). *The values of the iterated vector used in the calculations at the iteration t come at best from the iteration $t-1$ (notion of delay in transmission): $\forall t \in \mathbb{N}^*$, $I_i^t \leqslant t-1$.*

Hypothesis 2 (H2). *$I_i^t \to +\infty$, when $t \to +\infty$: the too old values of the components of the iterated vector must be definitively discarded as the calculations progress.*

Hypothesis 3 (H3). *No subvector stops being updated (so-called pseudo-periodic strategies). In other words, t appears an infinite number of times in S.*

This a specific framework of iterations, but the purpose remains relatively general: blocks are considered rather than components; real numbers are manipulated; and delays are taken into account, which depend on the blocks from which the information comes. The only constraints are that no component of the iterated vector should cease to be permanently updated, and that the values of the components associated with too old iterations should cease to be used as the calculations progress. It should be noted, to finish with the introduction of this model, that the above Hypotheses H2 and H3 find their natural justification in the fact that the initiators of this theory were exclusively seeking the convergence of asynchronous iterations.

2.2.2. Asynchronous Iterations with Memory

Asynchronous iterations with memory use at each iteration several values of each component of the iterated vector, which may be related to different iteration numbers. This gives the following definition:

Definition 2 (Asynchronous iterations with memory)**.** *Let $n \in \mathbb{N}, \alpha \in [\![0; n]\!]$, and the following decomposition of $\mathbb{R}^n$: $\mathcal{X} = \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_\alpha}$, where $\sum_{i=1}^{\alpha} n_i = n$. Let $F : \mathcal{X}^m \to \mathcal{X}$, where $m \in \mathbb{N}^*$. One asynchronous iteration with $m-1$ memories associated to the application F and to the subset Y of the m first vectors $\{x^0, x^1, \ldots, x^{m-1}\}$, is a sequence $\left(x^j\right)_{j \in \mathbb{N}}$ of vectors of $\mathcal{X}$ such that, for $i \in [\![1; \alpha]\!]$ and $j \geqslant m$, we have:*

$$\begin{cases} x_i^j = F_i\left(z^1, \ldots, z^m\right) & \text{if } i \in S^j \\ x_i^j = x_i^{j-1} & \text{if } i \notin S^j \end{cases}$$

where $\forall r \in [\![1; m]\!], z^r$ is the vector constituted by the subvectors $z_l^r = x_l^{I_l^r(j)}$, $\left(S^j\right)_{j \geqslant m}$ is a sequence of non-empty subsets of $[\![1; \alpha]\!]$, and $I = \left\{I_1^1(j), \ldots I_\alpha^1(j), I_1^2(j), \ldots I_\alpha^2(j), I_1^m(j), \ldots I_\alpha^m(j) \ / \ j \geqslant m\right\}$ is a sequence of $[\mathbb{N}^\alpha]^m$.

In addition, S and I satisfy the following conditions:

- $\max_{r \in [\![1;m]\!]} \left\{I_i^r(j) \ / \ r \in [\![1; m]\!]\right\} \leqslant j - 1$, *for all $j \geqslant m$;*
- $\min_{r \in [\![1;m]\!]} \left\{I_i^r(j) \ / \ r \in [\![1; m]\!]\right\}$ *tends to infinity when j tends to infinity; and*
- *i appears an infinite number of times in S.*

These iterations, which are also a special case of Definition 1, have been studied by Miellou [14], El Tarazi [15] and Baudet [13]. Asynchronous iterations with memory have been proposed to deal with the case where an application, whose fixed point is searched by a classical method, is not explicitly defined. They allow us to assign some processors to intermediate function calculations, while the others are in charge of updating the components of the iterated vector [16].

2.2.3. Bertsekas Model

The Bertsekas model of fully asynchronous iterations differs significantly from the two models presented above. By introducing another formulation of these objects, it makes it possible to better understand their nature, in particular by allowing them to be categorized by classes.

Let:

- $\mathcal{X}_1, \ldots, \mathcal{X}_n$ some sets, and $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$.
- $F_i : \mathcal{X} \to \mathcal{X}_i$ some functions, and $F(x) = (F_1(x), \ldots, F_n(x))$ defined from $\mathcal{X}$ to $\mathcal{X}$.

The Bertsekas model of totally asynchronous algorithms, another special case of the Definition 1, is the following [16].

Definition 3 (Bertsekas model)**.** *We assume that there is a sequence of events $T = \{0, 1, 2, \ldots\}$ for which one or more components x_i of the iterative vector x are updated by one of the processors of the parallel or distributed architecture.*

Let T^i be the sub-series of events for which x_i is updated. We assume too that the processor updating x_i may not have access to the most recent values of the x components, and for any $t \notin T^i$, x_i remains unchanged.

An asynchronous iteration of the Bertsekas model is a x^t sequence of vectors of $\mathbb{R}^n$ such that for any $i \in [\![1; n]\!]$:

$$\begin{cases} x_i^{t+1} = F_i\left(x_1\left(\tau_1^i(t)\right), \ldots, x_n\left(\tau_n^i(t)\right)\right) & \text{if } t \in T^i \\ x_i^{t+1} = x_i^t & \text{if } t \notin T^i \end{cases}$$

where $\tau_l^i(t) \in [\![0; t]\!], \forall l \in [\![1; n]\!], \forall t \in T$.

The T elements are the indices of the sequence of moments at which the updates are delivered. The difference $t - \tau_l^i(t)$, on the other hand, represents the delay in accessing the i-th component of the iterated vector, when updating the i-th component at the moment t [16].

For the model to be complete, one of the following two assumptions regarding calculations and communications must be added to the above definition:

- **Hypothesis of total asynchronism.** The T^i sets are infinite. Moreover, if t^k is a sub-series of elements of T^i that tends to infinity, then $\lim_{k \to +\infty} \tau_l^i(t_k) = +\infty$, for all $l \in [\![1; n]\!]$.
- **Partial asynchronism hypothesis.** There is a positive integer B, called asynchronism character, such as:

  1. For any $i \in [\![1; n]\!]$, and for any t, at least one element of the set $[\![t; t + B - 1]\!]$ belongs to T^i: each component is refreshed at least once during an interval containing B refreshes.
  2. $t - B < \tau_l^i(t) \leqslant t, \forall i, l \in [\![1; n]\!]$, and $t \in T^i$: the information used to update a component has a maximum delay of B.
  3. $\tau_i^i(t) = t$, $\forall i \in [\![1; n]\!]$, $\forall t \in T^i$: when updating the component assigned to it, each processor uses the last value of the same component.

Partially asynchronous iterations were introduced by Bertsekas and Tsitsiklis in [17]. They are less general than totally asynchronous iterations: markers are placed on the delays and the duration of the interval between two consecutive updates of the same component. However, they may be of great interest when excessive asynchronism leads to divergence, or does not guarantee convergence.

The use of asynchronous iterative algorithms raises two types of problems: establishing their convergence, and ensuring the termination of algorithms. These problems are reviewed in the next two sections.

2.3. On the Usefulness of Convergence Situations

Convergence in the asynchronous model is more difficult to achieve than in the synchronous model due to the lack of synchronization, and therefore the less regular behaviour of iterative patterns. However, a number of general results could be established. They are for various contexts: (non-linear) systems, fixed point problems, etc.

2.3.1. Case of Equation Systems

The first convergence result, published by Chazan and Miranker in 1969 [11], is a necessary and sufficient condition for the convergence of the asynchronous iterations, as part of the resolution of linear systems. It requires the definition of $H-$matrices:

Definition 4 ($H-$matrices). *A matrix N is a $H-$matrix if the $\tilde{N}$ matrix consisting of diagonal elements of N minus the absolute value of non-diagonal elements, is a matrix such that its diagonal coefficients are strictly positive, its non-diagonal elements are negative or null, and the opposite of $\tilde{N}$ exists and has its positive coefficients.*

The necessary and sufficient condition for convergence for linear systems can then be stated [11]:

Proposition 1. *Let us say the system of equations $\mathcal{A}x^* = z$, where x^* and z are two vectors of $\mathbb{R}^n$. Then any asynchronous algorithm defined by the Chazan and Miranker model, where F then corresponds to a Jacobi type matrix per point, converges towards the solution of the problem if and only if $\mathcal{A}$ is a $H-$matrix.*

Various sufficient conditions have since been set out in specific frameworks of equation systems, both linear and non-linear [10], for the various asynchronous iteration models mentioned above. Such results can be found in [18,19], or [16].

2.3.2. Fixed Point Problems

In the same vein, various convergence results for asynchronous iterations applied to fixed point problem solving have been obtained [9]. One of the most remarkable results is related to the contraction of the function, and is stated as follows [12]:

Proposition 2. *Let be E a reflexive Banach space finished product of a family of Banach spaces $(E_i, ||.||_i)$, $i \in [\![1; \alpha]\!]$. Let us denote by $\varphi(x) = (||x_1||_1, \ldots, ||x_\alpha||_\alpha)$ the canonical vectorial norm of E. Let $F : D(F) \to D(F)$ a function, where $D(F) \subset E$ is non-empty. If*

- *F has a fixed point in $x^* \in D(F)$,*
- *and F is contracting in x^* for the vectorial norm φ,*

Then the asynchronous iterative algorithm defined by the Chazan and Miranker model belongs to $D(F)$, and converges to x^*.

