Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Diplomová práca

Autor: Libor Havlíček

Vedúci: doc. RNDr. Daniel Olejár, PhD.

Bratislava, 2007

Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Generátory pseudonáhodných čísel

Libor Havlíček

Vedúci: doc. RNDr. Daniel Olejár, PhD

Bratislava, 2007

Čestne prehlasujem, že som prácu vypracoval samostatne s použitím uvedenej literatúry a iných zdrojov.

.....

Poďakovanie

Touto cestou ďakujem vedúcemu diplomovej práce doc. RNDr. Danielovi Olejárovi, PhD. za cenné odborné rady, pripomienky a návrhy, ktoré mi pomohli pri realizácií výslednej podoby tejto práce.

Licencia

Abstrakt

Cieľom tejto práce bolo popísať existujúce ako aj navrhnúť a zdôvodniť nové spôsoby testovania kvality "náhodnosti" postupnosti, resp. generátora pseudonáhodných čísel na základe jeho výstupnej postupnosti. Ďalej zorientovať sa v problematike a navrhnúť konkrétnu batériu testov, či postup ako najlepšie pristupovať k testovaniu a následne svoj výskum odskúšať na dostupných generátoroch. Na záver bolo mojou úlohou navrhnúť ďalší postup pri štúdiu náhodných/pseudonáhodných generátorov.

Príloha vo forme CD obsahuje dokumenty potrebné k štúdiu generátorov, k ich implementácií, ako aj množstvo dokumentov nevyhnutných k testovaniu a štúdiu náhodnosti. Ďalej obsahuje voľne dostupné batérie testov, ako aj mnou naimplementovaný jednoduchý systém na prehľadnejšiu analýzu výsledkov testovania.

Obsah

Kapitola 1

Úvod

1.1 Základné priblíženie problematiky

Kryptografické riešenia, ktoré sa využívajú na zaistenie bezpečnosti IKT systémov, závisia podstatne od kvality kryptografických kľúčov, inicializačných reťazcov, salt a nonce a ďalších podobných údajov. Ak by protivník bol schopný zistiť, resp. aspoň podstatne redukovať množinu potenciálnych hodnôt kryptografických kľúčov (resp. iných údajov s podobným určením), môže úspešne zaútočiť na systém, ktorý dané kryptografické riešenie využíva. Preto je kľúčovou požiadavkou, na ktorej stojí bezpečnosť kryptografického riešenia rovnako, ako robustnosť použitého kryptosystému, spoľahlivé generovanie náhodných parametrov systému (FIPS 140-1,2). Okrem relatívne krátkych kryptografických kľúčov, inicializačných, doplňujúcich a iných náhodných vektorov sa v kryptológii používajú aj "dlhé" pseudonáhodné postupnosti v tzv. prúdových šifrách, simulujúcich Vernamovu šifru. Z uvedených dôvodov sa generovaniu pseudonáhodných/náhodných čísel venuje v kryptológii veľká pozornosť. Pri generovaní "krátkych" pseudonáhodných/náhodných reťazcov je možné použit fyzikálne generátory (šumové diódy, generátory založené na spracovaní rádioaktívneho rozpadu a i.) a náhodný parameter (napr. kryptografický kľúč) doručiť používateľovi spoľahlivým kanálom. V prípade "dlhých" náhodných postupností (prúdové šifry) by to situáciu značne kom-

plikovalo, pretože náhodný reťazec sa použije len raz a príjemca potrebuje pre každú zašifrovanú správu nový dešifrovací kľúč. Preto sa v prípade prúdových šifier používajú tzv. pseudonáhodné generátory, ktoré sú realizované deterministickými algoritmami a vytvárajú postupnosť "náhodne vyzerajúcich čísel". Skutočne náhodným parametrom je inicializačný vektor, ktorý aj odosielateľovi aj príjemcovi správy umožní generovať tú istú postupnosť náhodne vyzerajúcich čísel, ktorá sa použije na šifrovanie/dešifrovanie správy. Keďže fyzikálne generátory náhodných čísel nie sú práve lacné ani praktické (a naviac technické poruchy, vplyvy prostredia môžu výrazne ovplyvniť výstupnú postupnosť), v reálnom živote sa využívajú namiesto generátorov náhodných čísel generátory pseudonáhodných čísel, postavené na deterministických algoritmoch, resp. využívajúce náhodné udalosti v systéme (napr. rozličné časové údaje). Kvalita takýchto generátorov je rozličná a absolútne kritériá na jej posudzovanie neexistujú. Preto má zmysel skúmať výstupy generátorov (náhodných aj pseudonáhodných) čísel a vyradiť tie, ktoré sú zjavne nedostačujúce. Náhodnosť je kľúčovým pojmom teórie pravdepodobnosti a existuje teória popisujúca reťazce náhodné zo štatistického hľadiska. Prostriedkami na posudzovanie toho, do akej miery spĺňa postupnosť štatistické požiadavky na náhodnú postupnosť, sú štatistické testy. Náhodnosť zo štatistického hľadiska však kryptografickým účelom nepostačuje. Výstupy LFSR s vhodne zvolenou spätnou väzbou majú skvelé štatistické parametre, ale bez ďalšej úpravy sú na šifrovanie nepoužiteľné, pretože tieto postopnosti sú ľahko zrekonštruovateľné už na základe poznania relatívne malého počtu členov postupnosti. V kryptológii teda potrebujeme postupnosti, ktoré sú štatisticky náhodné (aby množina potenciálnych náhodných parametrov nebola príliš zúžená a neumožnila úplné preberanie) a ťažko predvídateľné na základe znalosti niekoľkých hodnôt. Existuje teória prúdových šifier, štatistická/pravdepodobnostná teória náhodných postupností a štatistické testy na skúmanie náhodne vyzerajúcich postupností. Novinkou je kvantové generovanie náhodných čísel, ktoré, ak by sa ukázalo ako dostatočne spoľahlivé,

by mohlo situáciu podstatne zmeniť.[1]

1.2 Ciele práce a dosiahnuté výsledky

Cieľom tejto práce bolo popísať existujúce ako aj navrhnúť a zdôvodniť nové spôsoby testovania kvality "náhodnosti" postupnosti, resp. generátora pseudonáhodných čísel na základe jeho výstupnej postupnosti. To vyžadovalo podrobné zorientovanie sa v problematike. Ďalším z cieľov bolo priniesť praktický prínos v podobe konkrétnej batérie testov, alebo systému, vhodného na testovanie. Kvalita mojich návrhov mala byť preverená praktickým testovaním. Na záver bolo mojou úlohou navrhnúť ďalší postup pri štúdiu náhodných/pseudonáhodných generátorov.

Na štúdium som si vybral testovací balík NIST, a plánoval som vykonať množstvo testov na ňom a následne ich porovnávať s inými testmi voľne dostupnými, čím som chcel získať závislosti medzi testmi, aby som mohol vytvoriť lepšiu, kvalitnejšiu verziu. Súčasným efektom by bolo vytvorenie databázy vhodných generátorov. Používal som manuálne postupy ktoré viedli k veľmi pomalým a kusým výsledkom a v žiadnom prípade nemohli viesť k zdarným výsledkom. Preto som zmenil metodiku a vyvinul jednoduchý prehľadný systém, pomocou ktorého sa dalo lepšie zbierať a analyzovať údaje. Taktiež som zmenil objekt svojho záujmu po objavení úplne nového testovacieho balíka TestU01, ktorý však vyžadoval veľké úsilie na praktickú implementáciu a ktorého funkcionalitu sa mi podarilo využiť len čiastočne. Následne som navrhol postup, ako vytvoriť nový testovací balík, bez znalosti konkrétneho pozadia jednotlivých testov.

Štruktúra tohto dokumentu je následovná. Po zadefinovaní základných pojmov a motívov sa venuje podrobne najprv histórii testovania, rôznym prístupom k testovaniu a následne zoznamu najznámejších testov, so slovným popisom ich princípov. V ďalšej časti analyzujem moje postrehy ohľadom súčasného stavu na poli testovania a navrhujem nové spôsoby prístupu k

[1] Tento odsek je prevzatý zo zadania mojej diplomovej práce od môjho vedúceho.

nim. V prílohách je zverejnený popis mojich programov ako aj obrazová príloha z výsledkov testov. Súčasťou prílohy je aj dokumentácia k CD, ktoré je súčasťou tejto práce a sú na ňom zaujímavé dokumenty a aplikácie ktoré s touto témou súvisia

1.3 Využitie náhodných čísel

Náhodné čísla hrajú rozhodujúcu úlohu vo viacerých oblastiach, vrátane šifrovania, hier, simulácie, výberu, rozhodovania a estetiky. V nasledujúcich odsekoch je popísané ich využitie v konkrétnych oblastiach, avšak ani zďaleka nie je tento zoznam kompletný.

Šifrovanie Šifrovanie je proces premeny zmysluplného textu na zdanlivo náhodný chaos takým spôsobom, že len majiteľ kľúča je schopný obnoviť ho do pôvodého stavu. V dnešnej dobe sú pravidlá šifrovania čoraz zložitejšie, čomu dosť napomáha neustále narastajúca výpočtová sila počítačov. Ciele ale zostávajú tie isté, vyhnúť sa možnosti, aby protivník mohol ťažiť z našich tajných informácií. Preto je potrebné dosiahnuť postupnosti dát, ktoré sú ťažko uhádnuteľné, pokiaľ nie je známy spôsob, ako vznikajú. A práve preto, srdcom všetkých šifrovacích systémov je vytváranie tajných, neuhádnuteľných náhodných čísel.

Hry Roztáčanie rulety, hádzanie kociek, miešanie kariet, sú najznámejšie prípady, ktoré pozná úplne každý. Náhodnosť je ústredná potreba všetkých rizikových hier a ťahúň celého hráčskeho odvetvia. Táto oblasť je podobne ako šifrovanie obzvlášť náročná na kvalitu a nepredikovateľnosť generátorov.

Simulácia Simulácia je umelé vytváranie prostredia, avšak v zjednodušených podmienkach ako u pôvodného javu, zároveň je aj procesom ponúkajúci užívateľovi príležitosť na spoznanie nových oblastí. Ak sa používa počítač na simuláciu prírodnej udalosti, náhodné čísla hrajú rozhodujúcu úlohy pri vytváraní realistických podmienok. Simulácia je využiteľná v mnohých praktických oblastiach začínajúc štúdiom jadrovej fyziky,počnúc simuláciou počasia končiac u simulácií života a vývoja rôznych populácii. Čím zložitejšie simulácie sa používajú, tým viac narastá potreba väčšieho množstva náhodných čísel, ktoré sú viac citlivé na kvalitu generátora.

Vzorkovanie(sampling) Je prakticky nemožné spísať všetky možné prípady využitia, ale náhodná vzorka ponúka pochopenie zákonitosti správania. Vzorkovanie spolu s náhodnými číslami dáva každému v populácii rovnakú šancu byť vybraný, s vyhnutím sa rôznym vplyvom. Náhodné čísla sa používajú na výber v mnohých oblastiach vývoja, či už v praktickej alebo na vedeckej úrovni.

Estetika a umenie Použitie náhodných čísel v umení či grafických systémoch sa stáva čoraz populárnejším. V geometrii napr. pri fraktáloch umožňuje vytvárať prirodzený vzhľad prírodných materiálov, ktorý by bez náhody pôsobil veľmi umelo. Hudobné syntetizátory využívajú náhodnosť na vytváranie jedinečných melódií.

1.4 Typy generátorov náhodných čísel

Existujú dva základné typy generátorov náhodných čísel, pravé(fyzikálne) generátory náhodných čísel(TRNG) a pseudonáhodné generátory (PRNG).Hlavný rozdiel medzi nimi je v tom, že TRNG je zdrojom entropie, kdežto PRNG je deterministický algoritmus ktorý vytvára náhodne čísla.

1.4.1 Pravé generátory náhodných čísel

TRNG vyžaduje prírodne sa vyskytujúci zdroj náhodnosti, t.j. entropiu, na vytváranie náhodných čísel. Vzorkuje tento zdroj entropie a vytvára z neho postupnosť náhodných čísel. TRNG je fyzikálny generátor a nemal by byť automaticky braný ako dokonale náhodný. Pravé náhodné čísla sú z definície naprosto neuhádnuteľné. Zdroj entropie zvyčajne pozostáva z nejakých fyzikálnych elementov, ako napr. atmosférický šum, uplynutý čas medzi vyžiarením rádioaktívnej častice, pohyb atómov.

1.4.2 Pseudo generátory náhodných čísel

Pseudonáhodne čísla nie sú skutočne náhodné. Ich proces vytvárania nezávisí na zdroji entropie. Známe sú aj pod pojmom deterministické náhodne čísla, čo značí že sú vytvárané pomocou algoritmu. Ak je známy algoritmus a jeho inicializácia (seed), potom sú všetky čísla týmto generátorom uhádnuteľné. Hlavný cieľ pri PRNG je dosiahnutie postupnosti, ktorá sa správa, akoby bola náhodná. Niektoré výstupy viacerých PRNG sú štatisticky nerozoznateľné od výstupov TRNG a často sa vďaka svojej konštrukcii javia byť viac náhodné. Z definície PRNG vyplýva, že maximálna dĺžka postupnosti, vytvorená ľubovoľným PRNG je konečná, pretože má konečný počet vnútorných stavov, a tieto postupnosti sú reprodukovateľné.

1.4.3 Porovnanie TRNG a PRNG

Aj TRNG aj PRNG majú svoje výhody a nevýhody. Vo všeobecnosti obmedzenia jedného typu sa stávajú výhodou toho druhého. Následný zoznam obsahuje výhody a nevýhody TRNG, pre PRNG je to presne naopak.
Výhody:

1. Vysoký stupeň zabezpečenia
2. Nemožné uhádnuť ďalšie čísla na základe znalosti predchádzajúcich
3. Nie sú žiadne závislosti v rámci postupnosti
4. Nemá cyklické postupnosti

Nevýhody:

1. Sú príliš drahé
2. Pomalý a neefektívny zdroj
3. Potreba zabezpečenia proti útokom na fyzickej úrovni
4. Nie je možné tie isté postupnosti opakovane vytvárať
5. Ťažkopádny spôsob prevádzky

Kapitola 2

Štatistické testovanie

Myšlienka štatistického testovania náhodnosti generovanej postupnosti vznikla ako reakcia na množstvo kryptografických útokov, ktoré využívali závislosti v dátach, ktoré by skutočne náhodne vygenerované dáta nemali mať. Vzniklo niekoľko tzv. batérií testov, ako aj väčšie množstvo samostatných testov.

Každá postupnosť, bez ohľadu na to, ako bola vytvorená, môže byť testovaná ľubovoľným testom. Pravdepodobný výsledok štatistického testu je známy vopred a môže byť štatisticky popísaný za predpokladu, že bol aplikovaný na skutočne náhodnú postupnosť. Existuje potenciálne nekonečne veľké množstvo rôznych štatistických testov, z ktorých každý testuje nejakú špecifickú vlastnosť, pre ktorú, ak ju generátor nesplní, je označený za nie náhodný. Práve preto vnikla snaha o vytvorenie čo najmenšej množiny testov, ktorá by bola vhodná na efektívne a takmer bezchybné rozhodovanie o náhodnosti testovaných dát.

2.1 Chronologický vývoj prístupu k testovaniu náhodnosti

Za jedného zo zakladateľov modernej štatistiky možno považovať Karla Pearsona [41], ktorý už v roku 1900 vymyslel "chi-square" test a publikoval ho. Pearsonová práca je považovaná za prelomovú, pretože dovtedy sa štatis-

tické vlastnosti merali len pomocou kreslených grafov. Nasledoval ho W.G. Cochran, 1952 [11], ktorý vo svojom zhrnujúcom článku veľmi dôsledne popísal "chi-square" test a bibliografiu k nemu prislúchajúcu. Rozličnými technikami spojenými s náhodnými číslami sa ako jeden z prvých zaoberal von Neumann ,1963 [53]. Kolmogorov,1965 [27] previazal náhodnosť postupnosti s najkratším možným algoritmom, ktorý je schopný danú postupnosť vygenerovať. Vzápätí sa objavili prvé algoritmy na meranie náhodnosti. V roku 1971 Kak [25] aplikoval Walsh-Fourierove transformácie na meranie množstva náhodnosti v konečnej postupnosti. O rok neskôr prišiel Phillips,1972 [42][43] s dvoma algoritmami ktoré vypočítavali autokoreláciu a "neistotu" binárnej postupnosti. Chaitin, 1975 [10] prišiel s myšlienkou, že náhodnosť postupnosti je previazaná s najkratšou funkciou, ktorá môže vytvoriť danú postupnosť. Benett ,1976 [3] sa zaoberal jazykovými koreláciami a náhodnými procesmi. Yuen, 1977 [56] testoval náhodné čísla pomocou Walshových transformácií. S myšlienkou, že niektoré testy majú zmysel iba pre konkrétne typy generátorov prišiel Atkinson, 1980 [1]. Priekopníkom a najviac citovaným autorom na poli testovania sa stal Donald Knuth, 1981 [26], avšak v dnešnej dobe sú už ním uvádzané testy považované za nedostatočné. V roku 1983 prišiel Hopkins [18] s algoritmom na spektrálny test. Konštrukciou a testovaním PRNG sa zaoberal Marsaglia,1985 [31]. V tom istom roku prišli Beker a Piper [2] s ďalšími testmi. O dva roky neskôr publikoval Feldman [14] štatistický test založený na rýchlych Walsh-Hadamardových transformáciách(FWT). V roku 1988 prišiel Wanders [54] s analýzou Goulombových postulátov náhodnosti. Rok 1989 sa stal obzvlášť plodným. Maurer a Massey [38] publikovali teóriu o dokázateľnosti kryptografickej bezpečnosti pseudonáhodných postupností. Carroll [7] popísal binárny odvodený test. Beth a Dai [4] zverejnili teóriu, v ktorej previazali Turing-Kolmogorov-Chaitin complexity a Linear Complexity. A nakoniec Jansen [22] objavil, ako nájsť najkratší nelineárny posuvný register so spätnou väzbou, čím napomohol k možnosti merať lineárnu zložitosť. O rok neskôr zverejnil Maurer [37] svoj univerzálny test. Vzápätí prišiel L'Ecuyer ,1992 [28] s prehľadom testov. Čoskoro pris-

peli aj Marsaglia a Zaman,1993 [32] so svojimi testami. Za zmienku stojí aj práca Compagnera, 1992 [13] [12], ktorý sa vo svojich prácach podrobnejšie venoval náhodnosti. Maclaran 1993[30], sa zaoberal rozličnosťou požiadaviek pre tvorbu RNG pre kryptografické a štatistické účely. Testovaním veľkých podmnožín sa zaoberali Gustafson, Dawson a Golic, 1995[16].

