Pokroky matematiky, fyziky a astronomie

Ľubomíra Balková; Jiří Hladký

Generátory pseudonáhodných čísel založené na nekonečných slovech

Pokroky matematiky, fyziky a astronomie, Vol. 59 (2014), No. 3, 211–222

Persistent URL: http://dml.cz/dmlcz/144026

Generátory pseudonáhodných čísel založené na nekonečných slovech

Ľubomíra Balková, Jiří Hladký, Praha

V článku ukážeme, jak lze využít nekonečných aperiodických slov ke konstrukci generátorů náhodných čísel. Zavedeme třídu slov s **dobře rozmístěnými výskyty**, pro niž jsme dokázali, že při kombinaci periodických generátorů podle slov z této třídy vznikají generátory aperiodické, jež navíc nemají mřížkovou strukturu. Slova s dobře rozmístěnými výskyty zahrnují např. slova sturmovská a Arnouxova–Rauzyova slova, z nichž některá se dají generovat rychlým algoritmem, protože jsou pevnými body morfismů. Právě generátory pseudonáhodných čísel založené na pevných bodech morfismů, které mají dobře rozmístěné výskyty, jsme také otestovali bateriemi statistických testů TestU01 a PractRand. Ukázalo se, že nová metoda podstatně zlepšuje statistické vlastnosti generovaných náhodných čísel za cenu pouze zanedbatelně zvýšené výpočetní a paměťové náročnosti.

Úvod

Kombinatorika na slovech je cca 100 let stará disciplína, za jejíhož zakladatele se často pokládá Axel Thue. Ten při studiu jistých kombinatorických vlastností nekonečných slov [1] vysvětlil, že neměl na mysli žádnou konkrétní aplikaci, nýbrž studoval tyto otázky, protože se mu zdály samy o sobě dostatečně zajímavé. Tímto tvrzením dnes rádi hájí svou práci teoretičtí „kombinatorici na nekonečných slovech“. V tomto článku ovšem ukážeme, že se znalosti kombinatoriky na slovech dají aplikovat i v tak praktické oblasti, jakou je generování náhodných čísel.

Generátory pseudonáhodných čísel se snaží produkovat náhodná čísla použitím deterministického procesu. Je jasné, že takové generátory mají mnoho vad na kráse. Nejčastější a nejlépe prostudované generátory – lineární kongruenční generátory – jsou periodické a jejich známým defektem je tzv. mřížková struktura. V článku [8] je dokázáno, že když se dva lineární kongruenční generátory (ale i jakékoliv jiné periodické generátory) zkombinují podle nekonečných slov kódujících jistou třídu kvazikrystalů nebo ekvivalentně cut-and-project množin, výsledná posloupnost je aperiodická a nemá mřížkovou strukturu. Další výsledky spjaté s kvazikrystaly jsou k nalezení také v článcích [6], [7].

Naším přínosem je nalezení kombinatorické podmínky – **dobře rozmístěné výskyty** (DRV) – zaručující aperiodicitu a absenci mřížkové struktury [3], [2]. Ukázali jsme, že vlastnost DRV splňují široké třídy nekonečných slov: sturmovská slova, Arnouxova–Rauzyova slova a další, z nichž mnohé umíme generovat rychlým způsobem,

Ing. Ľubomíra Balková, Ph.D., Katedra matematiky, FJFI ČVUT v Praze, Trojanova 13, 120 00 Praha 2, e-mail: lubomira.balkova@fjfi.cvut.cz, hladky.jiri@gmail.com

protože jsou pevnými body morfismů. Tím jsme výsledky z [8] zobecnili pro širší třídy binárních slov, ale dokonce i pro slova nad vícepísmennou abecedou, podle kterých se kombinuje více lineárních kongruenčních generátorů (nebo i jiných periodických generátorů).

Poté jsme se věnovali testování kvality generátorů založených na slovech z takových tříd, konkrétně na Fibonacciově a Tribonacciově slově, přičemž jsme používali balíčky statistických testů TestU01 [5] a PractRand [4]. Tím jsme potvrdili, že když se kombinují lineární kongruenční generátory pomocí nekonečných slov s dobře rozmístěnými výskyty, vzniká aperiodická pseudonáhodná posloupnost, která nejen že nemá mřížkovou strukturu, ale i řada jejích dalších vlastností se více podobá náhodné posloupnosti.

V článku nejprve uvedeme motivaci z oblasti generování náhodných čísel. Ve druhé kapitole poté připomeneme pojmy z kombinatoriky na slovech, které se již v PMFA zčásti objevily [1]. Ve třetí kapitole zavedeme generátory založené na nekonečných slovech. Ve čtvrté kapitole potom prostudujeme kombinatorickou vlastnost dobře rozmístěných výskytů. Pátá kapitola shrne výsledky empirického testování a na závěr zformulujeme otevřené problémy a směry dalšího možného výzkumu.