This result has been extended to fixed point applications with relaxation parameter [15,20], asynchronous iterative algorithms with memory in a context of classical contraction [20], and partial order [21]. Other classic results can be found in [14,22].

2.3.3. Bertsekas' Asynchronous Convergence Theorem

Bertsekas' theorem provides a set of sufficient conditions for the convergence of asynchronous algorithms for fixed point problems. This result, which is based on Lyapunov's theory of stability, is based on the study of a series of sets to which the elements of the iterated vector suite belong. Its advantage is that it provides a more abstract framework for the analysis of the convergence of asynchronous iterations, which includes in particular the contraction and partial order aspects.

Proposition 3 (Asynchronous convergence of Bertsekas [23]). *Let $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$ a cartesian product of sets. Suppose there is a series of non-empty subsets X^j of $\mathcal{X}$, increasing for inclusion, such as $\forall j$, there are sets $X_i^j \subset \mathcal{X}_i$ checking $X^j = X_1^j \times \ldots \times X_n^j$. Let us assume too that:*

- *$F(x) \subset X^{j+1}, \forall j, \forall x \in X^j$,*
- *if x is a sequence of vectors such as $x^j \in X^j$ for all j, then x tends to a fixed point of F.*

Under these conditions, and if $x^0 \in X^0$, then any fully asynchronous iteration defined according to the Bertsekas model converges to a fixed point of F.

Bertsekas and his team used this theorem to obtain convergence results for various asynchronous algorithms applied to solving a wide variety of problems: dynamic programming, search for minimum paths, optimization problems, network flow, optimal routing, etc. Other convergence results can be found in the literature. Thus, Lubachewski and Mitra have established a sufficient convergence result for asynchronous iterations with bounded delays applied to the resolution of singular Markovian systems [24]. Finally, Frommer and Szyld studied the so-called multisplitting and asynchronous decomposition methods [25,26].

2.4. The Problem of Algorithm Termination

The final termination of the iterative algorithm must occur when the iterated vector is sufficiently close to a solution to the problem, and a special procedure must be designed to detect this termination. Since the number of iterations of the algorithm can be infinite, the calculation processes can never be inactive. There are relatively few effective termination methods for asynchronous algorithms. Indeed, termination presents many difficulties, especially in cases where processors do not share a global clock, or where communication delays can be arbitrarily long.

The most frequently used termination methods are designed empirically. For example, a particular processor can be assigned the task of observing the local termination conditions in each processor: the algorithm terminates when all local conditions are met. This approach is functional in the unique case where the degree of asynchronism is very low. Other empirical methods are possible. For example, Bertsekas and Tsitsiklis proposed in [23] that each processor sends termination and restart messages to a central processor in charge of these problems. The Chajakis and Zenios [27] method does not require a central processor: a processor completes its calculations if its local termination condition is satisfied, and if it has received termination messages from other processors, and acknowledgments of all its termination messages. No termination method has been formally validated in the most general case, or almost in the most general case: the Bertsekas and Tsitsiklis solution is one of the few with formal validity. However, this method has a number of disadvantages: complex protocol, many communications, restrictive convergence conditions, etc.

As we can see, the problems of convergence of asynchronous iterations and their applications have been widely studied over the past fifty years. The inverse problem of the divergence of these iterations has been studied more recently, over the past decades, and has also proved to be rich in applications. The formal framework and these applications are the subjects of the following two sections.

3. Theoretical Foundations of the Divergence

3.1. Asynchronous Iterations as a Dynamical System

In the absence of "delay", asynchronous iterations can be rewritten as a recurring sequence on the product space $\mathcal{X} = (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}} \times \mathcal{P}([\![1, \mathsf{N}]\!])^{\mathbb{N}}$, consisting of the calculated vectors on the one hand, and the series of components to be updated on the other hand. If you enter the functions $i : \mathcal{P}([\![1, \mathsf{N}]\!])^{\mathbb{N}} \to \mathcal{P}([\![1, \mathsf{N}]\!])$, $(s^n)_{n\in\mathbb{N}} \longmapsto s^0$, producing the first subset of the sequence s and $\sigma : \mathcal{P}([\![1, \mathsf{N}]\!])^{\mathbb{N}} \to \mathcal{P}([\![1, \mathsf{N}]\!])^{\mathbb{N}}$, $(s^n)_{n\in\mathbb{N}} \longmapsto (s^{n+1})_{n\in\mathbb{N}}$, performing a shift to head the list, then the asynchronous iterations of the Equation (1) are rewritten as a discrete dynamical system G_f on $\mathcal{X}$: $X^0 \in \mathcal{X}$, et $\forall n \in \mathbb{N}$,

$$\begin{aligned} X^{n+1} &= \left(F_f(X_1^n, i(X_2^n)); \sigma(X_2^n)\right) \\ &= G_f(X^n), \end{aligned} \tag{2}$$

where

$$F_f : (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}} \times \mathcal{P}([\![1, \mathsf{N}]\!]) \longrightarrow (\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}}, (x, e) \longmapsto (x_i \mathcal{X}_e(i) + f(x)_i \overline{\mathcal{X}_e(i)})_{i\in[\![1,\mathsf{N}]\!]},$$

with $\mathcal{X}_X$ as the characteristic function of the set X and $\overline{x} = x + 1 \ (mod\ 2)$.

Finally, a relevant distance can be introduced on $\mathcal{X}$, as follows [28]:

$$d((S,E);(\check{S};\check{E})) = d_e(E,\check{E}) + d_s(S,\check{S})$$

where $d_e(E,\check{E}) = \sum_{k=1}^{\mathsf{N}} \delta(E_k,\check{E}_k)$, δ being the Hamming distance, and

$$d_s(S,\check{S}) = \frac{9}{\mathsf{N}} \sum_{k=1}^{\infty} \frac{|S^k - \check{S}^k|}{10^k}.$$

With such a distance, we can state that [8]:

Proposition 4. *$G_f : (\mathcal{X},d) \to (\mathcal{X},d)$ is a continuous map.*

Asynchronous iterations had until now been studied with discrete mathematics. Such a rewrite therefore makes it possible to study them with the tools of mathematical analysis. This is all the more relevant since the results of distributed computation algorithms are usually fixed points, and mathematical analysis contains various frameworks for studying fixed points of dynamical systems. Note that we iterate on a topological space composed only of integers, when the associated algorithms manipulate machine numbers: the theoretical framework of study is exactly that of practical applications. We can also, through a topological semi-conjugation, reduce these asynchronous iterations to a simple dynamic system over an interval of $\mathbb{R}$, but the inherited topology is not that of the order [8]. This rewriting of asynchronous iterations in the form of discrete dynamic systems has allowed us to study their dynamics using mathematical analysis tools: mathematical topology, and more recently measurement theory for ergodicity concepts.

In what follows, we will first recall the key concepts of the study of the disorder and randomness of discrete dynamic systems, and then we will see to what extent asynchronous iterations can exhibit such dynamics.

3.2. The Mathematical Theory of Chaos

3.2.1. Notations and Terminologies

Let us start by introducing the usual notations in discrete mathematics, which may differ from those found in the study of discrete dynamic systems. The n-th term of the sequence s is denoted by s^n, the i-th component of vector v is v_i, and the k-th composition of function f is denoted by f^k. Thus $f^k = f \circ f \circ \ldots \circ f$, k times. The derivative of f is f', while $\mathcal{P}(X)$ is the set of subsets of X. $\mathbb{B}$ stands for the set $\{0;1\}$ with its usual algebraic structure (Boolean addition, multiplication, and negation), while $\mathbb{N}$ and $\mathbb{R}$ are the notations of the natural numbers and real ones. $\mathcal{X}^{\mathcal{Y}}$ is the set of applications from $\mathcal{Y}$ to $\mathcal{X}$, and so $\mathcal{X}^{\mathbb{N}}$ means the set of sequences belonging in $\mathcal{X}$. $\lfloor x \rfloor$ stands for the integral part of a real x (the greatest integer lower than x). Finally, $[\![a;b]\!] = \{a, a+1, \ldots, b\}$ is the set of integers ranging from a to b.

With these notations in place, we are now able to introduce various classical notions of disorder or randomness for discrete dynamic systems.

3.2.2. Devaney-Based Approaches

In these approaches, three ingredients are necessary for unpredictability [29]. First, the system must be inherently complicated, indecomposable: it cannot be simplified into two systems. Subsystems that do not interact, allowing a divide and conquer strategy to be adopted applied to the system is ineffective. In particular, many orbits must visit the entire space. Second, an element of regularity is added, to offset the effects of inflation. The effects of the first ingredient, leading to the

fact that closed points can behave in a completely different way, and this behavior can not be predicted. Finally, system sensitivity is required as a third ingredient, so that close points can eventually become distant during system iterations. This last requirement is often implied by the first two ingredients. Having this understanding of an unpredictable dynamic system, Devaney formalized in the following definition of chaos.