Rok 1995 možno považovať na poli praktického testovania za prelomový, pretože bola po prvý krát zverejnená kompletná tzv. batéria testov, ktorá bola vytvorená Marsagliom a niesla meno [33]. Aj keď sa podľa [50]u medzi batérie testov radí Knuthova kniha "Art of Computer Programing"[26], ktorá je už zastaralá [17] , či podrobne spracovaná kniha "Handbook of Applied Cryptography" [40] od Alfreda Menezesa, bola batéria testov DIEHARD prvé kompletné softwarové riešenie. Avšak v roku 2002 vydal Marsaglia a Tang [34] novú sadu troch testov, ktorú považuje za kvalitnejšiu ako DIEHARD a ľahšie implementovateľnú. Medzi menej známe batérie testov patrí podľa [50]u aj Crypt-XS [20]. Ďalším veľkým hráčom na poli batérií testov sa stal v roku 1999 National Institute of Standards & Technology(NIST), USA, ktorý vydal komplexnú sadu 16 naprogramovaných testov[50] prednostne určených na testovanie kryptografickej bezpečnosti. Čoskoro sa stala sa hlavným rivalom DIEHARDu a pokorila ho, avšak ani v tomto prípade nešlo o dokonalé riešenie. Objavených bolo niekoľko chýb v niektorých testoch a podrobnejšie sa tomu venujú v "Corrections of the NIST Statistical Test Suite for Randomness"[49]. Najnovšie riešenie priniesol testovací balík TestU01,2002 [44], ktorý podľa slov autorov ponúka flexibilné riešenie na testovanie postupností určených na rôzne účely a nemá ani zďaleka také veľké množstvo obmedzení ako DIEHARD. Obsahuje veľké množstvo naprogramovaných generátorov a veľmi veľa testov ako aj viacero batérii, vrátane vylepšeného DIEHARDU a NISTu. TestU01 uvádza ďalšie tri softwarové riešenia, a to program ENT[55] obsahujúci niekoľko elementárnych testov , knižnicu SPRNG [36], ktorá obsahuje naimplementované všetky Knuthove testy a tiež niektoré ďalšie a knižnicu gsl [51], ktorá síce neobsahuje testy, ale veľké množstvo skvele otestovaných generátorov.

2.2 Základné podklady pre testovanie náhodnosti

Tento odsek vznikol na základe dokumentov NIST[50].Štatistický test testuje platnosť nulovej hypotézy H_0. V rámci tohto dokumentu nulová hypotéza H_0 je, že postupnosť, ktorá je testovaná, je skutočne náhodná. Pridruženou hypotézou k H_0 je alternatívna hypotéza H_a, ktorá pre tento dokument znie, že postupnosť nie je skutočne náhodná. Pre každý jeden vykonaný test, je vykonané rozhodnutie o výsledku či o ukončení testovania, na základe toho, či akceptuje alebo odmieta nulovú hypotézu H_0, teda či generátor vytvára alebo nevytvára náhodne čísla. Jednotlivé testy majú rôzne matematické pozadie a preto každý z nich potrebuje špecifickú adekvátnu "štatistiku" na rozhodovanie o akceptácii či odmietnutí H_0. Na základe predpokladu náhodnosti takáto štatistika má svoju distribúciu a teda aj distribučnú funkciu. Teoretická, referenčná distribúcia pre túto štatistiku pri H_0 je určená konkrétnym matematickým pozadím. Od tejto referenčnej distribúcie je odvodená tzv. kritická hodnota, zvyčajne veľmi vzdialená od teoretickej hodnoty. Počas testovania pre každé testované dáta je vypočítaná štatistická hodnota, ktorá sa potom porovná s kritickou hodnotou. Ak táto hodnota prekročí kritickú hodnotu, tak je H_0 odmietnutá, v opačnom prípade je akceptovaná.

Dôvod, prečo štatistické testovanie vôbec funguje je, že referenčná distribúcia a kritická hodnota sú závislé navzájom a generované na základe pokusného predpokladu náhodnosti. Ak je pre testované dáta predpoklad náhodnosti skutočne pravdivý, potom vypočítaná štatistická hodnota má veľmi malú pravdepodobnosť prekročenia kritickej hodnoty. Na druhej strane, ak vypočítaná štatistická hodnota prekročí kritickú hodnotu, t.j. nastane malo pravdepodobný jav, potom aj pre celý princíp štatistického testovania nastane malo pravdepodobná udalosť. V takomto prípade prídeme k záveru, že pôvodný predpoklad náhodnosti bol chybný, takže odmietneme H_0 a akceptujeme H_a, t.j. odmietneme hypotézu o náhodnosti testovaných dát.

Pri štatistickom testovaní hypotéz môžu nastať dva prípady, môže sa

dospieť k akceptovaniu H_0 alebo H_a. Keďže celý test vychádza z hypotéz, môže sa stať, že dáta sú v skutočnosti náhodne, ale test skončí odmietnutím H_0. Vtedy nastáva chyba Typu I. Podobne nastáva chyba Typu II, ak v skutočnosti nie sú dáta náhodné, ale test ich vyhodnotí ako náhodné. Pravdepodobnosť výskytu chyby Typu I považujeme za mieru významu (level of significance) testu. Táto hodnota sa dá určiť vopred, značíme ju α. Čiže α určuje pravdepodobnosť, že test nevyhodnotí postupnosť ako náhodnú, aj keď v skutočnosti náhodná je, t.j. postupnosť má vlastnosť, ktorá porušuje náhodnosť napriek tomu, že bola vygenerovaná kvalitným generátorom. Pre zmysluplné testovanie je vhodné nastaviť α aspoň na $\alpha = 0.01$, v praxi sa používa oveľa menšia hodnota.

Pravdepodobnosť výskytu chyby Typu II sa značí β. Je to vlastne pravdepodobnosť, že test vyhodnotil postupnosť ako náhodnú, aj keď bola vygenerovaná nekvalitným generátorom. Keďže existuje nekonečne veľa možností štatistického testovania toku dát, existuje teda aj nekonečne veľa spôsobov detekcie porušenia náhodnosti a teda každému testu prislúcha odlišné β. Týmto sa stáva výpočet β oveľa ťažším ako výpočet α. Jedným z hlavných cieľov jednotlivých testov je minimalizovať pravdepodobnosť výskytu chyby Typu II. Pravdepodobnosti α , β a dĺžka n testovanej postupnosti sú previazané navzájom tak, že zo znalosti dvoch z nich je možné vypočítať tretiu hodnotu. V praxi sa zvykne najprv vyberať veľkosť vzorky n a hodnota α, potom sa vyberá kritická hodnota, ktorá stlačí β k najmenšej možnej hodnote.

Princíp každého z testov je založený na výpočte štatistickej hodnoty ako funkcie vstupných dát. Ak túto štatistickú hodnotu označíme S a kritickú hodnotu t, potom pravdepodobnosť výskytu
chyby Typu I je $P(S > t|H_0 jetrue) = P(odmietni H_0|H_0 jetrue)$ a
chyby Typu II je $P(S \leq t|H_0 jefalse) = P(akceptuj H_0|H_0 jefalse)$.

Napokon sa pre každý test vypočíta tzv. $P - hodnotu$ ktorá popisuje silu dôkazu proti H_0. $P-hodnota$ určuje pre každý test jeho pravdepodobnosť, že dokonalý generátor náhodných čísel vyprodukuje postupnosť menej náhodnú

ako je testovaná postupnosť. Ak sa $P-hodnota$ blíži k 1, potom sa testovaná postupnosť javí byť dokonale náhodnou. Naopak, ak sa blíži k 0, tak zrejme nie je náhodná. Prakticky sa v testoch volí miera významu α, ktorá sa následne porovnáva s $P-hodnotou$. Ak $P-hodnota \geq \alpha$, potom sa akceptuje H_0, v opačnom prípade $P-hodnota < \alpha$, H_0 je odmietnutá a teda testovaná vzorka vyhodnotená ako nenáhodná. Ak sa napr. zvolí $\alpha = 0.001$, tak treba očakávať, že bude odmietnutá jedna vzorka z tisíca za predpokladu, že bola skutočne náhodná. Napríklad, pre $P - hodnotou \geq 0.001$ bude postupnosť s pravdepodobnosťou 99.9% náhodná, pre $P - hodnotou < 0.001$ nebude s 99.9% pravdepodobnosťou náhodná.

2.3 Interpretácia výsledkov testovania

Rozlišujú sa tri typické empirické výsledky, ku ktorým sa dá analýzou $P-hodnoty$ dospieť.

- neukazuje na náhodnosť testovaných dát
- ukazuje na náhodnosť testovaných dát
- z analýzy sa nedá rozhodnúť o náhodnosti.

Interpretácia empirických výsledkov môže byť vedená rôznymi spôsobmi. [50] uvádza dva prístupy ako k výsledkom pristupovať. Vyšetrenie pomeru počtu postupností, ktoré vyhoveli testu a uniformnosť distribúcie $P-hodnoty$. V prípade, ak obidva prístupy zlyhajú (t.j. H_0 musí byť odmietnutá), musia byť vykonané dodatočné experimenty na iných vzorkách toho istého generátora, aby sa dalo určiť, či sa jednalo o štatistickú anomáliu alebo o zrejmý dôkaz náhodnosti.

2.3.1 Pomer postupností vyhovujúcich testu

Na základe empirických výsledkov pre daný test sa vypočíta pomer postupností vyhovujúcich testu. Napríklad ak bolo testovaných 1000 postupností $m = 1000$, $\alpha = 0.01$ a 996 postupností malo $P - hodnotu > 0.01$, potom pomer je $996/1000 = 0.996$.

Rozsah akceptovateľných pomerov je daný dôveryhodným intervalom,
$p^{'} \pm 3\sqrt{(\frac{(p^{'}(1-p^{'})}{m})}$,

kde $p^{'} = 1 - \alpha$ a m je počet testovaných postupností. Ak tento pomer padne mimo intervalu, je to dôkazom, že dáta nie sú náhodné. Interval dôvery bol vypočítaný pomocou normálnej distribúcie ako aproximácia binomickej distribúcie, čo dáva dostatočne presné výsledky pre veľké $m \geq 1000$.

2.3.2 Uniformná distribúcia P-hodnoty

U distribúcie $P - hodnoty$ je potrebné skúmať jej uniformnosť. Vizuálne sa dá použiť ako pomôcka histogram, kde sa rozdelí interval medzi 0 a 1 na 10 rovnomerne veľkých disjunktných intervalov, kde sa pre každý z nich spočíta počet $P-hodnot$, ktoré v nich ležia. Počty by v jednotlivých intervaloch mali byť rovnaké. Uniformita sa dá zistiť aj aplikáciou "chi-square" testu (χ^2) a určením $P - hodnoty$ korešpondujúcej s "Goodness-of-Fit Distributional" testom získaným pre ľubovoľný štatistický test. Dosiahne sa to vypočítaním
$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{s}{10})^2}{(\frac{s}{10})}$,

kde F_i je počet $P - hodnot$ v intervale i a s je veľkosť testovanej vzorky. Potom sa $P - hodnota$ vypočíta
$P - hodnota_T = igamc(\frac{9}{2}, \frac{\chi^2}{2})$
Ak $P - hodnota_T \geq 0.0001$, potom možno považovať testované postupnosti za uniformne distribuované.

2.4 Univerzálne kódy

Odkedy Claude Shannon publikoval jeho známu prácu "A mathematical theory od comunication" [8], začali hrať myšlienky a výsledky teórie informácií (Information Theory) dôležitú úlohu v šifrovaní [52][9], matematickej štatistike[19] a veľa iných oblastiach[47], ktoré sú od telekomunikácii veľmi vzdialené. Teória univerzálnych kódov bola od jej vzniku efektívne aplikovaná v mnohých oblastiach. Preto iniciovala aplikácia získaných výsledkov[6] v oblasti univerzálnych kódov k novému prístupu k náhodnosti[23].

V tejto kapitole bude rozobraný jeden z najnovších prístupov testovania hypotéz, ktorý je založený na myšlienkach vychádzajúcich z univerzálnych kódov. Tento prístup previazal štatistické testovanie s teóriou kompresie textu a urobil z nich nástroj testovania náhodnosti. V dnešnej dobe sú moderné nástroje určené na kompresiu založené na hlbokých teoretických základoch a preukázali veľmi efektívne výsledky pri kompresii textu, DNA kódu a pri veľa iných typoch reálnych dát. Univerzálne kódy a komprimačné nástroje môžu nájsť skryté podobnosti rôznych typov, čo ich automaticky predurčuje ako vhodný nástroj na testovanie.

Nech je ϑ univerzálny kód. A nech $\vartheta(x_1, \ldots, x_t)$ je zakódovaná postupnosť. $l_\vartheta(x_1, \ldots, x_t)$ je dĺžka zakódovaného slova a α je požadovaná miera významu. Intuitívne možno predpokladať, že postupnosť nie je možné skomprimovať ak platí H_0. A naopak, ak ju je možné skomprimovať, tak H_0 neplatí. Prislúchajúci formálny test znie: Ak $(t - l_\vartheta(x_1, \ldots, x_t) > \log(\frac{1}{\alpha}))$ potom musí byť H_0 zamietnutá (Pre účely tejto kapitoly platí $log \equiv log_2$). Je dokázané [5], že chyba typu I tohto testu je menšia alebo rovná ako α pre ľubovoľný unikátne dekódovateľný kód ϑ, vzhľadom k tomu, že chyba typu II ide k nule pre kód ϑ, keď dĺžka kódu t rastie.

Ak je H_0 pravdivá tak, priemerná dĺžka ľubovoľného kódu nie je menšia ako dĺžka kódu t. Preto, ak sa zadefinuje dĺžka najlepšieho kódu ako $l_{H_0}(x_1, \ldots, x_t)$, tak musí platiť $l_{H_0}(x_1, \ldots, x_t) = t$. Potom schéma navrhnutého testu môže znieť: Ak $l_{H_0}(x_1, \ldots, x_t) - l_\vartheta(x_1, \ldots, x_t) \leq \log(\frac{1}{\alpha})$, potom H_0 platí, inak platí H_1 . Táto schéma sa používa vo všetkých známych šta-

tistických problémoch. Niekedy sa nahrádza $l_{H_0}(x_1, \ldots, x_t)$ nižšou hraničnou hodnotou určenou na základe tzv. Shannonovej empirickej entropie.

2.5 Kritéria na určovanie kvality generátorov náhodných čísel

Všetky aplikácie, pri ktorých sa využívajú generátory náhodných čísel vyžadujú, aby na výstupe dávali postupnosť nezávislých uniformných náhodných premenných, zvyčajne v intervale $(0, 1)$, prípadne ako binárne dáta v podobe množiny $\{0, 1\}$. V prvom prípade za testovanú hypotézu H_{0i} považujeme, že ľubovoľná postupnosť hodnôt generátora je postupnosťou nezávislých náhodných premenných s uniformnou distribúciou z intervalu $(0, 1)$. V druhom prípade za hypotézu H_{0b} považujeme, že generátor na výstupe vracia postupnosť čísel 0 a 1 s rovnakou pravdepodobnosťou nezávisle na predchádzajúcich výstupoch. Oba prístupy k testovaniu sú navzájom previazané, pretože napr. postupnosť bitov sa dá previezť na čísla z intervalu $(0, 1)$ a naopak.

V prípade tvrdenia o intervale je hypotéza H_{0i} zhodná s tvrdením, že pre $t > 0$, vektor $\vec{u}(u_0, \ldots, u_t)$ je uniformne distribuovaný v t-rozmernej kocke $(0, 1)^t$. Avšak toto tvrdenie nemôže byť pravdivé pre PRNG, pretože tieto vektory získavajú hodnoty z konečnej množiny S_t všetkých t-rozmerných vektorov pozostávajúcich z hodnôt, ktoré môžu byť vytvorené generátorom zo všetkých jeho možných inicializácií alebo zrniek(seeds). Mohutnosť takto získanej množiny nemôže prekročiť počet prípustných inicializácií. Za predpokladu, že zrniečko je vybrané náhodne (pomocou TRNG), potom sú vektory v S_t generované náhodne pričom aproximujú uniformnú distribúciu. Z toho vyplýva, že S_t bude veľmi riedko obsiahnuté v t-rozmernej kocke $(0, 1)^t$. Teoretické znalosti na meranie tejto distribúcie sú základným predpokladom pre dizajn dobrých generátorov. Najľahšie sa vypočítavajú kritéria na meranie distribúcie pre lineárne generátory. Toto je hlavný dôvod pre tak veľkú popularitu lineárnych generátorov ako sú linárne kongruentné generátory, multi rekurzivné generátory, lineárne posuvné registrové generátory so spätnou väzbou, či všeobecné posuvné registrové generátory so spätnou väzbou.

