A Visual Analysis Method of Randomness for Classifying and Ranking Pseudo-random Number Generators

Jeaneth Machicao[1,3*], Quynh Quang Ngo[2,*], Vladimir Molchanov[2], Lars Linsen[2], Odemir Bruno[1]

[1] *Scientific Computing Group, São Carlos Institute of Physics, University of São Paulo, São Carlos - SP, PO Box 369, 13560-970, Brazil*

[2] *Institute of Computer Science, Westfälische Wilhelms-Universität Münster, Einsteinstr. 62, 48149 Münster, Germany*

[3] *Department of Computer Engineering and Digital Systems. Polytechnical School, University of São Paulo, São Paulo, SP, 05508-010, Brazil*

Abstract

The development of new pseudo-random number generators (PRNGs) has steadily increased over the years. Commonly, PRNGs' randomness is "measured" by using statistical pass/fail suite tests, but the question remains, which PRNG is the best when compared to others. Existing randomness tests lack means for comparisons between PRNGs, since they are not quantitatively analysing. It is, therefore, an important task to analyze the quality of randomness for each PRNG, or, in general, comparing the randomness property among PRNGs. In this paper, we propose a novel visual approach to analyze PRNGs randomness allowing for a ranking comparison concerning the PRNGs' quality. Our analysis approach is applied to ensembles of time series which are outcomes of different PRNG runs. The ensembles are generated by using a single PRNG method with different parameter settings or by using different PRNG methods. We propose a similarity metric for PRNG time series for randomness and apply it within an interactive visual approach for analyzing similarities of PRNG time series and relating them to an optimal result of perfect randomness. The interactive analysis leads to an unsupervised classification, from which respective conclusions about the impact of the PRNGs' parameters or rankings of PRNGs on randomness are derived. We report new findings using our approach in a study of randomness for state-of-the-art numerical PRNGs such as LCG, PCG, SplitMix, Mersenne Twister, and RANDU as well as chaos-based PRNG families such as K-Logistic map and K-Tent map with varying parameter K.

Keywords: PRNG, Visualization, Interactive visual analysis, Ensemble, Unsupervised classification

1. Introduction

Pseudo-random number generators (PRNGs) are deterministic algorithms to produce a unique sequence of random numbers by inputting an initial condition (seed point). The quality of a PRNG can be characterized by its ability to represent a uniform distribution, unpredictability and long periodicity of the output number series, uncorrelated successive outputs, and portability of the algorithm. PRNGs are used in various applications such as: (i) statistics and probability theory for data sampling; (ii) decision theory as a strategy for the optimization of computational algorithms; (iii) numerical calculation, e.g., in the Monte Carlo integration method [36]; (iv) simulation and modeling of systems as a tool to simulate stochastic behavior, as in phenomena of nature [7]; (v) gaming and entertainment industry, e.g., for rendering scenarios and textures in films; (vi) programming languages and any algorithms or heuristics that require sampling, for instance, pattern recognition, artificial intelligence, genetic algorithms, deep learning, and even in more critical scenarios; (vii) cryptography, e.g., as source of randomness, initialization vector, or password generation, where

*Equally contributing first authors.

cryptographically secure random numbers generators [22, 29] are of paramount importance for encryption algorithms that operate in military communications, intelligence tools, banking systems, e-commerce, etc.

Several PRNGs have been proposed in the literature. Typically, mathematical equations, or good sources of randomness (quantum theory, chaos theory), among others, are used to design PRNGs. For instances, classical PRNGs are based on numerical algorithms such as the *Linear Congruential Generator* (LCG) [24] and *Shift Registers* [14], which were two of the first PRNGs proposed in literature. LCG is based on simple mathematical operations (modulus and addition), while *Shift Registers* were inspired on digital circuits as a cascade of flip-flop registers. From that starting point, several PRNGs have emerged such as nonlinear LCG [8], *Linear Feedback Shift Registers* (LFSR) [41], and *Mersenne Twister* [35] whose algorithm is based on a matrix linear recurrence over a finite binary field and nowadays is the default PRNG on various software systems, *SplitMix* [46], and *Permuted Congruential Generator* (PCG) [39] which is an improvement of classical LCG with modulus power of two by using the most significant bits of the final state and applying a bitwise rotation or shift operation to produce the output. Another variant of classical LCG is old-fashioned RANDU, where parameters are allowed to be broken [33]. As an alternative to numerical PRNGs, other PRNG approaches have considered chaos theory [32, 21, 10, 40, 42, 30] to extract time series in order to randomize their outputs, which is motivated by the strong relationship of chaos properties to being suitable sources of high pseudo-randomness quality.

In 2017, Machicao and Bruno [30] introduced an approach for producing a family of PRNGs whose pseudo-randomness goes from weak to high accordingly to the increasing of its parameter K. These PRNGs are based on chaos theory, specifically on the Logistic map and the Tent map. They deploy the exploration of the K-digits to the right of the decimal separator (least significant digits) of a chaotic system, which is referred to as *deep zoom*. Recently, the relationship between parameter K and the quality of the pseudo-random sequences was analytically demonstrated [31], showing a rapid randomness increase as K increases.

Disproportionately to the growing effort to produce new PRNGs algorithms, the methods to evaluate their randomness have stagnated. In general, when a new design is published, its PRNG performance is only compared in terms of computational cost, speed, ease of implementation, efficiency, long periodicity, portability, and complexity, yet its randomness is mainly given in terms of pass/fail statistical tests. Since the output of the tests are usually pass/fail values rather than numerical values, they are of limited use for a measure of randomness quality. Therefore, there is still a lack of methods that provide a measure of PRNG quality in terms of pseudo-randomness tests. In fact, in literature, less attention has been given to the exploration of new randomness tests and its commonly used statistical tests. Most existing randomness tests are based on statistical hypotheses such as ENT [48], DIEHARD [34], NIST [43], or TESTU01 [26], which are used to analyze the pattern distribution of sequences generated by PRNGs. In general, it would be desirable to rank the randomness behavior of PRNGs and to classify their generators.

A first attempt to apply unsupervised machine-learning methods on encrypted data were made leading to interesting results [23]. Machine-learning techniques have also been used for other purposes related to pseudo-randomness such as to predict chaotic time series [27, 16, 9] and to classify encrypted data using supervised methods [20, 6, 4, 13]. Other attempts aimed at crypto-analysis purposes using genetic algorithms [45, 12], however with minor success.

Given the discussion above, one may ask whether there are ways to formulate a quantifiable estimator of the strength of pseudo-randomness. Thus, in this manuscript, we aim to offer an alternative approach to explore patterns and, consequently, the weaknesses of PRNG systems. We propose a visual method to classify the outcome of PRNGs and build a framework that supports analyzing classification and ranking tasks among PRNGs. We base our analysis on the Markovian transition matrix that exhibits features such as randomness and predictability of PRNGs from the outcome time series, see Section 3.2. We enhance the analysis by defining the Markovian transition matrix for a perfect time series and compare it to the PRNGs' time series to form a ranking among PRNGs, see Section 3.3. We propose to define a similarity among outcome time series based on the Markovian transition matrix and analyze the similarities within a visual framework to classify PRNGs through their outputs, see Section 3.4 and Section 3.5. The framework is built up in a top-down approach that starts with an overview and allows for iteratively zooming into regions of interests, where a provenance graph allows for intuitive navigation between the levels of zooming and for storing intermediate analysis results, see Section 4. To achieve the main goal, we made an analysis taking

advantage of these chaos-based PRNGs, which shows a growing randomness from weak to high, and divided the experiment analysis into three main tasks: First, to classify the chaos-based PRNGs based on their pseudo-randomness; second, to rank the PRNGs by their pseudo-randomness quality; third, to investigate the impact of the parameter K on the pseudo-randomness quality. Besides that, further analysis was made to evaluate our visualization approach.

Our contributions can be summarized as presenting

- a novel method for classifying and ranking PRNGs with respect to the quality of pseudo-randomness,
- a visualization framework for the interactive analysis of PRNG classifications and rankings, and
- a study for classifying and ranking the current state-of-the-art PRNGs including numerical approaches (LCG, PCG, Mersenne Twister, Splitmix, RANDU) and chaos-based approaches (K-Logistic map, K-Tent map) using our framework.

2. Related Work

2.1. Comparing PRNGs

Several attempts have been made trying to compare PRNGs [47, 44, 37]. Vattulainen et al. [47] recommend comparing PRNGs with two types of tests, i.e., from the application perspective and by using different standard tests. We, on the other hand, propose to use randomness similarity of PRNGs time series by exploiting concepts of transition matrices and information entropy to focus on randomness of PRNGs only.

Skanderova et al. [44] compare performances of pseudo-random numbers generators and chaotics number generators on differential evolution (DE). They empirically measure the speed of convergence to the global minimum by running DEs on different test functions using the numbers generators. We do not compare PRNGs in a specific application perspective, but empirically compare them using their time series outputs.

Along the line with our work, De Micco et al. [37] review randomness quantifiers based on information theory, recurrence plots, and intrinsic computation. They investigated their ability to discern the hallmark of chaos in time-series used in connection with PRNGs, which focus on the main statistical characteristics (namely invariant measure and mixing constant) of a chaotic map. A PRNG is indicated to be ideal, if $H(P) = 1$ and $C(P) = 0$, where $H(P)$ denotes the Shannon entropy of the probability distribution function P that is extracted from a time series under consideration and $C(P)$ denotes the intensive statistical complexity. We employ the indication in an approximating fashion to numerically measure how close a PRNG is to the ideal (or perfect) case as a demonstration for our approach when P is derived from the time series histogram for its intuitive description. We also refer here to other work in the context of measuring statistical complexity as a randomness quantifier of PRNGs [28, 25, 15]. We would like to emphasize the universal applicability of our visualization framework, which can be applied using any provided similarity measure derived from the quantifiers. Thus, our visualization framework would work with any pairwise similarity measure of PRNG time series.

2.2. Visual Similarity Analysis for Ensembles of Time Series

Vattulainen et al. [47] recommended comparing PRNGs with visual tests that provide further qualitative comparisons of PRNGs. However, the tests are rather raw depictions of the PRNGs' random number series outcome. Their plots have a low scalability in the number of time series, which makes it hard to compare larger number of PRNGs. Moreover, the views require expertise knowledge about PRNGs to be able to interpret.

In our visual approach, the similarity reflecting randomness of PRNGs is visualized in the form of 2D scatterplots which is both simple (and thus intuitive) and effective. Moreover, each additional time series to be investigated just adds one dot to the scatterplot, which leads to an excellent scalability for the size of the PRNG ensemble. Points may still clutter if the PRNG runs are very similar, but then it is desired to

see point clusters. Furthermore, visual clutter is addressed in our framework by interactively zooming into regions of interest.

In terms of visualization tasks and methodology, the approach we present in this paper can be characterized as visually analyzing the similarity of time series ensembles. Our visual representation uses projection methods to visually encode similarities. Visualization of the similarity of large sets of time series using projection-based approach was also proposed by Alencar et al. [2]. Their method provides an overview revealing groups of time series that share similar behavior. However, the clutter issue caused by dense clusters in the embedding was not addressed. We, on the other hand, address this issue and propose zooming operations into dense regions of interest, for which new projection layouts are computed. To keep track of the zooming operations performed during an analysis session we build a provenance graph. Hao et al. [17] proposed a recursive algorithm to lay out large sets of time series data, where a treemap is used to present time series. The layout requires a prior importance measure of the time series. We also fulfill the task of ranking time series in PRNGs datasets, but our method does not require any prior importance measure. While there exist many approaches to visually analyze time series (we refer to the book by Aigner et al. [1] for a comprehensive overview), Vattulainen et al.'s work [47] discussed above is to our knowledge the only one that addresses visualization for generated time series from PRNGs.