Definition 5. *A discrete dynamical system $x^0 \in \mathcal{X}, x^{n+1} = f(x^n)$ on a metric space $(\mathcal{X}, d)$ is chaotic according to Devaney if:*

1. *Transitivity: For each couple of open sets $A, B \subset \mathcal{X}$, $\exists k \in \mathbb{N}$ s.t. $f^k(A) \cap B \neq \varnothing$.*
2. *Regularity: periodic points are dense in $\mathcal{X}$.*
3. *Sensitivity to the initial conditions: $\exists \varepsilon > 0$ s.t.*

$$\forall x \in \mathcal{X}, \forall \delta > 0, \exists y \in \mathcal{X}, \exists n \in \mathbb{N}, d(x,y) < \delta \text{ and } d(f^n(x), f^n(y)) \geqslant \varepsilon.$$

With regard to the sensitivity ingredient, it can be reformulated as follows.

- $(\mathcal{X}, f)$ is unstable if all its points are unstable: $\forall x \in \mathcal{X}$, $\exists \varepsilon > 0$, $\forall \delta > 0$, $\exists y \in \mathcal{X}$, $\exists n \in \mathbb{N}$, $d(x,y) < \delta$ and $d(f^n(x), f^n(y)) \geqslant \varepsilon$.
- $(\mathcal{X}, f)$ is expansive if $\exists \varepsilon > 0, \forall x \neq y, \exists n \in \mathbb{N}, d(f^n(x), f^n(y)) \geqslant \varepsilon$

The system can be intrinsically complicated too for various other understandings of this desire, which are not equivalent to each other, such as:

- Topological mixing: for all pairs of open disjointed sets that are not empty U, V, $\exists n_0 \in \mathbb{N}$ s.t. $\forall n \geqslant n_0, f^n(U) \cap V \neq \varnothing$.
- Strong transitivity: $\forall x, y \in \mathcal{X}, \forall r > 0, \exists z \in B(x,r), \exists n \in \mathbb{N}, f^n(z) = y$.
- Total transitivity: $\forall n \geqslant 1$, the composition f^n is transitive.
- Undecomposable: it is not the union of two closed, non-empty subsets that are positively invariant ($f(A) \subset A$).

These various definitions lead to various notions of chaos. For example, a dynamic system is chaotic according to Wiggins if it is transitive and sensitive to initial conditions. It is said to be chaotic according to Knudsen if it has a dense orbit while being sensitive. Finally, we speak of expansive chaos when the properties of transitivity, regularity and expansiveness are satisfied.

3.2.3. Approach from Li and Yorke

The approach to chaos presented in the previous section, considering that a chaotic system is an inherently complicated (non-decomposable) system, with a possibly of an element of regularity and/or sensitivity, has been supplemented by another understanding of chaos. Indeed, as "randomness" or "infiniteness", it is impossible to find a single universal definition of chaos. The types of behaviours we are trying to describe are too complicated to fit into a single definition. Instead, a wide range of mathematical descriptions have been proposed over the past decades, all of which are theoretically justified. Each of these definitions illustrates specific aspects of chaotic behaviour.

The first of these parallel approaches can be found in the pioneering work of Li and Yorke [30]. In their famous article entitled "The Third Period Involves Chaos", they rediscovered a weaker formulation of Sarkovskii's theorem, which means that when a discrete dynamic system $(f, [0.1])$, with continuous f, has a cycle 3, then it also has a cycle n, $\forall n \leqslant 2$. The community has not adopted this definition of chaos, as several degenerate systems satisfy this property. However, on their article [30], Li and Yorke also studied another interesting property, which led to a notion of chaos "according to Li and Yorke" recalled below.

Definition 6. *Let $(\mathcal{X}, d)$ a metric space and $f : \mathcal{X} \longrightarrow \mathcal{X}$ a continuous map on this space. $(x, y) \in \mathcal{X}^2$ is a scrambled couple of points if $\liminf_{n\to\infty} d(f^n(x), f^n(y)) = 0$ and $\limsup_{n\to\infty} d(f^n(x), f^n(y)) > 0$ (in other words, the two orbits oscillate each-other).*

A scrambled set is a set in which any couple of points are a scrambled couple, whereas a Li–Yorke chaotic system is a system possessing an uncountable scrambled set.

3.2.4. Lyapunov Exponent

The next measure of chaos that will be considered in this document is the Lyapunov exponent. This quantity characterizes the rate of separation of the trajectories infinitely close. Indeed, two trajectories in the phase space with initial separation δ diverge at a rate approximately equal to $\delta e^{\lambda t}$, where λ is the exponent Lyapunov, which is defined by:

Definition 7. *Let $x^0 \in \mathbb{R}$ and $f : \mathbb{R} \longrightarrow \mathbb{R}$ be a differentiable function. The Lyapunov exponent is defined by* $\lambda(x^0) = \lim_{n\to+\infty} \frac{1}{n} \sum_{i=1}^{n} \ln \left| f'\left(x^{i-1}\right)\right|$.

Obviously, this exponent must be positive to have a multiplication of initial errors by an exponentially increasing factor, and therefore be in a situation of chaos according to this formulation.

3.2.5. Topological Entropy

Let $(\mathcal{X}, d)$ a compact metric space and $f : \mathcal{X} \longrightarrow \mathcal{X}$ a continuous map for this space. $\forall n \in \mathbb{N}$, a new distance d_n is defined on $\mathcal{X}$ by

$$d_n(x, y) = \max\{d(f^i(x), f^i(y)) : 0 \leq i < n\}.$$

With $\varepsilon > 0$ and $n \geqslant 1$, two points of $\mathcal{X}$ are ε closed compared to this measure if their first n iterates are ε closed. This measurement makes it possible to distinguish in the vicinity of an orbit the points that move away from each other during the iteration of the points that travel together. A subset E of $\mathcal{X}$ is said to be (n, ε)-separated if each pair of distinct points of E is at least ε separated in the metric d_n. Indicates by $N(n, \varepsilon)$ the maximum cardinality of a separate set (n, ε). $N(n, \varepsilon)$ represents the number of distinct orbit segments of length n, assuming that we cannot distinguish the points in ε from each other.

Definition 8. *The topological entropy of the map f is equal to*

$$h(f) = \lim_{\epsilon\to 0} \left(\limsup_{n\to\infty} \frac{1}{n} \log N(n, \epsilon) \right).$$

The limit defining $h(f)$ can be interpreted as a measure of the average exponential growth of the number of distinct orbit segments. In this sense, it measures the complexity of the dynamical system $(\mathcal{X}, f)$.

3.3. The Disorder of Asynchronous Iterations

The topological space over which asynchronous iterations are defined was first studied, leading to the following result [28]:

Proposition 5. *$\mathcal{X}$ is an infinitely countable metric space, being both compact, complete, and perfect (each point is an accumulation point).*

These properties are required in a specific topological formalisation of a chaotic dynamic system, justifying their proof. Concerning G_{f_0}, it was stated that [28].

Proposition 6. *G_{f_0} is surjective, but not injective, and so the dynamical system $(\mathcal{X}, G_{f_0})$ is not reversible.*

It is now possible to recall the topological behavior of asynchronous iterations.
We have firstly stated that [28]:

Theorem 1. *G_{f_0} is regular and transitive on $(\mathcal{X}, d)$, so it is chaotic as defined by Devaney. In addition, its sensitivity constant is greater than $\mathsf{N} - 1$.*

Thus the set $\mathcal{C}$ of functions $f : \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$ making asynchronous iterations of Definition 2 a case of chaos according to Devaney, is a not empty set. To characterize the functions of $\mathcal{C}$, we first stated that transitivity implies regularity for these particular iterated systems [31].

To achieve characterization, the function F_f allows us to define a graph Γ_f, where the vertices are the vectors of $\mathbb{Z}/2\mathbb{Z}$, and there is a ridge labeled $s \in \mathcal{P}(\llbracket 1, \mathsf{N} \rrbracket)$ from i to j if, and only if $F_f(i, s) = j$. We have shown that the properties of the dynamic system G_f are strongly related to those of the graph $\mathcal{G}_f$. Thus, for example, if the latter is strongly related, then the asynchronous iterations are highly transitive and regular, and therefore chaos in Devaney's mathematical sense. Other properties, such as topological entropy, expansiveness, or sensitivity to initial conditions, defined in topological terms, could also be studied. On the other hand, the subsets of $\llbracket 1, \mathsf{N} \rrbracket$ can be drawn according to a certain probability distribution, which allows us to study the associated Markov chain (ergodicity, mixing time, etc.) These various disorder results are presented below [31].

Theorem 2. *G_f is transitive, and thus chaotic according to Devaney, if and only if $\Gamma(f)$ is strongly connected.*

This characterization allows to quantify the number of functions in $\mathcal{C}$: it is equal to $\left(2^{\mathsf{N}}\right)^{2^{\mathsf{N}}}$. Then, the study of the topological properties of the disorder of these iterative systems was the subject of a more detailed study which led to the following results.