2.6 Praktický prístup k testovaniu náhodných čísel

Štandartné knihy o štatistickom testovaní uvádzajú[50][44], že pri testovaní hypotézy by mala byť vopred daná miera významu α, napr. $\alpha = 0.01$, a hypotéza H_0 by mala byť odmietnutá práve vtedy ak $p - hodnota < \alpha$. Avšak podľa [44] tento postup je vhodný iba v prípade, ak test prebieha na malej, pevne zvolenej vzorke, ale nepovažujú to za najlepší postup v prípade testovania náhodných čísel. V skutočnosti pri testovaní náhodných čísel je testovaná vzorka obrovská a zvyčajne môže byť zvyšovaná jej dĺžka podľa potreby. Preto odporúčajú namiesto určovania čísla α, a následného vyhodnotenia testu na základe $p-hodnoty$, aby sa ako výstup testu použila samotná $p - hodnota$, definovaná ako

$p = P[Y \geq y|H_0]$

pričom y je hodnota získaná štatistickým testom Y. V prípade že Y má **spojitú distribúciu**, tak p je náhodná premenná z intervalu $(0, 1)$ podľa hypotézy H_0. Pre isté testy môže byť toto p využité na meranie uniformity, v zmysle takom, že p bude nadobúdať hodnoty blízke 1 ak generátor produkuje čísla nadmerne uniformné a naopak, blízke 0 v opačnom prípade.

Ak $p - hodnota$ je extrémne malá, napr. 10^{-10}, potom je očividné, že generátor neprešiel testom. Naopak, ak nie je $p - hodnota$ v blízkosti 0 alebo 1, tak nebola zistená žiadna anomália, a generátor testom prešiel. V prípade že nie je možné jednoznačne určiť, či bol test úspešný alebo nie,(napr. $p - hodnota = 0.003$), potom môže byť test aplikovaný odznova, na nových disjunktných vzorkách generátora, pokiaľ nebude potvrdené, že generátor neprešiel testom, alebo vyvrátené podozrenie s následným vyhodnotením úspešnosti generátora. Tento postup je možný v prípade, že nie je testovaná vzorka obmedzená veľkosťou, t.j. generátor môže neobmedzene generovať náhodné dáta.

Ak sa použije viacero testov na daný generátor a generátor je objektívne

odolný voči týmto testom, (t.j. vytvára v rámci ich kritérií náhodne dáta) , tak získané $p - hodnoty$ nadobúdajú hodnotu menšiu ako 0.01 alebo väčšiu ako 0.99 náhodne s pravdepodobnosťou približne 2% všetkých testov. V tomto prípade by teda $p - hodnoty$ nemali nadobúdať extrémne hodnoty systematicky, ale veľmi sporadicky. Preto by mal v drvivej väčšine prípadov nastať neúspech generátora pri teste na základe štruktúry množiny S_t a len zriedkavo na základe toho, ktorá časť výstupu generátora bola testovaná. Naviac, ak generátor začne byť neúspešný jednoznačným spôsobom, $p - hodnota$ zvyčajne začne konvergovať k 0 alebo 1 exponenciálne rýchlo ako funkcia veľkosti testovanej vzorky. Preto podozrivé $p - hodnoty$ môžu byť ľahko rozhodnuté zvýšením testovanej vzorky.

V prípade, že Y má **diskrétnu distribúciu**, je potrebná istá dávka opatrnosti pri definícii $p-hodnoty$. V tomto prípade sa rozlišujú dva prípady, pravá $p - hodnota$ $p_r = P[Y \geq y|H_0]$ a ľavá p-hodnota $p_l = P[Y \leq y|H_0]$. Hypotéza H_0 je odmietnutá v prípade ak je jedna z týchto hodnôt príliš blízko k 0. Ak by sa použil rovnaký postup ako v prípade spojitej distribúcie, kde $p_l = 1 - p_r$, bolo by možné odmietnuť H_0 na základe toho, že $p - hodnota$ je príliš blízko 1. Avšak, $P[Y = 0|H_0] = \frac{1}{e} \approx 0,368$, takže je nezmyselné odmietnuť H_0 v tomto prípade. V skutočnosti, ľavá $p - hodnota$ je $p_l = 0,368$, takže ani p_r, ani p_l nie sú blízke k 0. Je potrebné mať na pamäti, že nie je možné definovať $p_l = 1-p_r = P[Y < y|H_0]$, v tomto prípade by vyšlo $p_l = 0$.

2.7 Dvojúrovňové testovanie

Vo viacerých publikáciách [50][45] sa uvádza dvojúrovňový test, na testovanie generátorov. Myšlienka spočíva vo vytvorení N nezávislých kópií Y, povedzme $Y_1, \ldots, Y_N$, aplikovaním prvoúrovňového testu N krát na disjunktné podreťazce vytvorené RNG. Nech F je teoretická distribučná funkcia pre Y podľa H_0. Ak F je spojitá, získané $U_1 = F(Y_1), \ldots, U_N = F(Y_N)$ sú náhodné premenné v intervale $(0, 1)$ podľa H_0. Jedným spôsobom, ako vykonať dvojúrovňový test, je porovnať pomocou GOF (goodness of fit) testu získanú distribúciu $U_1, \ldots, U_N$ spolu s uniformnou distribúciou. Ako GOF test je možné použiť buď **Kolmogorov-Smirnovov test**, **Andersonov-Darlingov** či **Crámerov- von Misesov** test, atd.

Pri vyhodnocovaní GOF testu sa vypočíta jeho $p - hodnota$, ktorá sa následne porovná s očakávanými hodnotami. Ak sú rozdiely príliš veľké, tak je H_0 odmietnutá. Pre dosiahnutie vyššej istoty, je možné použiť viacero typov GOF naraz, pre ľubovoľný dvojúrovňový test. Táto flexibilita testovania môže byť využitá pri testovaní účinnosti GOF testov na detekciu slabín špecifických tried generátorov.

Jedným z možných vylepšení výkonnosti GOF testov, pomocou ktorého je niekedy možné lepšie detekovať slabiny generátora, je pretransformovanie ich vstupu. Pretransformovaním vstupu je možné získať hodnoty, na základe ktorých je test lepšie schopný rozpoznať isté typy anomálii. Vynikajúcim príkladom takejto transformácie je priestorová transformácia[46]. Ďalšiou možnou transformáciou je podielovo-mocninná transformácia.

Existujú dva hlavné dôvody, pre ktoré má zmysel používať dvojúrovňové testovanie.

- V prípade, že prvoúrovňový test nie je schopný pracovať s väčšími vstupnými vzorkami. napr. kvôli nedostatku pamäti
- V prípade, že prvoúrovňový test nie je schopný zmerať nevhodné vlastnosti lokálne, t.j. keď pre vstupnú vzorku vyhodnotí globálnu vlastnosť,

ale podreťazec vstupnej vzorky so zlou vlastnosťou nezistí, pretože v globálnom meradle sa táto vlastnosť stratí. Ako príklad chybného generátora s dobrou globálnou vlastnosťou, ale zlou lokálnou, je možné uviesť generátor, ktorý na výstupe dáva čísla $\frac{i}{2^n}$, pre $i = 1, \ldots, 2n$, pre pevné n. Jednoduchý test uniformnosti pre celú postupnosť nenájde žiadny problém, avšak opakovaný test pre disjunktné podreťazce ho ľahko odhalí.

Za predpokladu, že motivácia dvojúrovňového testovania vznikla na podnet iba prvého z oboch dôvodov tu uvedených, existuje jednoduchší spôsob ako tento test riešiť. Stačí vytvoriť N prvoúrovňových testov (napr. pokryť nimi postupne celý reťazec), sčítať ich, a následne odmietnuť hypotézu H_0, ak ich súčet je príliš veľký alebo príliš malý. Pre drvivú väčšinu testov distribučná hodnota výslednej sumy je N krát distribučná hodnota pôvodného testu.(platí pre chi-square, normálnu a Poissonovu distribúciu)

[44] uvádza, že pri testovaní vzorky s pevnou dĺžkou $N.n$ je test často viac efektívnejší pre $N = 1$ ako pre $N > 1$. To znamená, že pre bežné RNG, ak je nájdená nejaká štruktúra pre dostatočné dlhú podpostupnosť, tak je prakticky nájditeľná pre všetky podpostupnosti rovnakej dĺžky. Inými slovami povedané, ak je PRNG jednoznačne odmietnutý istým testom pri nejakej inicializácii(pre nejaké seed), tak je veľmi často neúspešný pre väčšinu prípustných vstupných inicializácií. V prípade, že $N > 1$ vykazuje sumačná metóda popísaná vyššie lepšie výsledky ako druhoúrovňové testovanie pomocou GOF testu.

2.7.1 Kolmogorovov-Smirnovov test

Nech $U_{(1)}, \ldots, U_{(N)}$ sú získané náhodné premenné, utriedené vzostupne. Potom

$D^+{}_N = \max\left(\frac{j}{N} - U_{(j)}\right)$, pre $1 \leq j \leq N$

$D^-{}_N = \max\left(U_{(j)} - \frac{j-1}{N}\right)$, pre $1 \leq j \leq N$

$D_N = \max\left(D^+{}_N, D^-{}_N\right)$.

2.7.2 Andersonov-Darlingov test

$A^2{}_N = -N - \frac{1}{N}\sum_{j=1}^{N}\left[(2j-1)\ln(U_{(j)}) + (2N+1-2j)\ln(1-U_{(j)})\right]$

2.7.3 Crámerov-von Misesov test

$W^2{}_N = \frac{1}{12N} + \sum_{j=1}^{N}(U_{(j)} - \frac{(j-0.5)}{N})^2$

2.7.4 Priestorová transformácia

Na vstupe je vzostupne utriedená postupnosť náhodných premenných $U_{(1)}, \ldots, U_{(N)}$. Priestorová transformácia počíta vzdialenosti medzi týmito premennými. t.j.

$S_i = U_{(i+1)} - U_{(i)}$, pre $0 \leq i \leq N$, kde
$U_{(0)} = 0$,
$U_{(N+1)} = 1$.
Následne utriedi postupnosť $S_0, \ldots, S_N$ tak, že vznikne $S_{(0)} \leq S_{(1)} \leq \cdots \leq S_{(N)}$.

Potom sa vypočíta S'_i:
$S'_0 = (N+1)S_{(0)}$
$S'_i = (N-i-1)(S_{(i)} - S_{(i-1)})$.
Následne sa získa konečný výstup transformácie
$V_i = S'_0 + S'_1 + \cdots + S'_{i-1}$, pre $i = 1, \ldots, N$.

Výstupné hodnoty $V_1, \ldots, V_N$ sú distribuované ako N nezávislých premenných z intervalu $(0, 1)$, ktoré sú usporiadané vzostupne. Pomocou tejto transformácie je možné rozpoznávať prípadné zhlukovanie $U_{(i)}$. Ak je viacero $U_{(i)}$ príliš blízko seba, potom bude niekoľko V_i blízke k 0, čo vedie k tomu, že napr. Andersonov-Darlingov test ľahko túto anomáliu rozpozná, avšak pri pôvodnom vstupe by mohol dospieť k pozitívnemu výsledku.

2.7.5 Podielovo-mocninná transformácia

Podobne ako predchádzajúca transformácia, aj táto slúži na detekciu zhlukovania. Z pôvodných hodnôt $U_{(i)}$ sa vypočíta
$U'_{(i)} = (\frac{U_{(i)}}{U_{(i+1)}})^{i}$, pre $i = 1, \ldots, N$.

Na výstup sa posielajú vzostupne usporiadané hodnoty $U'_{(i)}$. Obe tu uvedené transformácie je možné ľubovoľne kombinovať s GOF testami.

Kapitola 3

Výber najznámejších testov

V tejto kapitole je uverejnený prehľad najznámejších testov, uverejnených hlavne Knuthom [26],NISTom [50] a v TestU01 [45]. Prvoúrovňové testovanie je možné rozlíšiť na dve základné kategórie, testovanie postupnosti reálnych čísel z intervalu $(0, 1)$ a testovanie postupnosti bitov. V zásade je jedno, či test testuje reálne čísla alebo bity, keďže je možné jeden typ pretransformovať na druhý a zase naopak.

3.1 Testy postupnosti reálnych čísel

3.1.1 Testy merajúce celú postupnosť naraz

Meranie celkovej uniformity

Okrem už spomínaného spôsobu, pri ktorom sa vypočíta distribúcia postupnosti $u_1, \ldots, u_n$ a porovná sa s očakávanou distribúciou pomocou GOF testov, existuje ešte jednoduchší spôsob merania uniformity. Stačí vypočítať strednú hodnotu testovanej postupnosti a jej disperziu a následne ich porovnať s teoretickými hodnotami $\frac{1}{2}$ a $\frac{1}{12}$. Tento test však meria iba globálny výsledok a nijako v ňom nie sú zohľadnené lokálne anomálie. Riešením môže byť nasekanie vstupného intervalu na malé časti a opakované použitie testu pre každú z nich, s následným dvojúrovňovým testovaním.

Meranie zhlukovania

Vstupná postupnosť čísel sa utriedi v zostupnom poradí a následne sa vypočítajú prekrývajúce intervaly dĺžky m, t.j.

$g_{m,i} = u_{(i+m)} - u_{(i)}$, pričom
$u_{(1)}, \ldots, u_{(n)}$ je utriedená postupnosť, pričom
$u_{(0)} = 0$
$u_{(n+1)} = 1 + u_{(i-1)}$, pre $i > 0$, potom test bude tvaru $H^{(c)}{}_{m,n} = \sum_{i=1}^{n} h(n.g_{m,i})$, kde h je hladká funkcia, ako napr. $h(x) = x^2$ alebo $h(x) = \log(x)$. Princíp tohto m-priestorového testu spočíva v tom, že testuje získanú distribučnú funkciu, či sa zhlukuje viac ako by mala. Napr. ak sa čísla zhlukujú do skupiniek po 3 až 5, takže sú príliš blízko seba, sú pomocou klasického KS testu nezistiteľné, avšak 3-priestorový test ich odhalí.

Gap test

Tento test ako prvý uviedol Knuth [26]. Podrobnejšie skúma lokálnu štruktúru postupnosti, a snaží sa odhaliť závislosti. Test počíta počet za sebou idúcich prvkov podpostupnosti, ktoré nepatria do vopred zvoleného intervalu, ale začiatočný a koncový člen tejto podpostupnosti tam patria. Čiže ak a,b sú dve reálne čísla, pre ktoré platí $0 < a < b < 1$, potom sa berie v úvahu dĺžka za sebou idúcej podpostupnosti $U_j, U_{j+1}, \ldots, U_{j+r}$, kde U_{j+r} leží medzi a a b, ale ostatné neležia. Táto podpostupnosť $r + 1$ čísel reprezentuje "gap" dĺžky r. Následne sa vypočíta počet všetkých "gaps" pre každú možnú dĺžku, t.j. nech X_i je počet "gaps" dĺžky i. Potom sa porovnajú získané hodnoty $X_1, X_2, \ldots$ s teoretickými hodnotami očakávanými v rámci H_0 pomocou chi-square testu. Častou úpravou pred vykonaním chi-square testu je spojenie všetkých "gaps" s príliš malou očakávanou pravdepodobnosťou výskytu do jednej skupiny, t.j. určí sa nejaké k, pre ktoré $X'_k = X_k + X_{k+1} + \ldots$ a následne sa X'_k použije namiesto X_k. Tento test je schopný merať zhlukovanie, keďže pri zhlukoch sa buď príliš dlho nevráti hodnota do intervalu (a, b) alebo naopak, je tam príliš často.

Run test

Podobne ako gap test, zisťuje závislosti na lokálnej úrovni, avšak v tomto prípade sa gap nahradí dĺžkou podpostupnosti, pre ktorú za sebou idúce čísla sú iba rastúce (alebo iba klesajúce), následne vypočíta počty ich dĺžok pre celú postupnosť a aplikuje chi-quare test, kde ich porovná s očakávanými hodnotami.

3.1.2 Testy merajúce viacero podpostupností rovnakej dĺžky

Rozdelenie hyperkocky a počítanie zásahov

Vzhľadom nato, že pre RNG ktorý produkuje reálne čísla z intervalu $(0, 1)$ je hypotéza H_0 ekvivalentná tvrdeniu, že pre každé $n \geq 0$ a $t > 0$, vektor výstupných hodnôt $(u_n, \ldots, u_{n+t-1})$ je uniformne distribuovaný v t-rozmernej jednotkovej hyperkocke $(0, 1)^t$. Prirodzený prístup k testovaniu uniformity je následovný. Jednotková hyperkocka sa rozdelí na k častí s objemami $p_0, \ldots, p_{k-1}$. Štandartne sa volí $p_0 = \cdots = p_{k-1} = \frac{1}{k}$, ale nemusí to tak byť vždy. Následne sa zoberie $n.t$ čísel zo vstupu, ktoré poslúžia na vytvorenie n bodov v t-rozmernej hyperkocke, t.j.
$u_i = (uti, \ldots, uti + t - 1)$, pre $i = 0, \ldots, n - 1$

Týchto n bodov poslúži nato, aby sa vypočítal počet bodov spadajúcich do každého z objemov. Čiže sa vypočíta $X_0, \ldots, X_{k-1}$, kde X_j je počet bodov spadajúcich do objemu j. V rámci H_0 má vektor $(X_0, \ldots, X_{k-1})$ polynomiálnu distribučnú funkciu s parametrami $(n, p_0, \ldots, p_{k-1})$. Akákoľvek miera nezrovnalosti medzi číslami X_j a očakávanými hodnotami $n.p_j$ môže byť predmetom testovania. Ďalej sú uvedené špecifické inštancie a variácie tohto postupu.

