Um Estudo Comparativo entre Três Geradores de Números Aleatórios

Carlos Eduardo Costa Vieira
email:cadu@inf.puc-rio.br

Reinaldo de Castro e Souza
Departamento de Engenharia Elétrica - PUC-Rio
email:reinaldo@ele.puc-rio.br

Celso da Cruz Carneiro Ribeiro
email:celso@inf.puc-rio.br

PUC-RioInf.MCC16/04 Junho, 2004

Abstract

The objective of this work is to compare three random number generators used by the research group on Heuristic and Parallelism for Optimization/Bioinformatics Problems of the Informatics' Department of PUC-Rio. The random number generators are used in probabilistic algorithms, such that Simulated Annealing, Genetics Algorithms and GRASP. Therefore, the use of a consistent random number generator method is highded recommended. Firstly, the three random number generators are presented. Secondly, theoretical and statistical tests used to analyze generators are discussed. Practical questions, such as algorithm efficiency, portability, facility of implementation are also used to analyze generators. Based on the tests results and many criteria used in the analysis, the best generator for the research group is pointed out.

Keywords: Random numbers generations, theoretical tests, statistical tests.

Resumo

O objetivo deste trabalho é comparar três geradores de números aleatórios utilizados no grupo de pesquisa Heurística e Paralelismo para Problemas de Otimização e de Bioinformática do Departamento de Informática da PUC-Rio. Os geradores de números aleatórios são utilizados em diversos algoritmos probabilísticos como Simulated Annealing, Algoritmos Genéticos e GRASP, tornando-se, assim, de suma importância a utilização de um bom método de geração de números aleatórios nessas técnicas. Primeiramente serão mostrados os geradores a serem testados. Em seguida, serão apresentados testes teóricos e empíricos utilizados para avaliar a qualidade dos geradores. Questões práticas como velocidade do algoritmo, portabilidade, facilidade de implementação também serão utilizadas na avaliação dos geradores. Baseando-se nos resultados dos testes e diversos critérios utilizados como análise, recomendações finais sobre os geradores serão dadas com o objetivo de apontar o mais adequado para as aplicações do grupo de pesquisa acima mencionado.

Palavras chaves: Geradores de números aleatórios, testes teóricos, testes estatísticos.

Sumário

1 Introdução

O objetivo dos geradores de números aleatórios é imitar ou simular o conceito matemático abstrato de variáveis aleatórias independentes uniformemente distribuídas sobre o intervalo $[0, 1]$ (i.u.d. $U[0, 1]$). Variáveis aleatórias de outras distribuições são simuladas empregando-se transformações apropriadas aos números aleatórios uniformes [1].

Os geradores de números aleatórios são algoritmos matemáticos específicos, sequenciais e determinísticos (se inicializados com o mesmo estado inicial ou semente produzem a mesma sequência de números). Os números gerados são, portanto, apenas pseudoaleatórios, e com um leve abuso de linguagem são chamados de aleatórios.

Dentre diversas outras aplicações, os geradores de números aleatórios são utilizados na área de otimização em algoritmos probabílisticos tais como as metaheurísticas Algoritmos Genéticos [2], *Simulated Annealing* [3], Pesquisa em Vizinhança Variável [4], [5] e *GRASP - Greedy Randomized Adaptive Search Procedures* [6]. Desempenham funções variadas como a fuga de ótimos locais, a construção de operadores genéticos, diversificam a busca no espaço de soluções, etc. Portanto, os geradores de números aleatórios influenciam fortemente a qualidade das soluções encontradas e o tempo total de processamento das técnicas acima descritas quando estas são utilizadas como métodos de resolução de diversos problemas.

Os geradores de números aleatórios deveriam ser construídos baseados em uma sólida análise matemática de suas propriedades estruturais. Uma vez construídos e implementados, devem ser submetidos a testes estatísticos que tentam detectar deficiências, procurando por evidências empíricas contra a hipótese nula H_0 : os números aleatórios gerados são *i.u.d.* no intervalo $U[0, 1]$.

O objetivo deste trabalho é avaliar a qualidade de três geradores de números aleatórios utilizados no grupo de pesquisa *Heurística e Paralelismo para Problemas de Otimização e de Bioinformática* do *Departamento de Informática* da *Pontifícia Universidade Católica do Rio de Janeiro* [7]. São eles: o gerador de números aleatórios utilizado na linguagem *ANSI-C* [8], o gerador de números aleatórios do *Linus Schrage* [9] e o gerador de números aleatórios de *Matsumoto* e *Nishimura* [10]. Para a avaliação dos geradores, serão utilizados diversos testes teóricos e estatísticos disponíveis na literatura. Aspectos práticos como a velocidade do algoritmo de geração dos números aleatórios, a facilidade de implementação, a possibilidade de utilização de técnicas paralelas e a disponibilidade de implementações portáveis serão também analisados.

Este trabalho está organizado como se segue: a seção 2 apresenta uma breve descrição dos geradores testados; a seção 3 descreve os testes utilizados para avaliar a qualidade dos geradores estudados; a seção 4 apresenta os resultados dos testes e uma análise dos mesmos e, por último, a seção 5 apresenta as considerações finais com as recomendações sobre os geradores testados.

2 Geradores de Números Aleatórios Testados

2.1 Gerador de Números Aleatórios *ANSI-C*

O gerador de números aleatórios utilizado na linguagem de programação *ANSI-C* é um Gerador Linear Congruente Misto ($x_{n+1} = (ax_n + c)\ mod\ m$) com a recorrência mostrada pela fórmula 1 [8].

$$x_{n+1} = (1.103.515.245\ x_n + 12.345)\ mod\ (2^{31}) \tag{1}$$

onde: o multiplicador $a = 1.103.515.245$, $c = 12.345$ e o valor do módulo $m = 2^{31} = 2.147.483.648$. Esta recursão produz uma sequência de números inteiros no conjunto $\{0, 1, 2, 3 \ldots, 2^{31} - 1 = 2.147.483.647\}$.

2.2 Gerador de Números Aleatórios de *Linus Schrage*

O gerador de números aleatórios do *Linus Schrage* é baseado em um Gerador Linear Congruente Multiplicativo ($x_{n+1} = ax_n\ mod\ m$) muito popular cuja recorrência é mostrada pela fórmula 2 [9].

$$x_{n+1} = 7^5 x_n\ mod\ (2^{31} - 1) \tag{2}$$

onde: o multiplicador $a = 7^5 = 16.807$ e o valor do módulo $m = 2^{31} - 1 = 2.147.483.647$ é chamado de número primo de *Mersenne*. Assim, todos os números inteiros produzidos pelo gerador estão na faixa entre 0 e $2^{31} - 1$. O gerador é considerado de ciclo completo desde que o multiplicador $a = 7^5$ é uma raiz primitiva do módulo $m = 2^{31} - 1$. Assim, cada inteiro de 1 até $2^{31} - 2 = 2.147.483.646$ é gerado exatamente uma única vez no ciclo. A semente inicial pode assumir valores inteiros entre 0 e $2^{31} - 1$. Uma característica interessante deste gerador é a portabilidade, isto é, pode gerar a mesma sequência de números aleatórios em diferentes máquinas [9].

2.3 Gerador de Números Aleatórios de *Matsumoto* e *Nishimura*

O gerador de números aleatórios proposto por *Makoto Matsumoto* e *Takuji Nishimura* é denominado *Mersenne Twister (MT)* e possui o período astronômico de $\rho = 2^{19.937} - 1$. Este gerador é uma nova variante do gerador proposto por *Makoto Matsumoto* e *Yoshiharu Kurita* [21], [22], que possui um período de $\rho = 2^{800} - 1$ e foi modificado para admitir um período de um número primo de *Mersenne*.

O *MT* é uma variante dos Geradores de Registradores de Deslocamento e cuja recorrência linear é mostrada na fórmula 3.

$$x_{k+n} = x_{k+m}\ \oplus\ (x_k^u \mid x_{k+1}^l)A,\ (k = 0, 1, \ldots) \tag{3}$$

onde: o vetor $x = (x_{w-1}, x_{w-2}, \ldots, x_0)$ é uma palavra de w bits, $\oplus$ é um operador ou exclusivo bit a bit e $\mid$ é a operação de concatenação. Existem diversas constantes: o inteiro n (grau da recorrência), o inteiro $0 \leq r \leq w - 1$ (escondido na definição de x_k^u), o inteiro $1 \leq m \leq n$ e uma matriz $A_{w \times w}$. Dado as sementes

iniciais $x_0, x_1, \ldots, x_{n-1}$, a recorrência acima gera x_n com $k = 0$. Os outros valores $x_{n+1}, x_{n+2}, \ldots$ são determinados fazendo-se $k = 1, 2, \ldots$ Do lado direito da recorrência, x_k^u significa os $w - r$ bits superiores de x_k e x_{k+1}^l significa os r bits inferiores de x_{k+1}. Desta maneira, se $x = (x_{w-1}, x_{w-2}, \ldots, x_0)$, x^u é o vetor de $w - r$ bits $(x_{w-1}, x_{w-2}, \ldots, x_r)$ e x^l é o vetor de r bits $(x_{r-1}, x_{r-2}, \ldots, x_0)$. Assim, $(x_k^u \mid x_{k+1}^l)$ é o vetor obtido pela concatenação dos $w - r$ bits superiores de x_k com os r bits inferiores de x_{k+1}, nessa ordem. Então, multiplica-se a matriz A pelo vetor concatenado e, logo em seguida, executa-se a operação $\oplus$ (adição bit a bit módulo 2) com x_{k+m}, gerando-se o próximo vetor x_{k+n}. A matriz A é escolhida de tal forma que a multiplicação de um vetor por ela seja feita rapidamente. Possui a seguinte forma:

$$A = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ & \vdots & \vdots & & & \\ 0 & 0 & 0 & \ldots & 0 & 1 \\ a_{w-1} & a_{w-2} & a_{w-3} & \ldots & a_1 & a_0 \end{pmatrix}$$

Então, o cálculo de xA pode ser realizado utilizando-se somente operações bit a bit:

$$xA = \begin{cases} deslocamento\_direita(x) & \text{se } x_0 = 0, \\ deslocamento\_direita(x) \ \oplus \ a & \text{se } x_0 = 1. \end{cases}$$

onde $a = (a_{w-1}, a_{w-2}, \ldots, a_0)$. Os valores x_k^u e x_{k+1}^l podem ser calculados através de operações *e* bit a bit. Assim, a fórmula 3 pode ser calculada através de operações de deslocamento, *ou exclusivo*, *ou* e *e*.

Os motivos da escolha de uma recorrência tão complicada (fórmula 3) e de um número primo de *Mersenne* são: evitar a utilização da fatoração, utilizando-se de uma operação mais fácil que é a checagem da primalidade de um número inteiro, facilitar a checagem da primitividade do polinômio característico e alcançar períodos extremamente longos. Assim, a forma da recorrência foi cuidadosamente selecionada para que a geração dos números aleatórios e a pesquisa dos parâmetros do gerador sejam realizados de forma eficiente [10]. Finalizando, o gerador MT é parametrizado pelos seguintes valores: $n = 624$, $m = 397$, $r = 31$, $w = 32$ e $a = 9908B0DF$ em hexadecimal [10]. Com esses parâmetros, obtém-se o período astronômico de $\rho = 2^{19.937} - 1$.

3 Testando-se os Geradores de Números Aleatórios

Duas classes de testes são comumente empregadas ao analisar os geradores de números aleatórios [23]: os testes teóricos e os testes estatísticos.

Os primeiros procuram analisar a estrutura do gerador com a finalidade de derivar propriedades comportamentais da sequência de pontos, usualmente sobre o período completo. Os testes teóricos são específicos para cada classe de geradores e incluem o cálculo do tamanho do período, a análise da estrutura

lattice via, por exemplo, Teste Espectral, cálculo de *bounds* para a discrepância, propriedade de equidistribuição, entre outros. Geradores de números aleatórios que não possuem um suporte teórico forte e convincente deveriam ser evitados [11]. O que se tem observado é que, geradores com boas propriedades teóricas produzem uma boa performance nos testes estatísticos, entretanto, não existe nenhuma comprovação matemática desse fato [19], [24].

Na prática, existem certas limitações na análise teórica. Ela pode ser realizada até uma certa dimensão t, na qual, muitas vezes, é relativamente pequena comparada com os tamanhos de amostras que são comumente utilizadas em simulações maiores. Mesmo uma boa distribuição até a dimensão t sobre o período completo do gerador não garante um bom comportamento empírico em amostras menores [24]. Assim, utilizam-se como complemento aos testes teóricos, os testes estatísticos com o objetivo de avaliar a qualidade de uma grande sequência produzida pelo gerador, de vários pontos sucessivos, de bits específicos dos números gerados, de certas transformações particulares, entre outros [24].