Theorem 3. *$\forall f \in \mathcal{C}$, Per $\left(G_f\right)$ is infinitely countable, G_f is strongly transitive and is chaotic according to Knudsen. It is thus undecomposable, unstable, and chaotic, as defined by Wiggins.*

Theorem 4. *$\left(\mathcal{X}, G_{f_0}\right)$ is topologically mixing, expansive (with a constant equal to 1), chaotic as defined by Li and Yorke, and has a topological entropy and an exponent of Lyapunov both equal to $ln(\mathsf{N})$.*

At this stage, a new type of iterative systems that only handle integers has been discovered, leading to the questioning of their computing for security applications. The applications of these chaotic machines are presented in the following section.

4. Applications of the Divergence

The theoretical developments around the disorder of asynchronous iterations, which had never been examined before, has led to interesting applications of such complex dynamics in various domains of computer security like hash functions [32] and digital watermarking [33]. Since then, we have broadened our field of investigation both theoretically and in terms of applications. The dynamics studied in this framework can also derive from computers (sensor networks, neural networks, pseudo-random number generators, etc.) or biology (protein folding, genome evolution, etc.). Some of these applications are detailed hereafter.

4.1. Some Theoretical Developments

In [8], authors explained how to design finite state machines with truly chaotic behavior. The idea is to decompartmentalize the machine, and to use at each iteration the values provided to it at the input to calculate the value to be produced at the output. By this process, even if the machine is finite-state,

it does not always enter a loop, since the input is not necessarily periodic. It has been formalized with Turing machines whose behavior is chaotic, in the sense that the effect of a slight alteration of the input–output tape provided to the machine cannot be predicted [34,35]. Since then, we have continued to study these chaotic Turing machines, proposing in particular, a characterization of chaotic Moore automatons, and deepening applications concerning steganography [36,37] and digital watermarking [33,38,39], hash functions [32,40], and the generation of pseudo-random numbers [41,42], see below. Each time, the "computer security machine" receives input data: an image to be encrypted, a video stream to be watermarked, a pseudo-random sequence to be post-operated, etc. We can therefore ensure that this processing is chaotic in the mathematical sense of the term, and that an opponent cannot predict what will be the hashed value, the watermarked media, the next bit of the generator, etc., knowing the past behavior of the machine. These applications have been deepened and deepened over the past decade.

A final work concerning the theoretical study of chaotic finite state machines consisted in effectively constructing chaotic neural networks, on the one hand, [43], and in showing on the other hand that it was possible to prove that a neural network was (or not) chaotic [44]: these are only finite state machines that receive new inputs at each iteration, and whose outputs may or may not be predicted according to the complexity of the dynamics generated by the associated iterative system. Finally, artificial intelligence tools play an important role in some branches of computer security such as steganalysis [45]: the detection of the presence of secret information within images is indeed done by support vector machines or neural networks that learn to distinguish natural images from steganographied images, and must then detect the presence of suspicious images in a given channel. We have shown that multilayer perceptrons (neural networks) are unable to learn from truly chaotic dynamics, and have shown as an application that steganalyzers can be faulted using chaotic concealment methods [45].

Let us now go into more detail in one of these applications. We have chosen the one that has so far produced the best results: the generation of randomness from chaos.

4.2. Asynchronous Iterations as Randomness Generators

4.2.1. Qualitative Relations between Topological Properties and Statistical Tests

Let us first explain why we have reasonable ground to believe that chaos can improve the statistical properties of inputs. We will show in this section that chaotic properties, as defined in the mathematical theory of chaos, are related to some statistical tests that can be found in the NIST battery of tests [46]. We will verify later in this section that, when mixing defective pseudorandom number generators (PRNGs) with asynchronous iterations, the new generator presents better statistical properties (this section summarizes and extends the work of [47]).

There are various relations between topological properties that describe an unpredictable behavior for a discrete dynamical system on the one hand, and statistical tests to check the randomness of a numerical sequence on the other hand. These two mathematical disciplines follow a similar objective in case of a recurrent sequence (to characterize an intrinsically complicated behavior), with two different but complementary approaches. It is true that the following illustrative links give only qualitative arguments, and proofs should be provided to make such arguments irrefutable. However they give a first understanding of the reason why chaotic properties tend to improve the statistical quality of PRNGs, which is experimentally verified as shown at the end of this section. Let us now list some of these relations between topological properties defined in the mathematical theory of chaos and tests embedded into the NIST battery.

- Regularity. As recalled earlier in this article, a chaotic dynamical system must have an element of regularity. Depending on the chosen definition of chaos, this element can be the existence of a dense orbit, the density of periodic points, etc. The key idea is that a dynamical system with no periodicity is not as chaotic as a system having periodic orbits: in the first situation, we can

predict something and gain knowledge about the behavior of the system, that is, it never enters into a loop. Similar importance for periodicity is emphasized in the two following NIST tests [46]:

  - Non-overlapping Template Matching Test. Detect the production of too many occurrences of a given non-periodic (aperiodic) pattern.
  - Discrete Fourier Transform (Spectral) Test. Detect periodic features (i.e., repetitive patterns that are close one to another) in the tested sequence that would indicate a deviation from the assumption of randomness.

- Transitivity. This topological property previously introduced states where the dynamical system is intrinsically complicated: it cannot be simplified into two subsystems that do not interact, as we can find in any neighborhood of any point another point whose orbit visits the whole phase space. This focus on the places visited by the orbits of the dynamical system takes various nonequivalent formulations in the mathematical theory of chaos, namely: transitivity, strong transitivity, total transitivity, topological mixing, and so on. Similar attention is brought on the states visited during a random walk in the two tests below [46]:

  - Random Excursions Variant Test. Detect deviations from the expected number of visits to various states in the random walk.
  - Random Excursions Test. Determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence.

- Chaos according to Li and Yorke. We recalled that two points of the phase space (x, y) define a couple of Li–Yorke when $\limsup_{n\to+\infty} d(f^{(n)}(x), f^{(n)}(y)) > 0$ and $\liminf_{n\to+\infty} d(f^{(n)}(x), f^{(n)}(y)) = 0$, meaning that their orbits always oscillate as the iterations pass. When a system is compact and contains an uncountable set of such points, it is claimed as chaotic according to Li–Yorke [30,48]. A similar property is regarded in the following NIST test [46].

  - Runs Test. To determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow.

- Topological entropy. The desire to formulate an equivalency of the thermodynamics entropy has emerged both in the topological and statistical fields. Once again, a similar objective has led to two different rewritings of an entropy-based disorder: the famous Shannon definition is approximated in the statistical approach, whereas topological entropy has been defined previously. This value measures the average exponential growth of the number of distinguishable orbit segments. In this sense, it measures the complexity of the topological dynamical system, whereas the Shannon approach comes to mind when defining the following test [46]:

  - Approximate Entropy Test. Compare the frequency of the overlapping blocks of two consecutive/adjacent lengths (m and $m + 1$) against the expected result for a random sequence.

- Non-linearity, complexity. Finally, let us remark that non-linearity and complexity are not only sought in general to obtain chaos, but they are also required for randomness, as illustrated by the two tests below [46].

  - Binary Matrix Rank Test. Check for linear dependence among fixed-length substrings of the original sequence.
  - Linear Complexity Test. Determine whether or not the sequence is complex enough to be considered random.

We have recalled in this article that asynchronous iterations are, among other things, strongly transitive, topologically mixing, chaotic as defined by Li and Yorke, and that they have a topological entropy and an exponent of Lyapunov both equal to $ln(\mathsf{N})$, where N is the size of the iterated vector, see theorems of Section 3. Due to these topological properties, we are driven to believe that a generator based on chaotic iterations could probably be able to pass batteries for pseudorandomness like the NIST one. Indeed, the following sections show that defective generators have their statistical properties improved by asynchronous iterations.

4.2.2. The CIPRNGs: Asynchronous Iterations Based PRNGs

This section focus on the presentation of various realizations of pseudorandom number generators based on asynchronous iterations, see Figure 1 for speed comparison.

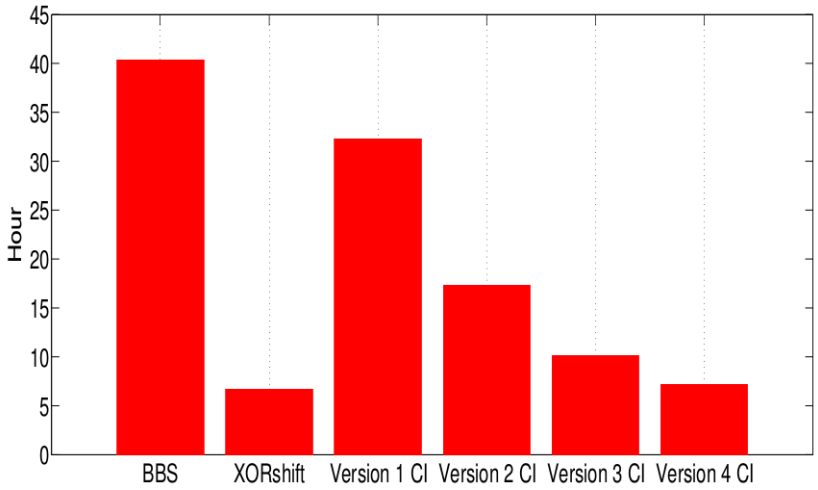

Figure 1. Speed comparison between BBS, XORshift, and CIPRNGs version 1–4.