1. Mřížková struktura generátorů pseudonáhodných čísel

Pod pojmem pseudonáhodná posloupnost nebo generátor pseudonáhodných čísel (PRNG – z anglického *pseudorandom number generator*) rozumíme v dalším textu jakoukoliv posloupnost čísel z $\mathbb{N} = \{0, 1, 2, \ldots\}$. Stále oblíbené generátory – lineární kongruenční generátory – jsou zářným příkladem generátorů, které mají slabinu zvanou mřížková struktura (a to dokonce již od dimenze rovné dvěma [10]). Nechť $Z = (Z_n)_{n\in\mathbb{N}}$ je PRNG, jehož výstupem je konečná množina $M \subset \mathbb{N}$. Říkáme, že Z má *mřížkovou strukturu*, pokud existuje kladné $t \in \mathbb{N}$ takové, že množina

$$\{(Z_i, Z_{i+1}, \ldots, Z_{i+t-1}) \mid i \in \mathbb{N}\}$$

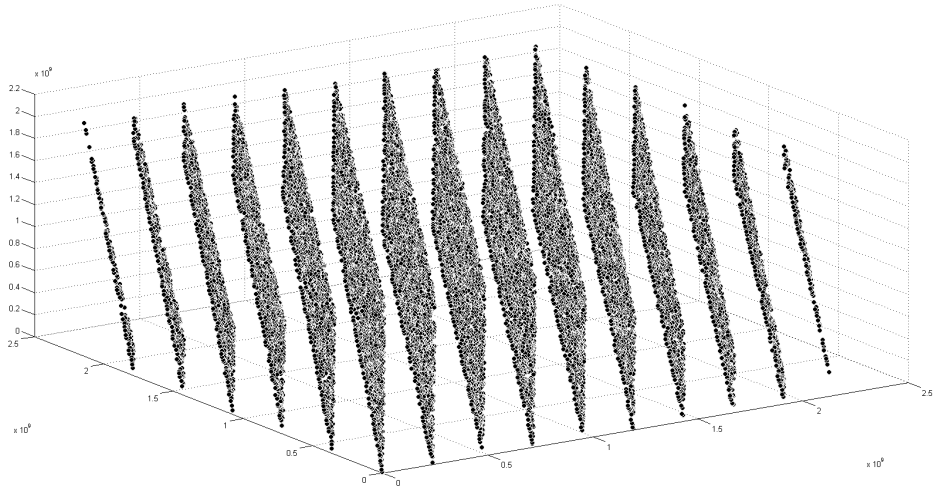
je pokryta soustavou rovnoběžných stejně vzdálených nadrovin v eukleidovském prostoru $\mathbb{R}^t$ a zároveň tyto nadroviny nepokrývají všechny body mřížky

$$M^t = \{(A_1, A_2, \ldots, A_t) \mid A_i \in M \text{ pro každé } i \in \{1, \ldots, t\}\}.$$

Připomeňme, že lineární kongruenční generátor, zkráceně LCG, $(Z_n)_{n\in\mathbb{N}}$ je dán parametry $a, m, c \in \mathbb{N}$ a definován rekurentním vztahem $Z_{n+1} = aZ_n + c \mod m$. Příkladem LCG s výraznou mřížkovou strukturou je generátor RANDU, který se hojně používal v šedesátých letech 20. století. Pro $t = 3$ jsou po sobě jdoucí trojice čísel z generátoru, tj. $\{(Z_i, Z_{i+1}, Z_{i+2}) \mid i \in \mathbb{N}\}$, pokryty pouhými 15 rovnoběžnými stejně vzdálenými rovinami, které ani zdaleka nepokrývají celou mřížku $\{1, 2, \ldots, m-1\}^3$, viz obr. 1.

Z článku [8] lze vyčíst postačující podmínku na absenci mřížkové struktury, i když přímo v něm je formulována v méně obecné formě (Lemma 2.3).

Věta 1. *Nechť Z je* PRNG, *jehož výstupem je konečná množina $M \subset \mathbb{N}$ obsahující alespoň dva prvky. Předpokládejme, že pro každé $A, B \in M$ a pro každé $\ell \in \mathbb{N}$ existuje ℓ-tice $(A_1, A_2, \ldots, A_\ell)$ taková, že jak $(A_1, A_2, \ldots, A_\ell, A)$, tak i $(A_1, A_2, \ldots, A_\ell, B)$ jsou $(\ell+1)$-tice z generátoru Z. Potom Z nemá mřížkovou strukturu.*



Obr. 1. Trojice generátoru RANDU, LCG s parametry $a = (2^{16} + 3), m = 2^{31}, c = 0$, jsou pokryty pouhými 15 rovnoběžnými stejně vzdálenými rovinami.

Poznámka 1. V řeči kombinatoriky na slovech, viz kapitola 2, lze znění předchozí věty formulovat následovně: Nechť Z je PRNG, jehož výstupem je konečná množina $M \subset \mathbb{N}$ obsahující alespoň dva prvky. Pokud Z obsahuje pro každá dvě písmena $A, B \in M$ a pro každou délku $\ell \in \mathbb{N}$ pravý speciální faktor ℓ s pravými extenzemi A a B, potom Z nemá mřížkovou strukturu.

2. Slovník kombinatoriky na slovech

Abychom mohli zformulovat kombinatorickou podmínku na absenci mřížkové struktury a také přiblížit třídy slov, které onu vlastnost dobře rozmístěných výskytů mají, potřebujeme nejprve zavést několik pojmů z kombinatoriky na slovech. *Abecedou* $\mathcal{A}$ rozumíme konečnou množinu symbolů, říkáme jim *písmena*. *Slovem* w nazýváme konečnou posloupnost písmen. Jeho *délkou* rozumíme počet písmen, která obsahuje, a značíme ji $|w|$. Symbolem $\mathcal{A}^*$ značíme množinu všech konečných slov nad abecedou $\mathcal{A}$ s přidáním prázdného slova ε (jde o neutrální prvek vůči operaci řetězení a jeho délku klademe rovnu nule). Množina $\mathcal{A}^*$ vybavená operací řetězení tvoří monoid. Pod *nekonečným slovem* $\mathbf{u}$ nad abecedou $\mathcal{A}$ pak rozumíme nekonečnou posloupnost písmen z $\mathcal{A}$, v níž se každé písmeno z $\mathcal{A}$ skutečně vyskytuje, tj. $\mathbf{u} = u_0u_1u_2\ldots$, kde $u_i \in \mathcal{A}$. Konečné slovo w nazveme *faktorem* (*podslovem*) konečného či nekonečného slova u, pokud existuje slovo v a slovo x (konečné či nekonečné) tak, že $u = vwx$. Je-li v prázdné slovo, nazveme w *prefixem* slova u. Podobně pokud je $x = \varepsilon$, říkáme, že w je *sufix* slova u.