A metodologia utilizada aqui é fazer uma análise teórica dos geradores testados, e, em seguida, empregar testes estatísticos apropriados aos mesmos. Diversos aspectos práticos serão também analisados: velocidade do algoritmo, facilidade de implementação, possibilidade de utilização de técnicas paralelas, disponibilidade de implementações portáveis, entre outros.

3.1 Testes Teóricos

3.1.1 Tamanho do período

Claramente, o período de um gerador não pode exceder a cardinalidade de seu espaço de estados. Assim, para uma utilização ótima de memória, o período deveria ser próximo da sua cardinalidade. Muitos geradores satisfazem o requisito acima se os parâmetros são escolhidos apropriadamente [16], [18], nos quais incluem-se o gerador do *C*, do *Linus* e de *Matsumoto/Nishimura*.

O tamanho do período de um gerador impõe um limite no tamanho de amostra utilizada. A geração de números aleatórios é equivalente a uma amostragem sem reposição. Daí, o tamanho da amostra deveria ser muito menor do que o tamanho do período do gerador. Um limite superior para o tamanho de amostra utilizável é a raiz quadrada do tamanho do período ($\sqrt{\rho}$) [19]. Esta recomendação é baseada em evidências empíricas, não existe uma análise teórica disponível [19]. Outros critérios que limitam o número de chamadas utilizadas em um gerador de números aleatórios são: $\rho/1.000$ [16] e $\rho \gg 200n^2$, onde n representa o número de chamadas ao gerador [8]. Utilizando-se os critérios mencionados acima para os geradores congruentes (*C* e *Linus*), aproximadamente 50.000 chamadas poderiam ser utilizadas segundo o primeiro critério, aproximadamente 2.000.000 chamadas usando-se o segundo critério e aproximadamente 1.000 chamadas utilizando-se o terceiro critério. Dado o tamanho do período do gerador de *Matsumoto* e *Nishimura* $\rho = 2^{19.937} - 1$, o período ρ é da ordem de:

$$\rho = log(2^{19.937}) = 19.937\ log(2) \approx 6001$$

Assim, de acordo com o primeiro, segundo e terceiros critérios, poderiam ser feitas, respectivamente: $\approx 10^{3.000}$, $\approx 10^{5.998}$ e $\approx 10^{2.000}$ chamadas para o

gerador de *Matsumoto* e *Nishimura*.

Atualmente, modernas simulações no computador requerem uma quantidade cada vez maior de números aleatórios. *Pierre L' Ecuyer* enfatiza em diversos artigos [1], [23], [25] que os geradores com períodos próximos de 2^{31}, ainda recomendados em diversos livros de simulação, deveriam ser descartados devido ao seu pequeno período (pode-se exaurir o período desses geradores em poucos minutos com um bom computador) [23]. Ele enfatiza também que períodos menores do que 2^{50} são muito baixos e que, devido aos últimos avanços nos algoritmos de geração de números aleatórios, deve-se utilizar períodos superiores a 2^{200} [23]. *B. D. Ripley* aconselha um período de 2^{31} para problemas com até 1.000 pontos e um período de pelo menos 2^{50} para problemas que utilizam pelo menos 1.000.000 de pontos [8].

3.1.2 Teste Espectral

O Teste Espectral, proposto por *Conveyou* e *McPherson* [26] , é um teste teórico que caracteriza o gerador sobre o seu período completo. Foi desenvolvido para os Geradores Congruentes Multiplicativos e não se aplica a todos os geradores de números aleatórios. Possui este nome pois foi primeiramente aplicado sobre o ponto de vista da análise espectral.

Seja o Gerador Linear Congruente Multiplicativo cuja fórmula de recorrência geral é dada por $x_{n+1} = ax_n \ mod \ m$. Para cada inteiro $t \geq 1$, define-se [23]:

$$T_t = \{u_n = (u_n, \ldots, u_{n+t-1}) | n \geq 0, x_0 \in \{0, \ldots, m-1\}\} \tag{4}$$

o conjunto de todas as t tuplas sobrepostas de valores sucessivos de u_n, de todas as sementes iniciais possíveis. O conjunto T_t é igual a interseção da estrutura *lattice* L_t com o cubo unitário t dimensional $(0,1]^t$ [16]. Do ponto de vista matemático, a estrutura *lattice* de dimensão t no espaço real R^t é um conjunto da forma [23]:

$$L_t = \sum_{j=1}^{t} z_j V_j \mid cada \ z_j \in Z \tag{5}$$

onde Z é o conjunto de todos os inteiros e $\{V_1, \ldots, V_t\}$ é o conjunto de vetores linearmente independentes chamado base *lattice* de R^t. Assim, a estrutura *lattice* L_t é o conjunto de todas as combinações lineares inteiras dos vetores $\{V_1, \ldots, V_t\}$.

O fato de que os pontos em T_t pertencem a estrutura *lattice* significa que os pontos em T_t estão em um conjunto de hyperplanos paralelos equidistantes. Entre todas as famílias de hyperplanos que cobrem todos os pontos, escolha aquela para o qual a distância entre os hyperplanos sucessivos é a maior possível e seja d_t essa distância. Quanto menor o valor de d_t, mais uniformemente distribuído estará T_t sobre o cubo unitário. Na literatura, examinar os valores de d_t associados com um dado gerador é denominado Teste Espectral. Em resumo, esse teste mede o quão densamente o conjunto T_t preenche o hyperespaço t-dimensional. Na prática, obter uma uniformidade $t-$dimensional para cada inteiro positivo t é impossível, mas uma boa uniformidade para pequenos valores de t é essencial.

Existe um limite no quão pequeno a distância máxima d_t pode estar.

Esse limite inferior geral para o valor d_t é dado por [16]:

$$d_t \geq d_t^* = \frac{1}{\gamma_t m^{1/t}} = \begin{cases} (4/3)^{-1/4} m^{-1/2}, \ se\ t = 2 \\ 2^{-1/6} m^{-1/3}, \ se\ t = 3 \\ 2^{-1/4} m^{-1/4}, \ se\ t = 4 \\ 2^{-3/10} m^{-1/5}, \ se\ t = 5 \\ (64/3)^{-1/12} m^{-1/6}, \ se\ t = 6 \\ 2^{-3/7} m^{-1/7}, \ se\ t = 7 \\ 2^{-1/2} m^{-1/8}, \ se\ t = 8 \end{cases}$$

A constante γ_t, denominada constante de *Hermite*, depende somente de t e seu valor exato é conhecido para $t \leq 8$ [16]. Para $t \geq 8$, limites inferiores e superiores para γ_t são disponíveis [27]. Normalizando-se d_t obtém-se a figura de mérito $S_t = d_t^*/d_t$ com o objetivo de avaliar os geradores. Assim, os valores de S_t estão entre 0 e 1 e valores mais próximos da unidade são preferidos. Um algoritmo para o cálculo de d_t foi proposto por *Conveyou* e *McPherson* [26] e posteriormente melhorado por *Knuth* [16] e *Dieter* [28].

Existem outros critérios de qualidade para avaliar os geradores sobre o ponto de vista da estrutura *lattice* além da máxima distância entre hyperplanos paralelos adjacentes (Teste Espectral) [29], [30]: mínimo número de hyperplanos paralelos, mínima distância entre os pontos, taxa do tamanho entre os maiores e os menores vetores em uma base *lattice* denonimado quociente de *Beyer*, discrepância, medida do empacotamento da estrutura *lattice* com esferas, etc. Porém, tipicamente essas figuras de mérito são mais complicadas de calcular do que o Teste Espectral e não acrescentam nenhuma informação importante.

Para geradores com pequenos períodos, a distância d_t entre os hyperplanos pode ser determinada por uma enumeração completa. Seja o exemplo abaixo para a ilustração do Teste Espectral [13].

Exemplo

Sejam os geradores $x_{n+1} = 3x_n \ mod\ 31$ e $x_{n+1} = 13x_n \ mod\ 31$. Utilizando-se a semente $x_0 = 15$ e plotando-se os pares sobrepostos $\{(u_n, u_{n+1}); 0 \leq n < 31\}$ para os dois geradores, tem-se os gráficos mostrados nas Figuras 1 e 2.

No gráfico da Figura 1, observa-se que os pontos estão em 3 linhas com inclinação positiva ou 10 linhas com inclinação negativa. Desde que a distância entre as linhas com inclinação positiva é maior do que a distância entre as linhas com inclinação negativa, as linhas com inclinação positiva serão consideradas, cujas equações são: $x_{n+1} = 3x_n$, $x_{n+1} = 3x_n - 31$ e $x_{n+1} = 3x_n - 62$. A distância entre 2 linhas paralelas $y = ax + c_1$ e $y = ax + c_2$ é $|c_2 - c_1|/\sqrt{1 + a^2}$. Assim, $d_2 = 31/\sqrt{10} = 9.80$ para o primeiro gerador.

Já no gráfico da Figura 2, observa-se que os pontos estão em 6 linhas com inclinação positiva ou 7 linhas com inclinação negativa, sendo a distância da inclinação negativa maior. As equações das linhas são dadas por: $x_{n+1} = -(5/2)x_n + (31/2)k$, *com* $k = 0, 1, \ldots, 5$. Assim, $d_2 = \frac{31}{2}/\sqrt{1 + (5/2)^2} = 5.76$ para o segundo gerador. Então, a menor distância máxima é dada pelo segundo gerador, possuindo assim uma distribuição melhor no plano do que o primeiro gerador.

A Tabela 1 mostra os valores do Teste Espectral nas dimensões $2 \leq$

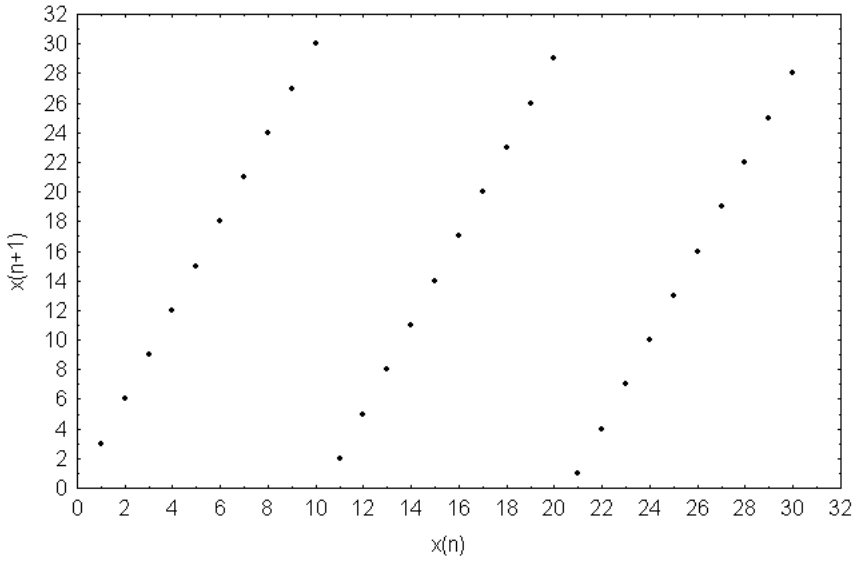

Figura 1: Pares $\{(u_n, u_{n+1}); 0 \leq n < 31\}$ para $x_{n+1} = 3x_n \ mod\ 31$

$t \leq 8$ para o gerador de números aleatórios do *Linus Schrage* que possui o multiplicador $a = 16.807$ e $m = 2^{31} - 1$ [15], [31].

t	d_t^*	d_t	S_t
2	0.0000201	0.0000595	0.3375
3	0.000690	0.001565	0.4412
4	0.00391	0.006791	0.5752
5	0.01105	0.0150	0.7361
6	0.02157	0.0334	0.6454
7	0.03450	0.0604	0.5711
8	0.04819	0.0791	0.6096

Tabela 1: Valores do Teste Espectral para o gerador do *Linus*

A Tabela 1 mostra uma forte presença da estrutura *lattice* no espaço bi-dimensional e tri-dimensional do gerador. Valores para $4 \leq t \leq 8$ são considerados melhores.