CIPRNG, Version 1

Let $\mathsf{N} \in \mathbb{N}^*$, $\mathsf{N} \geqslant 2$, and $\mathcal{M}$ be a finite subset of $\mathbb{N}^*$. Consider two possibly defective generators called PRNG1 and PRNG2 we want to improve, the first one having his terms into $[\![1, \mathsf{N}]\!]$ whereas the second ones return integers in $\mathcal{M}$, which is always possible. The first version of a generator resulting in a post-treatment on these defective PRNGs using asynchronous iterations has been denoted by CIPRNG(PRNG1,PRNG2) version 1. This (inefficient) proof of concept is designed by the following process [49,50]:

1. Some asynchronous iterations are fulfilled, with the vectorial negation and PRNG1 as strategy, to generate a sequence $(x^n)_{n\in\mathbb{N}} \in \left(\mathbb{B}^{\mathsf{N}}\right)^{\mathbb{N}}$ of Boolean vectors: the successive internal states of the iterated system.
2. Some of these vectors are randomly extracted with PRNG2 and their components constitute our pseudorandom bit flow. Algorithm 1 provides the way to produce one output.

Algorithm 1: An arbitrary round of CIPRNG(PRNG1,PRNG2) version 1

Input: The internal state x (an array of N 1-bit words)
Output: An array of N 1-bit words
1: **for** $i = 0, \ldots, PRNG1()$ **do**
2: $S \leftarrow PRNG2()$;
3: $x_S \leftarrow \overline{x_S}$;
4: return x;

In other words, asynchronous iterations are realized as follows. Initial state $x^0 \in \mathbb{B}^{\mathsf{N}}$ is a Boolean vector taken as a seed and strategy $(S^n)_{n\in\mathbb{N}} \in [\![1,\mathsf{N}]\!]^{\mathbb{N}}$ is a sequence produced by PRNG2. Lastly, iteration function f is the vectorial Boolean negation. So, at each iteration, only the S^i-th component of state x^n is updated, as follows

$$x_i^n = \begin{cases} x_i^{n-1} & \text{if } i \neq S^i, \\ \overline{x_i^{n-1}} & \text{if } i = S^i. \end{cases} \tag{3}$$

Finally, some x^n are selected by a sequence m^n as the pseudorandom bit sequence of our generator, where $(m^n)_{n\in\mathbb{N}} \in \mathcal{M}^{\mathbb{N}}$ is obtained using PRNG2. That is, the generator returns the following values: the components of x^{m^0}, followed by the components of $x^{m^0+m^1}$, followed by the components of $x^{m^0+m^1+m^2}$, etc.

Generators investigated in the first set of experiments are the well-known Logistic map, XORshift, and ISAAC, while the reputed NIST [46], DieHARD [51], and TestU01 [52] test suites have been considered for statistical evaluation. Table 1 contains the statistical results (number of tests successfully passed) obtained by the considered inputted generators, while Table 2 shows the results with the first version of our CIPRNGs: improvements, published in [49,50], are obvious.

Table 1. Statistical results of well-known PRNGs.

	Logistic	**XORshift**	**ISAAC**
NIST SP 800-22 (15 tests)	14	14	15
DieHARD (18 tests)	16	15	18
TestU01 (516 tests)	250	370	516

Table 2. Statistical results for the CIPRNG version 1.

Test Name	**CIPRNG Version 1**			
	Logistic + Logistic	**XORshift + XORshift**	**ISAAC + XORshift**	**ISAAC + ISAAC**
NIST (15)	15	15	15	15
DieHARD (18)	18	18	18	18
TestU01 (516)	378	507	516	516

We have enhanced this CIPRNG several times, and tested these generators deeply during the last decade. We only explain in this article the XOR CIPRNG version, as an example of a good generator produced by such an approach.

XOR CIPRNG

Instead of updating only one cell at each iteration as the previous versions of our CIPRNGs, we can try to choose a subset of components and to update them together. Such an attempt leads to a kind of merger of the two random sequences. When the updating function is the vectorial negation, this algorithm can be rewritten as follows [53]:

$$\begin{cases} x^0 \in [\![0, 2^{\mathsf{N}} - 1]\!], S \in [\![0, 2^{\mathsf{N}} - 1]\!]^{\mathbb{N}} \\ \forall n \in \mathbb{N}^*, x^n = x^{n-1} \oplus S^{n-1}, \end{cases} \tag{4}$$

and this rewriting can be understood as follows. The n-th term S^n of the sequence S, which is an integer of N binary digits, whose list of digits in binary decomposition is the list of cells to update in the state x^n of the system (represented as an integer having N bits too). More precisely, the k-th component of this state (a binary digit) changes if and only if the k-th digit in the binary decomposition of S^n is 1.

This generator has been called XOR CIPRNG, it has been introduced, theoretically studied, and tested in [47,53]. It uses a very classical pseudorandom generation approach, the unique contribution is its relation with asynchronous iterations: the single basic component presented in the previous equation is of ordinary use as a good elementary brick in various PRNGs. It corresponds to the discrete dynamical system in asynchronous iterations.

4.2.3. Preserving Security

This section is dedicated to the security analysis of the proposed PRNGs, both from a theoretical and from a practical point of view.

Theoretical Proof of Security

The standard definition of indistinguishability used is the classical one as defined for instance in ([54] Chapter 3). This property shows that predicting the future results of the PRNG cannot be done in a reasonable time compared to the generation time. It is important to emphasize that this is a relative notion between breaking time and the sizes of the keys/seeds. Of course, if small keys or seeds are chosen, the system can be broken in practice. However, it also means that if the keys/seeds are large enough, the system is secured. As a complement, an example of a concrete practical evaluation of security is outlined in the next subsection.

In a cryptographic context, a pseudorandom generator is a deterministic algorithm G transforming strings into strings and such that, for any seed s of length m, $G(s)$ (the output of G on the input s) has size $\ell_G(m)$ with $\ell_G(m) > m$. The notion of secure PRNGs can now be defined as follows.

Definition 9. *A cryptographic PRNG G is secure if for any probabilistic polynomial time algorithm D, for any positive polynomial p, and for all sufficiently large m's,*

$$|\Pr[D(G(U_m)) = 1] - Pr[D(U_{\ell_G(m)}) = 1]| < \frac{1}{p(m)},$$

where U_r is the uniform distribution over $\{0,1\}^r$ and the probabilities are taken over U_m, $U_{\ell_G(m)}$ as well as over the internal coin tosses of D.

Intuitively, it means that there is no polynomial time algorithm that can distinguish a perfect uniform random generator from G with a non negligible probability. An equivalent formulation of this well-known security property means that it is possible in practice to predict the next bit of the generator, knowing all the previously produced ones. The interested reader is referred to ([54] Chapter 3) for more information. Note that it is quite easily possible to change the function ℓ into any polynomial function ℓ' satisfying $\ell'(m) > m$) ([54] Chapter 3.3).

The generation schema developed in the XOR CIPRNG is based on a pseudorandom generator. Let H be a cryptographic PRNG. Let $S_1, \ldots, S_k$ be the strings of length N such that $H(S_0) = S_1 \ldots S_k$ ($H(S_0)$ is the concatenation of the S_i's). The XOR CIPRNG X defined previously is the algorithm mapping any string of length $2N$ $x_0 S_0$ into the string $(x_0 \oplus S_0 \oplus S_1)(x_0 \oplus S_0 \oplus S_1 \oplus S_2) \ldots (x_o \bigoplus_{i=0}^{i=k} S_i)$. We have proven in [53] that,

Theorem 5. *If H is a secure cryptographic PRNG, then the XOR CIPRNG X is a secure cryptographic PRNG too.*

Practical Security Evaluation

Given a key size, it is possible to measure in practice the minimum duration needed for an attacker to break a cryptographically secure PRNG, if we know the power of his/her machines. Such a concrete security evaluation is related to the (T, ε)-security notion, which has been evaluated for various

CIPRNGs in [53] and in submitted papers. A short example of such a study for the XOR CIPRNG is provided as an illustrative example in Figure 1.

Let us firstly recall that,

Definition 10. *Let $\mathcal{D} : \mathbb{B}^M \longrightarrow \mathbb{B}$ be a probabilistic algorithm that runs in time T. Let $\varepsilon > 0$. $\mathcal{D}$ is called a $(T, \varepsilon)-$distinguishing attack on pseudorandom generator G if*

$$\left| Pr[\mathcal{D}(G(k)) = 1 \mid k \in_R \{0,1\}^\ell] - Pr[\mathcal{D}(s) = 1 \mid s \in_R \mathbb{B}^M] \right| \geqslant \varepsilon,$$

where the probability is taken over the internal coin flips of $\mathcal{D}$, and the notation "$\in_R$" indicates the process of selecting an element at random and uniformly over the corresponding set.