Najjednoduchší spôsob, ako rozdeliť hyperkocku $(0, 1)^t$ na $k = d^t$ podkociek s objemom $\frac{1}{k}$, je rozdeliť interval $(0, 1)$ na d rovnako dlhých úsekov, pre nejaké zvolené $d > 1$. Potom, $p_j = \frac{1}{k}$ pre všetky j. Testy, ktoré merajú

celkovú odchýlku medzi získanými X_j a očakávanými $\frac{n}{k}$ hodnotami, sa vo všeobecnosti nazývajú sériové testy uniformity [26]. Štandartne sa na meranie odchýlky od očakávanej distribúcie používa chi-square test

$\chi^2 = \sum_{j=0}^{k-1} \frac{(X_j - \frac{n}{k})^2}{(\frac{n}{k})}$,

kde stupeň nezávislosti je $k-1$, za predpokladu, že $\frac{n}{k}$ je dostatočne veľké. Pre štandartné sériové testy sa odporúča aby $\frac{n}{k} \geq 5$, aby bola zachovaná ich korektnosť. To však obmedzuje výšku k zhora. Našťastie toto obmedzenie nie je skutočne nutné. Riešenie ponúka zavedenie **riedkeho** a **hustého sériového testu**, kde riedky test je definovaný pre $\frac{n}{k} \ll 1$ a hustý pre $\frac{n}{k} \gg 1$, pričom riedky test vykazuje oveľa lepšie výsledky ako hustý. Podrobnejšie informácie o tomto prístupe, ako aj ďalšie všeobecnejšie metódy merania odchýliek využívajúce napr. mocninné divergencie možno nájsť v [44].

Testy s prekrývaním (overlapping)

Pri prekrývanej verzii testov, sa definícia vytváraného bodu v kocke upravuje. Namiesto pôvodnej verzie, kde sa z výstupu generátora brali čísla postupne a používali sa iba raz ako súradnica jednej dimenzie jedného bodu hyperkocky, v prekrývanom prípade sa čísla z výstupu generátora použijú viac krát. Konkrétne každý bod kocky je definovaný ako
$u_i = (u_i, \ldots, u_{i+t-1})$, pre $i = 0, \ldots, n-1$.

Jednou z výhod tejto verzie je, že poskytuje viacej bodov, a táto vlastnosť sama o sebe zvyčajne stačí na zvýšenie kvality testu. Prekrývané verzia sériových testov sú známe tiež aj ako opičie(monkey) testy. Tento názov vznikol ako predstava opice pišúcej na písacom stroji, ktorý má d znakovú abecedu. Nevýhodou prekrývanej verzie testu je, že analýza získaných výsledkov je oveľa zložitejšia, lebo po sebe idúce body kocky nie sú už naďalej nezávislé a vytvárajú Markovovský reťazec. V hustej verzii testu už nie je vždy možné použiť chi-square metódu. V takom prípade je potrebné použiť metódu navrhnutú v [44] . V prípade riedkeho testu nebol doteraz nájdený žiadny použiteľný postup.

Iné spôsoby delenia jednotkovej hyperkocky

Okrem delenia jednotkovej hyperkocky na d^t rovnakých častí existujú aj iné spôsoby ako k hyperkocke pristupovať. Jedným zo spôsobov je hľadanie permutácie súradníc vektora u_i, ktorá ich usporiada vzostupne. Počet všetkých možných permutácií je $k = t!$. Nech X_j je počet vektorov u_i, ktorých súradnice sú vzostupne usporiadateľné pomocou permutácie j, kde $j = 0, \ldots, k$. Potom má vektor $(X_0, \ldots, X_{k-1})$ polynomiálnu distribučnú funkciu s parametrami $(n, \frac{1}{k}, \ldots, \frac{1}{k})$ a teda rovnakú ako v predchádzajúcom prípade. To znamená, že je možné použiť všetky postupy, ktoré sa aplikovali predtým. Test kolízií(kolízny test) je vhodný na testovanie generátorov, ktoré sa používajú na miešanie kariet, pretože test je schopný zistiť permutácie, ktoré sa vyskytujú častejšie ako ostatné. [26] odporúča použitie chi-square metódy na spracovanie získaných údajov, avšak v [44] sa uvádza, že tento postup vedie k nepresným výsledkom a odporúča zvýšiť hodnotu t.

Argmax test

Ako už z názvu testu vyplýva, test skúma maximum svojich súradníc a definuje X_j ako počet vektorov u_i, ktorých najväčšia súradnica je j-ta v poradí. Tento test vyhodnocuje, či je maximum rovnomerne rozdelené medzi súradnicami.

Sumačné, súčinné a iné testy

Sčítaním, vynásobením či aplikovaním iných funkcií na súradnicu vektora je možné vytvárať nové triedy polynomiálnych testov. Stačí rozdeliť množinu všetkých možných výsledkov na k rovnakých častí, a potom porovnávať očakávané výsledky s nameranými. Pre malé t a veľké n, sú tieto testy vhodné na zisťovanie zhlukovania príliš malých alebo príliš veľkých hodnôt.

Poker test

Poker test pristupuje k rozdeleniu hyperkocky odlišným spôsobom. Vygeneruje t celých čísel z množiny $\{0, \ldots, d-1\}$, následne zistí, koľko rôznych čísel takto vzniklo. Tento postup opakuje n krát a počíta frekvencie výskytov každého z rôznych čísel. Pre $t \leq d$, je tento postup ekvivalentný s rozdelením jednotkovej hyperkocky $(0,1)^t$ na d^t podkociek, s následným preusporiadaním podkociek do t tried. Postup preusporiadania do tried je nasledujúci : Vezmi roh podkocky, ktorý je najbližšie k testovanému bodu a vlož túto podkocku do triedy j, kde j je počet nezávislých súradnic. Týchto t tried tvorí konečné rozdelenie jednotkovej hyperkocky na účely testu. V tomto prípade nemajú jednotlivé časti rovnaké pravdepodobnosti výskytu.

Cat Test

Cat Test je variáciou kolízneho testu, avšak s tým rozdielom, že kolízie sú počítané iba pre jednu bunku. Bunky sú vytvárané podobne ako v prekrývanej verzii testu, ale princíp je možné použiť aj pri nezávislej verzii (neprekrývanej), a funguje aj pre ľubovoľné rozdelenie. Za predpokladu platnosti H_0 a pre veľké n má počet bodov spadajúcich do vybranej bunky približne Poissonové rozdelenie so strednou hodnotou rovnou počtu bodov vynásobených objemom sledovanej bunky, za predpokladu, že body sú po dvojici nezávislé pre $n \to \infty$. V prípade, že body sú nezávisle, je rozdelenie binomické. Test má zmysel iba ak je skúmaná bunka často zasiahnutá, alebo priemer zásahov je dostatočné veľký a vybraná bunka je zasiahnutá veľmi zriedka, čím sa vlastne zmeria špecifická zlá vlastnosť generátora.

Časové úseky medzi dvoma návštevami bunky

Test využíva pokročilejší prístup. Zameriava sa na počet krokov, ktoré uplynuli medzi dvoma návštevami bunky. Tento postup sa urobí pre všetky bunky a všetky návštevy. Maurer[39] navrhol špeciálny prípad tohto testu, pri ktorom sa využíva ako štatistický test priemer logaritmov dĺžok všetkých úse-

kov. Dôvod je ten, že takto získaný priemer meria istým spôsobom entropiu testovanej postupnosti.

Narodeninový test

Narodeninový test je zjemnením sériového testu, pričom sa prerozdelí n bodov do k buniek, presne ako pre sériový test. Pre každý bod sa zoberie číslo bunky, a týchto n čísel sa utriedi vzostupne, t.j. $I_1 < I_2 < \cdots < I_n$. Test vypočíta vzdialenosti $I_{j+1} - I_j$, pre $1 \leq j < n$, následne spočíta počet kolízií medzi týmito vzdialenosťami. Distribučná funkcia podľa H_0 má pre tento test Poissonové rozdelenie, so strednou hodnotou $\frac{n^3}{4k}$. Knuth[26] uvádza ako ďalší postup chi-square test, ale podľa [44] je lepšie vykonať tento test dvojúrovňovo, t.j. aplikovať ho N krát a porovnáť s Poissonovým rozdelením so strednou hodnotou $N.\frac{n^3}{4k}$ a tak získať $p - hodnotu$. Odchýlka tejto aproximácie bude od originálnej distribučnej funkcie minimálna ak $N.n^3 \leq k^{\frac{5}{4}}$. Zmysel tohto testu spočíva v tom, že isté typy generátorov vytvárajú body, ktoré majú približne rovnaké vzdialenosti medzi sebou. Túto vlastnosť majú všetky LCG, pre $t > 1$, vzhľadom na ich rovnomernú štruktúru pre 2 a viac dimenzií.

Blízke dvojice bodov v priestore

V tomto prípade sa zase používa vytváranie bodov nezávisle a uniformne v jednotkovej hyperkocke, ale tento krát sa skúmajú vzdialenosti bodov v priestore. Na určenie vzdialenosti je potrebné zaviesť normu $\lambda_p|\cdot|_p^o$ v jednotkovom toruse(prstenci), ktorý vznikne zlúčením protiľahlých strán po dvojiciach u jednotkovej hyperkocky. Potom body, ktoré sú v kocke na opačných stranách, sú vlastne blízko seba. Vzdialenosť dvoch bodov u_i a u_j sa definuje ako $D_{n,i,j} = |u_j - u_i|_p^o$, kde

$$|x|_p^o = \begin{cases} [\min(|x_1|, 1-|x_1|)^p + \cdots + \min(|x_t|, 1-|x_t|)^p]^{\frac{1}{p}} & \text{ak } 1 \leq p < \infty \\ \max(\min(|x_1|, 1-|x_1|) + \cdots + \min(|x_t|, 1-|x_t|)) & \text{ak } 1 = \infty, \end{cases} \tag{3.1}$$

Výhoda torusu spočíva v odstránení okrajov, takže je oveľa ľahšie získať lepšie aproximácie relevantnej distribúcie. Podrobne rozpísanú túto metódu možno nájsť v [44].

3.1.3 Testy merajúce viacero podpostupností náhodnej dĺžky

Do tejto kategórie testov spadajú všetky testy, ktoré vytvárajú každú podpostupnosť takým spôsobom, že vytvárajú body u_i dovtedy, pokiaľ nenastane nejaká udalosť a súčasne požadované číslo u_i je náhodne.

Zberač lístkov(coupon collector) test

V tomto teste sa rozdelí interval $(0, 1)$ na d rovnako dlhých častí, a vypočíta sa, koľko náhodných čísel u_j musí byť vytvorených, aby bolo obsiahnuté aspoň jedno číslo v každom intervale. Následne sa spustí simulácia na testovaných číslach a spočítajú sa dĺžky takto získaných postupností. Následne sa porovnajú získané hodnoty s teoretickými pomocou chi-square testu. Tento test je možné rozšíriť na ľubovoľne rozdelenie jednotkovej hyperkocky pri zachovaní rovnakých objemov.

3.2 Testy pre bitové reťazce

V tom odseku sú popísané testy určené pre bitové generátory, ktoré vytvárajú reťazce bitov $b_0, b_1, b_2, \ldots$, pre ktoré je potrebné testovať hypotézu H_0, že b_i sú nezávislé a nadobúdajú hodnoty 0 alebo 1 s rovnakou pravdepodobnosťou.

3.2.1 Testy merajúce jeden dlhý reťazec

Testy tohto typu boli vytvorené hlavne na základe potrieb šifrovania, kde je hlavnou požiadavkou vysoká entropia a zložitosť. Medzi testy entropie už uvádzané patria Maurerov[39] test a binárne verzia polynomiálnych testov sú tiež zamerané hlavne na entropiu.

Meranie lineárnej zložitosti

Jedným zo spôsobov, ako testovania zložitost je vyhodnocovanie, ako sa lineárna zložitosť L_l pre prvých l bitov reťazca zvyšuje ako funkcia s parametrom l. Lineárna zložitosť L_l je definovaná ako najmenší možný stupeň rekurencie riadiacej sa reťazcom. Táto funkcia je neklesajúca a rastie v celočíselných skokoch v určitých hodnotách z l. Ak celý reťazec vzniká lineárnou rekurziou so stupňom $k \ll n$, potom sa zastaví test na $l = k$ a odhalí túto štruktúru.

Skákajúci test zložitosti(jump complexity test)

Počíta počet skokov J vo funkcii uvedenej vyššie. Pre vysoké hodnoty n a J má približne normálnu distribučnú funkciu so strednou hodnotou a disperziou uvedenou v [44].

Test dĺžky skoku(jump size test)

Tento test počíta koľko bolo skokov rovnakej dĺžky a následne ich porovná s teoretickou geometrickou distribúciou s parametrom $\frac{1}{2}$ pomocou chi-square testu.

Dvojúrovňový lineárny test

Odlišný prístup k tomuto testovaniu uvádza [48], ktorého implementáciu možno nájsť aj v tetovacom balíku NISTu [50]. Používa sa dvojúrovňové testovanie s veľkým počtom N prvotných testov a pomerne malou dĺžkou reťazca. Vypočíta N hodnôt L_n, spočíta koľkokrát každá z hodnôt nastala a následne použije chi-square test na porovnanie získaných hodnôt s očakávanými. [44] uvádza, že tento test ani zďaleka nedosahuje také dobré výsledky ako predchádzajúce dva.

Lempel-Ziv

Tento test využíva skomprimovateľnosť reťazca, pomocou Lempel-Zip kompresie. Princíp spočíva v spočítaní počtu postupne vytváraných rôznych slov v reťazci. Reťazec je vyhlásený za náhodný, ak ho nie je možné dostatočne skomprimovať. Skutočne náhodná postupnosť by mala mať charakteristický počet rôznych slov.[44] uvádza, že pre veľmi veľké n počet slov je normálne distribuovaný so strednou hodnotou $\frac{n}{\log_2 n}$, a disperziou $\frac{0.266n}{(\log_2 n)^3}$. Avšak tento odhad strednej hodnoty a disperzie už nestačí pre $n = 2^{24}$. Balík TestU01[44] používa lepšie odhady, avšak aj tak tento test nedosahuje dostatočnú kvalitu a existuje pomerne dosť testov, ktorými je nahraditeľný.

Spektrálny test

Počíta z bitového reťazca niektoré z diskrétnych Fourierových koeficientov. Fourierov koeficient je komplexné číslo, definované ako
$f_l = \sum_{j=0}^{n-1} (2b_j - 1)e^{\frac{2\pi\iota l}{n}}$, pre $l = 0, 1, ..., n-1$,kde
$\iota = \sqrt{-1}$.
$|f_l|$ je veľkosť komplexného čísla.
Prvý test počíta počet P_h všetkých $|f_l|$, pre $l \leq \frac{n}{2}$, ktoré sú menšie ako nejaká zvolená konštanta h. Za predpokladu, že n je veľké a $h = \sqrt{2.995732274n}$, P_h je približne normálne distribuované so strednou hodnotou $\mu = 0.95\frac{n}{2}$ a disperziou 0.05μ. V TestU01[44] je naimplementovaných niekoľko spektrálnych

testov, avšak známe aproximácie distribúcie týchto testov niesu dostatočné.

Autokorelácie

Automatické opravovanie vzorky dĺžky s v bitovom reťazci, definované ako
$A_s = \sum_{i=0}^{n-s-1} b_i \oplus b_{i+s}$, kde
$\oplus$ je výlučne alebo (xor),
definuje zaujímavý štatistický test. Pre H_0, má A_s binomické rozdelenie s parametrami $(n-s, \frac{1}{2})$, čo je približne normálne rozdelenie pre dostatočne veľké $n-s$.

Run a gap testy

Binárna verzia run testu sa dá definovať následne. Postupnosť (beh/"run") rovnakých bitov za sebou v reťazci tvorí beh jednotiek alebo núl. Tieto jednotkové a nulové behy sa navzájom striedajú. Test počíta dĺžky týchto behov, tak aby ich nazbieral $2n$, kde n je počet behov jedného typu, pričom celková dĺžka reťazca potrebná na získanie $2n$ behov je vopred neznáma. Následne sa vytvorí t čísel $P_1, \ldots, P_t$, kde P_i je počet behov dĺžky i (spomedzi jednotkových aj nulových). Následne sa zvolí $k \leq t$, a vypočíta sa $P_k' = P_k + \cdots + P_t$, čo vlastne zgrupuje málo pravdepodobné výskyty do jednej skupiny. Potom sa použije chi-square test, na $P_1, \ldots, P_{k-1}, P_k'$. Očakavána pravdepodonosť výskytu behu dĺžky j je 2^{-j}. Každý beh núl a jednotiek je vlastne gap medzi dvoma výskytmi odlišných bitov. Preto je gap test a run test v binárnej verzii identický.

3.2.2 Testy merajúce viacero reťazcov rovnakej dĺžky

Rozdelenie množiny reťazcov na podmnožiny a počítanie zásahov v nich

V tejto časti sú popísané testy, ktoré sa snažia rozoznať závislosti v reťazcoch dĺžky m,kde počet týchto reťazcov je n, v zmysle takom, že niektoré z 2^m možností sa vyskytujú častejšie ako ostatné. Všetky tieto testy využívajú na-

sledujúci postup: Preskup 2^m možností do k kategórií, pre nejaké k, spočítaj koľko zo všetkých n reťazcov padne do každej z kategórií a porovnaj výsledok s očakávanými hodnotami.

Sériové testy

Spôsob, ako pracujú sériové testy v binárnom prípade je nasledujúci: Očísluj možné m-bitové reťazce od 0 do $2^m - 1$ a nech číslo X_j je počet výskytov reťazca j spomedzi všetkých reťazcov. Podobne ako v nebinárnom prípade, je možné použiť všetky metódy tam popísané. Pri prekrývanej verzii testu, je n bitov umiestnených do kruhu a každý blok m po sebe idúcich bitov určuje číslo bloku. Teoretické rozdelenie je také isté ako pre štandardný m-rozmený prekrývaný sériový test s $d = 2$ a $t = m$.

