KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Petras Bieliauskas

Funkcinių testinių rinkinių vėlinimo gedimams atrinkimo programinės įrangos sudarymas ir tyrimas

Magistro darbas

Darbo vadovas
prof. V. Jusas

KAUNAS, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Petras Bieliauskas

Funkcinių testinių rinkinių vėlinimo gedimams atrinkimo programinės įrangos sudarymas ir tyrimas

Magistro darbas

Recenzentas
doc. A. Lenkevičius
2010-05-26

Vadovas
prof. V. Jusas
2010-05-26

Atliko
IFM-4/2 gr. stud.
Petras Bieliauskas
2010-05-26

KAUNAS, 2010

Turinys

1 ĮVADAS....6
2 ANALITINĖ DALIS....8
2.1 Egzistuojantys sprendimai....8
2.1.1 TetraMAX® ATPG....8
2.1.2 TurboFault™....9
2.1.3 ATALANTA....10
2.2 Vėlinimo gedimų nustatymo modeliai....11
2.2.1 Perėjimo vėlinimo gedimo modelis....11
2.2.2 Kelio vėlinimo gedimo modelis....11
2.2.3 Modelių palyginimas....12
2.3 Kokybinės funkcinių testų charakteristikos....12
2.4 AntiRandom metodas....13
2.5 Įgyvendinimo problemos....13
2.5.1 Testų generavimo vykdymo laiko ir testų kokybės santykio problema....13
2.5.2 Grafinės vartotojo sąsajos problema....13
2.5.3 Funkcinio testo kokybės įvertinimo problema....14
2.5.4 Klaidų padengimo problema....14
2.5.5 Daugiavartojiškumo problema....14
2.6 Išvados....14
3 PROJEKTINĖ DALIS....15
3.1 Programų sistemos funkcijos....15
3.2 Realizavimo priemonės....17
3.3 Sistemos kontekstas....17
3.4 Veiklos kontekstas....18
3.5 Sistemos išskaidymas į paketus....18
3.5.1 Paketas „Klientas“....18
3.5.2 Paketas „Nuotolinė DB“....19
3.6 Veiklos padalinimas....20
3.7 Duomenų vaizdas....21
3.8 Realizuotas algoritmas....21
3.9 AntiRandom metodo realizacija ir pritaikymas funkcinių testų generavimui....23
3.10 Programinės įrangos eksperimentas....25
3.10.1 Schemų parsisiuntimas iš nuotolinės sistemos....25
3.10.2 Schemos prototipo pasirinkimas....26
3.10.3 Testinių rinkinių analizė....26
3.10.4 Algoritmo iteracijų keitimas....27
3.10.5 Generatoriaus pasirinkimas....27
3.10.6 Generavimo proceso rezultatai realiu laiku....28
3.10.7 Rezultatų perdavimas į serverį....28
3.10.8 Vartotojų valdymas....29
3.10.9 Schemų prototipų administravimas....30
3.10.10 Schemų tyrimai....31
3.10.11 Schemų tyrimų analizė....33
3.11 Išvados....36
4 Funkcinių testų generavimo tyrimas....37
4.1 Tyrimų vykdymo tvarka....37
4.2 Gautų rezultatų įvertinimo kriterijai....38
4.3 Tiriamos schemos....39
4.4 Eksperimentinio tyrimo rezultatai....40
4.4.1 S1196 schemos eksperimentinis tyrimas....40

4.4.2 S1238 schemos eksperimentinis tyrimas..... 43
4.4.3 S13207 schemos eksperimentinis tyrimas..... 46
4.4.4 S15850 schemos eksperimentinis tyrimas..... 49
4.4.5 S35932 schemos eksperimentinis tyrimas..... 52
4.4.6 S38417 schemos eksperimentinis tyrimas..... 55
4.5 Gautų rezultatų apibendrinimas..... 58
4.6 Išvados..... 60
5 IŠVADOS..... 61
6 LITERATŪRA..... 62
7 TERMINŲ IR SANTRUMPŲ ŽODYNAS..... 64

Santrauka

Dėl didėjančio integrinių schemų sudėtingumo ir darbinių dažnių vėlinimo gedimų nustatymas tampa svarbia schemų kūrimo dalimi. Programiniai schemų prototipai leidžia atlikti schemų testavimą ankstyvojoje stadijoje.

Šiame darbe yra pateikiama vėlinimo gedimų nustatymo metodų analizė ir jų palyginimas. Tyrimo objektu pasirinktas perėjimo gedimų modelis. Dokumente aprašomas AntiRandom metodo pritaikymo galimybės funkcinių testų generavimui. Taip pat yra trumpai apžvelgiami egzistuojantys sprendimai rinkoje.

Projektavimo skyriuje yra aprašoma suprojektuota ir realizuota sistema, kuri susideda iš dviejų posistemių: funkcinių testų generatoriaus bei rezultatų kaupimo ir analizės posistemės. Funkcinių testų generatoriuje realizuoti du atsitiktinio ir AntiRandom metodai.

Paskutinėje dokumento dalyje yra pateiktas atliktas eksperimentinis tyrimas su realizuota sistema. Taip pat yra pateikiami eksperimentinio tyrimo metu pasiekti rezultatai bei padarytos viso darbo išvados.

Summary

The increasing complexity of integrated circuits and operating frequency led delay fault identification to become an important part of the schemes development. Software prototypes allow to start testing phase at an early stage.

This work covers the delay fault detection method analysis and comparison. For the study is selected transition fault identification. The paper describes the AntiRandom method and customization possibilities for the functional test generation. There is also a brief overview of an existing solutions on the market.

The design section describes the designed and implemented system which consists of two subsystems: functional tests generator and results storage and analysis subsystem. Functional test generator has two random methods and customized AntiRandom method.

The last part of the document covers an experimental study for the created system. It consists of results of the experiments and conclusions of the whole work.

1 ĮVADAS

Tobulėjant puslaidininkių integrinių schemų gamybai sparčiai pradėjo augti šių schemų darbiniai dažniai. Dėl šios priežasties yra susiduriama su poreikiu nustatyti schemos vėlinimo gedimus. Vėlinimo gedimai gali atsirasti dėl įvairių priežasčių. Šiuos gedimus gali įtakoti tiek vidiniai, tiek išoriniai faktoriai. Prie vidinių faktorių priskiriami korozija, įsisenėjimas, nutrūkę ar perdegę kontaktai. Tuo tarpu išoriniai arba aplinkos faktoriai lemiantys gedimus gali būti temperatūra, slėgis, elektros energijos svyravimai ar net radiacija[1].

Vėlinimo gedimams nustatyti yra naudojami funkciniai testai. Funkciniai testai remiasi schemos vykdoma funkcija. Tam, kad būtų galima atlikti schemos testavimą lygiagrečiai su pačios schemos projektavimu yra sukuriamas schemos programinis prototipas pagal schemos specifikaciją. Šis prototipas imituoja schemos vykdomą funkciją [2]. Gamykliniai testai būna skirti konkrečiai schemos realizacijai, tuo tarpu funkcinis testas nėra susietas su konkrečia schemos realizacija. Tai leidžia sukurti universalius testus, kurie remiasi vien testuojamos schemos vykdoma funkcija ir neatsižvelgia į realizaciją. Funkciniai testai dažniausiai būna ilgi tam, kad būtų užtikrintas aukštas klaidų padengimas prie įvairių realizacijų. Tačiau funkcinis testas, kuris yra nesusietas su konkrečia realizacija yra pranašesnis tuo, jog leidžia atlikti testavimą ankstyvoje schemos kūrimo stadijoje, bei testuoti įvairias realizacijas.

Generuojant funkcinius testus yra keliamas tikslas pasiekti kuo aukštesnį klaidų padengimą bei sumažinti testų kiekį ir laiką reikalingą testavimui atlikti.

Atsitiktiniu būdu sugeneruoti testai neišnaudoja ankstesnių testų sukauptos informacijos. Todėl reikia generuoti tokius testus, kurie padengtų dar nepadengtą funkcionalumą. Atsitiktinis testų generavimas panaudojant ankstesnių testų sukauptą informaciją yra vadinamas „AntiRandom“ generavimu.

Nėra žinoma apie praktinį AntiRandom pritaikymą generuojant funkcinius testus, tačiau AntiRandom pagalba gaunami geresni rezultatai nei naudojant atsitiktinį generavimą, todėl vienas iš pagrindinių tikslų ir yra pabandyti pritaikyti AntiRandom funkcinių testų generavimui vėlinimo gedimams nustatyti bei ištirti šio metodo pasiekiamus rezultatus ir naudą vėlinimo gedimų nustatymui.

Darbo tikslas ir uždaviniai:

- Suprojektuoti, realizuoti ir ištirti sistemą funkcinių testų generavimui panaudojant AntiRandom generatorių
- Išanalizuoti taikymo sritį ir susipažinti su esamomis technologijomis bei metodikomis;

- Suprojektuoti ir sukurti programinę įrangą, skirtą funkcinių testų generavimui;
- Realizuoti generavimo algoritmą;
- Realizuoti du atsitiktinius ir AntiRandom generatorius generuojant funkcinius testus vėlinimo gedimų nustatymui;
- Suprojektuoti ir sukurti internetinę duomenų bazę tyrimų rezultatų kaupimui ir analizei.
- Atlikti tyrimą su skirtingais schemų prototipais panaudojant realizuotus generatorius;

2 ANALITINĖ DALIS

Skyriuje aprašomi jau egzistuojantys susiję sprendimai. Nagrinėjami vėlinimo gedimų testavimo metodai.

Taip pat yra nagrinėjama galimybė pritaikyti AntiRandom metodą funkcinių testų generavimui.

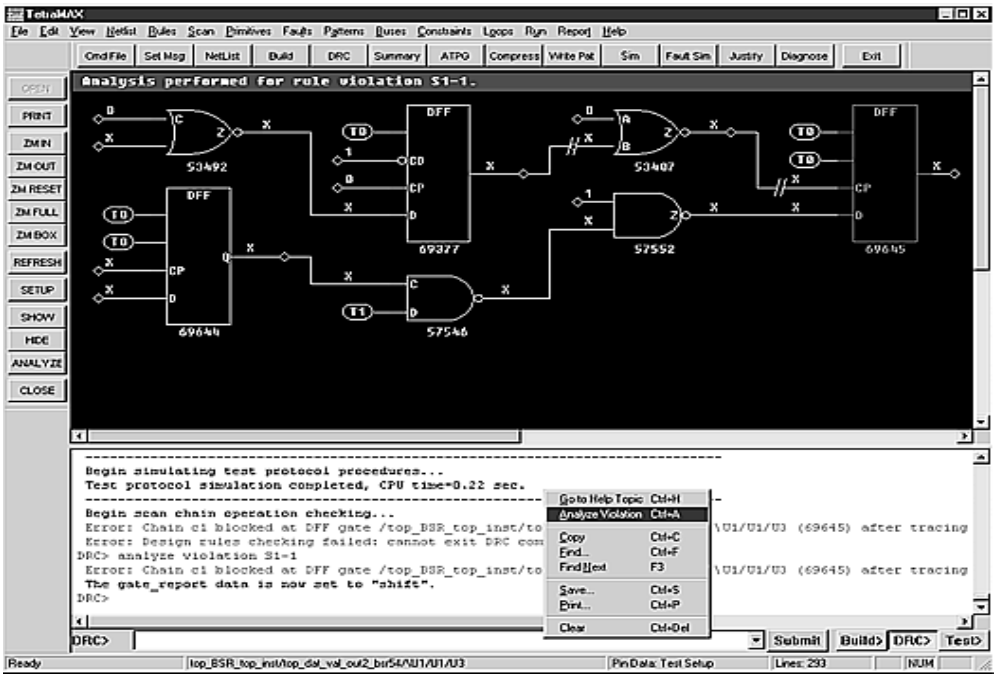
2.1 Egzistuojantys sprendimai

Projekto nagrinėjamoje srityje pasaulyje yra vos keletas tikrai stiprių komercinių produktų, kurių pagalba galima atlikti įvairiapusį schemos testavimą, įskaitant ir vėlinimo gedimų suradimą.

2.1.1 TetraMAX® ATPG

Vienas rimčiausių produktų rinkoje yra Synopsys kompanijos sukurtas įrankis TetraMAX® ATPG įrankis. Synopsis kompanija pirmauja pasaulyje pagal pagamintus elektronikos projektavimo įrankius (EDA) skirtus globaliajai elektronikos rinkai. Kompanija siūlo išskirtines projektavimo technologijas skirtas projektuoti sudėtingoms integrinėms schemoms ir elektroninėms sistemoms.

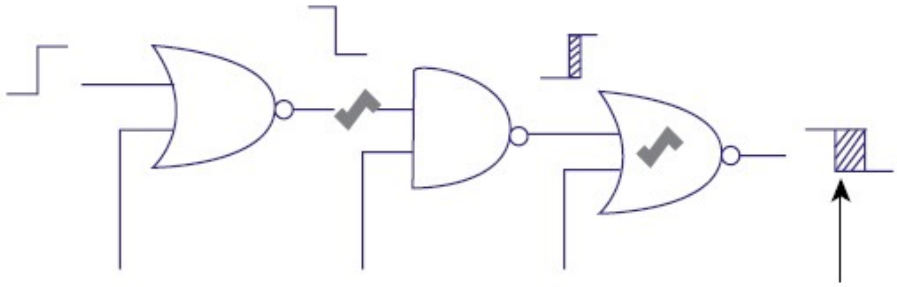
Šios kompanijos sukurtas produktas automatiškai generuoja aukštos kokybės gamyklinius testinius vektorius. Tai yra vienas iš nedaugelio produktų optimizuotų įvairiems testavimo modeliams. Taip pat tai yra bene vienintelis komercinis paketas, turintis galingus testavimo įrankius. Programos langas pavaizduotas paveikslėlyje 2.1.1.1. pav.



2.1.1.1. pav. TetraMAX programos langas

Pagrindinės produkto savybės:

- Greitas veikimas;
- Grafinė vartotojo sąsaja;
- Integruotas grafinis imitatoriaus įrankis;
- Testinių vektorių sudarymas pagal įvairius modelius;
- Turi integruotą gedimų imitatorių funkciniams vektoriams;
- Galimybė testuoti kritinius kelius 2.1.1.2. pav.;



2.1.1.2. pav. Kritinių kelių vėlinimo gedimų stebėjimas su TetraMAX

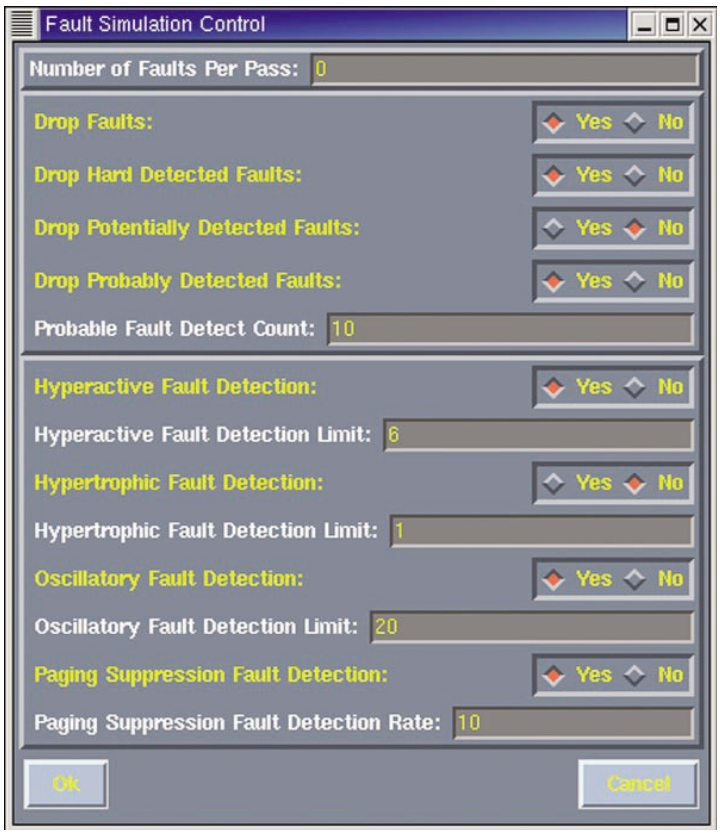
TetraMax įrankis minimizuoja testinius vektorius iki kompaktiškų, tai padeda sumažinti testų ciklą, reikalingą ištestuoti kiekvienam įrenginiui. Tai lemia testavimo sąnaudų mažinimą.

2.1.2 TurboFault™

Syntest kompanijos sukurtas įrankis padeda automatizuoti schemų testavimą. Kaip ir TetraMax, TurboFault™ palaiko daugybę klaidų modelių įskaitant ir perėjimo klaidų modelį. Programa kategorizuoja surastus gedimus į grupes: potencialios, sunkiai aptinkamos ir pan. TurboFault™ palaikomos funkcijos:

- „Stuck-at“ klaidų suradimas;
- Perėjimo gedimų suradimas, panaudojant „slow-to-raise“ ir „slow-to-fall“ modelius;
- Padeda optimizuoti įrenginius, pašalinant nereikalingas operacijas.

Programos langas pavaizduotas 2.1.2.1. pav.



2.1.2.1. pav. TurboFault programos langas

2.1.3 ATALANTA

Virdžinijos valstijos universitete sukurta nekomercinio pobūdžio programinė įranga skirta kombinacinių schemų ATPG generavimui. Ši programa pakeitė dar 1991 metais tame pačiame universitete sukurtą SOPRANO programą. ATALANTA programa valdoma per komandinę eilutę. Programą galima nemokamai parsisiųsti iš šios programinės įrangos kūrėjų svetainės. Internetinėje svetainėje taip pat yra patalpintas išsamus žinynas kaip reikia naudotis šia programa.

Pagrindinis programos tikslas yra generuoti testinius vektorius. Schemos duomenis programa paima iš failo, o sugeneruotus vektorius įrašo į rezultatų failą.

Programoje taip pat yra realizuota nemažai parinkčių, kurių pagalba galima valdyti testų generavimą. Viena iš tokių parinkčių būtų, testų suspaudimo algoritmas.

2.2 Vėlinimo gedimų nustatymo modeliai

Poskyryje pateikiami ir aprašomi vėlinimo gedimų nustatymo modeliai.

2.2.1 Perėjimo vėlinimo gedimo modelis

Perėjimo vėlinimo gedimo (angl. Transition Delay Fault TDF) efektas yra stebimas tada, kai į įėjimą paduodamas kylantis arba krintantis signalo frontas nepasiekia stebimo išėjimo per nustatytą laiko tarpą.

Perėjimo vėlinimo gedimai gali būti dviejų rūšių [7]:

- Lėtai kylančio fronto (*angl. slow-to-rise*);
- Lėtai krintančio fronto (*angl. slow-to-fall*).

Jeigu vėlinimo gedimas yra didelis, šio gedimo elgsena yra panaši į „užstrigusio 0 arba 1“ (*angl. stuck-at*) klaidą, taigi gali būti taikoma „stuck-at“ klaidų suradimo metodika.[17]. Testavimo metu yra naudojami du testiniai vektoriai. Pirmasis testinis vektorius atlieka pradinės būsenos nustatymo į tam tikrą reikšmę, o antrasis fiksuoja perėjimo pasikeitimus.

Lyginant su kitais modeliais, perėjimo vėlinimo gedimo modelis yra efektyvesnis didesniems vėlinimo gedimams aptikti, kurie gali atsirasti schemoje atsitiktine tvarka. Tuo tarpu, pavyzdžiui kelio vėlinimo gedimo modelis yra skirtas aptikti mažoms klaidoms konkrečiame schemos kelyje. Perėjimo vėlinimo gedimo modelis taip pat turi labai svarbų pranašumą prieš kitus modeliuos, tuo, jog galima įvertinti testo kokybę apskaičiuojant klaidų padengimą. Kelio vėlinimo gedimo modelis tokio parametro neturi [5].

2.2.2 Kelio vėlinimo gedimo modelis

Kelio vėlinimo modelio (angl. Path Delay Fault PDF) pagalba yra tikrinami nedideli, tačiau suminiai schemos vėlinimai pasiskirstę visame kelyje. Šie maži vėlinimai kuomet stebimi pavieniai, gedimo gali ir negeneruoti, tačiau juos visus susumavus gali atsitikti taip, kad schema normaliai funkcionuoti nebegalės. Kaip ir išplaukia iš pavadinimo, šis modelis yra skirtas surasti klaidas tam tikruose apibrėžtuose keliuose. Kadangi šiais laikais schemos yra labai didelės, dėl to ir kelių skaičius yra milžiniškas, todėl visų kelių patikrinimas tampa labai sudėtingas. Šis modelis dažniausiai yra naudojamas aptinkant klaidas kritiniuose schemos keliuose [7].

Kaip ir perėjimo vėlinimo gedimai, kelio vėlinimo gedimai taip pat gali būti kylančio arba krintančio fronto vėlinimo (*angl. slow-to-rise, slow-to-fall)*.

Kelio vėlinimo gedimai klasifikuojami į dvi grupes:

- „nepriklausomas“ (*angl. robust)*. Vėlinimo gedimo fiksavimas yra nepriklausomas nuo vėlinimų kituose keliuose.
- „priklausomas“ (*angl. non-robust*) . Vėlinimo gedimo fiksavimas yra priklausomas nuo vėlinimų kituose keliuose [16].

2.2.3 Modelių palyginimas

Perėjimo vėlinimo gedimų modelio ir kelio vėlinimo gedimo modelio palyginimas yra pateiktas 2.2.3.1. lentelėje [7].

2.2.3.1. lentelė Vėlinimo gedimų modelių palyginimas

Metodas	Privalumai	Trūkumai	Panaudojimas
Perėjimo vėlinimo	Tinkamas aptikti didelius gedimus; Paprastesnis testų generavimas; Linijinis klaidų kiekio didėjimas; „Geografiškai“ įvairus klaidų padengimas.	Gali neaptikti smulkių vėlinimo gedimų.	Platiems defektams padengti; Procesų stebėjimas.
Kelio vėlinimo	Tikrinamas kelias aiškiai apibrėžtas; Tinkamas smulkiems gedimams aptikti	Eksponentinis klaidų kiekio didėjimas	Procesų stebėjimas; I/O charakterizavimas.

2.3 Kokybinės funkcinių testų charakteristikos

Testų kokybiškumui nustatyti yra naudojamas taip vadinamas klaidų padengimo parametras FC (*angl. Fault coverage)*. Šis parametras parodo santykį surastų klaidų su visų galimų klaidų skaičiumi [1]. Praktikoje jau yra modelių kaip sukurti funkcinius testus, kurie aptinka daugiau nei 99% klaidų nepriklausomai nuo schemos realizacijos. Tačiau šie testai yra žymiai sudėtingesni, negu tie kurie yra pritaikyti konkrečiai schemos realizacijai [10].

Testuojant realizuotus algoritmus klaidų padengimas yra viena iš svarbiausių charakteristikų. Atskirų pavienių testų kokybiškumas taip pat vertinamas priklausomai nuo to, kaip jis įtakoja schemos signalo pasikeitimus [12].

Vėlinimas įrenginyje aptinkamas naudojant testinių rinkinių porą. Pirmasis rinkinys suformuoja pradines reikšmes, o antras – pokyčius schemos įėjimuose. Reakcija į pasikeitusį

įėjimą matuojame išėjime po apibrėžtos laiko trukmės. Jeigu išėjimuose signalai nespėja pasikeisti, tai indikuoja apie vėlinimo gedimą schemoje. Vėlinimo gedimų tikrinimui signalų pokyčius yra tikslinga perduoti kuo ilgesniais schemos keliais. Taip bus patikrintas didesnis kiekis galimų gedimų viena funkcinių testo rinkinių pora. Pačio funkcinio testo kokybei nustatyti yra būtina nustatyti koks įėjimo pokytis kokius išėjimus įtakoja [13].

2.4 AntiRandom metodas

Esminė AntiRandom prielaida yra pasirinkti naujus testinius vektorius, kurie būtų labiausiai nutolę nuo egzistuojančių testinių vektorių. Atstumo matas yra Haming'o arba Cartesian'o atstumas. Nauji testiniai rinkiniai yra pasirenkami tokie, kad išplėstų atstumą tarp vektorių.

Šis metodas turi ir trūkumų. Metodas reikalauja apskaičiuoti kiekvieno vektoriaus atstumą. Tai apsunkiną šio metodo panaudojimą ilgiems testams generuoti.

Šis metodas yra taikomas jau turint atsitiktinių vektorių aibę ir nėra žinoma apie jo panaudojimą funkciniams testams generuoti. Tai bus vienas iš pagrindinių tyrimo objektų viso darbo metu.

2.5 Įgyvendinimo problemos

Norint įgyvendinti projektą reikės išspręsti tam tikras su funkcinių testų generavimu susijusias problemas. Dauguma iš šių problemų šiuo metu neturi aiškios sprendimo strategijos, kas smarkiai apsunkina darbus. Tačiau iš kitos pusės tokia situacija ir išankstinio sprendimo neturėjimas suteikia prielaidas naujiems sprendimams atrasti, kurie vėliau gali būti išplėtoti vykdant funkcinių testų generavimą.

2.5.1 Testų generavimo vykdymo laiko ir testų kokybės santykio problema

Testų generavimo trukmė tiesiogiai priklausys nuo testuojamos schemos prototipo sudėtingumo. Todėl reikia minimizuoti sistemos vykdymo trukmę. Tai galima pasiekti atsirenkant kokybiškiausias poveikių vektorių poras iš tam tikro jų kiekio. Šiuo atveju reikia surasti kompromisą tarp testo kokybės ir vykdymo laiko. Kuo didesnį testinių vektorių porų kiekį pasirinksime, tuo ilgiau užtruks kokybiškiausios poros atrinkimas, kas tiesiogiai įtakos viso testavimo trukmę.