3. Theoretical Concepts

3.1. Chaos-based PRNGs

Chaos theory is known to be affected by its sensitivity to initial conditions, ergodicity, transitivity, complexity, instability, and its unpredictability, which are understood as adequate characteristics to be sources of pseudo-randomness [32, 21, 10, 40, 42, 30]. Machicao and Bruno [30] proposed a novel PRNG using a deep zoom approach on chaotic maps. The method generates a sequence of pseudo-random numbers by extracting K-digits to the right of the decimal separator of the points from the map orbit. As a proof of concept, the authors applied the proposed methodology to the K-Logistic map and K-Tent map.

In chaos theory, the classical Logistic map given in Eq. (1) and the Tent map given in Eq. (2) are two of the most studied recurrence maps, mainly because of the fact that even relatively simple equations can produce chaotic behavior as their parameters vary. They are given by:

$$x_{n+1} = \mu\, x_n\, (1 - x_n), \tag{1}$$

$$x_{n+1} = \begin{cases} \delta\, x_n, & \text{if } x_n < 0.5, \\ \delta\, (1 - x_n), & \text{if } x_n \geq 0.5. \end{cases} \tag{2}$$

In both equations, x_n represents a point of the orbit at the n-th time step and the phase space is defined in the unit interval $[0, 1]$, while the parameters are chosen as $\mu \in [0, 4]$ and $\delta \in [0, 2]$, respectively. It should be noted that, according to the deep-zoom method, the control parameter is fixed for every iteration, and its value must guarantee a time series in a chaotic regime in order to be used for randomness purposes, which can be verified by calculating a positive Lyapunov exponent. In this work, we have focused on the parameters $\mu = 3.99999999$ and $\delta = 1.99999999$ as suggested by Machicao and Bruno [30].

The K-Logistic map [30] is given by an initial condition x_0 (seed point), a real-valued hyper-parameter μ, and a positive integer K to compute $\{x_n\}$ according to Eq. (1) by:

$$x_n^K = x_n \cdot 10^K - \lfloor x_n \cdot 10^K \rfloor\,, \tag{3}$$

where $\lfloor \cdot \rfloor$ is the floor operation. It can be observed that the x_n^K values remain in the unit interval. Finally, each x_n^K value is discretized using parameter d by

$$R_n = \lfloor x_n^K \cdot 10^d \rfloor\,. \tag{4}$$

The resulting series $\{R_n\}_{n>0}$ is a sequence of pseudo-random integers ranging from 0 to $10^d - 1$ generated by the K-Logistic map PRNG. In our tests, we used $d = 1$ and varied K (see Section 3.6). The quality of the generated series $\{R_n\}_{n>0}$ is subject to the presented study.

Similarly, the K-Tent map is computed by applying Eqs. (3) and (4) to the outcome of Eq. (2), effectively replacing Eq. (1) with Eq. (2) and considering parameter δ instead of μ.

3.2. PRNG as Markov Process

According to the definition of the Logistic map and Tent map in Eqs. (1) and (2), the entire subsequence $\{x_i\}_{i>n}$ depends only on the value of x_n for any $n \geq 0$ and μ being fixed. Such independence is referred to as the *Markov property* in the field of stochastic processes and is usually written as follows:

$$\mathbb{P}(x_{n+1} = s_{n+1} \,|\, x_n = s_n, \ldots, x_0 = s_0) = \mathbb{P}(x_{n+1} = s_{n+1} \,|\, x_n = s_n). \tag{5}$$

Here, $\mathbb{P}$ is a probability measure and s_i, $i = 0, \ldots, n+1$, are possible states.

Obviously, the deep-zoom approach does not break the Markov property, so that $\{R_i\}_{i>n}$ is still independent of all previous values except R_n. Thus, the series $\{R_i\}_{i\geq 0}$ can be considered as a Markov chain. Since Eq. (1) or Eq. (2) do not explicitly depend on n, the Markov chain is *time homogeneous*. Further, it has a finite number of states, which is determined by the value of hyper-parameter d only and is equal to 10^d.

The study of a Markov chain with a finite number of states 10^d can be fulfilled by examining its transition matrix. Entries p_{ij} of the transition matrix M of dimensionality $10^d \times 10^d$ are the probability values of being in state i and reaching state j in the next step. Then, all $p_{ij} \in [0, 1]$ and each row of M sums up to unity. Thus, we can consider each row of the matrix to be a discrete probability distribution. Knowing the transition matrix, one can numerically characterize each separate state and the whole stochastic process in terms of periodicity, steady states, and limiting distribution.

The transition matrices are not known a priori for the series generated with the proposed PRNGs. However, one can numerically compute the *empirical Markov transition matrix* for any given finite time series $\{R_n\}_{n=0}^N$ and a sufficiently large number N. In particular, we calculate p_{ij} to be the number of occurrences where state i is followed by state j in the sequence $\{R_n\}_{n=0}^N$ divided by N. Specifically, given a sequence $\{R_n\}_{n=0}^N$ that is a realization of a PRNG, an empirical transition matrix T of size $10^d \times 10^d$ is a matrix that counts the frequency of the time series transits from one state to another among the 10^d states. We will make use of the following notations: let $M = \max\limits_{n\geq 0}\{R_n\}$, $m = \min\limits_{n\geq 0}\{R_n\}$, $h = \frac{M-m}{N}$, and $h_j = m + jh$ with $j \in \{0, 1, \ldots, 10^d - 1\}$. Then, we set $T = (p_{ij})_{i,j\in\{0,1,\ldots,10^d-1\}}$, where p_{ij} is the frequency that $R_k \in [h_i, h_{i+1})$ and $R_{k+1} \in [h_j, h_{j+1})$, i.e.,

$$p_{ij} = \frac{|\{R_{k+1} \in [h_j, h_{j+1}) | R_k \in [h_i, h_{i+1}), k = 0, 1, \ldots, N-1\}|}{N}$$

where $|.|$ denotes the cardinality (or size) of a set. As N increases, the approximation quality of the empirical transition matrix to the Markov transition matrix associated with the underlying PRNG also increases.

3.3. Entropy as a Measure of Randomness

For a given PRNG and a sufficiently long finite sequence of pseudo-random numbers $\{R_n\}_{n=0}^N$ generated from the PRNG, an empirical Markov transition matrix can be calculated as described in the previous section. Each row of the empirical matrix is a probability vector characterizing the transitions between respective states. Then, such probability vectors can be used for evaluating the randomness of the studied PRNG and for a comparative analysis of a family of PRNGs. Entropy

$$E_i = -\sum_j p_{ij} \log p_{ij} \tag{6}$$

is a well-known measure of randomness of the probability vector p_{ij} with i fixed. If $p_{ij} = 0$, the corresponding term in Eq. (6) vanishes.

In particular, fully predictable number generators would result in empirical transition matrices having a specific structure, where each row of such a matrix would consist of only zeros except for one single entry that would be equal to one. Note that the entropy of such a probability vector vanishes. Such generators represent the worst case of PRNGs. On the other hand, an ideal random number generator should have no dependence of its outcome on the previous state, i.e., it should be perfectly unpredictable. Thus, entries of the empirical transition matrix associated with the ideal generator should all be equal to 10^{-d}. Note that the resulting probability vectors (rows of the transition matrix) then all have the maximal possible entropy, which is equal to d. Thus, the entropy of the probability vectors forming the empirical Markov transition matrix can serve as a measure of randomness of the corresponding PRNG.

Based on the concept of entropy for a probabilistic distribution, we propose three *entropy measurements* for an empirical transition matrix T that reflect the randomness of the corresponding PRNGs. First, the average of the entropies of all rows of the transition matrix provides the *average entropy* of the matrix. Second, the maximum among the rows' entropies provides the *maximum entropy* for the matrix. It measures the entropy of the state where the randomness of a transition from this state to others is highest. Thus, it provides an upper boundary of the entropy of the transition matrix. Third, we also recommend to use the minimum among the rows' entropies (referred to as *minimum entropy*) as an entropy measurement for the transition matrix. This measurement provides the entropy of the state where the randomness of transition from this state to others is lowest. Thus, it gives a lower boundary of the entropy for the transition matrix.

3.4. Characterization of PRNG

Our goal is to numerically estimate and visually encode the impact of parameters on the quality of a given PRNG. The PRNG's quality is understood as the randomness of the generated time series, i.e., its low predictability, long periodicity, and weak correlation. Further aspects of PRNGs such as complexity and portability of the algorithm are left out of the scope of this study.

For the proposed analysis method, the input is a set of finite sequences of pseudo-random numbers, i.e., an ensemble of time series data. It is of no theoretical significance if distinct sequences in the ensemble have been generated by varying one or several parameters of a single PRNG or by using different PRNGs to generate the sequences. An important requirement is that all generator algorithms used to produce the sequences satisfy the Markov property. Then, for each ensemble member, we compute an empirical Markov transition matrix. Here, one can control if the length of each time series is sufficient for a reliable estimation of the matrix entries. For instance, if length N is sufficiently large, then further increasing length N should have a negligible effect on the statistical properties of the sequence, i.e., the resulting transition matrix should remain approximately the same. In fact, the lengths of different ensemble members may vary but the number of states has to be the same for all sequences.

Mapping the sequences of pseudo-random numbers to the space of probabilistic matrices T_j of size $10^d \times 10^d$ is an effective dimensionality reduction method, which preserves important stochastic characteristics of the ensemble members. When equipping this space with a proper similarity measure, one can efficiently compute similarities of distinct time series, thus, performing classification or cluster analysis on the set of original PRNGs. The existence of transition matrix T_* corresponding to the ideal random number generator (all entries of T_* are equal to 10^{-d}), allows us to compute similarity of each PRNG to the perfect case, i.e., rate its randomness (e.g., for ranking purposes).

3.5. Similarity Measure

Given empirical transition matrices of generated time series from different PRNGs, a similarity among the matrices shall provide a quantity measurement comparing the randomness among the PRNGs. Given two empirical transition matrices T_1 and T_2 of same size, various metrics can be applied to measure the similarity between them. As a proof of concept, we propose to investigate the metric based on *Euclidean distance*. The Euclidean distance is calculated by computing the difference in the L_2-norm after vectorization of the matrices. Thus, for the given matrices T_1 and T_2, we first copy all their entries row by row into respective vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively, and then compute their Euclidean distance $\|\mathbf{v}_1 - \mathbf{v}_2\|_2$. Using this metric, we expect that different realizations of the same PRNG will have small distances to each other due to the expected dissimilarity in the transition pattern.

3.6. Datasets

We conducted experiments with three distinct types of PRNGs in order to evaluate our methodology as well as to compare with classical statistical random methods and even to generated streams from the decimal digits of the irrational π. We created various PRNG time series ensembles, each of them containing sequences of integers $\{R_n\}_{n=0}^{N}$ with a range from 0 to $10^d - 1$. In this study, we focused on $d = 1$, but the same analysis could be applied for any d. The first dataset consists of chaos-based PRNGs, the second dataset is composed of numerical-based PRNGs, and the third dataset comprises sequences of π digits. Detail information about them is given as follows:

- Chaos-based PRNG dataset: We used both the K-Logistic map and the K-Tent map PRNGs [30] with different parameter settings K ($K \in \{0, 1, \ldots, 10\}$) to create time series of different lengths N ($N \in \{100, 1000, 10000, 1000000\}$). For each combination of parameter values K and lengths N, we generated 30 time series using different arbitrary seed points, leading to 1320 runs in total for each of the two PRNG approaches (K-Logistic map and K-Tent map).

- Numerical PRNG dataset: five numerical PRNGs were considered, namely LCG, RANDU, Mersenne Twister, SplitMix, and PCG. More information about their algorithms and implementations is given in Table 1. Similar to the chaos-based PRNG dataset, we also generated 30 series with different lengths ($N \in \{100, 1000, 10000, 1000000\}$) for each of the five listed PRNG algorithms. Thus, a total of 600 time series was generated. (For the sake of brevity, the Mersenne Twister and the SplitMix PRNGs names are abbreviated as MT and SM in the following.)

Table 1: Details about the numerical PRNGs used for the experiments.

