KAUNO TECHNOLOGIJOS UNIVERSITETAS
PROGRAMŲ INŽINERIJOS KATEDRA

MARIUS RADAVIČIUS

TESTŲ SUDARYMO METODŲ VĖLINIMO GEDIMAMS TYRIMAS

MAGISTRO DARBAS

Darbo vadovas prof. dr. V. Jusas

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

PROGRAMŲ INŽINERIJOS KATEDRA

TESTŲ SUDARYMO METODŲ VĖLINIMO GEDIMAMS TYRIMAS

INFORMATIKOS INŽINERIJA

Vienlustės sistemos

MAGISTRO DARBAS

Magistrantas

Marius Radavičius, IFM-2/5 gr.

2008 m. gegužės mėn. d.

Vadovas

prof. dr.V. Jusas

2008 m. gegužės mėn. d.

Recenzentas doc. A. Misevičius

2008 m. gegužės mėn. d.

KAUNAS, 2008

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

1.

Radavičius Marius Research of Test Generation Methods for Delay Faults: Master's work in informatics engineering / supervisor prof. dr. V. Jusas; Kaunas University of Technology, Faculty of Informatics, Department of Software Engineering – Kaunas, 2008. – 58 pages.

SANTRAUKA

Šio darbo tikslas yra realizuoti ir ištirti funkcinių vėlinimo testų generavimo algoritmus. Gaminant programuojamuosius įrenginius visada atsiranda galimybė jog gali atsirasti vėlinimo klaidų (įėjimo signalas dėl kažkokių priežasčių išeina vėliau nei yra numatyta).

Šiuo metu lustai naudojami civilinėje bei karinėje pramonėje įvairiems gamybos procesams valdyti, pvz: medicinoje. Vėlinimo gedimų testavimai yra labai reikšmingi tiek sistemų saugumo užtikrinime, tiek sistemos patikimumo atžvilgiu. Šiais laikais tokie reikalavimai yra keliami vis didesni, nes žmogaus saugumas yra visų svarbiausia.

Realizuojamas funkcinis vėlinimo testo sudarymo algoritmas pagal pokyčius išėjimuose, kuris remiasi tik programinio prototipo pirminių įėjimų ir pirminių išėjimų reikšmėmis. Sukūrus sistemą, buvo atliktas eksperimentinis jos tyrimas. Gauti panaudoto algoritmo rezultatai yra patenkinami, kurie yra pateikti šio dokumento priede. Eksperimento metu atliekami testinių rinkinių generavimai 24 loginėms schemoms.

Darbe yra 42 panaudoti paveikslėliai, 27 lentelės bei santrumpų žodynas. Medžiaga surinkta iš penkiolikos literatūros šaltinių.

SUMMARY

The object of this work is to research functional test generation methods for delay faults. Manufacturing of programmable chips always has possibility that in those systems will be delay faults that means: input signal for some reasons appear in the output after longer time than the time was definite.

In our days, programmable chips are used in the industry for the management of various civil and military manufacturing processes, for instance medicine. Delay fault testing is very important part for the system safety and trustiness. Today those requests become higher and higher, because human safety is common importance.

Created test generation algorithm of the functional test that is based solely on the primary input values and the primary output values of the programming prototype. System was tested by simple step by step model,(which is presented in the fourth part of this document). The obtained results are useful and acceptable. All results are presented in the appendix of this document. The experiment contains 24 logic schemas processed test generation algorithm.

The work includes of 42 pictures, 27 tables and conceptual dictionary. 15 bibliographical sources have been used.

ĮVADAS

Lustas (angl. *chip*) – mažų gabaritų įtaisas (mikroprocesorius), puslaidininkių kristalas, integrinė mikroschema. Lustai naudojami įvairiose kortelėse (tapatybės kortelėse, mokėjimo kortelėse ir t.t.). Juose modernių technologijų pagalba talpinama pakankamai daug informacijos.

Integrinė mikroschema - Integrinis grandynas (IG) – vientisu technologiniu procesu pagamintas užbaigtas funkcinis įtaisas, susidedantis bent iš keleto neišardomai sujungtų elementų ir hermetizuotas viename korpuse. Elektroniniuose įrenginiuose yra pagrindinis vientisas nedalomas elementas toks, koks įprastinėje technikoje rezistorius, kondensatorius arba tranzistorius. IG susideda iš elementų ir komponentų.

IG elementu vadinama grandyno dalis, kuri atlieka puslaidininkinio elemento (diodo, tranzistoriaus), kondensatoriaus ir t.t. funkciją ir konstruktyviai neatskiriama nuo IG.

IG komponentu vadinama ta grandyno dalis, kuri atlieka elektroninio elemento funkciją, bet iki montavimo yra savarankiškas gaminys.

IG klasifikuojamos pagal gamybos technologiją, integracijos laipsnį, funkcinę paskirtį ir aktyviųjų elementų tipą.

Šiuolaikinių IG gamybos technologijoje yra daug operacijų, paminėsiu keletą iš jų:

- Oksidavimas
- Ėsdinimas
- Fotolitografija
- Difuzija.

Gaminant lustus visada atsiranda galimybė jog gali atsirasti vėlinimo klaidų (įėjimo signalas dėl kažkokių priežasčių išeina vėliau nei yra numatyta).

Šiuo metu lustai naudojami civilinėje bei karinėje pramonėje įvairiems gamybos procesams valdyti, pvz: medicinoje. Dėl šių priežasčių mikroschemoms labai svarbūs tapo patikimumas ir testavimas. Todėl reikia kurti naujus arba kokiu nors būdu pritaikyti esamus schemoms skirtus testavimo metodus.

Testuojant logines schemas galimi du testavimo scenarijai:

1. Gamintojas pats testuoja pagamintą lustą;
2. Vartotojas testuoja lustą po to, kai šis užprogramuojamas vykdyti konkrečią funkciją.

Gamintojai paprastai parduoda jau patikrintus ir funkcionuojančius įrenginius, tačiau sandėliavimo, transportavimo ar eksploatavimo metu programuojamosios logikos įrenginiuose gali atsirasti gedimų. Dėl šios priežasties vartotojo testai taip pat yra aktualūs ir reikalingi.

Tyrimo objektas – atlikti eksperimentus su ITC99 tipo loginėmis schemomis.

Tyrimo tikslas – realizuoti algoritmą ir sukurti programinę įrangą, kuri pagal duotus algoritmus sugeneruotų tiriamosioms schemoms funkcinius vėlinimu testus.

Tyrimo uždaviniai:

1. Apžvelgti veikiančias programinės įrangos sistemas, kurios gali sudaryti funkcinius vėlinimo testus.
2. Padaryti automatinio testavimo ir „draugiškos“ vartotojo sąsajos programą.
3. Pateikti gautus rezultatus.

Darbas sudarytas iš penkių dalių.

Analitinė dalis. Aptariamas apibendrintas vėlinimo gedimų testavimo modelis, vėlinimo gedimų savybės, funkcinių vėlinimo testų generavimas bei jų kokybės vertinimo kriterijai ir esminis funkcinio vėlinimo testo generavimas.

Projektinė dalis. Magistratūros studijų metu sukurtos įrangos techninė-projektinė dokumentacija. Joje pateikiamas pasirinkto sprendimo realizacijos kelias.

Tyrimo dalis. Sukurtos įrangos kokybės analizė. Sistemos kokybės tobulinimo galimybės.

Eksperimentinė dalis. Eksperimentinėje dalyje yra atliekamas magistratūros studijų metu sukurtos ir įdiegtos sistemos bei jos patobulinimų eksperimentinis tyrimas. Svarbiausi eksperimentų rezultatai pateikiami prieduose.

Išvados. Šioje dalyje pateikta apibendrinta informacija apie vėlinimo gedimų testų sudarymo sistemos funkcionalumą ir išskirtinumą bei tolimesnes jos plėtojimo galimybes.

Tyrimo metodika:

1. Mokslinės literatūros analizė;
2. Publikacijų ir pranešimų analizė.

2. ANALITINĖ DALIS

Aptariamas apibendrintas vėlinimo gedimų testavimo modelis, vėlinimo gedimų savybės, funkcinių vėlinimo testų generavimas bei jų kokybės vertinimo kriterijai ir funkcinio vėlinimo testo generavimas.

2.1. TYRIMO SRITIS

Funkcinis testas remiasi schemoje vykdoma funkcija, kuri gali būti realizuota daugeliu būdu. Schemos galimi gedimai priklauso nuo schemos realizacijos. Praktikoje apsiribojama schemos konkrečios realizacijos galimais gedimais ir testas sudaromas tik vienos realizacijos gedimams. Testą galima sudaryti tik turint konkrečią schemos realizaciją. Tuo tarpu funkcinis testas nepririštas prie konkrečios realizacijos. Tokiu atveju funkcinis testas turi tikrinti visas galimas realizacijas kas yra žymiai sudėtingesnė problema. Pagrindinė problema su kuria susiduriama sudarant funkcinį testą kaip vertinti funkcinio testo kokybę neturint konkrečios schemos realizacijos. Šiuo metu jau yra pasiūlyti funkcinių gedimų modeliai, kurie leidžia gauti funkcinį testą, kuris tikrina daugiau kaip 99% bet kokios realizacijos konstantinių gedimų. Bet toks funkcinis testas yra keletą kartų ilgesnis negu testas paskaičiuotas konkrečiai schemos realizacijai. Todėl funkcinis testas dar turi būti minimizuotas konkrečios realizacijos gedimų atžvilgiu išmetant iš jo testinius rinkinius, kurie netikrina naujų konkrečios realizacijos gedimų. Be to neatmetama galimybė esant aukštiems testo pilnumo reikalavimams funkcinį testą papildyti netikrinamų konkrečios realizacijos gedimų atžvilgiu. Funkcinio testo naudojimo pagrindinis privalumas yra tas, kad funkcinį testą galima projektuoti pradiniuose schemos projektavimo etapuose pagal schemos programinį prototipą lygiagrečiai su schemos sintezės procesu. Tuo tarpu funkcinio testo minimizavimas besiremiantis jau po sintezės gauta konkrečia realizacija nėra darbui imlus veiksmas ir mažai įtakoja bendrą schemos sintezės ir testų projektavimo trukmę.

Didėjant integracijai bei schemos darbiniams dažniams vis didesnę įtaką įgauna vėlinimo gedimai.Vėlinimo gedimai tikrinami testinių rinkinių pora, kur pirmasis testinis rinkinis suformuoja pradines įėjimų reikšmes, o antras testinis rinkinys formuoja pokyčius schemos įėjimuose, o reakcija išėjimuose po antro testinio rinkinio matuojama po apibrėžtos laiko trukmės. Jeigu dėl vėlinimo gedimų signalai išėjimuose nespėja pasikeisti tai matavimo metu užfiksuojamos pirmu testiniu rinkiniu nustatytos signalų reikšmės, kas indikuoja apie schemoje egzistuojančius vėlinimo gedimus. Funkciniu vėlinimo testu vadinsime testą, kuris buvo gautas nesiremiant schemos struktūra.

Iki šiol mažai tyrinėta, kaip gauti funkcinį testą vėlinimo gedimams ir kaip funkcinis testas tikrina konkrečios realizacijos vėlinimo gedimus. Be to mums nėra žinoma pasiūlytų funkcinio testo kokybės vertinimo kriterijų vėlinimo gedimų atžvilgiu. Naudojami ir ištirti funkcinio testo kokybės kriterijai

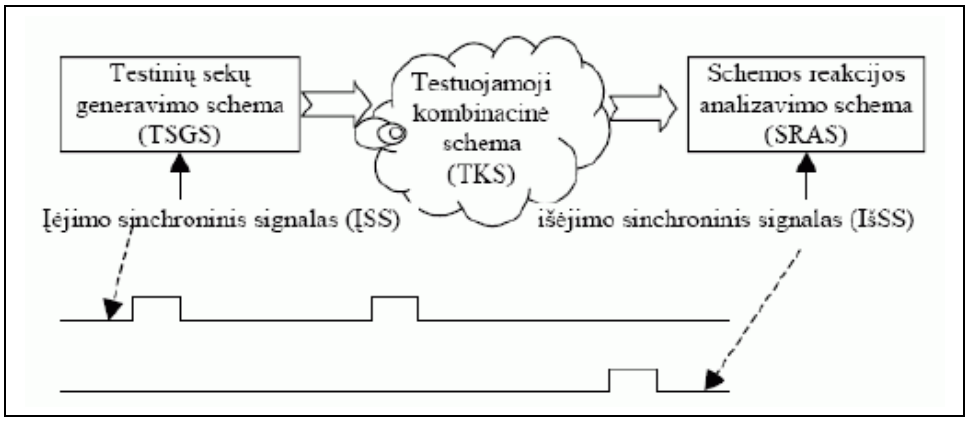
konstantinių gedimų atžvilgiu nėra labiausiai tinkami vėlinimo gedimams, nes vėlinimams reikia vertinti testinių rinkinių poras. Šis darbas skirtas minėtoms problemoms spręsti.

2.2. APIBENDRINTAS VĖLINIMO GEDIMŲ TESTAVIMO MODELIS

Apibendrintas modelis, skirtas vėlinimo gedimams testuoti, pavaizduotas 1 pav. Testinių sekų generavimo schemoje (TSGS) surašytos testinių vektorių poros (TVP), kuriomis sugeneruojamas signalo fronto sklidimas keliu, kuriame reikia patikrinti vėlinimo gedimą. Pirmu įėjimo sinchroninio signalo (ĮSS) frontu kombinacinės schemos (TKS) įėjimuose pateikiamas pirmasis testinis vektorius iš TVP. Tai atlikus, palaukiama (laiko tarpą, ne trumpesnį už laiko tarpą tarp dviejų gretimų sinchroninio signalo, kuris yra siunčiamas į realiai veikiančią schemą, impulsų), kol TKS išėjime gaunamas stabilus išėjimo signalas. Tuomet siunčiamas antrasis ĮSS frontas, kurio metu TSGS schema pateikia antrąjį testinį vektorių iš TVP.

Antrasis testinis vektorius iš TVP sudarytas taip, kad generuotų signalų pasikeitimus iš loginio „0“ į loginį „1“ (arba atvirkščiai) TSGS išėjimuose, bet tik tuose, kurie gali sugeneruoti fronto sklidimą reikiamu patikrinti keliu. Po antrojo ĮSS fronto palaukiama laiko tarpą, lygų intervalui tarp dviejų gretimų sinchroninio signalo impulsų, kurie siunčiami realiai veikiančiai schemai, ir tuomet sugeneruojamas išėjimo schemos sinchroninis signalas (IšSS). IšSS signalo momentu schemos reakcijos analizavimo schema (SRAS) analizuoja TKS schemos išėjimo vektorių ir nustato, ar yra vėlinimo gedimas testuojamajame kelyje, ar nėra.

Schemos TKS įėjime sugeneruotas signalo pasikeitimas nebūtinai gali būti perduotas į TKS išėjimą. Visiškai įmanoma, kad TKS įėjime sugeneruotas signalo fronto pasikeitimas kur nors testuojamosios kombinacinės schemos viduje dėl loginių operacijų „užges“, t.y. nesugeneruos signalo pasikeitimo TKS išėjime.



1. pav. Aparatinis modelis, skirtas vėlinimo gedimams testuoti
(http://www.ee.ktu.lt/journal/2005/7/Tamosevicius.pdf)

Laikas, reikiamas signalui sklisti iš testuojamosios schemos įėjimo į išėjimą, yra lygus kiekvieno loginio elemento, esančio signalo sklidimo kelyje, vėlinimo trukmių sumai. Šiame modelyje taip pat tariama, kad kiekvieno elemento vėlinimo trukmė gali labai skirtis priklausomai nuo aplinkos temperatūros, maitinimo įtampos svyravimų ir kitų veiksnių.

2.3. VĖLINIMO GEDIMAI

Kombinacinės sinchroninės schemos vėlinimo gedimų testavimu vadinamas tos schemos tikrinimas, ar ji gali perduoti signalo pasikeitimą į išėjimą per laiką, neviršijantį laiko tarpo, lygaus intervalui tarp dviejų gretimų sinchroninio signalo impulsų, kurie siunčiami realiai veikiančiai schemai. Jei toks testas duoda neigiamą atsakymą, tai reiškia, kad tikrinamoji schema turi kelio vėlinimo gedimą.

Kelio vėlinimo gedimas gali atsirasti bet kurioje elektroninėje schemoje. Tai gali sukelti net ir vienas tokios schemos elementas, esantis tikrinamajame kelyje, kurio realus vėlinimas viršija nurodytą vėlinimo trukmę jo specifikacijoje. Tačiau gali atsitikti ir taip, kad schema veiks korektiškai, nors tikrinamąjį kelią sudarys elementai, kurių vėlinimo trukmės viršys nurodytas specifikacijoje.

Taip pat negalima teigti, kad TKS neturės kelio vėlinimo gedimo, nors visi schemą sudarantys elementai tenkins jų specifikacijoje nurodytas vėlinimų trukmes. Taip gali atsitikti tokiu atveju, jei tikrinamajame kelyje pasitaikys tokių elementų, kurių vėlinimai bus artimi leidžiamiems pagal specifikaciją vėlinimo trukmių maksimumams, bet jų dar neviršys. Tačiau suminis tokių elementų grandinėlės vėlinimas gali būti didesnis už sinchroninio signalo periodą. Taigi, schema tokiu atveju turės kelio vėlinimo gedimą, nors visi elementai bus geri.