2.5.2 Grafinės vartotojo sąsajos problema

Tam, kad neatrodytų jog generavimo proceso metu programa „pakibo“ ir nereaguoja į vartotojo veiksmus, reikia pastoviai atnaujinti generavimo proceso parametrų išvedimą ekrane. Tai leis pastoviai matyti proceso eigą, bei leis suprasti, kad programa nėra „pakibusi“.

2.5.3 Funkcinio testo kokybės įvertinimo problema

Įvertinti funkcinio testo kokybę neturint konkrečios schemos realizacijos yra sudėtinga. Metodai leidžiantys gauti universalius funkcinius testus nepririštus nuo schemos realizacijos nėra optimalūs, kadangi testo ilgis didelis. Dėl šios priežastis funkcinis testas turi būti minimizuotas konkrečiai schemos realizacijai.

2.5.4 Klaidų padengimo problema

Norint pilnai ištestuoti schemą, reikią kad jos klaidų padengimas būtų 100%, tačiau praktikoje tai yra sunku pasiekti, ypač kuomet schema yra sudėtinga ir turi daug elementų.

2.5.5 Daugiavartojiškumo problema

Kadangi klientinė programa (lokali sistemos dalis) perdavinės duomenis iš įvairių vietų, reikia užtikrinti, kad nebus prarasti duomenys jų perdavimo metu.

2.6 Išvados

1. Funkciniai testai nėra susieti su schemos realizacija. Tai leidžia sukurti universalius testus, kurie remiasi vien testuojamos schemos vykdoma funkcija ir neatsižvelgia į realizaciją.
2. Funkciniai testai leidžia atlikti testavimą ankstyvoje schemos kūrimo stadijoje;
3. AntiRandom metodas parodė geresnius rezultatus nei atsitiktinis generavimas, todėl galima tikėtis pasiekti aukštesnių rezultatų šį metodą pritaikius funkcinių testų generavimui.

3 PROJEKTINĖ DALIS

Skyriuje yra aprašomos sistemos funkcijos, pasirinkti sprendimo metodai, realizavimo priemonės. Taip pat yra pateikiama sistemos architektūra bei vartotojo sąsajos.

3.1 Programų sistemos funkcijos

Suprojektuotos ir realizuotos programinės įrangos pagalba galima atlikti šias funkcijas:

- Schemų prototipo pasirinkimas. Sukurtos programinės įrangos pagalba galima integruoti programinius schemų prototipus. Prototipai gali būti įvairiausių sudėtingumų, tai padės atlikti sudėtingesnius testavimus ir gauti objektyvius eksperimentų rezultatus. Pagal atliktus eksperimentų rezultatus galima daryti tam tikras išvadas apie vieno ar kito algoritmo efektyvumą ieškant vėlinimo gedimų konkrečiose schemose.
- „Random“ generatorius. Sistemoje realizuotas „atsitiktinis“ (angl. *Random*) generatorius. Jo pagalba yra generuojamos pavienės 0 ir 1 sekos. Sugeneravus vieną reikšmę (0 arba 1), ji yra jungiama į bendrą testinį vektorių tol, kol yra pasiekiamas norimas testo ilgis.
- „Random 2“ generatorius. Sistemoje taip pat yra realizuotas „Random 2“ generatorius. Nuo generatoriaus „Random“ jis skiriasi tuo, jog čia yra sugeneruojamas didelis skaičius ir jis paverčiamas dvejetainiu. Generuojama tiek skaičių, kiek yra reikalinga testo ilgiui pasiekti.
- „AntiRandom“ generatorius. Realizuotas ir pritaikytas funkcinių testų generavimui metodas.
- Parametrų valdymas. Tam, kad būtų galima pasiekti didesnį naudojamų algoritmų efektyvumą sistemoje yra galimybė valdyti tam tikrus algoritmų parametrus, kurie turi tiesioginės įtakos eksperimentų rezultatams. Kuomet bus sukaupta pakankamai duomenų apie parametrų įtaką, bus galima daryti išvadas apie tai kokios parametrų reikšmes yra optimaliausia naudoti prie konkrečių apibrėžtų sąlygų.
- Generavimas. Generavimo proceso vykdymas. Realizuotas autonominis generavimo proceso vykdymą. Tai leidžia sutaupyti nemažą dalį laiko bei sumažinti sistemos palaikymo kaštus. Automatizuotas generavimas, pats didina generavimo iteracijas, kol pasiekia pabaigos požymį. Rezultatai automatiškai internetu yra perduodami į internetinę duomenų bazę. Taip pat yra numatyta galimybė išsiųsti SMS pranešimą, po generavimo proceso pabaigos.

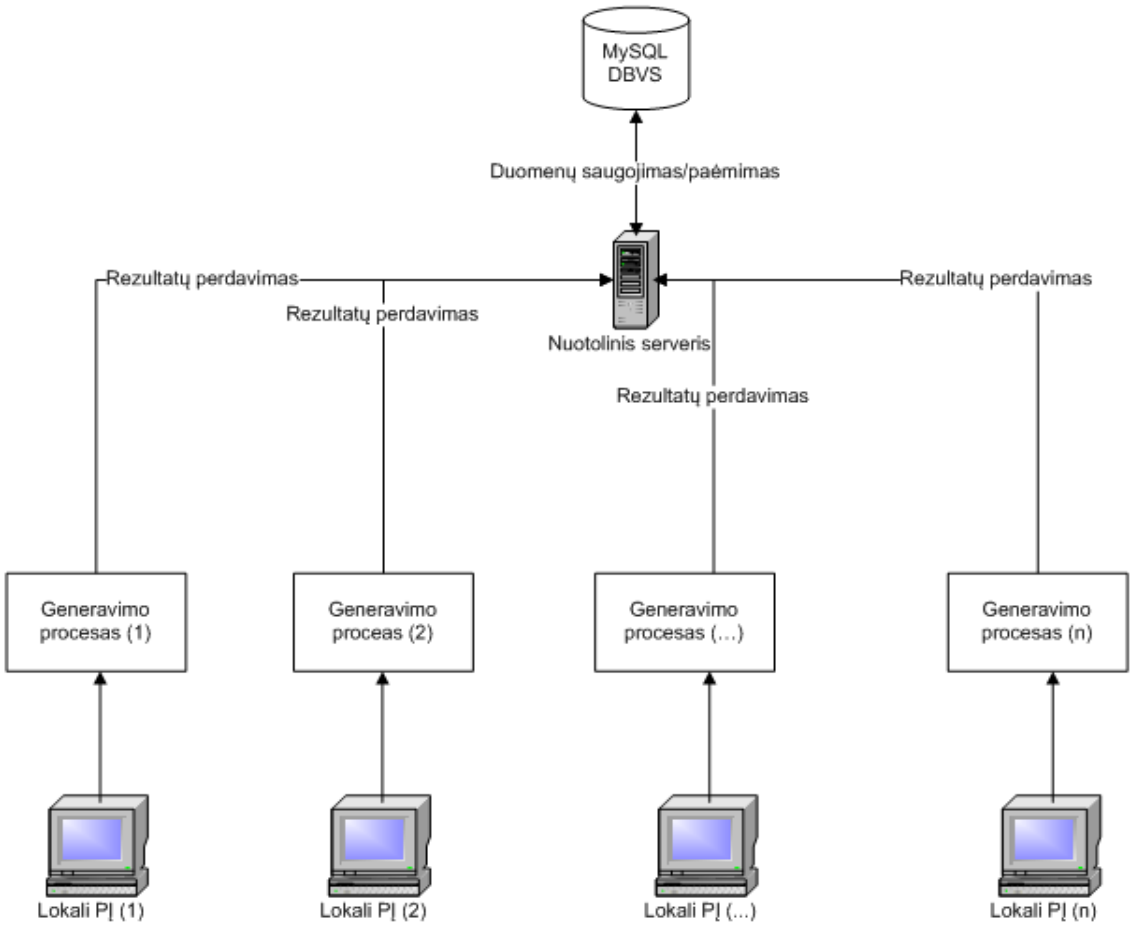
- Testinių rinkinių analizė. Sistemoje yra galimybė atlikti jau sugeneruotų testinių rinkinių analizę, nustatant kokį padengimą jie pasiekia. Pirmiausiai testiniai rinkiniai yra nuskaitomi iš failo ir tuomet yra atliekama jų analizė.
- Rezultatų perdavimas į DB. Atlikus testinių rinkinių generavimą yra galimybė perduoti gautus rezultatus ir naudotus parametrus į nuotolinę sistemą (duomenų bazę). Autentifikavus ir autorizavus siuntėjo vartotoją, persiunčiamas sugeneruotų testinių rinkinių failas, bei statistinė informacija. Ši informacija yra saugoma centralizuotoje duomenų bazėje neribotą laiką.
- Schemos prototipų atsisiuntimas. Nuotolinėje sistemoje yra kaupiami schemų prototipų H ir DLL failai, kuriuos vartotojai gali parsisiųsti į kompiuterį ir naudoti lokalioje sistemos versijoje.
- Rezultatų peržiūra. Prisijungęs prie sistemos, vartotojas gali matyti visus iš lokalios sistemos atsiųstus rezultatus. Yra pateikiama visa su rezultatais susijusi informacija, tokia kaip: atsiuntimo laikas, vykdymo trukmė, algoritmo parametrai, naudoto kompiuterio procesorius ir kiti.
- Rezultatų analizė. Internetinėje duomenų bazėje yra galimybė kaupti ir analizuoti gautų eksperimentų rezultatus. Pagal šiuos rezultatus daromos svarbios išvados apie algoritmus, jų kokybę, parametrų naudą ir pan. Rezultatai pateikiami grafiškai (diagramomis).
- Vartotojų kūrimas/valdymas. Nuotolinės sistemos administratorius, gali kurti bei valdyti sistemos vartotojus. Tik nuotolinėje sistemoje užregistruoti vartotojai gali perduoti duomenis į nuotolinę sistemą. Yra saugoma bendrinė vartotojo informacija: vardas, pavardė, el. pašto adresas, telefonas (šiuo numeriu bus siunčiamas SMS pranešimas apie generavimo proceso pabaigą).
- Naujų schemų prototipų registravimas. Nuotolinės sistemos administratorius turi galimybę kurti bei redaguoti schemas. Prie kiekvienos schemos yra nurodomas jos pavadinimas, tipas, bei įkeliami DLL bei H failai. Perduoti iš lokalios sistemos į nuotolinę sistemą rezultatus galima tik su tomis schemomis, kurios yra užregistruotos sistemoje.

3.2 Realizavimo priemonės

Kadangi lokali arba klientinė sistemos dalis turės veikti Windows tipo operacinėje sistemoje, todėl ši dalis yra realizuota C++ programavimo kalba naudojant Microsoft Visual C++ programavimo aplinką. Tuo tarpu nuotolinės sistemos kūrimui, kuri yra patalpinta nutolusiame serveryje, pasirinkta internetinių sistemų kūrimo programavimo kalba PHP, bei MySQL DBVS. Abu šie produktai yra atvirojo kodo.

3.3 Sistemos kontekstas

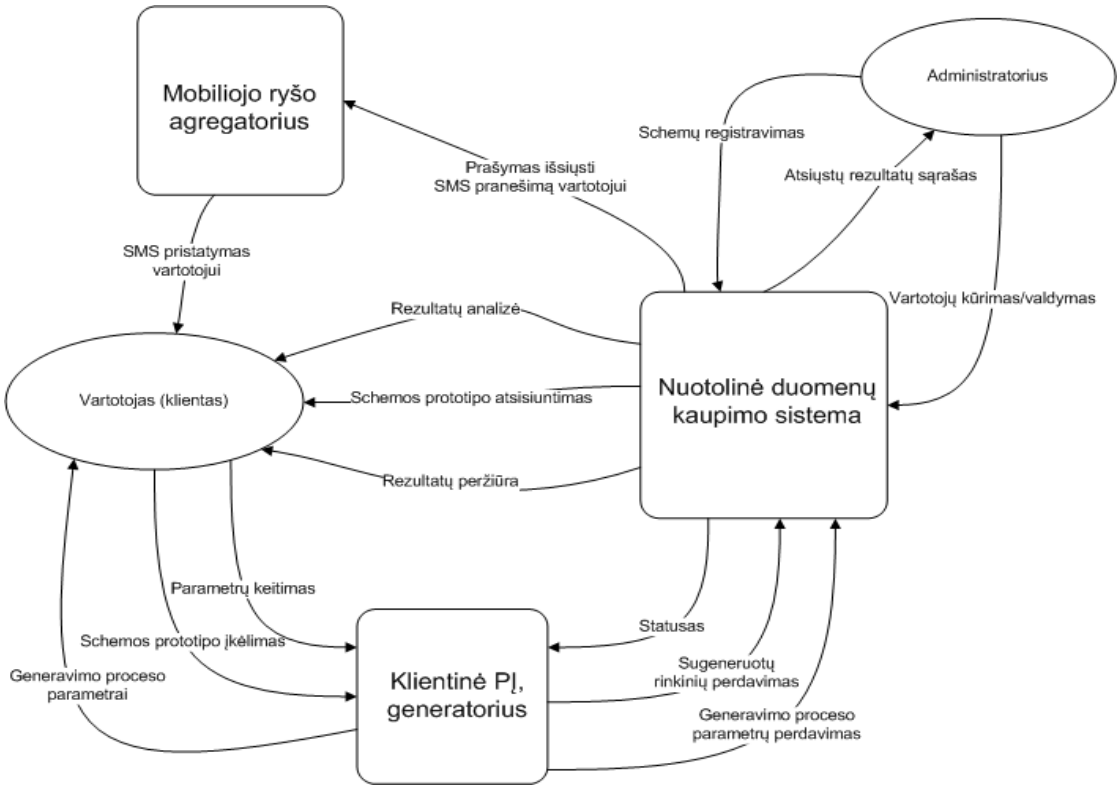
Visa sistema yra sudaryta iš dviejų posistemių: lokalios ir nuotolinės. Lokalios sistemos programinė įranga diegiama vartotojo kompiuteryje, kuriame bus vykdomas testinių rinkinių generavimas. Nuotolinė sistema yra patalpinta serveryje ir gali aptarnauti gaunamus duomenis iš įvairių vietų, nepriklausomai nuo to, kurioje vietoje yra lokali programinė įranga. 3.3.1. pav.



3.3.1. pav. Sistemos kontekstas

3.4 Veiklos kontekstas

Veiklos kontekstas pateiktas 3.4.1. pav.



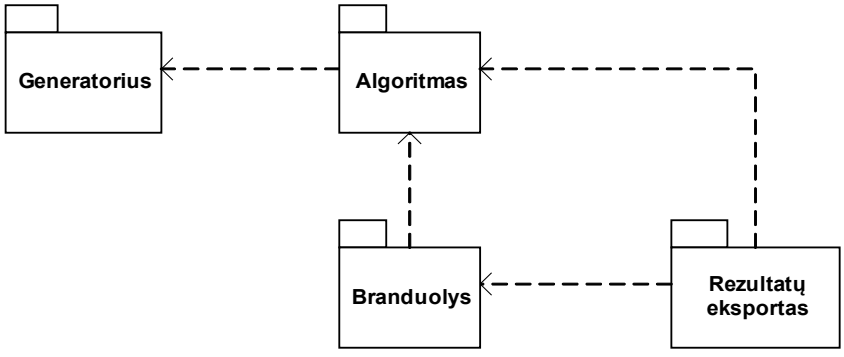
3.4.1. pav. Veiklos kontekstas

3.5 Sistemos išskaidymas į paketus

Sistema yra išskaidyta į du paketus: „Klientas“ ir „Nuotolinė DB“.

3.5.1 Paketas „Klientas“

Paketas atitinka sistemos dalį, kuri yra vartotojo kompiuteryje, kitaip dar vadinama klientu. Tai yra lokali versija, kurią galima instaliuoti bet kuriame Windows operacinę sistemą turinčiame kompiuteryje. Šio paketo pagalba yra vykdomas schemų prototipų testinių rinkinių generavimas, analizė bei rezultatų persiuntimas į centralizuotą nuotolinę duomenų bazę. Šio paketo suskirstymas į smulkesnius paketus pateiktas 3.5.1.1. pav.

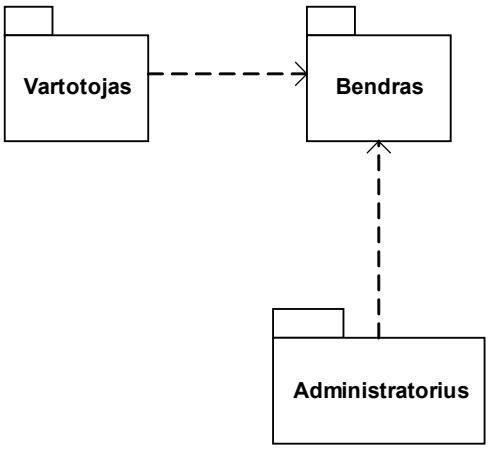


3.5.1.1. pav. Paketas „Klientas“

- Branduolys. Tai yra pagrindinis klientinės dalies paketas organizuojantis visą darbą. Šis paketas yra atsakingas už schemų prototipų pasirinkimą, užkrovimą, parametrų valdymą, generavimo proceso inicializavimą bei gautų rezultatų persiuntimą į nuotolinę duomenų bazę.
- Algoritmas. Paketas yra atsakingas už testinių rinkinių generavimo proceso organizavimą ir valdymą.
- Rezultatų eksportas. Paketas yra skirtas sugeneruotų testinių rinkinių bei generavimo statistikos perdavimui į nuotolinę duomenų bazę. Paketo pagalba yra patikrinamas ryšys su nuotoline duomenų baze. Testinių rinkinių failas yra perduodamas per FTP, tuo tarpu parametrai yra siunčiami HTTP POST metodu.
- Generatorius. Paketas yra atsakingas už testinių vektorių generavimą. Testiniai vektoriai generuojami trimis būdais, tai gali būti pilnai atsitiktiniai Random ir Random2 bei atvirkščiai atsitiktinis (AntiRandom). Paketas yra realizuotas taip, kad ateityje galima nesunkiai įtraukti naujus generatorius.

3.5.2 Paketas „Nuotolinė DB“

Paketas atitinka nutolusią sistemos dalį, kuri yra serveryje ir gali aptarnauti daug klientinių programų. Ši dalis yra centralizuota testinių rinkinių generavimo duomenų bazė. Tik nuotolinėje duomenų bazėje registruoti vartotojai gali atsiųsti generavimo rezultatus į serverį. Sistemoje matoma atsiųstų rezultatų istorija, galima parsisiųsti schemų prototipus, bei matyti rezultatų grafinį atvaizdavimą diagramomis. Šio paketo suskirstymas į smulkesnius paketus pateiktas 3.5.2.1. pav.



3.5.2.1. pav. Paketas „Nuotolinė DB“

- Administratorius. Šis paketas yra skirtas nuotolinės sistemos administravimui. Į paketą įeina vartotojų kūrimas bei valdymas, naujų schemų registravimas ir schemų prototipų įkėlimas. Tik administratoriaus užregistruoti vartotojai gali atsiųsti duomenis iš klientinės programos.
- Vartotojas. Šis paketas yra skirtas nuotolinės sistemos vartotojams. Vartotojai gali peržiūrėti ir pakoreguoti savo atsiųstus generavimo proceso rezultatus, bei analizuoti rezultatus pagal sistemos sugeneruotus grafikus.
- Bendras. Pagrindinis nuotolinės posistemės paketas organizuojantis šios posistemės darbą.

3.6 Veiklos padalinimas

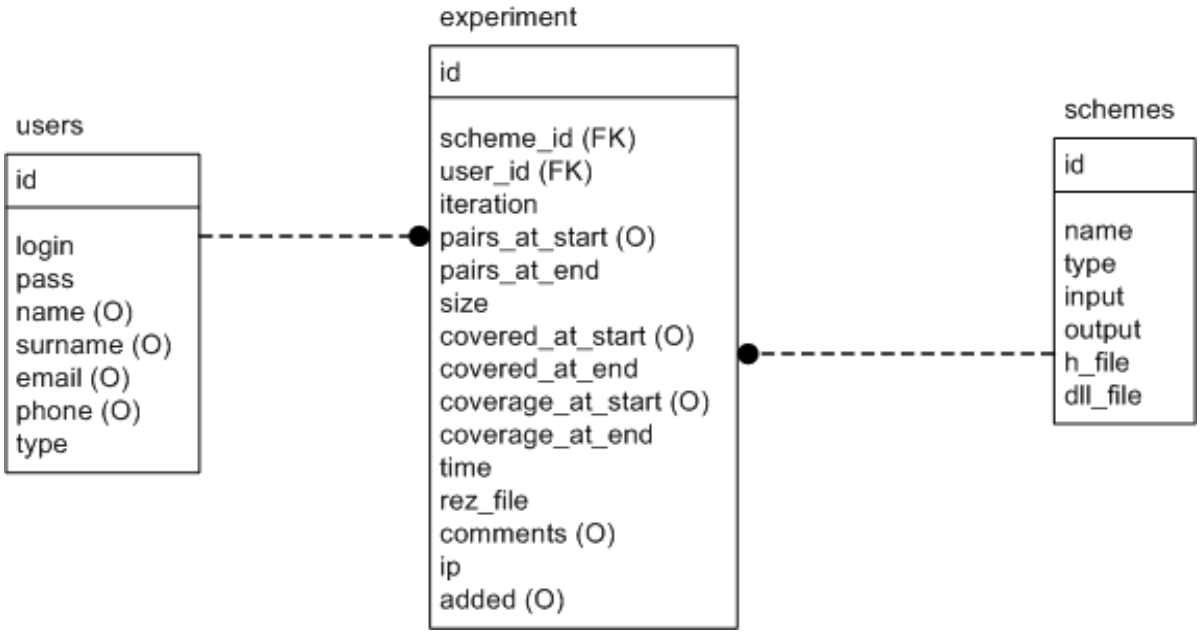
Veiklos padalinimas pateiktas 3.6.1. lentelėje

3.6.1. lentelė Veiklos padalinimas

Eil. Nr.	**Įvykio pavadinimas**	**Įeinantys/išeinantys informacijos srautai**
1	Schemos parsisiuntimas iš nutolusios sistemos į lokalų kompiuterį.	Schemos DLL failas (in)
2.	Schemos prototipo įkėlimas	Schemos DLL failas (out)
3.	Generavimo proceso parametrų stebėjimas	Vykdymo trukmė, sugeneruotos poros, padengimas (in)
4.	Parametrų keitimas	Iteracijų kiekis (out)
5.	Rezultatų peržiūrėjimas	Testinių rinkinių failas, generavimo proceso statistiniai duomenys (in)
6.	Rezultatų analizė	Grafikai (in)
7.	Schemos registravimas	Schemos pavadinimas, schemos grupė, H failas, DLL failas (out)
8.	Vartotojų kūrimas	Vardas, pavardė, prisijungimo vardas, slaptažodis, telefonas, el. pašto adresas (out)
9.	Atsiųstų rezultatų peržiūrėjimas	Testinių rinkinių failas, generavimo proceso statistiniai duomenys (in)
10.	Prašymas išsiųsti SMS pranešimą	Telefono numeris, prisijungimo vardas, slaptažodis, žinutės tekstas (out)
11.	SMS gavimas	Žinutės tekstas (in)
12.	Susijungimo su nuotoline sistema statuso gavimas	Statusas (in)
13.	Sugeneruotų rinkinių perdavimas	Tekstinis failas (out)
14.	Generavimo proceso parametrų perdavimas	Parametrai (out)

3.7 Duomenų vaizdas

Duomenų bazės modelis pateiktas 3.7.1. pav.



3.7.1. pav. Duomenų bazės modelis

Duomenų bazės modelyje esančių esybių aprašymai pateikti 3.7.1. lentelėje.

3.7.1. lentelė. Duomenų bazės modelio esybės

Esybė	**Aprašymas**
users	Saugoma informacija apie sistemos vartotojus. Vartotojai gali būti dviejų tipų: administratoriai ir klientai.
schemes	Saugoma informacija apie schemas ir jų prototipus.
experiment	Saugoma informacija apie iš klientinių programų atsiųstus generavimo rezultatus.

3.8 Realizuotas algoritmas

Pramoninėje gamyboje yra naudojamos trys populiariausios skleidimo architektūros:

- išplėstas (angl. Enhanced);
- užkrovimas su poslinkiu (angl. Skewed-load arba load-on-shift);
- funkcinis patvirtinimas (angl. Broadside arba functional justification).

Naudojant išplėstą skleidimą, trigeriai turi galimybę saugoti du individualius testinius rinkinius, tuo tarpu kitos architektūros leidžia saugoti tik vieną. Išplėstas skleidimas leidžia bet kokių dviejų testinių rinkinių taikymą. Užkrovimo su poslinkiu metu antrasis testinis rinkinys yra gaunamas atliekant poslinkį pirmajame testiniame rinkinyje. Funkcinio

patvirtinimo metu antrasis rinkinys gaunamas išsaugojant pirmojo rinkinio būseną ir naudojant ją antrajam rinkiniui. Techninės įrangos apkrovimai susiję su skleidimu riboja išplėstos architektūros panaudojimą. Užkrovimas su poslinkiu ir funkcinis patvirtinimas turi trūkumą, kadangi gali neaptikti visų gedimų, net jeigu šie gedimai gali būti aptinkami naudojant išplėstą skleidimą. Išplėstas skleidimas turėtų būti naudojamas kuomet yra siekiama pagerinti naujos architektūros našumą arba kuomet yra reikalingas aukštas patikimumo laipsnis.