Binárne verzie už uvedených testov

CAT test, podobne aj test merajúci časové úseky medzi dvoma bunkami, má svoju binárnu verziu, kde bunky sú nahradené m-bitovými reťazcami, vytvorené buď s alebo bez prekrývania. Pre Maurerov test Maurer[39] dokázal, že je univerzálny, v zmysle, že je schopný rozpoznať ľubovoľný defekt v binárnom reťazci, za predpokladu, že parametre testu a veľkosť testovanej vzorka sa blížia do nekonečna vhodným spôsobom. Avšak v praxi je tento test menej citlivý ako test kolízii pri rovnakom testovanom objeme dát.

Hodnosť binárnej matice

Za veľmi kvalitný test, schopný odhaľovať lineárne závislosti medzi blokmi bitov je test hodnosti matice[35]. Matica o rozmeroch $k \times l$ sa naplní reťazcom dĺžky $m = k \times l$, a vypočíta sa hodnosť R tejto binárnej matice. Tento postup sa zopakuje n krát, pre všetky reťazce a následne sa porovná pomocou chi-square testu získané rozdelenie s teoretickým. Reťazce, ktoré sa lineárne opakujú, neprejdú týmto testom pre dostatočné veľké m.

Test najdlhšej postupnosti jednotiek

Tento test je variant run testu. Pre reťazec dĺžky m nájde jeho najdlhší podreťazec pozostávajúci zo samých jednotiek. Tento postup zopakuje n krát pre všetky reťazce. Získané rozdelenie sa porovná s teoretickým pomocou chi-square testu.

Hammingove váhy

Hammingova váha reťazca je počet symbolov, ktoré sú odlišné od nulového symbolu v použitej abecede. V binárnej verzii je to teda počet nenulových bitov v reťazci. Test založený na Hammingových váhach umožňuje rozpoznávať zhlukovanie jednotiek a núl. Princíp spočíva v meraní rozdelenia Hammingových váh na disjunktných podreťazcoch dĺžky m. Nadmerné zhlukovanie vedie k vyšším hodnotám disperzie Hammingových váh a súčasne vznikajú závislosti medzi Hammingovými váhami za sebou idúcich reťazcov. Hammingov test váh vytvára n neprekrývaných blokov dĺžky m a vypočíta pre ne Hammingove váhy, t.j. vypočíta H_i pre blok i. Pre hypotézu H_0 je očakávané binomické rozdelenie s parametrami $(m, \frac{1}{2})$.
Nech X_j je počet blokov s Hammingovou váhou j, pre $j = 0, \ldots, m$.

- Prvý test porovná rozdelenie X_j s očakávanými hodnotami štandartným spôsobom. Napr. pomocou chi-square testu.

- Druhý test, vypočíta
  $\chi^2 = (\frac{4}{m}) \sum_{i=1}^{n} (H_i - \frac{m}{2})^2$,
  kde pre m dostatočne veľké dosahuje približne chi-square rozdelenie so stupňom nezávislosti n. Pre $m = n$ je tento test známy ako monobit test, ktorý iba spočíta pomer jednotiek v reťazci dĺžky n.

- Tretí test počíta iba lineárne vsťahy medzi susednými H_i.
  $\widehat{\rho}_1 = \frac{4}{(n-1)m} \sum_{i=1}^{n-1} (H_i - \frac{m}{2})(H_{i+1} - \frac{m}{2})$
  Pre veľké n má $\widehat{\rho}_1 \sqrt{n-1}$ približne normálne rozdelenie.

- Štvrtý, odlišný test nezávislosti využíva $2n$ blokov po m bitov. Dvojica (H_i, H_{i+1}), pre $i = 1, 3, \ldots, 2n-1$ môže nadobúdať $(m+1)^2$ možných hodnôt. Test počíta počet výskytov každej z možností, a porovnáva s očakávanými hodnotami pomocou chi-square testu. Tento test ukázal podstatné závislosti medzi Hammingovými váhami za sebou idúcich reťazcov a špeciálnymi typmi LCG [29].

Testy náhodných ciest(random walk)

Náhodná cesta je postupnosť krokov, kde v každom kroku sa rozhodne náhodne, ktorým smerom sa vybrať. Na molekulárnej úrovni je tento jav známy tiež ako Brownov pohyb. V tomto prípade sa náhodná cesta vytvára na základe binárnej postupnosti, kde počiatočný krok začína v stave 0 a j-ty krok je vľavo ak $b_j = 0$ a vpravo ak $b_j = 1$. Formálne, ak sa zadefinuje
$S_0 = 0$, a
$S_k = \sum_{j=1}^{k} 2b_j - 1$,pre $k > 0$,
potom proces $\{S_k, k \geq 0\}$ je náhodná cesta. V rámci H_0 ,z binomického rozdelenia vzniká

$$p_{k,y} = P[S_k = y] = 2^{-k}\binom{k}{\frac{(k+y)}{2}} \text{ ak } k+y \text{ je párne a}$$
$$p_{k,y} = 0 \text{ inak.}$$

Naďalej platí predpoklad l je párne. Zadefinujme
$H = \frac{l}{2} + \frac{S_l}{2}$,
$M = \max\{S_k, 0 \leq k \leq l\}$,
$J = 2\sum_{k=1}^{\frac{l}{2}} \mathbb{I}[S_{2k-1} > 0]$,
$P_y = \min\{k : S_k = y\}$, pre $y > 0$,
$R = \sum_{k=1}^{l} \mathbb{I}[S_k = 0]$,
$C = \sum_{k=3}^{l} \mathbb{I}[S_{k-2}S_k < 0]$,
kde $\mathbb{I}$ slúži ako ukazovateľ smeru,
H je počet krokov do prava,

M je maximálna hodnota dosiahnutá na ceste,
J je podiel času strávený napravo od očakávanej cesty,
P_y je prvý časový okamih, kedy bolo dosiahnuté y,
R je počet návratov do bodu 0,
C je počet zmien trasy.

Teoretické pravdepodobnosti výskytu pre tieto zadefinované štatistiky sú:

$$P[H = k] = P[S_l = 2k - l] = p_{l,2k-l} = 2^{-l}\binom{l}{k}, 0 \leq k \leq l,$$

$$P[M = y] = p_{l,y} + p_{l,y+l}, 0 \leq y \leq l,$$

$$P[J = k] = p_{k,0}p_{l-k,0}, 0 \leq k \leq l, k \text{ je párne},$$

$$P[P_y = k] = (\frac{y}{k})p_{k,y}$$

$$P[R = y] = p_{l-y,y}, 0 \leq y \leq \frac{l}{2},$$

$$P[C = y] = 2p_{l-1,2y+1}, 0 \leq y \leq \frac{l-1}{2}.$$

Omnibus test, naimplementovaný v TestU01[44] volí 2 párne čísla $m > m_0 > 0$ ako parametre, a vytvorí n náhodných ciest dĺžky m. Pre každé $l \in \{m_0, m_0 + 2, \ldots, m\}$, vypočíta test n hodnôt pre $H, \ldots, C$, a porovná získané rozdelenie s očakávaným teoretickým za pomoci chi-square testu.

Kapitola 4

Porovnanie a rozbor známych testovacích balíkov

V tomto odseku sú popísané základné rozdiely medzi voľne dostupnými testovacími balíkmi, ich praktická využiteľnosť a postrehy získané pri práci s nimi.

Knuth V knihe Donalda Knutha, „The Art of Computer Programming“ ,1969 [26], možno nájsť dvanásť podrobne rozobraných testov. Knuthovu knihu možno považovať za bibliu testovania náhodných čísel. Stala sa najčastejšie citovaným zdrojom v tejto oblasti, a štandardom na veľmi dlhú dobu. V dnešnej dobe je však prudko zastaralá, v dobe keď bola publikovaná neexistovala ešte potreba na zabezpečenie generátorov pre účely šifrovania, preto je pre tieto účely kniha absolútne nevhodná. Existuje značné množstvo generátorov, ktoré nie je možné pokladať za kvalitné a ktoré bez problémov prejdú týmito testami. Kniha popisuje testy iba v teoretickej rovine a neponúka konkrétne implementácie. Preto vzniklo viacero implementácií týchto testov, čo však nie vždy ich autori zvládli bez chýb. Najlepšia implementácia všetkých týchto testov je obsiahnutá v balíku TestU01 [44].

Diehard Diehard je prvé komplexné softwarové riešenie vytvorené Marsagliom[33]. Vzniklo ako reakcia na naprostý nedostatok testovacích nástrojov. Marsaglia

sa zameral na vytvorenie nástroja, ktorý riešil zásadné nedostatky Knuthovej práce. A aj keď sa mu podarilo vytvoriť prakticky použiteľný nástroj, ktorý dosahoval výrazne lepšie výsledky ako Knuth, nedostalo sa jeho práci zaslúženého uznania u širšej verejnosti. Podľa mňa je to preto, že dokumentácia k projektu je značne odfláknutá a hlavne samotné testovanie má niekoľko principiálnych nedostatkov. Za hlavný problém považujem, že samotné testy sú prednastavené na pevno a nemožno im meniť parametre. Vyžaduje pevne daný formát vstupného súboru, s vopred predgenerovanými dátami, čoho dôsledkom je kvalitatívny problém testov. Prakticky to znamená, že testuje dosť malé vzorky vstupných dát, čo vedie z časového hľadiska k veľmi rýchlym výsledkom testov a teda k prakticky povrchnému otestovaniu.
V roku 2002 publikovali Marsaglia a Tang[34] sadu troch testov, ktorú považovali za lepšiu ako DIEHARD. Nepodarilo sa mi objaviť implementáciu presne tejto sady testov, ale balík TestU01[44] obsahuje zdrojové kódy niektorých z nich. Odhliadnuc od všetkých múch, je Diehard aj naďalej množstvom publikácií odporúčaný ako praktický nástroj na odhaľovanie nenáhodnosti.

Crypt-X Je balík určený na komerčné účely [21], preto nie je voľne dostupný ale možno ho zakúpiť v najlacnejšiej licencii na akademícke účely za $600, pričom ponúkajú verziu aj pre Windows aj pre Linux. Z literatúry sa mi o kvalite a chybách tohto balíka nepodarilo nič zistiť, takže uvádzam iba oficiálny zoznam testov, s alternatívnymi zdrojmi ich implementácie.

ENT Ent[24] je veľmi jednoduchý program, s naimplementovanou batériou testov, vyvinutý v roku 1998. V dokumentácii k nemu je napísané, že je vhodný na testovanie generátorov určených na šifrovanie, na testovanie skomprimovateľnosti a hustoty reťazca. Program je šírený v zdrojovej podobe aj v binárnej a je bez grafického rozhrania. Účinnosť tohto balíka testov spochybnil Louise Foley a navrhol balík [15] piatich testov, ktorý ho mal nahradiť.

NIST National Institute of Standarts and Technology vydal v roku 1999 balík pozostávajúci zo 16 štatistických testov [50]. Vznikol ako reakcia na

nedostatok možností komplexného testovania dlhých binárnych postupností či už TRNG alebo PRNG. Implementácia obsahuje výber z najlepších dovtedy známych testov a niekoľko nových riešení. Je to prvý voľne dostupný program s grafickým rozhraním, umožňujúci veľmi pohodlné testovanie. Obsahuje aj sadu predprogramovaných generátorov a teda umožňuje ich oveľa dôslednejšie testovanie ako v prípade vstupu zo súboru. Veľmi rýchlo sa stal štandardom na poli testovania, t.j. prejdenie týmito testami sa stalo štandardom merania kvality. Ani toto riešenie sa nevyhlo elementárnym chybám v implementácii a v roku 2004 boli publikované opravy [49].

TestU01 Najnovší prírastok na poli balíkov testov je TestU01[44] z roku 2005, vytvorený Pierrom L'Ecuyerom. Tento balík testov je výnimočný vo viacerých oblastiach. Za hlavnú výhodu považujem, že obsahuje naimplementované takmer všetky doteraz známe testy. Okrem toho obsahuje kvantum naimplementovaných generátorov a rôznych nástrojov na ich manipuláciu. Ponúka doteraz nevídané množstvo možností pre zostavovanie vlastných batérií testov a ich ďalšie štúdium. Obrovskou nevýhodou, ktorá bráni praktickému využitiu je fakt, že sa jedná o knižnice bez aplikácie. Akékoľvek praktické použitie vyžaduje znalosť programovacieho jazyka C, pričom knižnice sú predkompilované pre Linux, avšak umožňuje vývoj aj pod Windows, za pomoci programu CygWin. Keďže sa jedná o nový produkt, zatiaľ sa nestihol dostať do povedomia širšej verejnosti. Aj keď sa jedná iba o knižnice, je možné napísať pomerne veľmi rýchlo program, ktorý by bol schopný otestovať konkrétny generátor. Svojim rozsahom a kvalitou by sa mal stať najlepším zdrojom na ďalšie štúdium náhodnosti.

4.1 Metódy prístupu k testovaniu

V tejto časti je podrobnejšie rozpísané, ako by sa podľa mňa malo správne pristupovať k testovaniu náhodných čísel. V minulej časti bol uvedený zoznam testovacích nástrojov, pričom každý z nich vznikal v inom období, a

dosť často s odlišnou motiváciou. Knuth a Marsaglia vytvorili testy, ktoré považovali za vhodné na testovanie čísel určených na rôzne simulácie. NIST sa zas zameral na bezpečnosť šifrovania. Ent a Diehard sa snažili o oboje. Tu vyvstáva základná dilema, či je potrebné vytvárať rôzne batérie testov v závislosti od nimi testovaných generátorov, alebo stačí a je možné vytvoriť univerzálny balík testov, ktorý bude dostačujúci na všetky účely. V roku 1992 fyzici objavili, že aj tie najlepšie generátory, ktoré prešli testami môžu vytvárať za istých okolností chybné dáta. Preto je nevyhnutné si uvedomiť, na aký účel je daný generátor používaný a prispôsobiť tomu svoje požiadavky. Ideálny spôsob analýzy ponúka kombinácia všeobecného testovania spojená s konkrétnymi testmi v závislosti od požiadaviek na kvalitu výstupu.

Ďalej sa budem venovať iba všeobecnému testovaniu, keďže špecifické testy vyžadujú podrobnú analýzu každého problému a hľadanie konkrétnych vhodných testov. Ak sa zameriame iba na všeobecné testovanie, bez prihľadnutia na konkrétne potreby, je nevyhnutné siahnuť po riešení, ktoré spĺňa najprísnejšie bezpečnostné štandardy. Za jediný takýto uznávaný nástroj je považovaný testovací balík od NIST[50]. Dôvody sú nasledujúce:

- National Institute od Standarts and Technology, ktoré je tvorcom tohto balíka, dáva záruku, že sú naplnené bezpečnostné štandardy.
- Oficiálne je považovaný za najrozsiahlejší zdroj testov určených na účely šifrovania, čím sa automaticky plní najdôležitejšie kritérium.
- Bol použitý na vyhodnotenie AES, takže slúžil ako nástroj na meranie jeho bezpečnosti.
- Používa iba binárne testy, hlavne na meranie uniformity, čo je pre praktické bezpečnostné účely výhodnejšie. Ostatné batérie sa testom uniformity vyhýbajú.
- Výskum a voľba testov prebiehal v NISTe, čo by malo byť istou zárukou neprekrývania sa testov. Avšak otázka ich závislosti nebola zatiaľ

vyriešená.

Mojim pôvodným zámerom bolo štúdium týchto testov NISTu s cieľom zistiť závislosti medzi jednotlivými testmi. Môj zámer sa zmenil v okamihu, keď som objavil najnovší balík TestU01 [44], ktorý bol publikovaný v tom istom roku, ako začala vnikať táto práca. Na prvý pohľad málo známy projekt ušiel mojej pozornosti a skončil medzi množstvom dokumentov určených na ďalšiu analýzu. Neskôr som sa k nemu vrátil a kompletne zmenil prístup. TestU01 je knižnica súborov, ktoré vznikali dvadsať rokov vďaka práci Pierra L'Ecuyera. Pomerne krátka dokumentácia ponúka zaujímavé myšlienky ohľadne odlišného prístupu k p-hodnotám ako v prípade NISTu. Projekt okrem implementácií snáď všetkých známych testov, až na niekoľko výnimiek, obsahuje aj predvolené batérie testov. Rozdeľuje ich na číselné a binárne. Ďalším prínosom je možnosť testovať celé triedy generátorov naraz.

Základný rozdiel medzi TestU01 a NISTom je v tom, že TestU01 je nekomerčná knižnica, určená na ďalšie štúdium a vývoj, neobsahujúca ani elementárny spustiteľný program. Každý test vyžaduje potrebu programovať, i keď veľmi jednoducho a s obrovskými možnosťami nastavení, ale aj tak je prakticky nepoužiteľná pre laika. A práve to, že nie je komerčne prednastavená, považujem za najväčšiu výhodu. Základne dôvody, prečo je TestU01 atraktívnejší ako NIST.

- Obsahuje naimplementované najznámejšie generátory a najrôznejšie nástroje na manipuláciu s nimi, spolu s možnosťou jednoducho implementovať vlastné, čo ponúka obrovskú výhodu pri testovaní. Je zásadný rozdiel, či sú k dispozícií iba predgenerované dáta generátora v súbore, alebo, či je možné na požiadanie vytvárať akékoľvek množstvo náhodných dát(obmedzené len počtom vnútorných stavov generátora).
- Perametre NISTovských testov sú prednastavené natvrdo, umožňujú iba minimálne možnosti nastavenia. TestU01 ponúka celú plejádu nastaviteľných parametrov.

- Prístup k testovaniu. NIST stavia hlavne na princípe dvojúrovňového testovania, čo však nie je zárukou, že odhalí závislosti. TestU01 sa orientuje na silu primárneho testu, s automatickou možnosťou dvojúrovňového testovania. Hlavný rozdiel je v tom, že NIST testuje príliš krátke sekvencie bez možnosti dodatočného dotestovania a to kvôli problémom s už spomínanými obmedzeniami ohľadne dĺžky testovaného reťazca.