Trys vėlinimo gedimų rūšis:

1. Kelio vėlinimo gedimas, kurio priežastis – suminis tą kelią sudarančių elementų vėlinimas;
2. Kelio vėlinimo gedimas, kurio priežastis – bent vienas elementas, kurio vėlinimas daug didesnis, nei numatyta jo specifikacijoje ir dėl to sukeliantis vėlinimo gedimą bent viename kelyje.

3. Perdavimo kelio vėlinimai - galimybė rasti gamybos metu atsiradusius defektus – pavyzdžiui, neteisingai padengtas oksido sluoksnis dėl per mažo storio gali sukelti didesnį negu reikia vėlinimą.

2.4. VĖLINIMO GEDIMŲ SAVYBĖS

Savybės, būdingos kelių vėlinimų gedimams:

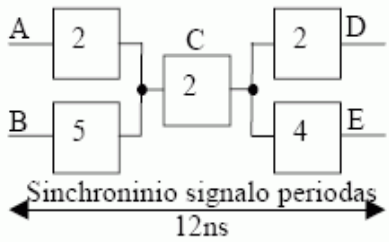
1. Net jeigu kelyje, kurį reikia patikrinti, vėlinimo gedimų yra daugiau nei vienas, tai netrukdo testuoti ir nustatyti, ar šiame kelyje yra vėlinimo gedimų. Taip yra todėl, kad tas kelias testuojamas neatsižvelgiant į kiekvieno elemento vėlinimo trukmę atskirai, todėl šiuo metodu išaiškinama suminė vėlinimo gedimo nagrinėjamame kelyje galimybė.
2. Jei testinių sekų generavimo schemoje (SRAS 1 pav.) schema duoda požymį, kad testuojamame kelyje yra gedimas, neįmanoma pasakyti, būtent kas sukėlė šį gedimą. Bet kuri elementų grandinėlė, kuri užsibaigia schemos išėjimu ir kurioje sklinda signalas su P simboliu, gali turėti tokį gedimą. Taip yra todėl, kad elementų grandinėlės gali susijungti ir išsiskirti. Be to, toks gedimas gali būti sukeltas ne tik kurio nors vieno elemento, esančio testuojamajame kelyje, bet jį gali sukelti ir suminis dviejų ir daugiau elementų vėlinimas.
3. Jei testinis poveikis gali testuoti kelią vėlinimo gedimams, tai testinis poveikis, atlikus toliau aprašytą transformaciją, taip pat gali testuoti tą kelią, ar jame nėra vėlinimo gedimų: pirmojo testinio vektoriaus iš testinį poveikį sudarančios poros vertės, turinčios simbolį P, pakeičiamos į vertę su simboliu S. Kadangi antrasis testinis vektorius iš testinį poveikį sudarančios poros lieka nepakeistas, tai iš to išeina, kad testinis poveikis nepraranda galimybės testuoti kelią, atlikus tokią transformaciją. Testinio vektoriaus vertės su simboliu S nekeičiamos.
4. Jei kiekvienas kelias testuotas nepriklausomai nuo elementų vėlinimų, jis būtinai yra testuotas ir vėlinimo gedimams. Taigi anksčiau aprašytu testuotų kelių identifikavimo metodu galima surinkti mažiausią testinių poveikių aibę, iki galo testuojančią schemą kelių vėlinimo gedimų prasme.
5. Procedūrą, nustatančią, ar testinis rinkinys gali patikrinti vieno elemento gedimą, pasireiškiantį dideliu vėlinimu, aprašo. Didelis vėlinimas – tai toks vėlinimas, kai sugedęs vienas elementas taip paveikia visus kelius, kuriuose jis yra, kad visi tie keliai turi kelio vėlinimo gedimą. Naudojantis čia aprašytu modeliu, tokie kelio vėlinimo gedimai yra būtinai aptinkami.

6. Naudojantis šiuo modeliu taip pat galima aptikti ir loginius elementų gedimus. Būtina sąlyga – loginis elemento gedimas testuojamosios schemos išėjimo vertę turi paveikti taip, kad ji būtų priešinga nei tada, kai schema gera.
7. Jei schemoje yra loginė klaida (kuri pasireiškia tuo, kad kokio nors elemento išėjime yra pastovus loginis 0 arba 1), tai kelio vėlinimo gedimas gali būti ir neaptiktas.

2.5. RIZIKOS FAKTORIŲ ANALIZĖ

Kadangi schemoje, kurioje norime testuoti kelių vėlinimo gedimus, gali būti milžiniškas skaičius kelių, būtina mažinti testuojamų kelių skaičių ir kartu išlaikyti kuo didesnę gedimų aptikimo tikimybę. Toliau aptarsime kelių parinkimo galimybes. Taip pat bus aptariami sunkumai, iškylantys tiesiogiai modeliuojant elementų vėlinimo gedimus.

Kombinacinė schema neturi vėlinimo gedimų tuomet ir tik tuomet, kai joje nėra kelių vėlinimo gedimų. Tačiau loginių elementų faktinių vėlinimo gedimų analizė rodo, kad pakanka testuoti nedidelį visų kombinacinės schemos kelių poaibį tam, kad su patenkinama tikimybe būtų galima pasakyti, ar testuojamojoje schemoje yra kelių vėlinimo gedimų, ar ne.



	Kelias	Sklidimo laikas	Klaidos aptinkamumas
Be klaidų	ACD	6	—
	ACE	8	—
	BCD	9	—
	BCE	11	—
Klaida elemente C: +2	ACD	8	Ne
	ACE	10	Ne
	BCD	11	Ne
	BCE	13	Taip
Klaida elemente C: +7	ACD	13	Taip
	ACE	15	Taip
	BCD	16	Taip
	BCE	18	Taip

2. pav. Elemento vėlinimo pokyčio įtaka klaidų aptinkamumui (http://www.ee.ktu.lt/journal/2005/7/Tamosevicius.pdf)

Kaip keičiasi tikimybė, kad elementas sukels kelio vėlinimo gedimą, kai elemento vėlinimas viršija vėlinimą, nurodytą jo specifikacijoje, parodoma.

Tarkime, kad kelio vėlinimo atsarga apibrėžiama kaip skirtumas tarp išėjimo schemos sinchroninio signalo fronto (1 pav.) ir tarp keliu sklindančio signalo paskutinio pokyčio schemos išėjime. Schemos, kurioje nėra kelių vėlinimo gedimų, visų kelių vėlinimo atsargos bus teigiamos. Kelyje, kuriame yra vėlinimo gedimas, vėlinimo atsarga bus neigiama.

Patyrinėkime kombinacinę schemą, pateiktą 2 pav. Per C elementą eina keturi keliai. Jei elementas C turi vėlinimo gedimą, C elemento vėlinimo atsargos dydis turi įtakos kelio BCE vėlinimo atsargai. Šio kelio vėlinimo atsargos dydis, be kelio vėlinimo gedimo, yra 12 – (5 + 2 + 4) = +1. Vėlinimo gedimas elemente C neįtakos kelio BCE vėlinimui tol, kol elemento vėlinimo atsarga neviršija 3.

Tokiu atveju tik kelias BCE turi neigiamą kelio vėlinimo atsargą, todėl šį gedimą galima aptikti tik testiniais poveikiais, kurie tikrina kelią BCE.

Kelių, turinčių vėlinimo gedimų, skaičius didėja didėjant neigiamai elemento C vėlinimo atsargai. Bendruoju atveju daug lengviau aptikti gedimą, kai neigiamos vėlinimo atsargos modulis kuo didesnis. Dėl šios priežasties labai nepatogu naudoti vėlinimo gedimų modelį, kuriame vertinami tik atskirų elementų vėlinimo gedimai.

Kiekvienam schemos elementui suradus per jį einantį kelią, kurio vėlinimo atsarga yra mažiausia, ir testavus tik šiuos kelius, ir nustačius, kad juose nėra kelių vėlinimo gedimų, su nemaža tikimybe galima teigti, kad schemoje kelių vėlinimo gedimų nėra. Šiai tikimybei padidinti galima imti daugiau kelių, einančių per tą patį elementą, nors jo vėlinimo atsarga ir nėra mažiausia. Kelio vėlinimo atsarga ir būtų tas kriterijus, kuriuo remiantis atrenkami keliai, taip pat juos tikrinantys testiniai poveikiai kelių vėlinimo gedimų paieškai.

2.6. FUNKCINIŲ VĖLINIMO TESTŲ GENERAVIMO METODŲ ANALIZĖ

Funkcinis testas yra parengiamas testavimo inžinieriaus pagal įrenginio specifikaciją. Funkcinis testas yra naudojamas tolimesnių projektavimo žingsnių verifikavimui bei naudojamas parengiant apdirbimo testą. Testavimo inžinieriaus paruoštas funkcinis testas tikrina dažniausiai naudojamas funkcijas, bet kaip taisyklė negali užtikrinti visų ventilinio lygio konstantinių gedimų (stuck-at - konstantiniai gedimai) patikrinimo. Todėl susintezavus įrenginį klaidų simuliacijos pagalba sudaromas sąrašas netikrinamų gedimų ir juos tikrinantys testai generuojami deterministiniais metodais. Bet koks

funkcinis testas gali būti naudojamas vėlinimo gedimų testavimui, kai testinio rinkinio reakcijos (išskyrus pirmą testinį rinkinį) yra matuojamos po apibrėžtos laiko trukmės.Taip pat gali būti sudarytos testinių rinkinių poros pakartojant tą patį rinkinį (išskyrus pirmą) du kartus. Tuo atvejų rinkinių poras sudarytų pirmas ir antras rinkiniai, antras ir trečias rinkiniai ir taip toliau.

Įrenginio projektavimo metu pagal specifikaciją sudaromas įrenginio programinis prototipas. Programinis prototipas imituoja įrenginio darbą ir duotiems poveikiams gali paskaičiuoti išėjimų reakcijas ir traktuojamas kaip įrenginio juodos dėžės modelis. Įrenginio prototipo naudojimas sudaro galimybę automatizuoti funkcinio testo sudarymą atrenkant poveikius pagal pasirinktus kriterijus iš atsitiktinai arba kitaip sugeneruotų įėjimo poveikių. Automatizuoto funkcinio testo sudarymo metu siekiama atrinkti testinius poveikius tikrinančius kuo daugiau susintezuoto įrenginio ventilinio lygio gedimų. Buvo pasiūlyta įvairių funkcinių gedimų modelių juodos dėžės modeliams, kurie nevertina įrenginio programinio prototipo teksto bet remiasi tik poveikių reakcijomis. Plačiausiai žinomas susijungimo klaidų modelis užtikrina išsamų sintezuoto modulio ventilinio lygio konstantinių gedimų patikrinimą, tačiau gaunamas testo ilgis didėja eksponentiškai didėjant schemos apimčiai. Todėl pasiūlyti metodai ir modeliai realiai gali būti taikomi tik moduliams iki keliolikos įėjimų ir išėjimų ir praktiškai tenka nagrinėti įrenginius kaip susidedančius iš sudėtingų modulių, ko pasėkoje komplikuojasi įrenginio išsamus ventilinio lygio konstantinių gedimų patikrinimas. Funkcinio testo gauto pagal susijungimo gedimų modelį galimybės tikrinti vėlinimo gedimus nėra išanalizuotos.

Funkciniai gedimų modeliai „kojų poros“ (angl.pin pairs – PP) ir „trilypių kojų“ (angl. pin triplets - PT) leidžia gauti funkcinį testą naudojant programinį prototipą ankstyvuose projektavimo etapuose, kol dar nesukurtas įrenginio sintezuojamas aprašas. Funkcinis vėlinimo testas, gautas iš testo tikrinančio PP gedimus, tikrina vidutiniškai daugiau kaip 95 procentus perdavimo (transition) gedimų. Funkcinio vėlinimo testo sudarymo metodų iš PP testo tyrimas parodė, kad galima sutrumpinti funkcinio testo ilgį ir pasiekti perdavimo gedimų patikrinimą vidutiniškai iki 98 procentų. Šiame straipsnyje nagrinėjamas funkcinio vėlinimo testo generavimas papildomai įvertinant netiesioginę įėjimo įtaką į išėjimus.

Turint įrenginio specifikaciją ir ją verifikavus programinio prototipo pagalba, sudaromas sintezuojamas įrenginio aprašas. Funkcinio testo sudarymui literatūroje pasiūlyta daug funkcinių gedimų modelių, kurie remiasi elgsenos aprašo tekstu. Naudojant įvairius sudėtingus funkcinio testo generavimo metodus pasiekiamas nuo 70 iki 91 procento ventilinio lygio konstantinių gedimų patikrinimas. Funkcinio vėlinimo testo sudarymas pagal sintezuojamą aprašą nėra nagrinėtas.

Šio darbo tyrimo objektu yra funkcinio vėlinimo testo generavimas pagal įrenginio programinį prototipą remiantis tik įėjimo poveikių reakcijomis.

Darbo tikslas sudaryti automatizuotus testo generavimo pagal programinį prototipą metodus leidžiančius patikrinti kuo daugiau susintezuoto įrenginio ventilinio lygio perdavimo gedimų ir kuo trumpesniu funkciniu testu. Funkcinis vėlinimo testas tikrinantis daugumą susintezuoto įrenginio ventilinio lygio perdavimo gedimų verifikavimo metu leidžia aptikti daugiau projektavimo klaidų ir iš jo perdavimo klaidų simuliacijos pagalba išmetus rinkinius, kurie nieko naujo netikrina konkrečioje įrenginio realizacijoje, tokį testą galima tiesiog naudoti kaip gamybinis vėlinimo testą. Tai leidžia projektavimo procese visai atsisakyti testų generavimo ventilinio lygio perdavimo gedimams etapo arba jį sutrumpinti, ir jei prireikia iš funkcinio vėlinimo testo gautą gamybinį vėlinimo testą dar papildyti. Tuo tarpu, trumpesnis funkcinis vėlinimo testas trumpina ir įrenginio verifikavimo trukmę bei supaprastina funkcinio vėlinimo testo adaptavimą klaidų simuliacijos pagalba konkrečiai realizacijai.

2.7. FUNKCINIO VĖLINIMO TESTO KOKYBĖS VERTINIMO KRITERIJAI

Vėlinimo gedimai tikrinami rinkinių pora. Rinkinių poros signalų reikšmės apsprendžia, kurie įėjimai keičiasi ir kurie išlieka pastovūs. Gali keistis vienas ar daugiau įėjimų. Pokyčiai įėjimuose, kurie iššaukia pokyčius ir išėjimuose, gali patikrinti vėlinimo gedimus. Pokyčiai iš schemos įėjimų į išėjimus perduodami schemos keliais. Vėlinimo gedimų tikrinimui tikslinga pokyčius iš įėjimų į išėjimus perduoti kuo ilgesniais ir kuo įvairesniais keliais. Jeigu pokytis įėjime įtakoja pokytį išėjime, tai galima daryti prielaidą, kad gali būti patikrinti kai kurie vėlinimo gedimai. Funkcinio testo kokybės vertinimui svarbu nustatyti, koks įėjimo pokytis kokius išėjimus įtakoja. Kai keičiasi daug įėjimų tai nėra paprastas uždavinys. Todėl tikslinga nagrinėti atskirų įėjimo pokyčių panaikinimo įtaką išėjimo pokyčiams. Kalbėsime apie atskiro įėjimo pokyčio panaikinimą, kai šio įėjimo antro rinkinio reikšmė prilyginama pirmo rinkinio reikšmei.

Laikykime, kad turime rinkinių porą {1010,0111}, kur keičiasi pirmo, antro ir ketvirto įėjimų reikšmės. Panaikinant pokytį pirmam įėjime gauname rinkinių porą {1010,1111}, panaikinant pokytį antram įėjime gauname rinkinių porą {1010,0011}, o panaikinant pokytį ketvirtam įėjime gauname rinkinių porą {1010,0110}. Įėjimo pokyčio panaikinimas gali sąlygoti ir kai kurių pokyčių išnykimą išėjimuose. Jei taip atsitinka, tai galima tvirtinti, kad įėjimas, kuriam buvo panaikintas pokytis įtakoja išėjimą, kuriam pokytis išnyko. Įtaka pasireiškia schemos keliais. Čia galima išskirti keturis atvejus: įėjimo pokyčio 0→1 panaikinimas sąlygoja išėjimo pokyčio 0→1 išnykimą, įėjimo pokyčio 0→1 panaikinimas sąlygoja išėjimo pokyčio 1→0 išnykimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja išėjimo pokyčio 0→1 išnykimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja išėjimo pokyčio 1→0 išnykimą. Kadangi vieno įėjimo pokyčio panaikinimas tiesiogiai sąlygoja pokyčio išnykimą išėjime, tai

tokią įtaką pagal analogiją su schemos kelių tikrinimu vadinsime tiesiogine (robust) įtaka. Gali būti ir netiesioginis įtakojimas, kai įėjimo pokyčio panaikinimas iššaukia papildomo išėjimo pokyčio atsiradimą. Tai reiškia, kad įėjimo pokytis blokavo kito pokyčio įtaką į išėjimą ir pokytį panaikinus išnyko šis blokavimas, o išėjime atsirado naujas pokytis. Čia taip pat galima išskirti keturis atvejus: įėjimo pokyčio 0→1 panaikinimas sąlygoja papildomo išėjimo pokyčio 0→1 atsiradimą, įėjimo pokyčio 0→1 panaikinimas sąlygoja papildomo išėjimo pokyčio 1→0 atsiradimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja papildomo išėjimo pokyčio 0→1 atsiradimą, įėjimo pokyčio 1→0 panaikinimas sąlygoja papildomo išėjimo pokyčio 1→0 atsiradimą. Netiesioginę įtaką vadinsime nonrobust pagal analogiją su perdavimo gedimų tikrinimu.