Vėlinimo gedimų funkcinių testų generavimas priklauso nuo aptikimo matricos $||X||_{2nx4m}$ čia n – įėjimų kiekis, m – išėjimų kiekis. Matricoje kiekvienas įėjimas atstovauja dvi eilutes ir kiekvienas išėjimas atstovauja keturis stulpelius. Dviejų įėjimo eilučių susikirtimas su pirmais dviem išėjimo stulpeliais suformuoja keturis elementus, bei yra naudojamas pažymėti vėlinimo gedimų perdavimo tiesioginę įtaką. Tam, kad pritaikyti šią matricą vėlinimo gedimų perdavimui žymėti nuosekliose schemose įėjimų skaičius didinamas trigerių išėjimų skaičiumi T ir įėjimų skaičius yra didinamas trigerių įėjimų skaičiumi T. Ankstesnės būsenos bitai yra naudojami kaip pirminiai įėjimai, tuo tarpu sekančios būsenos bitai naudojami kaip išėjimai. Aptikimo matricai X priskirsime tokią formą: $|X||_{2(n+T)x2(m+T)}$. Panaikinsime stulpelius naudojamus netiesioginės įtakos žymėjimui. Bus naudojamos tos pačios celės, tik su skirtingomis reikšmėmis, tam kad pažymėti tiesioginę ir netiesioginę įtaką.

Priklausomai nuo tiriamo generatoriaus (Random, Random2 ar AntiRandom) kinta random(0, 1) funkcija. Matricoje X poveikiai gali įgyti keturias reikšmes. 4 ir 3 naudojami pažymėti tiesioginę (robust) įtaką, 2 ir 1 netiesioginę (non-robust) įtaką. 4 nurodo dvipusį aktyvumą esant tiesioginei įtakai, 2 nurodo dvipusį aktyvumą esant netiesioginei įtakai.

Algoritme buvo realizuota išplėsta skleidimo architektūra.

1. $V = ||V_b=<v^b_1,v^b_{2,}\ ...,v^b_{(n+T)}>||_{np}$;
2. $X = ||x_{i,j} := 0||_{2(n+T)\ x\ 2(m+T)}$; h := 1;
3. Do s := 1,3, …, L;
4. is := false;
5. If (s > np) Then
6. P1(n + T) ← random(0, 1);
7. Case (mode)
8. enhanced : P2(n + T) ← random(0, 1);
9. shift : Do i := 1,2,...,n+T-1; $p2_{i+1} := p1_i$; End Do; $p2_1 := p1_{n+T}$;
10. broadside : P2(n) ← random(0,1); R1 := function(P1); P2[1↔n+T] := P2[1↔n] || R1[n+1↔n+T];
11. End Case;
12. Else
13. P1 := $(v^S_1,v^S_2,....,v^S_{(n+T)})$; P2 := $(v^{S+1}_1,v^{S+1}_2,....,v^{S+1}_{(n+T)})$;
14. End If;
15. R1 := function(P1); R2 := function(P2);
16. Do i := 1,2,...,n+T;
17. If ($p^1_i \neq p^2_i$) Then
18. P3 := P1; $p^3_i := p^2_i$;
19. R3 := function(P3);

```
20.        P4 := P2; p4i := p1i;
21.        R4 := function(P4);
22.        d := 1 - p2i ;
23.        Do j := 1,2,...,m+T;
24.          c := 1 - r2j ;
25.          If (r2j ≠ r4j) Then
26.            If (r1j ≠ r2j) Then
27.              If (r1j ≠ r3j) Then
28.                If (x2i-d,2j-c < 4) Then x2i-d,2j-c := 4; is := true; End If;
29.              Else
30.                If (x2i-d,2j-c < 3) Then x2i-d,2j-c:= 3; is := true; End If;
31.              End If;
32.            Else
33.              If (r1j ≠ r3j ) Then
34.                If (x2i-d,2j-c < 2) Then x2i-d,2j-c := 2; is := true; End If;
35.              Else
36.                If (x2i-d,2j-c < 1) Then x2i-d,2j-c := 1; is := true; End If;
37.              End If;
38.            End If;
39.          End If;
40.        End Do;
41.      End If;
42.    End Do;
43.    If (is = true) Then V1h := P1; V1h+1 := P2; h := h+2; End If;
44. End Do;
45. h := h-1:
46. Do i := 1,3,...,h-1; Vi := V1h-i; Vi+1 := V1h-i+1;  End Do;
```

3.9 AntiRandom metodo realizacija ir pritaikymas funkcinių testų generavimui

AntiRandom testinis vektorius yra apibrėžtas kaip vektorius maksimaliai nutolęs nuo visų ankstesnių vektorių. Hamming'o atstumas yra apibrėžiamas kaip skaičius besiskiriančių bitų tarp vektorių. Cartesian'o atstumas tarp dviejų vektorių A ir B yra apibrėžiamas taip:

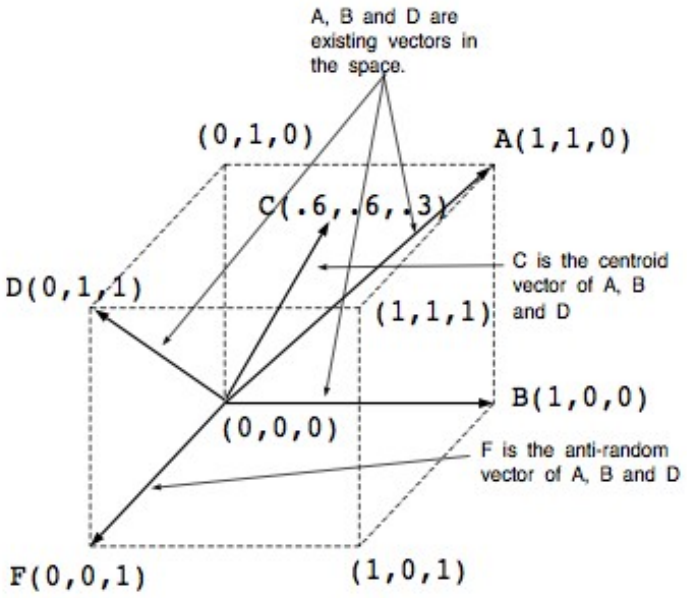
$$\sqrt[2]{(A_M - B_M)^2 + (A_{M-1} - B_{M-1})^2 + ... + (A_1 - B_1)^2}$$

Pavyzdžiui, dviejų vektorių A=[1, 1, 0] ir B=[0, 1, 1] Hamming'o atstumas būtų 2 (skiriasi pirmas ir trečias bitas). Tuo tarpu Cartesian'o atstumas būtų:

$$\sqrt{(0-1)^2 + (1-1)^2 + (1-0)^2} = \sqrt{2}$$

Išsamus AntiRandom testų generavimas paskaičiuoja Hamming'o ir Cartesian'o atstumus visiems galimiems testiniams vektoriams. Greitas AntiRandom (angl. Fast AntiRandom (FAR)) sugeneruoja naujus vektorius centralizuodamas visus egzistuojančius testinius vektorius į vieną testinį vektorių. Centroidinis vektorius yra jų vidurkis. Tuomet FAR suranda visus galimus vektorius, kurie yra ortogonalūs centralizuotam vektoriui. Yra surandamas vektorius, kuris yra toliausiai nutolęs nuo centroidinio vektoriaus.

3.9.1. pav. pavaizduotas subalansuotos erdvės konceptas. Vektoriai A, B ir D reprezentuoja tris taškus erdvėje. Vektorius C yra centroidinis vektorius A, B ir D vektoriams. Vektorius F yra ortogonalus vektorius centroidiniam C vektoriui, turintis maksimalų atstumą nuo A, B ir D vektorių.



3.9.1. pav. Centroidinis (C) ir AntiRandom (F) vektoriai.

Pabandykime pavyzdžiu iliustruoti šešių bitų AntiRandom vektoriaus generavimą pagal esamus vektorius. Tarkime turime šešių dimensijų erdvę (M = 6) su penkiais įėjimo vektoriais (N=5). Egzistuojantys testiniai vektoriai:

A = [1, 1, ,1 0, 1, 0]

B = [0, 1, 0, 1, 1, 0]

C = [1, 0, 0, 0, 1, 1]

D = [0, 1, 1, 0, 0, 0]

E = [1, 0, 0, 0, 0, 1]

Pirmas žingsnis: centralizavimas. Surandamas vektorius X, kuris yra ankstesnių vektorių (A-E) vidurkis: X = (A+B+C+D+E)/5 = [3, 3, 2, 1, 3, 2] / 5 = [0.6, 0.6, 0.4. 0.2, 0.6, -.4]. Atitinkamai dvejetainis vektorius būtų X = [1, 1, 0, 0, 1, 0].

Antras žingsnis: ortogonalių vektorių apskaičiavimas. Randami visi ortogonalūs vektoriui X vektoriai:

X1 = [0, 0, 1, 1, 0, 1]

X2 = [0, 0, 0, 1, 0, 1]

X3 = [0, 0, 1, 0, 0, 1]

X4 = [0, 0, 1, 1, 0, 0]

X5 = [0, 0, 0, 0, 0, 1]

X6 = [0, 0, 1, 0, 0, 0]

X7 = [0, 0, 0, 1, 0, 0]

X8 = [0, 0, 0, 0, 0, 0]

Trečias žingsnis: maksimalaus atstumo apskaičiavimas. Didžiausią atstumą turi X1 vektorius.

AntiRandom pritaikymas funkcinių testų generavimo algoritme buvo vykdomas tokiais žingsniais:

1. Pirmą kartą testinis vektorius P1 generuojamas atsitiktiniu būdu (Random arba Random2);
2. P2 visada generuojamas atsitiktiniu būdu;
3. Gauti testiniai vektoriai kaupiami aibėje;
4. Pagal turimą aibę naujas P1 yra apskaičiuojamas pagal AntiRandom metodą;
5. Kartojami 2 - 4 žingsniai.

3.10 Programinės įrangos eksperimentas

3.10.1 Schemų parsisiuntimas iš nuotolinės sistemos

Turint vartotojo vardą ir slaptažodį galima prisijungti prie sistemos. Jeigu slaptažodžio nėra, reikėtų kreiptis į sistemos administratorių dėl naujo vartotojo sukūrimo.

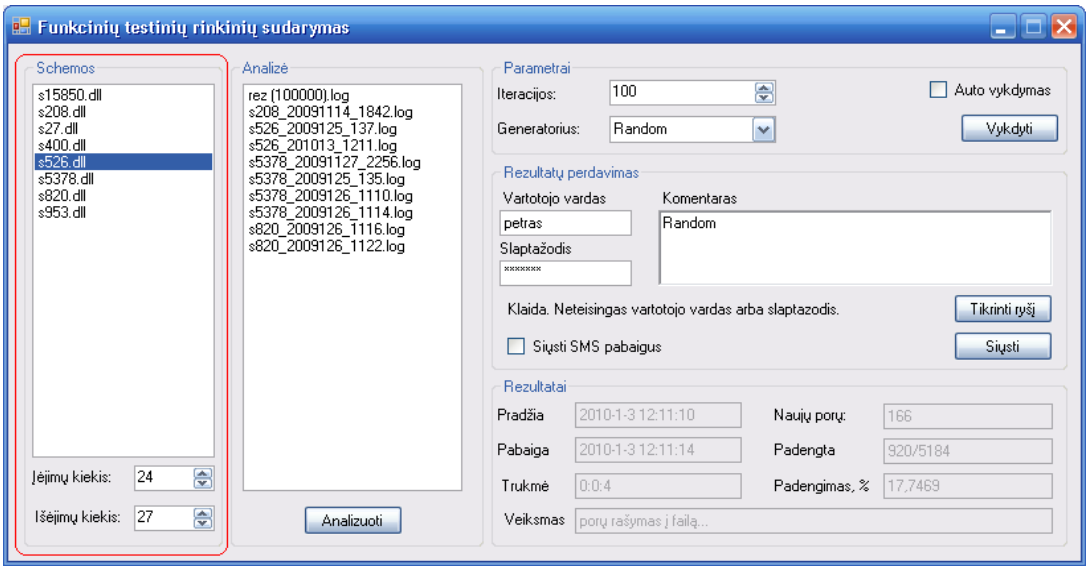
Prisijungus prie internetinės duomenų bazės, matomas visų schemų sąrašas. Galima parsisiųsti schemas DLL arba H failų pavidalu 3.10.1.1. pav.

Nr.	Schema	Grupė	Įėjimai	Išėjimai	H failas	DLL failas	Tyrimai
1	s27	S	7	4	s27.h	s27.dll	0
2	s208	S	19	10	s208.h	s208.dll	6
3	s820	S	23	24	s820.h	s820.dll	0
4	s382	S	24	27	s382.h	s382.dll	7
5	s400	S	24	27	s400.h	s400.dll	0
6	s526	S	24	27	s526.h	s526.dll	5
7	s420	S	35	18	s420.h	s420.dll	0

3.10.1.1 Schemų prototipų parsisiuntimas

3.10.2 Schemos prototipo pasirinkimas

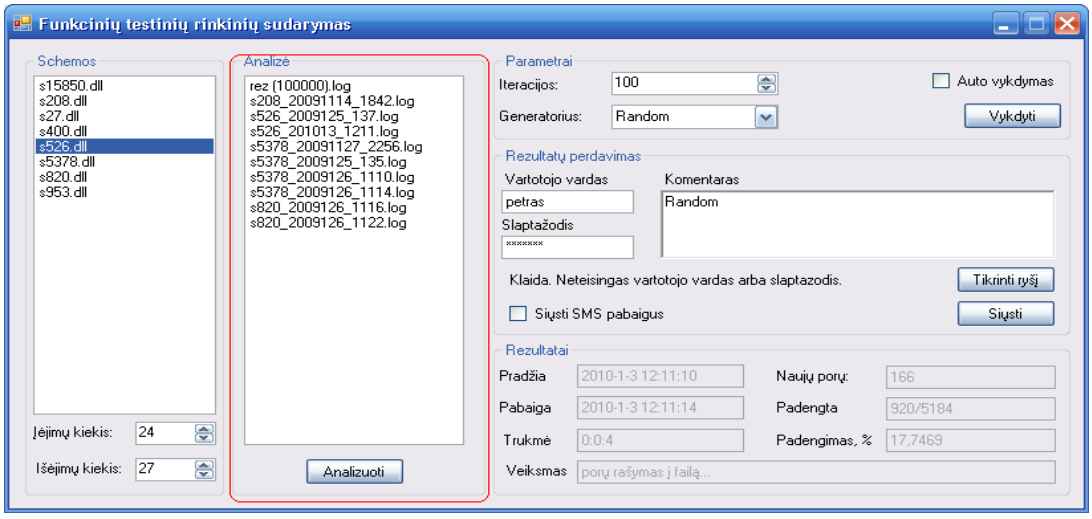
Programos darbas pradedamas, kuomet pasirenkama testuojama schema. Schemos DLL failai, turi būti tame pačiame kataloge kaip ir programa. Schemos prototipo DLL pasirinkimas matomas 3.10.2.1. pav.



3.10.2.1. pav. Schemos prototipo pasirinkimas

3.10.3 Testinių rinkinių analizė

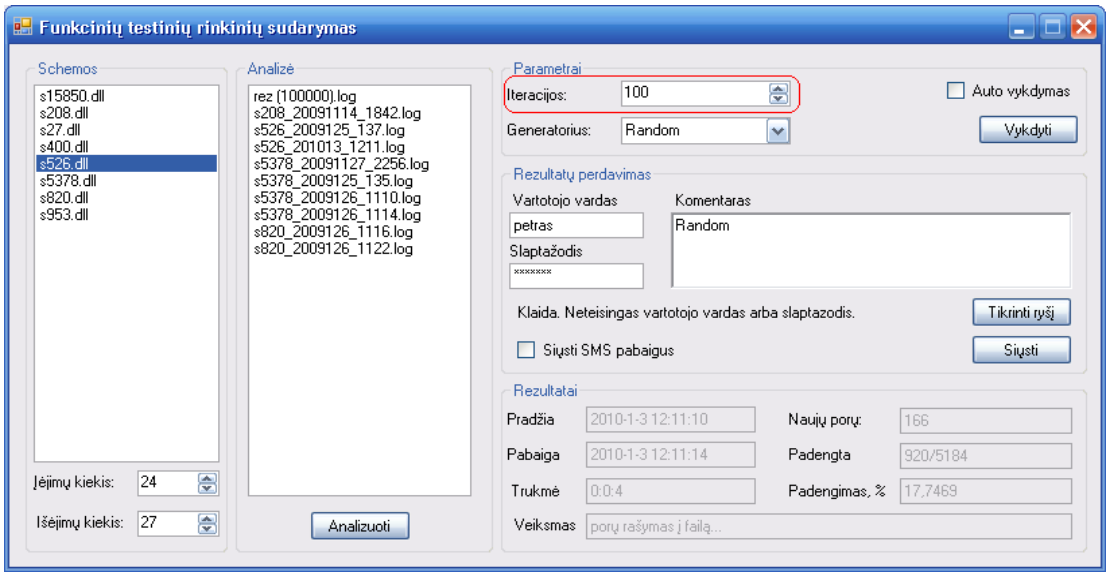
Pasirinkus testuojamą schemą, galima atlikti jau sugeneruotų testinių rinkinių analizę iš failo. Testiniai rinkiniai skaitomi iš *.log failų. Testinių rinkinių failai turi būti patalpinti tame pačiame kataloge kaip ir programa. Rodomas visas sugeneruotų testinių rinkinių failų sąrašas 3.10.3.1. pav. Pasirinkus testinių rinkinių failą ir paspaudus mygtuką „Analizuoti“ yra atliekama testinių rinkinių analizė. Dešinėje programos pusėje pateikiami analizės rezultatai. Atlikus analizę galima tęsti generavimą nuo esamos būsenos. Tokiu atveju naujai sugeneruoti testiniai rinkiniai bus pridedami į to pačio failo pabaigą.



3.10.3.1. pav. Testinių rinkinių analizė

3.10.4 Algoritmo iteracijų keitimas

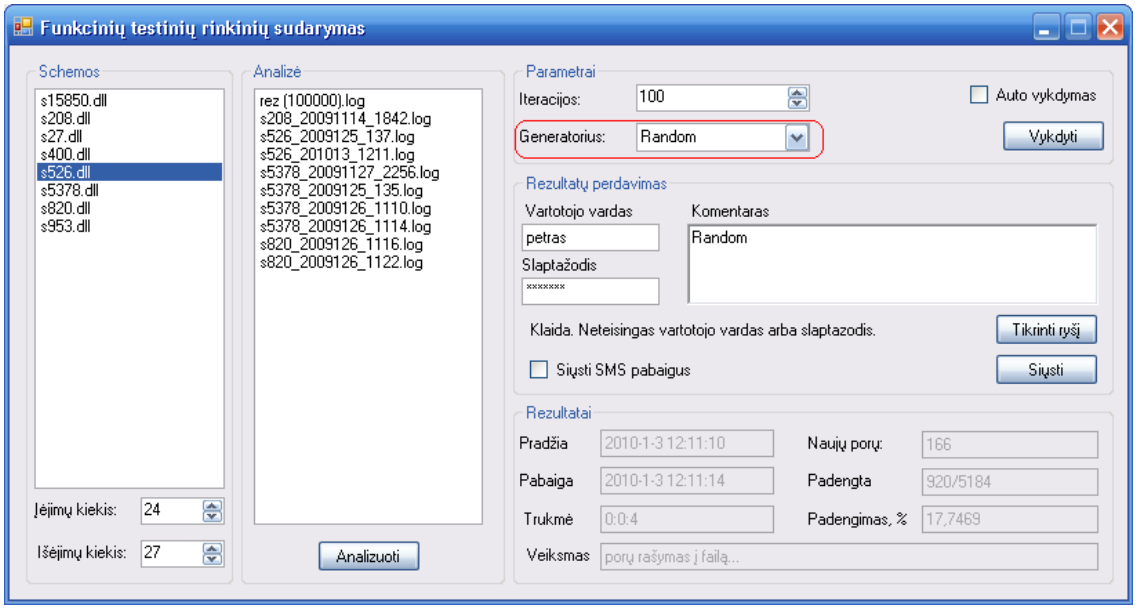
Testinių rinkinių generatoriaus algoritmui galima nustatyti iteracijų kiekį, kuris turi būti įvykdytas tam, kad būtų atrinkta pati geriausia sugeneruota testinių rinkinių pora 3.10.4.1. pav.



3.10.4.1. pav. Algoritmo iteracijų keitimas

3.10.5 Generatoriaus pasirinkimas

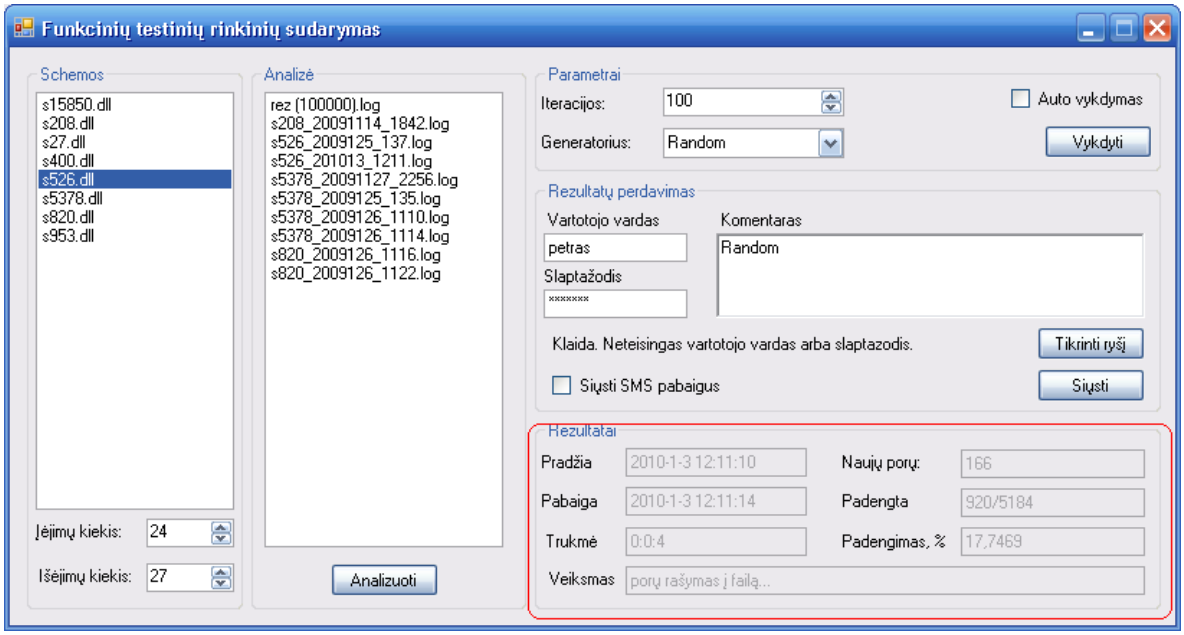
Testinių rinkinių generatorių galima pasirinkti iš trijų galimų variantų. Galima rinktis atsitiktinius Random, Random 2 arba AntiRandom generavimo algoritmą. 3.10.5.1. pav.



3.10.5.1. pav. Generatoriaus pasirinkimas

3.10.6 Generavimo proceso rezultatai realiu laiku

Algoritmui vykdant testinių rinkinių generavimo ir atrinkimo procesą, galima realiu laiku stebėti sugeneruotų ir atrinktų porų skaičių, padengimą, vykdymo trukmę, bei konkretų atliekamą veiksmą 3.10.6.1. pav. Siekiant kuo mažiau apkrauti procesorių, ši informacija yra atnaujinama kas 1000 iteracijų.