PRNG	Description
LCG [24]	Implemented using Equation $x_{n+1} = (ax_n + c) \bmod m$ and parameters $a = 25,214,903,917$, $c = 11$, and $m = 2^{48} - 1$, which are the default values within the Java library `java.util.Random`.
RANDU [24]	It is a variation of LCG. We used the parameters $a = 65,539$, $c = 0$, and $m = 2^{31}$.
Mersenne Twister [35]	We used a Java implementation given in https://cs.gmu.edu/~sean/research/mersenne/MersenneTwister.java Copyright (c) 2003 by Sean Luke.
SplitMix [46]	We used the implementation from the Java SDK http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/SplittableRandom.java.
PCG [39]	We used a Java implementation given at https://github.com/KilianB/pcg-java.

- π-digits dataset: We also considered to use the sequence of π digits. We treated a sub-sequence of n digits at a time and divided the digits by 10^n such that every number obtained is within the range $[0.0, 1.0]$. We downloaded the first 240 million digits from Ref. [38]. For this particular dataset, we generated 30 samples with $N \in \{1000000\}$ using $n = 8$ as the length of each number.

4. Interactive Visual Analysis Framework

4.1. Analysis Tasks

Given an ensemble of PRNG runs, we want to analyze the quality of the PRNG runs based on their pseudo-randomness. First, we are interested in examining whether there are groups of PRNG runs that have similar quality. Thus, the first goal of the interactive visual analysis is to (T1) classify the PRNGs based on their pseudo-randomness. Then, we are interested in investigating, which PRNGs perform better than others. Thus, the second task is to (T2) rank the PRNGs by their pseudo-randomness quality. Classifying and ranking PRNG runs shall be performed in a way such that PRNG ensembles generated by varying a PRNG parameter can be analyzed to (T3) investigate the impact of PRNG parameters on the pseudo-randomness quality. In this regard, it is also of interest to (T4) relate the pseudo-randomness quality to the entropy measures. Finally, when interactively refining the analysis, all steps that were taken within an analysis session shall be recorded and made available to allow for reproducibility of the analysis process and for traversing the various zoom levels of the analysis interactively. Thus, the interactive visual analysis shall (T5) allow for visual documentations of the steps and findings made during the interactive sessions.

4.2. Visual Encodings and Interactions

Task T1: To address *Task T1*, we need to investigate the similarity of all ensemble members. Since many PRNG runs may be considered simultaneously, a manual investigation of all pairwise similarities would be very tedious. Visualization methods can help, as graphical representations are intuitive and can be efficiently processed by humans. Hence, our goal is to develop a visual encoding that represents all pairwise similarities with as few loss of information/precision as possible. Pairwise similarities of a set of objects can be depicted in the form of a similarity or dissimilarity matrix. However, such matrices are difficult to analyse when the number of objects is high. For better visualization and exploration, dimensionality reduction algorithms can be applied to generate embeddings of the high-dimensional similarity space into a low-dimensional (commonly two-dimensional) visual space. Scatterplots are classical representations of projected data in a form of a set of two-dimensional points. Many different embeddings aiming at preserving various characteristics of high-dimensional data in the projection space exist. The multi-dimensional scaling (MDS) [5] approach optimally preserves (dis)similarities of high-dimensional samples in the projection space. Dissimilarity of samples is computed as the Euclidean distances of points in the original or visual space. Thus, we apply a classical MDS approach to create 2D scatterplots, where each PRNG run is represented by a point in the projection space and the 2D (Euclidean) distances of the points reflect the dissimilarities of the respective PRNG runs (according to the dissimilarity measure defined in Section 3.5).

Figure 2 shows examples of dissimilarity plots for different sets on PRNG runs. When subsets of runs are more similar to each other than to other runs, they form clusters. The clusters can be observed visually and interactively selected using a lasso tool leading to a interactive classification. However, one can also observe, especially in Figure 2a, that points tend to clutter due to overplotting. This reason for the clutter is that runs belonging to one cluster are much more similar to each other than to runs not belonging to that cluster. While this is a desirable feature for interactive classification, it may be the case that a detected cluster actually consists of multiple sub-clusters. Hence, one would be interested in splitting the cluster into sub-clusters. We provide interaction mechanisms to support such an analysis: the user may interactively select any subset of points in the similarity plot and recompute the embedding by applying the MDS approach only to the selected subset. Thus, a user can interactively refine the visual representation and analyze substructures of already detected structures (cf. Figure 2 and its discussion in Section 5).

The interactive analysis sessions follow a top-down approach, where the user first investigate the entire ensemble to detect main structures, which are further refined in subsequent analysis steps. As a final step, one may be interested in seeing the temporal behavior of some selected runs in the form time series. This is supported by our system by selected quantities for selected runs and plotting them over the number of time steps. Figure 10 shows examples of plotting maximum entropy for selected chaos-based PRNG runs. Such plots allow for the investigation of whether the system has already converged to a stable entropy value or whether more computation steps would be necessary.

Task T2: While *Task T1* was concerned with identifying clusters of PRNG runs with similar pseudo-randomness quality, it was not yet investigated which of the clusters perform better than others. *Task T2* is

concerned with ranking PRNG runs. Hence, we would like to know how close each run is to the perfect case. We support this investigation by computing the empirical transition matrix for the perfect case as described in Section 3.3 and adding it to the ensemble before computing the similarity matrix and the embedding. Hence, the perfect case is now included in the similarity plot and distances to the perfect case provide a visual clue of the (approximate) ranking. In Figure 2 the perfect case is displayed using a bright red color. Points close to the bright red point indicate runs with high pseudo-randomness quality, while the quality (at least, approximately) decreases with increasing distance to the bright red point.

Task T3: This task is concerned with the analysis of the impact of PRNG parameters on the pseudo-randomness quality. Thus, we want to observe of systematic changes to the values of a parameter leads to an increase or decrease of the quality. We provide means for such an analysis by generating a color map for the parameter's value range and color-code the points in the similarity plot according to that color map. As the parameters are numerical values, one should choose a continuously changing color map. Perceptual studies showed that a luminance color map has preferable properties for intuitiveness. Since we may have multiple PRNG families with different parameter settings, we propose to use a single-hue luminance color map for each PRNG family, where the hue is different for different families. Figure 1 uses a red luminance color map for the investigated PRNG family, while Figure 6 uses two single-hue luminance color maps for investigating two PRNG families. To distinguish different PRNG families and further PRNG runs that do not belong to the families one should use hues that can be best distinguished. Since there is no order between families and other PRNG runs, a categorical color map should be used. We used categorical color maps by ColorBrewer [18] to generate the hues. A result can be found in Figure 8. When interactively hovering over a point in the similarity plot, there will also be a label popping up that displays the name of the PRNG method used to generate the run.

Task T4: While the similarity plots are based on the pseudo-randomness similarity, we may want to relate them to the PRNG entropy values, which was formulated as *Task T4*. These values can be added to the visual encoding by enhancing the point renderings. Entropy is a numerical attribute and different visual cues could be considered for mapping. Color and size of the points are easier to perceive and interpret than orientation, shape, or texture. Since the visual cue of color was already taken for mapping the parameter values, we decided to map the entropy values to the points' size. Figure 4 shows point sizes that vary by the maximum entropy values of the PRNG runs.

Task T5: Finally, *Task T5* is concerned with allowing for visual documentations of the findings made during the interactive sessions. As the interactive top-down approach allows for diving deeper into details of subsets of ensemble, it is important that the user is aware of the current level of detail that is provided and that the user can go back to an earlier intermediate result to continue from there. Also, the findings made during the analysis should be documented as well as which analysis steps were taken to get to the result, which is required for reproducibility of the findings.

The history of the analysis steps taken can be stored in a provenance graph. In the provenance graph, the nodes reflect the states of the interactive visual analysis, while the edges reflect the transformations made to get from one state to the next. For our top-down analysis strategy, a typical pattern is to start the analysis with the overview of all ensemble members, interactively select subgroups, reconfigure the similarity plot for selection, possibly repeat the selection and reconfiguration steps several times, before going back to an earlier step and investigating a different subgroup in the same manner. Hence, the provenance graph has a tree structure with the root being the overview visualization. We provide a visualization of the provenance graph for visual documentation and for interactively switching between any of the analysis steps. Figure 1 shows such a visualization of the provenance graph. The first analysis steps describe a linear path, which we depict as row. Any branching from this linear path then shows up as a new row with respective lines being drawn that depict the branching. The states of the interactive visual analysis are represented by the respective similarity plots. The transition between two states is shown by depicting the interaction that triggered the transition when performed on the similarity plot. For example, the first row of Figure 1 shows three pictures: the first represents the overview, which is the state before the transition; the second is the transition in the form of a selection that triggers a reconfiguration; and the third shows the state after the transition. Using the provenance graph visualization, the user can easily go back to earlier steps in the analysis and continue from there, while all steps and findings are being kept in the provenance graph. The

provenance graph can be saved and loaded to a new session at a later time.

5. Results and Discussion

In this section, we apply the interactive visual analysis approach of Section 4 based on the theoretical concepts described in Section 3 for the classification and ranking of PRNG runs. We analyze the influence of PRNG parameters and compare the different PRNG methods discussed above (Section 5.1). We further discuss the number of time steps computed for each PRNG run (Section 5.2). Finally, we also compare our findings with results obtained by the DIEHARD randomness test (Section 5.3).

5.1. PRNG Classification and Ranking

5.1.1. K-Logistic map

In a first experiment, we want to analyze the influence of the parameter K on the randomness quality of the K-Logistic map. Hence, we compute the similarity map for all K-Logistic map runs, where parameter K is color-coded using a single-hue luminance color map, see Figure 2a. The entropy measure is encoded by the point size. Furthermore, we compute and add the perfect case shown with a bright red color. In Figure 2a, we observe that the PRNG runs form five clusters. We systematically analyze all five clusters and their sub-clusters. The provenance graph of the analysis is shown in Figure 1. All the individual similarity plots that are analyzed within this interactive sessions are shown in Figure 2 including some labels and the color legend.

The five clusters in Figure 2a are analyzed in a systematic way by decreasing distance to the perfect case, i.e., in the reverse order of their rankings. The first cluster includes all the runs with parameter $K = 0$. The first observation here is that the runs with $K = 0$ produce the lowest randomness quality. The second observation is that all runs with $K = 0$ group in one cluster, i.e., the impact of the choice of the seed point is negligible when compare to the choice of K. We will see that this holds true for all K. This was to be expected from a theoretical point and shows that our approach was able to validate this expectation. A third observation is that the entropy for $K = 0$ is much smaller than for all other runs.

Similarly, we can investigate the second, third, and fourth cluster. We observe that they consist of runs for $K = 1$, $K = 2$, and $K = 3$, respectively, see Figures 2c, 2d, and 2e. Thus, the different values of K form homogeneous clusters and higher values of K lead to higher randomness quality.

When analyzing the fifth cluster, which includes the perfect case, we observe that the cluster exhibits four sub-clusters, see Figure 2f. As before, we systematically analyze all four sub-clusters by their decreasing distance to the perfect case as documented by the provenance graph in Figure 1. The first, second, and third sub-cluster correspond to the runs for $K = 4$, $K = 5$, and $K = 6$, respectively, see Figures 2g, 2h, and 2i.

The fourth sub-cluster contains more run including the perfect case, see Figure 2j. There are no very obvious "sub-sub-clusters", but two clusters can still be distinguished, see Figures 2k and 2l as well as the respective selections in the provenance graph in Figure 1. The first sub-sub-cluster in Figure 2k consists mainly of the runs with $K = 7$, while the second sub-sub-cluster in Figure 2l mainly consists of the runs with $K = 8$, $K = 9$, and $K = 10$ as well as the perfect case. In the second sub-sub-cluster, there is no visible separation of the runs with $K = 8$, $K = 9$, and $K = 10$.

In general, though, we can report that runs with the same parameter K form clusters and that the randomness quality increases with increasing K.