Bendru atveju turime du įėjimo poveikius $P^1=<p_1^1,p_2^1,p_3^1,\ldots,p_i^1,\ldots,p_n^1>$ ir $P^2=<p_1^2,p_2^2,p_3^2,\ldots,p_i^2,\ldots,p_n^2>$ ir dvi reakcijas į tuos poveikius $R^1=<r_1^1,r_2^1,r_3^1,\ldots,r_j^1,\ldots,r_m^1>$ ir $R^2=<r_1^2,r_2^2,r_3^2,\ldots,r_j^2,\ldots,r_m^2>$. Įėjimų įtaką į išėjimus galima atvaizduoti matrica $||X||_{2n \times 4m}$. Matricos elementas $x_{2i-1,4j-3}=1$, jei $p_i^1=0$, $p_i^2=1$, $r_j^1=0$, $r_j^2=1$, ir prilyginus $p_i^2=0$ gauname $r_j^2=0$. Matricos elementas $x_{2i-1,4j-2}=1$, jei $p_i^1=0$, $p_i^2=1$, $r_j^1=1$, $r_j^2=0$, ir prilyginus $p_i^2=0$ gauname $r_j^2=1$. Matricos elementas $x_{2i-1,4j-1}=1$, jei $p_i^1=0$, $p_i^2=1$, $r_j^1=0$, $r_j^2=0$, ir prilyginus $p_i^2=0$ gauname $r_j^2=1$. Matricos elementas $x_{2i-1,4j}=1$, jei $p_i^1=0$, $p_i^2=1$, $r_j^1=1$, $r_j^2=1$, ir prilyginus $p_i^2=0$ gauname $r_j^2=0$. Matricos elementas $x_{2i,4j-3}=1$, jei $p_i^1=1$, $p_i^2=0$, $r_j^1=0$, $r_j^2=1$, ir prilyginus $p_i^2=1$ gauname $r_j^2=0$. Matricos elementas $x_{2i,4j-2}=1$, jei $p_i^1=1$, $p_i^2=0$, $r_j^1=1$, $r_j^2=0$, ir prilyginus $p_i^2=1$ gauname $r_j^2=1$. Matricos elementas $x_{2i,4j-1}=1$, jei $p_i^1=1$, $p_i^2=0$, $r_j^1=0$, $r_j^2=0$, ir prilyginus $p_i^2=1$ gauname $r_j^2=1$. Matricos elementas $x_{2i,4j}=1$, jei $p_i^1=1$, $p_i^2=0$, $r_j^1=1$, $r_j^2=1$, ir prilyginus $p_i^2=1$ gauname $r_j^2=0$. Matome, kad visais atvejais p_i^1 ir p_i^2 reikšmės yra priešingos ir keičiant p_i^2 reikšmę turi keistis ir r_j^2 reikšmė. Įėjimą i matricoje X atitinka dvi eilutės 2(i-1) ir 2i. Eilutė 2(i-1) atitinka įėjimo pokytį 0→1, o eilutė 2i atitinka įėjimo pokytį 1→0. Išėjimą j matricoje X atitinka keturi stulpeliai. Stulpelis 4(j-3) rodo tiesioginę įtaką išėjimo pokyčiui 0→1, stulpelis 4(j-2) rodo tiesioginę įtaką išėjimo pokyčiui 1→0, stulpelis 4(j-1) rodo netiesioginę įtaką išėjimo pokyčiui 0→1, stulpelis 4j rodo netiesioginę įtaką išėjimo pokyčiui 1→0.

Paimkime paprastą dviejų įėjimų AND ventilį ir poveikių porą $P^1=<0,0>$, $P^2=<1,1>$ su reakcijomis $R^1=<0>$, $R^2=<1>$. Abiejuose įėjimuose ir išėjime yra pokyčiai $p_1(0\rightarrow1)$, $p_2(0\rightarrow1)$, $r_1(0\rightarrow1)$. Įėjimų įtaką į išėjimą vaizduos matrica $||X||_{4x4}$. Panaikinus pokytį pirmam įėjime $p_1(0\rightarrow0)$ išnyksta pokytis ir išėjime $r_1(0\rightarrow0)$ ir atžymimas matricos elementas $x_{1,1}=1$. Analogiškai panaikinus pokytį antram įėjime $p_2(0\rightarrow0)$ išnyksta pokytis ir išėjime $r_1(0\rightarrow0)$ ir atžymimas matricos elementas $x_{3,1}=1$. Poveikių porai $P^1=<1,1>$, $P^2=<0,1>$ su reakcijomis $R^1=<1>$, $R^2=<0>$ ir esant pokyčiams $p_1(1\rightarrow0)$, $p_2(1\rightarrow1)$, $r_1(1\rightarrow0)$ atžymimas matricos elementas $x_{2,2}=1$, o poveikių porai $P^1=<1,1>$, $P^2=<1,0>$ su reakcijomis $R^1=<1>$, $R^2=<0>$ ir esant pokyčiams $p_1(1\rightarrow1)$, $p_2(1\rightarrow0)$, $r_1(1\rightarrow0)$ atžymimas matricos elementas $x_{4,2}=1$. Poveikių pora $P^1=<1,1>$, $P^2=<0,0>$ su reakcijomis $R^1=<1>$, $R^2=<0>$ esant pokyčiams $p_1(1\rightarrow0)$, $p_2(1\rightarrow0)$, $r_1(1\rightarrow0)$ neatžymi jokio

matricos elemento, nes panaikinus pokytį bet kuriam įėjime išėjime pokytis neišnyksta. Poveikių poros P^1=<0,1>, P^2=<1,0> su reakcijomis R^1=<0>, R^2=<0> ir P^1=<1,0>, P^2=<0,1> su reakcijomis R^1=<0>, R^2=<0>, esant pokyčiams $p_1(0\rightarrow1)$, $p_2(1\rightarrow0)$, $r_1(0\rightarrow0)$ ir $p_1(1\rightarrow0)$, $p_2(0\rightarrow1)$, $r_1(0\rightarrow0)$ nekeičia išėjimo reikšmės tačiau panaikinus pokytį $p_2(1\rightarrow0)$ arba $p_1(1\rightarrow0)$, t. y. padavus signalus (1→1) išėjime atsiranda pokytis $r_1(0\rightarrow1)$. Tai įėjimo įtaka į išėjimą netiesioginio tipo ir matricoje X atžymimi elementai $x_{4,3}=1$ ir $x_{2,3}=1$. Rezultatai visoms galimoms poveikių poroms parodyti Lentelėje 1. Kiekvienai poveikių porai apatinėse eilutėse parodyta kokie pokyčių panaikinimai įtakoja išėjimą ir kokie atžymimi matricos elementai. Matome, kad penkios poveikių poros nenustato įėjimų įtakos į išėjimus. Poveikių pora 3 nustato tą pačią įtaką tarp įėjimų ir išėjimų kaip ir poveikių poros 6 ir 9 kartu. Lentelėje 2 parodyta matrica X.

Aprašysime algoritmą, kuris duotiems įėjimo poveikiams P^1 ir P^2 suformuoja matricą $||X||_{2n\ x\ 4m}$. Pradžioje visi matricos elementai prilyginami nuliui. Nagrinėjami visi įėjimai ir kintamasis d prilyginamas invertuotai pirmo poveikio reikšmei (eilutė 3). Besiskiriantiems įėjimo signalo reikšmėms (eilutė 4) trečiam poveikyje P^3 įėjimo signalo reikšmė invertuojama, paskaičiuojama išėjimo reakcija R^3 ir išnagrinėjus atstatoma (eilutė 12). Toliau nagrinėjami visi išėjimai (eilutė 5). Jeigu keičiasi išėjimo signalo reikšmė (eilutė 6) ir to išėjimo poveikių P^1 ir P^2reakcijos taip skirtingos, matricoje X užrašomi atitinkami vienetukai. Kintamųjų d ir c naudojimas leidžia kompaktiškai indeksuoti matricos X elementus.

1. $X=||x_{i,j}:=0||_{2nx4m}$;
2. $R^1:=f(P^1)$; $R^2:=f(P^2)$;
3. DO i: =1,2,3,...,n; P^3: =P^2; d:=1- p^1_i;
4. IF ($p^1_i \neq p^2_i$) THEN p^3_i:=1- p^2_i; $R^3=f(P^3)$;
5. DO j: =1,2,3,...,m; c:=3- r^1_j;
6. IF ($r^3_j \neq r^2_j$) THEN
7. IF($r^2_j \neq r^1_j$) THEN $x_{2i-d,4j-c}$:=1;
8. ELSE $x_{2i-d,4j-c+2}$:=1;
9. ENDIF;
10. ENDIF;
11. ENDDO;
12. ENDIF;
13. ENDDO;
14.

Dvi poveikių poros nustatančios tą pačią įtaką tarp schemos įėjimų ir išėjimų nėra ekvivalentiškos, nes gali tikrinti skirtingus vėlinimo gedimus net ir tos pačios schemos realizacijos. Įėjimo įtaka į išėjimą pasireiškia įvairiais schemos keliais. Todėl kaip nekokybiškas galima atmęsti tik tokias poveikių poras, kurios nenustato jokios įtakos tarp įėjimų ir išėjimų. Vykdant įvairius eksperimentinius tyrimus buvo pastebėta, kad tokių poveikio porų didesnėm schemom yra nedaug. Todėl apibūdinant funkcinio testo kokybę tenka ieškoti kompromiso tarp funkcinio testo ilgio ir jo kokybės. Kompromisiniu atveju galima

apriboti kiek testo poveikių porų turi nustatyti įėjimo įtaką į išėjimą. Įvedus koeficientą Δ, galima lanksčiai surasti kompromisą tarp testo ilgio ir jo kokybės. Ribiniu atveju, kai Δ=1, pakanka tik vieną kartą nustatyti įėjimo įtaką į išėjimą ir testo kokybę tuo atveju charakterizuos funkcinio testo poveikio porų matricos X vienetinių elementų kiekis. Tai yra funkcinio testo kokybės apatinė riba, nes akivaizdu, kad jeigu funkcinis testas nenustato kai kurių įėjimų įtakos į išėjimus, tai toks testas tikrai netikrina dalies schemos kelių, konstantinių gedimų ir tuo pačiu vėlinimo gedimų. Reikia pastebėti, kad matricos X kai kurie elementai visuomet lygūs nuliui (įėjimas neturi elektrinio ryšio su išėjimu), kas apsunkina funkcinio testo kokybės laipsnio (pilnumo) vertinimą, tačiau netrukdo palyginti kokybę skirtingų funkcinių testų.

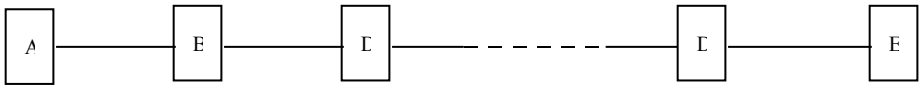
1. Lentelė .Poveikio porų analizė.

No.	Poveikių pora	P_1	P_2	R_1	X
1	P^1=<0,0>, P^2=<0,1>,R^1=<0>, R^2=<0>	0→0	0→1	0→0	
2	P^1=<0,0>, P^2=<1,0>,R^1=<0>, R^2=<0>	0→1	0→0	0→0	
3	P^1=<0,0>, P^2=<1,1>,R^1=<0>, R^2=<1>	0→1	0→1	0→1	
		0→0	0→1	0→0	$X_{1,1}$=1
		0→1	0→0	0→0	$X_{3,1}$=1
4	P^1=<0,1>, P^2=<0,0>,R^1=<0>, R^2=<0>	0→0	1→0	0→0	
5	P^1=<0,1>, P^2=<1,0>,R^1=<0>, R^2=<0>	0→1	1→0	0→0	
		0→1	1→1	0→1	$X_{4,3}$=1
6	P^1=<0,1>, P^2=<1,1>,R^1=<0>, R^2=<1>	0→1	1→1	0→1	
		0→0	1→1	0→0	$X_{1,1}$=1
7	P^1=<1,0>, P^2=<0,0>,R^1=<0>, R^2=<0>	1→0	0→0	0→0	
8	P^1=<1,0>, P^2=<0,1>,R^1=<0>, R^2=<0>	1→0	0→1	0→0	
		1→1	0→1	0→1	$X_{2,3}$=1
9	P^1=<1,0>, P^2=<1,1>,R^1=<0>, R^2=<1>	1→1	0→1	0→1	
		1→1	0→0	0→0	$X_{3,1}$=1
10	P^1=<1,1>, P^2=<0,0>,R^1=<1>, R^2=<0>	1→0	1→0	1→0	
11	P^1=<1,1>, P^2=<0,1>,R^1=<1>, R^2=<0>	1→0	1→1	1→0	
		1→1	1→1	1→1	$X_{2,2}$=1
12	P^1=<1,1>, P^2=<1,0>,R^1=<1>, R^2=<0>	1→1	1→0	1→0	
		1→1	1→1	1→1	$X_{4,2}$=1

2. Lentelė. Matrica X

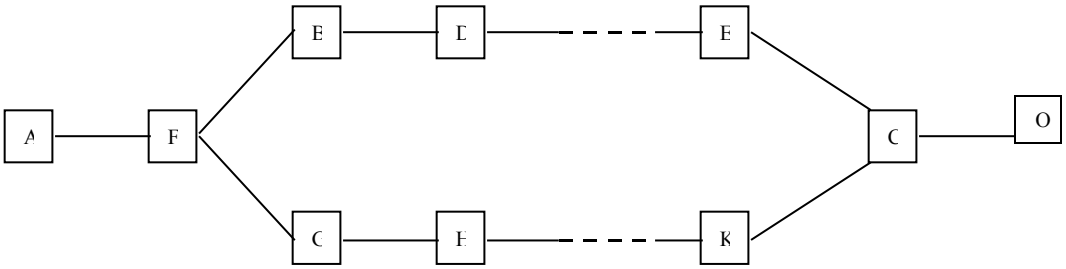
Įėjimai	Išėjimas			
	Robust		Nonrobust	
Pirmas įėjimas	1	0	0	0
	0	1	1	0
Antras įėjimas	1	0	0	0
	0	1	1	0

Tiesioginio tipo įėjimo įtaka į išėjimą rodo, kad poveikių pora aktyvina vienmatį arba daugiamatį kelią nuo įėjimo į išėjimą. Jeigu aktyvinamas vienmatis kelias tai poveikių pora patikrina to kelio visus perdavimo gedimus. Tarp įėjimo ir išėjimo gali būti daug vienmačių kelių. Kiek yra tokių vienmačių kelių apsprendžia konkreti schemos realizacija. Įėjimo poveikių pora tikrina tik kažkurio vieno aktyvaus vienmačio kelio perdavimo gedimus. Jeigu įėjimas įtakoja išėjimą tai bent vienas kelias turi visuomet egzistuoti. Tačiau negalime tvirtinti atvirkščiai. Jeigu schemoje egzistuoja kelias tarp įėjimo ir išėjimo tai dar nereiškia, kad įėjimas įtakoja išėjimą. . Taip pat negalime tvirtinti, jeigu įėjimas neturi tiesioginio tipo įtakos į išėjimą, tai schemoje neegzistuoja kelias tarp to įėjimo ir išėjimo. Tačiau galime tvirtinti, kad kai schemoje neegzistuoja kelias tarp įėjimo ir išėjimo, tai įėjimas negali turėti tiesioginio tipo įtaką į išėjimą. Grafiškai aktyvinamą vienmatį kelią vaizduosime kaip parodyta pav. 3, kur stačiakampiai rodo ventilius, o linijos ryšius tarp jų, o punktyrinė linija ventilių ir ryšių daugkartinį atkartojimą.



3. pav. Vienmatis aktyvinamasis kelias.

Daugiamatis aktyvinamas kelias pavaizduotas pav. 4.

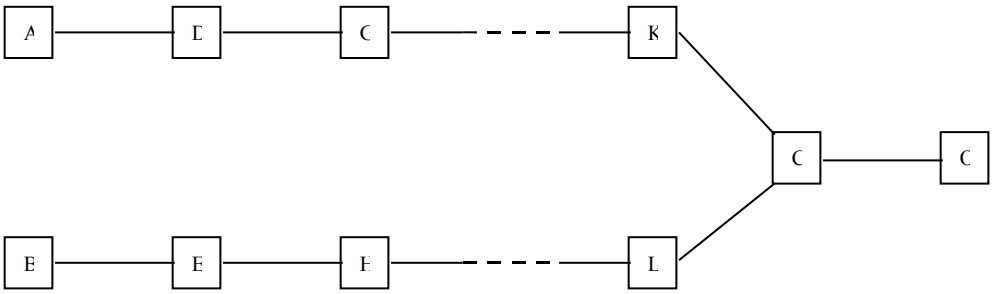


4. pav. Daugiamatis aktyvinamasis kelias.