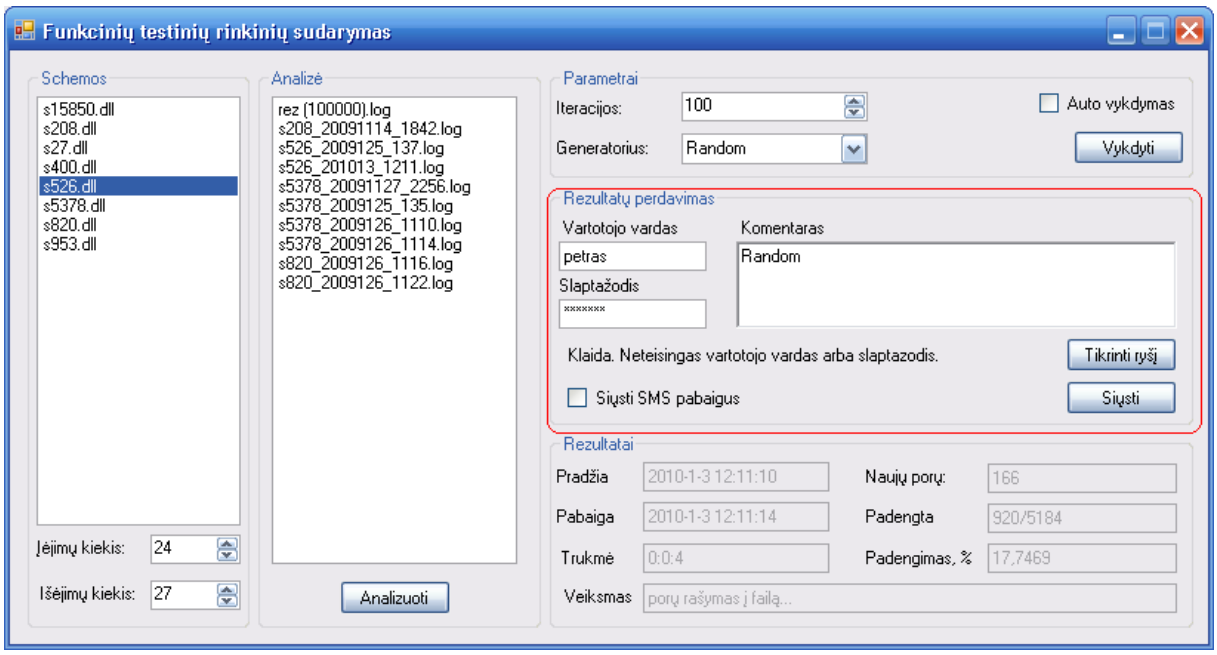
3.10.6.1. pav. Generavimo rezultatai realiu laiku

3.10.6.1. lentelė Parametrų reikšmės

Parametras	**Reikšmė**
Pradžia	Generavimo proceso pradžios laikas
Pabaiga	Generavimo proceso pabaigos laikas (faktinis)
Trukmė	Generavimo proceso trukmė
Veiksmas	Atliekamas veiksmas: • Porų skaitymas iš failo • Naujų porų generavimas... žingsnis n • Porų rašymas į failą
Naujų porų	Naujai sugeneruotų porų skaičius.
Padengta	Padengimo erdvės matricos X užpildymas.
Padengimas	Matricos padengimas procentais

3.10.7 Rezultatų perdavimas į serverį

Sugeneravus ir atrinkus testinių rinkinių poras, jas galima perduoti į serverį. Taip pat yra perduodama ir visa kita informacija. Tam, kad būtų galima perduoti rezultatus į serverį reikia turėti vartotojo vardą ir slaptažodį. Suvedus šiuos duomenis galima patikrinti ar pavyko užmegzti ryšį su serveriu (mygtukas „Tikrinti ryšį“). 3.10.7.1. pav.

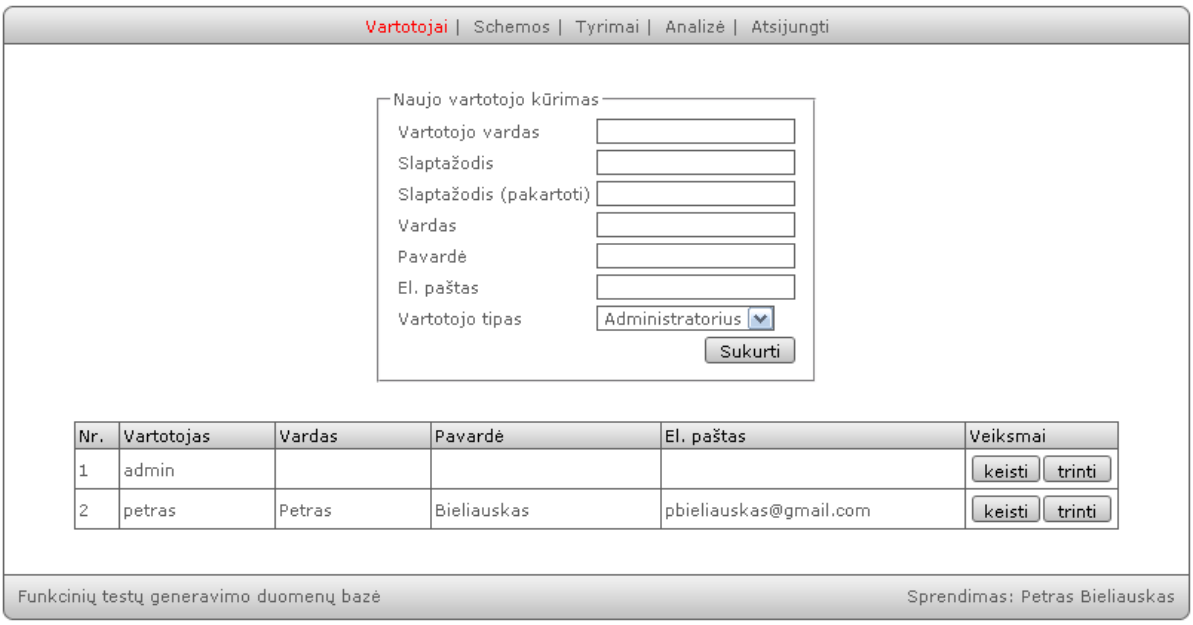


3.10.7.1. pav. Rezultatų perdavimas į serverį

3.10.8 Vartotojų valdymas

Prisijungti prie internetinės sistemos ir perduodi duomenis į šią sistemą gali tik užregistruoti administratoriaus vartotojai. Kuriant naują vartotoją, reikia nurodyti vartotojo vardą ir slaptažodį. Papildomai galima įvesti šią informaciją; vardą, pavardę, el. pašto adresą. 3.10.8.1. pav. ir 3.10.8.2. pav.

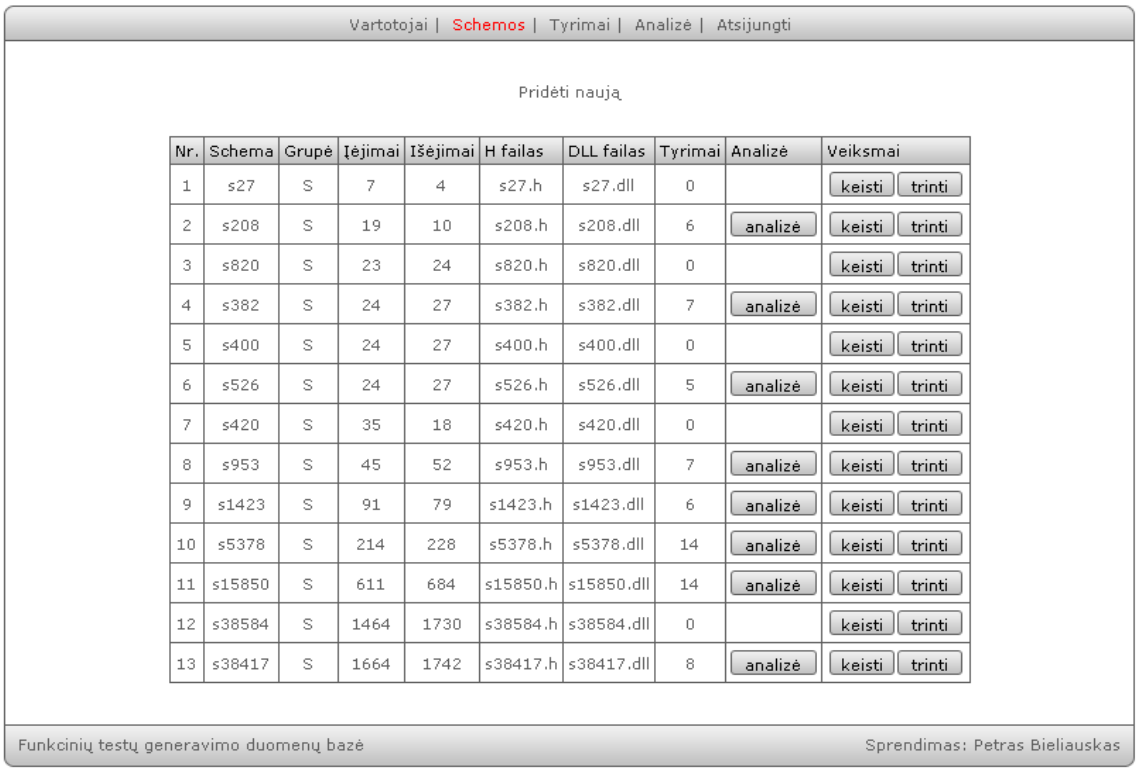
3.10.8.1. pav. Sistemos vartotojų sąrašas



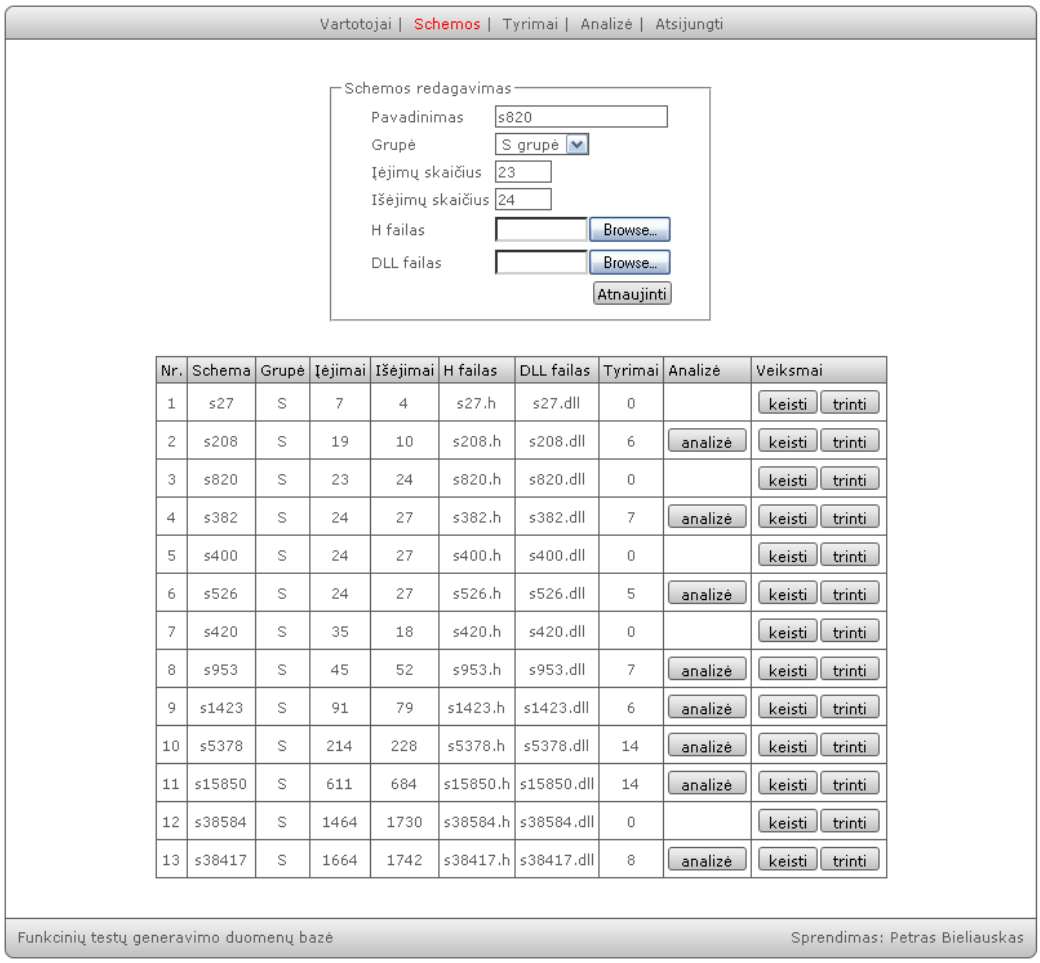
3.10.8.2. pav. Vartotojų kūrimas / redagavimas

3.10.9 Schemų prototipų administravimas

Testuojamos schemos turi būti užregistruotos sistemoje, tik tokiu atveju bus galima perduoti rezultatus iš kompiuterių į sistemą. Sistemoje saugoma reikalinga informacija apie schemas: pavadinimas, įėjimų skaičius, išėjimų skaičius, H failas, DLL failas. H ir DLL failus gali parsiųsti vartotojai ir atlikti testinių rinkinių generavimą (3.10.9.1 . pav., 3.10.9.2 pav.).



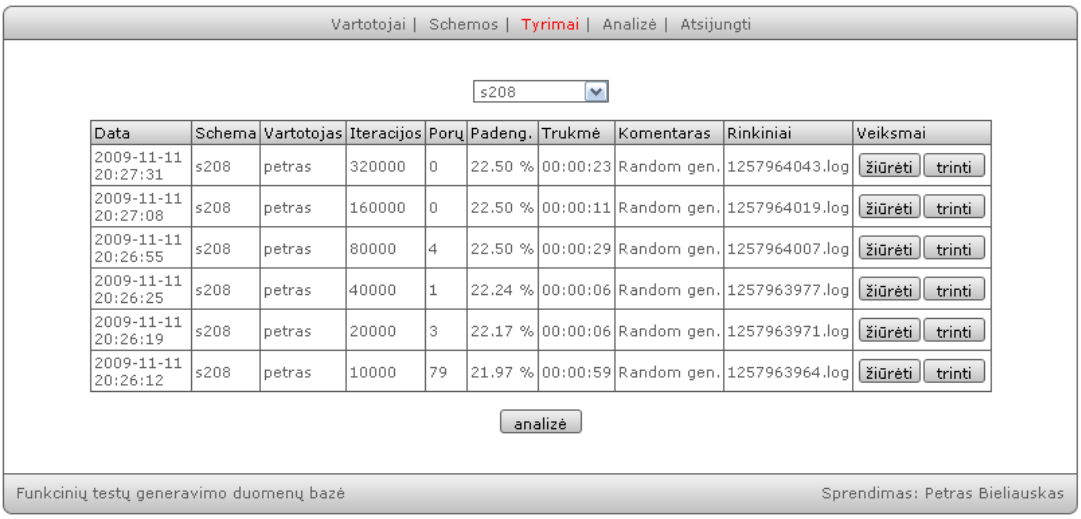
3.10.9.1. pav. Schemų prototipų sąrašas



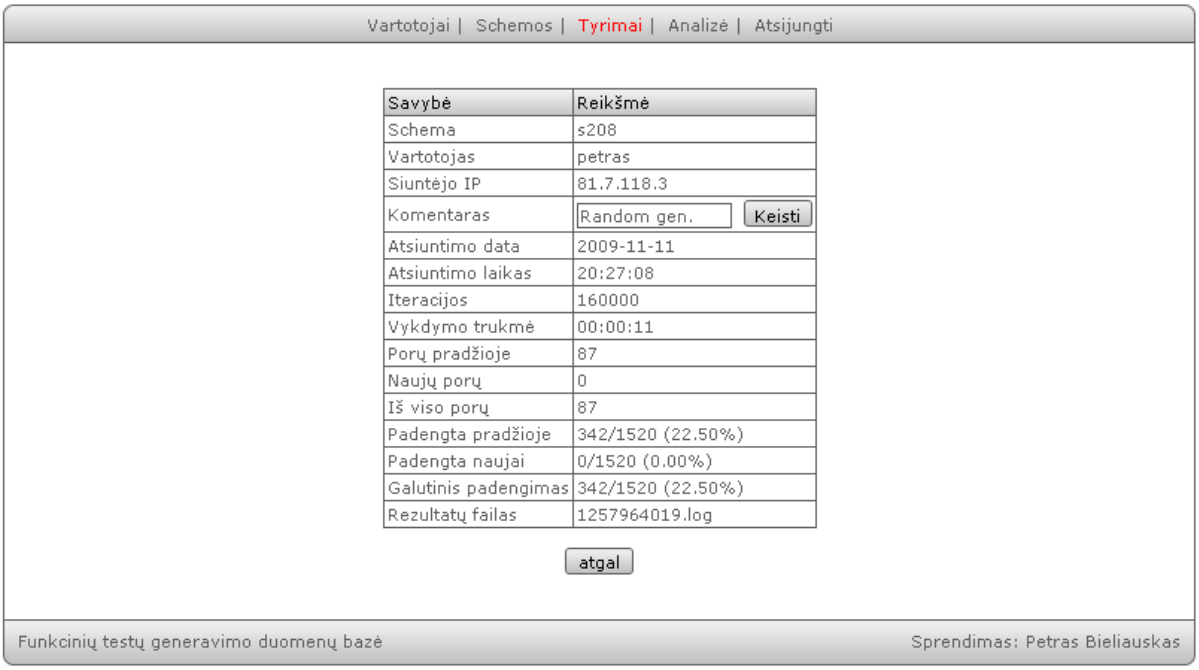
3.10.9.2. pav. Schemų kūrimas / redagavimas

3.10.10 Schemų tyrimai

Atsiųsti generavimo rezultatai matomi sąraše. Galima žiūrėti bendrą sąrašą, arba tyrimus pagal schemas. Prie kiekvieno tyrimo matoma bendra informacija ir rinkinių failas, kurį galima parsisiųsti arba peržiūrėti. Pasirinkus platesnį aprašymą galima matyti visą tyrimo informaciją (3.10.10.1 . pav., 3.10.10.2.pav.).



3.10.10.1. pav. Tyrimų sąrašas



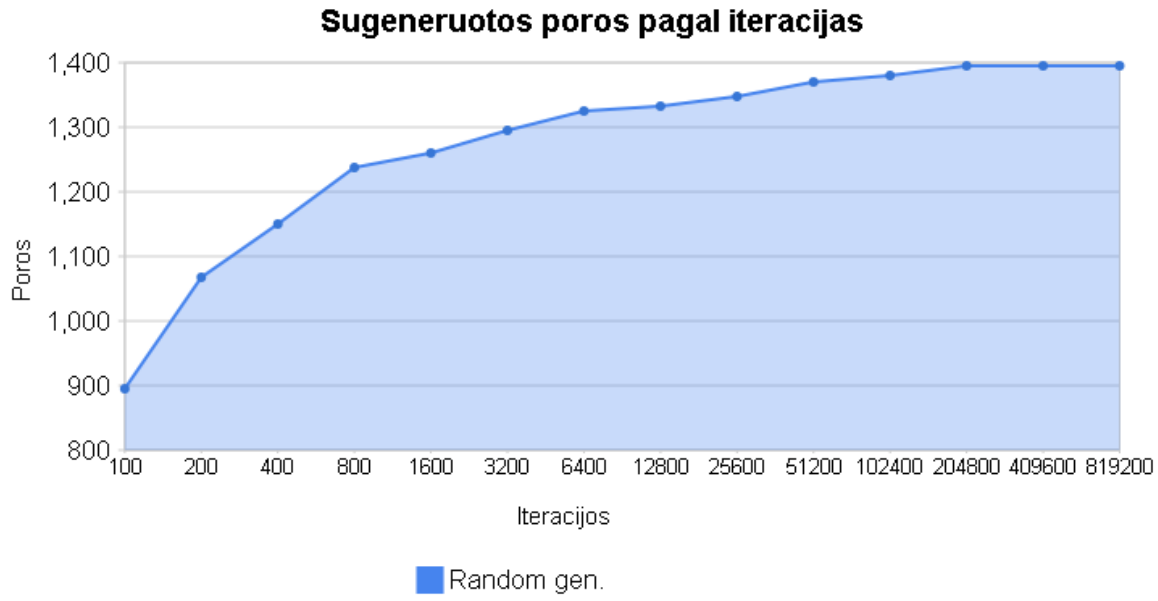
3.1010.2.pav. Detalus tyrimo aprašymas

Detaliame tyrimo aprašyme yra rodoma visa informacija susijusiu su atliktu eksperimentiniu tyrimu:

- Schemos pavadinimas;
- Vartotojas atsiuntęs tyrimą;
- Siuntėjo kompiuterio IP adresas;
- Generatoriaus pavadinimas (Random, Random2, AntiRandom);
- Atsiuntimo data ir laikas;
- Iteracijų kiekis;
- Generavimo trukmė;
- Testinių rinkinių kiekis tyrimo pradžioje;
- Testinių rinkinių kiekis tyrimo pabaigoje;
- Matricos X užpildymas tyrimo pradžioje;
- Matricos X užpildymas tyrimo pabaigoje;
- Testinių rinkinių failas atsisiuntimui į kompiuterį.

3.10.11 Schemų tyrimų analizė

Pagal atsiųstus generavimo rezultatus yra sudaromi grafikai, pagal kuriuos vartotojas gali atlikti rezultatų analizę ir juos įvertinti. (3.10.11.1. pav. - 3.10.11.6.pav.).

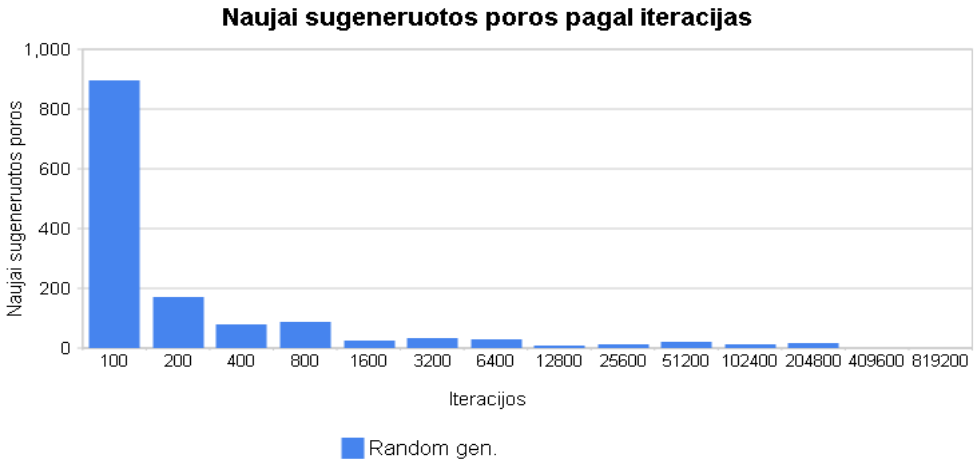


3.10.11.1. pav. Grafikas „Sugeneruotos poros pagal iteracijas“

Sugeneruotų porų pagal iteracijas grafikas (3.10.11.1. pav.) parodo sugeneruojamų ir atrinktų testinių rinkinių porų augimą didėjant iteracijų kiekiui.

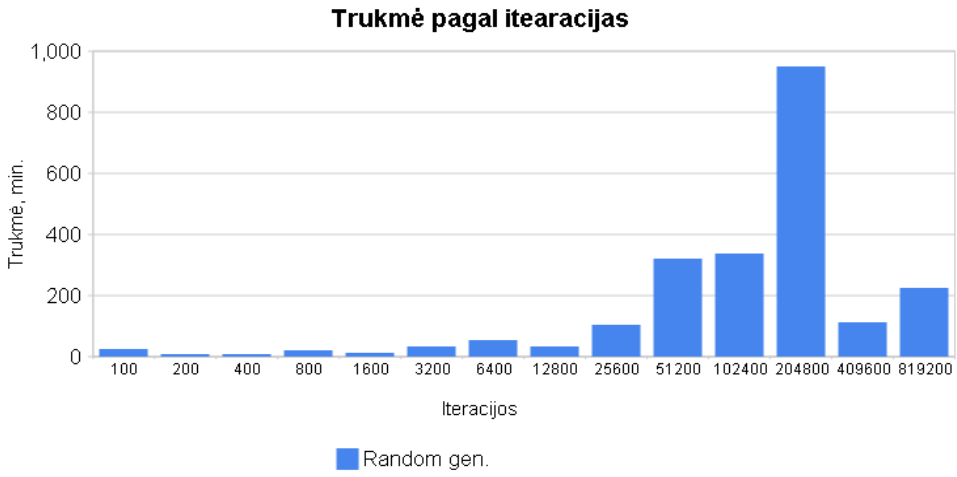
3.10.11.2. pav. Grafikas „Padengimas pagal iteracijas“

Padengimo pagal iteracijas grafikas (3.10.11.2. pav.) parodo matricos X užpildymo augimą didėjant iteracijų kiekiui.



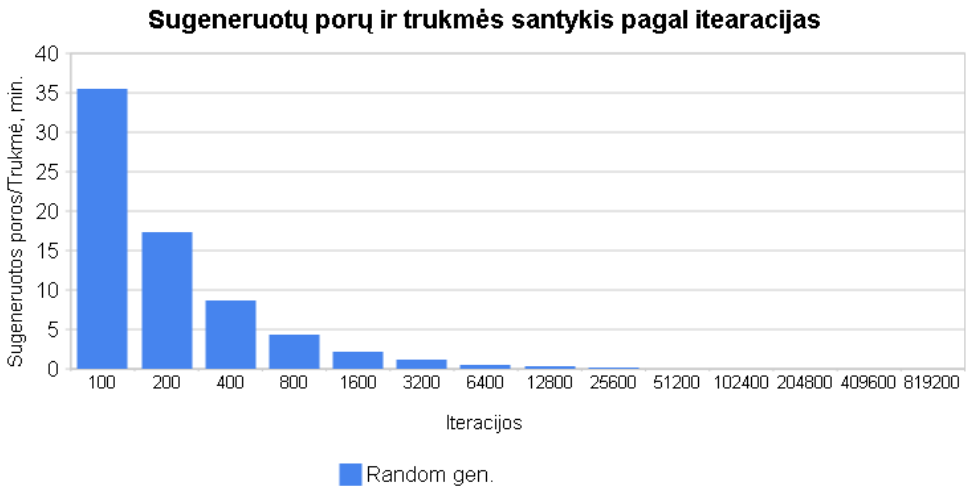
3.10.11.3. pav. Grafikas „Naujai sugeneruotos poros pagal iteracijas“

Naujai sugeneruotų porų pagal iteracijas grafikas (3.10.11.3. pav.) parodo naujai sugeneruotų ir atrinktų testinių rinkinių kiekį prie atitinkamo iteracijų kiekio.