Fishman e *Moore* fizeram uma análise exaustiva dos multiplicadores a para os Geradores Congruentes Lineares Multiplicativos com módulo $m = 2^{31} - 1$. O critério utilizado foi $S_t \geq 0.8$ para $2 \leq t \leq 6$, garantindo assim que, para cada t, a distância entre os hyperplanos paralelos adjacentes não exceda o limite inferior por mais do que 25% [29]. Foram examinados 534.6 milhões de multiplicadores candidatos (devido a simetria esse número reduz-se pela metade, 267.3 milhões). Desse total, os autores encontraram 414 multiplicadores candidatos (207 no conjunto reduzido) que satisfaziam o critério adotado. Isto é um multiplicador para cada 1.3 milhões. Assim, segundo esse critério, o multiplicador $a = 16.807$ do gerador do *Linus* não poderia ser utilizado. Já *Ripley* considera os critérios adotados por *Fishman* e *Moore* rigorosos. Assim, segundo

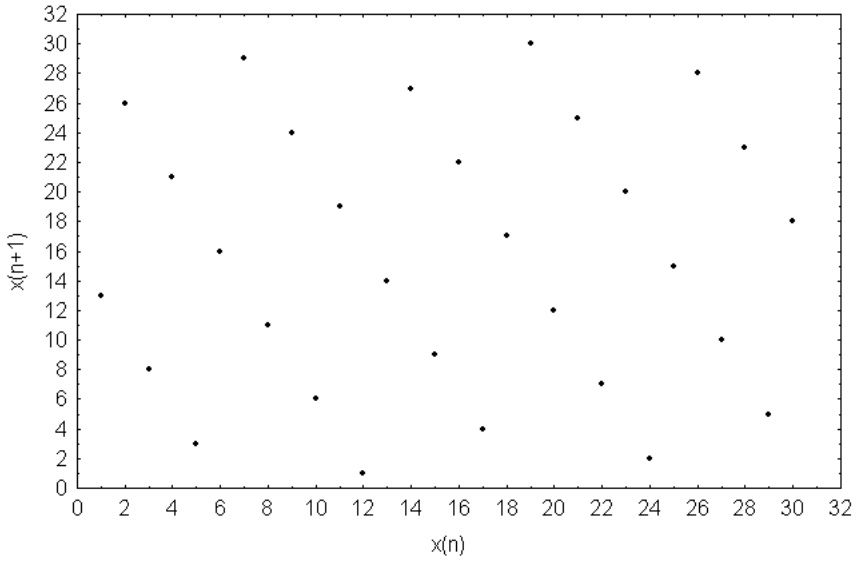

Figura 2: Pares $\{(u_n, u_{n+1}); 0 \leq n < 31\}$ para $x_{n+1} = 13x_n \ mod\ 31$

ele, tanto a estrutura *lattice* do gerador do *Linus* quanto a do gerador do *C* são adequadas em até 8 dimensões [8].

3.1.3 Teste da $K-$Distribuição

Para os Geradores de Registradores de Deslocamento e suas variantes, o conjunto de todas as tuplas sobrepostas não está em uma estrutura *lattice* no espaço real como acontece nos Geradores Congruentes Lineares e, sim em um espaço mais abstrato, denominado espaço de séries formais [24]. Essa estrutura pode ser utilizada para analisar a uniformidade de todas as tuplas sobrepostas no hypercubo unitário via a noção de $k-$distribuição que é dada a seguir.

Uma sequência pseudo-aleatória x_i de inteiros de w bits com período ρ é dita ser $k-$distribuída com v bits de precisão se satisfaz a seguinte condição [10], [22]:

Seja $trunc_v(x)$ o número formado pelos primeiros v bits de x e considere vetores $kv-$bits do tipo:

$$(trunc_v(x_i), trunc_v(x_{i+1}), \ldots, trunc_v(x_{i+k-1})),\ com\ 0 \leq i < \rho$$

Então, cada uma das 2^{kv} combinações possíveis de bits ocorrem o mesmo número de vezes em um período, com exceção da combinação com todos os bits zero que ocorre uma vez menos.

Seja agora $x_0, x_1, \ldots$ uma sequência de inteiros de w bits e ρ o seu período. Para cada $v = 1, 2, \ldots, w$, seja $k(v)$ o número máximo tal que a sequência é $k(v)-$distribuída com $v-$bits de precisão. Pode-se assumir a desigualdade $2^{k(v)v} - 1 \leq \rho$, desde que, no máximo ρ padrões podem ocorrer em um período e o número de possíveis padrões de bits nos v bits mais significativos de $k(v)$ palavras consecutivas é $2^{k(v)v}$. No caso do gerador de *Matsumoto* e *Nishimura*, $\rho = 2^{nw-r} - 1$, então tem-se o seguinte limite superior no valor de

$k(v)$ para o gerador:

$$2^{k(v)v} - 1 \leq 2^{nw-r} - 1 \rightarrow k(v)v \leq nw - r \rightarrow k(v) \leq k(v)^* = \left\lfloor \frac{nw-r}{v} \right\rfloor$$

O valor $k(v)$ é denominado Teste da $K-$Distribuição e, consequentemente, quanto maior $k(v)$ para cada v assegura uma maior equidistribuição com $v-$bits de precisão. Se o gerador consegue atingir o limite superior para cada v, com $1 \leq v \leq w$, ele pode ser denominado de Maximamente Equidistribuído ou Assintoticamente Aleatório. Esta é a melhor distribuição que se pode esperar para esses tipos de geradores. Segundo *Pierre L'Ecuyer*, esses geradores são relativamente fáceis de encontrar e apresenta em [32] vários exemplos e implementações concretas de Geradores de Registradores de Deslocamento que possuem essa propriedade.

A propriedade de Maximamente Equidistribuído não pode ser alcançada pelo gerador de *Matsumoto* e *Nishimura*. A Tabela 2 mostra os resultados do Teste de $K-$Distribuição para esse gerador com até 32 bits de precisão, isto é, $1 \leq v \leq 32$ [10]. A coluna $k(v)^*$ indica o limite superior para cada bit de precisão e a coluna $k(v)$ indica o resultado do teste. Os parâmetros desse gerador são: $(w, n, r) = (32, 624, 31)$.

v	$k(v)^*$	$k(v)$	v	$k(v)^*$	$k(v)$
1	19.937	<u>19.937</u>	17	1.172	623
2	9.968	<u>9.968</u>	18	1.107	623
3	6.645	6.240	19	1.049	623
4	4.984	<u>4.984</u>	20	996	623
5	3.987	3.738	21	949	623
6	3.322	3.115	22	906	623
7	2.848	2.493	23	866	623
8	2.492	<u>2.492</u>	24	830	623
9	2.215	1.869	25	797	623
10	1.993	1.869	26	766	623
11	1.812	1.248	27	738	623
12	1.661	1.246	28	712	623
13	1.533	1.246	29	687	623
14	1.424	1.246	30	664	623
15	1.329	1.246	31	643	623
16	1.246	<u>1.246</u>	32	623	<u>623</u>

Tabela 2: Teste de $K-$Distribuição para o gerador de *Matsumoto* e *Nishimura*

Pela Tabela 2 observa-se que, com 32 bits de precisão, as 623 tuplas de saída sobre o período completo do gerador são equidistribuídas no cubo unitário 623-dimensional. Com 16 bits de precisão, as 1246 tuplas são equidistribuídas no cubo unitário 1246-dimensional e, assim, sucessivamente, a partir dos números sublinhados na coluna $k(v)$.

3.2 Testes Estatísticos

Os testes estatísticos visualizam os geradores de números aleatórios como uma caixa preta, procurando detectar deficiências através de evidências empíricas contra a hipótese nula H_0 : as observações u_n são *i.u.d.* no intervalo $U[0, 1]$.

Qualquer função de um número finito de variáveis aleatórias $U[0,1]$ cuja distribuição sobre H_0 é conhecida (algumas vezes, uma distribuição aproximada também é válida) pode ser utilizada como uma estatística T para definir um teste estatístico para os geradores de números aleatórios. Assim, o número de testes que podem ser definidos é infinito e não existe um teste universal ou uma bateria de testes que possa garantir que um dado gerador é completamente confiável para todos os tipos de simulações. Ao passar em uma bateria de testes, aumenta-se a confiança no gerador, mas não se tem nenhuma garantia de que o gerador é a prova de falhas, isto é, de que ele irá passar em outro teste estatístico ou mesmo no mesmo teste estatístico em dimensões e/ou tamanhos de amostras diferentes. De fato, nenhum gerador pode passar em todos os testes estatísticos [34].

A principal idéia dos testes estatísticos é procurar situações onde o comportamento de alguma função da saída do gerador é significativamente diferente do comportamento normal ou esperado da mesma função aplicada em uma sequência de variáveis aleatórias uniformemente distribuídas no intervalo [0,1] [31]. Seja o exemplo abaixo para uma pequena ilustração [31].

Suponha n números aleatórios gerados cuja saída procura "imitar" variáveis aleatórias i.u.d. $U[0,1]$. Seja T o número de valores que estão abaixo de $1/2$, dentre os n valores. Para um grande valor de n, T deveria normalmente está próximo de $n/2$. De fato, espera-se que T comporte-se como uma variável aleatória binomial com parâmetros $(n, 1/2)$. Assim, ao repetir este experimento várias vezes, isto é, gerando-se N valores de T, a distribuição dos valores de T obtida deveria se assemelhar a distribuição binomial (e a distribuição normal com média $n/2$ e desvio padrão $\sqrt{n}/2$ para um grande valor de n). Se $N = 100$ e $n = 10.000$, a média e o desvio padrão são 5.000 e 50, respectivamente. Se, com esse parâmetros, observa-se, por exemplo, que 98 valores dos 100 valores de T são menores do que 5.000, pode-se concluir que algo está errado com o gerador de números aleatórios. Se, por outro lado, os valores de T comportam-se como o esperado, pode-se concluir que o gerador reproduz o comportamento correto para esta estatística T e para este tamanho de amostra.

A metodologia adotada aqui é executar um grande números de testes estatísticos aos geradores estudados. Foram utilizados diversos testes considerados clássicos na literatura extraídos de [13] e [16]. Contudo, esses testes tem-se mostrado fracos, no sentido de que diversos geradores suspeitos passam nos mesmos [15], [24]. Assim, foi utilizado também a bateria de testes estatísticos desenvolvida por *George Marsaglia*, denominada *Diehard*, cujos testes são considerados mais rigorosos que os testes clássicos [33].

Os testes foram executados em uma máquina *Pentium III 750 MHz* com *256 Mbytes* de memória *RAM - Random Access Memory* sobre o sistema operacional *Linux RedHat 7.0*. Para cada teste e para cada gerador testado, um total de 100 sementes foram selecionadas. Também houve uma variação no tamanho das amostras em cada teste, quando possível.

3.2.1 Testes Clássicos

A notação adotada aqui foi a mesma utilizada em [16]. Os testes podem ser aplicados em sequências $u_0, u_1, u_2, \ldots$ de números reais ou em sequências de números inteiros $y_0, y_1, y_2, \ldots$ com $y_n = \lfloor d \times u_n \rfloor$. No primeiro caso pressupõem-se que a sequência seja independente e uniformemente distribuída entre 0 e 1

e no último caso pressupõem-se que a sequência seja independente e uniformemente distribuída entre 0 e $d-1$. O número d é escolhido convenientemente; por exemplo, poderia ser $d = 64 = 2^6$, assim y_n representaria os 6 bits mais significativos de u_n [16]. O nível de significância adotado nos testes clássicos foi $\alpha = 0.05$.

Teste Chi-Quadrado no intervalo [0,1]

Para efetuar o Teste Chi-Quadrado, divide-se o intervalo em k subintervalos e geram-se n números aleatórios $u_1, u_2, \ldots, u_n$ entre 0 e 1. Calcula-se o número de amostras contidas em cada um dos intervalos e sejam esses números $N_1, N_2, \ldots, N_k$. Calcula-se agora a estatística $D = (o_i - e_i)^2/e_i$, onde o_i e e_i são as frequências observadas e esperadas da $i-$ésima célula. Assim, a estatística D é dada por:

$$D = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} = \frac{k}{n} \sum_{i=1}^{k} (N_j - n/k)^2$$

A estatística D segue uma distribuição Chi-Quadrado com $k-1$ graus de liberdade. A hipótese nula de que as observações vêm de uma distribuição uniforme não podem ser rejeitadas a um nível de significância α se o valor de D é menor do que o valor da Tabela Chi-Quadrado equivalente à $\chi^2_{[1-\alpha,k-1]}$. Assim, aceita-se a hipótese nula se $D \leq \chi^2_{[1-\alpha,k-1]}$ e rejeita-se a hipótese, caso contrário. Para que o teste fique mais robusto, é importante que cada subintervalo k contenha pelo menos 5 observações [13], [16].