Jeigu aktyvinamas daugiamatis kelias tai poveikių pora visuomet patikrina to kelio visus perdavimo gedimus iki išsišakojimo ventilio F, kadangi ši atkarpa atitinka vienmatį kelią. Kai aktyvinamas daugiamatis kelias, tai priklausomai nuo loginės funkcijos šakų suėjimo taške gali būti tikrinami visų šakų perdavimo gedimai, tik vienos šakos perdavimo gedimai arba perdavimo gedimai šakose gali visai nesitikrinti. Panagrinėkime paveiksle 4 pavaizduotą daugiamatį išsišakojantį į dvi šakas ir po to sueinantį kelią. Laikykime, kad pokytis įėjime iššaukia pokyčius abiejose šakose ir šie pokyčiai matomi abiejuose sueinančio ventilio šakose. Priklausomai nuo inversijų kiekio šakose suėjimo ventilyje galimi keturi pokyčių atvejai: $p_1(1\rightarrow0)$, $p_2(1\rightarrow0)$ arba $p_1(0\rightarrow1)$, $p_2(0\rightarrow1)$ arba $p_1(1\rightarrow0)$, $p_2(0\rightarrow1)$ arba $p_1(0\rightarrow1)$, $p_2(1\rightarrow0)$. Jeigu suėjimo ventilyje C vykdoma funkcija IR tai pirmuoju atveju šakose perdavimo gedimai nesitikrina, bet tikrinasi perdavimo gedimai nuo suėjimo ventilio iki išėjimo. Antruoju atveju perdavimo gedimai tikrinasi abiejose šakose ir nuo suėjimo ventilio iki išėjimo. Trečiu ir ketvirtu atveju tikrinasi perdavimo gedimai šakų, kuriose suėjimo taške vyksta pokytis (1→0). Bet trečiuoju ir ketvirtuoju atveju išnyksta pokytis po suėjimo ventilio ir perdavimo gedimai nuo suėjimo ventilio C iki išėjimo nesitikrina. Kadangi išėjime nėra pokyčio tai pokyčio panaikinimas įėjime negali panaikinti pokyčio išėjime ir jokia įėjimo įtaka į išėjimą nenustatoma, nors pokytis tikrina vienos šakos perdavimo gedimus. Simetriška situacija gaunama, kai suėjimo ventilyje vykdoma funkcija ARBA ir apsikeičia vietomis pirmas ir antras atvejai, o trečiuoju ir ketvirtuoju atveju tikrinasi šakos, kuriose suėjimo taške vyksta pokytis (0→1). Bendru atveju kiekviena šaka gali turėti kitus išsišakojimus ir suėjimus ir t. t. Todėl esant išsišakojantiems ir sueinantiems keliams, bendru atveju poveikių poros tikrinami perdavimo gedimai gali nesudaryti ištisinio kelio. Ištisinį kelią sudaro tik vienmačio kelio tikrinami perdavimo gedimai, kai kiekvienam tikrinamam ventilio perdavimo gedimui kelyje yra kitas tikrinamas ventilio ir elektrinės grandinės

perdavimo gedimas. Išsišakojančiam ir sueinančiam kelyje tikrinami perdavimo gedimai susideda iš atskirų segmentų, kurie kai kuriais atvejais gali sudaryti ir ištisinį kelią.

Netiesioginio tipo įėjimo įtaka į išėjimą rodo, kad išėjimo reikšmę įtakoja daugiau kaip vienas įėjimas ir nagrinėjamas įėjimo pokytis blokuoja kito įėjimo pokytį. Todėl panaikinus pokytį nagrinėjamam įėjime išėjime atsiranda pokytis. Tokiu būdu poveikių pora nustatanti netiesioginio tipo įėjimo įtaką į išėjimą neaktyvina schemoje ištisinio kelio nuo įėjimo į išėjimą. Be to nuo blokavimo taško iki išėjimo perdavimo gedimai nesitikrina. Ši situacija pavaizduota pav. 5.



5. pav. Neaktyvinamasis kelias.

Laikykime, kad į sueinantį ventilį C , kuris vykdo funkciją IR iš įėjimo A ateina pokytis (0→1), o iš įėjimo B ateina pokytis (1→0) ir tokiu atveju išėjime pokytis nestebimas. Panaikinus pokytį įėjime B išėjime O atsiras pokytis, nes įėjimo A pokytis praeis iki išėjimo O. Tai reiškia, kad įėjimo šakos einančios iš įėjimo B perdavimo gedimai tikrinasi išėjime kaip netiesioginiai. Todėl tokią įėjimo įtaką į išėjimą vadiname netiesioginio tipo. Šiuo atveju visada tikrinasi neištisinis kelias nuo panaikinto pokyčio įėjimo iki išėjimo, kuriam atsirado pokytis. Bendru atveju galime laikyti, kad tiesioginio tipo įtakos nustatymas nuo įėjimo iki išėjimo tikriausiai leidžia patikrinti daugiau gedimų, nes tikrinami perdavimo gedimai gali sudaryti ištisinį kelią ir dar papildomai gali tikrinti kitų šakų susietų su keliu perdavimo gedimus. Tuo tarpu netiesioginio tipo įėjimo įtakos nustatymas į išėjimą siejasi visuomet su perdavimo gedimais sudarančiais neištisinį kelią. Tikriausiai galima pateikti prieštaraujantį pavyzdį, kai netiesioginio tipo įtakos nustatymas nuo to įėjimo iki išėjimo patikrina daugiau perdavimo gedimų negu tiesioginio tipo įtakos nustatymas. Bendru atveju laikysimės prielaidos, kad tiesioginio tipo įtakos nustatymas testavimo požiūriu yra vertingesnis negu netiesioginio tipo įtakos nustatymas. Netiesioginio tipo įtakos nustatymas didina funkcinio vėlinimo testo ilgį, nes į testą įtraukiami papildomi testiniai rinkiniai susieti su netiesiogine įtaka. Todėl ir čia galima kompromiso paieška tarp funkcinio testo ilgio ir kokybės kriterijaus.

Bendriausiu atveju laikysime, kad vėlinimo funkcinio testo kokybę kai koeficientas $\Delta=1$ charakterizuoja matricos X, paskaičiuotos testo poveikio poroms, vienetukų suma K. Jeigu šis kokybės kriterijus netenkina, atskiroms schemų klasėms po išsamių eksperimentinių tyrimų funkcinio testo kokybės kriterijus gali būti patikslintas ieškant kompromiso tarp funkcinio testo ilgio ir schemos realizacijos vėlinimo gedimų tikrinimo pilnumo. Patikslinimo metu reikia atsižvelgti į tai kokio statistiškai ilgio funkcinis testas išpildo pasirinktą testo kokybės kriterijų. Eksperimentinis tyrimas turi konstatuoti kiek statistiškai vėlinimo gedimų tikrina testai nustatantys įėjimų tiesioginę įtaką į išėjimus, kiek statistiškai vėlinimo gedimų tikrina testai nustatantys įėjimų netiesioginę įtaką į išėjimus, ir kiek statistiškai vėlinimo gedimų tikrina testai nustatantys įėjimų tiesioginę ir netiesioginę įtaką į išėjimus. Analogiškai, turi būti ištirta koeficiento Δ svarba funkcinio testo ilgiui ir vėlinimo gedimų patikrinimui.

Funkcinio vėlinimo testo kokybės kriterijus remiasi faktu, kad įėjimas įtakoja išėjimą, bet ši įtaka gali būti skirtinga. Poveikių poros nustatančios tiesioginę ir netiesioginę įtaką tarp įėjimų ir išėjimų negali garantuoti visų schemos vėlinimo gedimų patikrinimo. Panagrinėsime kokios yra galimybės didinti schemų vėlinimo gedimų patikrinimo laipsnį. Įėjimo įtakos į išėjimą nustatymas nieko nesako apie schemos aktyvinamų kelių ilgius, kurie susieti su konkrečia schemos realizacija. Jeigu poveikių pora nustatanti įėjimo įtaka į išėjimą aktyvina ilgesnį kelią schemoje, galima tikėtis, kad tokia poveikių pora patikrins daugiau schemos vėlinimo gedimų. Bet kaip nustatyti funkcinio testo poveikių poros aktyvinamo kelio ilgį? Tam tikslui ieškosime poveikių poros faktorių proporcingų aktyvizuojamo kelio ilgiui.

Įėjimo pokyčio įtaka į išėjimą priklauso nuo kitų įėjimų signalo reikšmių ir pokyčių. Pakeitus kitų įėjimų signalų reikšmes arba panaikinus pokyčius gali nebepasireikšti nagrinėjamo įėjimo įtaką į nagrinėjamą išėjimą. Remsimės viena iš prielaidų, kuo daugiau įėjimų signalų reikšmių pakeitimų blokuoja nagrinėjamo įėjimo įtaką į nagrinėjamą išėjimą tuo ilgesnis yra schemoje aktyvinamas kelias. Sakykime poveikių poros pokytis įėjime p_i įtakoja signalo reikšmę išėjime r_j. Tuo tarpu panaikinus pokytį įėjime p_k įėjimo p_i įtaka signalo reikšmei išėjime r_j išnyksta. Įėjimą k vadinsime poveikių poros blokuojančiu įėjimu. Jeigu įėjimas k poveikių poroje turi vienodas signalų reikšmes $p_k^2 := p_k^1$, reikia išnagrinėti pokytį $p_k^2 := 1-p_k^1$. Jei suformavus pokytį įėjime k įėjimo p_i įtaka signalo reikšmei išėjime r_j išnyksta, tai įėjimą k taip pat vadinsime poveikių poros blokuojančiu įėjimu. Nagrinėjamo įėjimo įtaką į išėjimą blokuojančių įėjimų kiekis charakterizuoja aktyvinamo kelio ilgį ir gali būti naudojamas kaip įėjimo įtakos į išėjimą svorinis parametras. Aprašysime algoritmą, kaip duotai poveikių porai apskaičiuoti įėjimo įtakos į išėjimus svorinius parametrus.

1. Duotiems įėjimo poveikiams P^1 ir P^2 suformuojame matricą $||X||_{2nx4m}$. Analogiškos matricos $||X^{sv}||_{2n\ x\ 4m}$ visus elementus prilyginame nuliui.
2. DO i:=1 iki n
3. IF $p_i^1 \neq p_i^2$ THEN $p_i^2 := p_i^1$; ELSE $p_i^2 := 1-p_i^1$
4. Pakeistai poveikių porai paskaičiuojame matricą X'
5. DO k:=1,2,3,...,2n;
6. DO h:=1,2,3,...,4m;
7. IF $x_{k,h} > x'_{k,h}$ THEN $x^{sv}_{k,h} := x^{sv}_{k,h}+1$; ENDIF:
8. ENDDO;
9. ENDDO;
10. ENDDO;

Trečia algoritmo eilutė keičia i-tojo įėjimo signalo reikšmes. Jeigu signalo reikšmės abiejose įėjimo poveikiuose yra skirtingos, tai jos sulyginamos, o jeigu vienodos tai antro poveikio signalo reikšmė prilyginama priešingai pirmo poveikio signalo reikšmei. Pakeistai poveikių porai paskaičiuojame matricą X'. Jeigu matricoje X' kai kurie vienetukai išnyksta, tai rodo, kad įėjimo poveikio pakeitimas blokuoja įėjimo-išėjimo įtaką ir atitinkamas svorinės matricos elementas padidinamas vienetu. Išnagrinėjus visus pakeitimus gauname įėjimo-išėjimo įtakos blokavimo kartų kiekius, kurie sukaupiami matricoje $||X^{sv}||_{2n\ x\ 4m}$. Matricos X^{sv} elementas rodo poveikių poros nustatyto įėjimo įtakos į išėjimą blokavimo kartų kiekį, kuri pagal mūsų prielaidą proporcinga tikrinamų perdavimo gedimų kiekiui kelyje nuo įėjimo į išėjimą. Turint kiekvienos poveikių poros svorinių parametrų matricą $||X^{sv}||_{2n\ x\ 4m}$, galima iš visų poveikio porų išrinkti didžiausius svorinius parametrus kiekvieno įėjimo p_i įtakai signalo reikšmei išėjime r_j. Visų didžiausių svorinių parametrų suma S charakterizuos funkcinio vėlinimo testo kokybę. Galima tikėtis, kad šis naujas funkcinio vėlinimo testo kokybės kriterijus, nedaug padidins funkcinio testo ilgį, bet tikrins daugiau realizacijos vėlinimo gedimų. Tik eksperimentinis tyrimas gali atsakyti į klausimą kiek įėjimų įtakos į išėjimą svorinių parametrų naudojimas testo kokybės vertinimui gali padidinti tikrinamų realizacijos vėlinimo gedimų kiekį.

Poveikių poroje esminis yra antras poveikis, nes jis apsprendžia kelio aktyvinimo sąlygas ir sąlygoja pokyčio išnykimą arba atsiradimą. Pirmas poveikis apsprendžia kuriuose išėjimuose vyksta pokyčiai ir tuo pačiu ar įtaka į išėjimą yra tiesioginio ar netiesioginio tipo. Šiomis savybėmis pasinaudosime testų generavime.

2.8. FUNKCINIO VĖLINIMO TESTO GENERAVIMAS

Turint funkcinio vėlinimo testo kokybės kriterijus galima vykdyti funkcinio testo generavimą. Kai testo generavimo metu naudojamas imitacinis schemos modelis ir nežinoma konkreti schemos realizacija,

tenka naudotis poveikio porų atsitiktine paieška. Bet prieš tai panagrinėsime, kaip galima poveikių porą modifikuoti, kad modifikavimo pasėkoje daugiau įėjimų turėtų įtakos į išėjimus. Tam tikslui galima keisti kai kurias fiksuotas įėjimo signalų reikšmes formuojant pokytį (0→1) arba (1→0) arba panaikinti pokytį. Įvedus pakeitimą galimi trys atvejai. Vienu atveju pakeitimas neįtakoja išėjimų ir jau nustatytų įėjimų įtakos į išėjimus. Šiuo atveju įvesti pakeitimą netikslinga. Pakeitimą tikslinga įvesti tuo atveju, kai jis nustato naujas įtakas į išėjimą ir nekeičia jau nustatytų įėjimų įtakos į išėjimus. Jeigu įvedant pakeitimą prarandamos kai kurios jau nustatytos įtakos į išėjimus, tikslinga turimos poveikių poros nekeisti, bet formuoti naują poveikių porą su kitais pakeitimais. Tokiu būdu išnagrinėja visas poveikių poros įėjimo signalų reikšmes, gausime modifikuotą pirminę poveikių porą ir naujas poveikių poras nustatančias papildomas įėjimų įtakas į išėjimus.

Aprašysime algoritmą PPG, kuris modifikuoja duotą poveikių porą ir suformuoja kitas panašias poveikių poras.

1.Suformuojame vektorių V, kur v_i=0, i=1,2,...,n ;PAB:=0;
2.DO WHILE (PAB=0) PAB:=1;
3. Duotiems įėjimo poveikiams P^1 ir P^2 paskaičiuojame matricą X.
4. /* modifikuojam poveikių porą
5. DO i:=1 iki n
6. IF $p_i^1 \neq p_i^2$ THEN $p_i^2 := p_i^1$;
7. Pakeistai poveikių porai paskaičiuojame matricą X';
8. IF(X'>X)| (X'=X) THEN; X=X';ELSE $p_i^2 := 1-p_i^1$; ENDIF;
9. ELSE $p_i^2 := 1-p_i^1$
10. Pakeistai poveikių porai paskaičiuojame matricą X';
11. IF (X'>X) THEN X=X'; ELSE $p_i^2 := p_i^1$; ENDIF;
12. ENDIF;
13. ENDDO; Užsaugom modifikuotą poveikių porą P^1 ir P^2;
14. /* Formuojam panašią poveikių porą
15. DO i:=1 iki n
16. IF (PAB=1) THEN
17. IF v_i=0 THEN v_i:=1;
18. IF $p_i^1 \neq p_i^2$ THEN $p_i^2 := p_i^1$;
19. Pakeistai poveikių porai paskaičiuojame matricą X';
20. IF X'≠X THEN PAB:=0; ELSE $p_i^2 := 1-p_i^1$; ENDIF;
21. ELSE $p_i^2 := 1-p_i^1$
22. Pakeistai poveikių porai paskaičiuojame matricą X';
23. IF X'≠ X THEN PAB:=0; ELSE $p_i^2 := p_i^1$; ENDIF;
24. ENDIF;
25. ENDIF;
26. ENDIF;
27. ENDDO;
28.ENDWHILE;

Pažymėjimas X‘> X reiškia, kad visi matricos X‘ elementai didesni arba lygūs už matricos X elementus, pažymėjimas X‘=X reiškia, kad visi matricų elementai yra lygūs, pažymėjimas X‘<X reiškia, kad visi matricos X‘ elementai mažesni arba lygūs už matricos X elementus, pažymėjimas X‘≠X reiškia, kad matricos X‘ kai kurie elementai yra mažesni už matricos X elementus,o kai kurie elementai yra didesni už matricos X elementus.