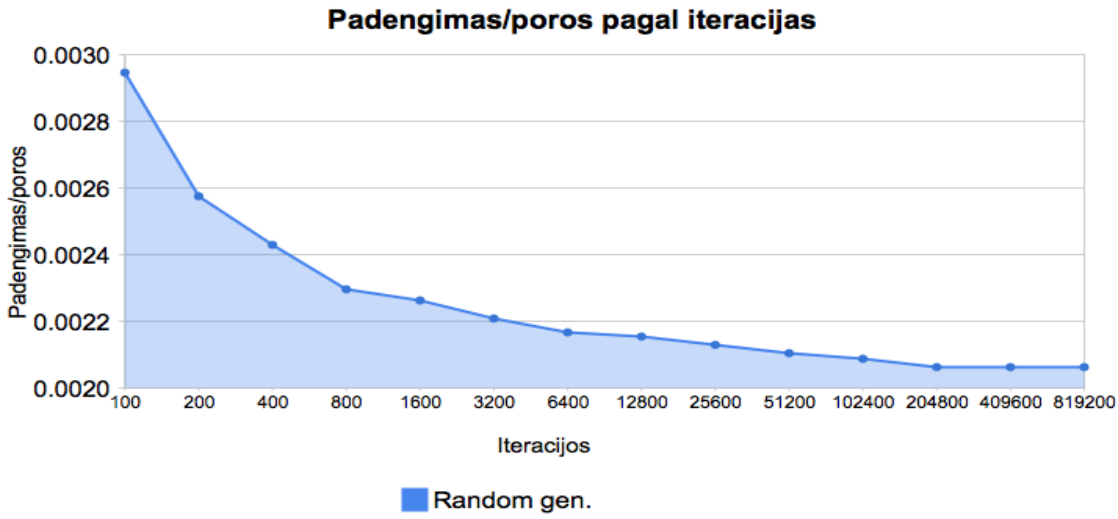
3.10.11.4. pav. Grafikas „Trukmė pagal iteracijas“

Trukmės pagal iteracijas grafikas (3.10.11.4. pav.) parodo kiek laiko užtruko naujų testinių rinkinių generavimas ir atrinkimas prie atitinkamo iteracijų kiekio.



3.10.11.5. pav. Grafikas „Sugeneruotų porų ir trukmės santykis pagal iteracijas“

Sugeneruotų porų ir trukmės santykio pagal iteracijas grafikas (3.10.11.5. pav.) parodo naujai sugeneruotų ir atrinktų testinių rinkinių santykį su generavimo trukme. Tai yra kokybinis matas, kuris parodo kada testiniai rinkiniai yra sugeneruojami ir atrenkami efektyviausiai. T.y. Atrenkama daugiau testinių rinkinių per mažesnį laiką.



3.10.11.6. pav. Grafikas „Padengimas/poros pagal iteracijas“

Padengimo/porų pagal iteracijas grafikas (3.10.11.6. pav.) parodo matricos X užpildymo santykį su naujai sugeneruotų ir atrinktų testinių rinkinių skaičiumi. Šis parametras nurodo kaip priklauso matricos X užpildymas nuo generuojamų ir atrenkamų porų skaičiaus. Didesnė šio parametro reikšmė reiškia geresnį generavimo efektyvumą.

3.11 Išvados

1. Tam, kad būtų galima pritaikyti AntiRandom metodą funkcinių testų generavimui jį reikia apjungti kartu su atsitiktiniu generavimu;
2. Darbo meto buvo suprojektuotas ir realizuotas funkcinių testų generatorius;
3. Gautų rezultatų kaupimui ir analizei buvo sukurta nuotolinė sistema, kuri priima duomenis iš generatorių kliento kompiuteriuose;
4. Nuotolinėje sistemoje buvo sukurtos įvairios gautų rezultatų atvaizdavimo diagramos, kurių pagalba galima vizualiai matyti vieno ar kito generatoriaus pasiekimus bei juos tarpusavyje lyginti;
5. Funkcinių testų kokybei įvertinti nepakanka įprastinių parametrų tokių kaip (generavimo trukmė, sugeneruoti ir atrinkti rinkiniai, matricos užpildymas). Todėl buvo išvesti papildomi santykiai, kurie leidžia nustatyti net tik funkcinių testų, bet ir viso generavimo proceso efektyvumą.

4 Funkcinių testų generavimo tyrimas

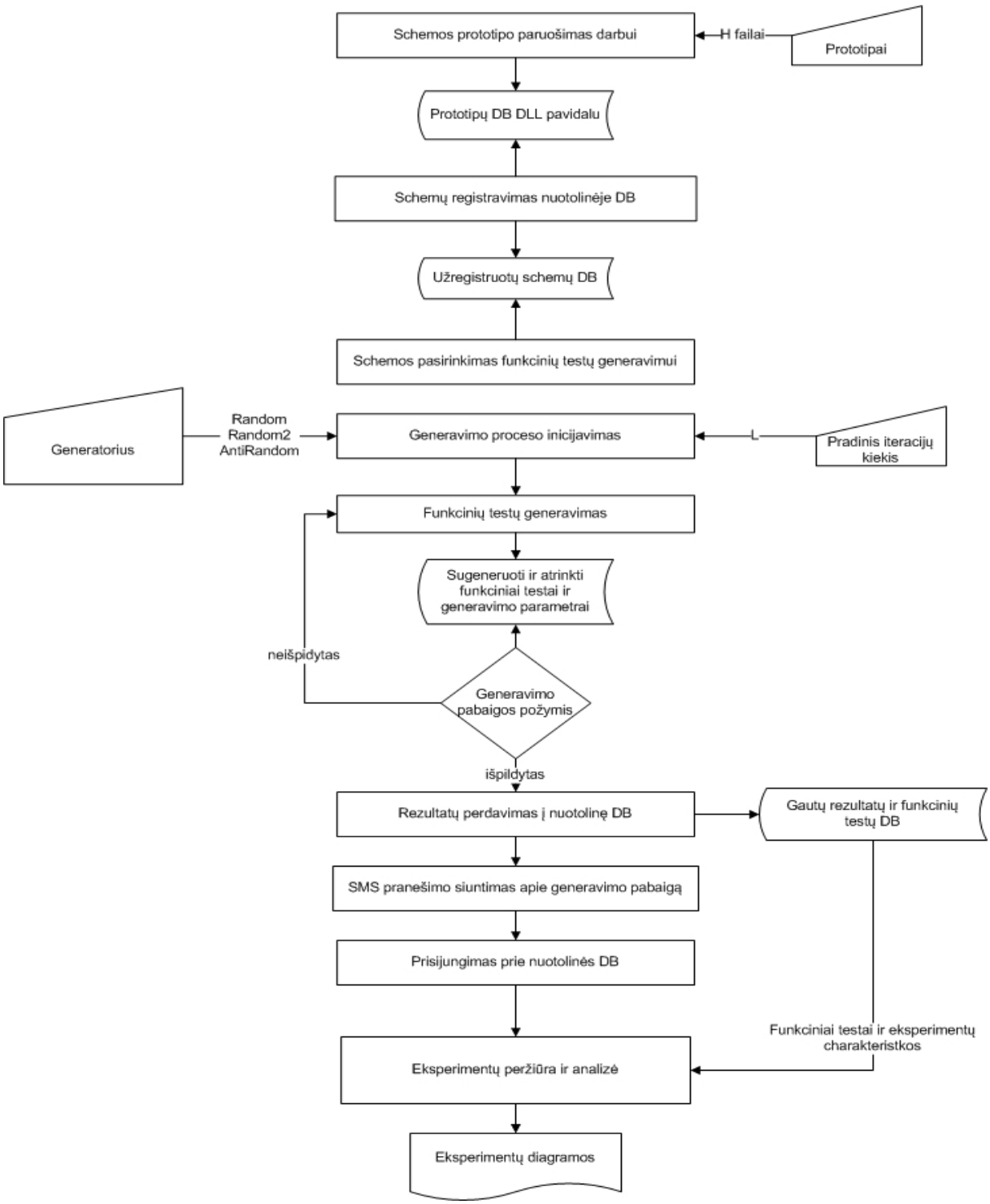
Sukurtos programinės įrangos pagalba buvo atliktas eksperimentinis tyrimas su schemų prototipais siekiant išbandyti ir palyginti skirtingus funkcinių testų generatorius.

4.1 Tyrimų vykdymo tvarka

Buvo sukurtas eksperimentinių tyrimų vykdymo planas pagal kurį buvo vykdomi sukurtos programinės įrangos eksperimentiniai tyrimai.

Eksperimentinių tyrimų eiga (4.1.1.pav.):

1. S tipo schemų prototipų paruošimas tyrimams;
2. Schemų užregistravimas sukurtoje nuotolinėje rezultatų kaupimo ir apdorojimo sistemoje;
3. Schemos prototipo pasirinkimas tyrimui;
4. Generatoriaus pasirinkimas (Random, Random 2, AntiRandom);
5. Pradinio iteracijų kiekio nustatymas;
6. Funkcinių testų generavimo proceso inicijavimas ir paleidimas;
7. Funkcinių testų saugojimas faile (automatizuotas);
8. Iteracijų kiekio didinimas (rankinis arba automatizuotas);
9. Generavimo proceso nutraukimas (rankinis arba automatizuotas);
10. Funkcinių testų ir generavimo charakteristikų perdavimas į nuotolinę DB (rankinis arba automatizuotas);
11. SMS pranešimo siuntimas apie generavimo pabaigą;
12. Rezultatų peržiūra ir analizė nuotolinėje DB pagal sugeneruotas diagramas;
13. Funkcinių testų parsisiuntimas iš nuotolinės DB ir perdavimas TetraMAX programinei įrangai.



4.1.1. pav. Eksperimentinių tyrimų eigos diagrama

4.2 Gautų rezultatų įvertinimo kriterijai

Testų generatorių palyginimui buvo naudojamos šios charakteristikos:

- Sugeneruotų ir atrinktų testinių porų skaičius;
- Generavimo trukmė;
- Iteracijų skaičius;

- Perėjimo gedimų padengimas TFC

Gauti rezultatai buvo vertinami panaudojant sukurtos nuotolinės sistemos generuojamas diagramas. Vertinimui buvo naudojamos šios diagramos:

- Sugeneruotų porų pagal iteracijas diagrama parodo sugeneruojamų ir atrinktų testinių rinkinių porų augimą didėjant iteracijų kiekiui;
- Matricos užpildymo pagal iteracijas diagrama parodo matricos X užpildymo augimą didėjant iteracijų kiekiui. Didesnis užpildymas reiškia, jog funkciniai testai gali pasiekti aukštesnį klaidų padengimo laipsnį;
- Matricos užpildymo ir sugeneruotų porų santykio diagrama parodo matricos X užpildymo santykį su naujai sugeneruotų ir atrinktų testinių rinkinių skaičiumi. Šis parametras nurodo kaip priklauso matricos X užpildymas nuo generuojamų ir atrenkamų porų skaičiaus. Didesnė šio parametro reikšmė reiškia geresnį generavimo efektyvumą. T.y. pasiekiamas aukštesnis užpildymo laipsnis su mažesniu rinkinių kiekiu;
- Trukmės pagal iteracijas diagrama parodo kiek laiko užtruko naujų testinių rinkinių generavimas ir atrinkimas prie atitinkamo iteracijų kiekio. Didėjanti šio parametro reikšmė parodo augančią generavimo proceso trukmę;
- Sugeneruotų porų ir trukmės santykio pagal iteracijas diagrama parodo naujai sugeneruotų ir atrinktų testinių rinkinių santykį su generavimo trukme. Tai yra kokybinis matas, kuris parodo kada testiniai rinkiniai yra sugeneruojami ir atrenkami efektyviausiai. T.y. Atrenkama daugiau testinių rinkinių per mažesnį laiką. Šį santykį dar būtų galima vadinti generavimo greičiu.

TFC parametras buvo gautas pasinaudojant TetraMAX programine įranga. Šios programos pagalba buvo apdorojami sugeneruoti funkciniai testai, bei gautas perėjimo gedimų padengimas.

4.3 Tiriamos schemos

Eksperimentiniam tyrimui atlikti buvo naudojami S tipo schemų programiniai prototipai parašyti C kalba.

Tyrimo metu naudotų schemų sąrašas pateiktas 4.3.1. lentelėje.

4.3.1. lentelė. Naudotos schemos

Schema	Įėjimų skaičius	Išėjimų skaičius
S1196	32	32
S1238	32	32
S13207	700	790
S15850	611	684
S35932	1763	2048
S38417	1664	1742

4.4 Eksperimentinio tyrimo rezultatai

Eksperimentinio tyrimo metu buvo generuojami funkciniai testai schemų prototipams. Kiekvienai schemai buvo generuojami funkciniai testai pagal tris generatorius: Random, Random 2 ir AntiRandom. Eksperimentų rezultatai pateikiami nuotolinės duomenų bazės sugeneruotų diagramų pavidalu su paaiškinimais.

4.4.1 S1196 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.1.1. - 4.4.1.8. pav.

Savybė	Reikšmė
Schema	s1196
Siuntėjo IP	88.119.128.223
Procesorius	Intel(R) Core(TM)2 CPU T5500 @ 1.66GHz
Komentaras	Random gen. Keisti
Atsiuntimo data	2010-03-04
Atsiuntimo laikas	08:36:47
Iteracijos	10000
Vykdymo trukmė	00:05:59
Porų pradžioje	0
Naujų porų	274
Iš viso porų	274
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	1739/8192 (21.23%)
Galutinis padengimas	1739/8192 (21.23%)
Rezultatų failas	1267684610.log

4.4.1.1. pav. Random generatorius. Pirmas tyrimas.

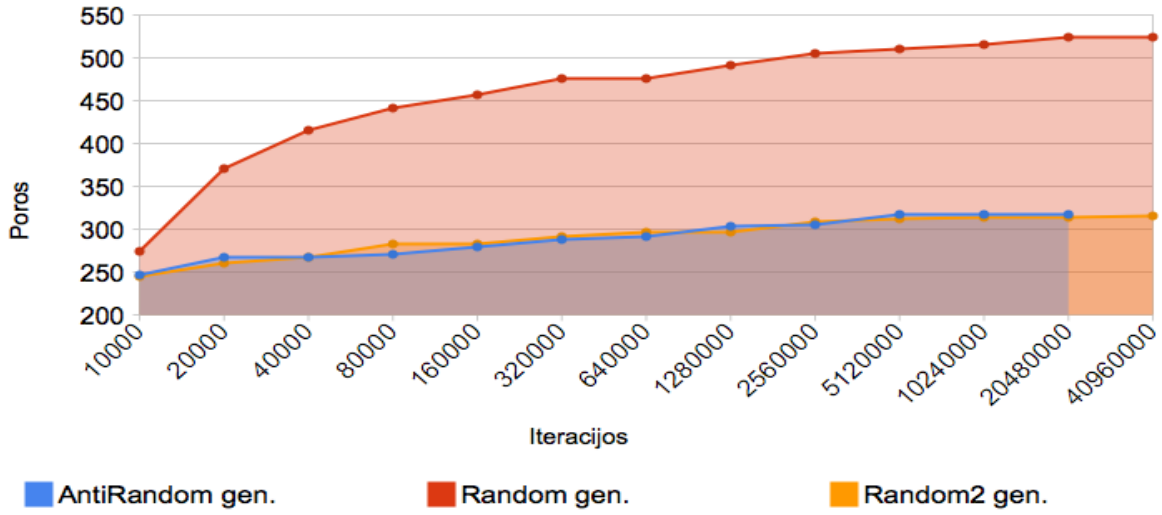
Savybė	Reikšmė
Schema	s1196
Siuntėjo IP	88.119.128.223
Procesorius	Intel(R) Core(TM)2 CPU T5500 @ 1.66GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-03-05
Atsiuntimo laikas	09:17:52
Iteracijos	10000
Vykdymo trukmė	00:05:20
Porų pradžioje	0
Naujų porų	245
Iš viso porų	245
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	2299/8192 (28.06%)
Galutinis padengimas	2299/8192 (28.06%)
Rezultatų failas	1267773471.log

4.4.1.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	s1196
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-23
Atsiuntimo laikas	23:31:59
Iteracijos	10000
Vykdymo trukmė	00:05:42
Porų pradžioje	0
Naujų porų	247
Iš viso porų	247
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	2304/8192 (28.12%)
Galutinis padengimas	2304/8192 (28.12%)
Rezultatų failas	1269379918.log

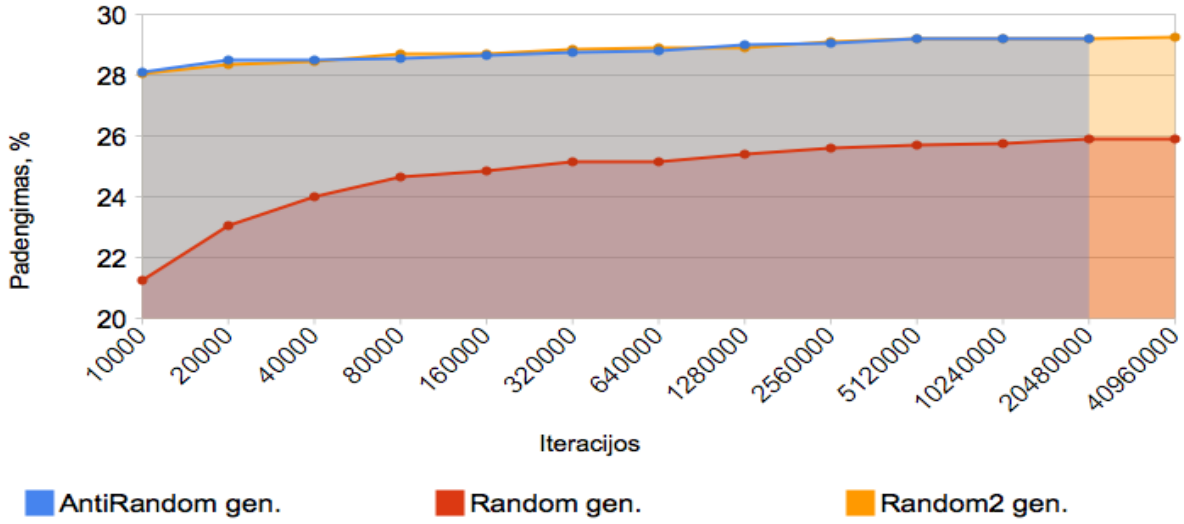
4.4.1.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 10000 iteracijų, šis iteracijų skaičius buvo dvigubinamas (20000, 40000).



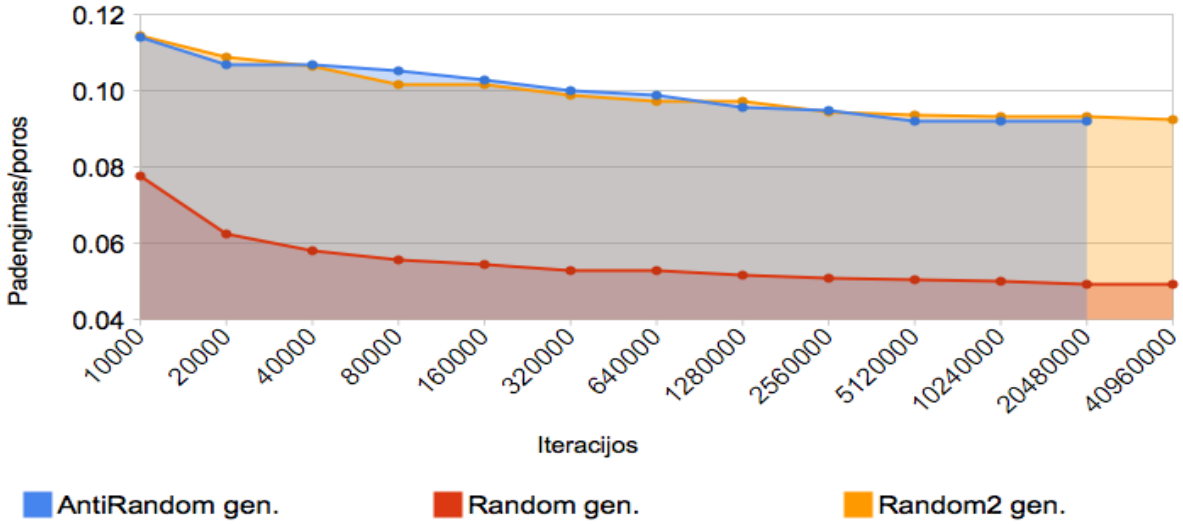
4.4.1.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.1.4. pav.) matome, kad Random generatorius sugeneruodavo ir atrinkdavo daugiau testinių rinkinių nei Random2 ir AntiRandom esant tam pačiam iteracijų kiekiui. Iš šios diagramos galima daryti prielaidą, kad tiriamai schemai Random generuoja mažiau efektyvius funkcinius testus.



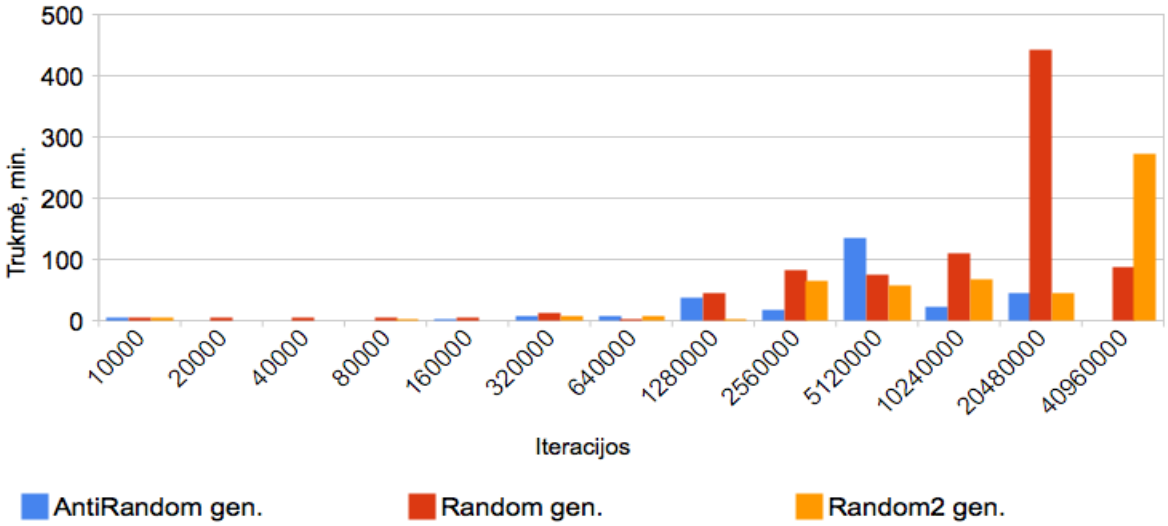
4.4.1.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.1.5. pav.) matome, kad Random2 ir AntiRandom užpildo didesnę matricos dalį nei Random. Todėl galime daryti prielaidą, jog šie generatoriai generuoja funkcinius testus kurie pasiekia didesnį gedimų padengimą.



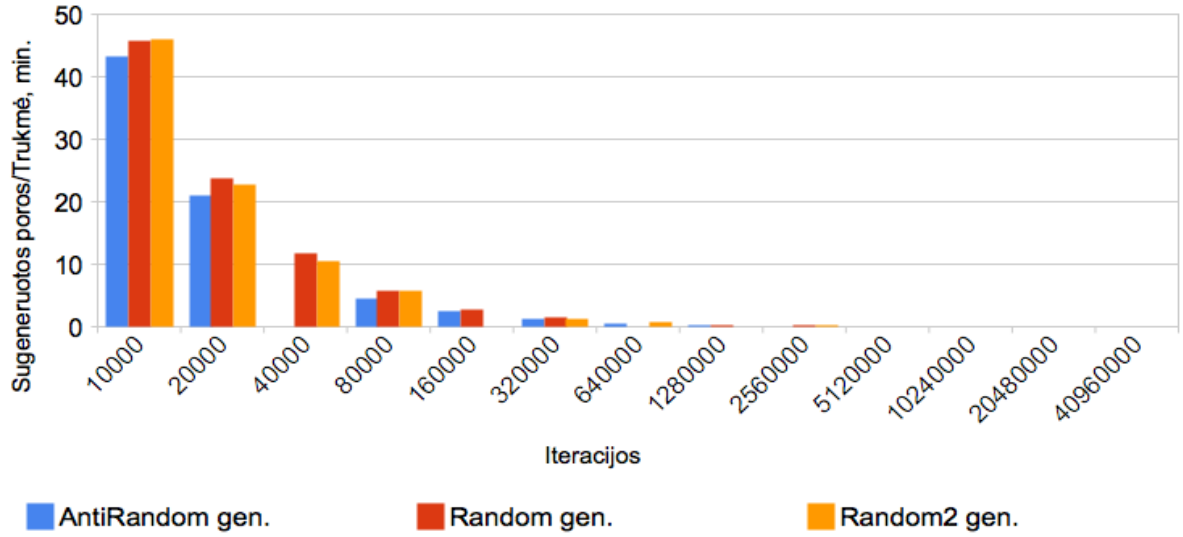
4.4.1.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.1.6. pav.) matome, kad Random2 ir AntiRandom generuoja efektyvesnius funkcinius testus nei Random, kurie pasiekia didesnį matricos X užpildymą su mažesniu atrinktų testinių rinkinių kiekiu.



4.4.1.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.1.7. pav.) matome, kad esant dideliam iteracijų kiekiui (nuo 10240000) Random ir Random2 vykdo generavimą ilgiau nei AntiRandom.



4.4.1.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.1.8. pav.) matome, kad visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra panašus.

4.4.2 S1238 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.2.1. - 4.4.2.8. pav.

Savybė	Reikšmė
Schema	s1238
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random gen. Keisti
Atsiuntimo data	2010-03-05
Atsiuntimo laikas	21:06:25
Iteracijos	10000
Vykdymo trukmė	00:06:22
Porų pradžioje	0
Naujų porų	263
Iš viso porų	263
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	1741/8192 (21.25%)
Galutinis padengimas	1741/8192 (21.25%)
Rezultatų failas	1267815984.log

4.4.2.1. pav. Random generatorius. Pirmas tyrimas.