- Vyhodnocovanie $p-honoty$. NIST si natvrdo určí, aké množstvo testov môže skončiť s $p - hodnotou < \alpha$, naopak TestU01 nerobí žiadne rozhodnutia týmto smerom a považuje za správne na výstup odovzdávať celú p-hodnotu a nie len výsledok, či bol test úspešný alebo nie.

- Typy testov. NIST zanevrel na nebitové testy, na rozdiel od TestU01, ktorý obsahuje implementáciu číselných aj bitových testov. To umožňuje skúmať podrobnejšie nakoľko bola voľba NISTu týmto smerom správna.

- Obsahuje viaceré prednastavené batérie testov, čo značne vychádza v ústrety praktickému využitiu. Batérie sú rozdelené podľa dĺžky trvania, podľa typu testov a obsahuje aj implementáciu DIEHARD a časť NISTovskej batérie.

- Obsahuje aj možnosť masového testovania celých rodín generátorov.

Najzávažnejšie nedostatky TestU01.

- Neobsahuje akýkoľvek hotový spustiteľný program.

- Dokumentácia k samotným testom a generátorom je dostatočná iba po technickej stránke. Teoretické princípy fungovania testov sú síce čiastočne a prehľadne spracované, ale na ich podrobnejšie štúdium to nestačí. Teoretické pozadie generátorov sa tam nenachádza.

- Základne prednastavenia. Toto je najzávažnejší nedostatok, na ktorom značne zlyhalo moje úsilie o podrobnejšie štúdium. Informácie o

zakladných parametroch jednotlivých testov sú kusé. Vyžadujú často dodatočné štúdium a je veľmi namáhavé uhádnuť nielen ich význam ale hlavne vhodné hodnoty. Naštastie batérie obsahujú prednastavené testy, takže je možné prakticky ich používať, ale ich prenastavenie je bez znalosti súvislostí hra na mačku a myš.

- Prekrývanie sa testov. Batérie, ktoré sú prednastavené obsahujú aj redundantné testy. NIST tvrdí, že ich testy nie sú redundantné, ale dôkaz nikdy nepodali.

Na základe skutočností tu uvedených, som sa rozhodol vypracovať systém jednoduchých programov, ktorý umožňuje využívať základnú funkcionalitu TestU01 a robí z komplikovaného systému knižníc, primárne určeného na prácu v jazyku C pod Linuxom, použiteľný nástroj na testovanie generátorov a podrobnejšie štúdium získaných výsledkov. Celý systém v konečnom dôsledku umožňuje prehľadne skúmať správanie sa testov. Finálne jednoduché grafické rozhranie je nezávislé na TestU01 a môže slúžiť ako prehľadný nástroj na lepenie výsledkov z rôznych testovacích balíkov. Viac v prílohe.

4.1.1 Interpretácia výsledkov testov pre TestU01

Po vykonaní ľubovoľnej série testov je potrebné vyhodnotiť výsledky, nakoľko bol generátor úspešný. NIST berie túto úlohu na seba a na základe presných pravidiel jednoznačne rozhoduje o tom, či bol generátor úspešný alebo nie (kapitola: Interpretácia výsledkov testovania). TestU01 nemá zavedené žiadne pevné kritéria a necháva toto rozhodnutie na osobnom uvážení, na základe získaných p-hodnôt jednotlivých testov. Môj osobný názor je, že nie je možné objektívne určiť pevnú hranicu, pre ktorú by bolo jasne možné robiť rozhodnutia typu áno/nie. Preto som sa zameral na hľadanie spôsobu ako rozumne pristupovať ku kvantu výsledkov jednotlivých testov.

Kvalitatívny vs. kvantitatívny prístup. Čo je vhodnejšie? Veľa testov, bez rozlišovania ich kvality, alebo menej, vo všeobecnosti považovaných za

kvalitnejšie? Ťažko povedať. Ideálne by bolo mať jednu batériu, o ktorej naisto vieme, že vykonáva testy, ktoré nie sú redundantné a pritom by testovala všetko čo obsahujú zvyšné testy. Táto batéria zatiaľ neexistuje a jej vytvorenie si vyžaduje veľmi podrobné štúdium , navrhnutie nových postupov pri vyhodnocovaní výsledkov testov a obrovské množstvo vykonaných testov. Z týchto dôvodov považujem za správne kombinovať oba prístupy.

TestU01 obsahuje tri základne bitovo orientované batérie testov Alphabit, BlockAlphabit, Rabbit, ktorých čas trvania je veľmi krátky(do sekundy) a pritom dokopy obsahujú 158 jednoduchých testov. Je až zarážajúce, že batéria DIEHARD, ktorej čas behu je rádovo 100 krát dlhší, vykazuje oveľa nepresvedčivejšie výsledky. Okrem týchto troch bitových verzií a nebitovej implementácie DIEHARD, existujú aj tri nebitové batérie, ktorých čas trvania je však podstatne dlhší. Najmenšia batéria SmallCrush trvá v priemere dve minúty, ale ďalšie dve Crush a BigCrush už majú trvanie v hodinách. NIST[50] obhajuje bitové testy za efektívnejšie ako nebitové a preto iné ani neobsahuje. Na základe zbežného skúmania som dospel k názoru, že z pohľadu kvantitatívneho prístupu je názor NISTu opodstatnený. Bitové verzie sú výrazne rýchlejšie a vo veľa prípadoch prísnejšie ako nebitové. Ale z pohľadu kvalitatívneho, nie je možné nebitové testy vylúčiť, pretože skúmajú hlbšie súvislosti, ktoré sú potrebné na dodatočné doskúmavanie. Preto navrhujem následný postup.

Postup pri vyhodnocovaní výsledkov pomocou mnou naprogramovaného systému programov Tento môj postup je primárne určený pre balík TestU01, ale je možné ho využiť aj vo všeobecnej rovine.

- Ako prvé treba vykonať rýchle testy, t.j. Alphabit, BlockAlphabit, Rabbit, PseudoDIEHARD, SmallCrush, výsledky z nich získane bohato stačia na aplikovanie kvantitatívneho prístupu.
- Vykoná sa kvantitatívne vyhodnotenie(postup je popísaný nižšie), ktorého výsledkom by malo byť zaradenie do jednej z troch kategórií.

1. generátory, ktoré úvodným testovacím systémom prešli bez problémov
2. generátory, ktoré boli vyhodnotené drvivou väčšinou testov za náhodné, ale niektoré testy ich zamietli.
3. generátory, ktoré boli zjavne nenáhodné

Moja skúsenosť s týmto zaradením je taká, že prvá kategória veľmi často zvládne aj dodatočné testy. Druhá zas v závislosti od toho, aké prísne kritéria sa zvolili, veľmi často neprejde dodatočnými testmi. V zásade druhá kategória nie je vhodná z bezpečnostných dôvodov, pretože do nej väčšinou spadajú generátory ako napr. LCG, ktoré vo všeobecnosti nepatria medzi najvhodnejšie. Tretiu kategóriu nieje potrebné podrobovať ďalšiemu skúmaniu, generátory v nej umiestnené sú nevhodné.

- Ak bol generátor zaradený do prvej kategórie, má zmysel vykonávať ďalšie testy a to Crush a BigCrush. V tomto prípade však vykonávanie týchto testov často nemusí viesť k rozhodnému výsledku. V prípade nejednoznačného výsledku je na zvážení, či ďalej testovať konkrétnymi testmi znovu kritické miesta, alebo vzhľadom na plánované využitie generátora ho uznať za dostatočne náhodný.
  Ak bol generátor v druhej kategórii, tak má zjavné vady, a dodatočné testy ich len viac odhalia. Ale tieto generátory nemusia byť nutne zavrhnuté a určite si nájdu svoje miesto v mnohých praktických oblastiach, nekritických na bezpečnosť.

Postup kvantitatívneho vyhodnocovania Hlavný princíp, na ktorom stojí táto myšlienka, spočíva v predpoklade, že pre skutočne náhodnú postupnosť je $p-hodnota$ dobrého testu uniformne distribuovaná, t.j. ak sa vykoná ten istý test na rôznych dokonale náhodných postupnostiach, tak $p-hodnoty$ jednotlivých testov majú uniformné rozloženie v intervale $(0, 1)$. Táto vlastnosť je využívaná NISTom[50] na účely druhoúrovňového testovania, kde sa meria ich uniformita pre dostatočne veľký počet primárne vykonaných tes-

tov. Moja úvaha spočíva v tom, že ak je $p-hotnoda$ uniformne distribuovaná pre N rovnakých testov, mala by byť uniformne distribuovaná aj v prípade, keby tých N testov nebolo rovnakých. Veď keď je pravdepodobnosť výskytu $p-hodnoty$ pre ľubovoľný test náhodná na intervale (0,1), tak bez ohľadu nato, aké testy použijem, vždy by mali mať náhodné $p-hodnoty$. V prípade, že by sa použila zakaždým nová postupnosť pre každý rôzny test, neexistujú podľa mňa žiadne dôvody, prečo by nemala byť táto úvaha správna. A je úplne jedno, či sa nejaký test použil viac krát, alebo sa použili závislé testy, testujúce podobným spôsobom. V prípade, že sa používa tá istá postupnosť vopred vygenerovaných dát, môžu nastať podľa mňa problémy. Ak je nejaký test vykonaný dvakrát, tak pre rovnakú postupnosť musí vrátiť rovnakú p-hodnotu, čo automaticky obmedzuje pozornosť testujúceho na to, či rôzne batérie neobsahujú rovnaké implementácie toho istého testu. Druhý výrazný problém pri násobnom používaní tých istých vstupných dát je s podobnosťou testov a ich vzájomnou závislosťou. Testy v batériách nie sú nezávislé, čo by mohlo viesť k tomu, že pri závislých testoch by mohla existovať istá závislosť medzi finálnymi $p-hodnotami$. Rozsah dôsledky týchto závislosti je podľa mňa aktuálne neurčitelný.

Tu nastáva praktické využitie týchto úvah. Je jedno, aké testy človek použije, stačí ich výsledky len ľubovoľne pridávať do databázy. Ak sa testuje univerzálna vlastnosť, je vhodné, aby počty rôznych testov boli približne vyrovnané. Ak sa testuje špeciálna vlastnosť, je možné používať testy z konkrétnej oblasti, a vyhnúť sa tak skresľujúcim údajom z iných jednoduchších testov. Preto je vhodné pri kvalitatívnom testovaní vymazať staré testy z kvantitatívnej fázy testovania a použiť iba tie, ktoré sú kritické. Avšak pre účely kvalitatívneho testovania považujem túto metódu za málo účinnú.

Postup rozhodovania o uniformite rozloženia p-hodnôt Postup, ako sa rozhodovať, či je rozloženie $p-hodnôt$ uniformné alebo nie, je popísaný v kapitole o štatistickom testovaní, sekcia uniformná distribúcia $p-hodnoty$. Je tam zadefinovaný spôsob pomocou chi-square testu. V mojom prípade som

sa motivoval grafmi NISTu[50] a naprogramoval som automatické generovanie obrázkov, kde jeden je graf bodov v rovine(jeden rozmer je číslo testu v poradí a druhý jeho $p-hodnota$) a druhý je histogram rozloženia $p-hodnôt$. V prílohe je pokus o grafický náčrt dôkazu mojich tvrdení o uniformnom rozdelení pre dokonale náhodný zdroj. Z grafu je možné okamžite zaregistrovať, koľko testov malo $p-hodnotu$ veľmi blízku k 0. Optický test je dobrý spôsob na okamžité zaradenie do tretej kategórie. Na rozhodovanie medzi prvou a druhou kategóriou slúži hlavne histogram. Ak nie je prvý stĺpec najvyšší a všetky stĺpce majú približne rovnakú výšku, je možné ho zaradiť do prvej kategórie. Ak ale prvý stĺpec patrí medzi tie vyššie, je potrebné si pozrieť, koľko testov bolo veľmi blízko k 0. Ak ten počet nie je veľký, tak patrí generátor do druhej kategórie. Ak je prvý stĺpec najvyšší, nie je čo riešiť, a je odsúdený do tretej kategórie. Celý proces zaraďovania do kategórií je voľný a je možné si ho upraviť podľa chuti.

Záruky kvality použitých testov V dokumentoch NISTu [50] je rozsiahly popis situácií, ktoré môžu spôsobiť, že naimplementované testy nevracajú $p-hodonty$ uniformne distribuované, aj keď sú skutočne náhodne. Táto situácia môže nastať buď zlou implementáciou testu, alebo zlým odhadom rozdelenia pre rôzne nastavenia. V knižnici TestU01 nie je žiadna záruka, že testy sú dostatočne kvalitne naimplementované a nastavené. Útechou môže byť, že jej autor Pierr L'Ecuyer je dosť známa autorita a knižnica vznikala dve desaťročia. Ja som sa rozhodol preveriť svoje úvahy o vhodnosti týchto testov prakticky a spustil som ich pre niekoľko typov generátorov, o ktorých som s určitosťou vedel, nakoľko sú kvalitné. Testy sa správali rozumne pre všetky batérie a nadobudol som pocit, že ak aj nejaké sú chybne naimplementované alebo nesprávne prednastavené, z hľadiska kvantitatívneho testovania by to nemalo robiť zásadné problémy. Vizuálny náznak dôkazu mojich pokusov sa nachádzajú v prílohe.

Porovnanie kvality testov balíku TestU01 s inými batériami Výskum v tejto oblasti bol z mojej strany značne obmedzený množstvom vyko-

naných testov. Dôvody, pre ktoré je komplikované dôkladne porovnávanie.

- Sila TestU01 spočíva hlavne vo veľkom množstve predprogramovaných generátorov. Testovanie iných generátorov vyžaduje ich implementáciu do systému, alebo predgenerované dáta v súbore. Problém je v tom, že TestU01 obsahuje veľa testov, pre ktoré je potrebné veľké množstvo údajov, takže súbory s predgenerovanými dátami v rozsahu desiatok MB nie sú pre ne dostatočné. Našťastie batérie bitových verzií testov obsahujú priamu možnosť načítavať dát zo súboru a stačí im aj menšie množstvo údajov. Ale ostatné batérie s touto možnosťou nepočítajú. Preto najjednoduchší postup testovania je otestovanie už naimplementovaných generátorov a následne nagenerovanie súborov s ich pomocou. Potom je možné tieto súbory použiť v iných testovacích balíkoch.

- Celková manipulácia so súbormi a ich manuálne spúšťanie medzi rôznymi balíkmi vedie k neprimeraným časovým nárokom.

Preto akékoľvek závery, ktoré som z týchto porovnávaní nadobudol, nie sú podložené dostatočným množstvom testov. Ak mám porovnávať NIST vs. TestU01, tak TestU01 má tvrdšie kritéria na kvalitu generátora, t.j. boli prípady kedy NIST vyhodnotil generátor za vhodný, TestU01 ho označil za problémový.

4.2 Ďalšie návrhy do budúcna

Je potrebné si uvedomiť, že existuje nekonečne veľa testov na testovanie nezávislosti a uniformity výstupov RNG. Výber tých "správnych" spomedzi nich vyžaduje aplikovanie subjektívnych kritérií. Našťastie existuje pomerne dosť takých, ktoré človeku prídu na myseľ prirodzeným spôsobom. Potreba použitia určitých testov je do značnej miery závislá od toho, na aké účely bude daný generátor použitý.

Aké sú vlastne možnosti pri porovnávaní účinnosti testov? Každé porovnávanie vyžaduje určenie konkrétnych kritérií, ktoré sú aplikovateľné na porovnávané objekty. To je v prípade testov takmer nemožné vzhľadom k tomu, že kvalita testu závisí od tvaru špecifických štruktúr, ktoré sa testom rozpoznávajú, čiže alternatívna hypotéza pre jeden test je ťažko porovnateľná s druhou. Keďže pri testovaní generátorov nie je definovaná alternatívna hypotéza, dá sa testovať proti čomukoľvek, čo sa líši od H_0. Je možné vybrať konkrétnu triedu generátorov a porovnávať rôzne vlastnosti (napr. efektívnosť, t.j. čas trvania) rôznych testov a ich schopnosti rozpoznávať nenáhodnosť v tejto špecifickej triede. Rozsah získaných výsledkov je silne závislý od zvolenej triedy generátorov. Vyhodnocovanie získaných výsledkov by malo byť vždy s prihliadnutím, nakoľko je daná trieda vo všeobecnosti dobrým zdrojom kvalitných generátorov.

4.2.1 Návrh na postup, ako vytvoriť novú batériu testov

V tejto časti by som chcel predniesť postup, ako by sa mohlo postupovať pri vytváraní batérie, ktorá by mala testovať čo najviac možných vlastností, ale nemala by mať závislé testy, v zmysle testovania rovnakej vlastnosti nenáhodnosti. Dokonalá batéria v tomto zmysle asi nikdy vytvorená nebude, ale vždy sa dá pokúsiť o lepšie aproximácie ideálneho stavu. Môj návrh na postup:

1. Aby bola zachovaná podmienka možnosti testovať čo najviac vlastností

nenáhodnosti, je potrebné vychádzať z množiny všetkých známych testov. Treba naplniť čo najviac výsledkov týchto testov na čo najväčšej množine generátorov.