Pirmiausia suformuojamas vektorius V, kurio elementai rodo išnagrinėtus poveikio įėjimus panašių poveikių porų formavimui ir pradžioje visi elementai prilyginami nuliui. Toliau duotai poveikių porai paskaičiuojama kokybę charakterizuojanti matrica X pagal antram skyriuje aprašytas taisykles. Algoritmas užbaigia darbą kai nei vienam įėjimui negalima suformuoti panašios poveikių poros, t. y. kai vektoriaus V visi elementai lygūs vienetui. Pirmiausia modifikuojamos įėjimo signalų reikšmės kurios nemažina matricos X vienetukų kiekio (punktai 4-13). Jeigu įėjimo poveikių poros reikšmės keičiasi, tai bandoma tą pokytį panaikinti ir jei šis panaikinimas padidina arba palieka tą patį matricos X vienetukų kiekį, tai šis pokyčio panaikinimas paliekamas ir toliau nagrinėjama taip modifikuota poveikių pora. Jeigu pokyčio panaikinimas matricoje X kai kuriuos vienetukus prideda, o kai kuriuos panaikina, tai pokytis nekeičiamas ir tuo bus pasinaudota formuojant panašius poveikių poras. Tuo tarpu naujo pokyčio suformavimas įėjime jau gali padidinti matricos X vienetukų kiekį ir toks pakeitimas paliekamas. Išnagrinėjus visus įėjimus gaunama modifikuota poveikių pora atžyminti matricoje X nemažiau elementų ir ji už saugojama.

Toliau keičiant antro poveikio signalų reikšmes bandoma suformuoti panašias poveikių poras (punktai 15-27). Jeigu įėjimo pokyčio panaikinimas matricoje X vienetukų ir sumažina ir padidina, tai pakeista poveikių pora traktuojama kaip pradinė ir pereinama prie jos modifikavimo. Jeigu matricoje X neatsiranda naujų vienetukų tai įėjime paliekama tas pats pokytis ir uždedamas požymis v_i:=1, kad įėjimas jau išnagrinėtas. Analogiškai nagrinėjami ir įėjimai, kurių poveikių poros reikšmės yra vienodos, tik čia bandoma įėjime suformuoti naują pokytį. Jei naujas pokytis ir padidina ir sumažina matricos X vienetukų skaičių tai gauta panašių poveikių pora perduodama nagrinėti modifikavimui. Jeigu matricos X vienetukų skaičius nesikeičia ar mažėja tai įėjime paliekama ta pati pastovi reikšmė ir uždedamas požymis v_i:=1. Algoritmas baigia darbą, kai visi įėjimai išnagrinėjami panašių poveikio porų formavimui.

Schemai turinčiai n įėjimų galima sugeneruoti 2^n skirtingų įėjimo poveikių ir $2^n(2^n-1)$ poveikių porų. Tačiau nemažai poveikių porų įtakos tarp įėjimų ir išėjimų nustatymo požiūriu yra ekvivalentiškos, tai yra nustato tas pačias įėjimų įtakas į išėjimus. Todėl galima apsiriboti atsitiktiniu antro poros poveikio generavimu, o pirmą poveikį iš pradžių prilyginti antram ir atlikti poros modifikavimą pagal aukščiau aprašytą algoritmą PPG.

Atsitiktinė paieška gali būti organizuojama dviem būdais. Vienu atveju atsitiktinai generuojamas antras poveikis ir pagal aprašytą algoritmą generuojamos poveikių poros. Kitu atveju atsitiktinai generuojami abu poros poveikiai. Eksperimentinis tyrimas gali atsakyti apie abiejų būdų privalumus ir trūkumus. Atsitiktinei paieškai labai svarbios paieškos pabaigos sąlygos. Atsitiktinė paieška gali būti organizuota porcijomis. Atsitiktinai generuojamas tam tikras kiekis poveikio porų, kurios po to modifikuojamos ir išplečiamos pagal algoritmą PPG, iš jų atrenkamos poros, kurios nustato dar nerastą įėjimų įtaką į išėjimus. Po to atsitiktinai generuojamas naujas kiekis poveikio porų. Kaip generavimo pabaigos sąlyga vertinama situacija kai naujai sugeneruotų poveikio porų kiekis (arba keli bandymai iš eilės) nebe atrenka poveikių poros, kuri nustato naują įėjimo įtaką į išėjimus.

Atsitiktinė paieška atrenka visas poveikių poras, kurios nustato bent vieną naują įėjimo įtaką į išėjimą. Aišku, kad principo „ pirmas tinkamas“ taikymas ne minimizuoja atrinktų poveikio porų kiekio. Atrinktų poveikio porų kiekio minimizavimui gali būti taikomas gradientinis principas, kai sudaromas tinkamų atrinkimui poveikio porų sąrašas ir iš jų atrenkama poveikių pora, kuri nustato daugiausia įėjimų įtakų į išėjimą. Likusios tinkamos poveikių poros įtraukiamos į tolimesnę atsitiktinę paiešką. Reikia pastebėti, kad atrinkus geriausią poveikių porą likusių porų tinkamumas keičiasi, nes įėjimo įtaką į išėjimą nustatoma tik viena poveikių pora.

Aprašysime poveikio porų generavimo algoritmą:

1.Atsitiktinai sugeneruojam K poveikio porų

2.Poveikio poras modifikuojam, išplečiam pagal algoritmą PPG ir paskaičiuojam jų kokybę. Kokybę charakterizuoja naujai nustatomų įėjimo įtakų į išėjimą kiekis arba jų svorių suma. Jei visų porų kokybė lygi nuliui, tai algoritmo pabaiga.

3.Pasirenkam kokybiškiausią poveikių porą, fiksuojame jos nustatomas įėjimo įtakas į išėjimą ir įtraukiam į galutinį funkcinį testą.

4.Perskaičiuojam nagrinėjamų poveikio porų kokybę įvertinant pasirinktą poveikių porą ir išmetam poveikių poras, kurios nenustato naujų įėjimų įtakų į išėjimus.

5.Papildom (jeigu reikia, t. y. išmetus poveikių poras, kurios nenustato naujų įėjimų įtakų į išėjimus, kai jų liko mažiau negu K) atsitiktinai generuodami poveikių poras iki kiekio K ir kartojam punktą 2.

Algoritmo konvergavimo greitis priklauso nuo sprendžiamo uždavinio ir dydžio K. Kuo didesnis K tuo iš daugiau poveikio porų pasirenkam kokybiškiausią, tuo trumpesnis gali būti galutinis funkcinis testas. Iš principo algoritmas gali pradėti iteracijas su viena poveikio pora, kuri pagal algoritmą PPG yra modifikuojama, pridedama panašių poveikio porų. Pridėtos poveikio poros kitos iteracijos metu pagal algoritmą PPG modifikuojamos ir vėl pridedama panašių poveikio porų. Iteracijos konverguos, kai pridėtos panašios poveikių poros bus nulinės kokybės tai yra nenustatys naujų įėjimo įtakų į išėjimus. Algoritmo penktam žingsnyje numatytas papildomas atsitiktinių poveikio porų generavimas leidžia išnagrinėti įvairiapusiškesnes poveikių poras ne tik panašias nagrinėtomis ir tuo pačiu gauti kokybiškesnį funkcinį testą. Funkcinio testo kokybė taip pat priklauso nuo dydžio K. Kuo daugiau nagrinėjamų poveikio porų nepagerina funkcinio testo kokybės tuo didesnė tikimybė, kad jau nustatytos visos galimos įėjimų įtakos į išėjimus. Tačiau didinant K žymiai išauga algoritmo vykdymo laikas ir tenka ieškoti kompromiso tarp skaičiavimų laiko ir funkcinio testo kokybės.

3. PROJEKTINĖ DALIS

3.1. PROJEKTO TIKSLAS

Projektavimas yra sunkus sistemos kūrimo etapas. Nuo projektavimo kokybės priklauso tai, kaip lengva bus sistemą realizuoti, kaip sudėtinga bus sistemą pakeisti pasikeitus kliento norams bei kiek sudėtinga sistemą bus prižiūrėti.

Įgyvendinant pagrindinį šio darbo tikslą, t. y. realizuojant algoritmus buvo sudaryta programos architektūra. Bus apžvelgtos galimos realizacijos alternatyvos, taip pat bus suprojektuotas sistemos vartotojo interfeisas ir parengtas testavimo modelis. Visa tai užtikrins kuriamos sistemos kokybę, garantuos valdymo patogumą ir paprastumą.

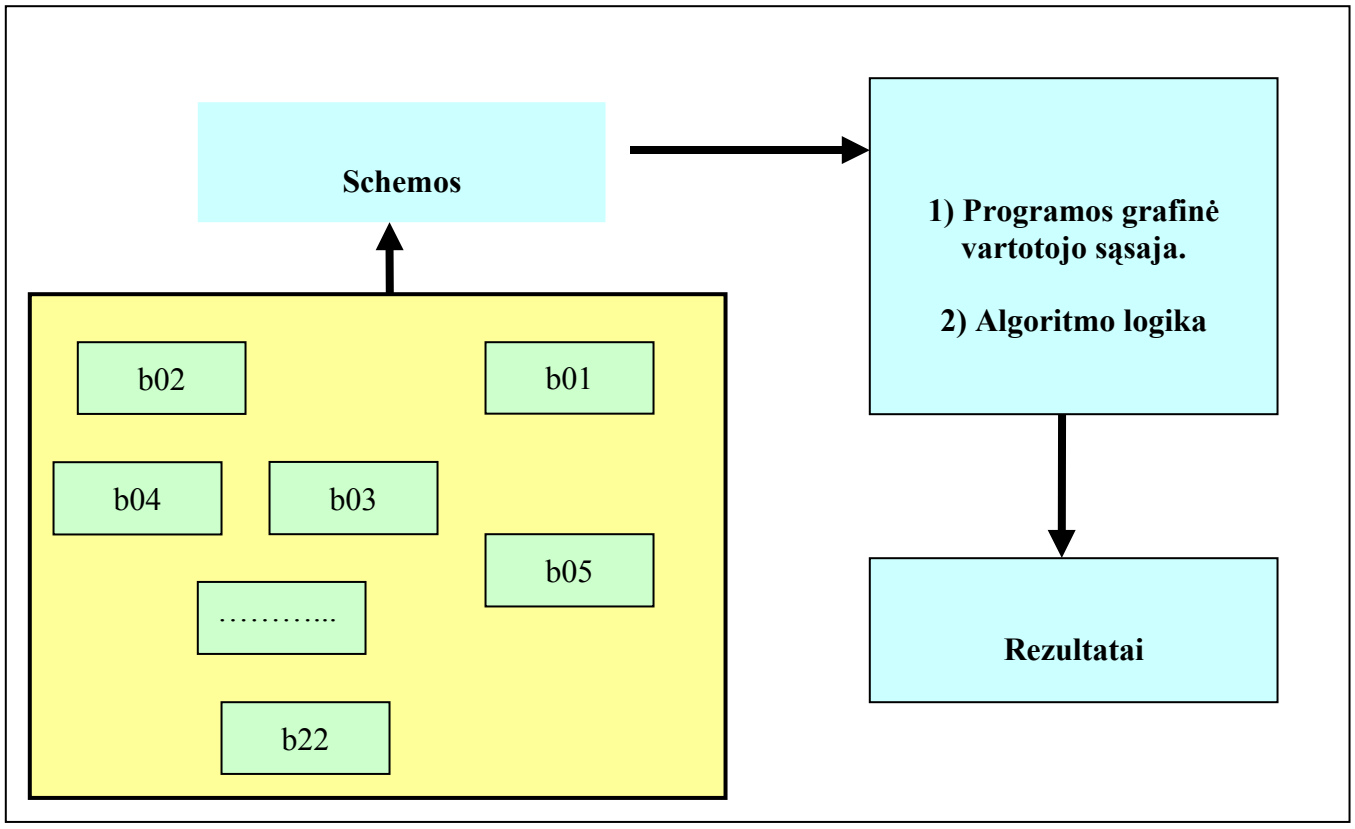
3.2. REIKALAVIMŲ MODELIS

Reikalavimų modelyje bus pasiūlyti bei suprojektuoti atskiri komponentai kuriamai programinei įrangai. Komponentai tai programos fragmentai, kurie išskaidyti į atskiras dalis. Analizės metu buvo nustatyta, kad sistema bus sudaryta iš trijų komponentų. Taigi, atitinkamai sistemos realizacija bus išskaidyta į:

1. Programinės įrangos grafinės sąsajos kūrimą.
2. Algoritmų realizavimas.
3. Schemų įterpimas į sistemą.

Tačiau pirmo ir antro komponento pilnai įgyvendinti nepavyko, todėl vartotojo grafinės sąsajos ir algoritmų logikos komponentai gavosi kaip vienas bendras.

Žemiau pateiktame paveiksle yra pavaizduotas supaprastintas šių komponentų apjungimas į vientisą pilnai funkcionuojančią sistemą (6 pav.). Diagrama sudaryta iš dviejų sluoksnių: vartotojo sąsajos ir algoritmų bei funkcinių schemų.



6. pav. Sistemos struktūra

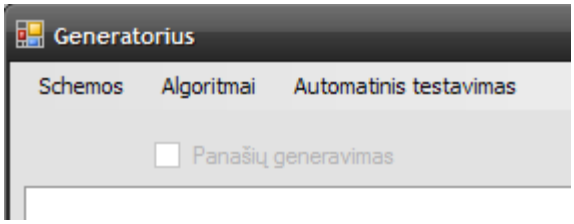
Remiantis projektuojamos sistemos struktūra, toliau bus projektuojami komponentai.

3.3. PROGRAMOS GRAFINĖS SĄSAJOS KŪRIMAS

Kuriama sistema turės vieną pagrindinį langą: kitaip sakant tai yra pagrindinis langas su visais galimais pasirinkimai (opcijos), kuriuos sudaro trys pagrindiniai valdymo meniu bei tam tikri informacijos laukai.

Pagrindiniai meniu:

1. Schemos – schemų pasirinkimas.
2. Algoritmai – algoritmų pasirinkimas.
3. Automatinis testavimas – testavimo modelio pasirinkimas.

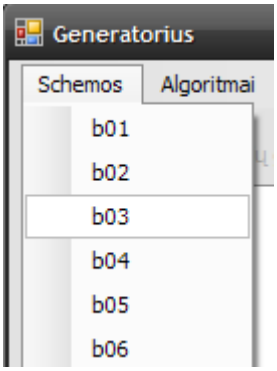


7. pav. Grafinės sąsajos pagrindiniai meniu.

Šiuos pasirinkimo laukus sudaro dar papildomi meniu punktai (sub-meniu) .

1. Schemos
    - **a.** b01
    - b. b02
    - c. b03

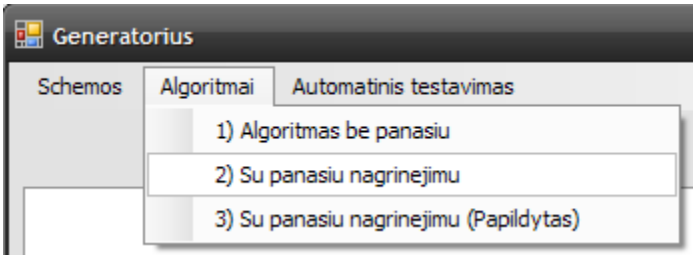
      …..
    - d. b22



8. pav. Grafinės sąsajos pagrindinis „Schemos“ meniu.

2. Algoritmai

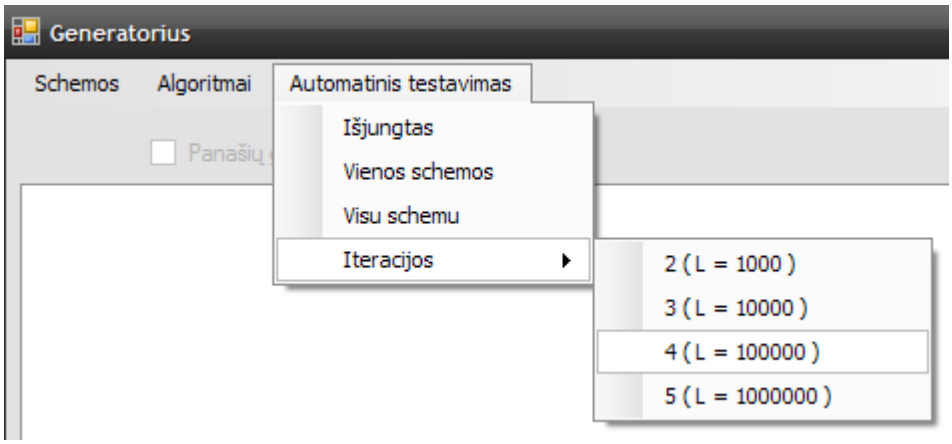
- a. Algoritmas be panašių
- b. Su panašiu nagrinėjimu
- c. Su panašiu nagrinėjimu(Papildytas)



9. pav. Grafinės sąsajos pagrindinis „Algoritmai“ meniu.