Para o Teste Chi-Quadrado no intervalo [0,1], os seguintes parâmetros foram adotados: $k = 5, 10$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000,$ 1.000.000 *e* 5.000.000.

Teste da Frequência ou Teste Chi-Quadrado no intervalo [0,d]

Seja d um número inteiro; para cada inteiro r, $0 \leq r < d$, conta-se o número de vezes que $y_j = r$ para $0 \leq j < n$ e aplica-se o Teste Chi-Quadrado utilizando $k = d$ categorias com probabilidade $p_s = 1/d$ para cada categoria [16].

Para o Teste da Frequência, os seguintes parâmetros foram adotados: $d = 32, 64$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste Kolmogorov-Smirnov

Para efetuar o Teste Kolmogorov-Smirnov ou Teste KS geram-se n números aleatórios $u_1, u_2, \ldots, u_n$ entre 0 e 1 e, logo em seguida, coloque-os em ordem crescente. Sejam esses números $\{u_1, u_2, \ldots, u_n\}$ tal que $u_{n-1} \leq u_n$. Então, as estatísticas K^+ e K^- são calculados como se segue:

$$K^+ = \sqrt{n}\ max_j\ [j/n - x_j]$$

$$K^- = \sqrt{n}\ max_j\ [x_j - (j-1)/n]$$

onde K^+ mede o desvio máximo quando a Função Densidade de Probabilidade Observada está acima da Função Densidade de Probabilidade Esperada e K^- mede o desvio máximo quando a Função Densidade de Probabilidade Observada está abaixo da Função Densidade de Probabilidade Esperada. Se os valores de K^+ e K^- são menores do que o valor da Tabela $KS_{[1-\alpha;n]}$, as observações são uniformemente distribuídas no intervalo [0,1] ao nível α de significância [13], [16].

Para o Teste KS no intervalo [0,1], os seguintes parâmetros foram adotados: $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000\ e\ 5.000.000$.

Teste em Dois Níveis

O Teste em Dois Níveis consiste em utilizar o Teste KS em r amostras de tamanho N e, em seguida, utilizar novamente o Teste KS no conjunto das r estatísticas obtidas no passo anterior. Assim, o Teste KS de primeiro nível procura detectar um comportamento anormal local dos segmentos e o Teste KS de segundo nível procura detectar um comportamento anormal global combinando as r estatísticas do teste de primeiro nível [13], [16]. Denote as duas estatísticas de saída do Teste KS de primeiro nível por $(K_N^+)_\rho$ e $(K_N^-)_\rho$, com $\rho = 1, 2, \ldots, r$. Colocando-se as duas estatísticas acima em ordem crescente, pode-se medir a diferença entre a função de distribuição empírica e a função de distribuição esperada no Teste KS de segundo nível através das fórmulas abaixo:

$$K_r^{++} = \sqrt{r}\ max_\rho\ [\rho/n - G_0(x, r)]$$

$$K_r^{--} = \sqrt{r}\ max_\rho\ [G_0(x, r) - (\rho - 1)/n]$$

onde a distribuição KS para $N = 1.000$ no segundo nível é dada por [16]:

$$G_0(x, r) = prob(K_r^{++} \leq x) = 1 - e^{-2x^2}$$

Se os valores de K_r^{++} e K_r^{--} são menores do que o valor da Tabela $KS_{[1-\alpha;r]}$, as observações são uniformemente distribuídas no intervalo [0,1] ao nível α de significância.

Para o Teste em Dois Níveis, os seguintes parâmetros foram adotados: $N = 1.000$ e $r = 5, 10, 50, 100, 500, 1.000, 5.000$.

Teste Serial

Para cada par de inteiros (q, r) com $0 \leq q, r < d$, conta-se o número de vezes que o par $(y_{2j}, y_{2j+1}) = (q, r)$ ocorre, para $0 \leq j < n$ e emprega-se o Teste Chi-Quadrado com $k = d^2$ categorias e probabilidade $p = 1/d^2$ para cada categoria. Aceita-se a hipótese nula de que os pares não sobrepostos são uniformemente distribuídos entre 0 e $d - 1$ se a estatística $D \leq \chi^2_{[1-\alpha,d^2-1]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste Serial, os seguintes parâmetros foram adotados: $d = 2, 3, 4, 5$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000\ e$ $5.000.000$.

Teste da Permutação

Para efetuar o Teste da Permutação, divide-se a sequência de entrada em n grupos de t elementos cada, isto é, $(u_{jt}, u_{jt+1}, \ldots, u_{jt+t-1})$ para $0 \leq j < n$. Os elementos em cada grupo podem ter $t!$ ordenações relativas possíveis. O teste consiste em contar o número de vezes que cada ordenação aparece. Logo em seguida, um Teste Chi-Quadrado é aplicado com $k = t!$ categorias (ordenações) e com probabilidade $p = 1/t!$ para cada ordenação. Aceita-se a hipótese nula se a estatística $D \leq \chi^2_{[1-\alpha,(t!)^2-1]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste da Permutação, os seguintes parâmetros foram adotados: $t = 2, 3, 4, 5$ e $n = 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste da Lacuna

Seja α e β 2 números reais com $0 \leq \alpha < \beta \leq 1$. O Teste da Lacuna considera os tamanhos de subsequências consecutivas $u_j, u_{j+1}, \ldots, u_{j+r}$ na qual u_{j+r} está entre α e β. Essa subsequência com $r + 1$ números representa uma lacuna de tamanho r. Esse teste, aplicado a uma sequência de números reais u_n e para qualquer valor de α e β, conta o número de lacunas de tamanho $0, 1, \ldots, t - 1$ e o número de lacunas de tamanho $\geq t$ até que n delas tenham sido encontradas. Logo em seguida, aplica-se o Teste Chi-Quadrado aos $k = t+1$ valores, utilizando-se as probabilidades $p_r = p(1 - p)^r$, para $0 \leq r \leq t - 1$ e $p_t = (1 - p)^t$, onde $p = \beta - \alpha$ é a probabilidade de $\alpha \leq u_j < \beta$. Aceita-se a hipótese nula se a estatística $D \leq \chi^2_{[1-\alpha,t]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste da Lacuna, os seguintes parâmetros foram adotados: $\alpha = 0.25$, $\beta = 0.5$, $t = 6$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste do Coletor de Cupons

O Teste do Coletor de Cupons conta o número de coberturas de tamanhos específicos, procurando sequências de valores que incluem pelo menos uma ocorrência de cada resultado, não incluindo assim coberturas menores. O nome coletor de cupons deriva-se da analogia com uma pessoa que coleta cupons de diferentes tipos, pesquisando por um conjunto completo consistindo de cupons de cada tipo.

Para esse teste utiliza-se a sequência $y_0, y_1, y_2, \ldots$ com $0 \leq y_j < d$ e observa-se os tamanhos dos n segmentos $y_{j+1}, y_{j+2}, \ldots, y_{j+r}$ que são necessários para se obter um conjunto completo de inteiros de 0 até $d-1$. No final do teste tem-se $cont[r]$ como o número de segmentos com tamanho r, para $d \leq r < t$ e $cont[t]$ como o número de segmentos com tamanho $\geq t$. O Teste Chi-Quadrado é então aplicado aos $k = t - d + 1$ valores de $cont[d], cont[d + 1], \ldots, cont[t]$, depois de n segmentos terem sido encontrados e para o cálculo da frequência esperada utiliza-se as probabilidades $p_r = \frac{d!}{d^r} S_{r-1,d-1}$, para $d \leq r < t$ e $p_t = 1 - \frac{d!}{d^{t-1}} S_{t-1,d}$, onde $S_{n,m}$ representa o número de *Stirlings* do segundo tipo e cujo valor pode ser calculado pela fórmula: $S_{n,m} = mS_{n-1,m} + S_{n-1,m-1}$ com $S_{n,1} = S_{n,n} = 1$ para qualquer $n \geq 1$. Aceita-se a hipótese nula se a estatística

$D \leq \chi^2_{[1-\alpha,t-d]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste do Coletor de Cupons, os seguintes parâmetros foram adotados: $d = 3$, $t = 6$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste do Poker

O Teste do Poker considera n grupos de $k = 5$ inteiros sucessivos $\{y_{5j}, y_{5j+1}, \ldots, y_{5j+4}\}$, para $0 \leq j < n$, e conta-se o número de quíntuplas com $r = 5$ valores diferentes:

- 5 valores: todos os números diferentes. Ex: {a,b,c,d,e}
- 4 valores: um par de números iguais e o resto diferente. Ex: {a,a,b,c,d}
- 3 valores: dois pares de números iguais ou três de um mesmo tipo. Ex: {a,a,b,b,c} ou {a,a,a,b,c}
- 2 valores: Três números de um mesmo tipo e dois de outro ou quatro de um mesmo tipo. Ex: {a,a,a,b,b} ou {a,a,a,a,b}
- 1 valor: cinco números de um mesmo tipo. Ex: {a,a,a,a,a}

Um Teste Chi-Quadrado é então aplicado utilizando-se a probabilidade $p_r = \frac{d(d-1)\ldots(d-r+1)}{d^k} S_{k,r}$ de que existem r números diferentes, onde $S_{n,m}$ representa o número de *Stirlings* do segundo tipo. Aceita-se a hipótese nula se a estatística $D \leq \chi^2_{[1-\alpha,r-1]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste do Poker, os seguintes parâmetros foram adotados: $d = 5$, $k = 5$, $r = 5$ e $n = 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste Máximo-de-t

Seja $V_j = max(u_{tj}, u_{tj+1}, \ldots, u_{tj+t-1})$ para $0 \leq j < n$. O Teste Máximo-de-t consiste em aplicar o Teste KS a sequência $V_0, V_1, \ldots, V_{n-1}$ com a função de distribuição esperada igual a $F(x) = x^t$, para $0 \leq x \leq 1$. Alternativamente, pode-se aplicar o Teste Chi-Quadrado a sequência $V_0^t, V_1^t, \ldots, V_{n-1}^t$. O Teste escolhido foi o Chi-Quadrado, portanto, aceita-se a hipótese nula se a estatística $D \leq \chi^2_{[1-\alpha,k-1]}$ e rejeita-se a hipótese nula, caso contrário [16].

Para o Teste Máximo-de-t, os seguintes parâmetros foram adotados: $k = 10$, $t = 5, 6$ e $n = 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste de Correlação Serial

Dada uma sequência de números aleatórios, pode-se calcular a covariância entre números que estão k valores distantes, isto é, entre x_n e x_{n+k}. Este cálculo é denominado autocovariância a uma distância *(lag)* k. A expressão para calculá-la é:

$$R_k = \frac{1}{n-k} \sum_{i=1}^{n-k} (U_i - \frac{1}{2})(U_{i+k} - \frac{1}{2})$$

Para valores grandes de n, R_k é normalmente distribuído com média 0 e uma variância de $1/[144(n-k)]$. O intervalo de confiança $100(1-\alpha)\%$ para a autocovariância é:

$$R_k \mp z_{1-\alpha/2}/(12\sqrt{n-k})$$

Se este intervalo não inclui o valor 0, pode-se dizer que a sequência possui uma correlação significativa [13].

Para o Teste da Correlação Serial, os seguintes parâmetros foram adotados: $k = 1, 2, \ldots, 15$ e $n = 500, 1.000, 5.000, 10.000, 50.000, 100.000, 500.000, 1.000.000$ *e* 5.000.000.

Teste da Colisão

Suponha que existam m urnas e que n bolas são arremessadas aleatoriamente dentro delas, sendo a quantidade de urnas m (categorias) muito maior do que a quantidade de bolas n (observações). Diz-se que uma colisão ocorreu quando uma bola cair em uma urna que não esteja vazia. O *Teste da Colisão* conta justamente o número de colisões ocorridas.

A probabilidade de que uma determinada urna conterá exatamente k bolas é:

$$p_k = \left(\begin{array}{c} n \\ m \end{array} \right) m^{-k}(1-m^{-1})^{n-k}$$

Então o número esperado de colisões por urna é:

$$C = \sum_{k \geq 1} (k-1)p_k$$

Quando $m \gg n$, o número total médio de colisões em todas as m urnas é $C \approx \frac{n^2}{2m}$ [16]. Um determinado gerador falha nesse teste se o número total de colisões não está em um intervalo pré-definido.

Para o Teste da Colisão, os seguintes parâmetros foram adotados: $m = 2^{20}$ e $n = 2^{14}$. Assim: $C \approx 128$. Os limites no número total de colisões, correspondentes a 5% e 95% são 109 e 146, respectivamente [16]. Assim, o gerador passa no teste se o número total de colisões está nessa faixa.