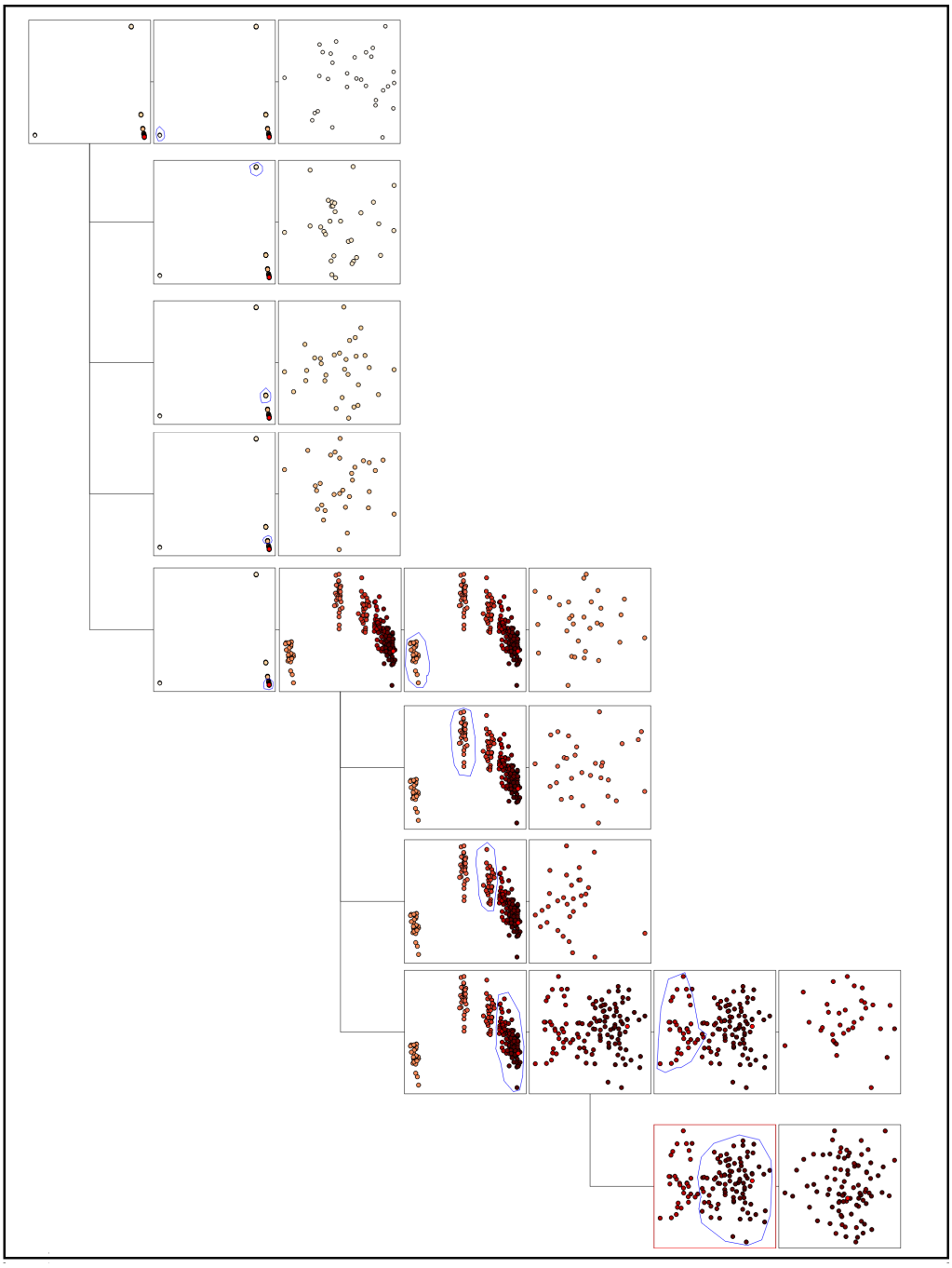


Figure 1: Provenance graph for interactive visual analysis session of K-Logistic map PRNG ensemble. The analyzed similarity plots are shown in Figure 2

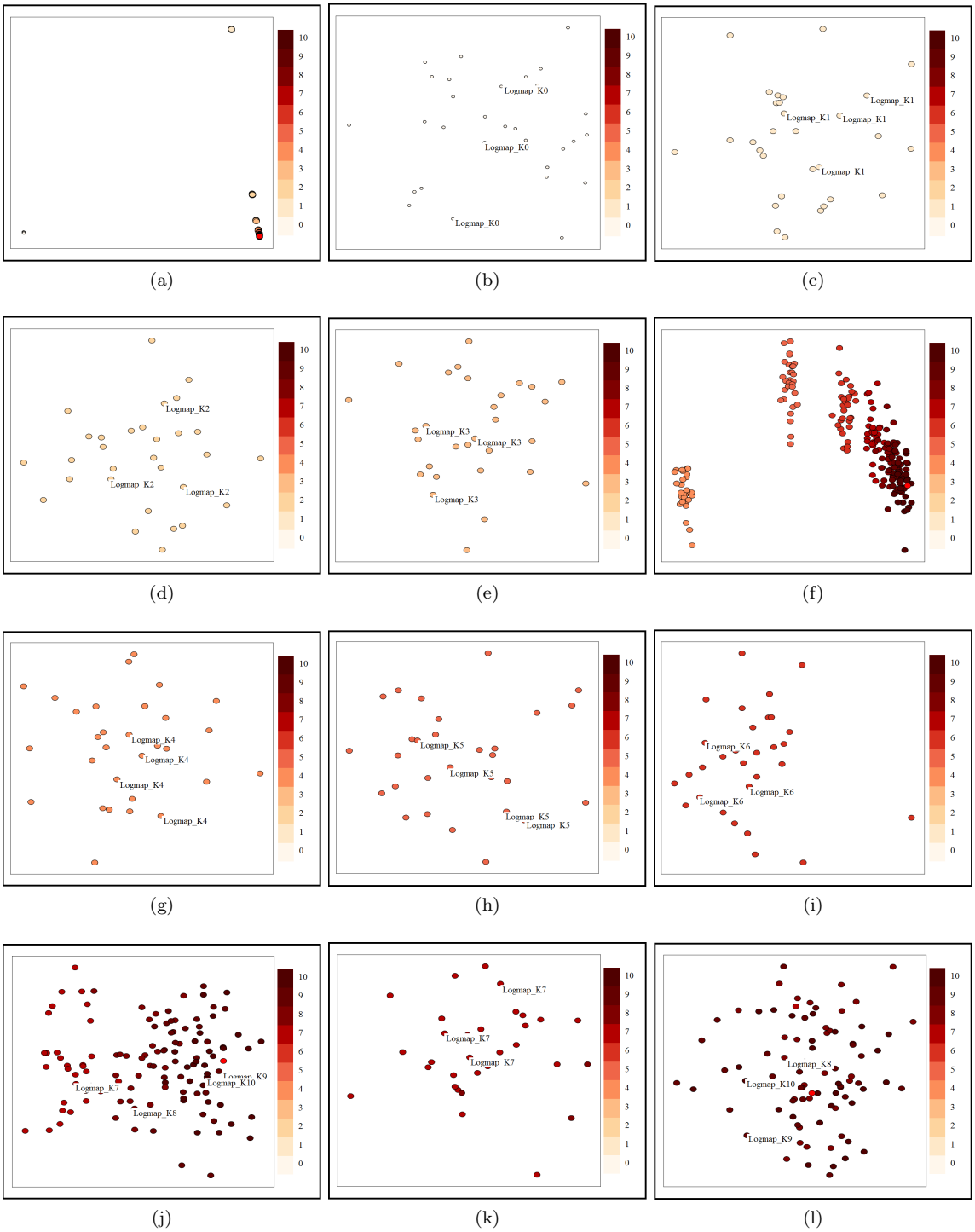


Figure 2: Similarity maps for classifying and ranking K-Logistic map PRNG runs using a single-hue color map for parameter K. Entropy is encoded by size. Perfect case of randomness quality is shown in bright red. The runs split into 5 clusters (a), of which 4 clusters correspond to runs for $K = 0$ (b), $K = 1$ (c), $K = 2$ (d), and $K = 3$ (e), while the fifth cluster exhibits 4 sub-clusters (f). 3 of the sub-clusters correspond to runs for $K = 4$ (g), $K = 5$ (h), and $K = 6$ (i), while the fourth sub-cluster contains the remaining runs (j). The latter roughly splits into a sub-sub-cluster containing runs for $K = 7$ (k) and a sub-sub-cluster containing runs for $K = 8$ to $K = 10$ and the perfect case. Thus, the higher K, the closer one gets to the perfect case. Provenance graph for interactive visual analysis session is shown in Figure 1.

5.1.2. K-Tent map

In our second experiment, we want to analyze the influence of the parameter K on the randomness quality of the K-Tent map. Hence, we compute the similarity map for all K-Tent map runs, where parameter K is color-coded using another single-hue luminance color map, see Figure 4a. As above, the entropy measure is encoded by the point size and we compute and add the perfect case shown with a bright red color. In Figure 4a, we observe that the PRNG runs form, again, five clusters. We systematically analyze all five clusters and their sub-clusters by decreasing distance to the perfect case. The provenance graph of the analysis is shown in Figure 3. All the individual similarity plots that are analyzed within this interactive sessions are shown in Figure 4 including some labels and the color legend.

The first cluster in Figure 4b includes all the runs with parameter $K = 0$. Again, runs with $K = 0$ form a cluster, produce the lowest randomness quality, and have a much smaller entropy value than other runs.

The second cluster forms two sub-cluster as shown in Figure 3. The first sub-cluster consists of runs for $K = 1$, $K = 2$, $K = 3$, $K = 4$, and $K = 5$, see Figure 4c. No further sub-sub-clusters can be observed. The second sub-cluster consists of runs for $K = 6$, see Figure 4d.

The third and fourth cluster consists of runs for $K = 7$ and $K = 8$, respectively, see Figures 4e and 4f, while the fifth cluster contains runs for $K = 9$ and $K = 10$ with no sub-sub-clusters, see Figure 4g.

Overall, we can, again, report that runs with the same parameter K tend to form clusters and that the randomness quality tends to increase with increasing K.

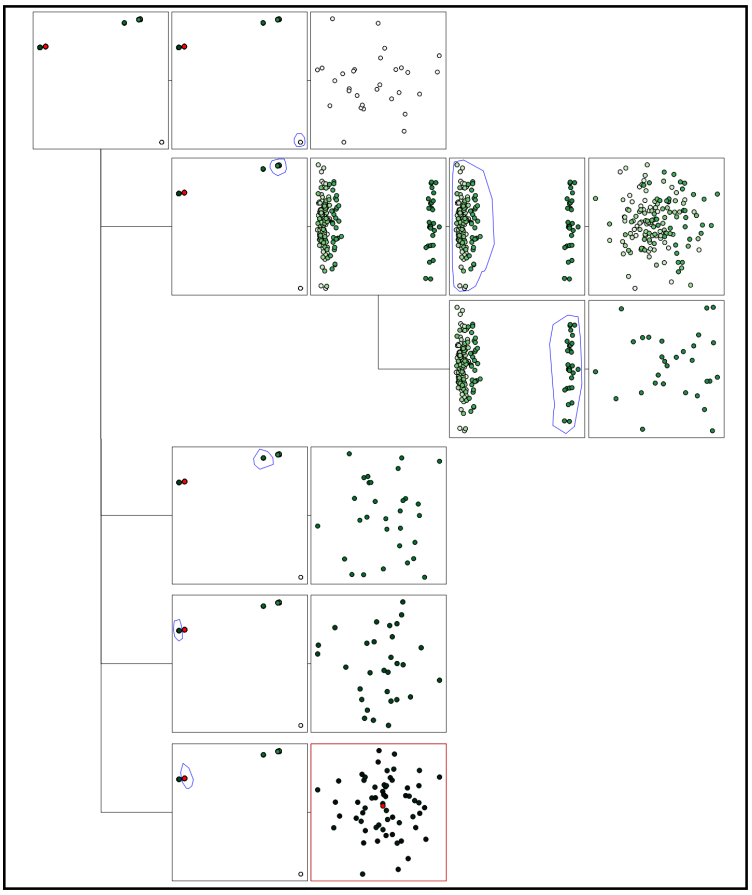


Figure 3: Provenance graph for interactive visual analysis session of K-Tent map PRNG ensemble. The analyzed similarity plots are shown in Figure 4.