3. Automatinis testavimas
    - **a.** Išjungtas – išjungti automatinį testavimą.
    - **b.** Vienos schemos – automatinis testavimas tik vienai pasirinktai schemai.
    - **c.** Visų schemų – automatinis testavimas visom schemom nuo pasirinktos pradinės.
    - **d.** Iteracijos – L dydžio pasirinkimas automatiniam testavimui.

1. 2 (L = 1000)
2. 3 (L = 10000)
3. 4 (L = 100000)
4. 5 (L = 1000000)

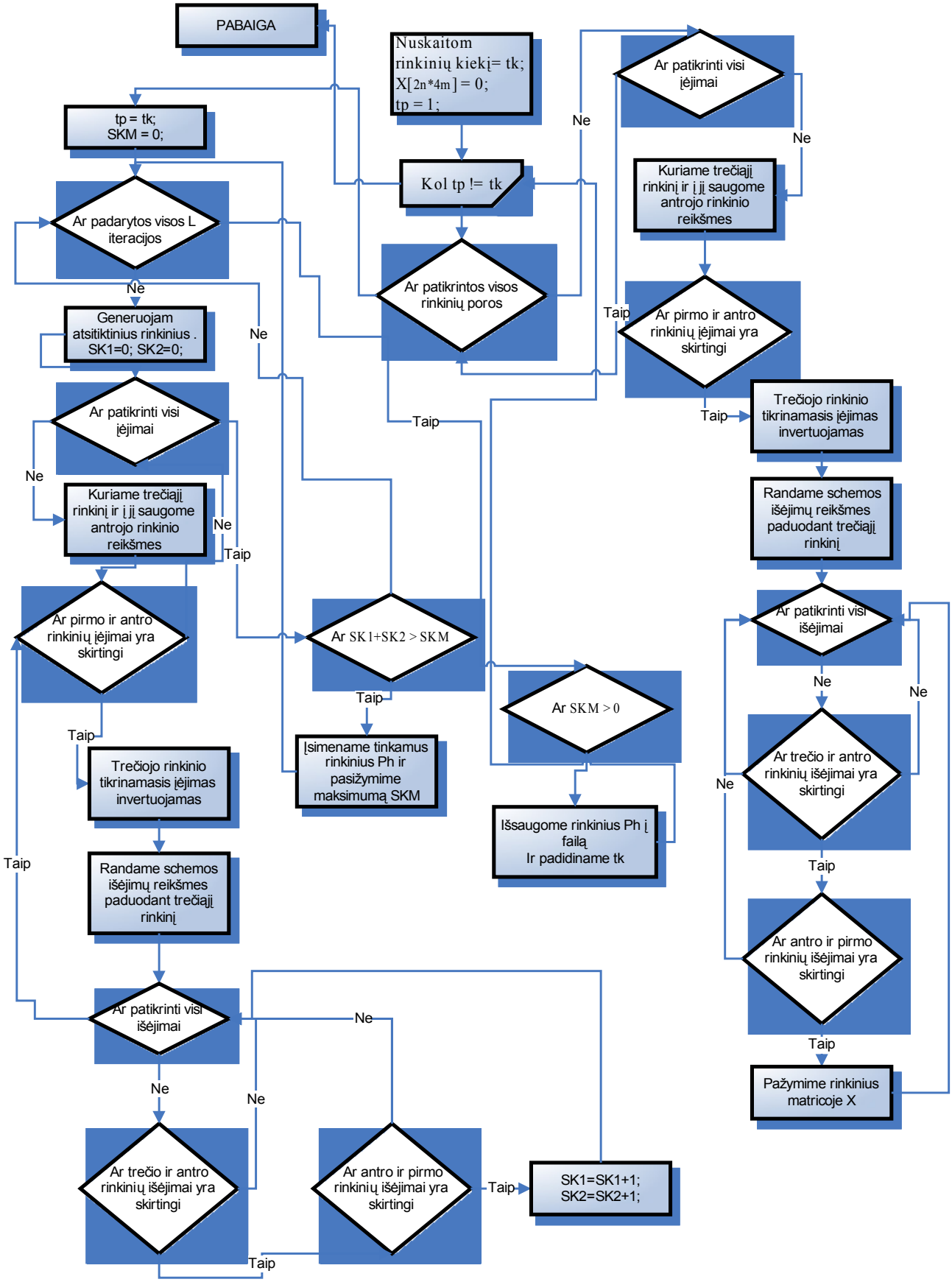


10. pav. Grafinės sąsajos pagrindinis „Automatinis testavimas“ meniu.

Kaip matosi iš paveikslėlių programos valdymas yra nesudėtingas ir paprastas.

3.4. ALGORITMŲ REALIZAVIMAS

Pagrindinis ir kitu dviejų algoritmų pagrindas yra pateiktas 11 pav. Algoritmas yra pateiktas pseudo kodo išraiška, todėl pirmasis algoritmo plėtojimo uždavinys yra išsiaiškinti tam tikrus žymėjimus ir simbolius. Kai kurie žymėjimai jau buvo aprašyti pirmoje analitinėje dalyje, bet šiuo atveju galima paminėti esmines vietas. Kaip matyti 11 pav. pagrinde figūruoja P1, P2, p1, p2, R1, R2, X, x. Didžioji raidė reiškia nulių ir vienetų masyvą, o mažoji raidė reiškia masyvo vieną kurį nors elementą. R1:=f(P1); reiškia, kad vienas rinkinys P1 yra perduodamas į nagrinėjamą schemą ir gaunamas rezultatas R1.

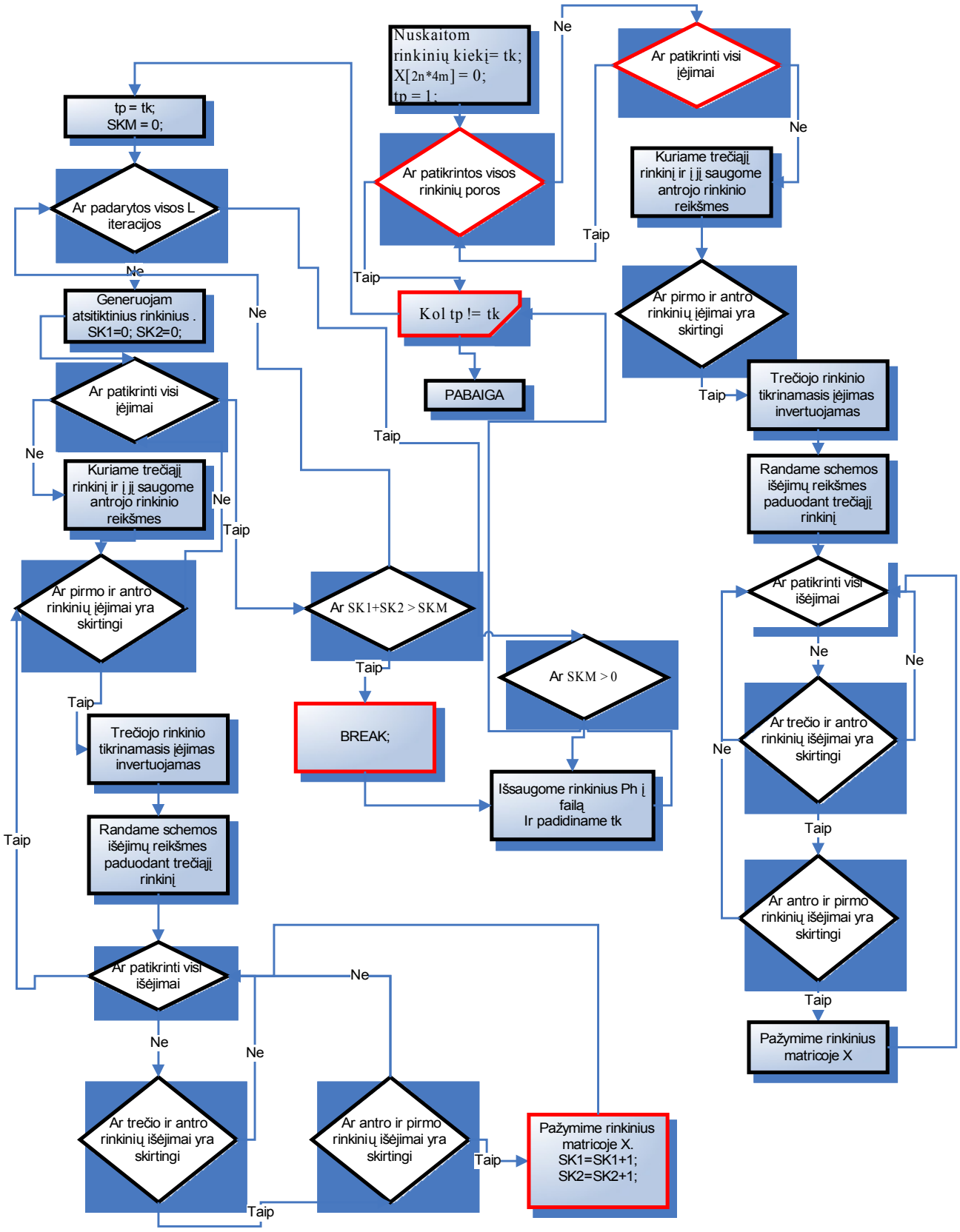

PABAIGA
Nuskaitom rinkinių kiekį= tk;
X[2n*4m] = 0;
tp = 1;
Kol tp != tk
Ar patikrintos visos rinkinių poros
Ne
Taip
Ar patikrinti visi įėjimai
Ne
Kuriame trečiąjį rinkinį ir į jį saugome antrojo rinkinio reikšmes
Ar pirmo ir antro rinkinių įėjimai yra skirtingi
Taip
Trečiojo rinkinio tikrinamasis įėjimas invertuojamas
Randame schemos išėjimų reikšmes paduodant trečiąjį rinkinį
Ar patikrinti visi išėjimai
Ne
Ar trečio ir antro rinkinių išėjimai yra skirtingi
Taip
Ar antro ir pirmo rinkinių išėjimai yra skirtingi
Taip
Pažymime rinkinius matricoje X
tp = tk;
SKM = 0;
Ar padarytos visos L iteracijos
Ne
Generuojam atsitiktinius rinkinius . SK1=0; SK2=0;
Ar patikrinti visi įėjimai
Ne
Kuriame trečiąjį rinkinį ir į jį saugome antrojo rinkinio reikšmes
Ne
Taip
Ar pirmo ir antro rinkinių įėjimai yra skirtingi
Taip
Trečiojo rinkinio tikrinamasis įėjimas invertuojamas
Randame schemos išėjimų reikšmes paduodant trečiąjį rinkinį
Ar patikrinti visi išėjimai
Ne
Ar trečio ir antro rinkinių išėjimai yra skirtingi
Taip
Ar antro ir pirmo rinkinių išėjimai yra skirtingi
Taip
SK1=SK1+1;
SK2=SK2+1;
Ar SK1+SK2 > SKM
Taip
Įsimename tinkamus rinkinius Ph ir pasižymime maksimumą SKM
Ar SKM > 0
Išsaugome rinkinius Ph į failą
Ir padidiname tk

11. pav. Pagrindinis algoritmas

3.5. ALGORITMO TOBULINIMAS

Testuojant schemas buvo pastebėti kai kurie algoritmo dėsningumai. Tarkim jei schema turi nuo 7 įėjimų ir 7 išėjimų iki ~ 200 įėjimų ir išėjimu, tuomet pirmoji algoritmo dalis t.y jau tinkamų rinkinių atžymėjimas matricoje X, užima nelabai pastebimą laiko tarpą. Bet jei schema yra didesnė nei prieš tai išvardytos sąlygos, tuomet algoritmo pirmoji dalis reikalauja labai daug laiko. Todėl šiuo atveju aš padariau kai kuriuos algoritmo pakeitimus kaip pagreitinti schemos testavimo procesą 12 pav.

Kaip matosi 12 paveiksle pirmoji algoritmo dalis yra iškelta iš while ciklo, tokiu būdu nevyksta papildomas for ciklas per jau atrinktus rinkinius ir vėl iš naujo juos atžymint. Nes kitaip, jei tarkim rinkinių yra apie 20000, o įėjimų ir išėjimų skaičius yra ~ 1000, tuomet atžymėti visus rinkinius matricoje X prireiktu apie 20 min. O kai atžymėjimas matricoje X yra padaromas tuoj pat kai nustatoma jog rinkinių pora yra tinkama pagal tam tikras sąlygas laikas yra sutaupomas naujų rinkinių generavimui. Bet noriu pabrėžti jog šis pakeitimas yra pastebimas tik testuojant labai dideles schemas t.y. didelis įėjimų ir išėjimų skaičius nuo ~ 700.



12. pav. Patobulintas algoritmas

3.6. SCHEMŲ ĮTERPIMAS Į SISTEMĄ

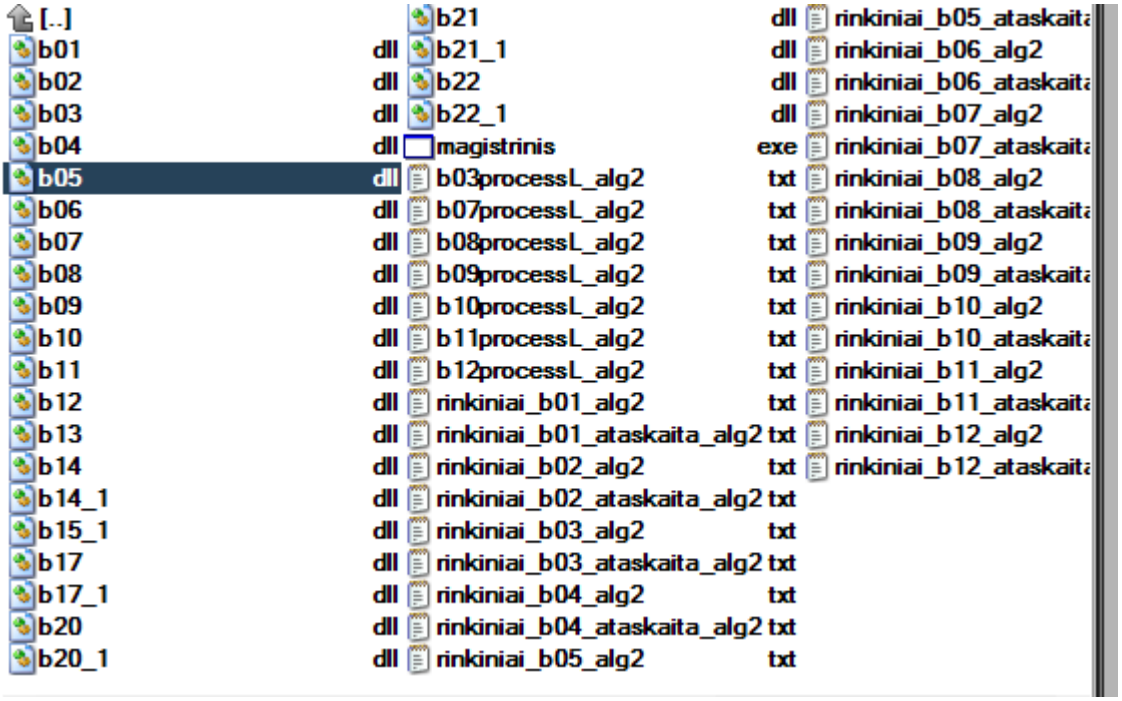
Plėtojant programą darbo pradžioje visos schemos buvo įterptos į vieną bendrą sistemą, kitaip sakant integruotos. Tačiau buvo susidurta su tam tikromis problemomis.

1. Kaip įterpti visas funkcijas su vienodais funkcijų pavadinimais? (void funkcija(char *gg,char *og,int in,int on,int *gerai))
2. Ir antra, kaip greitai ir nesunkiai sukompiliuoti visą projektą ? (Šita problema iškilo išsprendus pirmąją.)

Taigi, pirmoji problema buvo išspręsta panaudojus bazinį/raktinį žodį „*namespace*“ . Bet bandant sukompiliuoti bendrą, pilną programą, iškilo programos kompiliavimo problema. Laikas išaugo iki keturių valandų, ir projektas gavosi neįgyvendinamas. Todėl ši problema buvo išspręsta: visos funkcijos sukurtos kaip *.dll failai (Dynamically Link Library). Taip buvo išvengtas ir funkcijų vardų pasikartojimas ir kartu projekto kompiliavimo problema, bei kas yra svarbiausia – funkcijos atskirtos nuo grafinės vartotojo sąsajos.

Visos schemos yra sukurtos kaip atskiri projektai. Kad sukurti DLL failus reikia vieną kartą sukompiliuoti visas schemas ir viskas.

DLL failas turi būti būtinai tame pačiame kataloge kaip ir programos paleidžiamasis failas (magistrinis.exe).



13. pav. Programos failų medis

Schemų atnaujinimas

Toliau kyla klausimas kokiu būdu galima atnaujinti, pakeisti, pataisyti, išmesti jau esamas funkcijas. Tai galima padaryti atsidarius programos kodo failą pagal norimą schemos pavadinimą. Jos yra patalpintos */dll_schemos/b**/b**.cpp*.