Análise dos Resultados

Para a avaliação dos geradores nos testes clássicos será utilizado o Teste de Análise de Variância com dois fatores sem repetição [35]. Os dois fatores analisados serão o tipo de gerador e o tamanho de amostra utilizado em cada teste. Assim, pode-se testar duas hipóteses: (1) não existe diferença significativa entre o tipo de gerador e (2) não existe diferença significativa entre o tamanho de amostra. Se essas duas hipóteses forem confirmadas, pode-se afirmar também que não existe nenhuma relação entre os dois fatores estudados. Caso isso não ocorra, verifica-se o causador dessa dependência (tipo de gerador ou tamanho de amostra). O próximo passo agora é realizar um outro teste estatístico denominado Método *LSD (Least Significant Difference)* indicando quais as médias são significativamente diferentes no fator causador da dependência [35].

	1	2	...	m
1	x_{11}	x_{12}		x_{1m}
2	x_{21}	x_{22}		x_{2m}
$\vdots$				
n	x_{n1}	x_{n2}		x_{nm}

Tabela 3: Teste de Análise de Variância com dois fatores sem repetição

A Tabela 3 mostra uma configuração geral para o Teste de Análise de Variância com dois fatores sem repetição. Para os testes clássicos, as linhas representam os tipos de geradores, as colunas os tamanhos de amostra e x_{ij} a quantidade de erros que aconteceram em um total de 100 testes estatísticos (100 sementes) do gerador i para o tamanho de amostra j.

A Tabela 4 mostra os resultados da Análise Variância para os testes clássicos, onde A indica aceita-se a hipótese nula e R^* indica rejeita-se a hipótese nula. De um total de 33 testes clássicos, 23 testes aceitam a hipótese nula. Para esses testes, pode-se concluir que não existe nenhuma relação entre o tipo de gerador e o tamanho da amostra, e, individualmente, pode-se concluir que não há diferença significante entre o desempenho dos 3 geradores no teste assim como no tamanho de amostra dentro de cada teste. Dos 10 testes que rejeitaram a hipótese nula, 6 deles tiveram como causador da dependência o tipo de gerador e 4 o tamanho da amostra.

É interessante aplicar, principalmente para os testes que existem uma diferença siginificativa no tipo de gerador, o Teste *LSD* com o objetivo de mostrar quais as médias são significativamente diferentes. A Tabela 5 mostra a aplicação do Teste *LSD* nos 6 testes, onde um SIM em Dif_{CL}, Dif_{CMN} e Dif_{LMN} indica uma diferença significativa entre os geradores do *C* e do *Linus*, do *C* e de *Matsumoto/Nishimura* e do *Linus* e de *Matsumoto/Nishimura* e NÃO, caso contrário.

As Figuras 3 e 4 das médias dos erros para todos os tamanhos de amostras ajuda no entendimento da Tabela 5. Para o Teste Serial ($d = 3$), Correlação Serial ($k = 2$) e ($k = 8$), o gerador de *Matsumoto* é responsável pela diferença nas médias e não existe diferença significativa do ponto de vista estatístico entre o gerador do *C* e do *Linus*. Assim, esses dois geradores possuem uma melhor performance do que o gerador de *Matsumoto/Nishimura* nesses testes. Já nos testes de Correlação Serial ($k = 4$) e ($k = 11$), os geradores do *Linus* e de *Matsumoto/Nishimura* e do *C* e de *Matsumoto/Nishimura* obtém uma melhor performance, respectivamente. Por último, no Teste da Frequência, o gerador de *Matsumoto* é o melhor dentre os três.

Em geral, pode-se concluir que não existe uma diferença significativa entre os três geradores estudados nos testes clássicos e seus respectivos parãmetros de configuração e tamanhos de amostra.

3.2.2 Testes Rigorosos

Os testes projetados por *George Marsaglia*, denominado *Bateria de Testes Aleatórios Diehard*, são considerados na literatura mais difíceis que os testes clássicos tais como o Teste Chi-Quadrado e o Teste de Correlação Serial [12], [15], [24].

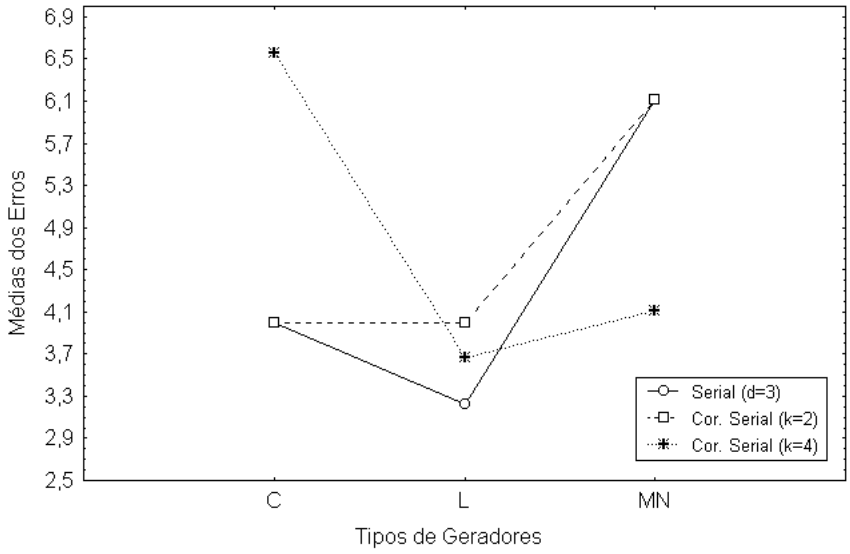

Figura 3: Gráfico 1 das Médias dos Erros

Esses testes fornecem como resultado uma série de $p-$valores. Uma evidência de falha muito séria ocorre quando os valores de saída são iguais a 0 ou 1 com seis ou mais casas decimais. Além disso, fornecem também como saída Testes KS dos próprios $p-$valores. Se a hipótese nula do Teste KS é rejeitada, o gerador de números aleatórios falha no teste. Mesmo se o Teste KS não rejeita a hipótese nula, deve-se observar também os $p-$valores individuais do teste. Se qualquer um desses valores for 0 ou 1, o gerador de números aleatórios também falha [33]. Como nos testes clássicos, um total de 100 sementes foram utilizadas. Para cada semente, gerou-se um arquivo binário de aproximadamente 270 Mbytes com 67 milhões de números aleatórios (inteiros de 32 bits) para que todos os testes do pacote fossem executados. Em média, executar todos os testes em cada semente levou cerca de 50 minutos. Por ser um pacote fechado, os tamanhos de amostra ficaram restritos aos configurados por *Marsaglia*.

Teste Espaçamentos do Aniversário

Escolha m aniversários, $I_1, I_2, \ldots, I_m$, em um ano de n dias. Os aniversários representam os inteiros produzidos pelo gerador de números aleatórios e o ano representa a variação dos números (entre 1 e n). Ordene os aniversários, $I_1 \leq I_2 \leq \ldots \leq I_m$. Seja j a quantidade de valores que aparecem mais de uma vez na lista dos espaçamentos $I_1, I_2 - I_1, I_3 - I_2, \ldots, I_m - I_{m-1}$; então j segue uma distribuição de *Poisson* com média $\lambda = m^3/(4n)$. Estudos experimentais mostram que o valor de n deve ser grande ($n \geq 2^{18}$). Utiliza-se, então, $n = 2^{24}$ e $m = 2^{10}$ para que a distribuição de j seja uma *Poisson* com $\lambda = (2^{30})/(2^{26}) = 16$. Aplica-se o Teste Chi-Quadrado em uma amostra de 200 valores de j. O primeiro teste utiliza os bits $1-24$ dos inteiros de 32 bits gerados (contados da esquerda para à direita). O segundo teste utiliza os bits $2-25$, o terceiro utiliza os bits $3-26$, e, assim sucessivamente, até os bits $9-32$.

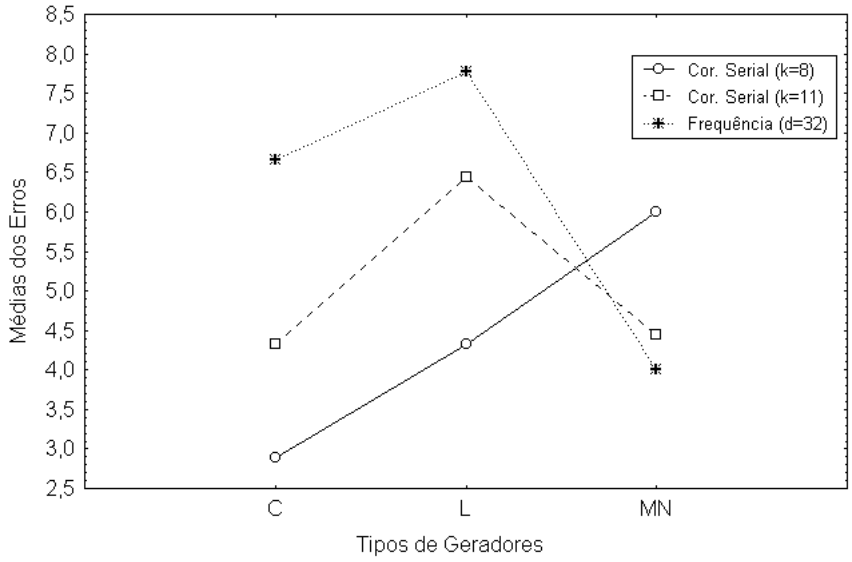

Figura 4: Gráfico 2 das Médias dos Erros

Uma variante desse teste utiliza $m = 2^{12} = 4096$ aniversários em um ano de $n = 2^{32}$ dias. Então, j segue uma distribuição de *Poisson* com $\lambda = 2^{36}/2^{34} = 4$. Cada conjunto de 4096 aniversários fornece um valor de j e aplica-se o Teste Chi-Quadrado em 500 desses valores com a finalidade de verificar se o resultado é consistente com uma distribuição de *Poisson* com $\lambda = 4$.

Teste do Máximo Divisor Comum

Sejam 2 inteiros sucessivos u, v produzidos pelo gerador de números aleatórios. Seja k o número de passos necessários para encontrar o Máximo Divisor Comum x de u e v através do algoritmo de *Euclides*. Então, k pode ser aproximado por uma distribuição binomial com $p = 0.376$ e $n = 50$, enquanto x possui uma distribuição muito próxima a $Pr(x = i) = c/i^2$ com $c = 6/pi^2$. Nsse teste são gerados 10 milhões de pares u, v e as frequências resultantes são comparadas com as distribuições de k e de x mencionadas acima.

Teste do Gorila

O Teste do Gorila procura testar bits específicos dos inteiros de 32 bits gerados. Especifica-se um determinado bit a ser estudado entre 0 e 31, por exemplo, o bit 3. Esse teste necessita de 67.108.889 ($2^{26} + 25$) milhões de números aleatórios e, a partir desses números, gera-se uma string de $2^{26}+25$ bits utilizando-se o bit 3. Contam-se, então, o número de segmentos de 26 bits que não aparecem nessa string. Esse valor aproxima-se de uma distribuição normal com média 24.687.971 e desvio padrão 4.170. O teste é aplicado em cada um dos bits (0 até 31).

Teste 5-Permutação Sobreposto

Este teste necessita de 10 milhões de inteiros de 32 bits. Cada conjunto de 5 inteiros consecutivos pode estar em um estado de 120 possíveis (5! ordenações). O teste conta a quantidade de ocorrências de cada estado. Então a forma quadrática na inversa fraca da matriz de covariância 120×120 produz um teste em que os 120 valores vêm de uma distribuição assintótica com médias específicas e covariância 120×120.

Teste do Rank Binário

Esse teste consiste em criar uma matriz binária 31×31 de valores $\{0, 1\}$ à partir dos 31 bits mais à esquerda dos 31 inteiros aleatórios da sequência a ser testada. O *rank* dessa matriz é determinado, com valores compreendidos entre 0 e 31. Como *ranks* < 28 são raros, as suas contagens são incorporadas aquelas com *rank* $= 28$. Geram-se 40.000 matrizes aleatórias e um Teste Chi-Quadrado é realizado entre as contagens observadas e esperadas para os *ranks* 31, 30, 29 e ≤ 28.