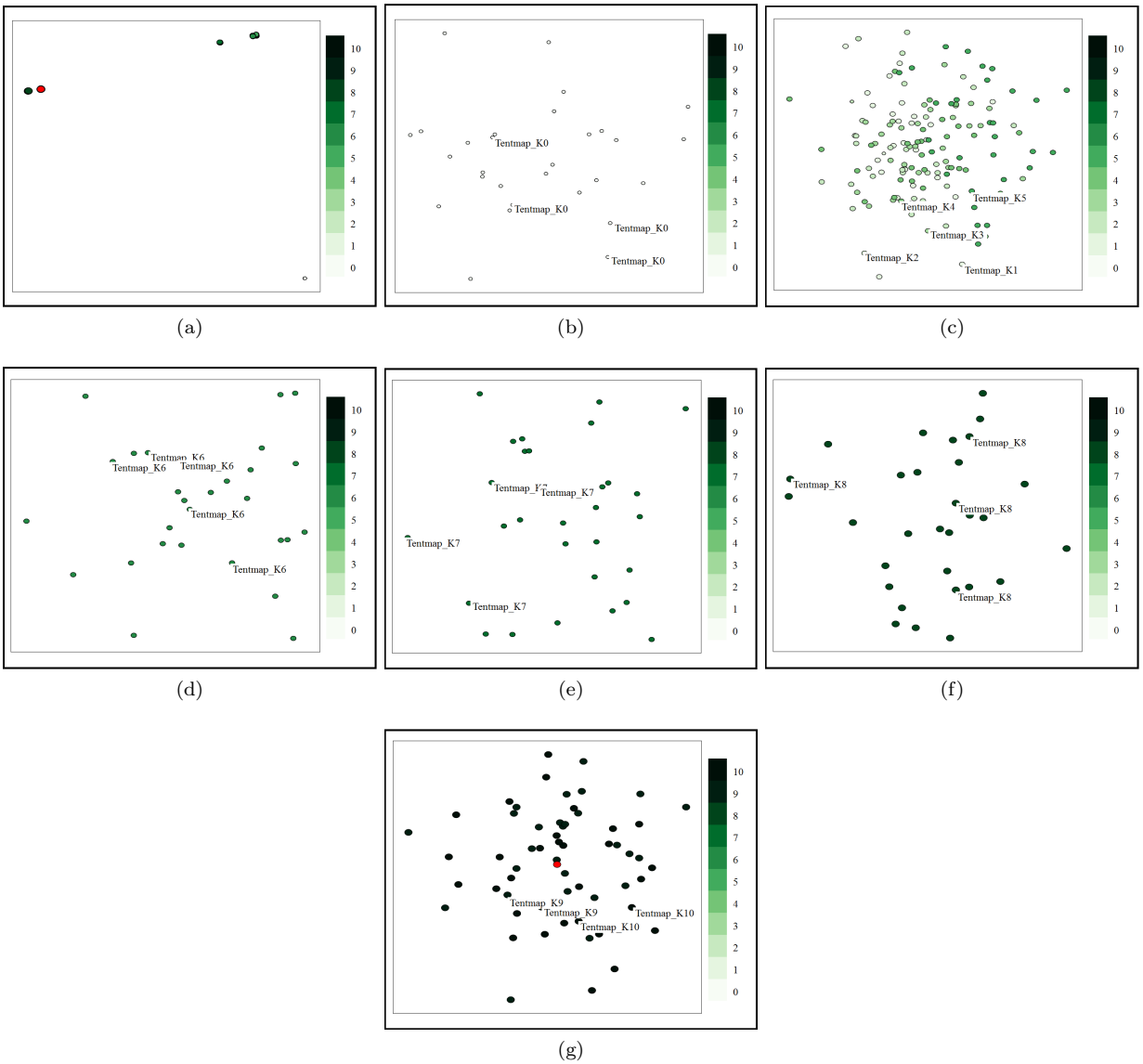


Figure 4: Similarity maps for classifying and ranking K-Tent map PRNG runs using a single-hue color map for parameter K. Entropy is encoded by size. Perfect case of randomness quality is shown in bright red. The runs split into 5 clusters (a), which correspond to runs for $K = 0$ (b), $K = 1$ to $K = 6$ with two sub-clusters (c,d), $K = 7$ (e), $K = 8$ (f), and $K = 9$ to $K = 10$ plus the perfect case. Thus, the higher K, the closer one gets to the perfect case. Provenance graph for interactive visual analysis session is shown in Figure 3.

5.1.3. Chaos-based PRNGs

In our third experiment, we compare the two chaos-based PRNGs, i.e., the K-Logistic map and the K-Tent map, against each other. Figure 6a shows the respective similarity map with the same two single-hue color maps as before and the perfect case in bright red. We observe a cluster around the perfect case as well as some clusters "above" the perfect case and some clusters "to the right" of the perfect case.

We first select all clusters to the right of the perfect case as shown in the provenance graph in Figure 5. The respective similarity map in Figure 6b reveals that the selected clusters include all runs from the K-Tent map for $K = 0$ to $K = 8$.

Then, we select all clusters above the perfect case as shown in the provenance graph in Figure 5. The respective similarity map in Figure 6c reveals that the selected clusters include all runs from the K-Logistic map for $K = 0$ and $K = 1$.

Eventually, we select the cluster that contains the perfect case. The respective similarity map in Figure 6d exhibits five sub-clusters, of which four contain the runs from the K-Logistic map for $K = 2$ to $K = 5$, while the fifth contains the perfect case is selected for further investigation. This sub-cluster contains three less separated sub-sub-clusters that correspond roughly to the runs from the K-Logistic map for $K = 6$ and $K = 7$, respectively, and the one that contains the perfect case, see Figure 6e. Investigating the latter as in Figure 6f shows a mixture of runs from the K-Tent map as well as the K-Logistic map for large values of K around the perfect case.

We can conclude, that the two chaos-based PRNG methods approach the ideal case "from different directions" in the similarity plot when K increases. It can also be observed that the K-Logistic map quicker approaches the perfect case for increasing K when compared to the K-Tent map. Finally, we observe that for large K, i.e., $K = 9$ or $K = 10$, both PRNG families have a similar randomness quality.

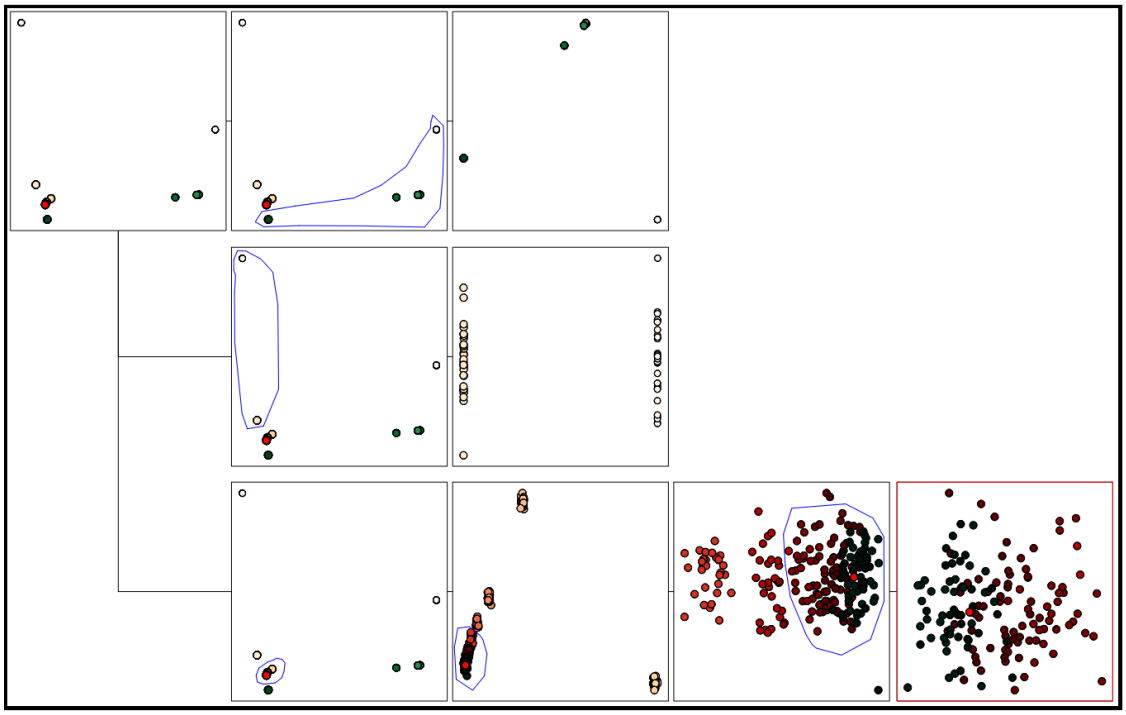


Figure 5: Provenance graph for interactive visual analysis session of comparing K-Logistic maps and K-Tent maps. The analyzed similarity plots are shown in Figure 6

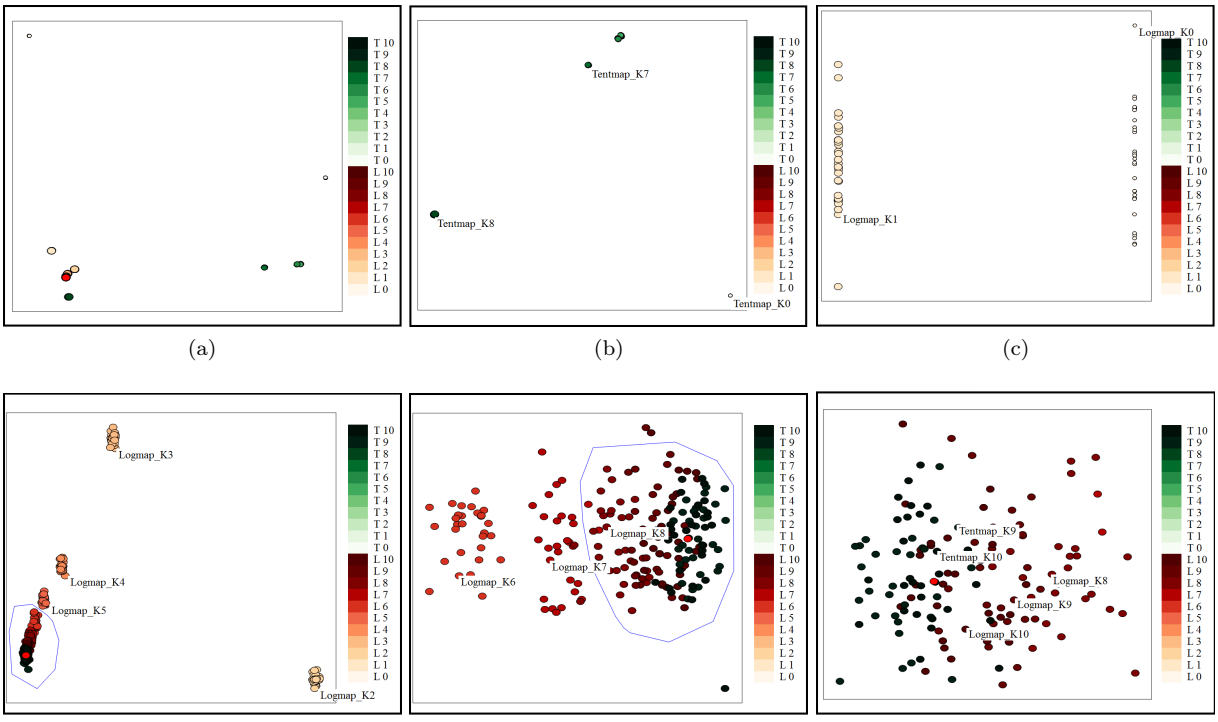


Figure 6: Similarity maps for comparing K-Logistic map and K-Tent map PRNG runs using two single-hue color maps for parameter K. Entropy is encoded by size. Perfect case of randomness quality is shown in bright red. The runs split into multiple clusters (a), of which the ones to the right of the perfect case correspond to K-Tent map runs for $K = 0$ to $K = 8$ (b), the ones above the perfect case correspond to K-Logistic map runs for $K = 0$ and $K = 1$ (c), and the cluster surrounding the perfect case splits into 5 sub-clusters (d). The sub-clusters correspond to K-Logistic map runs for $K = 2$, $K = 3$, $K = 4$, and $K = 5$, respectively, and a bigger sub-cluster. The latter roughly splits into a sub-sub-cluster containing K-Logistic map runs for $K = 6$ and $K = 7$, respectively, and one containing all runs from K-Logistic map and K-Tent map for large values of K as well as the perfect case. K-Logistic map converges faster to the perfect case with increasing K, but for large K the two approaches are comparable with respect to randomness quality. Provenance graph for interactive visual analysis session is shown in Figure 5.