Pro každý faktor w nekonečného slova $\mathbf{u}$ nazýváme *výskytem* w v $\mathbf{u}$ každý index i, pro který platí, že slovo w je prefixem nekonečného slova $u_iu_{i+1}u_{i+2}\ldots$ Nekonečné slovo je *rekurentní*, když se v něm každý jeho faktor vyskytuje nekonečněkrát. Symbolem $|w|_a$ značíme počet výskytů písmene a ve slově w.

Často pracujeme s morfismem $\varphi\colon \mathcal{A}^* \to \mathcal{A}^*$, tedy zobrazením, které splňuje pro každá dvě konečná slova $v, w \in \mathcal{A}^*$:

$$\varphi(vw) = \varphi(v)\varphi(w).$$

Morfismus je samozřejmě jednoznačně dán, pokud známe obrazy písmen abecedy $\mathcal{A}$. Jeho působení lze přirozenou cestou rozšířit i na nekonečná slova:

$$\varphi(u_0u_1u_2\ldots) := \varphi(u_0)\varphi(u_1)\varphi(u_2)\ldots$$

Pokud nekonečné slovo $\mathbf{u}$ splňuje $\varphi(\mathbf{u}) = \mathbf{u}$, nazveme je *pevným bodem* morfismu φ.

Jazykem $\mathcal{L}(\mathbf{u})$ nekonečného slova $\mathbf{u}$ nazýváme množinu všech faktorů tohoto slova. Říkáme, že jazyk je *uzavřený vzhledem k reverzi* (angl. *closed under reversal*), když s každým slovem obsahuje i jeho reverzi, tj. slovo vzniklé čtením pozpátku. Nechť n je přirozené číslo, symbolem $\mathcal{L}_n(\mathbf{u})$ značíme množinu všech faktorů délky n slova $\mathbf{u}$. Nekonečné slovo $\mathbf{u}$ nazýváme *posléze periodické* (angl. *eventually periodic*), pokud je tvaru $\mathbf{u} = wv^\omega$, kde $w, v \in \mathcal{A}^*$ a ω značí nekonečné opakování. V opačném případě nazýváme slovo $\mathbf{u}$ *aperiodické*. *(Faktorovou) komplexitou* slova $\mathbf{u}$ je zobrazení $\mathcal{C} : \mathbb{N} \to \mathbb{N}$ definované vztahem

$$\mathcal{C}(n) := \text{počet různých faktorů délky } n \text{ obsažených ve slově } \mathbf{u}.$$

Je známo, že komplexita posléze periodických slov je omezená, zatímco komplexita aperiodických slov splňuje $\mathcal{C}(n) \geq n + 1$ pro všechna $n \in \mathbb{N}$, viz [11].

Pravou extenzí faktoru w slova $\mathbf{u}$ nazveme libovolné písmeno $a \in \mathcal{A}$ takové, že wa je faktor slova $\mathbf{u}$. Zřejmě každý faktor má alespoň jednu pravou extenzi. Takové faktory, které mají alespoň dvě pravé extenze, se nazývají *pravé speciály*. Podobně lze definovat *levou extenzi* a *levé speciály*.

Příklad 1. Ilustrujme uvedené pojmy na nekonečném slově $\mathbf{u} = (abb)^\omega$. Abeceda je $\mathcal{A} = \{a, b\}$. Slovo *babb* je faktorem délky 4 slova $\mathbf{u}$. Jeho jedinou pravou extenzí je písmeno a a jeho jedinou levou extenzí je písmeno b. Slovo *abbabba* je prefixem délky 7 slova $\mathbf{u}$. Snadno nahlédneme, že množina všech faktorů délky 5 slova $\mathbf{u}$ je $\mathcal{L}_5(\mathbf{u}) = \{abbab, bbabb, babba\}$. Dále si také můžeme rozmyslet, že všechna slova délky alespoň dva mají vždy jedinou pravou (a také jedinou levou) extenzi, což vlastně znamená, že z každého faktoru délky alespoň dva se zrodí jediný faktor délky o jednu větší. Proto komplexita bude $\mathcal{C}(n) = 3$ pro každé $n \geq 2$. (To odpovídá tvrzení, že komplexita periodických slov je omezená.) Definujeme-li morfismus $\varphi(a) := abb$ a $\varphi(b) := abb$, pak je zřejmě $\mathbf{u}$ pevným bodem φ.

3. Kombinování generátorů pseudonáhodných čísel

Chceme-li eliminovat mřížkovou strukturu, pomáhá kombinovat generátory šikovným způsobem. Jednu z možných metod představili L.-S. Guimond, Jan Patera a Jiří Patera v článku [7]. Nechť $X = (X_n)_{n\in\mathbb{N}}$ a $Y = (Y_n)_{n\in\mathbb{N}}$ jsou dva ne nutně různé PRNG se stejným výstupem $M \subset \mathbb{N}$ a stejnou periodou $m \in \mathbb{N}$. Dále nechť $\mathbf{u} = u_0u_1u_2\ldots$ je nekonečné binární slovo nad abecedou $\{a, b\}$. Potom *generátor*

$$Z = (Z_n)_{n\in\mathbb{N}} \tag{1}$$

založený na slově $\mathbf{u}$ získáme následujícím algoritmem:

1. Čti krok za krokem písmena **u**.
2. Čteš-li a po i-té, zkopíruj i-tý symbol z X na konec konstruované posloupnosti Z.
3. Čteš-li b po i-té, zkopíruj i-tý symbol z Y na konec konstruované posloupnosti Z.