Duas variantes desse teste são utilizadas: o Teste do Rank Binário em matrizes 32×32 e matrizes 6×8. No primeiro, forma-se uma matriz binária 32×32 à partir dos 32 bits dos inteiros aleatórios e determina-se o seu *rank*. Os valores podem estar entre 0 e 32, contudo *ranks* < 29 são raros e as suas contagens são incorporadas com aquelas para o *rank* $= 29$. Geram-se 40.000 matrizes aleatórias e um Teste Chi-Quadrado é realizado entre as contagens observadas e esperadas para os *ranks* 32, 31, 30 e ≤ 29. No segundo, para cada 6 bits de 6 inteiros de 32 bits, um byte específico é escolhido formando-se uma matriz binária 6×8 cujo *rank* é determinado. Os valores podem estar entre 0 e 6, contudo ranks 0,1,2 e 3 são raros e suas contagens são incorporadas com aquelas para o *rank* $= 4$. Geram-se 100.000 matrizes aleatórias e um Teste Chi-Quadrado é realizado entre as contagens observadas e esperadas para os *ranks* 6, 5 e ≤ 4.

Teste do Fluxo de Bits

Consideram-se os números aleatórios como um fluxo de bits de palavras de 20 letras sobrepostas $(b_1, b_2, b_3, \ldots, b_{20})$, $(b_2, b_3, b_4, \ldots, b_{21})$, ... Em um alfabeto com 2 letras, 0 e 1, existem 2^{20} possíveis palavras de 20 letras. Nesse teste são geradas $2^{21} = 2.097.152$ palavras de 20 letras e conta-se o número de células vazias, isto é, palavras de 20 letras que não aparecem na sequência completa. Essa contagem aproxima-se de uma distribuição normal com média 141.909 e desvio padrão 428. Repete-se o teste 20 vezes.

Teste de Ocupação Esparsa dos Pares Sobrepostos

O Teste OPSO(Overlapping-Pairs-Sparse-Occupancy) considera palavras de 2 letras em um alfabeto de $2^{10} = 1024$ letras, sendo cada letra determinada por 10 bits específicos de um inteiro de 32 bits. Existem $2^{20} = 1.048.576$ palavras de 2 letras em um alfabeto de 1024 letras. Nesse teste são geradas $2^{21} = 2.097.152$ palavras de 2 letras (sobrepostas) e conta-se o número de células vazias, isto é, palavras de 2 letras que não aparecem na sequência completa. Essa

contagem aproxima-se de uma distribuição normal com média 141.909 e desvio padrão 290. A média e o desvio padrão foram teoricamente calculados.

Teste de Ocupação Esparsa das Quádruplas Sobrepostas

O Teste OQSO(Overlapping-Quadruples-Sparse-Occupancy) é similar ao anterior, contudo considera palavras de 4 letras em um alfabeto de $2^5 = 32$ letras, sendo cada letra determinada agora por 5 bits específicos. Existem $2^{20} = 1.048.576$ palavras de 4 letras em um alfabeto de 32 letras. Nesse teste são geradas $2^{21} = 2.097.152$ palavras de 4 letras (sobrepostas) e conta-se o número de palavras de 4 letras que não aparecem na sequência completa. Essa contagem aproxima-se de uma distribuição normal com média 141.909 e desvio padrão 295. A média foi teoricamente calculada enquanto o desvio padrão é o resultado de extensivas simulações.

Teste do DNA

O Teste do DNA considera palavras de 10 letras em um alfabeto de $2^2 = 4$ letras: C,G,A e T, sendo cada letra determinada por 2 bits específicos. Portanto, existem 2^{20} possíveis palavras de 10 letras em um alfabeto de 4 letras. Nesse teste são geradas $2^{21} = 2.097.152$ palavras de 10 letras (sobrepostas) e conta-se o número de palavras de 10 letras que não aparecem na sequência completa. Essa contagem aproxima-se de uma distribuição normal com média 141.909 e desvio padrão 339. A média foi teoricamente calculada enquanto o desvio padrão é o resultado de extensivas simulações.

Teste Contagem de 1´s em um Fluxo de Bytes

Considere o arquivo a ser testado como um fluxo de bytes (4 bytes para cada inteiro de 32 bits). Cada byte pode conter de 0 a 8 1´s com probabilidades 1/256, 8/256, 28/256, 56/256, 70/256, 56/256, 28/256, 8/256 e 1/256, respectivamente. Através do fluxo de bytes gera-se uma string de palavras de 5 letras sobrepostas, com cada letra podendo assumir os seguintes valores: A, B, C, D e E. As letras são determinadas pelo número de 1´s em um byte: 0, 1 ou 2 produz a letra A, 3 produz a letra B, 4 produz a letra C, 5 produz a letra D e 6,7 e 8 produz a letra E. Tem-se então as probabilidades das letras A, B, C, D e E iguais a 37/256, 56/256, 70/256, 56/256 e 37/256, respectivamente. Existem $5^5 = 3.125$ possíveis palavras de 5 letras e em uma string de 256.000 palavras, conta-se a frequência de cada palavra. A forma quadrática na inversa fraca da matriz de covariância das contagens da célula fornece um Teste Chi-Quadrado: Q5-Q4, a diferença das somas do coeficiente de *Pearson* $((Obs - Esp)^2/Esp)$ nas contagens para 4 e 5 letras.

Teste Contagem de 1´s em Bytes Específicos

Considere agora no arquivo a ser testado um byte específico de cada inteiro, como por exemplo, os bits de 1 até 8. Cada byte pode conter de 0 a 8 1´s com probabilidades 1/256, 8/256, 28/256, 56/256, 70/256, 56/256, 28/256, 8/256 e 1/256, respectivamente. Através do byte específico, gera-se uma string de palavras de 5 letras sobrepostas, com cada letra podendo assumir os seguintes

valores: A, B, C, D e E. As letras são determinadas pelo número de 1´s em um byte: 0, 1 ou 2 produz a letra A, 3 produz a letra B, 4 produz a letra C, 5 produz a letra D e 6,7 e 8 produz a letra E. Tem-se então as probabilidades das letras A, B, C, D e E iguais a (37/256, 56/256, 70/256, 56/256, 37/256). Existem 5^5 possíveis palavras de 5 letras e em uma string de 256.000 palavras, conta-se a frequência de cada palavra. A forma quadrática na inversa fraca da matriz de covariância das contagens da célula fornece um Teste Chi-Quadrado: Q5-Q4, a diferença das somas do coeficiente de *Pearson* $((Obs - Esp)^2/Esp)$ nas contagens para 4 e 5 letras.

Teste do Estacionamento Lotado

Em um quadrado de lado 100, aleatoriamente estacione um carro (um círculo de raio 1). Tente então estacionar um segundo, terceiro, e assim sucessivamente. Se uma tentativa de estacionar um carro causa uma batida com um já estacionado, tente novamente em uma nova localização aleatória. Cada tentativa conduz a uma batida ou a um sucesso, e observa-se, então, o número de carros estacionados com sucesso k depois de $n = 12.000$ tentativas. Resultados de simulações mostram que k aproxima-se de uma distribuição normal com média 3.523 e desvio padrão 21.9.

Teste da Distância Mínima

Primeiramente, escolhe-se $n = 8.000$ pontos aleatórios em um quadrado de tamanho 10.000. O teste consiste em encontrar a distância mínima d entre os $(n^2 - n)/2$ pares de pontos. Se os pontos são independentes e uniformes, então, o quadrado da distância mínima d^2 deveria se aproximar de uma distribuição exponencial com média 0.995. Esse teste é repetido 10 vezes e um Teste KS é empregado nos 10 valores p resultantes (p é calculado através da equação $1 - exp(-d^2/0.995)$).

Teste das Esferas 3D

Escolhe-se 4.000 pontos aleatórios em um cubo de aresta 1.000. Em cada ponto, centraliza-se uma esfera grande o suficiente para alcançar o próximo ponto mais próximo. Então, o volume da menor esfera aproxima-se de uma distribuição exponencial com média $120\pi/3$. Desta maneira, o raio do cubo aproxima-se também de uma exponencial com média 30. Esse valor foi obtido através de extensivas simulações. Esse teste gera 4.000 esferas e é repetido 20 vezes. Cada raio cúbico mínimo conduz a uma variável uniforme através da equação $1 - exp(-r^3/30)$, assim um Teste KS é realizado nos 20 $p - values$.

Teste da Compressão

Primeiramente, os números inteiros são transformados em números reais no intervalo $[0, 1)$ $(U(1), U(2), \ldots)$. Iniciando-se com $k = 2^{31} = 2.147.483.647$, o teste encontra o número de iterações necessárias j para reduzir o número k a 1, utilizando a equação $k = \lceil k \times U \rceil$, onde $\lceil x \rceil$ denota o menor inteiro maior ou igual a x. O teste é repetido 100.000 vezes e um Teste Chi-Quadrado é utilizado no número de js observados e esperados ($j \leq 6$,$j = 7$,$j = 8$, ...,$j = 47$ e $j \geq 48$).

Teste das Somas Sobrepostas

Primeiramente, os números inteiros são transformados em números reais no intervalo $[0, 1)$ $(U(1), U(2), \ldots)$. Em seguida, calculam-se somas sobrepostas, $S(1) = U(1) + \ldots + U(100)$, $S(2) = U(2) + \ldots + U(101)$, ... Essas somas são virtualmente normais com uma certa matriz de covariância. Uma transformação linear converte essas somas em uma sequência de variáveis normais padrões independentes, nas quais são também convertidas para variáveis uniformes por um Teste KS.

Teste das Rodadas Acima-Abaixo

Uma rodada acima de tamanho n possui $x_1 < x_2 < \ldots < x_n$ e $x_n > x_{n+1}$, enquanto uma rodada abaixo possui $x_1 > x_2 > \ldots > x_n$ e $x_n < x_{n+1}$. O número que finaliza a rodada não faz parte da rodada posterior, então uma longa sequência de números contém rodadas independentes, acima ou abaixo, de tamanho k com probabilidade $2k/(k+1)!$. Geram-se os números até atingir 100.000 rodadas acima e 100.000 rodadas abaixo. Testam-se as frequências dos tamanhos nas 100.000 rodadas para cada tipo e a quantidade de números necessários para atingir as 100.000 rodadas.

Teste Craps

Jogam-se 200.000 rodadas de *craps* e conta-se o número de vitórias e o número de arremessos necessários para terminar o jogo. O número de vitórias aproxima-se de uma distribuição normal com média $200.000p$ e variância $200.000p(1-p)$, sendo $p = 244/495$. Os arremessos necessários para finalizar o jogo podem variar de um até o infinito, mas contagens > 21 são acrescentadas ao valor 21. Um Teste Chi-Quadrado é então realizado na contagem do número de arremessos. Cada inteiro de 32 bits fornece um valor para o arremesso de um dado, primeiro transformando-o em um número entre $[0, 1)$, multiplicando-o por 6 e tomando-se 1 menos a parte inteira do resultado.

Uma variante desse teste é o *Teste Craps* com diferentes dados. Cada valor arremessado é determinado pelos 3 bits mais à direita dos inteiros de 32 bits gerados; valores de 1 até 6 são aceitos, enquanto os outros são rejeitados. Como no teste anterior, jogam-se 200.000 rodadas e conta-se o número de vitórias e o número de arremessos necessários para terminar o jogo. Novamente, o número de vitórias aproxima-se de uma distribuição normal com média $200.000p$ e variância $200.000p(1-p)$, sendo $p = 244/495$. Os arremessos necessários para finalizar o jogo podem variar de um até o infinito, mas contagens > 21 são acrescentadas ao valor 21. Um Teste Chi-Quadrado é então realizado na contagem do número de arremessos.

Análise dos Resultados

A Tabela 6 indica o resumo dos resultados da *Bateria de Testes Diehard* aplicadas aos geradores *Ansi C* (coluna C), do *Linus Schrage* (coluna L) e do *Matsumoto* e *Nishimura* (coluna M). Um valor P é um indicador de sucesso no teste, isto é, que o gerador na coluna correspondente passa no teste e um valor

F indica a falha no teste, isto é, que o gerador na coluna correspondente falha ao ser submetido ao teste específico.

O gerador de *Matsumoto* e *Nishimura* passa em todos os testes do pacote (23 testes), enquanto os demais geradores só passam em 5 testes. É interessante notar que a escolha das sementes iniciais não faz nenhuma diferença nos testes, isto é, uma falha (sucesso) de um gerador com uma semente em um teste implica em uma falha (sucesso) do gerador em todas as outras sementes no mesmo teste. Um padrão também ocorre para os geradores da mesma classe (*C* e *Linus*): eles falham e passam nos mesmos testes. Para esses geradores, os resultados são bastantes insatisfatórios nos bits de mais baixa ordem, o que comprova que os mesmos não são aleatórios mesmo quando o módulo é uma potencia de 2 (gerador do *C*) ou um número primo (gerador do *Linus*). Por exemplo, no Teste OPSO, OQSO e DNA, ambos os geradores falham nos bits de 1 a 10, 1 a 5 e 1 a 2, respectivamente. O gerador de *Matsumoto* e *Nishimura* mostrou um excelente comportamento estatístico em todos os testes.