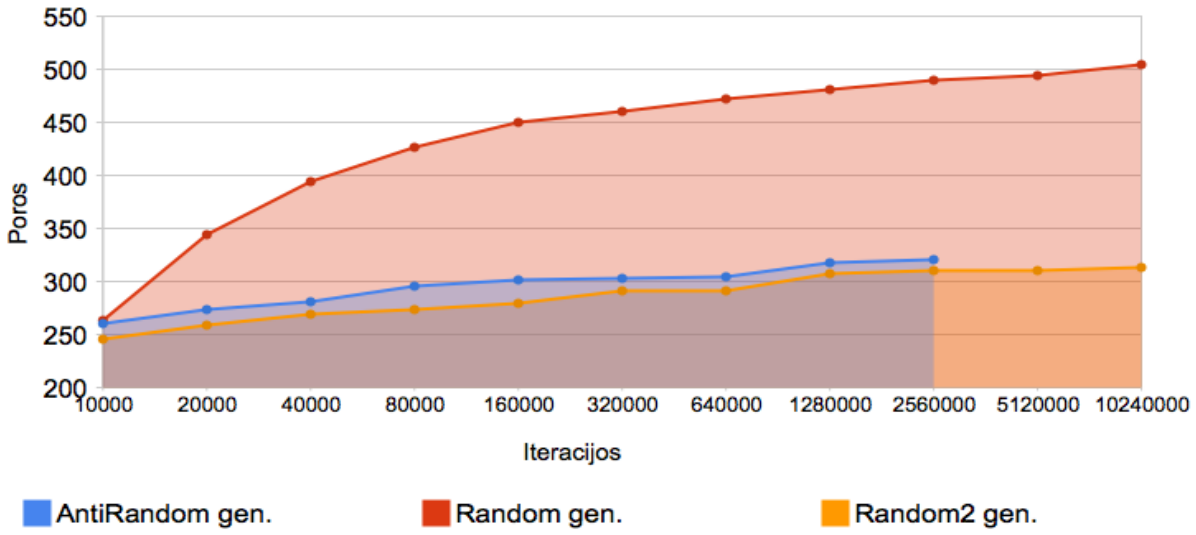
Savybė	Reikšmė
Schema	s1238
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-03-06
Atsiuntimo laikas	12:08:35
Iteracijos	10000
Vykdymo trukmė	00:05:36
Porų pradžioje	0
Naujų porų	246
Iš viso porų	246
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	2307/8192 (28.16%)
Galutinis padengimas	2307/8192 (28.16%)
Rezultatų failas	1267870113.log

4.4.2.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	s1238
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-25
Atsiuntimo laikas	22:44:29
Iteracijos	10000
Vykdymo trukmė	00:06:28
Porų pradžioje	0
Naujų porų	260
Iš viso porų	260
Padengta pradžioje	0/8192 (0.00%)
Padengta naujai	2312/8192 (28.22%)
Galutinis padengimas	2312/8192 (28.22%)
Rezultatų failas	1269549868.log

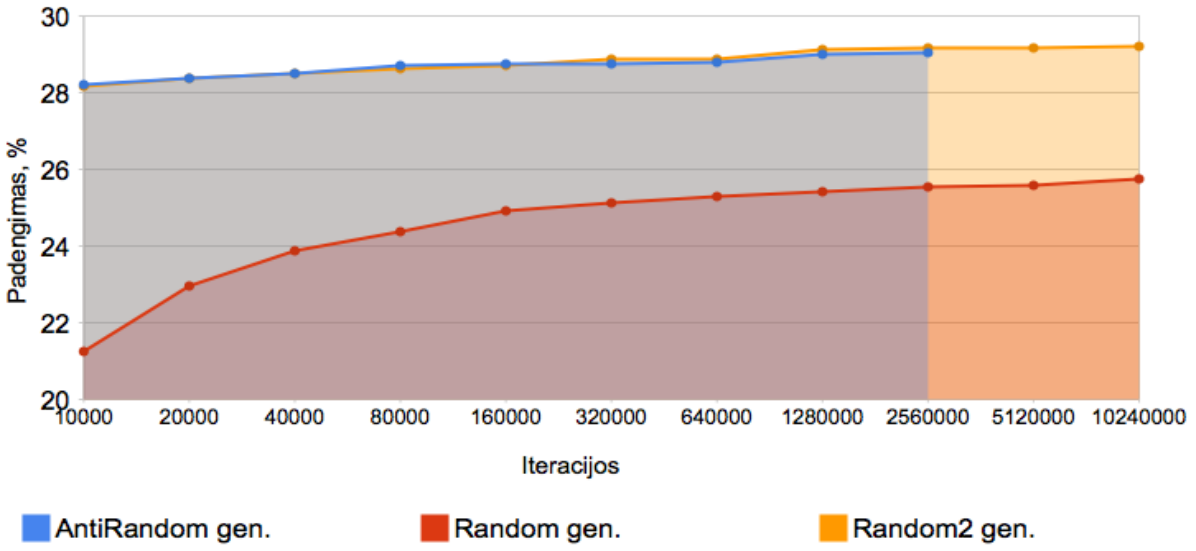
4.4.2.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 10000 iteracijų, šis iteracijų skaičius buvo dvigubinamas (20000, 40000). Generavimui buvo naudojamas Inetl(R) Pentium(R) 4 CPU 3.20 GHz procesorius.



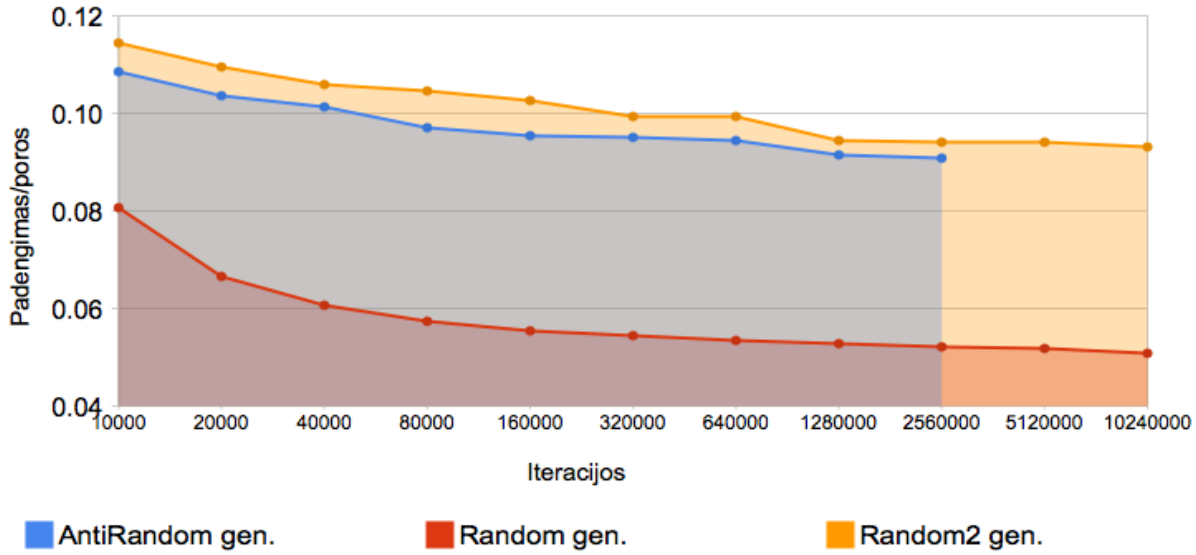
4.4.2.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.2.4. pav.) matome, jog Random generatorius sugeneruoja ir atrenka daugiau testinių rinkinių nei Random2 ir AntiRandom esant tam pačiam iteracijų kiekiui. Tai vėlgi gali būti neefektyvaus generavimo pasekmė.



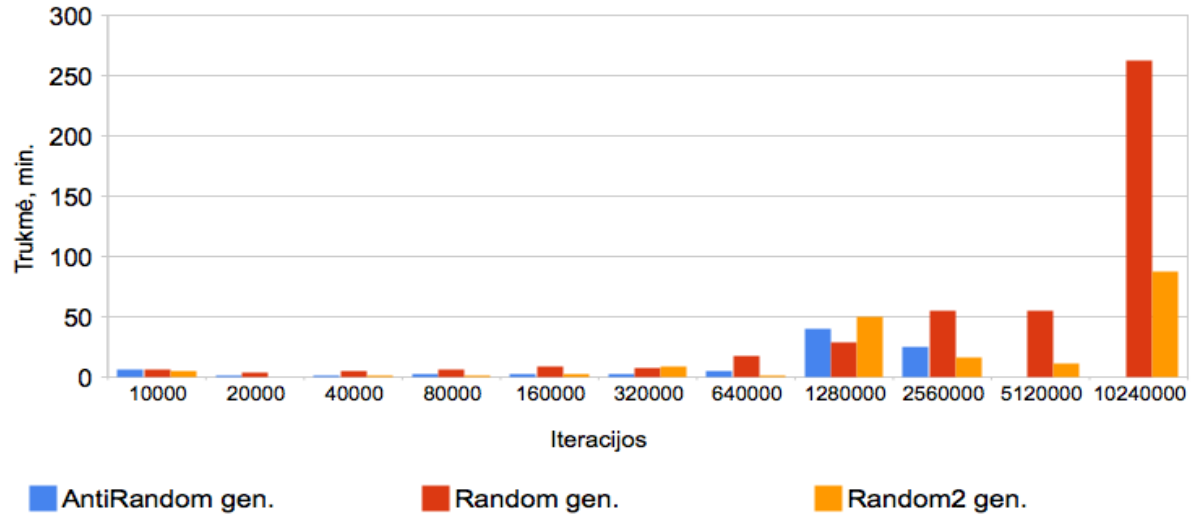
4.4.2.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.2.5. pav.) matome, kad Random2 ir AntiRandom užpildo didesnę matricos dalį nei Random. Matome, jog ir šiai schemai Random2 ir AntiRandom generatoriai generuoja efektyvesnius funkcinius testus.



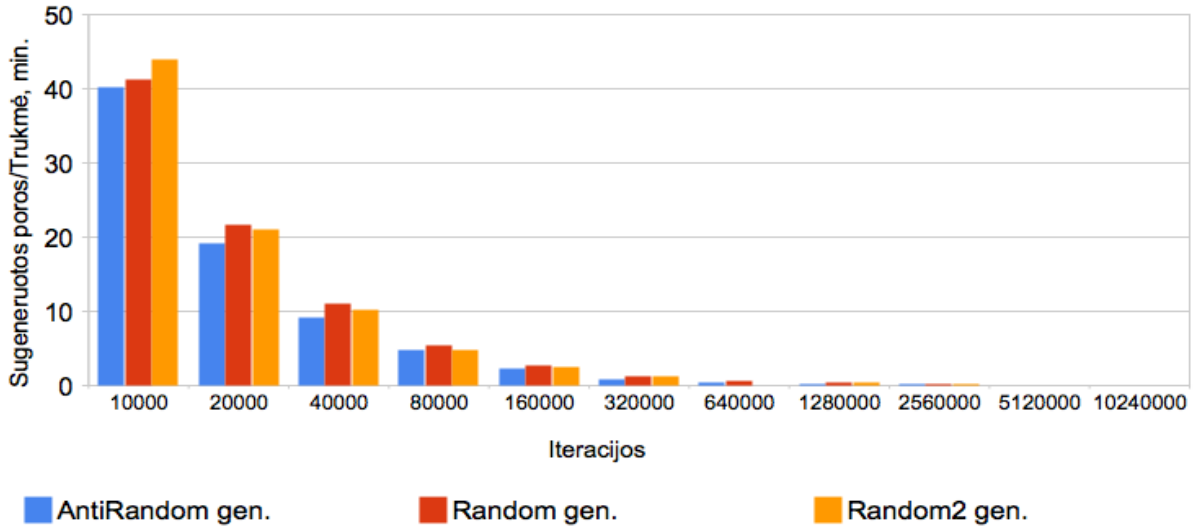
4.4.2.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.2.6. pav.) matome, kad AntiRandom generuoja gerokai efektyvesnius funkcinius testus nei Random, kurie pasiekia didesnį matricos X užpildymą su mažesniu atrinktų testinių rinkinių kiekiu. Random2 pagal šį santykį šiek tiek lenkia AntiRandom.



4.4.2.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.2.7. pav.) matome, kad didėjant iteracijų kiekiui smarkiai padidėja Random generavimo trukmė.



4.4.2.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.2.8. pav.) matome, kad visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra panašus. Random ir Random2 šiek tiek lenkia AntiRandom.

4.4.3 S13207 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.3.1. - 4.4.3.8. pav.

Savybė	Reikšmė
Schema	13207
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random gen. Keisti
Atsiuntimo data	2010-03-07
Atsiuntimo laikas	21:37:43
Iteracijos	100
Vykdymo trukmė	01:00:03
Porų pradžioje	0
Naujų porų	1085
Iš viso porų	1085
Padengta pradžioje	0/4424000 (0.00%)
Padengta naujai	14592/4424000 (0.33%)
Galutinis padengimas	14592/4424000 (0.33%)
Rezultatų failas	1267990660.log

4.4.3.1. pav. Random generatorius. Pirmas tyrimas.

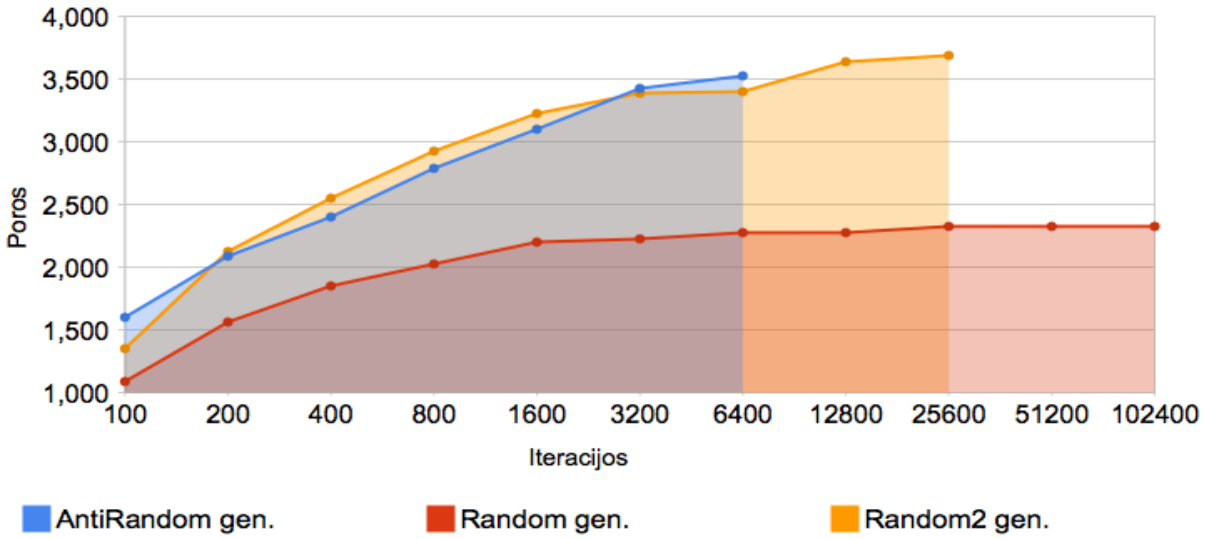
Savybė	Reikšmė
Schema	13207
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-03-12
Atsiuntimo laikas	14:36:00
Iteracijos	100
Vykdymo trukmė	01:17:19
Porų pradžioje	0
Naujų porų	1355
Iš viso porų	1355
Padengta pradžioje	0/4424000 (0.00%)
Padengta naujai	18595/4424000 (0.42%)
Galutinis padengimas	18595/4424000 (0.42%)
Rezultatų failas	1268397357.log

4.4.3.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	13207
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-24
Atsiuntimo laikas	08:35:18
Iteracijos	100
Vykdymo trukmė	01:26:43
Porų pradžioje	0
Naujų porų	1603
Iš viso porų	1603
Padengta pradžioje	0/4424000 (0.00%)
Padengta naujai	19396/4424000 (0.44%)
Galutinis padengimas	19396/4424000 (0.44%)
Rezultatų failas	1269412516.log

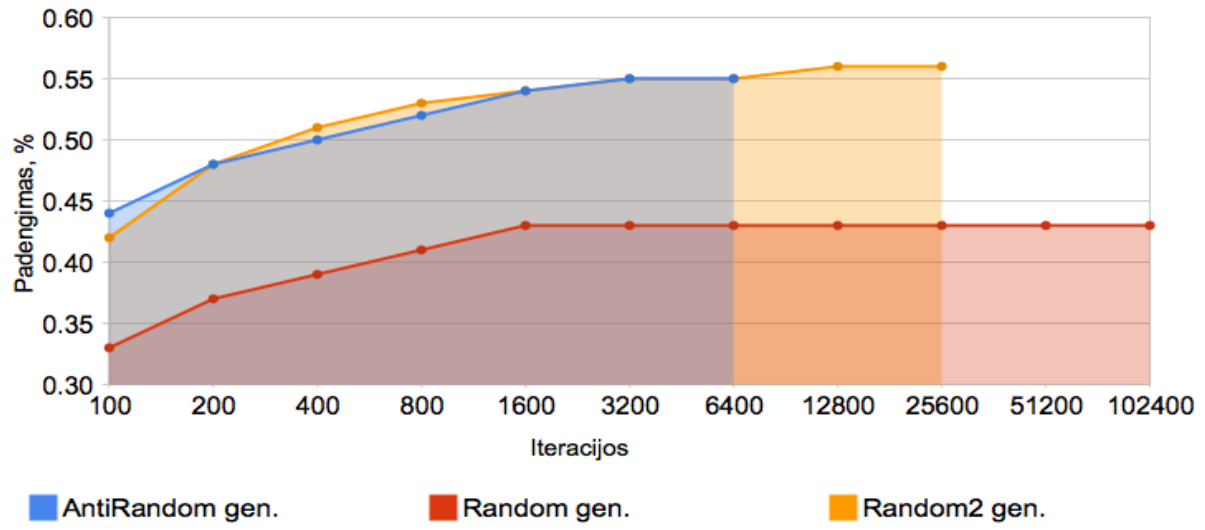
4.4.3.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 100 iteracijų, šis iteracijų skaičius buvo dvigubinamas (200, 400). Generavimui buvo naudojamas Inetl(R) Pentium(R) 4 CPU 3.20 GHz procesorius.



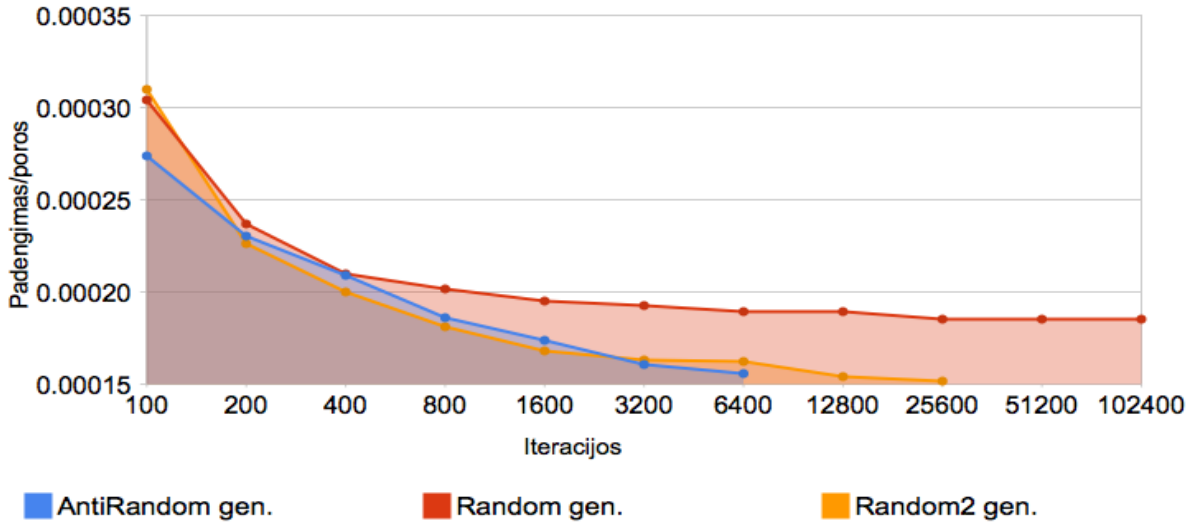
4.4.3.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.3.4. pav.) matome, jog su šia schema priešingai nei su ankstesnėmis Random2 ir AntiRandom generatoriai sugeneruoja ir atrenka daugiau testinių rinkinių nei Random esant tam pačiam iteracijų kiekiui.



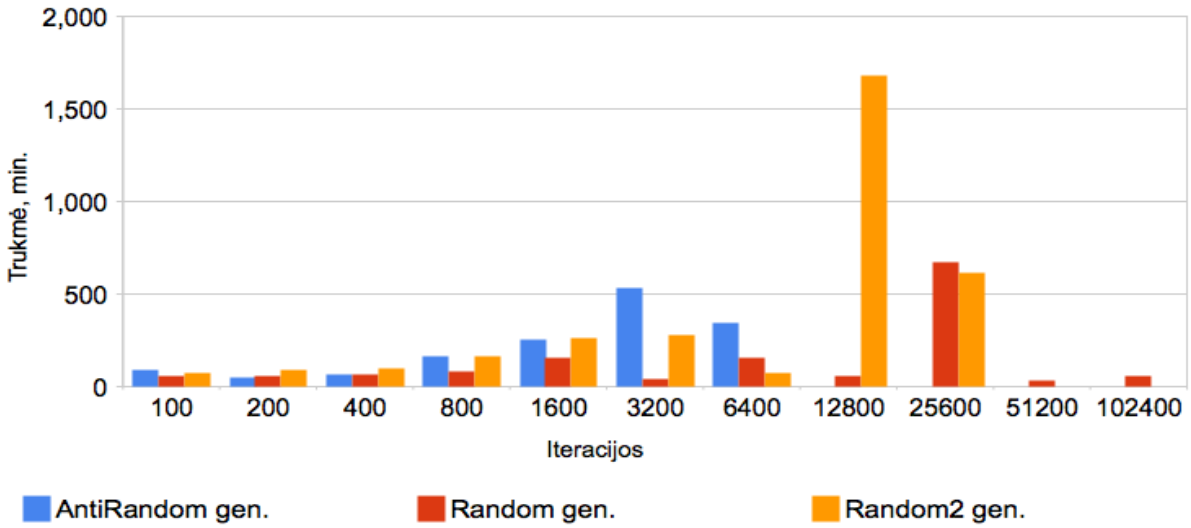
4.4.3.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.3.5. pav.) matome, kad Random2 ir AntiRandom vis dar užpildo didesnę matricos dalį nei Random.



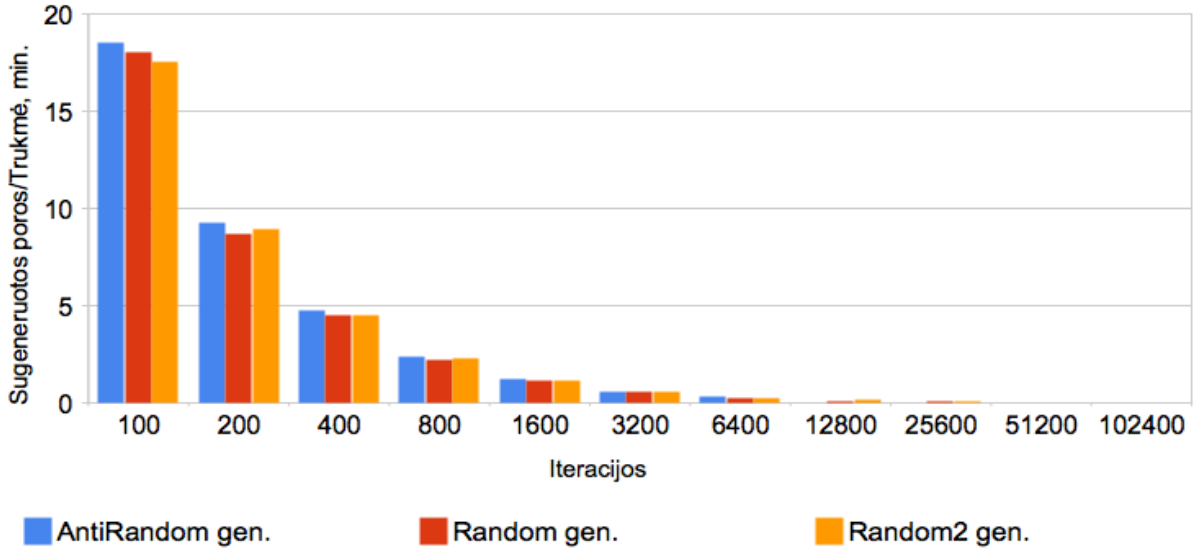
4.4.3.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.3.6. pav.) matome, kad šį kartą Random generatorius generuoja šiek tiek efektyvesnius funkcinius testus nei Random2 ir AntiRandom.



4.4.3.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.3.7. pav.) matome, kad didėjant iteracijų kiekiui smarkiai padidėja Random ir Random2 generavimo trukmės.



4.4.3.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.3.8. pav.) matome, kad visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra beveik identiškas.