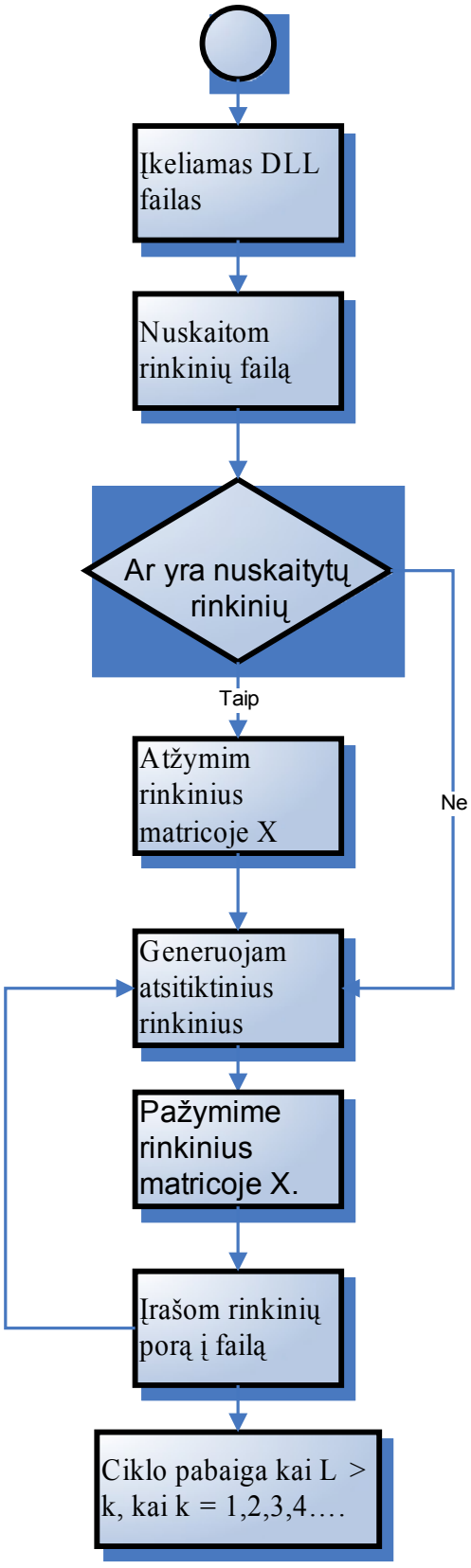
Visą funkcijos kodą reikia įterpti į funkciją *void funkcija(char *gg, char *og, int in, int on, int *gerai)* ir pakeisti funkcijos įėjimų ir išėjimų parametrus funkcijose *int getN()* ir *int getM()*. Ir po šių veiksmų užtenka atsidarytą projektą iš naujo sukompiliuoti.

3.7. SISTEMOS PROJEKTAS

Praeitoje dalyje buvo projektuojami atskiri komponentai. Šio poskyrio tikslas yra sujungti komponentus į vientisai veikiančią sistemą, kad geriau būtų galima perprasti kuriamos sistemos struktūrą, jos veikimą bei suvokti komponentų tarpusavio priklausomybę. Tuo tikslu pamėginsiu atvaizduoti grafiškai kaip veikia programa.

14 paveiksle pavaizduota veiklos diagrama, tiksliai atvaizduoja programos veikimą. Kaip matosi pirmas ir pagrindinis punktas yra schemos(funkcijos) užkrovimas. Programa neveiks jei nebus schemų *.dll failų. Toliau yra ieškoma anksčiau sugeneruotų rinkinių. Jei rinkiniai yra rasti, jie yra atžymimi matricoje X, o jei tokių nerasta, tuomet iškarto pereinama prie atsitiktinių rinkinių generavimo ir jų tinkamumo tikrinimo. Tinkama rinkinių pora yra įrašoma į rezultatų failą ir ciklas prasideda iš naujo.

Surastos tinkamos poros įrašymas į failą yra naudingas tuo, jog tarkim jei yra vykdomas ilgas generavimas ir programa dėl kažkokių priežasčių išsijungia, paleidus programą iš naujo, jau sugeneruoti rinkiniai yra nuskaitomi ir jų iš naujo nebegeneruoja, taip yra sutaupomas laikas.



14. pav. Veiklos diagrama

4. TYRIMO DALIS

4.1. TOLIMESNĖS SISTEMOS TOBULINIMO GALIMYBĖS

Tolimesnis programos plėtojimo galimybės yra kelios:

Pirma, sistemos trūkumas yra tas, kad vartotojo sąsaja yra vientisa su generavimo algoritmais. Juos būtų galima padaryti kaip atskirus sistemos komponentus. Tokiu atveju vartotojo sąsaja taptų kaip sistemos pagrindinis karkasas, kuris valdytų kitus sistemos komponentus.

Antra, reiktų padaryti dinaminį schemų pasirinkimo meniu. Tuomet norint įterpti norimą funkciją nereikėtų laikytis dabartinės sistemos reikalavimų, t.y. būtini schemų pavadinimai, kurie šiuo metu negali keistis. Toks sprendimas būtų patogesnis ir lengvesnis sistemos plėtojime

Trečia, reikia išsiaiškinti kodėl papildyti algoritmų variantai veikia neteisingai. (Šiuo metu tai nustatyti yra sunku, todėl kad, šių algoritmų pseudo kodas yra tiksliai perrašytas į C++ kodą, bet veikimas nėra teisingas)

5. EKSPERIMENTINĖ DALIS

5.1. SUKURTOS SISTEMOS KOKYBĖS TYRIMAS

Eksperimentams buvo naudojamos darbo vadovo duotos schemos:

Schema	Įėjimai	Išėjimai	Trigeriai	Įėjimai++	Išėjimai++	Ventiliai
B01	2	2	5	7	7	39 + 10 = 49
B02	1	1	4	5	5	24 + 4 = 28
B03	4	4	30	34	34	144 + 16 = 160
B04	11	8	66	77	74	632 + 105 = 737
B05	1	36	34	35	70	821 + 177 = 998
B06	2	6	9	11	15	49 + 7 = 56
B07	1	8	49	50	57	380 + 61 = 441
B08	9	4	21	30	25	157 + 26 = 183
B09	1	1	28	29	29	146 + 24 = 170
B10	11	6	17	28	23	174 + 32 = 206
B11	7	6	31	38	37	622 + 148 = 770
B12	5	6	121	126	127	963 + 113 = 1076
B13	10	10	53	63	63	310 + 52 = 462
B14	32	54	245	277	299	8567 + 1531 = 10098
B14_1	32	54	245	277	299	5812 + 1088 = 6900
B15_1	36	70	449	485	519	12547
B17	37	97	1415	1452	1512	30783

B17_1	37	97	1415	1452	1512	32971 + 6694 = 39666
B20	32	22	490	522	512	19724
B20_1	32	22	490	522	512	13930
B21	32	22	490	522	512	20078
B21_1	32	22	490	522	512	13934
B22	32	22	735	767	757	29219
B22_1	32	22	735	767	757	21031

3. Lentelė. Testavimo schemos.

Tikslas buvo sugeneruoti visom šiom schemom (lentelė 3) funkcinius testus ir išsaugoti rezultatus. Taigi sistemos testavimo etapai buvo keli, pirmasis etapas vyko programos plėtojimo etape, o antrasis ir pagrindinis etapas vyko kuomet sistema buvo užbaigta. Testavimo etapai susidėjo iš gautų rezultatų ataskaitos analizės.

Pirmasis etapas:

Schemos pasirinkimas. Šiuo atveju kaip pavyzdį paimsiu lengviausią b01 schemą, kad būtų aiškiau.

Algoritmo pasirinkimas. Pasirinktas pirmas algoritmas.

Dydžio L nustatymas. Nustatom L = 10.

Ir spaudžiam mygtuką „Run“.

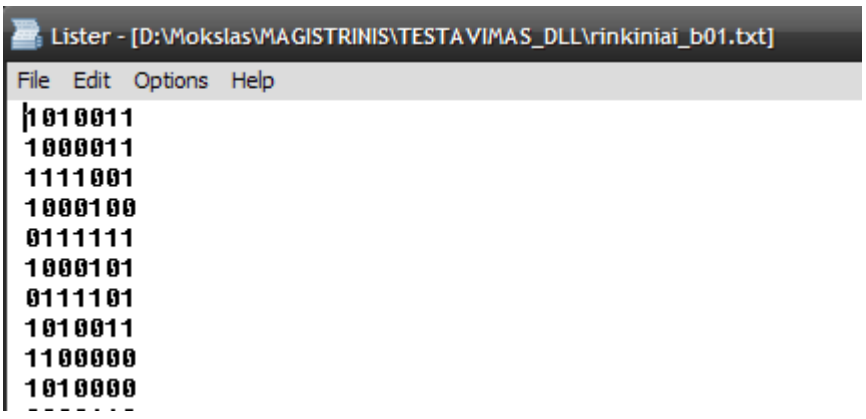
Dydžio L nustatymas. Nustatom L = 100.

Ir spaudžiam mygtuką „Run“.

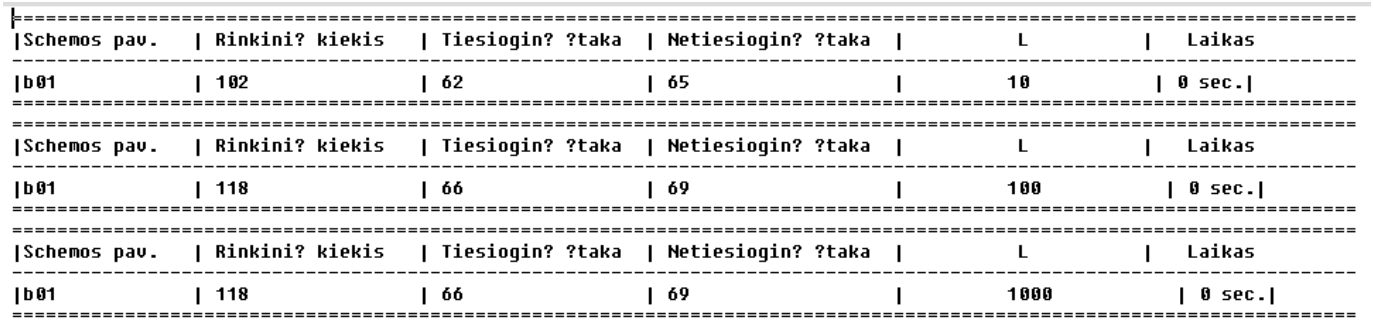
Dydžio L nustatymas. Nustatom L = 1000.

Ir spaudžiam mygtuką „Run“.

Po šių veiksmų yra sugeneruojami rinkinių ir ataskaitos failai - rinkiniai_b01.txt ir rinkiniai_b01_ataskaita.txt.



15. pav. rinkiniai_b01.txt failas.



Schemos pav.	Rinkini? kiekis	Tiesiogin? ?taka	Netiesiogin? ?taka	L	Laikas
b01	102	62	65	10	0 sec.

Schemos pav.	Rinkini? kiekis	Tiesiogin? ?taka	Netiesiogin? ?taka	L	Laikas
b01	118	66	69	100	0 sec.

Schemos pav.	Rinkini? kiekis	Tiesiogin? ?taka	Netiesiogin? ?taka	L	Laikas
b01	118	66	69	1000	0 sec.

16. pav. rinkiniai_b01_ataskaita.txt failas.

Iš šių rezultatų matosi jog padidėjus rinkinių skaičiui nuo 102 iki 118 (viso aštuonios poros), įtakojimas atitinkamai taip pat padidėjo. Tiesioginis įtakojimas padidėjo keturiais vienetais, netiesioginis įtakojimas padidėjo taip pat keturiais vienetais. Viso gaunasi, jog matricoje buvo atžymėtos aštuonios naujos rinkinių poros. Padidinus dydį L iki 1000 matosi jog jokių naujų rinkinių nebuvo papildomai sugeneruota. Iš to galima daryti išvadą jog generatorius veikia teisingai. O galutinis rezultatas yra vidurinė eilutė iš 16 paveikslėlio (su dydžiu L = 100).

Antrasis etapas:

Pradinės schemos pasirinkimas. Pasirenkam b01 (tačiau jei prieš tai buvo vykdomas automatinis testavimas, ir dėl kažkokių techninių priežasčių generavimas nutrūko, tarkim ant b12 schemos, tuomet pradinę schemą pažymime b12 schemą ir atliekam tolesnius veiksmus).

Algoritmo pasirinkimas. Pasirinktas pirmas algoritmas.

Dydžio L nustatymas. Nustatom L = 10.

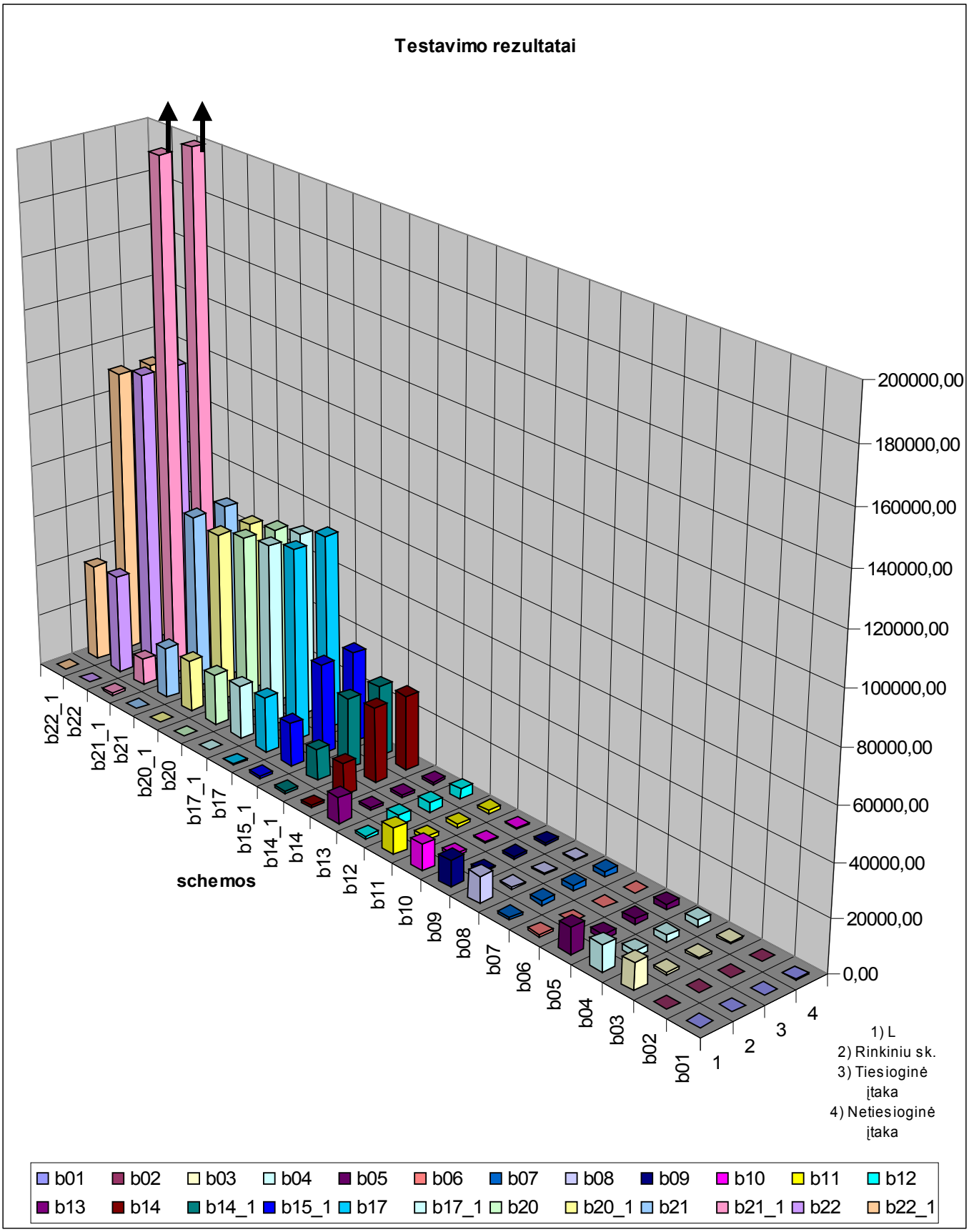
Pasirenkam *Automatinis testavimas → visu schemu*.

Pasirenkam *Automatinis testavimas → iteracijos → 3(L = 10000)*.