Snadno lze konstrukci zobecnit pro nekonečná slova nad vícepísmennou abecedou a kombinovat více PRNG.

Příklad 2. Nechť $X = (X_1X_2X_3X_4)^\omega$, $Y = (Y_1Y_2Y_3Y_4)^\omega$ a $\mathbf{u} = (abb)^\omega$. Potom

$$Z = X_1Y_1Y_2X_2Y_3Y_4X_3Y_1Y_2X_4Y_3Y_4X_4Y_1Y_2\ldots$$

4. Slova s dobře rozmístěnými výskyty

V článku [8] je dokázáno, že PRNG založený na nekonečných slovech kódujících jistou třídu cut-and-project množin je aperiodický a nemá mřížkovou strukturu. Nám se podařilo výsledek zobecnit a najít širší třídu slov, která zaručují aperiodicitu a absenci mřížkové struktury pro generátory založené na takových slovech [2], [3].

Definice 1. (Vlastnost DRV pro binární slova.) Říkáme, že aperiodické binární slovo **u** nad abecedou $\{a, b\}$ má *dobře rozmístěné výskyty*[1] (nebo má *vlastnost DRV*), pokud **u** splňuje pro každé $m \in \mathbb{N}$ a pro každý faktor w slova **u** následující podmínku. Označíme-li $i_0, i_1, i_2, \ldots$ výskyty slova w ve slově **u**, potom

$$\left\{\left(|u_0u_1\ldots u_{i_j-1}|_a, |u_0u_1\ldots u_{i_j-1}|_b\right) \bmod m \mid j \in \mathbb{N}\right\} = \mathbb{Z}_m^2\,,$$

kde mod m je aplikováno po složkách, $|w|_a$ značí počet výskytů písmene a ve slově w a $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$.

Vlastnost DRV definujeme pro aperiodická slova, protože je zřejmé, že nikdy neplatí pro slova posléze periodická. Pro slova s vlastností DRV jsme dokázali následující větu.

Věta 2. *Nechť Z je* PRNG *z* (1) *založený na binárním nekonečném slově* **u** *s vlastností* DRV. *Potom je Z aperiodický a nemá mřížkovou strukturu.*

Poznámka 2. Pokud bychom měli zadané konkrétní dva generátory s periodou m a právě pro ně chtěli garantovat absenci mřížkové struktury, stačilo by zkontrolovat, že platí vlastnost DRV pouze pro modul rovný m.

Příklad 3. Abychom viděli, že existují aperiodická slova, která vlastnost DRV nemají, prozkoumejme Thueovo–Morseovo slovo

$$\mathbf{t} = abbabaabbaababbabaababbaabbabaab\ldots,$$

které je pevným bodem morfismu $\varphi(a) = ab$, $\varphi(b) = ba$.

Vskutku, uvažujeme-li $m = 2$ a $w = aa$, potom z tvaru morfismu plyne, že se w vyskytuje pouze na lichých pozicích i_j. Například

$$\begin{array}{llll} i_0 = 5, & t_0\ldots t_4 = abbab, & (|t_0\ldots t_4|_a, |t_0\ldots t_4|_b) = (2,3), \\ i_1 = 9, & t_0\ldots t_8 = abbabaabb, & (|t_0\ldots t_8|_a, |t_0\ldots t_8|_b) = (4,5), \\ i_2 = 17, & t_0\ldots t_{16} = abbabaabbaababbab, & (|t_0\ldots t_{16}|_a, |t_0\ldots t_{16}|_b) = (8,9). \end{array}$$

[1] Angl. *words with well-distributed occurrences*

Proto je $(|t_0t_1\dots t_{i_j-1}|_a + |t_0t_1\dots t_{i_j-1}|_b) = i_j$ liché číslo. Tudíž například

$$(|t_0t_1\dots t_{i_j-1}|_a, |t_0t_1\dots t_{i_j-1}|_b) \bmod 2 \neq (0,0).$$

Definovali jsme kombinatorickou podmínku – dobře rozmístěné výskyty – zaručující absenci mřížkové struktury. Nyní je důležité najít slova, která takovou vlastnost mají. V článcích [2], [3] jsme dokázali, že *sturmovská slova*, což jsou slova s minimální komplexitou mezi slovy aperiodickými, tj. s komplexitou splňující $\mathcal{C}(n) = n+1$, vlastnost DRV mají. Uveďme alespoň nejznámějšího zástupce této třídy – slavné Fibonacciovo slovo: $\mathbf{f}$ je pevným bodem morfismu $\varphi(a) = ab$, $\varphi(b) = a$, tj.

$$\mathbf{f} = abaababaabaab\dots$$

Kromě zobecnění výsledku z článku [8] pro slova nad binární abecedou se nám podařilo také najít kombinatorickou podmínku pro slova nad vícepísmennou abecedou; generátory na nich založené pak kombinují více vstupních generátorů a opět nemají mřížkovou strukturu. Ukázalo se, že stačí nejpřirozenějším možným způsobem zobecnit definici binárního slova s dobře rozmístěnými výskyty.

Definice 2. (Vlastnost DRV pro vícepísmenná slova.) Říkáme, že aperiodické slovo $\mathbf{u}$ nad abecedou $\{a_1, a_2, \dots, a_d\}$ má *dobře rozmístěné výskyty* (nebo má *vlastnost DRV*), pokud $\mathbf{u}$ splňuje pro každé $m \in \mathbb{N}$ a pro každý faktor w slova $\mathbf{u}$ následující podmínku. Označíme-li $i_0, i_1, i_2, \dots$ výskyty slova w ve slově $\mathbf{u}$, potom

$$\left\{\left(|u_0u_1\dots u_{i_j-1}|_{a_1}, |u_0u_1\dots u_{i_j-1}|_{a_2}, \dots, |u_0u_1\dots u_{i_j-1}|_{a_d},\right) \bmod m \mid j \in \mathbb{N}\right\} = \mathbb{Z}_m^d,$$

kde mod m je aplikováno po složkách.