4.4.4 S15850 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.4.1. - 4.4.4.8. pav.

Savybė	Reikšmė
Schema	s15850
Siuntėjo IP	81.7.118.3
Procesorius	
Komentaras	Random gen. Keisti
Atsiuntimo data	2009-11-29
Atsiuntimo laikas	12:42:38
Iteracijos	100
Vykdymo trukmė	01:21:36
Porų pradžioje	0
Naujų porų	1585
Iš viso porų	1585
Padengta pradžioje	0/3343392 (0.00%)
Padengta naujai	31480/3343392 (0.94%)
Galutinis padengimas	31480/3343392 (0.94%)
Rezultatų failas	1259491352.log

4.4.4.1. pav. Random generatorius. Pirmas tyrimas.

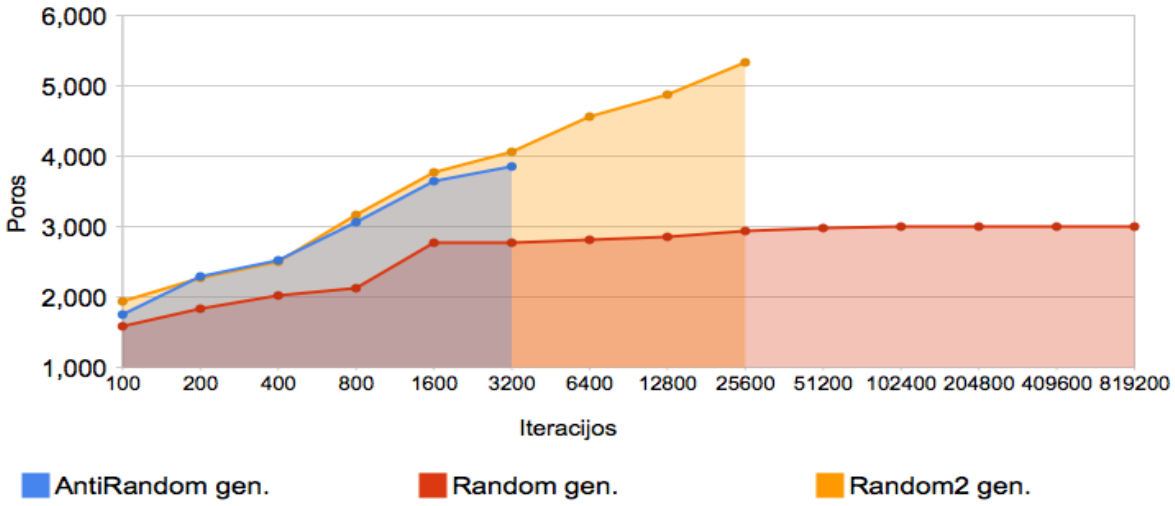
Savybė	Reikšmė
Schema	s15850
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-02-02
Atsiuntimo laikas	23:06:32
Iteracijos	100
Vykdymo trukmė	01:52:46
Porų pradžioje	0
Naujų porų	1937
Iš viso porų	1937
Padengta pradžioje	0/3343392 (0.00%)
Padengta naujai	33910/3343392 (1.01%)
Galutinis padengimas	33910/3343392 (1.01%)
Rezultatų failas	1265144788.log

4.4.4.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	s15850
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-26
Atsiuntimo laikas	01:48:12
Iteracijos	100
Vykdymo trukmė	01:27:39
Porų pradžioje	0
Naujų porų	1760
Iš viso porų	1760
Padengta pradžioje	0/3343392 (0.00%)
Padengta naujai	33140/3343392 (0.99%)
Galutinis padengimas	33140/3343392 (0.99%)
Rezultatų failas	1269560890.log

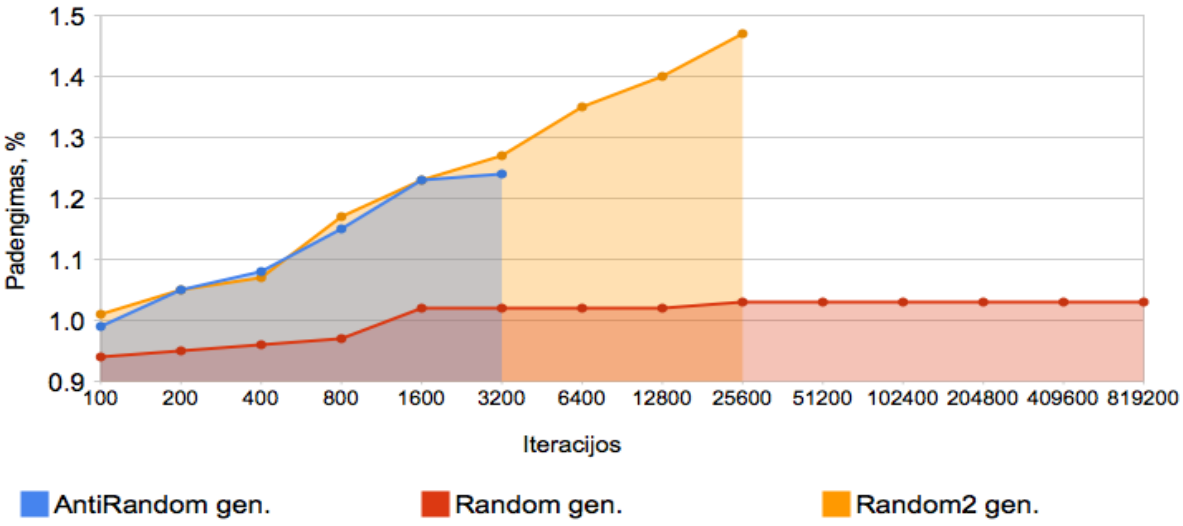
4.4.4.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 100 iteracijų, šis iteracijų skaičius buvo dvigubinamas (200, 400). Generavimui buvo naudojamas Inetl(R) Pentium(R) 4 CPU 3.20 GHz procesorius.



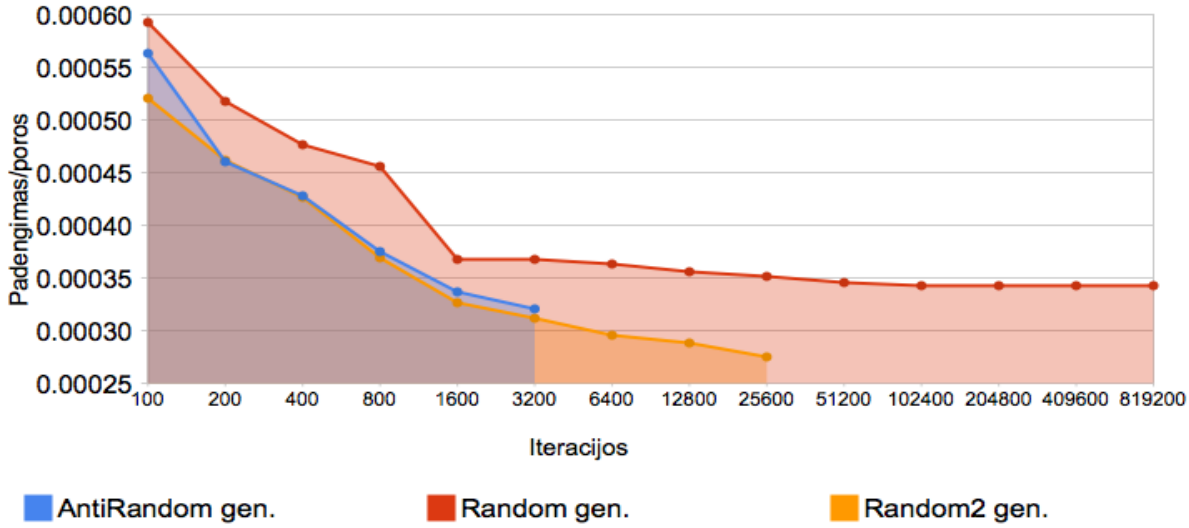
4.4.4.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.4.4. pav.) matome, jog Random2 ir AntiRandom generatoriai sugeneruoja ir atrenka daugiau testinių rinkinių nei Random esant tam pačiam iteracijų kiekiui.



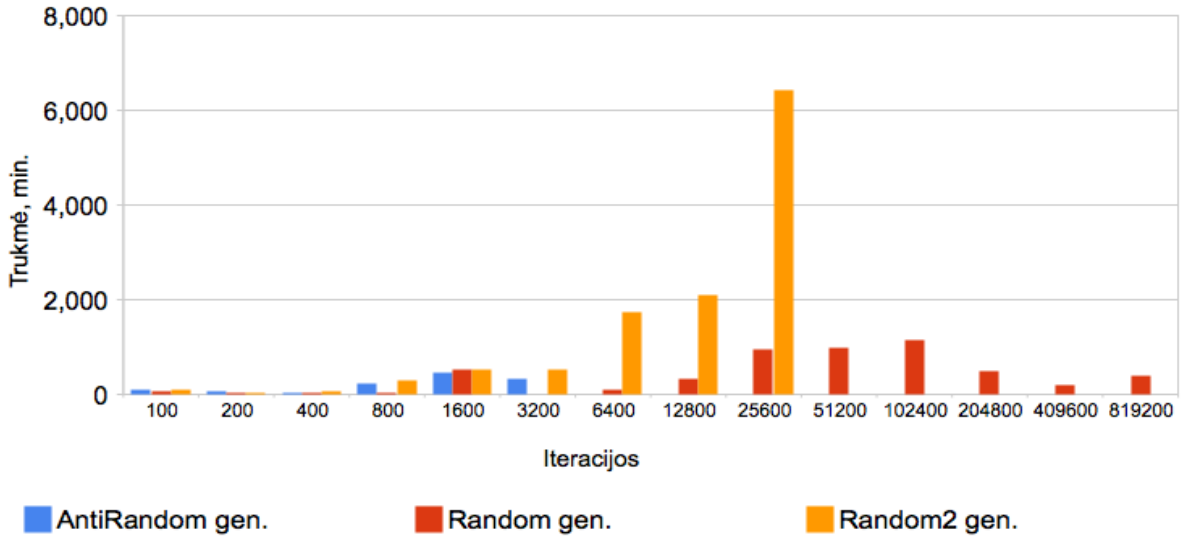
4.4.4.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.4.5. pav.) matome, kad Random2 ir AntiRandom vis užpildo didesnę matricos dalį nei Random. Tačiau reikia pastebėti, kad šis pokytis nėra didelis. Tai lemia ir didelis matricos X dydis.



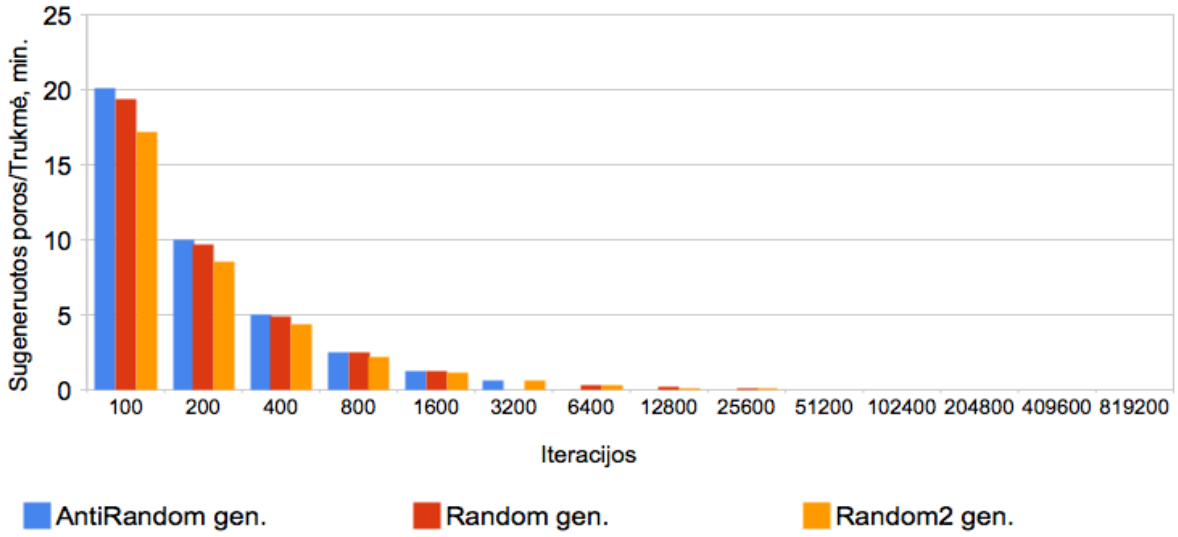
4.4.4.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.4.6. pav.) matome, kad šį kartą Random generatorius generuoja efektyvesnius funkcinius testus nei Random2 ir AntiRandom. AntiRandom pagal tą patį rodiklį tik šiek tiek lenkia Random2.



4.4.4.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.4.7. pav.) matome, kad esant nedideliam iteracijų kiekiui generavimo trukmės yra panašios. Matome, jog prie didelių iteracijų smarkiai išauga generavimo laikas, todėl praktikoje generuojant funkcinius testus reikėtų laikytis mažesnio iteracijų skaičiaus.



4.4.4.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.4.8. pav.) matome, kad kaip ir ankstesnėje schemoje visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra beveik identiškas.

4.4.5 S35932 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.5.1. - 4.4.5.8. pav.

Savybė	Reikšmė
Schema	s35932
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random gen. Keisti
Atsiuntimo data	2010-03-19
Atsiuntimo laikas	09:23:21
Iteracijos	100
Vykdymo trukmė	00:50:49
Porų pradžioje	0
Naujų porų	223
Iš viso porų	223
Padengta pradžioje	0/28884992 (0.00%)
Padengta naujai	40148/28884992 (0.14%)
Galutinis padengimas	40148/28884992 (0.14%)
Rezultatų failas	1268983400.log

4.4.5.1. pav. Random generatorius. Pirmas tyrimas.

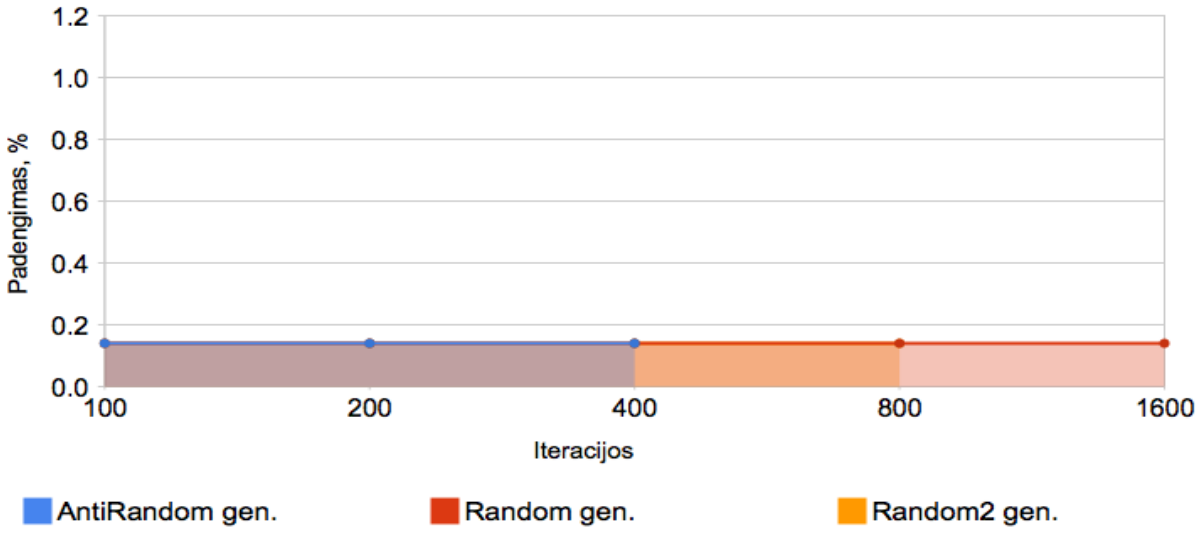
Savybė	Reikšmė
Schema	s35932
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-03-19
Atsiuntimo laikas	12:16:24
Iteracijos	100
Vykdymo trukmė	00:50:21
Porų pradžioje	0
Naujų porų	224
Iš viso porų	224
Padengta pradžioje	0/28884992 (0.00%)
Padengta naujai	40150/28884992 (0.14%)
Galutinis padengimas	40150/28884992 (0.14%)
Rezultatų failas	1268993783.log

4.4.5.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	s35932
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-28
Atsiuntimo laikas	10:03:11
Iteracijos	100
Vykdymo trukmė	00:54:09
Porų pradžioje	0
Naujų porų	221
Iš viso porų	221
Padengta pradžioje	0/28884992 (0.00%)
Padengta naujai	40149/28884992 (0.14%)
Galutinis padengimas	40149/28884992 (0.14%)
Rezultatų failas	1269759790.log

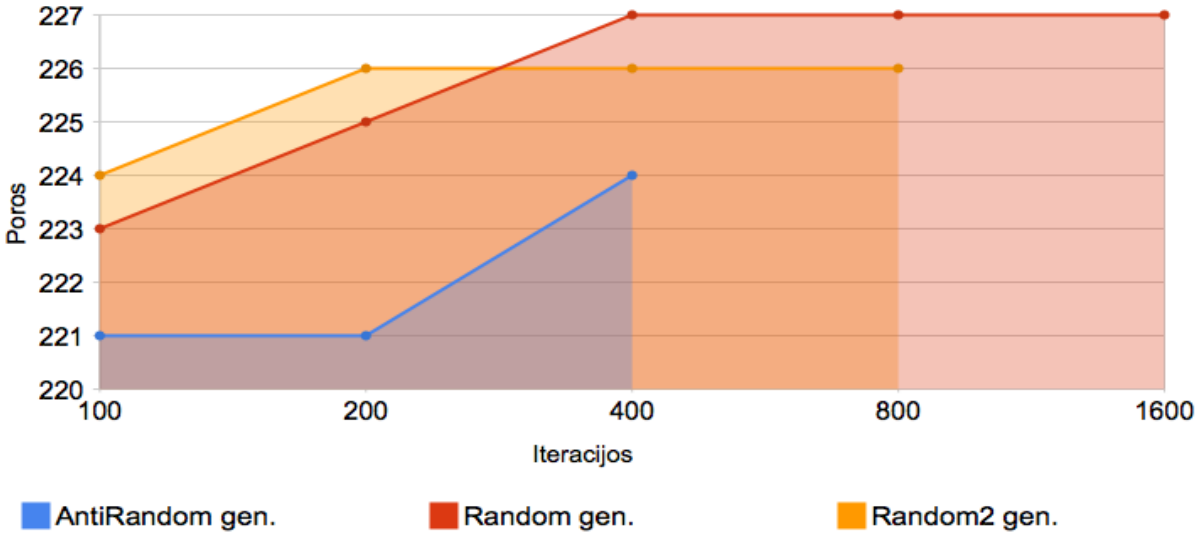
4.4.5.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 100 iteracijų, šis iteracijų skaičius buvo dvigubinamas (200, 400). Generavimui buvo naudojamas Inetl(R) Pentium(R) 4 CPU 3.20 GHz procesorius.



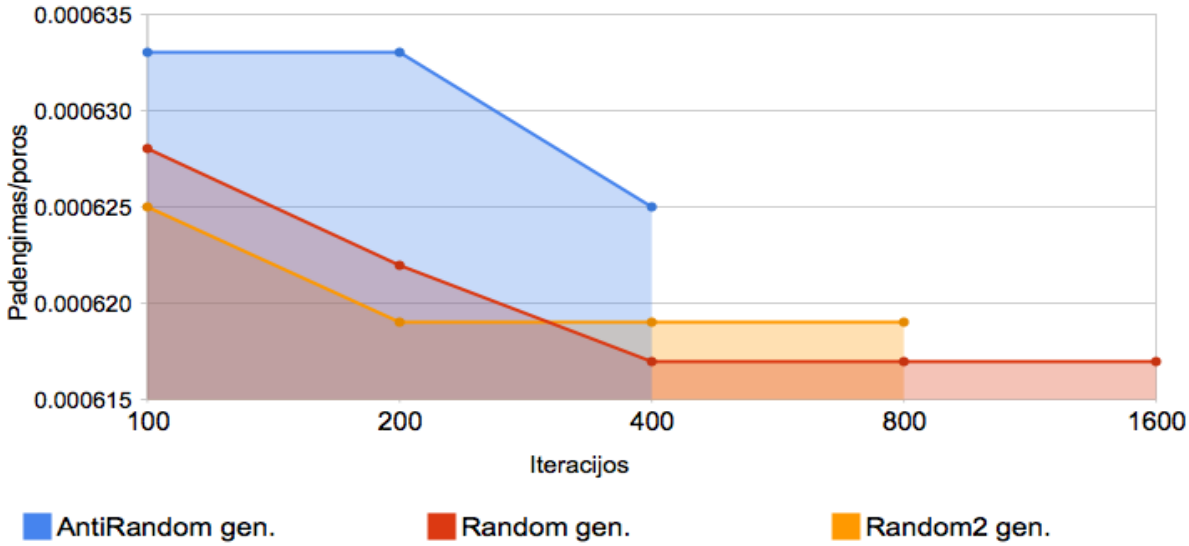
4.4.5.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.5.4. pav.) matome, jog Random ir Random2 generatoriai sugeneruoja ir atrenka daugiau testinių rinkinių nei AntiRandom esant tam pačiam iteracijų kiekiui.



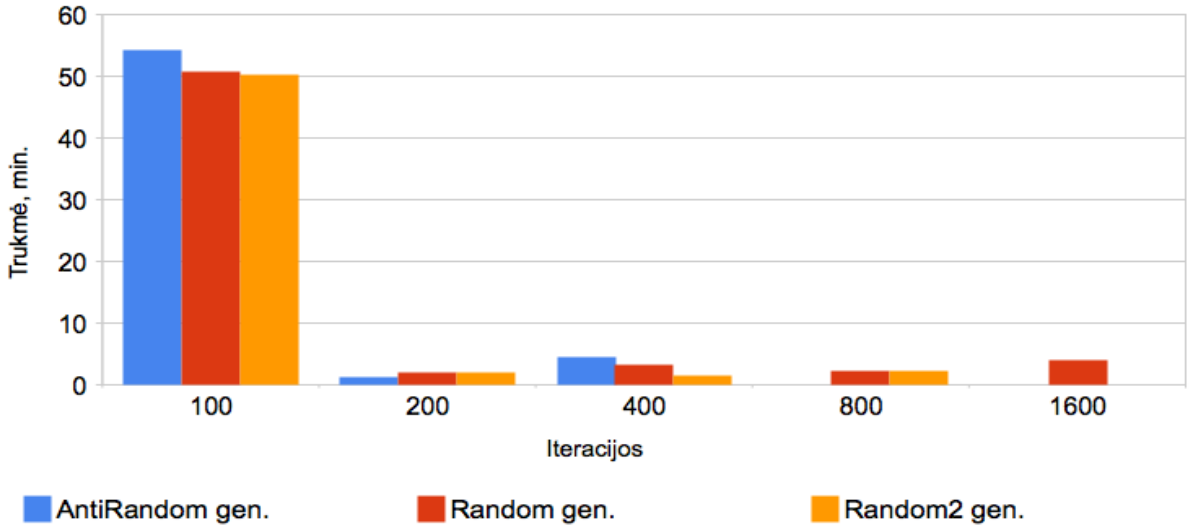
4.4.5.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.5.5. pav.) matome, kad visi generatoriai pasiekia identišką matricos X užpildymą.



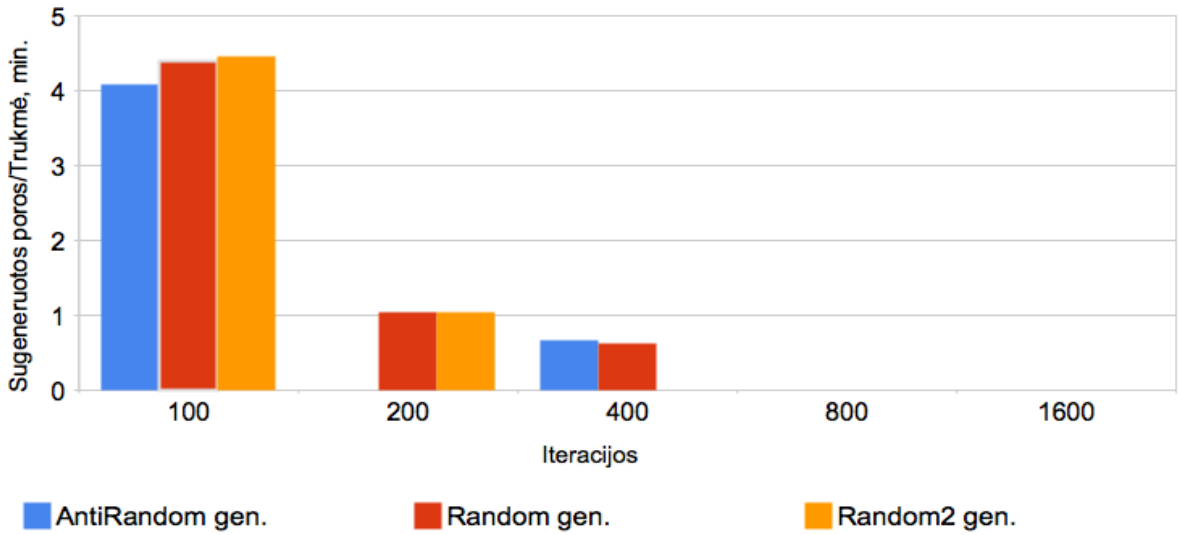
4.4.5.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.4.6. pav.) matome, kad šį kartą Random generatorius generuoja efektyvesnius funkcinius testus nei Random2 ir AntiRandom. AntiRandom pagal tą patį rodiklį tik šiek tiek lenkia Random2.



4.4.5.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.5.7. pav.) matome, kad esant generavimo trukmės yra panašios.