5.1.4. Comparing Chaos-based to Numerical PRNGs

We also compare the chaos-based PRNG families to common and classical numerical PRNG methods. The similarity map in Figure 8a is similar to the one we had seen in Figure 6a. We used the same colors as before, but added the numerical PRNG runs using the concept of a categorical color map. Given the similar structure of the similarity maps, we perform a similar analysis to the one above. Figure 7 shows the provenance graph of the interactive analysis session.

The first group of clusters "to the right" of the perfect case consists again of K-Tent map runs for small and middle-range values of K, see Figure 8b. The second group of clusters "above" of the perfect case consists again of K-Logistic map runs for small K, see Figure 8c. Finally, the cluster that contains runs close to the perfect case splits into four sub-clusters with K-Logistic map runs the middle-range values of K and the fifth sub-cluster containing the perfect case, see Figure 8d. The latter sub-cluster contains again some less separable sub-sub-clusters for K-Logistic map runs with $K = 7$ and $K = 8$ as well as the sub-sub-cluster containing the perfect case, see Figure 8e. This sub-sub-cluster contains again the chaos-based PRNG runs for large K, but also basically all the numerical PRNG runs.

Hence, we can conclude that chaos-based PRNGs for large K have a randomness quality comparable to that of numerical PRNGs.

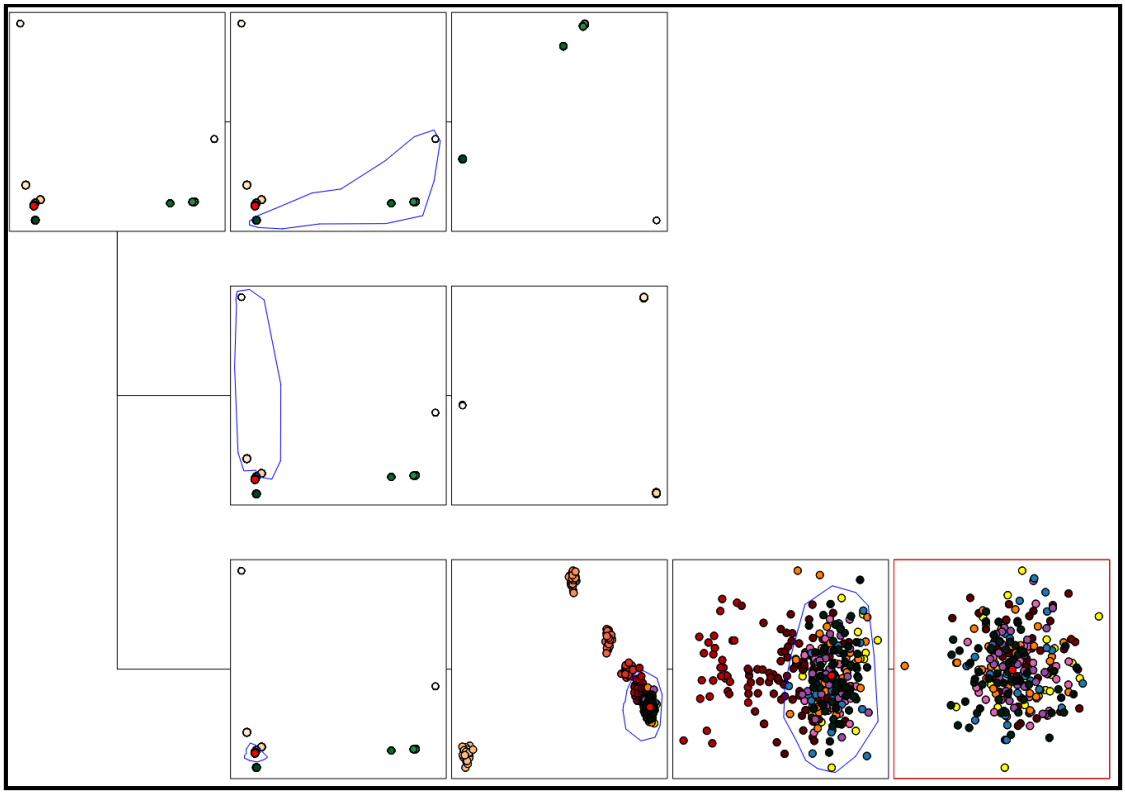


Figure 7: Provenance graph for interactive visual analysis session of comparing chaos-based PRNGs and numerical PRNGs. The analyzed similarity plots are shown in Figure 8

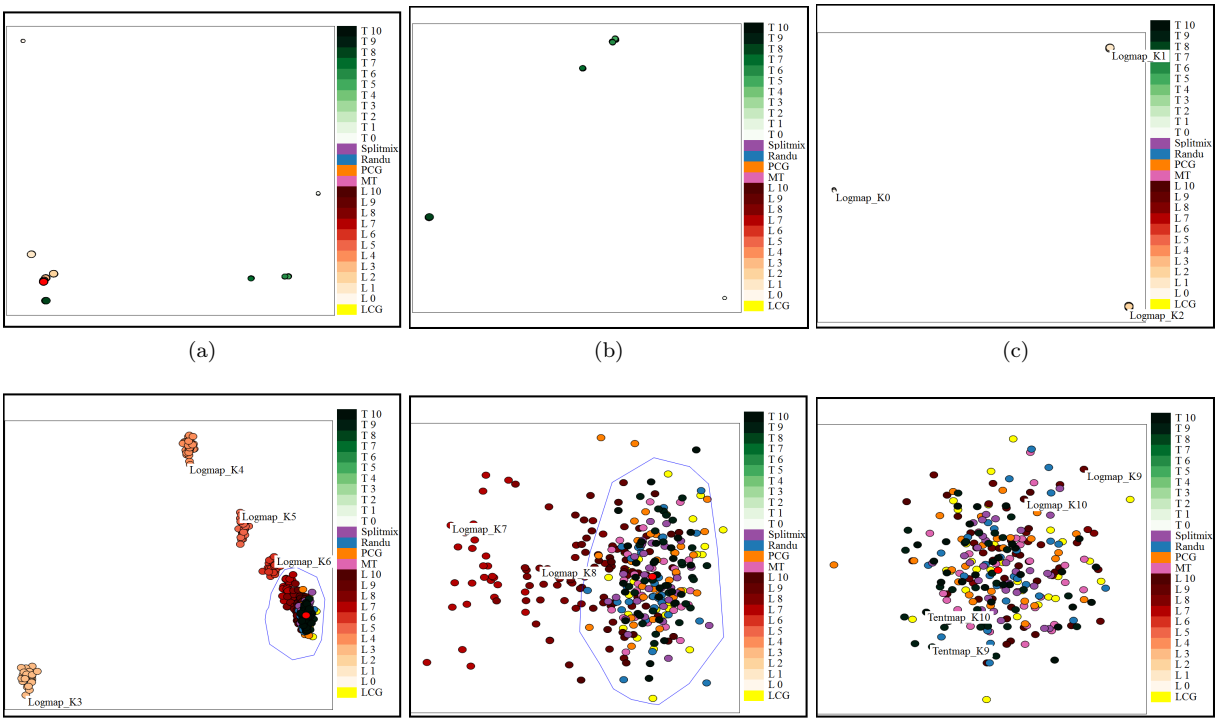


Figure 8: Similarity maps for comparing chaos-based and numerical PRNG runs using two single-hue color maps for parameter K of chaos-based approaches and categorical color map for assigning hues. Entropy is encoded by size. Perfect case of randomness quality is shown in bright red. The structure of the similarity maps is similar to Figure 6. The numerical PRNG runs are mainly contained in the sub-sub-cluster that also contains the perfect case (f). For large K, the chaos-based PRNGs have randomness quality comparable to numerical PRNGs. Provenance graph for interactive visual analysis session is shown in Figure 7.

5.1.5. Comparing Chaos-based PRNGs with π-digits as Random Sequence

Although answering the old and still open question of whether the digit expansion of π is random or only appears to be random [3, 11, 19] goes beyond that scope of this work, here we are intrigued to compare the chaos-based PRNG with the PRNG based on the π-digits dataset as proposed in Section 3.6. Thus, we applied our method to the random numbers derived from π, see Figure 9. We observe that the π-digits sequence is comparable with K-Logistic map for $K = 9$ and $K = 10$ in terms of randomness quality and is superior to K-Logistic map from $K = 8$ downwards. Moreover, thd π-digit dataset tends to be close to the perfect-case.

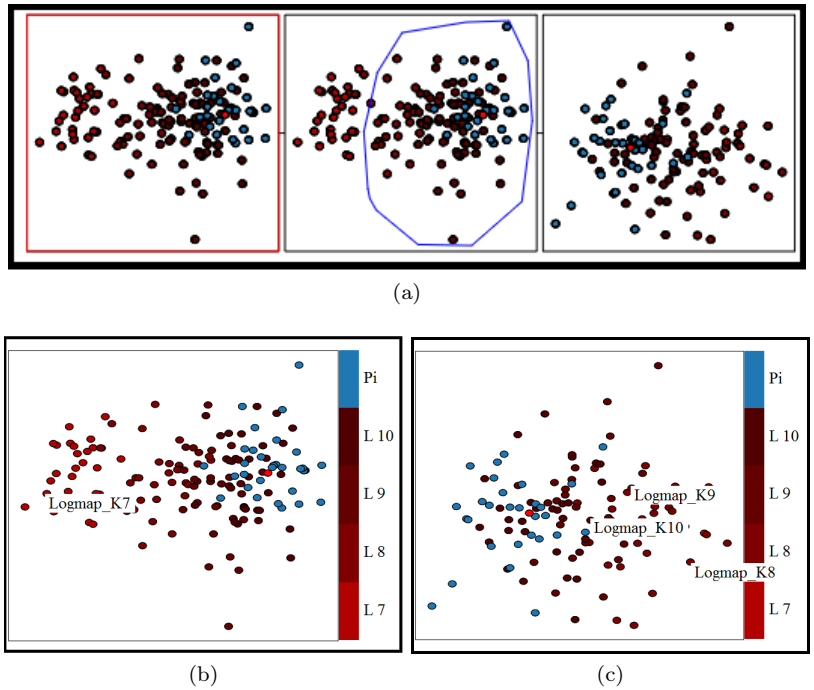
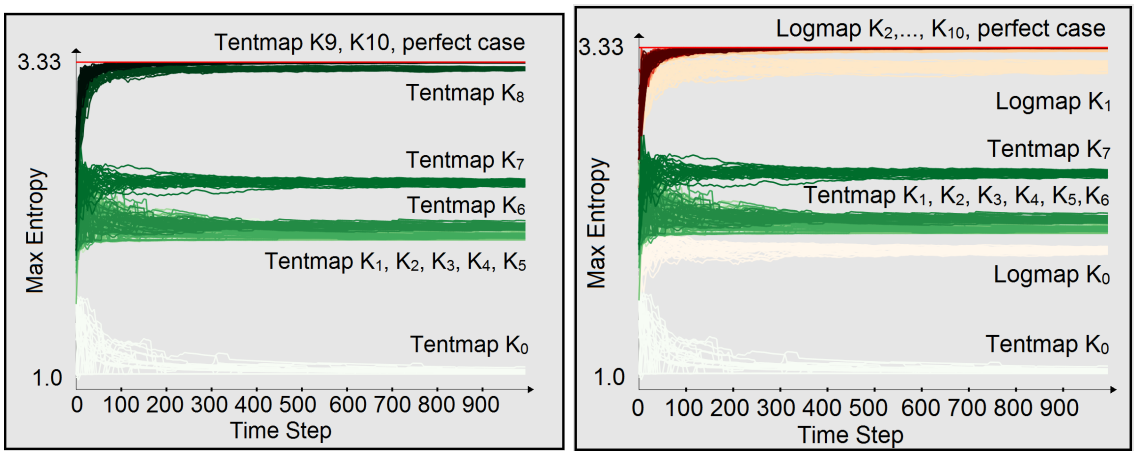


Figure 9: (a) Provenance graph for interactive visual analysis session of comparing K-Logistic maps for $K = 7$ to 10 with π-digits dataset and (b-c) respective similarity maps. The first map (b) contains all runs from the K-Logistic map for $K = 7$ to 10, the π-digit samples, and the perfect case. The second map (c) excludes samples for $K = 7$. The π-digits dataset is comparable with the K-Logistic map for $K = 9$ and $K = 10$ with respect to randomness quality.