Poznámka 3. Vlastnost DRV je postačující, nikoliv nutná podmínka pro absenci mřížkové struktury. Není těžké ukázat, že modifikované Fibonacciovo slovo, které vzniklo z Fibonacciova slova tak, že jsme za každé písmeno napsali c, tj.

$$\mathbf{u} = acbcacacbcacbcac\dots,$$

nemá vlastnost DRV, ale mřížkovou strukturu také nemá. Jak se ale ukazuje ve statistických testech, vlastnost DRV je důležitá nejen pro absenci mřížkové struktury, ale zřejmě zaručuje i další dobré vlastnosti generátorů.

Příklad 4. Uveďme nejprve triviální příklad slova s dobře rozmístěnými výskyty. Říkáme, že nekonečné slovo nad abecedou $\mathcal{A}$ je *univerzální*, jestliže obsahuje všechna konečná slova nad $\mathcal{A}$ jako své faktory. Univerzální slovo nad $\{0, 1, \dots, d-1\}$ lze získat například řetězením zápisů po sobě jdoucích přirozených čísel v bázi d. Je snadné si rozmyslet, že univerzální slova mají vlastnost DRV.

Stejně jako nad binární abecedou i nad vícepísmennou abecedou je potřeba ukázat, že existuje široká třída slov, která mají vlastnost DRV. V hledání takového příkladu jsme uspěli, protože jsme zjistili, že *Arnouxova–Rauzyova slova* vlastnost DRV mají. Jde o slova, která mají jazyk uzavřený vzhledem k reverzi a pro každou délku n obsahují právě jeden pravý speciální faktor délky n, a tento pravý speciál má navíc jako pravé extenze všechna písmena abecedy. Všimněme si, že jde o přímé zobecnění sturmovských

slov na vícepísmenné abecedy, protože sturmovská slova jsou přesně takovými vlastnostmi charakterizována nad binární abecedou. Asi nejznámějším třípísmenným příkladem je Tribonacciovo slovo $\mathbf{u}$, které je pevným bodem morfismu $\varphi(a) = ab$, $\varphi(b) = ac$, $\varphi(c) = a$, tj.

$$\mathbf{u} = abacabaabacab\ldots$$

Poznámka 4. Existuje jednoduchá metoda, jak vyrábět ze slov s vlastností DRV opět slova s DRV vlastností nad abecedou s menším počtem písmen. Je-li abeceda rovna $\{a_1, a_2, \ldots, a_d\}$, kde $d \geq 3$, a slovo $\mathbf{u}$ má vlastnost DRV, potom pokud ve slově $\mathbf{u}$ přepíšeme písmeno a_d na jiné ale pevně dané písmeno a_i, bude mít výsledné slovo opět vlastnost DRV. Takovým způsobem získáme slova odlišná od sturmovských a Arnouxových–Rauzyových.

Poznámka 5. Další transformací slov, která zachovává vlastnost DRV a tentokrát zachovává i abecedu, je využití morfismu, jehož matice má determinant roven ± 1 neboli je unimodulární. Maticí morfismu φ nad abecedou $\{a_1, a_2, \ldots, a_d\}$ rozumíme matici $\mathbf{\Phi}$, jejíž ij-tý prvek je definován jako $\mathbf{\Phi}_{ij} = |\varphi(a_i)|_{a_j}$.

Příklad 5. Uvažujme morfismus $\varphi : a \to aab,\ b \to ab$. Pak jeho matice má tvar $\mathbf{\Phi} = \left(\begin{smallmatrix} 2 & 1 \\ 1 & 1 \end{smallmatrix}\right)$. Aplikujeme-li tento morfismus na Fibonacciovo slovo $\mathbf{f} = abaababaabaab\ldots$, pak dostaneme slovo $\varphi(\mathbf{f}) = aababaabaababaababaabaababaabaab\ldots$, které má také dobře rozmístěné výskyty.

5. Statistické testy generátorů pseudonáhodných čísel

V předchozí části jsme vysvětlili, že PRNG založené na slovech s DRV vlastností nemají mřížkovou strukturu. V této kapitole ukážeme, že také v empirických statistických testech dopadnou velmi dobře, za předpokladu, že jsou kombinovány dostatečně kvalitní LCG.

Pro testování jsou potřeba dlouhé prefixy nekonečných slov, podle kterých kombinujeme LCG. Naštěstí existuje celá řada sturmovských a Arnouxových–Rauzyových slov, která jsou pevnými body morfismů, a takové pevné body umíme efektivně generovat [12]: prefix délky n získáme v čase $O(n)$ a paměťové nároky jsou $O(\log n)$. Chceme-li generování zrychlit, je výhodnější si místo obrazů písmen $\varphi(a)$ pamatovat $\varphi^k(a)$ pro každé $a \in \mathcal{A}$. Právě takového vylepšení jsme využili, čímž se rychlost generování prefixů stala vyšší než rychlost generování výstupu kombinovaných LCG. Například pro získání 10^{10} 32bitových hodnot výstupu LCG s modulem 2^{64} jsme potřebovali 14.3 sekund, zatímco se stejným hardwarem jsme vygenerovali za půl sekundy 10^{10} písmen pevného bodu morfismu. Sečteno a podtrženo, použití PRNG založených na pevných bodech morfismů znamená oproti použití původních LCG pouze zanedbatelnou časovou penalizaci.