3.3 Aspectos Práticos

3.3.1 Facilidade de Implementação e Eficiência

Implementar os Geradores Congruentes Lineares em uma linguagem de alto nível requer, em geral, uma certa habilidade porque o módulo m é tipicamente próximo do maior inteiro representável na máquina e os produtos envolvidos no cálculo geram *overflow* [18].

Para os módulos que são uma potência de 2 e utilizando-se de inteiros sem sinal sem a checagem de *overflow*, os produtos módulo m são fáceis de calcular: basta descartar o *overflow* ocorrido. Por esta razão, muitos Geradores Congruentes Lineares utilizam o módulo como uma potência de 2, devido a facilidade de implementação [18]. Esse deve ser o caso do gerador do *C*, contudo não se tem acesso ao código do gerador para uma análise mais criteriosa.

Para um módulo m mais geral, por exemplo, $m = p$, sendo p primo, representável como um inteiro no computador alvo, existe uma maneira facilmente implementável e eficiente de calcular $ax \ mod \ m$ para $0 < a, x < m$, denominado Método de Fatoração Aproximada, que deve obedecer a condição de que $a(m \ mod \ a) < m$. Esta condição é satisfeita se, e somente se, $a = i$ ou $a = \lfloor m/i \rfloor$ para $i < \sqrt{m}$ [18], [31].

Para implementar o Método de Fatoração Aproximada, inicialmente calcula-se uma única vez as constantes $q = \lfloor m/a \rfloor$ e $r = m \ mod \ a$. Então, para qualquer inteiro positivo $x < m$, a instrução abaixo possui o mesmo efeito do que a atribuição $x \leftarrow ax \ mod \ m$, mas com todos os resultados inteiros intermediários permanecendo estritamente entre $-m$ e m [18], [31]:

$y \leftarrow \lfloor x/q \rfloor$
$x \leftarrow a(x - yq) - yr$
$Se \ (x < 0) \ Entao \ x \leftarrow x + m$

A implementação portável do gerador do *Linus* é baseada nesse método [9], se a máquina é capaz de representar todos os inteiros no intervalo $[-2^{31}+1, 2^{31}-1]$. Sendo $m = 2^{31} - 1$ e $a = 16807$, o gerador satisfaz a condição, desde que $16807 < \sqrt{m}$ e, nesse caso, tem-se $q = 127773$ e $r = 2836$. Existem diversas

outras técnicas para calcular o produto módulo m e são melhores discutidas e comparadas em [36].

Para a implementação do gerador de *Matsumoto/Nishimura* necessita-se de um profundo conhecimento sobre o assunto. A recorrência utilizada para a geração dos números, por si só, dá uma noção do grau de dificuldade de entender e implementar esse tipo de gerador.

Em termos de utilização de memória, os geradores do *C* e do *Linus* necessitam de uma palavra de memória para inicialização, enquanto o gerador de *Matsumoto* necessita de 624 palavras de memória.

3.3.2 Possibilidade de Utilização de Técnicas Paralelas e Criação de Subsequências Disjuntas

Existem problemas adicionais ao se utilizar geradores desenvolvidos para um único processador em um ambiente com múltiplos processadores. Além dos requerimentos de uniformidade, independência e repetibilidade, o fluxo de números aleatórios utilizado por cada um dos processadores deve ser independente um do outro [14]. Assim, mesmo geradores confiáveis não são seguros quando submetidos as técnicas de paralelização [19].

Basicamente, existem os seguintes métodos para a geração de números aleatórios em processadores paralelos [19]: (i) pode-se associar L diferentes geradores a L diferentes processadores e (ii) pode-se associar L diferentes sub-fluxos de um gerador com um grande período a L processadores. A técnica (ii) possui duas variações: (a) Método *Leap-Frog* onde associa-se o subfluxo $(x_{nL+j})_{n\geq 0}$ ao processador j, $0 \leq j < L$ ou (b) Método da Divisão ou *Splitting* que associa um segmento completo $(x_n)_{n\geq n_j}$ ao processador j, onde $n_1, \ldots, n_L$ é um conjunto apropriado de valores iniciais que assegura a não sobreposição dos sub-fluxos.

A aproximação (i) não pode ser recomendada em geral pois pode existir intercorrelações fortes entre os fluxos produzidos pelo vários geradores [14]. A aproximação (ii) também não é muito segura devido as correlações existentes entre os subfluxos gerados por uma mesma semente no caso do Método *Leap-Frog* e os subfluxos gerados por sementes diferentes (Método da Divisão). Assim, recomenda-se cautela ao utilizar a aproximação (ii) na geração de números aleatórios em um ambiente paralelo [19].

No emprego das técnicas como *Leap-Frog* e Divisão, torna-se importante pular para partes da sequência rapidamente sem ter que gerar todos os estados intermediários. Em um Gerador Congruente Linear Multiplicativo ($c = 0$), pode-se utilizar a seguinte relação para saltar ν valores na sequência [31]:

$$x_{n+\nu} = (a^{\nu} \ mod \ m) x_n \ mod \ m$$

Exemplo - Calcular x_8 para $x_{n+1} = 5x_n \ mod \ 37$

Seja $a = 5$, $c = 0$, $m = 37$ e $x_0 = 1$. A sequência de inteiros aleatórios geradas por essa recorrência é: $1, 5, 25, 14, 33, 17, 11, 18, 16, 6, 30, \ldots$ Então, deseja-se ir diretamente para $x_8 = 16$. Assim $x_{n+8} = (a^8 \ mod \ 37)x_n \ mod \ 37 = (390625 \ mod \ 37)x_n \ mod \ 37 = 16x_n \ mod \ 37$. Portanto, $x_8 = 16x_0 \ mod \ 37 = 16$.

Já para o Gerador Congruente Linear Misto ($c \neq 0$), utiliza-se a seguinte relação [31]:

$$x_{n+\nu} = \left(\ a^{\nu} x_n + \tfrac{c(a^{\nu}-1)}{a-1} \ \right) \ mod \ m$$

Exemplo - Calcular x_8 para $x_{n+1} = (5x_n + 1)\ mod\ 16$

Seja $a = 5$, $c = 1$, $m = 16$ e $x_0 = 1$. A sequência de inteiros aleatórios geradas por essa recorrência é: $1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, 1, 6, \ldots$ Deseja-se ir diretamente para $x_8 = 9$. Assim, $x_{n+8} = (390625x_n + 97656)\ mod\ 16$. Portanto, $x_8 = (390625x_n + 97656)\ mod\ 16 = 9$.

Para um grande valor de ν, a operação $a \times a \times \ldots\ \nu$ vezes pode ser realizada em $O(log_2\nu)$ multiplicações através de um algoritmo de dividir e conquistar dado em [16].

Os geradores do *C* e do *Linus* possuem essa facilidade por serem Geradores Congruentes Lineares, porém possuem um período relativamente pequeno. Assim, devem ser utilizados com cautela em ambientes paralelos, evitando-se a sobreposição de sequências e, consequentemente, uma perda de computação. Devido ao imenso tamanho do período do gerador de *Matsumoto*, a probabilidade de se conseguir sequências sobrepostas é muito pequena, o que pode facilitar a utilização do mesmo em processamento paralelo.

3.3.3 Portabilidade

Existem diversas situações na qual é desejável que o gerador de números aleatórios seja independente da máquina, como, por exemplo, para propósitos de depuração. É importante então que um programa produza resultados que são os mesmos de máquina para máquina quando o tamanho da palavra do computador é suficiente dado a mesma entrada para o programa. Sob esse ponto de vista, o gerador do *Linus* é portável, o que não acontece com o gerador do *C* e de *Matsumoto/Nishimura*.

3.3.4 Velocidade dos algoritmos

Como forma de avaliar a velocidade dos algoritmos testados, mediu-se o tempo gasto para gerar diversos tamanhos de amostra de números aleatórios como mostra as Tabelas 7 e 8. Em ambas as Tabelas, a coluna n indica a quantidade de números aleatórios gerados, as colunas C, MN e L indicam, respectivamente, o tempo gasto pelos geradores de números aleatórios do *C*, de *Matsumoto/Nishimura* e do *Linus* para gerar os diversos tamanhos de amostra da coluna n. A Tabela 7 mostra o tempo médio gasto em microsegundos (μs) pelos geradores com tamanho de amostras menores (variando-se n de 1.000 até 1.000.000) e a Tabela 8 mostra o tempo médio gasto em segundos (s) pelos geradores com tamanho de amostras maiores (variando-se n de 5.000.000 até 2.000.000.000). Para cada tamanho de amostra, 100 sementes foram utilizadas e o tempo médio gasto por cada algoritmo para gerar n números aleatórios foi calculado.

Observando-se as Tabelas e os Gráficos, o gerador do *Linus* possui uma pior performance do que os outros geradores, independente do tamanho da amostra e existe uma pequena vantagem do gerador do *Matsumoto/Nishimura* sobre o gerador do *C*, com execeção de $n = 1.000$ e $n = 5.000$ números aleatórios gerados.

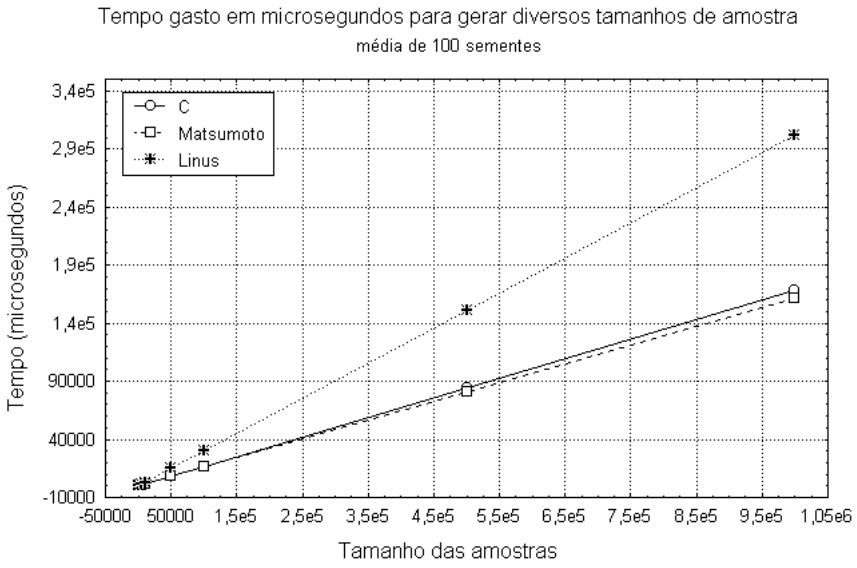

Figura 5: Tamanho de amostras menores

4 Considerações Finais

Este trabalho comparou três geradores de números aleatórios levando-se em consideração diversos aspectos: a análise estrutural ou testes teóricos dos geradores onde incluiu-se o estudo do tamanho do período de cada gerador e os testes de uniformidade no espaço $k-$dimensional (Teste Espectral e o Teste da $K-$Distribuição); a análise estatística ou empírica, considerando-se os testes clássicos com variações nos parâmetros e utilizando-se diversos tamanhos de amostra e os testes definidos no pacote *DieHard* considerados mais rigorosos na literatura; e questões de relevância puramente prática como facilidade de implementação, quantidade de memória utilizada, possibilidade de utilização de técnicas paralelas, métodos para pular adiante na sequência, portabilidade e velocidade do algoritmo.

Assim, baseando-se nos critérios acima mencionados e nos resultados obtidos nos testes podem ser feitas as seguintes recomendações:

- Devido aos avanços nos algoritmos de geração de números aleatórios e o aumento dos problemas de simulação que requerem uma quantidade cada vez maior de números, é aconselhável a utilização de geradores com grandes períodos. Essa propriedade também facilita a utilização desses geradores em um ambiente paralelo, com a ressalva de que as subsequências disjuntas devem ser testadas com o objetivo de avaliar a correlação entre as mesmas.