5.2. Convergence Analysis

For our computations, we needed to restrict ourselves to a finite number of time steps. The question that remained was how many time steps would suffice to capture the randomness characteristics of the PRNGs. To address this question, we apply our approach for data of ascending number of time steps and plot the maximum entropy over time. Figure 10 shows the temporal pattern for maximum entropy for the chaos-based PRNGs. We observe some fluctuations at the beginning of the time series, but they stabilize quite quickly. After 900 time steps, we can state that the maximum entropy for all runs is stable. Hence, $N = 900$ was a sufficiently large choice. Larger values of N do not change the behavior.

Figure 10 also compares the entropy time curves for K-Tent map (a) and K-Logistic map runs (b). We had observed in the preceding section that the randomness quality for K-Logistic map increased faster for increasing K. Here, we observe a matching behavior for maximum entropy. Hence, the entropy values for K-Logistic map are also quicker increasing towards the perfect case with increasing K. In more detail, we observe that for $K = 0$ and $K = 1$, the K-Logistic map curves lie clearly above the respective K-Tent map curves. For $K = 2$ on, the K-Logistic map curves approximate the perfect case, while for the K-Tent map curves this is only true for $K = 9$ and $K = 10$. Hence, the findings match those of the similarity maps.



(a) (b)

Figure 10: Max Entropy as a function of time steps for the chaos-based PRNGs. (a) K-Tent map for K_0 to K_{10}. (b) K-Logistic map for K_0 to K_{10}.

The convergence plots in Figure 10 used the maximum entropy. We also proposed to investigate average and minimum entropy. According to our observation with the given datasets, the three entropy measures lead to equivalent results. Specifically, given two empirical transition matrices of PRNGs, the result of the comparison among two matrices for the given datasets is the same using any of the three entropy measures. Still, we believe that for other datasets, there may be differences among the entropy measures. Thus, we decided to provide the description of all three options in this paper.

We also examined the provenance graphs for increasing number of time steps. We observed that larger number of time steps lead to a better capability in separating PRNG groups of similar randomness. Therefore, in all figures of similarity plots and provenance graphs presented in this paper, we used a large number of time steps. We consistently used 1000000 time steps for all the generated visualizations.

5.3. Comparison to DIEHARD Randomness Tests

The DIEHARD [34] tests comprise of a number of statistical tests to measure the randomness quality of PRNGs. It consists of 18 tests enclosing 221 sub-tests in total. Each of these tests are designed to analyze an input file (a binary sequence generated by a PRNG), for instances, by checking overlapping windows of bits, blocks of bits, and so on. According to the DIEHARD manual [34], each of these tests is formulated to test a null hypothesis ($H0$: the input file being tested is sufficiently random) associated with an alternative hypothesis ($H1$: the input file is not sufficiently random). Then, the χ^2 goodness-to-fit technique is used to compute a probability p-value, which should be uniform on the interval $[0, 1)$. For each test applied, a decision is derived that accepts or rejects the null hypothesis. It is assumed that these p-values follow a normal distribution. Thus, a test is *passed* when the p-value satisfies $0 \ll p \ll 1$, i.e., the file is considered statistically random. Otherwise, the test is *failed* when the p-value is close to 0 or 1, i.e., it deviates from normal distribution, which implies that the input file is not sufficiently random. However, it should be noted that even for good PRNGs, it may happen that p-values fall into the "fail" region, cf. [34].

To carry out the DIEHARD experiment, we generated 100 files each containing 11.2 MB, equivalent to 28×10^5 numbers, for each of the PRNGs studied in this paper. The DIEHARD tests require file sizes of more than 10MB [34]. Each input file was generated from a random seed, when necessary was transformed by a 32 bit mask, and written in a bit stream format.

The results of the DIEHARD experiment are reported in Figure 11a. For the sake of clarity, we only show some sub-tests (see accompanying supplementary material for the complete table). The table includes values for the numerical PRNGs as well as the chaos-based PRNGs (K-Logistic map and K-Tent map with $K = 0$ (K0) to $K = 10$ (K10). Each column shows the number of files whose p-values passed a corresponding

test (rows). It is assumed that a high number of passed files indicates a high quality in terms of statistical randomness tests.

(a)

Diehard tests						LOGISTIC MAP											TENT MAP										
	RANDU	LCG	SM	MT	PCG	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
BirthdaySpacings	85	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	99	100	100	100
OverlappingPermutations	4	98	97	98	92	96	98	96	97	97	99	98	96	98	97	96	96	99	99	96	99	100	100	97	96	98	97
Ranks31x31	91	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Ranks32x32	91	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Ranks6x8	0	100	99	100	100	0	0	25	99	100	100	100	100	100	100	100	0	0	0	100	100	100	100	100	100	100	100
Monkey20bitWords	0	100	100	100	100	0	99	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
OPSO	0	58	100	100	100	98	99	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
OQSO	0	0	100	100	100	98	100	100	100	100	100	100	100	100	100	100	0	0	0	100	100	100	100	100	100	100	100
DNA	0	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Count1sStreamBytes	0	100	100	100	100	0	0	0	98	100	100	100	100	100	100	100	0	0	0	0	100	100	100	100	100	100	100
Count1sSpecificBytes	96	100	100	100	100	0	0	0	0	94	100	100	100	100	100	100	0	0	0	0	0	0	100	100	100	100	100
ParkingLot	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
MinimumDistance	0	100	100	100	100	96	100	100	100	100	100	100	100	99	100	100	81	95	94	100	98	98	97	99	98	99	97
RandomSpheres	57	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	99
Sqeeze	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
OverlappingSums	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Runs	1	100	100	100	100	100	100	100	100	100	100	99	100	100	100	100	100	100	99	99	100	100	100	100	100	100	100
Craps	0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

(b)

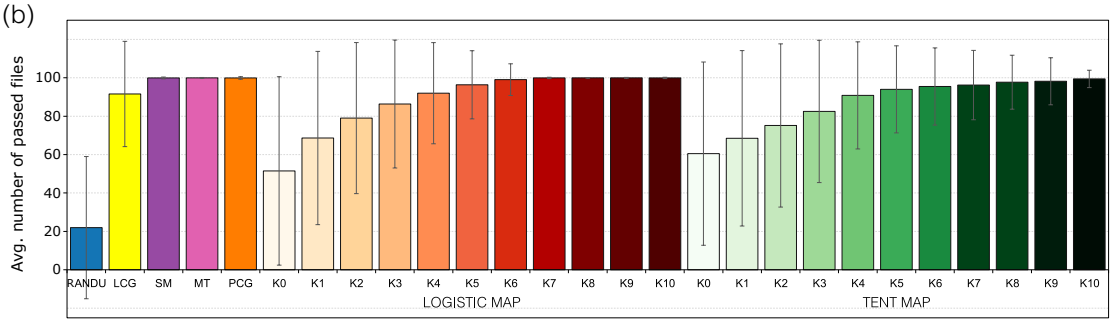


Figure 11: (a) Number of files that passed DIEHARD tests using the PRNGs studied from 100 input files. Severely failed tests are shown in gray. All tests passed using the interval $0.0001 <$ p-value < 0.9999. (b) Bar chart of average number of files that passed the 221 DIEHARD test including standard deviation (whiskers).

A large number of failed sub-tests can mainly be observed for RANDU and LCG as well as for the chaos-based methods when $K = 0$. Concerning the K-Logistic map, it shows major fails for $K = 0$ to $K = 3$, e.g., for *Ranks6x8*, *Monkey20bitWords*, *Count1sStreamBytes*, and *Count1sSpecificBytes*, where the number of fails decrease for increasing K. A similar behavior is observed for K-Tent map for $K = 0$ to $K = 5$. On the other hand, it can also be observed that most of the 100 files were successfully passed for K-Logistic map when $K \geq 4$ and for K-Tent map for $K \geq 6$, which was already reported in literature [30]. For the numerical PRNGs, it can be observed that all but RANDU and LCG perform quite well for all tests.

The outcome of all 221 tests can be summarized in a bar chart as shown in Figure 11b, where the bars indicate the average number of passed files and the whiskers display the standard deviation. We observe that it is hard to draw conclusions from here. For example, we cannot distinguish SM, MT, PCG, K-Logistic map with $K > 6$, or K-Tent map with $K > 8$. In fact, the only conclusion we can make from statistical randomness tests is that many PRNGs cannot be distinguished at all, which can also be verified using a pairwise Wilcoxon test. This observation shows the need for further methods such as the one we present in this paper.

In order to make a fair comparison between pairs of PRNGs we used the Wilcoxon T-test for comparing paired data samples. We used the complete results from DIEHARD, i.e., 221 values for each PRNG. By using this strategy we aim to verify if there is a significant difference between the number of files that passed the tests in relation to the performance of the other PRNGs, the Wilcoxon test was applied, which is a non-parametric statistical hypothesis test to determine whether two data samples have the same or different distributions. To perform this test, a significance level of 5% was assigned to investigate if the results of

the Wilcoxon test exhibit a significant difference between the performances of the PRNGs studied. Table 2 shows the outcome of the statistical tests in a matrix, where an asterisk (*) indicates that the method listed on the left-hand side (row) obtained significantly better DIEHARD results than the respective method listed on top (column). For examples, we can read from the table that LCG outperformed the K-Logistic map for $K = 1$ and $K = 2$ as well as the K-Tent map for $K = 1$, $K = 2$, and $K = 3$ in terms of randomness. On the other hand, K-Logistic map for $K = 5$ is superior to LCG but not to SM, MT or PCG.