Dále ukážeme výsledky testování PRNG založených:

- na Fibonacciově slově (jako příklad slova sturmovského): jde o pevný bod morfismu $a \mapsto ab, b \mapsto a$,
- na modifikovaném Fibonacciově slově – Fibonacci2 – s písmenem c vloženým na každou druhou pozici (viz poznámka 3),
- na Tribonacciově slově (jako ilustraci ternárního Arnouxova–Rauzyova slova): jde o pevný bod morfismu $a \mapsto ab, b \mapsto ac, c \mapsto a$.

slovo	Fibonacci	Tribonacci
uložení φ	115 s	107 s
uložení φ^k	0.41 s	0.36 s

Tab. 1. Porovnání času v sekundách pro generování 10^{10} písmen Fibonacciova a Tribonacciova slova s využitím původního algoritmu [12] a jeho vylepšené verze. Počet iterací k v pravidle φ^k je vybírán tak, aby délka $\varphi^k(a)$ nepřekročila 4096 bajtů pro žádné písmeno z abecedy. Měření bylo provedeno na Intel Core i7-3520M CPU s frekvencí 2.90 GHz.

Implementovali jsme PRNG založené na pevných bodech morfismů pro více sturmovských a ternárních Arnouxových–Rauzyových slov. Jelikož jsou výsledky podobné, představíme zde právě jen výsledky pro tři výše uvedené zástupce. Náš program generující PRNG založené na pevných bodech morfismů je dostupný online spolu s popisem [9].

Do testů jsme zahrnuli modifikované Fibonacciovo slovo, které nemá vlastnost DRV, ale zároveň produkuje PRNG bez mřížkové struktury. Ovšem je u něj vidět, že výsledky v testech jsou horší než pro Fibonacciovo slovo, a to ačkoliv je Fibonacciovo slovo binární a modifikované Fibonacciovo slovo ternární.

Při kombinování LCG nepoužíváme všechny bity výstupu. Námi testované LCG mají periodu m v rozsahu od $2^{47} - 115$ do 2^{64}, ale my používáme pouze 32 horních bitů jako výstup, protože právě 32bitové posloupnosti jsou potřeba jako vstup sad statistických testů.[2]

Aplikujeme dvě sady testů náhodných čísel – TestU01 BigCrush a PractRand. Fungují rozdílně. První obsahuje 160 statistických testů, přičemž mnoho z nich je šito na míru konkrétním typům generátorů. Je to test s dobrým renomé, ovšem jeho nevýhodou je, že pracuje s pevným počtem bitů a nejnižší jeden až dva bity vždy zahazuje. Druhá sada se skládá ze tří různých testů, přičemž jeden se soustřeďuje na korelace na blízko, druhý na korelace na dálku a poslední je variací klasického „gap testu“ (sledování mezer). Navíc PractRand aplikuje na vstupní data automaticky různé filtry. Pro naše účely je zajímavý filtr dolních bitů – posílá na testování různý, ale předem daný pevný počet bitů z výstupu generátoru (tím se dá kontrolovat, zda byla odstraněna výše zmíněná slabina, kdy dolní bity LCG s modulem rovným mocnině dvojky mají menší periodu). PractRand umí testovat velmi dlouhé posloupnosti, až do několika exabajtů. Abychom kontrolovali čas, omezili jsme se na vstupu na posloupnosti délky 16 TB.

První sloupec tabulky 2 ukazuje testované LCG. Sloupec BigCrush udává, kolik testů sady TestU01 BigCrush neuspělo. Sloupec PractRand udává $\log_2$ délky vstupních dat v bajtech, pro která začaly být výsledky PractRand velmi podezřelé (p-hodnota menší než 10^{-5}). Jeden LCG neukázal v PractRand testu žádné slabiny, což jsme označili jako > 44. Poslední sloupec ukazuje čas v sekundách pro generování prvních 10^{10} 32bitových posloupností výstupu na Intel i7-3520M CPU s frekvencí 2.90 GHz.

Z tabulky 2 vidíme, že LCG s $m \in \{2^{47} - 115, 2^{63} - 25\}$ dávají nejlepší statistické výsledky. Zároveň je ale čas generování jejich výstupu 20krát pomalejší než

[2]Nepoužíváme přímo LCG s modulem 2^{32}, protože u takových generátorů je známo, že k-tý bit LCG s modulem 2^ℓ pro nějaké $\ell \in \mathbb{N}$ má periodu délky pouze 2^k. Navíc i generátory, které nemají modul ve tvaru 2^ℓ, ale mají periodu blízkou 2^{32}, zůstávají i po kombinování podle slov s dobře rozmístěnými výskyty v testech příliš slabé.

Generátor	Zkratka	BigCrush	PractRand	Čas 10^{10}
LCG$(2^{47}-115, 71971110957370, 0)$	L47-115	14	40	281 s
LCG$(2^{63}-25, 2307085864, 0)$	L63-25	2	>44	277 s
LCG$(2^{59}, 13^{13}, 0)$	L59	19	27	14.1 s
LCG$(2^{63}, 5^{19}, 1)$	L63	19	33	14.4 s
LCG$(2^{64}, 2862933555777941757, 1)$	L64_28	18	35	14.0 s
LCG$(2^{64}, 3202034522624059733, 1)$	L64_32	14	34	14.1 s
LCG$(2^{64}, 3935559000370003845, 1)$	L64_39	13	33	14.0 s

Tab. 2. Seznam použitých LCG(m, a, c) a jejich výsledky v testech BigCrush a PractRand

u ostatních použitých LCG. To je dáno faktem, že operaci modulo musíme opravdu počítat, zatímco modulování 2^ℓ odpovídá posouvání binární tečky.