Let us recall that the running time of a probabilistic algorithm is defined to be the maximum of the expected number of steps needed to produce an output, maximized over all inputs; the expected number is averaged over all coin flips made by the algorithm [55]. We are now able to define the notion of cryptographically secure PRNGs:

Definition 11. *A pseudorandom generator is $(T, \varepsilon)-$secure if there exists no $(T, \varepsilon)-$distinguishing attack on this pseudorandom generator.*

We have proven in [53] that,

Proposition 7. *If the inputted PRNG is (T, ε)-secure, then this is the case too for the XOR CIPRNG.*

Suppose for instance that the XOR CIPRNG with the cryptographically secure BBS as input will work during $M = 100$ time units, and that during this period, an attacker can realize 10^{12} clock cycles. We thus wonder whether, during the PRNG's lifetime, the attacker can distinguish this sequence from a truly random one, with a probability greater than $\varepsilon = 0.2$. We consider that the modulus of BBS N has 900 bits, that is, contrarily to previous sections, we use here the BBS generator with relevant security parameters.

Predicting the next generated bit knowing all the previously released ones by the XOR CIPRNG is obviously equivalent to predicting the next bit in the BBS generator, which is cryptographically secure. More precisely, it is (T, ε)-secure: no (T, ε)-distinguishing attack can be successfully realized on this PRNG, if [56]

$$T \leqslant \frac{L(N)}{6N(log_2(N))\varepsilon^{-2}M^2} - 2^7 N\varepsilon^{-2}M^2 log_2(8N\varepsilon^{-1}M) \tag{5}$$

where M is the length of the output ($M = 100$ in our example), and $L(N)$ is equal to

$$2.8 \times 10^{-3} exp\left(1.9229 \times (N\ ln\ 2)^{\frac{1}{3}} \times (ln(N\ ln\ 2))^{\frac{2}{3}}\right)$$

is the number of clock cycles to factor a N-bit integer.

A direct numerical application shows that this attacker cannot achieve his $(10^{12}, 0.2)$ distinguishing attack in that context.

4.3. Other Concrete Applications

4.3.1. Information Security Field

The application of complex dynamics from asynchronous iterations to information security has been developed in various directions over the past decade. New dissimulation algorithms were first proposed, each with its own particularity: watermarking (without extraction) or information hiding [38,57], robust or fragile, chaotic or not [39], inserting only one bit or a large quantity of

information [33], coupled to our pseudo-random generator [45], using (or not) the information contained in the host support (via Canny's filters for example), etc.

In the same way, the study of hash functions based on asynchronous iterations has been investigated. The objective was not to propose a novel complete hash function [28,58], but to perform a post-processing of asynchronous iterations on pre-existing hash functions [32]. This results in proven chaotic hash functions, and when the provided hash functions are defective (in terms of diffusion, confusion or avalanche effect [59]), the chaos properties improve them, repair these defects. It was also proven that the hash function resulting from post-processing preserves some of the cryptographic properties possessed by the provided input, such as resistance to first and second preimage, or collision resistance, defined in complexity theory [60]. Finally, these hash functions are keyed [40,61], and they can be coupled to generators based on asynchronous iterations that are described below.

4.3.2. Biological Modeling

A second field of investigation seemed very interesting based on the modeling, study and simulation of complex systems from disciplines other than computing, that is, processes whose complex dynamics can take the following form: an operation taken from a possible set of functions, and applied only to a variable subset of system coordinates. Such complex dynamics occur naturally in molecular biology, and more particularly in the spatial folding of proteins and in the evolution of genomes over time.

The protein folding model commonly used in conformation prediction tools, known as the 2D/3D HP square lattice model, has been rewritten using a discrete dynamic system in asynchronous iterations, and we proved that this system had several chaos properties [62]. Such a complex dynamic raises many questions [63]. First, since the problem of predicting the 3D conformation of a protein is proven NP-complete [64], the prediction of the shape of a protein is done by artificial intelligence. However, we have shown that at least a number of these tools do not know how to manage some complex dynamics, such as those we found in the HP model [65]. This raises the question of the quality of the predicted conformations, especially since for cost reasons these predictions are rarely confronted with reality [66]. In addition, we found that some prediction tools consisted in finding the 3D conformation ("protein") minimizing the free enthalpy [67] of self-avoiding walks (SAWs) built by elongation, when other tools operate by bending the straight line having the size of the protein [68]. We proved that these two sets were disjointed, and that the proof of NP-completeness was only valid for the first set. Thus, in the second case, the dynamics seem too complex to be predicted by artificial intelligence tools, and the use of these tools is a priori unfounded, the problem not being proven at this difficult time [69]. Finally, it should be noted that current biological research tends to show that proteins are predominantly "intrinsically disordered", and that our study of the chaos of protein folding using asynchronous iterations could help people working in this field [70].

The second iterative system subject to local modifications in biology is the genome, which changes during evolution by mutation, insertion-deletion of nucleotides, by larger changes (inversion, or simple copy or deletion of long DNA strands), or by other specific modifications of the repetitions (segment duplication, tandem repetitions, and displacement of transposable elements TEs). These different operations could be modeled using asynchronous system tools, while no biomathematical modelling of evolution had so far been attempted, with the exception of certain types of mutation, and elementary models for certain types of TE. However, being able to predict this evolution, in the past and in the future, could have many interests: predicting the evolution of viruses such as HIV or influenza, reconstructing the past history or even the ancestor of bacteria that are highly aggressive to our species, in order to better fight them, etc. We have therefore modelled several genomic rearrangement operations, studied their complexity, and proved that, although complex, this dynamic could be predicted to some extent [71]. We rewrote and studied nucleic mutations as a discrete dynamic system, and generalized and studied the GTR mutation model [72]. We proposed

matrix models of mutation with six parameters, which have been applied to concrete cases of evolution of *Saccharomyces cerevisiae* [73].

5. Conclusions

As we can see, until now, the rewriting of asynchronous iterations as discrete dynamical systems has only been used to study the disorder, the maximum divergence that can be obtained by such iterations, the application framework targeted being computer security: the generation of pseudo-random numbers, hash functions, and symmetric encryption operation modes. For example, in the context of pseudo-random number generators, we want to improve existing generators by adding chaos properties while preserving the cryptographically secure character, and we verify on statistical test batteries that the randomness of this post-processing is of better quality than the one provided as input. Here, the sequence of subsets is produced by the input generator, and the sequence of vectors of $(\mathbb{Z}/2\mathbb{Z})^{\mathsf{N}}$ is the output generator. At the biomathematical level, the objective was to understand the evolution over time of "disordered" biological processes such as protein folding or gene expression.

With these elements in mind, we plan to pursue this research in three directions, building on the work carried out over many years on the convergence criteria for asynchronous iterative methods. This will make it possible to deepen the knowledge of cases of divergence (and convergence, studied in particular with tools of measurement theory) of such iterations, taking into account the delay.

Thus, first of all, we wish to continue the study of the disorder of asynchronous iterations, by analyzing, in depth, the associated Markov chains, recently highlighted. In parallel, we intend to focus on the Kolmogorov complexity of such iterations, his study can be done by considering asynchronous iterations as Turing machines (formulation already realized). It will then be a question of characterizing which Turing machines are chaotic, ergodic, etc. Describing pseudo-random number generators (and more generally, any algorithm or Turing machine) in the form of discrete dynamic systems leads to (at least) two advantages: one can use a solid mathematical theory, namely symbolic dynamics, to have a useful description of the system, and then one can use ergodic theory to study, in depth, the statistical properties of the system. A standard example of such a process and the shift function is recalled below.

Let be Σ an alphabet (finite or countable), for example $\Sigma = \{0.1\}$, and $X = 2^{\mathbb{N}}$ the set of all words with symbols of Σ. We consider the offset operator $\sigma : X \to X$ defined by:

$$\sigma(\omega_1, \omega_2, \ldots, \omega_n, \ldots) = (\omega_2, \omega_3, \ldots).$$

If now we are interested in the statistical properties of such an operator, then it is sufficient to introduce an ad hoc σ-algebra and a measure invariant, i.e., a measure μ such as $\mu(\sigma^{-1}(A)) = \mu(A)$). The result is called a measured dynamic system, i.e., a measured space with particular properties (measurement preserved with respect to the operator) and it is relatively easy to verify the statistical properties of the function, since we have a notion of an integral. For example, the system introduced above is a mixing system.

Such an approach, for instance, makes it possible to study in a very useful way the statistical properties of a pseudo-random number generator. However, it focuses on the mixing properties (in the ergodic sense) of a system. In other words, when we conclude that an algorithm is "random", we have in fact proved that the output of the algorithm, after many iterations, is distributed uniformly (if we use, as is usually done, the uniform distribution for invariant measurement). However, "being chaotic" can have a number of meanings and, for example in cryptography, one would like to have a link between security and randomness (in the ergodic sense), such as: randomness leads to some form of security. The link between these concepts, as well as to other concepts of randomness that may be found in mathematics, is not currently clearly defined, and we would like to explore this point further with the proposed approach. To the best of our knowledge, this will be the first time that the issue of

IT security has been questioned from the perspective of ergodic theory. This approach is of interest, particularly for auxiliary channel attacks, for which there is currently no formal framework to assess the security of software or physical devices.