	LCG	SM	MT	PCG	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10
RANDU																										
LCG						*	*										*	*	*							
SplitMix	*				*	*	*	*	*	*						*	*	*	*	*	*	*	*			
MersenneT	*				*	*	*	*	*	*	*					*	*	*	*	*	*	*	*	*	*	
PCG	*				*	*	*	*	*	*	*					*	*	*	*	*	*	*	*			
LM-K0																										
LM-K1					*											*										
LM-K2					*	*										*	*									
LM-K3					*	*	*									*	*	*	*							
LM-K4					*	*	*	*								*	*	*	*							
LM-K5	*				*	*	*	*	*							*	*	*	*	*						
LM-K6	*				*	*	*	*	*	*						*	*	*	*	*	*	*	*			
LM-K7	*				*	*	*	*	*	*	*					*	*	*	*	*	*	*	*	*	*	
LM-K8	*				*	*	*	*	*	*	*					*	*	*	*	*	*	*	*	*	*	*
LM-K9	*				*	*	*	*	*	*	*					*	*	*	*	*	*	*	*	*	*	*
LM-K10	*				*	*	*	*	*	*						*	*	*	*	*	*	*	*			
TM-K0					*																					
TM-K1					*											*										
TM-K2					*	*										*	*									
TM-K3					*	*	*									*	*	*								
TM-K4					*	*	*	*								*	*	*	*							
TM-K5					*	*	*	*								*	*	*	*	*						
TM-K6					*	*	*	*	*							*	*	*	*	*						
TM-K7	*				*	*	*	*	*							*	*	*	*	*	*					
TM-K8	*				*	*	*	*	*							*	*	*	*	*	*					
TM-K9	*				*	*	*	*	*							*	*	*	*	*	*					

Table 2: Paired comparison of PRNGs using Wilcoxon test applied to the number of files that passed the DIEHARD test. Asterisks (gray) indicate which PRNG on the left-hand side (rows) obtained significantly different distributions of samples from the PRNG on top (columns).

In summary, when comparing the outcome of Figure 8 to the outcome in Table 2, neither the proposed method nor existing statistical randomness methods allowed us to distinguish the numerical PRNG methods. However, our tool allowed us to separate the different families of K-Logistic map and K-Tent map even until $K = 8$ in some cases.

6. Conclusion

We have presented an interactive visual analysis approach that addresses the issue of classical pass/fail randomness tests failing to provide a fine granularity to distinguish and rank PRNGs from weak to strong. We proposed a method that allows for checking randomness quality by clustering families of PRNGs when generated time series are provided. Additionally, our work can be extended to seek for randomness on sequences of irrational digits such as π, e, $\sqrt{2}$, $\sqrt{3}$. Our experiments have focused specifically on the chaos-based PRNGs K-Logistic map and K-Tent map. With our tool, we have found that their corresponding clusters get far apart from each other and converge to the "perfect" case as K is increased. This same fact has been previously analytically demonstrated [31]. In this work, we not only reinforced this statement but also provided means to distinguish among families of PRNGs. However, classical PRNGs with good randomness properties are still hard to distinguish. The developed visualization tool places samples corresponding to

such PRNGs very close to the "perfect" case. This finding is subject to further investigations. Perhaps it retains for them to be good randomness sources until a new method says otherwise.

Acknowledgement

Jeaneth Machicao acknowledges the scholarship from the National Council for Scientific and Technological Development (CNPq grant #155957/2018-0) and the São Paulo Research Foundation (FAPESP #2020/03514-9). The research of Jeaneth Machicao at the Institute of Computer Science, Department of Mathematics and Computer Science of the University of Münster was supported by a grant from the Brazil Center of the University of Münster, under the auspices of the German Academic Exchange Service (DAAD) and the German Federal Ministry of Education and Research. The research of Quynh Quang Ngo at the Institute of Computer Science, Department of Mathematics and Computer Science of the University of Münster was supported by a travel grant from the Brazil Center of the University of Münster, under the auspices of the German Academic Exchange Service (DAAD) and the German Federal Ministry of Education and Research. This work was also supported in part by DFG grant MO 3050/2-1. Odemir M. Bruno acknowledges support from CNPq (grant #307897/2018-4) and FAPESP (grant #16/18809-9).

References

[1] W. Aigner, S. Miksch, H. Schumann, C. Tominski, Visualization of Time-Oriented Data, Springer Publishing Company, Incorporated, 1st edn., ISBN 0857290789, 2011.

[2] A. B. Alencar, F. V. Paulovich, R. Minghim, M. G. de Andrade Filho, M. C. F. de Oliveira, Similarity-Based Visualization of Time Series Collections: An Application to Analysis of Streamflows, in: 2008 12th International Conference Information Visualisation, IEEE, 280–286, 2008.

[3] D. H. Bailey, J. M. Borwein, R. P. Brent, M. Reisi, Reproducibility in Computational Science: A Case Study: Randomness of the Digits of Pi, Experimental Mathematics 26 (3) (2017) 298–305.

[4] F. Barbosa, A. Vidal, F. Mello, Machine Learning for Cryptographic Algorithm Identification, Journal of Information Security and Cryptography (Enigma) 3 (1) (2016) 3.

[5] I. Borg, P. Groenen, Modern Multidimensional Scaling: Theory and Applications, Springer, 2005.

[6] R. Bost, R. A. Popa, S. Tu, S. Goldwasser, Machine learning classification over encrypted data., in: NDSS, vol. 4324, 4325, 2015.

[7] P. Bratley, B. Fox, L. Schrage, A guide to simulation, Springer Verlag, New York, 1987.

[8] J. Eichenauer, J. Lehn, A non-linear congruential pseudo random number generator, Statistische Hefte 27 (1) (1986) 315–326.

[9] F. Fan, G. Wang, Learning From Pseudo-Randomness With an Artificial Neural Network–Does God Play Pseudo-Dice?, IEEE Access 6 (2018) 22987–22992.

[10] M. François, T. Grosges, D. Barchiesi, R. Erra, Pseudo-random number generator based on mixing of three chaotic maps, Communications in Nonlinear Science and Numerical Simulation 19 (4) (2014) 887–895.

[11] R. E. Ganz, The decimal expansion of π is not statistically random, Experimental Mathematics 23 (2) (2014) 99–104.

[12] P. Garg, Cryptanalysis of Simplified Data Encryption Standard Using Genetic Algorithm, American Journal of Networks and Communications 4 (3) (2015) 32.

[13] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy, in: International Conference on Machine Learning, 201–210, 2016.

[14] S. W. Golomb, et al., Shift register sequences, Aegean Park Press, 1967.

[15] C. González, H. Larrondo, O. Rosso, Statistical complexity measure of pseudorandom bit generators, Physica A: Statistical Mechanics and its Applications 354 (2005) 281–300.

[16] M. Han, J. Xi, S. Xu, F.-L. Yin, Prediction of chaotic time series based on the recurrent predictor neural network, IEEE transactions on signal processing 52 (12) (2004) 3409–3416.

[17] M. C. Hao, U. Dayal, D. A. Keim, T. Schreck, Importance-Driven Visualization Layouts for Large Time Series Data, in: J. T. Stasko (Ed.), IEEE Symposium on Information Visualization (InfoVis 2005), Minneapolis, MN, USA, October 23-25, 2005, ISBN 0-7803-9464-X, 203–210, 2005.

[18] M. Harrower, C. A. Brewer, ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps, The Cartographic Journal 40 (1) (2003) 27–37.

[19] H. C. Heien, M. Rahman, Revisiting the Digits of [pi] and Their Randomness, Ph.D. thesis, Minnesota State University, Mankato, 2005.

[20] D. D. Hosfelt, Automated detection and classification of cryptographic algorithms in binary programs through machine learning, CoRR abs/1503.01186, URL http://arxiv.org/abs/1503.01186.

[21] H. Hu, L. Liu, N. Ding, Pseudorandom sequence generator based on the Chen chaotic system, Computer Physics Communications 184 (3) (2013) 765–768.

[22] A. Huang, Hacking the Xbox: An introduction to reverse engineering, No Starch Press, San Francisco, 2003.

[23] A. Jäschke, F. Armknecht, Unsupervised Machine Learning on Encrypted Data, in: C. Cid, M. J. Jacobson Jr. (Eds.), Selected Areas in Cryptography – SAC 2018, 453–478, 2019.
[24] D. Knuth, The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms, Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA, ISBN 0-201-89684-2, 1997.
[25] P. Lamberti, M. Martin, A. Plastino, O. Rosso, Intensive entropic non-triviality measure, Physica A: Statistical Mechanics and its Applications 334 (1) (2004) 119–131.
[26] P. L'Ecuyer, R. Simard, TestU01: A C Library for Empirical Testing of Random Number Generators, ACM Trans. Math. Softw. 33 (4) (2007) 22:1–22:40.
[27] H. Leung, T. Lo, S. Wang, Prediction of noisy chaotic time series using an optimal radial basis function neural network, IEEE Transactions on Neural Networks 12 (5) (2001) 1163–1172.
[28] R. López-Ruiz, H. Mancini, X. Calbet, A statistical measure of complexity, Physics Letters A 209 (5) (1995) 321–326.
[29] M. Luby, M. Luby, Pseudorandomness and cryptographic applications, Princeton computer science notes, Princeton University Press, Princeton, 1996.
[30] J. Machicao, O. Bruno, Improving the pseudo-randomness properties of chaotic maps using deep-zoom, Chaos: An Interdisciplinary Journal of Nonlinear Science 27 (2017) 53116.
[31] J. Machicao, O. M. Bruno, M. S. Baptista, Zooming into chaos for a fast, light and reliable cryptosystem, 2020.
[32] J. Machicao, A. Marco, O. M. Bruno, Chaotic encryption method based on life-like cellular automata, Expert Systems with Applications 39 (2012) 12626–12635.
[33] G. Markowsky, The Sad History of Random Bits, Journal of Cyber Security and Mobility 3 (2014) 1–24.
[34] G. Marsaglia, The Marsaglia random number CDROM, with the DIEHARD battery of tests of randomness, URL http://www.stat.fsu.edu/pub/diehard, 1998.
[35] M. Matsumoto, T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and. Computer Simulation 8 (1998) 3–30.
[36] N. Metropolis, S. Ulam, The Monte Carlo method, Journal of the American Statistical Association 44 (247) (1949) 335–341.
[37] L. D. Micco, H. A. Larrondo, A. Plastino, O. A. Rosso, Quantifiers for randomness of chaotic pseudo-random number generators, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 367 (1901) (2009) 3281–3296.
[38] MIT Student Information Processing Board (SIPB), One billion digits of π, URL https://stuff.mit.edu/afs/sipb/contrib/pi/, 2019.
[39] M. E. O'Neill, PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation, Tech. Rep. HMC-CS-2014-0905, Harvey Mudd College, Claremont, CA, 2014.
[40] İ. Öztürk, R. Kılıç, A novel method for producing pseudo random numbers from differential equation-based chaotic systems, Nonlinear Dynamics 80 (3) (2015) 1147–1157.
[41] A. Peinado, A. Fúster-Sabater, Generation of pseudorandom binary sequences by means of linear feedback shift registers (LFSRs) with dynamic feedback., Mathematical and Computer Modelling 57 (11-12) (2013) 2596–2604.
[42] A. G. Radwan, S. H. AbdElHaleem, S. K. Abd-El-Hafiz, Symmetric encryption algorithms using chaotic and non-chaotic generators: A review, Journal of Advanced Research 7 (2) (2016) 193–208.
[43] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, et al., NIST Special Publication 800-22: A statistical test suite for random number generator for criptographic applications, Tech. Rep., National Institute of Standards and Technology, Gaithersburg, MD, USA, 2001.
[44] L. Skanderova, A. Řehoř, Comparison of Pseudorandom Numbers Generators and Chaotic Numbers Generators used in Differential Evolution, in: I. Zelinka, P. N. Suganthan, G. Chen, V. Snasel, A. Abraham, O. Rössler (Eds.), Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems, Springer International Publishing, Cham, 111–121, 2014.
[45] R. Spillman, M. Janssen, B. Nelson, M. Kepner, Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers, Cryptologia 17 (1) (1993) 31–44.
[46] G. L. Steele, D. Lea, C. H. Flood, Fast splittable pseudorandom number generators, in: Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications - OOPSLA' 14, ACM Press, 453–472, 2014.
[47] I. Vattulainen, K. Kankaala, J. Saarinen, T. Ala-Nissila, A comparative study of some pseudorandom number generators, Computer Physics Communications 86 (3) (1995) 209–226, ISSN 0010-4655.
[48] J. Walker, A pseudorandom number sequence test program, URL http://www.fourmilab.ch/random/, 1998.