4.4.5.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.5.8. pav.) matome, kad kaip ir ankstesnėje schemoje visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra beveik identiškas. Galime teigti jog šiai schemai generuojamų funkcinių testų generatoriaus tipas didelės įtakos neturi.

4.4.6 S38417 schemos eksperimentinis tyrimas

Gauti rezultatai pateikti 4.4.6.1. - 4.4.6.8. pav.

Savybė	Reikšmė
Schema	s38417
Siuntėjo IP	81.7.118.3
Procesorius	
Komentaras	Random gen. Keisti
Atsiuntimo data	2009-12-06
Atsiuntimo laikas	13:53:57
Iteracijos	100
Vykdymo trukmė	02:12:32
Porų pradžioje	0
Naujų porų	506
Iš viso porų	506
Padengta pradžioje	0/23189504 (0.00%)
Padengta naujai	54629/23189504 (0.24%)
Galutinis padengimas	54629/23189504 (0.24%)
Rezultatų failas	1260100427.log

4.4.6.1. pav. Random generatorius. Pirmas tyrimas.

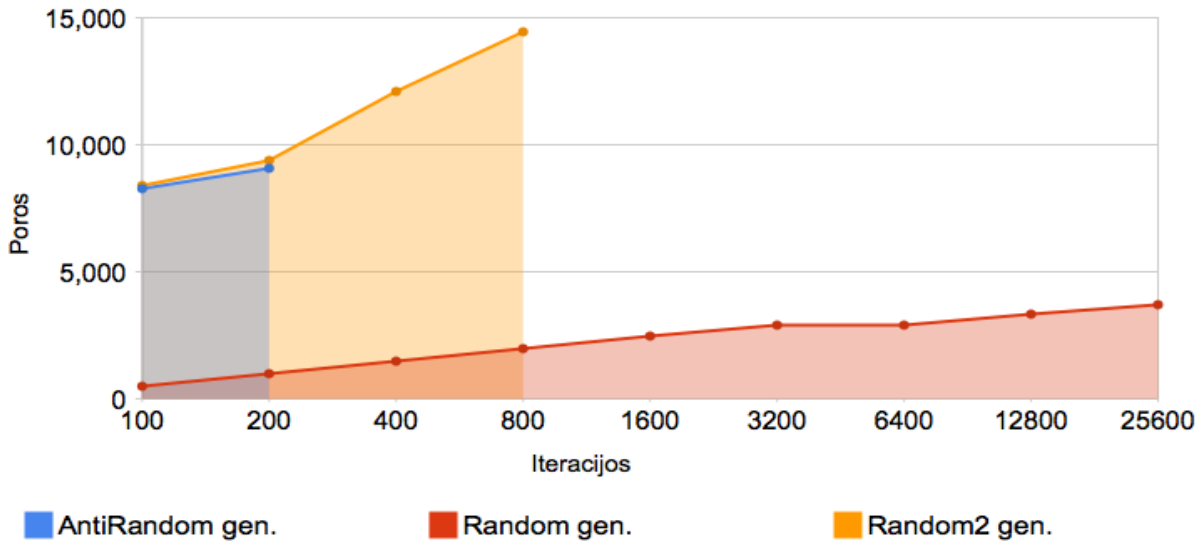
Savybė	Reikšmė
Schema	s38417
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	Random2 gen. Keisti
Atsiuntimo data	2010-02-16
Atsiuntimo laikas	04:56:50
Iteracijos	100
Vykdymo trukmė	36:50:48
Porų pradžioje	0
Naujų porų	8384
Iš viso porų	8384
Padengta pradžioje	0/23189504 (0.00%)
Padengta naujai	188239/23189504 (0.81%)
Galutinis padengimas	188239/23189504 (0.81%)
Rezultatų failas	1266289114.log

4.4.6.2 pav. Random2 generatorius. Pirmas tyrimas.

Savybė	Reikšmė
Schema	s38417
Siuntėjo IP	195.12.182.140
Procesorius	Intel(R) Pentium(R) 4 CPU 3.20GHz
Komentaras	AntiRandom gen. Keisti
Atsiuntimo data	2010-03-29
Atsiuntimo laikas	22:42:20
Iteracijos	100
Vykdymo trukmė	35:38:01
Porų pradžioje	0
Naujų porų	8259
Iš viso porų	8259
Padengta pradžioje	0/23189504 (0.00%)
Padengta naujai	186953/23189504 (0.81%)
Galutinis padengimas	186953/23189504 (0.81%)
Rezultatų failas	1269891706.log

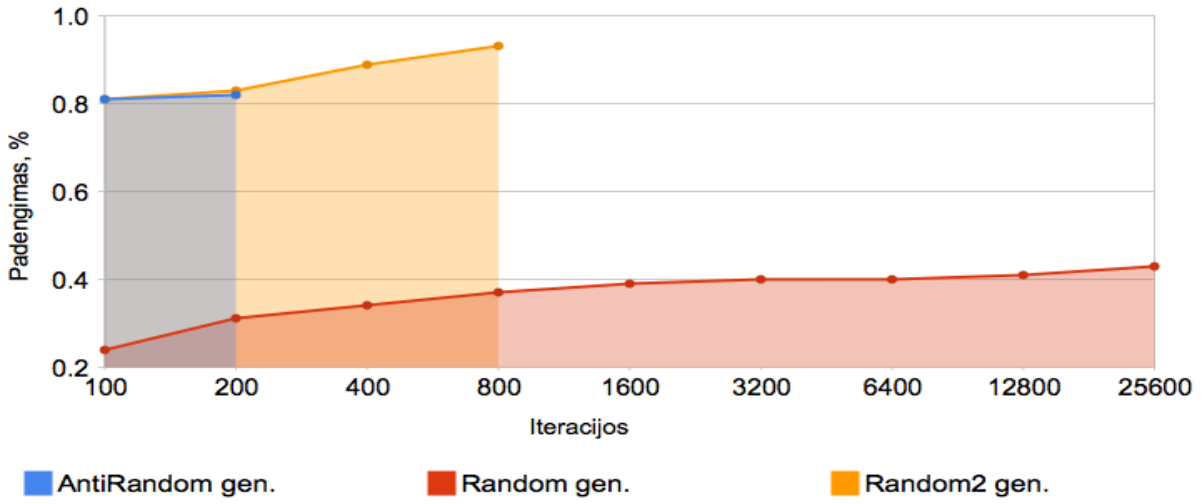
4.4.6.3 pav. AntiRandom generatorius. Pirmas tyrimas.

Generavimas buvo pradedamas nuo 100 iteracijų, šis iteracijų skaičius buvo dvigubinamas (200, 400). Generavimui buvo naudojamas Inetl(R) Pentium(R) 4 CPU 3.20 GHz procesorius.



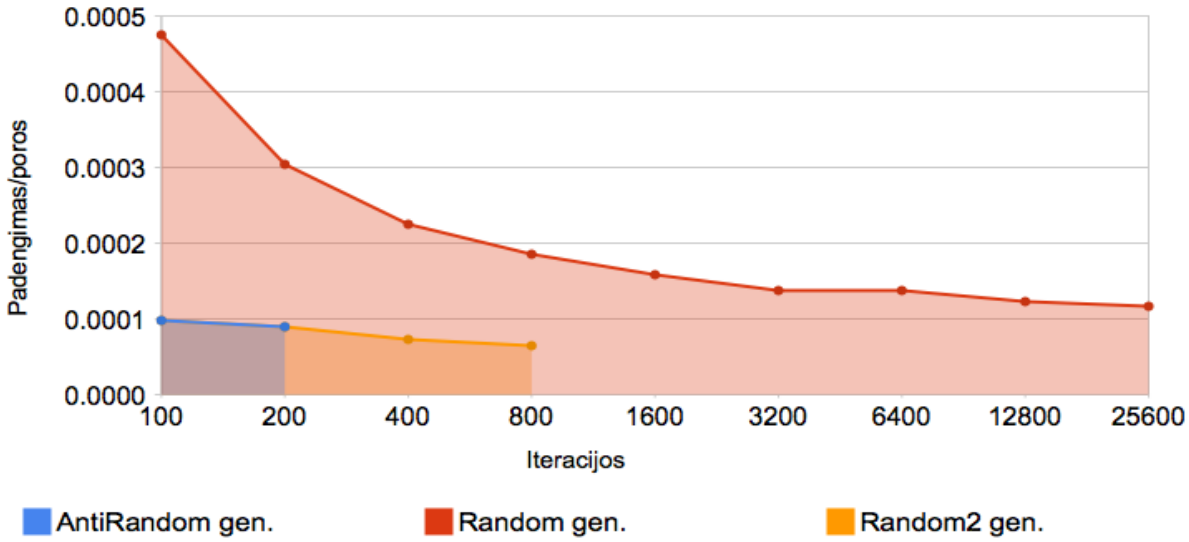
4.4.6.4. pav. Sugeneruotų porų pagal iteracijas diagrama

Iš sugeneruotų porų pagal iteracijos diagramos (4.4.6.4. pav.) matome, jog Random2 ir AntiRandom generatoriai sugeneruoja ir atrenka žymiai daugiau testinių rinkinių nei Random esant tam pačiam iteracijų kiekiui.



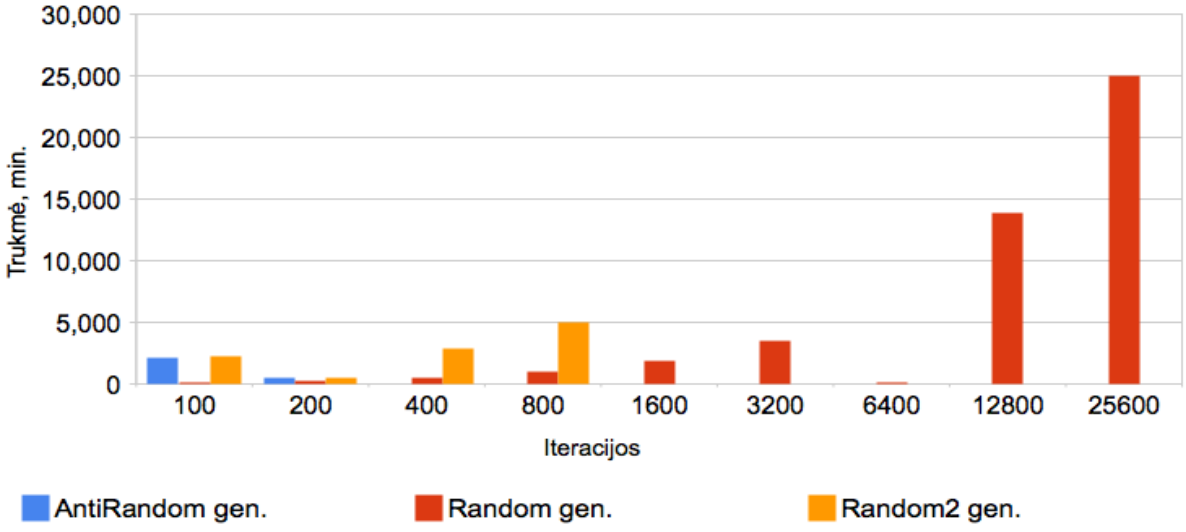
4.4.6.5. pav. Matricos užpildymas pagal iteracijas diagrama

Iš matricos užpildymo pagal iteracijos diagramos (4.4.6.5. pav.) matome, kad Random2 ir AntiRandom užpildo didesnę matricos X dalį nei Random. Verta pastebėti, kad kelis kartus didesnis užpildymas yra pasiekiamas jau nuo generavimo pradžios.



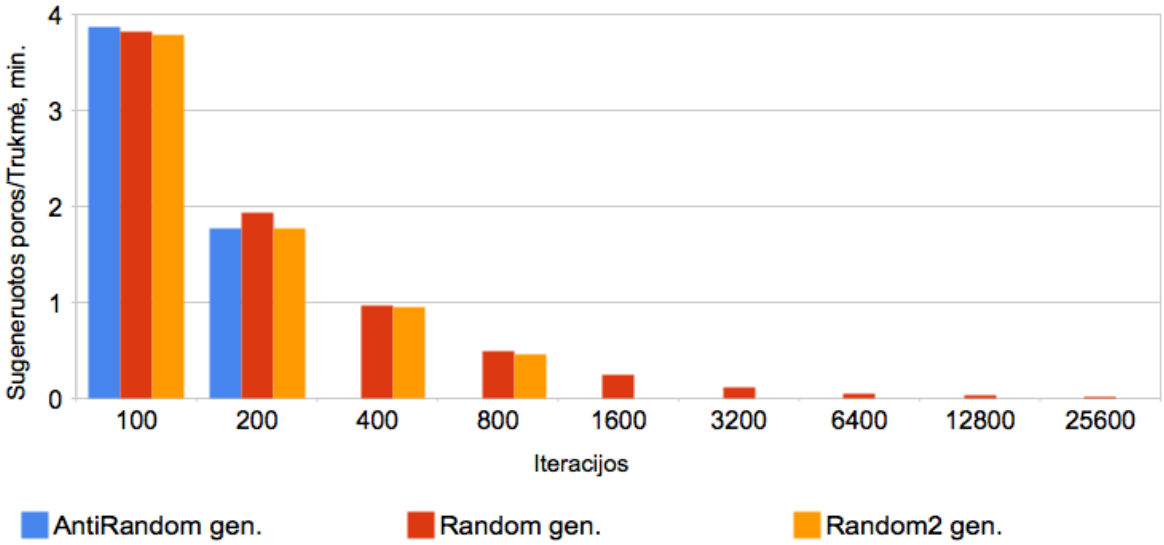
4.4.6.6. pav. Matricos užpildymo ir sugeneruotų porų santykio pagal iteracijas diagrama

Iš matricos užpildymo ir sugeneruotų porų santykio pagal iteracijos diagramos (4.4.6.6. pav.) matome, kad šį kartą Random generatorius generuoja efektyvesnius funkcinius testus nei Random2 ir AntiRandom. AntiRandom ir Random2 santykis yra identiškas. Tačiau reikėtų nepamiršti, kad Random generatoriaus pranašumą įtakoja ir labai didelis matricos X dydis. T.y. AntiRandom ir Random2 generatorių pasiekiamas matricos X užpildymas nėra didelis visos matricos atžvilgiu, tačiau jis didelis Random generatoriaus atžvilgiu.



4.4.6.7. pav. Generavimo trukmės pagal iteracijas diagrama

Iš generavimo trukmės pagal iteracijos diagramos (4.4.6.7. pav.) matome, kad Random generatorius generavimą vykdė trumpiau nei AntiRandom ir Random2.



4.4.6.8. pav. Sugeneruotų porų ir generavimo trukmės santykio pagal iteracijas diagrama

Iš sugeneruotų porų ir generavimo trukmės santykio santykio pagal iteracijos diagramos (4.4.4.8. pav.) matome, kad visų generatorių sugeneruotų ir atrinktų porų skaičius per laiko vienetą yra beveik identiškas. T.y. Generavimo greitis yra vienodas.

4.5 Gautų rezultatų apibendrinimas

4.5.1. lentelėje pateiktas Random, Random2 ir AntiRandom generatorių palyginimas esant vienodam iteracijų kiekiui.

4.5.1. lentelė Generatorių palyginimas esant vienodam iteracijų kiekiui

Schema	Generatorius	Atrinkta porų	Iteracijų kiekis	Trukmė, min	TFC, %
S1196	Random	515	10240000	5 h 44	88.84
	Random2	314	10240000	3 h 29	90.44
	AntiRandom	317	10240000	3 h 53	91.55
S1238	Random	490	2560000	2 h 14	90.41
	Random2	310	2560000	1 h 27	90.41
	AntiRandom	320	2560000	1 h 22	91.30
S13207	Random	2271	6400	10 h 10	94.70
	Random2	3401	6400	17 h 16	97.01
	AntiRandom	3524	6400	25 h 03	97.41
S15850	Random	2779	3200	11 h 44	95.67
	Random2	4065	3200	25 h 40	97.72
	AntiRandom	3857	3200	19 h 36	97.79
S35932	Random	227	400	56	100

	Random2	226	400	53	100
	AntiRandom	224	400	59	100
S38417	Random	1015	200	6 h 36	84.35
	Random2	9362	200	44 h 04	98.53
	AntiRandom	9062	200	43 h 10	98.37

Pagal šios lentelės duomenis galima daryti išvadas apie generatorius esant vienodam iteracijų kiekiui:

- Random2 ir AntiRandom generatoriai esant vienodam iteracijų kiekiui yra pranašesni prieš Random pagal visus parametrus (atrinktų porų kiekis, trukmė, TFC) schemose S1196 ir S1238.
- Generuojant funkcinius testus sudėtingesnėms schemoms (S13207, S15850, S38417) Random generatorius yra pranašesnis pagal atrinktų porų kiekį ir mažesnę generavimo trukmę.
- Pagal TFC klaidų padengimą Random2 visais atvejais lenkia Random;
- AntiRandom generatorius pagal TFC nenusileidžia Random2 ir netgi daugelyje schemų yra didesnis už Random2.

4.5.2. lentelėje pateiktas Random, Random2 ir AntiRandom generatorių palyginimas esant geriausiam atsitiktiniam generavimui.

4.5.2. lentelė Generatorių palyginimas esant geriausiam atsitiktiniam generavimui

Schema	Generatorius	Atrinkta porų	Iteracijų kiekis	Trukmė, min	TFC, %
S1196	Random	370	20000	10	84.34
	Random2	314	20480000	4 h 13	90.44
	AntiRandom	317	10240000	3 h 53	91.55
S1238	AntiRandom	320	2560000	1 h 22	91.30
	Random	344	20000	10	84.63
	AntiRandom	317	1280000	57	91.30
S13207	Random	2326	25600	29 h 29	94.78
	Random2	3643	12800	45 h 20	97.30
	AntiRandom	3524	6400	25 h 03	97.41
S15850	Random	2938	25600	34 h 47	95.77
	Random2	3174	800	8 h 24	96.21
	AntiRandom	3857	3200	19 h 36	97.79
S38417	Random	2458	1600	63 h 11	86.96

	Random2	8384	100	36 h 51	92.23
	AntiRandom	8259	100	35 h 38	92.37

Apibendrinus abi lenteles galima daryti tokias išvadas apie generatorius:

- AntiRandom generatorius pasiekia aukščiausią vėlinimo gedimų padegimą (TFC) iš visų generatorių;
- AntiRandom generatoriaus pagalba galima pasiekti aukštesnį padengimą generuojant mažiau funkcinių testų;
- Mažesnėse schemose (S1196 ir S1238) AntiRandom generatorius atrenka mažiau funkcinių testų, tam kad pasiektų aukštesnį padengimą;
- Didesnėse schemose Random generatorius atrenka mažiau funkcinių testų;

Random2 ir AntiRandom generatorių pasiekiami rezultatai yra panašūs, tačiau AntiRandom generatorius aplenkia Random2 generatorių šiek tiek didesniu vėlinimo gedimų padengimu.

4.6 Išvados

1. Funkciniai testai sugeneruoti AntiRandom ir Random2 generatoriais testuojant mažesnio sudėtingumo schemas yra pranašesni pagal visus parametrus prieš paprastą Random;
2. AntiRandom pralenkia Random2 pagal klaidų padengimą (TFC);
3. Testuojant sudėtingesnes schemas AntiRandom ir Random2 pranašesni prieš paprastą Random tik pagal klaidų padengimą;
4. Didinant iteracijų kiekį generavimo efektyvumas mažėja;
5. Dėl didelio generavimo ir atrinkimo laiko funkcinių testų generavimas sudėtingoms schemoms turėtų būti vykdomas prie nedidelio iteracijų skaičiaus (100-200).

5 IŠVADOS

1. Funkciniai testai nėra susieti su sch emos realizacija. Tai leidžia sukurti universalius testus, kurie remiasi vien testuojamos schemos vykdoma funkcija ir neatsižvelgia į realizaciją.
2. Funkciniai testai leidžia atlikti testavimą ankstyvoje schemos kūrimo stadijoje;
3. Funkcinių testų generavime AntiRandom metodas negali visiškai pakeisti atsitiktinio generavimo, todėl jie turi būti apjungti;
4. Gautų rezultatų kaupimui ir analizei buvo sukurta nuotolinė sistema, kuri priima duomenis iš generatorių kliento kompiuteriuose;
5. Nuotolinėje sistemoje buvo sukurtos įvairios gautų rezultatų atvaizdavimo diagramos, kurių pagalba galima vizualiai matyti vieno ar kito generatoriaus pasiekimus bei juos tarpusavyje lyginti;
6. Funkcinių testų kokybei įvertinti nepakanka įprastinių parametrų, tokių kaip (generavimo trukmė, sugeneruoti ir atrinkti rinkiniai, matricos užpildymas). Todėl buvo išvesti papildomi santykiai: matricos užpildymo ir sugeneruotų porų santykis (parodo generatoriaus efektyvumą), sugeneruotų porų ir trukmės santykis (parodo generavimo greitį);
7. Buvo sukurtas tyrimų planas, pagal kurį buvo vykdomi tyrimai bei registruojami gauti rezultatai;
8. Didžiausią vėlinimo gedimų padengimą pasiekia AntiRandom generatorius, tačiau sudėtingesnėms schemoms šis generatorius funkcinius testus generuoja šiek tiek ilgiau nei Random ir Random2;
9. Funkciniai testai sugeneruoti AntiRandom ir Random2 generatoriais testuojant mažesnio sudėtingumo schemas yra pranašesni lyginant su Random pagal visus parametrus;
10. Dėl didelio generavimo ir atrinkimo laiko funkcinių testų generavimas sudėtingoms schemoms turėtų būti vykdomas prie nedidelio iteracijų skaičiaus (100-200);
11. Pagal atliktus tyrimus yra pradėtas rašyti mokslinis straipsnis.

6 LITERATŪRA

[1] **N. Jha, S. Gupta**, Testing of Digital Systems. *Cambridge University Press*, 2003, 1000.

[2] **V. Jusas, K. Motiejūnas**, Impact of functional delay test compaction on transition fault coverage, *Information Technology And Control, Kaunas, Technologija*, Vol.36, No.2, ISSN 1392-124X, 2007.

[3] **Ž. Tamoševičius**, Save testuojančių schemų testavimas, *Elektronika ir elektrotechnika*, ISSN 1392-1215, 2005. Nr. 7(63).

[4] **V. Jusas, K. Motiejūnas**, Generation of functional delay test with multiple input transitions, *Information Technology And Control, Kaunas, Technologija*, Vol.36, No.3, ISSN 1392-124X, 2007.

[5] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas**, Delay fault models and metrics, *Information Technology And Control, Kaunas, Technologija*, Vol.34, No.4, ISSN 1392-124X, 2005.

[6] **G. M. Luong, D. M. H. Walker**, Test Generation for Global Delay Faults. *Dept. of Computer Science Texas A&M University*.

[7] **D. Gizopoulos**, Advances in Electronic Testing: Challenges and Methodologies, ISBN 0387294082, 9780387294087, 2006

[8] **B. Davis**, The Economics of Automated Testig, *McGraw-Hill: London, UK.*.

[9] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas**, Functional Delay Test Construction Approaches, *Electronics And Electrical Engineering*, ISSN 1392-1215, 2007. No. 2(74).

[10] **A. Virazel, R. David**, Delay Fault Testing: Choosing Between Random SIC and Random MIC Test Sequences, Journal of Elektronic Testing: Theory and Applications, Netherlands, 17, 233-241, 2001.

[11] **Synopsys Inc.**, User manuals For SYNOPSYS ToolSet, *Synopsys, Inc., 2005*.

[12] **V. Abraitis, Ž. Tamoševičius**, The Transition Fault Model of Programmable Logic, *Electronics and Electrical Engineering,* ISSN 1392-1215, 2008. No. 1(81).

[13] **E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas**, Functional Delay Test Quality Assessment At High Level of Abstraction, *Information Technology And Control, Kaunas, Technologija*, Vol.34, No.1, ISSN 1392-124X, 2007.

[14] **A. K. Pramanick, S. M. Reddy**, On the Fault Coverage of Delay Fault Detecting Tests, *Edac,* 334-338, 1990.

[15] **M. Michael, S. Tragoudas**, ATPG Tools for Delay Faults at the Functional Level, *ACT Transactions on Design Automation of Electronics Systems,* Vol. 7, No.1, 2002, 33-57.

[16] **K. T. Cheng, H. C. Chen**, Delay testing for non-robust untestable circuits, *Proc. International Test Conf*., pp. 954-961, 1993.

[17] **A. Krstic, K.T.Cheng**, Delay Fault Testing for VLSI Circuits, ISBN 0792382951, 9780792382959, 1998.

7 TERMINŲ IR SANTRUMPŲ ŽODYNAS

ATPG – automatinis testinių rinkinių generavimas (*angl. Automatic test pattern generation)*

EDA – elektroninių įrenginių projektavimo įrankiai (*angl. Electronic design automation)*

TDF – perėjimo vėlinimo gedimo modelis (*angl. transition delay fault*)

PDF – kelio vėlinimo gedimo modelis (*angl. path delay fault*)

I/O – įvestis / išvestis (*angl. input / output*)

FC – klaidų padengiamumas (*angl. fault coverage*)

DLL – dinaminė nuorodų biblioteka (*angl. Dynamic-link library*)

TFC - perėjimo gedimų padengimas (angl. Transition fault coverage)

FTP – bylų persiuntimo protokolas (*angl. File transfer protocol*)

FAR – greitas AntiRandom (*angl. Fast AntiRandom*)

DBVS – duomenų bazių valdymo sistema

angl. slow-to-rise - lėtai kylantis signalo frontas

angl. slow-to-fall - lėtai krintantis signalo frontas

angl. robust - tiesioginė įtaka

angl. non-robust – netiesioginė įtaka