V tabulkách 3, 4, 5 ukážeme výsledky PRNG založených na Fibonacciově, modifikovaném Fibonacciově slově a Tribonacciově slově při kombinování různých LCG z tabulky 2. (Kombinujeme tak, že čtení písmene a ve slově odpovídá použití výstupu generátoru ve sloupci a a analogicky pro další písmena b, c a sloupce b, c.) To zahrnuje také použití různých instancí stejného generátoru. Všechny LCG mají jako násadu číslo 1. PRNG pak byly "zahřáty" generováním 10^9 hodnot, než byly spuštěny statistické testy. Protože frekvence písmen jsou odlišné (např. u Fibonacciova slova je poměr frekvence a ku b roven $\tau = \frac{1}{2}(1+\sqrt{5}) \doteq 1.618$), zahřívací kolo způsobí, že i dvě instance stejného generátoru se budou po chvíli hodně lišit. Sloupec BigCrush používá následující značení: první číslo indikuje, kolik testů sady BigCrush neuspělo, a druhé číslo v závorce udává, kolik testů mělo podezřele malou p-hodnotu v rozsahu od 10^{-6} do 10^{-4}. Sloupec PractRand ukazuje $\log_2$ délky vstupních dat v bajtech, pro které začaly být výsledky PractRand velmi podezřelé (p-hodnota menší než 10^{-5}). Maximální objem vstupních dat byl 16TB $\doteq 2^{44}$B. Sloupec s časem udává dobu generování 10^{10} 32bitových slov na Intel i7-3520M CPU s frekvencí 2.90 GHz. Zdrojový kód je dostupný v [9].

Na základě výsledků statistických testů jsme učinili následující pozorování:

1. Kvalita LCG se podstatně zlepšila, když jsme je zkombinovali podle slov s dobře rozmístěnými výskyty. Toto je dobře patrné v testu BigCrush. Zatímco pro původní LCG neuspělo 13 až 19 testů (jedinou výjimkou byl generátor L63-25 s dvěma neúspěchy – viz tabulka 2), po zkombinování téměř všechny BigCrush testy prošly. Nejhorším výsledkem byl jeden neúspěšný test pro kombinaci podle Tribonacciova slova, resp. podle Fibonacciova slova pro LCG L47-115. Pravděpodobnou příčinou je ovšem nejkratší perioda tohoto generátoru mezi všemi uvažovanými LCG.

   Výsledky sady PractRand také potvrzují zlepšení. Například v případě LCG s modulem 2^{64} začal test nacházet neregularity v distribuci posledního bitu až pro výstup velikosti 2TB. Čtenář nechť tento fakt porovná s velikostí dat 8 GB až 32 GB, kdy původní LCG vykazovaly slabiny v tomto testu. Sada testů PractRand aplikuje na vstupní data různé filtry a všechny chyby se objevily pro filter `Low1/32`, kde je kontrolována distribuce pouze posledního bitu. Bylo by

Slovo	Skupina	a	b	BigCrush	PractRand	Čas 10^{10}
Fib	A	L64_28	L64_28	0	41	30.2 s
	A	L64_32	L64_28	0(1)	41	29.3 s
	A	L64_39	L64_28	0 (2)	41	31 s
	A	L64_28	L64_32	0	41	30.2 s
	A	L64_32	L64_32	0	41	30.1 s
	B	L47-115	L47-115	1(1)	>44	302 s
	B	L63-25	L63-25	0(1)	>44	299 s
	B	L59	L59	0(1)	34	28.7 s
	C	L63-25	L59	0	38	198 s
	C	L59	L63-25	0(1)	35	134 s
	C	L63-25	L64_39	0	>44	199 s
	C	L64_39	L63-25	0	41	135 s
	C	L59	L64_39	0	35	30.4 s
	C	L64_39	L59	0	37	31.3 s

Tab. 3. Shrnutí výsledků statistických testů pro generátory založené na Fibonacciově slově a různých kombinacích LCG z tabulky 2

tedy jistě možné zlepšit kvalitu generátoru tím, že by se ze vstupních bitů bralo pouze 16 horních bitů, případně kombinováním generátorů, které nemají modul tvaru 2^ℓ.

2. Kvalita kombinovaných generátorů silně závisí na kvalitě kombinovaných generátorů, viz např. některé generátory skupiny B v tabulkách 3, 4, 5, které měly dobré výsledky v testu PractRand již v původním tvaru.
3. Dalším zajímavým pozorováním je fakt, že užíváním LCG se stejnými parametry a různou násadou (inicializační hodnotou) nic nepokazíme. Jen je třeba pohlídat, aby byly počáteční stavy dostatečně různé, viz skupina A a B v tabulkách 3, 4 a 5.
4. Kombinujeme-li LCG různé kvality, pak LCG nejnižší kvality určuje kvalitu výsledného generátoru, viz skupina C v tabulkách 3, 4, 5. Pokud tedy kombinujeme LCG různé kvality, je dobré použít nejlepší z nich tak, aby odpovídal nejčastějšímu písmenu ve slově s vlastností DRV.
5. Na druhé straně, když kombinujeme generátory stejné kvality, pak na jejich pořadí nezáleží, viz skupina A v tabulkách 3, 4, 5.
6. Modifikované Fibonacciovo slovo neprodukuje lepší výsledky než Fibonacciovo slovo, ačkoliv je nad větší abecedou. Je to pochopitelné, protože vzniklo pravidelným vkládáním nového písmena za každé písmeno Fibonacciova slova.
7. Tribonacciovo slovo ukazuje lepší výsledky než Fibonacciovo slovo. Takové pozorování bylo platné pro každé ternární Arnouxovo–Rauzyovo slovo v porovnání se sturmovskými slovy.