2. Následne je potrebné určiť kritéria, na základe ktorých sa bude určovať závislosť. Závislostí môže byť viacero. Ak zoberieme množinu vlastností, ktoré testuje jeden test a množinu vlastností, ktoré testuje druhý, tak tieto množiny môžu byť buď disjunktné, môžu mať prienik, alebo jedna množina môže byť nadmnožinou druhej. Ak tieto množiny nemajú prienik, tak sú navzájom nezávislé. Ak je jedna množina podmnožinou druhej, tak je jej test závislý jednoznačne. To aké sú tie množiny a či a nakoľko sa prekrývajú, nie je možné určiť jednoznačne. Preto kritérium závislosti nebude získané na základe analýzy pozadia testov, ale na základe ich spoločného správania v kritických situáciách. Ak dva testy v nejakom prípade zlyhajú, tak každý z nich objavil vlastnosť, ktorú generátor nespĺňal. To, či táto vlastnosť bola rovnaká, sa nevie. Ale ak vychádzame s obrovského množstva testov a máme možnosť analyzovať správanie týchto dvojíc testov a ak jeden z týchto testov zlyhá, vieme určiť, ako sa v tom prípade správal druhý test. V závislosti od % zlyhaní druhého testu je možné aspoň čiastočne určiť závislosti medzi týmito testmi. Ak druhý test zlyhal v 100% prípadov kedy zlyhal prvý, tak to znamená, že prvý test bol zbytočný, lebo druhý test ho dokáže nahradiť. Táto vlastnosť je symetrická a teda % súčasne zlyhaných testov je potrebné vypočítať pre oba testy. Kritéria závislosti nie je možné budovať na tom, že test bol úspešný, ale iba na tom, že bol neúspešný.

3. Vytvorí sa komplexný graf K_n s n vrcholmi a každá hrana je obojsmerná, pričom vrchol symbolizuje test a hrana symbolizuje % prípadov zlyhania, t.j. pre 2 vrcholy A, B hrana $A- > B$ symbolizuje % zlyhaní B keď zlyhal A a $B- > A$ symbolizuje % zlyhaní, kedy A zlyhalo, vtedy keď zlyhalo B.

4. Orezávanie grafu. Odstránia sa všetky vrcholy, z ktorých viedla hrana obsahujúca číslo 100%. Tým sa odstránia automaticky všetky testy, ktoré prinášali duplicitné výsledky. V prípade, že hrana obsahuje v oboch smeroch 100% tak sa neodstránia oba vrcholy, ale sa nahradia iba jedným z nich, a to tým, ktorý ma menší súčet % z neho vychádzajúcich hrán. Tento graf je možné orezávať podľa ľubovoľných kritérií, obohatených o časové hodnoty, či váhy hrán v závislostí od počtu vykonaných testov.(je rozdiel či testovaná hrana mala v 3/10 testov závislosť alebo či pri 300/1000). Percento závislosti je možné postupne znižovať, kým sa nedostane počet vrcholov v grafe na dostatočne nízke číslo. Ďalším z možných vylepšení je obohatenie cien hrán o vnorené prelievanie závislostí. t.j. ak sú 3 vrcholy A, B, C, a $A- > B = x\%, B- > C = y\%$ tak hrana $A- > C$ by mohla byť obohatená o váhu $x.y$. Dobrý spôsob ponúka aj algoritmus Google Page Rank.

Celý postup je len návrh, ktorého konkrétna implementácia si vyžiada viacero obmedzení a parametrov. Napr. má zmysel ho aplikovať na konkrétnej triede generátorov, prípadne vtedy, ak sú počty všetkých testov rovnaké. Celý systém váh je možné odignorovať a použiť iba pôvodné %, ak boli všetky testy vykonané rovnaký počet krát na rovnakých generátoroch.

4.2.2 Návrh na zjednodušenie testovacích systémov

Vytvorenie komplexného softwarového riešenia Ako sa ukázalo počas mojej práce najväčšie obmedzenia spôsobujúce nemalé problémy boli vyvolané nepraktickým dizajnom všetkých známych aplikácií. Ideálne pre ďalšie možnosti analýzy by bolo, keby sa podarilo vytvoriť komplexný systém, ktorý by umožňoval spúšťať testy priamo cez GUI rozhranie. Testy by mali mať viacero prednastavených možností, od úplných nastavení natvrdo, až po kompletnú možnosť meniť všetky parametre. Taktiež by bolo vhodné mať možnosť vybrania si zo zoznamu všetky možné naimplemnetované a hlavne prednastavené generátory. Každý výsledok vykonaného testu by sa mal ukla-

dať automaticky do databázy aby bola možnosť kedykoľvek vrátiť sa k výsledkom a súčastne by sa tak automaticky tvorila databáza, ktorá je alfou a omegou ďalšieho postupu pri analyzovaní závislosti testov a hľadaní nových batérií na konkrétne účely. Ďalej by mal existovať spôsob ako jednoducho pridať implementáciu testu či generátora. Malo by byť možné ho pridať či už ako zdrojový kód, alebo knižnicu či plugin. Celý systém by mal byť umiestnený na Internete, aby bola databáza prístupná všetkým. Mal by umožňovať dynamické ťahanie výstupov z generátora cez stream ako aj dynamické testovanie cez stream. Potom by mohol byť projekt komerčne využiteľný ako generátor čísel a overovateľ náhodnosti. Môj názor je taký, že pokiaľ sa nezautomatizuje manuálna práca, tak stále zostane hľadanie nových výsledkov len na bedrách nadšencov matematikov. Celý projekt je podľa mňa ako informatika realizovateľný a v konečnom dôsledku aj rentabilný, za predpokladu že sa podarí vytvoriť rozumný protokol ktorý by dokázal zabaliť všetku konkrétnu implementáciu a bol by dostatočne univerzálny, aby mohol slúžiť na posielanie streamov dát a prijímanie a spracovávanie výsledkov. Tento protokol by mal byť dostatočne rýchly aby bolo možné testovať v reálnom čase a súčastne dostatočne robustný, aby zvládol akúkoľvek implementáciu generátorov a testov. Pokiaľ nebude vytvorený nejaký takýto automatizovaný systém, ktorý by jednoducho púšťal, ukladal a spracovával testy, bude podľa mňa manuálna analýza výsledkov bez znalostí matematického pozadia a podrobnej analýzy princípov jednotlivých testov iba mrhanie času. Jediné čo bude možné takto dosiahnuť bude určenie či je nejaký test alebo batéria vhodná alebo nie, ale určite to nepovedie k objavovaniu prevratných batérií. Akonáhle by sa objavila takáto databáza voľne dostupná, som presvedčený že by na nej mohlo byť realizovaných viacero grafových algoritmov(podobných ako som navrhol vyššie).

Kapitola 5

Záver

Moja práca sa začala štúdiom teórie, ktorá je však veľmi prepojená s praktickou časťou, čo ma automaticky nasmerovalo k viacerým sofvérovým riešeniam. Ich štúdium a praktické hranie sa s testmi bolo veľmi náročné a neviedlo k žiadnym zmysluplným záverom. Najväčší problém bol v komplikovanosti vstupov a neúplnosti, prípadne ťažkej použiteľnosti výstupov. Odpoveď či je generátor dobrý alebo nie, je závislá od individuálnych požiadavok, takže odpoveď áno/nie nemohla viesť k serióznemu výskumu. Takže som sa rozhodol zamerať viacej na matematické pozadie skúmaných testov a pokúsiť sa nájsť tam nové možnosti. To si vyžiadalo opätovné štúdium a hľadanie ďalších zdrojov. Následný pokus o úspech v tejto oblasti viedol ku stavu, kedy som nevedel akým smerom sa vydať. Našťastie som objavil medzi dokumentmi výborný balík TestU01, ktorý bol publikovaný v tom istom roku ako začala vnikať táto práca. Obsahoval snaď všetky možné implementácie testov ako aj veľmi veľa naimplementovaných generátorov. Tento balík mal zásadnú nevýhodu, že nebol vo forme spustitelného programu, ale iba ako sada knižníc. Množstvo testov ma natoľko fascinovalo, že som sa vrátil k praktickému bádaniu. Tento krát ale nebol problém v neprehľadnosti výstupov, ale skôr v obrovských komplikáciách so vstupom. Každý jeden test vyžadoval samostatný program upravený na mieru. To ma priviedlo k úmornej a nudnej práci, pri ktorej som vytvoril jednoduchý program, ktorý umožňoval volať jednot-

livé testy bez potreby programovania. Vzhľadom nato, že balík je vyvíjaný v programovacom jazyku C a to ešte využíva Linuxovské knižnice, rozhodol som sa, že práca v tomto jazyku a za týchto okolností by bola pre mňa utpením a tak som vytvoril iba najnutnejšiu aplikáciu. Toto však stále nestačilo, pretože aj keď som mohol testovať koľko som chcel, nemal som stále možnosť prehľadného štúdia výsledkov. Tak som následne vytvoril jednoduchý databázový systém s web prostredím na vizualizáciu výsledkov. Následne som vytvoril program, ktorý automaticky prečíta a zanalyzuje neprehľadné výstupy a naplní ich do databázy. Veľkým a pretrvávajúcim obmedzením zostalo, že sa mi nepodarilo získať presné a vhodné nastavenia všetkých testov. Využíval som iba prednastavené batérie. Ďalším nedostatkom je, že ku naimplementovaným generátorom neexistujú vhodné parametre, takže som bol značne obmedzený v ich analýze. Z výskumu ktorý som následne robil a porovnával rôzne balíky s týmto, jednoznačne vyšiel TestU01 víťazne, so svojimi batériami testov. Keďže sa mi zdali všetky batérie použiteľné a o žiadnej z nich neexistuje podrobnejšia dokumentácia, navrhol som postup ako ich využívať všetky spolu ako nástroj testovania náhodnosti. Následne som vymyslel spôsob ako pristupovať k vytváraniu testovacích balíkov bez skúmania funkcionality jednotlivých testov, iba na základe znalosti výsledkov vykonaných testov. Tento nápad sa mi nepodarilo realizovať z časových dôvodov, ale hlavne z dôvodu neexistencie komplexnej databázy výsledkov. Záverom môžem povedať, že môj prístup v konečnom dôsledku viedol k jedinému mne známemu nástroju, ktorý by sa dal použiť ako odrazový mostík na podrobné skúmanie závislosti testov. Myslím si však, že podobné oveľa lepšie aplikácie musia mať dizajnéri už známych batérií, inak by ich prístup viedol ku krkolomným analýzam.

Kapitola 6

Prílohy a dodatky

6.1 Systém aplikácií na využitie testovacieho balíka TestU01

Celý systém obsahuje:

- Súbory TestManager.c a TestManager.exe. Jedná sa o aplikáciu vytvorenú v jazyku C, pomocou ktorej je možné spúšťať všetky testy, všetky batérie a niekoľko z generátorov. Program berie ako parameter názov textového súboru, z ktorého číta konfiguráciu. Konfiguračný súbor obsahuje generátor a zoznam testov, prípadne batérií, ktoré má program na danom generátore vykonať. Súbory s prednastavené generátory aj s parametrami, ako aj testy s parametrami majú tvar *.in. Aplikáciu je možné ďalej vyvíjať, keďže je zdrojový kód k dispozícii, jedinou podmienkou je nainštalovaný cyqwin pod Windows.

- Javovská aplikácia Distributor.class, ktorá sa volá pomocou dávkového súboru distribute.bat. Táto aplikácia na základe parametrov vytvorí zo súboru obsahujúceho zoznam generátorov a súboru obsahujúceho zoznam testov systém N dávkových súborov popísaných v predchádzajúcom bode. Pričom N je počet generátorov vo vstupnom súbore.

- Pre potreby testovania pomocou batérií je v adresári batteries systém dávkových súborov, ktorý vytvára zo vstupného súboru s generátormi systém adresárov, pre každú batériu jeden. Potom je možné ľubovoľne púšťať jednotlivé batérie naraz pre všetky generátory. Pre potreby testovania generátora zo súboru je možné testovaný súbor nakopírovať do adresáru batteries pod názvom binfile.dat. Pri veľmi letmom štúdiu dávkových súborov pozorný čitateľ ľahko pochopí systém organizácie a môže si ho upraviť podľa potrieb. Aktuálne existuje obmedzenie, že je možné testovať iba jeden generátor naraz, pokiaľ je zo súboru.

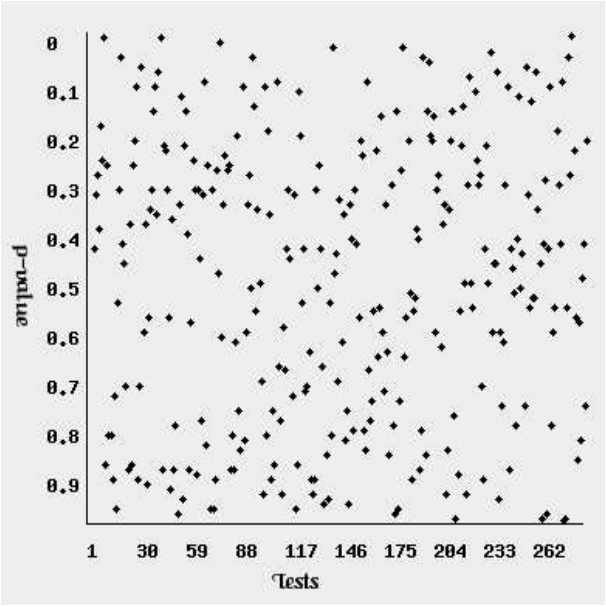
- Po spustení ľubovoľného testu je výstup zo súboru TestManager.exe, pokiaľ bol pustený za pomoci dávkového súboru, uložený na disk v podobe súboru s príponou *.res. Tento súbor obsahuje výpis všetkých testov vykonaných na jednom generátore. Pre každý jeden generátor sa vytvorí jeden súbor, bez ohľadu nato, koľko testov na ňom zbehlo. Každý tento res súbor je potrebné ďalej spracovať nástrojom popísaným v ďalšom bode.

- PHP parser a SQL databáza sa nachádza v adresári GUI. Keďže neprehľadnosť štandardných výstupov všetkých známych balíkov je základným kameňom úrazu pri štúdiu ich vlastností, navrhol som jednoduchú databázu pozostávajúcu zo štyroch tabuliek. Tabuľky testov a ich nastavení, generátorov a ich nastavení, batérií a získaných výsledkov tvoria jadro grafického výstupu systému. Jediný spôsob, ako napĺňať údaje, je za pomoci PHP parsera, ktorý na vstupe zoberie *.res súbor a automaticky z neho naplní všetky tabuľky. Tento parser zvláda všetky výstupy vytvorené pomocou môjho systému pre balík TestU01. Akékoľvek iné výsledky z iných systémov treba naplniť ručne cez databázového klienta, prípadne vytvoriť ďalšie parsery pre iné balíky. NISTovký systém výstupov je pomerne komplikovaný, tak som ho neimplementoval z časových dôvodov. Na beh celého systému je potrebné mať nainštalované PHP+SQL+Apache.

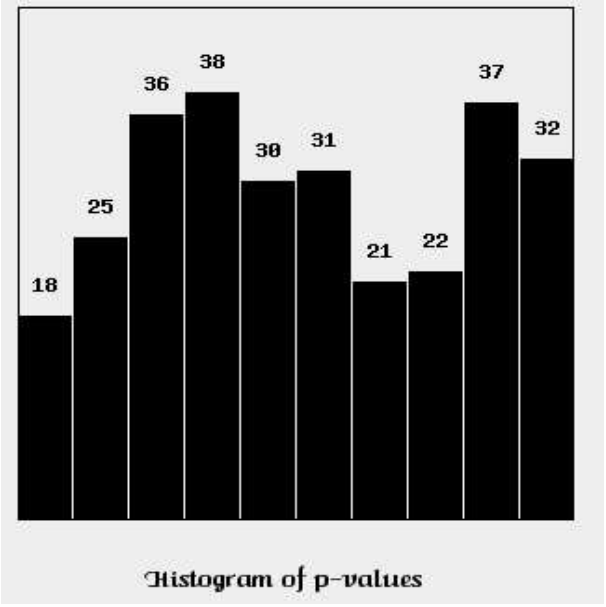
- Grafické rozhranie. Je veľmi jednoduchý prehľadný systém tabuliek vytvorený v PHP. V podstate je to len vyzualizačný nástroj, ktorý umožňuje ľahšie porovnávať rôzne testované balíky, ako aj výsledky pre generátory. Súčasťou je aj automatické generovanie histogramu a grafu s rozdelením p-hodnôt. Tieto obrázky sa vytvárajú buď pre všetky testy na nejakom generátore, alebo pre konkrétnu batériu testov na nejakom generátore. Slúžia ako skvelá vizualizačná pomôcka pri určovaní kvality generátorov, obzvlášť v prípadoch keď sa použije väčšie množstvo testov. Pre funkčné vykresľovanie obrázkov je potrebné mať nainštalované grafické knižnice pre PHP, ktoré sa tam štandartne nenachádzajú. Funkčný príklad, bez potreby inštalácie čohokoľvek som uverejnil na adrese http://www.e-providing.com/rngtesting/generators.php.

6.2 Obrazová príloha

Získané údaje, pre kvalitný generátor pseudonáhdných čísel.

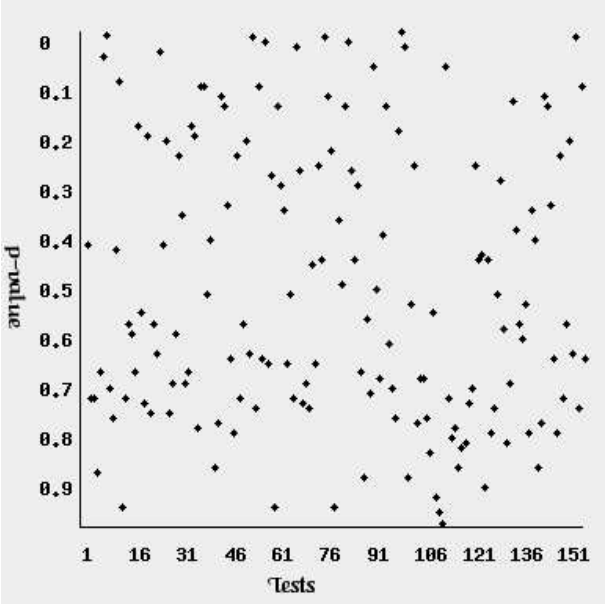


Graf rozdelenia p-hodnôt

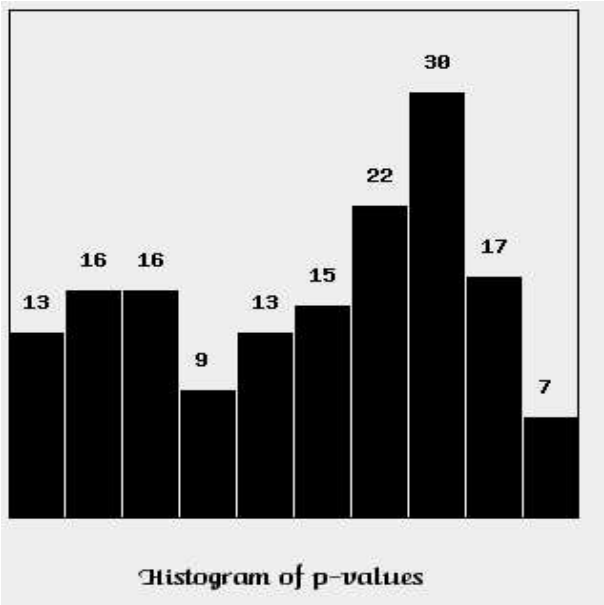


Histogram p-hodnôt.