In terms of complexity, there is a notion of randomness in the sense of Kolmogorov. The complexity (Kolmogorov) of a chain is the length of the smallest description of that chain in a fixed universal description language. Thus, we can define that a chain is random if, and only if it is smaller than the size of any computer program (for a specified universal Turing machine) that can produce this chain. For dynamic systems, Kolmogorov's notion of complexity is strictly related to the classical notion of topological entropy, through Brudno's theorem. However, it is still unclear how the Kolmogorov randomness is related to mixing, security, or recurrence properties.

Moreover, if the asynchronous iterations were initially studied for their convergence, in particular within the framework of distributed digital algorithmics, their reformulation in the form of a dynamic system, then a graph, was only studied for divergence purposes. We would therefore like, in a second step, to study what these reforms can bring to the study of the convergence of such asynchronous iterations. We also intend to bring convergence results from the world of dynamic systems to that of discrete mathematics. Finally, until now, the delay has not been taken into consideration: we consider that the vector at time t is deduced from the vector at time $t-1$, and depends on a continuous function and a sequence of coordinates to be updated. In other words, we assume that the stochastic process associated with asynchronous iterations satisfies Markov's property, and we wish in further work to consider the case where this is no longer true. This applies to both convergence and divergence. These results will be applied to a better understanding of the spatial and temporal evolution dynamics of biological sequences (genomes and proteins), and in particular, will make it possible to increase knowledge about the generation of pseudo-random numbers.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Kozyakin, V. A short introduction to asynchronous systems. In *Sixth International Conference on Difference Equations*; CRC: Boca Raton, FL, USA, 2004; pp. 153–165.
2. Wolfson-Pou, J.; Chow, E. Convergence models and surprising results for the asynchronous Jacobi method. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Vancouver, BC, Canada, 21–25 May 2018; pp. 940–949.
3. Wolfson-Pou, J.; Chow, E. Modeling the asynchronous Jacobi method without communication delays. *J. Parallel Distrib. Comput.* **2019**, *128*, 84–98. [CrossRef]
4. Guyeux, C. An update on the topological and ergodic properties of asynchronous iterations. In Proceedings of the Sixth International Conference on Parallel, Distributed, GPU and Cloud Computing for Engineering, Pécs, Hungary, 4–5 June 2019; Iványi, P., Topping, B.H.V., Eds.; Civil-Comp Press: Stirlingshire, UK, 2019.
5. Robert, F. *Discrete Iterations, a Metric Study*; Volume 6 of Series in Computational Mathematics; Springer: Heidelberg, Germany, 1986.
6. Kaszkurewicz, E.; Bhaya, A. *Matrix Diagonal Stability in Systems and Computation*; Springer Science & Business Media: New York, NY, USA, 2012.
7. Kozyakin, V. An Annotated Bibliography on Convergence of Matrix Products and the Theory of Joint/Generalized Spectral Radius. Available online: https://drive.google.com/file/d/0Bxw63g5l4P7pLXgwcWxVZ3RoTVk/view?usp=sharing (accessed on 16 October 2020).
8. Guyeux, C. *Le Désordre des Itérations Chaotiques—Applications aux Réseaux de Capteurs, à la Dissimulation D'information, et aux Fonctions de Hachage*; Éditions Universitaires Européennes: Saarbrücken, Germany, 2012; p. 362, ISBN 978-3-8417-9417-8.
9. Miellou, J.-C.; Spitéri, P. Un critère de convergence pour des méthodes générales de point fixe. *Rairo Modélisation Mathématique Analyse Numérique* **1985**, *19*, 645–669. [CrossRef]

10. Spitéri, P. Contribution à L'étude de la Stabilite au Sens de Liapounov de Certains Systemes Differentiels Non Lineaires. Ph.D. Thesis, Université de Franche-Comté, Besançon, France, 1974.
11. Chazan, D.; Miranker, W. Chaotic relaxation. *Linear Algebra Appl.* **1969**, *2*, 199–222. [CrossRef]
12. Miellou, J.-C. Algorithmes de relaxation chaotique à retards. *Rairo* **1975**, *9*, 148–162, . [CrossRef]
13. Baudet, G.M. Asynchronous iterative methods for multiprocessors. *J. ACM* **1978**, *25*, 226–244. [CrossRef]
14. Miellou, J.-C. Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés. *C. R. Acad. Sci. Paris* **1975**, *280*, 233–236.
15. El Tarazi, M.N. Contraction et Ordre Partiel Pour l'Etude d'Algorithmes Synchrones et Asynchrones en Analyse Numérique. Ph.D. Thesis, Faculté des Sciences et Techniques de l'Université de Franche-Comté, Besançon, France, 1981.
16. El Baz, D. Contribution à l'Algorithmique Parallèle. Le Concept d'Asynchronisme: Étude Théorique, Mise en œuvre, et Application. Ph.D. Thesis, Habilitation à Diriger des Recherches, Institut National Polytechnique de Toulouse, Paris, France, 1998.
17. Bertsekas, D.P.; Tsitsiklis, J.N. Parallel and Distributed Iterative Algorithms: A Selective Survey. Available online: https://www.researchgate.net/publication/37594457_Parallel_and_distributed_iterative_algorithms_a_selective_survey (accessed on 16 October 2020).
18. Bahi, J.M. Algorithmes Asynchrones Pour des Systèmes Différentiels-Algébriques. Simulation Numérique sur des Exemples de Circuits Électriques. Ph.D. Thesis, Université de Franche-Comté, Besançon, France, 1991.
19. Bahi, J.M. *Méthodes Itératives Dans des Espaces Produits. Application au Calcul Parallèle*; Habilitation à Diriger des Recherches, Université de Franche-Comté: Besançon, France, 1998.
20. El Tarazi, M.N. Some convergence results for asynchronous algorithms. *Numer. Math.* **1982**, *39*, 325–340. [CrossRef]
21. El Tarazi, M.N. Algorithmes mixtes asynchrones. Etude de convergence monotone. *Numer. Math.* **1984**, *44*, 363–369. [CrossRef]
22. Jacquemard, C. Contribution à l'Etude d'Algorithmes à Convergence Monotone. Ph.D. Thesis, Université de Franche-Comté, Besançon, France, 1977.
23. Bertsekas, D.P.; Tsitsiklis, J.N. *Parallel and Distributed Computation: Numerical Methods*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1989.
24. Lubachevsky, B.; Mitra, D. A chaotic asynchronous algorithm for computing the fixed point of a nonnegative matrix of unit spectral radius. *J. ACM* **1986**, *33*, 130–150. [CrossRef]
25. Frommer, A.; Szyld, D.B. Asynchronous two-stage iterative methods. *Numer. Math.* **1994**, *69*, 141–153. [CrossRef]
26. Frommer, A.; Schwandt, H.; Szyld, D.B. Asynchronous weighted additive Schwarz methods. *Electron. Trans. Numer. Anal.* **1997**, *5*, 48–61.
27. Chajakis, E.D.; Zenios, S.A. Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization. *Parallel Comput.* **1991**, *17*, 873–894. [CrossRef]
28. Guyeux, C.; Bahi, J. A topological study of chaotic iterations. Application to hash functions. In *Computational Intelligence for Privacy and Security*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 51–73.
29. Formenti, E. Automates Cellulaires et Chaos: De la Vision Topologique à la Vision Algorithmique. Ph.D. Thesis, École Normale Supérieure de Lyon, Lyon, France, 1998.
30. Li, T.Y.; Yorke, J.A. Period three implies chaos. *Am. Math. Mon.* **1975**, *82*, 985–992. [CrossRef]
31. Bahi, J.; Couchot, J.-F.; Guyeux, C.; Richard, A. On the link between strongly connected iteration graphs and chaotic boolean discrete-time dynamical systems. In Proceedings of the 18th International Symposium on Fundamentals of Computation Theory, Oslo, Norway, 22–25 August 2011; Volume 6914, pp. 126–137.
32. Bahi, J.; Couchot, J.-F.; Guyeux, C. Quality analysis of a chaotic proven keyed hash function. *Int. J. Adv. Internet Technol.* **2012**, *5*, 26–33.
33. Guyeux, C.; Bahi, J.M. A new chaos-based watermarking algorithm. In Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT), Athens, Greece, 26–28 July 2010; pp. 1–4.
34. Bahi, J.; Guyeux, C. *Discrete Dynamical Systems and Chaotic Machines: Theory and Applications*; Chapman & Hall, CRC Press: Boca Raton, FL, USA, 2013; p. 212.
35. Guyeux, C.; Wang, Q.; Fang, X.; Bahi, J. Introducing the truly chaotic finite state machines and their applications in security field. In Proceedings of the 2014 24th International Symposium on Nonlinear Theory and its Applications (NOLTA), Luzern, Switzerland, 14–18 September 2014.