- É da natureza intrínseca dos Geradores Congruentes Lineares o aparecimento da estrutura *lattice* no espaço $k-$dimensional. Esses geradores também possuem um padrão cíclico nos bits menos significativos, comprovado pela rejeição nos testes mais rigorosos como o OQSO, OPSO e

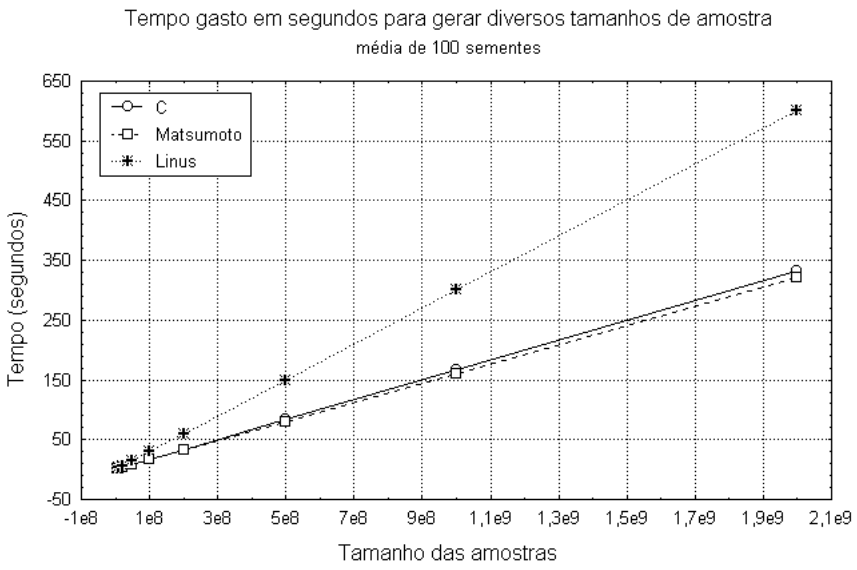

Figura 6: Tamanho de amostras maiores

DNA. O gerador de *Matsumoto* não é um gerador Maximamente Equidistribuído, mas possui boas propriedades teóricas.

- Em relação aos testes clássicos, não existe uma diferença significativa entre os geradores. Contudo, essa diferença apareceu nos testes mais rigorosos com uma clara diferença em favor do gerador de *Matsumoto*. Vale lembrar que nenhum gerador é a prova de falhas e os resultados dos testes estatísticos são específicos para os parâmetros e tamanhos de amostra estudados. Nada pode ser estendido para outros testes nem mesmo para os mesmos testes com configurações diferentes.

- A implementação do gerador de *Matsumoto* é extremamente difícil, a cargo dos especialistas na área o que não acontece com os Geradores Congruentes Lineares. A inicialização também requer mais memória, mas nada muito grande que afete a memória das máquinas atuais. A portabilidade é um ponto a favor do gerador do *Linus*, contudo quanto a velocidade do algoritmo para gerar diversos tamanhos de amostra, foi o que teve piores resultados. O gerador de *Matsumoto* chega a ser mais rápido que o gerador do *C* na maioria das amostras realizadas.

Assim, recomenda-se a utilização do gerador de *Matsumoto* e *Nishimura* devido a uma sólida base matemática aliada a uma boa performance empírica, excelente velocidade do algoritmo de geração e um período astronômico.

Referências

[1] L´Ecuyer, P., *Random Numbers*, Int. Encyc. Social and Behavioral Sciences, 4 June, 2001.

[2] Reeves, C. R., *Modern Heuristics Techiques for Combinatorial Problems*, Halsted Press: an Imprint of John Wiley & Sons,Inc., 1993.

[3] Aarts, E., and Korst, J. *Simulated Annealing and Boltzmann Machines - A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.

[4] Mladenovic, N., *A variable neighbourhood algorithm - a new metaheuristic for combinatorial optimization*, Presented at Optimization Days, Montreal, 1995.

[5] Mladenovic, N., and Hansen, P., *Variable Neighbourhood Search* , Computers and Operations, v. 24, pp. 1097-1100, 1997.

[6] Resende, M. G. C., and Ribeiro, C. C., *Greedy Randomized Adaptive Search Procedures (GRASP)*, State-of-the-Art Handbook of Metaheuristics (F. Glover e G. Kochenberger, eds.), 219-249, Kluwer, 2002.

[7] Grupo de Pesquisa, *Heurísticas e Paralelismo para Problemas de Otimização e de Bioinformática*, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, [on-line]: http://www-di.inf.puc-rio.br/∼ celso/grupo_de_pesquisa.htm, 2004.

[8] Ripley, B. D., *Thoughts on pseudorandom number generators*, Journal of Computational and Applied Mathematics, 31, pp. 153-163, 1990.

[9] Schrage, L., *A More Portable Fortran Random Number Generator*, ACM Transactions on Mathematical Software, v. 5, n. 2, p. 132-138.

[10] Matsumoto, M. and Nishimura, T. , *Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator*, ACM Transactions on Modeling and Computer Simulation, v. 8, n. 1, pp. 3-30, 1998.

[11] L´Ecuyer, P., *Uniform random number generation*, Annals of Operations Research, v. 53, pp. 77-120, 1994.

[12] Marsaglia, G., *A current view of random number generators*, in: Billard, Ed., Computer Science and Statistics: Proc. 16th Symposium on the Interface (North-Holland, Amsterdam), p. 3-10, 1985.

[13] Jain, R., *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling*, John Wiley & Sons, Inc, 1991.

[14] Eddy, W. F., *Random number generators for parallel processors*, Journal of Computational and Applied Mathematics, v. 31, p. 63-71, 1990.

[15] Anderson, S. L., *Random Numbers Generators on Vector Supercomputers and Other Advanced Architectures*, SIAM Review, v. 32, n. 2, p. 221-251, 1990.

[16] Knuth, D. E., *The Art of Computer Programming, v. 2 - Seminumerical Algorithms*, Third Edition, Addison Wesley Longman, 1998.

[17] Random Numbers, *Computacional Science Education Project*, [on-line]: $http://csep1.phy.ornl.gov/CSEP/PS\_FILES/RN.PS$.

[18] L´Ecuyer, P., *Random Numbers for Simulation*, Communications of the ACM, v. 33, n. 10, pp. 85-97, 1990.

[19] Hellekalek, P., *Good random number generators are (not so) easy to find*, Mathematics and Computers in Simulation, v. 46, pp. 485-505, 1998.

[20] Brent, R. P., *Uniform random number generators for supercomputers*, Proc. Fifth Australian Supercomputer Conference, pp. 95-104, Melbourne, December 1992, [on-line]: http://www.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub132.html.

[21] Matsumoto, M. and Kurita, Y. , *Twisted GFSR generators*, ACM Transactions on Modeling and Computer Simulation, v. 2, pp. 179-194, 1992.

[22] Matsumoto, M. and Kurita, Y. , *Twisted GFSR generators II*, ACM Transactions on Modeling and Computer Simulation, v. 4, pp. 254-266, 1994.

[23] L´Ecuyer, P., *Testing Random Numbers Generators*, Proceedings of the 1992 Winter Simulation Conference, pp. 305-313, 1992.

[24] L´Ecuyer, P. and Hellekalek, P. , *Random Number Generators: Selection Criteria and Testing*, in Random and Quasi-Random Point Sets, Lectures Notes In Statistics, n. 138, Springer, pp. 223-266, 1998.

[25] L´Ecuyer, P., Simard, R. , Chen, E. J. and kelton, W.D., *An Object-Oriented Random-Number Package with Many Long Streams and Substreams*, Operations Research v. 50, n.6, pp. 1073-1075, 2002.

[26] Coveyou, R.R. and MacPherson, R.D., *Fourier analysis of uniform random number generators*, Journal of the ACM, v. 14, pp. 100-119, 1967.

[27] L´Ecuyer, P., *Tables of Linear Congruential Generators of Differents Sizes and Good Lattice Structure*, Mathematics of Computation, v. 68, n. 225, pp. 249-260, 1999.

[28] Dieter, U., *How to Calculate Shortest Vectors in a Lattice*, Mathematics of Computation, v. 29, n. 131, pp. 827-833, 1975.

[29] Fishman, G.S. and Moore, L.R., *An exhaustive analysis of multiplicative congruential random number generators with modulus* $2^{31}-1$, SIAM Journal on Scientific and Statistical Computing, v. 7, n. 1, pp. 24-45, 1986.

[30] Fishman, G.S., *Multiplicative congruential random number generators with modulus* 2^{β}*: an exhaustive analysis for* $\beta = 32$ *and a partial analysis for* $\beta = 48$, Mathematics of Computation, v. 54, n. 189, pp. 331-344, 1990.

[31] L´Ecuyer, P., *Random number generation*, Handbook of Simulation (Chapter 4), Ed.: Jerry Banks, Wiley, 1998.

[32] L´Ecuyer, P., *Tables of Maximally-Equidistributed Combined LFSR Generators*, Mathematics of Computation, v. 68, n. 225, pp. 261-269, 1999.

[33] Marsaglia, G., *DIEHARD: a battery of tests of randomness*, [online]:http://stat.fsu.edu/∼geo/diehard.html, 1996.

[34] L'Ecuyer, P., *Uniform Random Number Generators*, Proceedings of the 1998 Winter Simulation Conference, Eds: D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, 1998.

[35] STATISTICA, *Livro texto do Statistica*, [on-line]-http://www.statsoft.com.br /download.html/textbook.zip.

[36] Fishman, G.S., *Monte Carlo - Concepts, Algorithms, and Applications*, Springer Series in Operations Research, Ed: Peter Glynn, 1996.

Testes	Configuração/Resultado	
Chi-Quadrado	$k = 5$	A
	$k = 10$	A
Frequência	$d = 32$	R^*
	$d = 64$	R^*
Serial	$d = 2$	A
	$d = 3$	R^*
	$d = 4$	A
	$d = 5$	A
Poker	—	A
Permutação	$t = 2$	A
	$t = 3$	A
	$t = 4$	A
	$t = 5$	R^*
Máximo de t	$t = 5$	A
	$t = 6$	A
Coletor de Cupons	—	A
Correlação Serial	$k = 1$	A
	$k = 2$	R^*
	$k = 3$	A
	$k = 4$	R^*
	$k = 5$	A
	$k = 6$	A
	$k = 7$	A
	$k = 8$	R^*
	$k = 9$	A
	$k = 10$	A
	$k = 11$	R^*
	$k = 12$	A
	$k = 13$	A
	$k = 14$	R^*
	$k = 15$	A
KS	—	A
Dois Níveis	—	R^*

Tabela 4: Resultados da Análise de Variância para os testes clássicos

Testes	Dif_{CL}	Dif_{CMN}	Dif_{LMN}
Serial ($d = 3$)	NÃO	SIM	SIM
Cor. Serial ($k = 2$)	NÃO	SIM	SIM
Cor. Serial ($k = 4$)	SIM	SIM	NÃO
Cor. Serial ($k = 8$)	NÃO	SIM	NÃO
Cor. Serial ($k = 11$)	SIM	NÃO	SIM
Frequência ($d = 32$)	NÃO	SIM	SIM

Tabela 5: Método *LSD* aplicado aos testes cujo causador da dependência é o tipo de gerador

Testes	**C**	**L**	**M**
Esp. do Aniver. I	F	F	P
Esp. do Aniver. II	F	F	P
MDC I	F	F	P
MDC II	P	P	P
Gorila	F	F	P
5-Permut. Sobrep.	P	P	P
Rank binário(31×31)	F	F	P
Rank binário(32×32)	F	F	P
Rank binário(6×8)	F	F	P
Fluxo de bits	F	F	P
OPSO	F	F	P
OPSO	F	F	P
DNA	F	F	P
Cont. de 1´s em um fluxo de bytes	F	F	P
Cont. de 1´s em bytes específicos	F	F	P
Estac. Lotado	F	F	P
Dist. Mínima	P	P	P
Esferas 3D	F	F	P
Compressão	F	F	P
Somas Sobrepostas	F	F	P
Rodadas Acima-Abaixo	P	P	P
Craps I	F	F	P
Craps II	P	P	P

Tabela 6: *Resumo dos Resultados* no pacote *Diehard*

n	C	MN	L
1.000	207.74	226.56	316.34
5.000	879.99	888.54	1520.61
10.000	1722.91	1690.08	3024.38
50.000	8489.52	8300.72	15080.81
100.000	16876.7	16199.46	30106.67
500.000	84737.59	80861.47	150997.77
1.000.000	168820.43	160853.59	301479.76

Tabela 7: Tamanhos de amostras menores

n	C	MN	L
5.000.000	0.86	0.79	1.53
10.000.000	1.68	1.63	3.00
20.000.000	3.33	3.21	6.02
50.000.000	8.36	8.05	15.02
100.000.000	16.65	16.08	30.02
200.000.000	33.35	32.18	60.04
500.000.000	83.33	80.37	150.05
1.000.000.000	166.62	160.74	300.14
2.000.000.000	333.25	321.51	600.18

Tabela 8: Tamanhos de amostras maiores