Slovo	Skupina	a	b	c	BigCrush	PractRand	Čas 10^{10}
Fib2	A	L64_28	L64_28	L64_28	0	40	28.4 s
	A	L64_39	L64_28	L64_28	0(2)	40	27.9 s
	A	L64_39	L64_32	L64_28	0	39	27.5 s
	A	L64_28	L64_39	L64_28	0	40	27.3 s
	A	L64_32	L64_39	L64_28	0	40	27.5 s
	B	L47-115	L47-115	L47-115	0(2)	>44	297.0 s
	B	L63-25	L63-25	L63-25	0(2)	>44	293.0 s
	B	L59	L59	L59	0(1)	32	27.4 s
	B	L63	L63	L63	0	38	27.3 s
	C	L63-25	L59	L64_39	0(1)	39	113.0 s
	C	L63-25	L64_39	L59	0	32	113.0 s
	C	L59	L63-25	L64_39	0	38	81.1 s
	C	L59	L64_39	L63-25	0	39	158.3 s
	C	L64_39	L63-25	L59	0	31	81.0 s
	C	L64_39	L59	L63-25	0	42	159.0 s

Tab. 4. Shrnutí výsledků statistických testů pro generátory založené na modifikovaném Fibonacciově slově a různých kombinacích LCG z tabulky 2

Slovo	Skupina	a	b	c	BigCrush	PractRand	Čas 10^{10}
Trib	A	L64_28	L64_28	L64_28	0(2)	42	27.2
	A	L64_39	L64_28	L64_28	0	43	27.1
	A	L64_39	L64_32	L64_28	0(1)	42	28.0
	A	L64_28	L64_39	L64_28	0(1)	42	28.1
	A	L64_32	L64_39	L64_28	0	42	27.1
	B	L47-115	L47-115	L47-115	1	>44	299.0
	B	L63-25	L63-25	L63-25	0(1)	>44	298.0
	B	L59	L59	L59	0	35	27.2
	B	L63	L63	L63	0(1)	41	27.2
	C	L63-25	L59	L64_39	0(1)	39	172.0
	C	L63-25	L64_39	L59	0(1)	41	173.0
	C	L59	L63-25	L64_39	0	35	106.0
	C	L59	L64_39	L63-25	0	34	70.5
	C	L64_39	L63-25	L59	0	41	107.0
	C	L64_39	L59	L63-25	0(1)	40	74.3

Tab. 5. Shrnutí výsledků statistických testů pro generátory založené na Tribonacciově slově a různých kombinacích LCG z tabulky 2

6. Otevřené problémy a další výzkum

Otevřených problémů zůstává mnoho. Co se týče kombinatorické části článku, bylo by dobré najít další velké třídy slov s dobře rozmístěnými výskyty a především by bylo dobré vědět, které pevné body morfismů mají vlastnost DRV. Také se zdá smysluplné studovat vlastnost DRV jen pro některé speciální moduly, kdy v definici vlastnosti DRV uvažujeme jedno konkrétní m místo libovolného modulu. V souvislosti se statistickými testy je pole působnosti ještě větší – kromě aperiodicity a absence mřížkové struktury není dosud žádný další úspěch v testech vysvětlen teoreticky. Samozřejmě pokračujeme také ve statistických testech, kdy kombinujeme jiné než LCG generátory a výsledky porovnáváme s obdobně rychlými PRNG.

Poděkování. První z autorů děkuje za finanční podporu grantu GAČR 13-03538 a L'Oréalu Česká republika za stipendium Pro ženy ve vědě.

Literatura

[1] Balková, Ľ.: *Nahlédnutí pod pokličku kombinatoriky na nekonečných slovech.* PMFA *56* (2011), 9–18.

[2] Balková, Ľ., Bucci, M., De Luca, A., Puzynina, S.: *Infinite words with well distributed occurrences.* J. Karhumäki, A. Lepistö, L. Zamboni (eds): Combinatorics on Words, LNCS *8079*, Springer, 2013, 46–57.

[3] Balková, Ľ., Bucci, M., De Luca, A., Hladký, J., Puzynina, S.: *Pseudorandom number generators based on infinite words.* Zasláno do Math. Comp. (2013), dostupné z arXiv: 1311.6002.

[4] Doty-Humphrey, C.: *Practically Random: C++ library of statistical tests for RNGs.* Dostupné z: `https://sourceforge.net/projects/pracrand`

[5] L'Ecuyer, P., Simard, R.: *TestU01: A C library for empirical testing of random number generators.* ACM Trans. Math. Software *33 (4)* (2007).

[6] Guimond, L.-S., Patera, J.: *Proving the deterministic period breaking of linear congruential generators using two tile quasicrystals.* Math. Comp. *71 (237)* (2002), 319–332.

[7] Guimond, L.-S., Patera, J., Patera, J.: *Combining random number generators using cut-and-project sequences.* Czechoslovak J. Phys. *51* (2001), 305–311.

[8] Guimond, L.-S., Patera, J., Patera, J.: *Statistical properties and implementation of aperiodic pseudorandom number generators.* Appl. Numer. Math. *46 (3–4)* (2003), 295–318.

[9] Hladký J.: *Random number generators based on the aperiodic infinite words.* Dostupné z: `https://github.com/jirka-h/aprng`

[10] Marsaglia, G.: *Random numbers fall mainly in the planes.* Proc. Natl. Acad. Sci. USA *61 (1)* (1968), 25–28.

[11] Morse, M., Hedlund, G. A.: *Symbolic dynamics.* Amer. J. Math. *60* (1938), 815–866.

[12] Patera, J.: *Generating the Fibonacci chain in $O(\log n)$ space and $O(n)$ time.* Phys. Part. Nuclei *33* (2002), 118–122.