Preverenie kvality testov na dátach vytvorených pomocou TRNG. Dáta boli stiahnuté z internetovej stránky www.randomnumber.org.

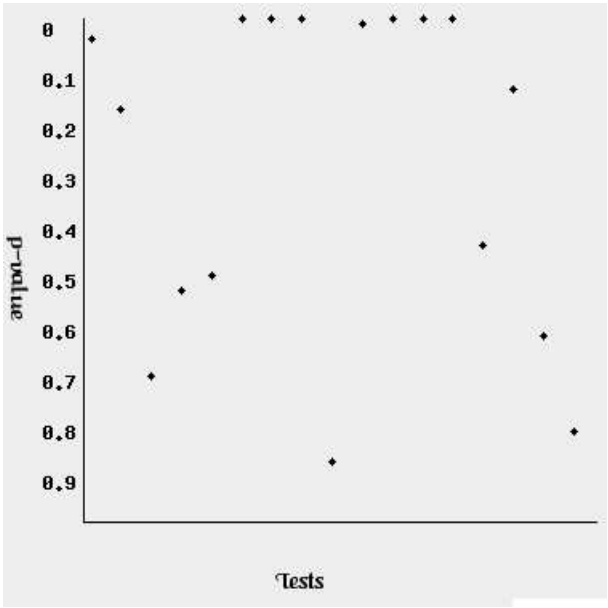


Graf rozdelenia p-hodnôt

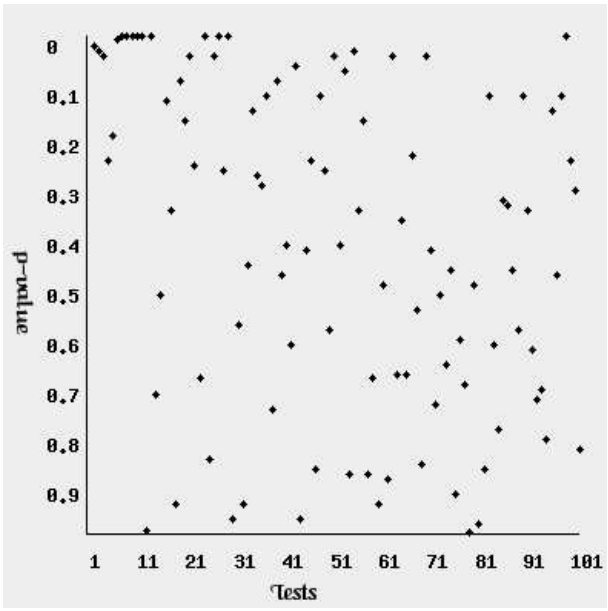


Histogram p-hodnôt.

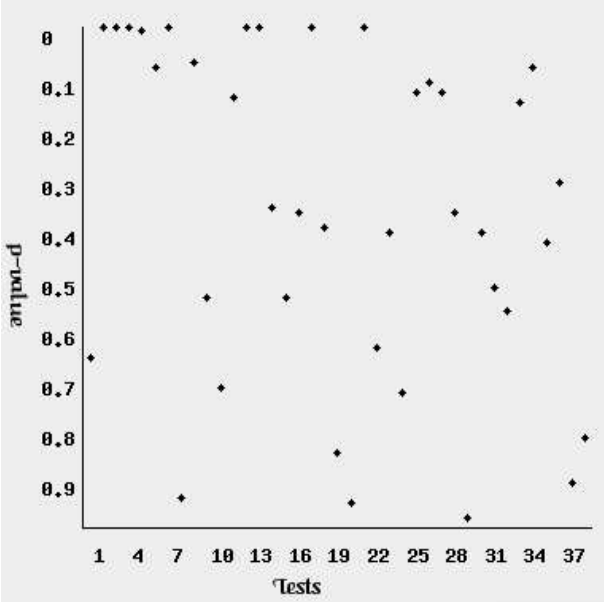
Porovnanie základných batérií. Testy prebehli na jednom z najlepšich LCG generátorov.



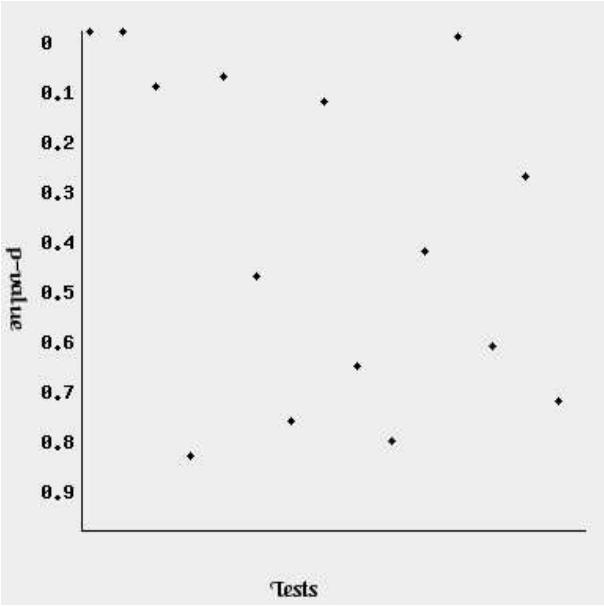
Graf rozdelenia p-hodnôt batérie Alphabit. Výrazná koncentrácia v oblasti 0 naznačuje, že veľa testov objavilo chyby generátora.



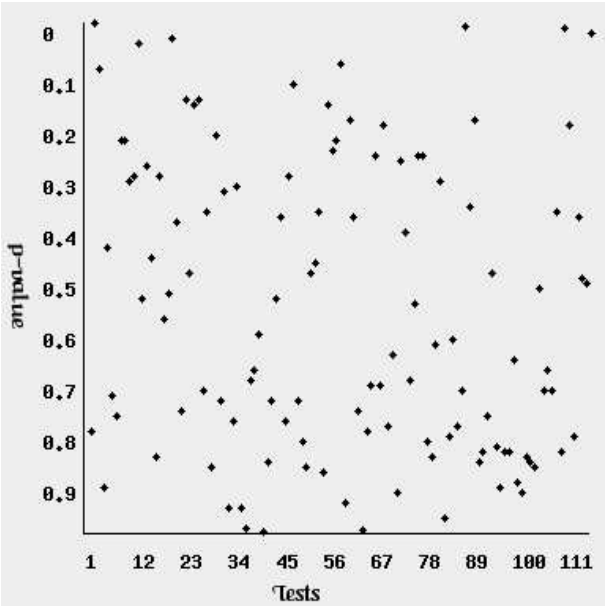
Batéria Blockalphabit. Tiež je zjavné, že veľa testov zlyhalo, čo naznačujú husté čierne miesta v oblasti bodu 0.



Graf rozdelenia p-hodnôt batérie Rabbit. Výsledok podobný ako v minulých prípadoch.



Batéria SmallCrush. Obsahuje nebitové testy. Narodiel od Diehard, ktorý je tiež nebitový, ojavil tiež nedostatky.



Batéria Diehard. Nejednoznačné výsledky.

Ako výsledky naznačujú, každá zo štvorice základných batérií testov objavila nedostatky. Jedine batéria Diehard, ktorá nie je orientovaná na bezpečnosť generátorov, nerozhodla jednoznačne, pričom tento jav je u ňej pomerne častý. Z toho dôvodu nie je možné brať batériu Diehard za vhodné meradlo kryptografickej bezpečnosti. Tato slabosť tejto batérie sa prejavila násobne, ale zas na druhej strane, jej testy vykázali isté schopnosti v prípadoch, kedy ostatné batérie neobjavili žiadne závislosti. Preto považujem batériu Diehard za vhodný doplnok k zvyšným testom, ale určite nie za smerodajnú. Tento generátor bol testovaný aj batériou od NISTu, a ten ho jednoznačným spôsobom vyhlásil za náhodný(obrázok nebolo možné urobiť vhľadom na to, že žiadny z NISTovských testov nebol napĺnený do môjho systému z technických dôvodov). Ani jeden zo všetkých druhoúrovňových testov nevybočil z očakávaných hodnôt. Toto je jeden z hlavných dôvodov, prečo považujem súčasné prístupy k testovaniu za nedostatočné.

6.3 CD-príloha

Základná štruktúra:

1. Adresár my_work, obsahuje celé moje programové riešenie, ako aj text tejto práce.

2. Adresár test_suites, obsahuje všetky tu rozoberané batérie testov spolu s dokumentáciou a spustiteľnými kódmi.

3. Adresár books obsahuje najznámejšie a na účely tejto práce použité dokumenty.

4. Adresár generators pozostáva z množstva súborov, ktoré súvisia s generátormi náhodných čísel.

Literatúra

[1] Atkinson, A. Tests of Pseudo-random Numbers. Applied Statistics. 29(2): 164-171., 1980.

[2] Beker, H. and F. Piper. Secure Speech Communications. Academic Press., 1985.

[3] Bennett, W. Scientific and Engineering Problem-Solving with The Computer. Prentice-Hall., 1976.

[4] Beth, T. and Z-D. Dai. On the Complexity of Pseudo-Random Sequences – or: If You Can Describe a Sequence It Can't be Random. Advances in Cryptology – EUROCRYPT '89. 533-543. Springer-Verlag., 1989.

[5] B.Ryabko, J.Astola. Universal Codes as a Basis for Time Series Testing(books/univcodes), 1998.

[6] B.Ya.Ryabko. Prediction of random sequences and universal coding,Problems of Inform.Transmission 24(2)(1988)87-96., 1988.

[7] Carroll, J. The binary derivative test: noise filter, crypto aid, and random-number seed selector. Simulation. 53(3): 129-135., 1989.

[8] C.E.Shannon. A mathematical theory of communication,Bell Sys.Tech.J., 27(1948)379-423,623-656, 1948.

[9] C.E.Shannon. Communication theory of secrecy systems, Bell Sys.Tech.J. 28(1948)656-715., 1948.

[10] Chaitin, G. Randomness and Mathematical Proof. Scientific American. 232(5): 47-52., 1975.

[11] Cochran ,W.G. Annals Math. Stat. 23 , 315-345, 1952.

[12] Compagner, A. Definitions of Randomness. American Journal of Physics. 59(8): 700-705., 1991.

[13] Compagner, A. The Hierarchy of Correlations in Random Binary Sequences. Journal of Statistical Physics. 63(5/6): 883-896. , 1991.

[14] Feldman, F. Fast Spectral Tests for Measuring Nonrandomness and the DES. Advances in Cryptology - CRYPTO 87. 243-254. Springer-Verlag., 1987.

[15] Foley, Louise. Analysis of an On-Line Random Number Generator, MSISS Project Report, April 2001, 2001.

[16] Gustafson, H, E. Dawson, and J. Golic. Randomness Measures Related to Subset Occurrence. Cryptography: Policy and Algorithms. 132-143. Springer-Verlag., 1995.

[17] Hernndez,J.C. ,Sierra, J.M. a Seznec, A. The SAC Test: A New Randomness Test , with Some Applications to PRNG Analysis . In ICCSA 2004,volume 3043of Lecture Notes in Computer Science, pages 960-967. Springer, , 2004.

[18] Hopkins, T. Algorithm AS 193. A Revised Algorithm for the Spectral Test. Applied Statistics. 32: 328-335., 1983.

[19] I.Csiszár, P.Shields. The consistency of the BIC Markov order estimation, Annals of Statistics,6(2000)1601-1619., 2000.

[20] Information Security Research Centre at Queensland. Crypt-XS Test Suite, University of Technology in Australia.

[21] Information Security Research Centre at Queensland University of Technology in Australia. Brief Description of the Crypt-X tests .

[22] Jansen, C. The Shortest Feedback Shift Register That Can Generate A Given Sequence. Advances in Cryptology – CRYPTO '89. 90-99. Springer-Verlag., 1989.

[23] J.Kieffer. Prediction and Information Theory,Preprint,1998., 1998.

[24] John Walker. Ent Test (books/ent.zip).

[25] Kak, S. Classification of Random Binary Sequences Using Walsh-Fourier Analysis. Proceedings of Applications of Walsh Functions. 74-77. Washington, D.C., 1971, 1971.

[26] Knuth, D. (1st ed. 1969.) The Art of Computer Programming. Volume 2: Seminumerical Algorithms. Addison-Wesley., 1981.

[27] Kolmogorov, A. Three Approaches to the Quantitative Definition of Information. Problems of Information Transmission. 1: 1-17, 1965.

[28] L'Ecuyer, P. Testing Random Number Generators. Proceedings of the 1992 Winter Simulation Conference. 305-313., 1992.

[29] LEcuyer,P., Simard,R. Beware of linear congruential generators with multipliers of the form $a = \pm 2^q \pm 2^r$. ACM Transactions on Mathematical Software 25, 3,367374., 1999.

[30] Maclaren, N. Cryptographic Pseudo-random Numbers in Simulation. Fast Software Encryption. Ross Anderson, ed., 185-190., 1993.

[31] Marsaglia, G. A Current View of Random Number Generators. Computer Science and Statistics: The Interface. 3-10. Elsevier Science., 1985.

[32] Marsaglia, G. and A. Zaman. Monkey Tests for Random Number Generators. Computers and Mathematics with Applications. 26(9): 1-10., 1993.

[33] Marsaglia,G. DIEHARD Statistical Tests, tests/DIEHARD , 1995.

[34] Marsaglia,G. , Tang, W.W. Somediffcult-to-pass tests of randomness ,Journal of Statistical Software, vol.7.(books/tuftest.pdf), 2002.

[35] Marsaglia,G., Tsay,L.-H. Matrices and the structure of random number sequences. Linear Algebra and its Applications 67, 147156., 1985.

[36] Mascagniand,M. ,Srinivasan,A. Algorithm 806: SPRNG: A scalable library for pseudorandom number generation. ACM Transactions on Mathematical Software,26:436-461,2000., 2000.

[37] Maurer, U. A Universal Statistical Test for Random Bit Generators. Advances in Cryptology – CRYPTO '90. 409-420. Springer-Verlag., 1990.

[38] Maurer, U. and J. Massey. Perfect Local Randomness in Pseudo-random Sequences. Advances in Cryptology – CRYPTO '89. 100-112. Springer-Verlag., 1989.

[39] Maurer,U.M. A universal statistical test for random bit generators. Journal of Cryptology 5, 2,89105., 1992.

[40] Menezes, A.J, vanOorschot, P.C. a Vanstone,S.A. Hand book of Applied Cryptography. CRC Press,(books/handapcr), 1997.

[41] Pearson, K. Philosophical Magazine, Series 5, 50, 157-1751, 1900.

[42] Phillips, J. Algorithm AS 48. Uncertainty Function for a Binary Sequence. Applied Statistics. 21: 97-99., 1972.

[43] Phillips, J. Algorithm AS 49. Autocorrelation Function for a Binary Sequence. Applied Statistics. 21: 100-103., 1972.

[44] PierreL'Ecuyer a Richard Simard. A Software Library in ANSI C for Empirical Testing of Random Number Generators, Departement d'Informatique et de Recherche Opérationnelle Université de Montréal,(books/testu01.pdf alebo tests/testu01) , 2005.

[45] PierreL'Ecuyer a Richard Simard. TestU01: A C Library for Empirical Testing of Random Number Generators,(books/ttestu01.pdf alebo tests/testu01) , 2005.

[46] R. B. DAgostinoand, M.S.Stephens,Eds.MarcelDekker. Tests for the uniform distribution. In Goodness-of-Fit Techniques, ,NewYorkandBasel,331366.), 1986.

[47] R.Cilibrasi, R.deWolf, P.M.B.Vitanyi . Algorithmic Clustering of Music,Computer Music Journal 28(4)(2004)49-67., 2004.

[48] Rukhin,A.L. Testing randomness: A suite of statistical procedures. Theory of Probability and Its Applications 45, 1,111132., 2001.

[49] Song-Ju,K., Umeno,K. a Hasegawa,A. Corrections of the NIST Statistical Test Suite for Randomness, (books/cnist.ps), 2004.

[50] Soto,J. Statistical Testing of Random Number Generators, National Institute of Standards and Technology, tests/NIST, 2000.

[51] sources.redhat.com/gsl. gsl library, .

[52] U.Maurer. Information-Theoretic Cryptography,in: Advances in Cryptology-CRYPTO'99, Lecture Notes in Computer Science, Springer-Verlag,1666(1999), 7-64, 1999.

[53] von Neumann, J. Various Techniques Used in Connection With Random Digits. John mon Neumann, Collected Works. A.H. Taub, ed., MacMillan., 1963.

[54] Wanders, H. On the Significance of Golomb's Randomness Postulates in Cryptography. Philips Journal of Research. 43(2): 185-222., 1988.

[55] www.fourmilab.ch. Ent Test Program .

[56] Yuen, C. Testing Random Number Generators by Walsh Transform. IEEE Transactions on Computers. C-26(4): 329-333., 1977.