36. Bahi, J.M.; Couchot, J.-F.; Guyeux, C. Steganography: A class of algorithms having secure properties. In Proceedings of the 2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Dalian, China, 14–16 October 2011; pp. 209–212.
37. Couchot, J.-F. Couturier, R.; Guyeux, C. Stabylo: Steganography with adaptive, Bbs, and binary embedding at low cost. *Ann. Telecommun.-Ann. Télécommun.* **2015**, *70*, 441–449. [CrossRef]
38. Guyeux, C.; Bahi, J.M. An improved watermarking scheme for internet applications. In Proceedings of the 2010 2nd International Conference on Evolving Internet, Porto, Portugal, 18–22 October 2010; pp. 119–124.
39. Bahi, J.M.; Friot, N.; Guyeux, C. Lyapunov exponent evaluation of a digital watermarking scheme proven to be secure. In Proceedings of the 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Piraeus, Greece, 18–20 July 2012; pp. 359–362.
40. Lin, Z.; Guyeux, C.; Yu, S.; Wang, Q.; Cai, S. On the use of chaotic iterations to design keyed hash function. *Clust. Comput.* **2019**, *22*, 905–919. [CrossRef]
41. Contassot-Vivier, S.; Couchot, J.-F.; Guyeux, C.; Heam, P.-C. Random walk in a n-cube without Hamiltonian cycle to chaotic pseudorandom number generation: Theoretical and practical considerations. *Int. J. Bifurc. Chaos* **2017**, *27*, 1750014. [CrossRef]
42. Couchot, J.-F.; Heam, P.-C.; Guyeux, C.; Wang, Q.; Bahi, J.M. Pseudorandom number generators with balanced gray codes. In Proceedings of the 2014 11th International Conference on Security and Cryptography (SECRYPT), Vienna, Austria, 28–30 August 2014; pp. 1–7.
43. Bahi, J.; Guyeux, C.; Salomon, M. Building a chaotic proven neural network. In Proceedings of the IEEE International Conference on Computer Applications and Network Security (ICCANS 2011), Malé, Maldives, 27–29 May 2011.
44. Bahi, J.M.; Couchot, J.-F.; Guyeux, C.; Salomon, M. Neural networks and chaos: Construction, evaluation of chaotic networks, and prediction of chaos with multilayer feedforward network. *Chaos* **2012**, *22*, 013122. [CrossRef]
45. Salomon, M.; Couturier, R.; Guyeux, C.; Couchot, J.-F.; Bahi, J.M. Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine. *La Recherche Européenne en Télémédecine* **2017**, *6*, 79–92. [CrossRef]
46. Barker, E.; Roginsky, A. Draft NIST Special Publication 800-131 Recommendation for the Transitioning of Cryptographic Algorithms And Key Sizes. Available online: https://csrc.nist.gov/CSRC/media/Publications/sp/800-131a/rev-2/draft/documents/sp800-131Ar2-draft.pdf (accessed on 16 October 2020).
47. Bahi, J.; Fang, X.; Guyeux, C. An optimization technique on pseudorandom generators based on chaotic iterations. In Proceedings of the 2012 4th International Conference on Evolving Internet (INTERNET), Venice, Italy, 24–29 June 2012; pp. 31–36.
48. Ruette, S. Chaos en Dynamique Topologique, en Particulier sur l'Intervalle, Mesures d'Entropie Maximale. Ph.D. Thesis, Université d'Aix-Marseille II, Marseille, France, 2001.
49. Bahi, J.M.; Guyeux, C.; Wang, Q. Improving random number generators by chaotic iterations. Application in data hiding. In Proceedings of the International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, China, 22–24 October 2010.
50. Wang, Q.; Guyeux, C.; Bahi, J.M. A novel pseudo-random number generator based on discrete chaotic iterations. In Proceedings of the 2009 First International Conference on Evolving Internet, Cannes, France, 23–29 August 2009; IEEE Computer Society: Washington, DC, USA, 2009; pp. 71–76.
51. Marsaglia, G. Diehard: A Battery of Tests of Randomness. 1996. Available online: http://stat.fsu.edu/~geo/diehard.html (accessed on 16 October 2020).
52. Simard, R.; De Montréal, U. Testu01: A C library for empirical testing of random number generators. *ACM Trans. Math. Softw.* **2007**, *33*. [CrossRef]
53. Bahi, J.M.; Couturier, R.; Guyeux, C.; Héam, P.-C. Efficient and Cryptographically Secure Generation of Chaotic Pseudorandom Numbers on GPU. *CoRR* **2011**. Available online: https://arxiv.org/abs/1112.5239 (accessed on 16 October 2020).
54. Goldreich, O. *Foundations of Cryptography: Basic Tools*; Cambridge University Press: Cambridge, UK, 2007.
55. Knuth, D.E. *The Art of Computer Programming, Volume 3: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley: Reading, MA, USA, 1997.

56. Fischlin, R.; Schnorr, C.P. Stronger security proofs for rsa and rabin bits. In *Advances in Cryptology—EUROCRYPT'97, Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques, Konstanz, Germany, 11–15 May 1997*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 267–279.
57. Guyeux, C.; Friot, N.; Bahi, J.M. Chaotic iterations versus spread-spectrum: Chaos and stego security. In Proceedings of the 6th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Darmstadt, Germany, 15–17 October 2010.
58. Guyeux, C.; Bahi, J.M. Hash functions using chaotic iterations. *J. Algorithms Comput. Technol.* **2010**, *4*, 167–182.
59. Lin, Z.; Guyeux, C.; Wang, Q.; Yu, S. Diffusion and confusion of chaotic iteration based hash functions. In Proceedings of the 2016 19th IEEE International Conference on Computational Science and Engineering (CSE), Paris, France, 24–26 August 2016; pp. 444–447.
60. Lin, Z.; Guyeux, C.; Yu, S.; Wang, Q. Design and evaluation of chaotic iterations based keyed hash function. In Proceedings of the 2017 8th iCatse Conference on Information Science and Applications (ICISA), Macau, China, 20–23 March 2017; Volume 424, pp. 404–414.
61. Bahi, J.; Couchot, J.-F.; Guyeux, C. Performance analysis of a keyed hash function based on discrete and chaotic proven iterations. In Proceedings of the 2011 3rd International Conference on Evolving Internet (INTERNET), Luxembourg, 19–24 June 2011; pp. 52–57.
62. Bahi, J.M.; Côté, N.; Guyeux, C. Chaos of protein folding. In Proceedings of the 2011 International Joint Conference on Neural Networks, San Jose, CA, USA, 31 July–5 August 2011; pp. 1948–1954.
63. Bahi, J.; Côté, N.; Guyeux, C.; Salomon, M. Protein folding in the 2D hydrophobic-hydrophilic (HP) square lattice model is chaotic. *Cogn. Comput.* **2012**, *4*, 98–114, . [CrossRef]
64. Berger, B.; Leighton, T. Protein folding in the hydrophobic-hydrophilic (HP) is HP-complete. In Proceedings of the Second Annual International Conference on Computational Molecular Biology, New York, NY, USA, 22–25 March 1998; ACM: New York, NY, USA, 1998; pp. 30–39.
65. Bahi, J.M.; Guyeux, C.; Mazouzi, K.; Philippe, L. Computational investigations of folded self-avoiding walks related to protein folding. *Comput. Biol. Chem.* **2013**, *47*, 246–256. [CrossRef]
66. Guyeux, C.; Côté, N.; Bienia, W.; Bahi, J.M. Is protein folding problem really a np-complete one? first investigations. *J. Bioinform. Comput. Biol.* **2014**, *12*, 1350017. [CrossRef]
67. Faver, J.C.; Benson, M.L.; He, X.; Roberts, B.P.; Wang, B.; Marshall, M.S.; Sherrill, C.D.; Merz, K.M. The Energy Computation Paradox and ab initio Protein Folding. *PLoS ONE* **2011**, *6*, e18868. [CrossRef] [PubMed]
68. Bornberg-Bauer, E. Chain growth algorithms for HP-type lattice proteins. In Proceedings of the First Annual International Conference on Computational Molecular Biology, Santa Fé, NM, USA, 19–22 January 1997; ACM: New York, NY, USA, 1997; pp. 47–55.
69. Guyeux, C.; Nicod, J.-M.; Philippe, L.; Bahi, J.M. The study of unfoldable self-avoiding walks—Application to protein structure prediction software. *J. Bioinform. Comput. Biol.* **2015**, *13*, 1550009. [CrossRef] [PubMed]
70. Braxenthaler, M.; Unger, R.R.; Auerbach, D.; Moult, J. Chaos in protein dynamics. *Proteins Struct. Funct. Bioinform.* **1997**, *29*, 417–425. [CrossRef]
71. Bahi, J.M.; Guyeux, C.; Perasso, A. Chaos in DNA evolution. *Int. J. Biomath.* **2016**, *9*, 1650076. [CrossRef]
72. Bahi, J.M.; Guyeux, C.; Perasso, A. Relaxing the hypotheses of symmetry and time-reversibility in genome evolutionary models. *Br. J. Math. Comput. Sci.* **2015**, *5*, 439–455. [CrossRef]
73. Bahi, J.M.; Guyeux, C.; Perasso, A. Predicting the evolution of two genes in the yeast saccharomyces cerevisiae. *Procedia Comput. Sci.* **2012**, *11*, 4–16. [CrossRef]