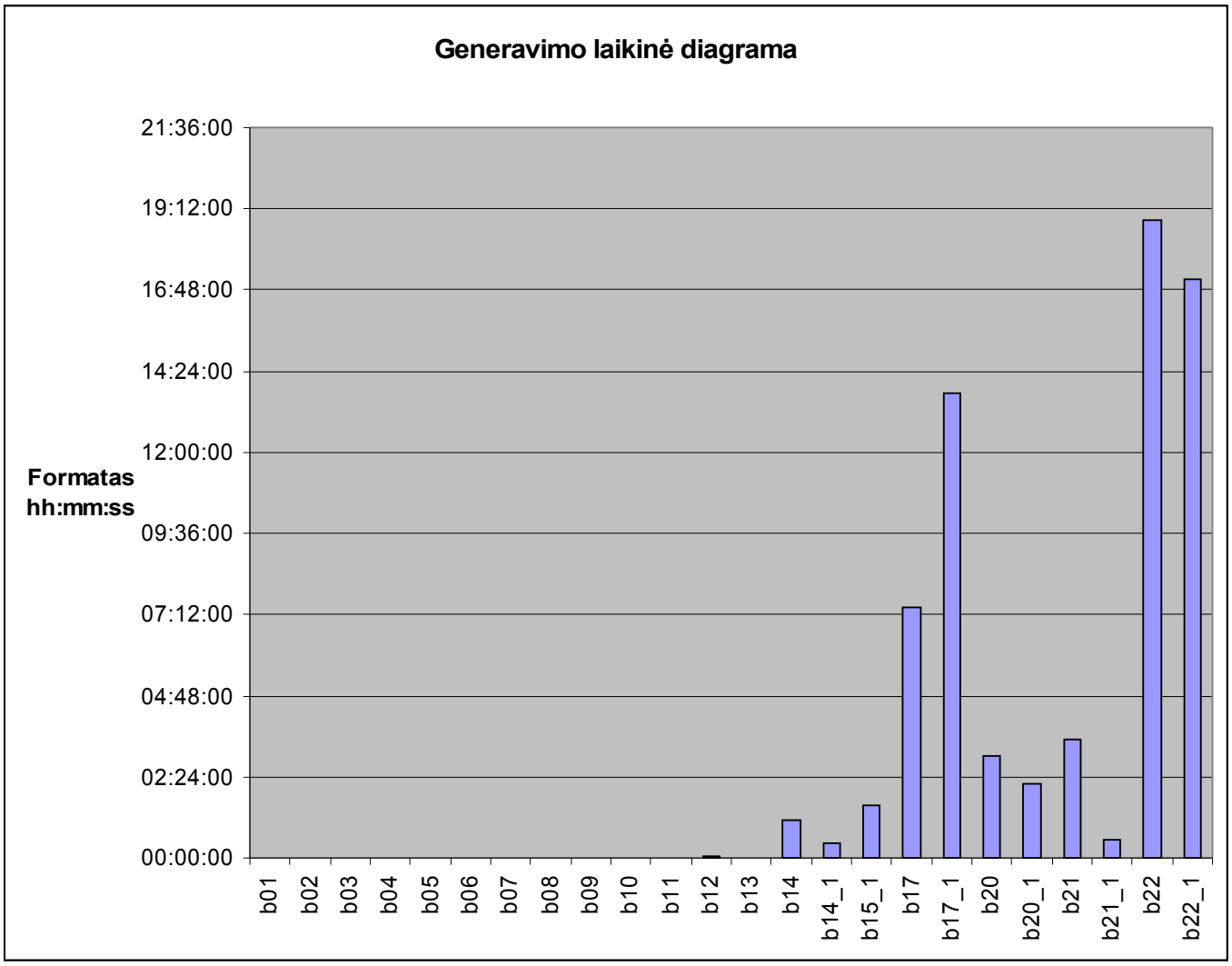
Ir spaudžiam mygtuką „Run“.

Po visų šių veiksmų dabar telieka laukti kol bus praeita per visas funkcijas ir sugeneruoti funkciniai testai. Pasibaigus automatiniam testavimui reikia peržiūrėti visus ataskaitos failus ir įsitikinti ar visos schemos yra pilnai ištestuotos. Jei schema nepilnai ištestuota galima išjungti automatinį testavimą ir rankiniu būdu ją pilnai užbaigti. (Tai galima padaryti ir vėliau, turiu omeny jog programą galima išjungti. Iš naujo įjungus programą ir paleidus generatorių bus nusiskaitomi anksčiau sugeneruoti rinkiniai ir toliau pratęsiamas generavimo ciklas.)

Taigi, atlikus antrąjį etapą buvo gauti galutiniai rezultatai, kurie yra pateikti prieduose. Žemiau pateiktas paveikslėlis (17 pav.) atvaizduoja apibendrintus visus pirmojo algoritmo rezultatus.



17. pav. Apibendrinti rezultatai



18. pav. Generavimo laikinė diagrama

5.1.1. Papildytų algoritmo testavimai

Kaip jau minėjau trečioje dalyje (Tyrimo dalis) papildyti algoritmų variantai neveikia. Algoritmų C++ kodas pilnai atitinka duotą algoritmų pseudo kodą, tačiau dėl kažkokių priežasčių jie neveikia. Pradėjus generavimo procesą su abejais algoritmais, generavimo procesas nesibaigia arba generavimo laikas, kad ir mažom schemom yra be galo ilgas.

IŠVADOS

Didėjant integracijai bei schemos darbiniams dažniams vis didesnę įtaką įgauna vėlinimo gedimai. Vėlinimo gedimai tikrinami testinių rinkinių pora, kur pirmasis testinis rinkinis suformuoja pradines įėjimų reikšmes, o antras testinis rinkinys formuoja pokyčius schemos įėjimuose, o reakcija išėjimuose po antro testinio rinkinio matuojama po apibrėžtos laiko trukmės. Jeigu dėl vėlinimo gedimų signalai išėjimuose nespėja pasikeisti tai matavimo metu užfiksuojamos pirmu testiniu rinkiniu nustatytos signalų reikšmės, kas indikuoja apie schemoje egzistuojančius vėlinimo gedimus.

Tokie testavimai yra labai reikšmingi tiek sistemų saugumo užtikrinime tiek sistemos patikimumo atžvilgiu. Šiais laikais tokie reikalavimai yra keliami vis didesni, nes žmogaus saugumas yra visų svarbiausia.

Norint testuoti perdavimo vėlinimo gedimus, būtina sugeneruoti testinius rinkinius. Tai taip pat reikalauja pakankamai didelių laiko sąnaudų, todėl svarbu testinių rinkinių generavimą atlikti kuo efektyviau. Darbo metu išanalizuotas perdavimo klaidų modelis, kurio pagalba galima greitai ir efektyviai generuoti testinius rinkinius perdavimo vėlinimo gedimams testuoti.

Sukūrus sistemą, buvo atliktas eksperimentinis jos tyrimas. Gauti panaudoto algoritmo rezultatai yra pateikti priede. Vykdant schemų testavimo darbus išryškėjo kai kurios sistemos stipriosios savybės:

- „Nulūžus“ (nenumatytas programos veikimo sutrikdymas) programai visi sugeneruoti rinkiniai yra išsaugomi rinkinių faile, ir norint pratęsti generavimo darbą toliau, visas procesas vyksta nuo paskutinio gero rinkinio. Taip yra sutaupomas laikas.
- Schemos (aprašytos funkcijos) į sistemą yra užkraunamos kaip atskiri komponentai. Taip sistema tampa labiau dinamiškesnė.
- Papildomai yra įdiegtas automatinis testavimas. Automatiškai galima ištestuoti tiek vieną pasirinktą schemą tiek ir visas pradedant nuo pirmosios.

Sukurta vatotojo sąsaja yra paprasta ir lengvai suprantama. Padaryti trys pagrindiniai valdymo meniu, informaciniai laukeliai ir aišku generavimo proceso paleidimo pagrindinis mygtukas.

LITERATŪRA

1. Bareiša, Eduardas; Jusas, Vacius; Motiejūnas, Kęstutis; Šeinauskas, Rimantas. The Criteria of Functional Delay Test Quality Assessment. Proceedings of the 10th euromicro conference on digital system design architectures, methods and tools (DSD'2007), 29 August - 31 August, 2007, Luebeck, Germany, pp. 207-214. vadovo paimta teorijos medžiaga „Functional transition fault generation".
2. Electronics conference and "Electronics and Electrical Engineering" journal official website. Prieiga per internetą

< http://www.ee.ktu.lt/journal/2005/3/Bareisa.pdf >.

3. Electronics conference and "Electronics and Electrical Engineering" journal official website. Prieiga per internetą

< http://www.ee.ktu.lt/journal/2003/5/Abraitis.pdf >.

4. Electronics conference and "Electronics and Electrical Engineering" journal official website. Prieiga per internetą

< http://www.ee.ktu.lt/journal/2005/7/Jusas.pdf >.

5. INFORMACINĖS TECHNOLOGIJOS' 2005:. Prieiga per internetą

< http://internet.ktu.lt/lt/mokslas/konf05/konf_02/IT2005/Sekc08.pdf >.

6. International Test Conference. Available from Internet

< http://www.itcprogramdev.org/35th_anniv/pdfs/1985_1.pdf >.

7. Electronics conference and "Electronics and Electrical Engineering" journal official website. Prieiga per internetą

< http://www.ee.ktu.lt/journal/2005/7/Tamosevicius.pdf >.

8. Synopsys World Leader in EDA Software and Services. Prieiga per internetą

< http://www.synopsys.com/products/test/tetramax_ds.pdf >.

9. L.C. Smith - Home Page. Prieiga per internetą:

< www.ecs.syr.edu/organizations/GROVE/data/Tetramax_UG.pdf >.

10. Elektronika.lt - Lietuviškas ITT, elektronikos portalas. Prieiga per internetą

< http://www.elektronika.lt/articles/electronics/2132/ >.

11. Ron's Homepage, Department of Electrical Engineering National Tsing-Hua University. Prieiga per internetą

< http://larc.ee.nthu.edu.tw/~cylo/document/Tutorial1.ppt >.

12. KTU habilitacijos procedūra. Prieiga per internetą

< http://www.ktu.lt/habilitacija/jusas/Jusas_hp.pdf >.

13. Test and Test Bench Generation by Algorithmic Models. Prieiga per internetą < http://www.elen.ktu.lt/etg/results.html >.

14. wikipedia – nemokama enciklopedija. Prieiga per internetą < http://lt.wikipedia.org/wiki/Lustas >.

15. wikipedia – nemokama enciklopedija. Prieiga per internetą < http://lt.wikipedia.org/wiki/Integrin%C4%97_mikroschema >.

TERMINŲ, SANTRUMPŲ ŽODYNAS

TSGS – Testinių sekų generavimo schema.

TVP - Testinių vektorių poros.

ĮSS - Įėjimo sinchroninis signalas.

TKS - Testuojama kombinacinė schema.

IšSS - Išėjimo schemos sinchroninis signalas.

SRAS - Schemos reakcijos analizavimo schema.

angl. *Nonrobust* - Netiesioginė įtaka.

angl. *Robust* - Tiesioginė įtaka.

angl. *stuck-at* Konstantiniai gedimai.

angl. *transition delay faults* - Perdavimo vėlinimo gedimai.

angl. *path delay faults* - Kritinio kelio vėlinimo gedimai.

IG - Integrinis grandynas

6. PRIEDAI

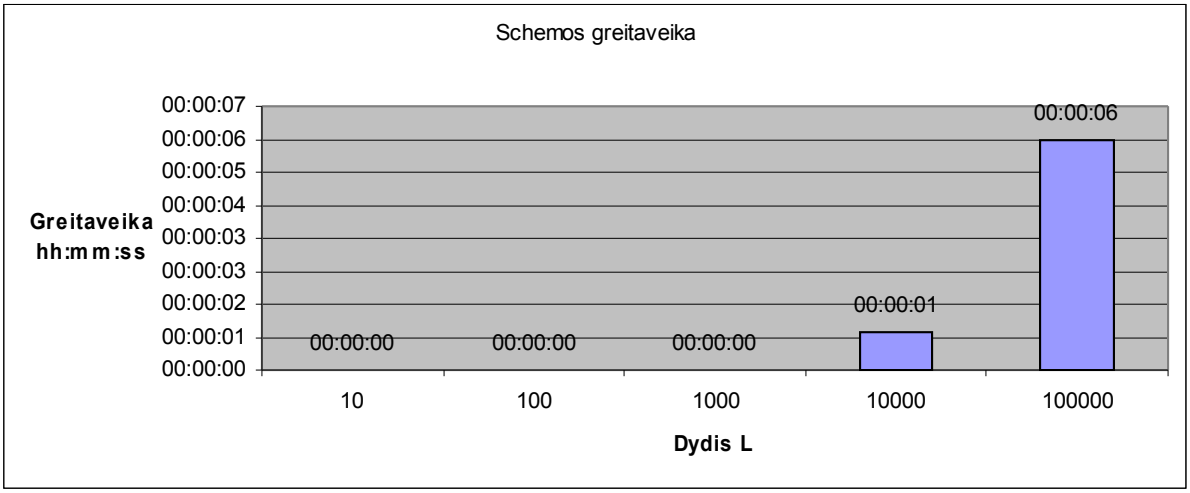
6.1. TESTAVIMO REZULTATAI

6.1.1.b01

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b01	102	62	65	10	0 sec.
2	b01	118	66	69	100	0 sec.
3	b01	118	66	69	1000	0 sec.

4	b01	118	66	69	10000	1 sec.
5	b01	118	66	69	100000	6 sec.

4. Lentelė. b01 schemos rezultatai.

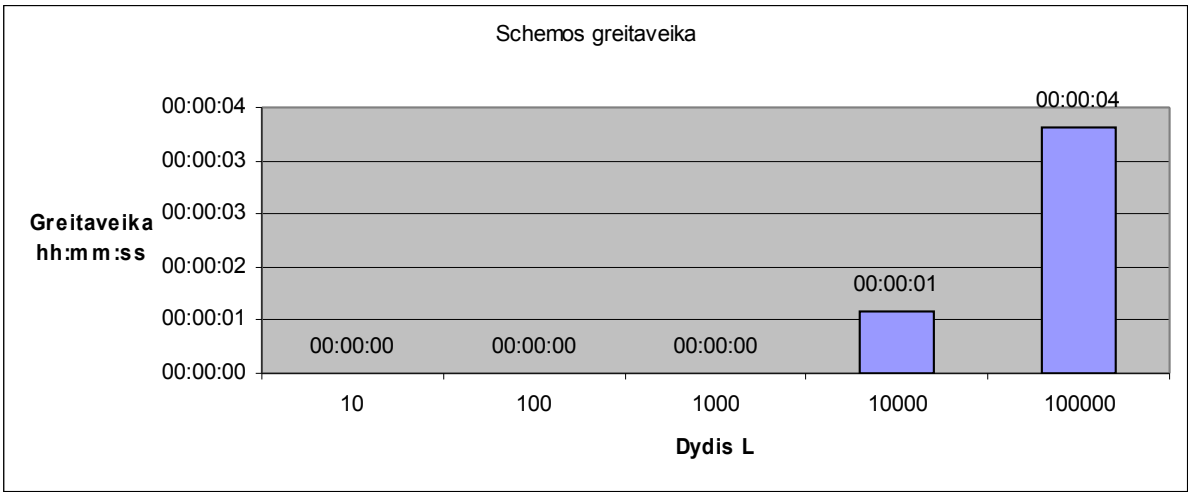


19. pav. b01 schemos greitaveika.

6.1.2.b02

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b02	50	23	28	10	0 sec.
2	b02	68	29	32	100	0 sec.
3	b02	68	29	32	1000	0 sec.
4	b02	68	29	32	10000	1 sec.
5	b02	68	29	32	100000	4 sec.

5. Lentelė. b02 schemos rezultatai.



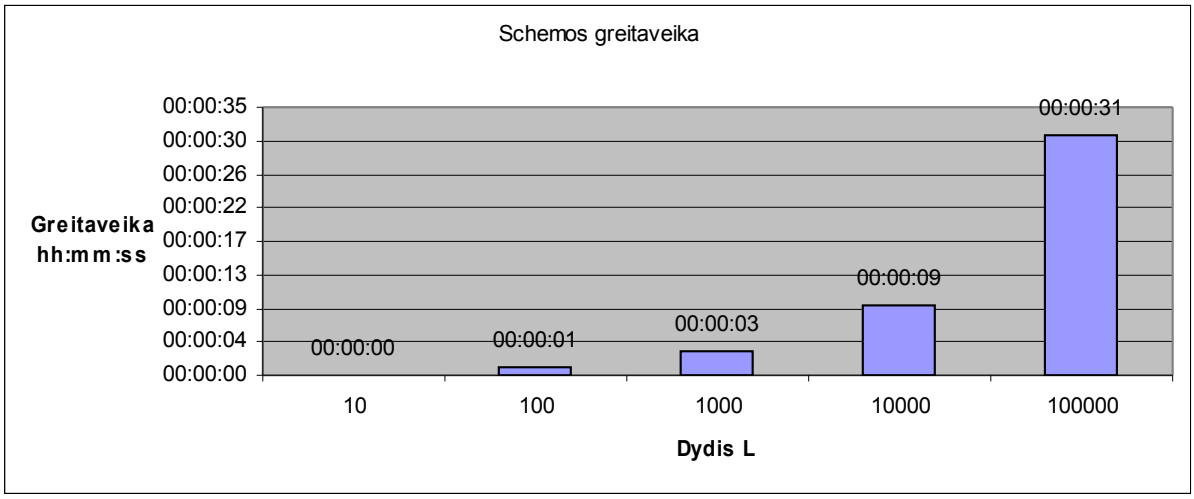
20. pav. b02 schemos greitaveika.

6.1.1.b03

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas

1	b03	334	477	483	10	0 sec.
2	b03	746	709	714	100	1 sec.
3	b03	836	736	736	1000	3 sec.
4	b03	860	742	742	10000	9 sec.
5	b03	860	742	742	100000	31 sec.

6. Lentelė. b03 schemos rezultatai.

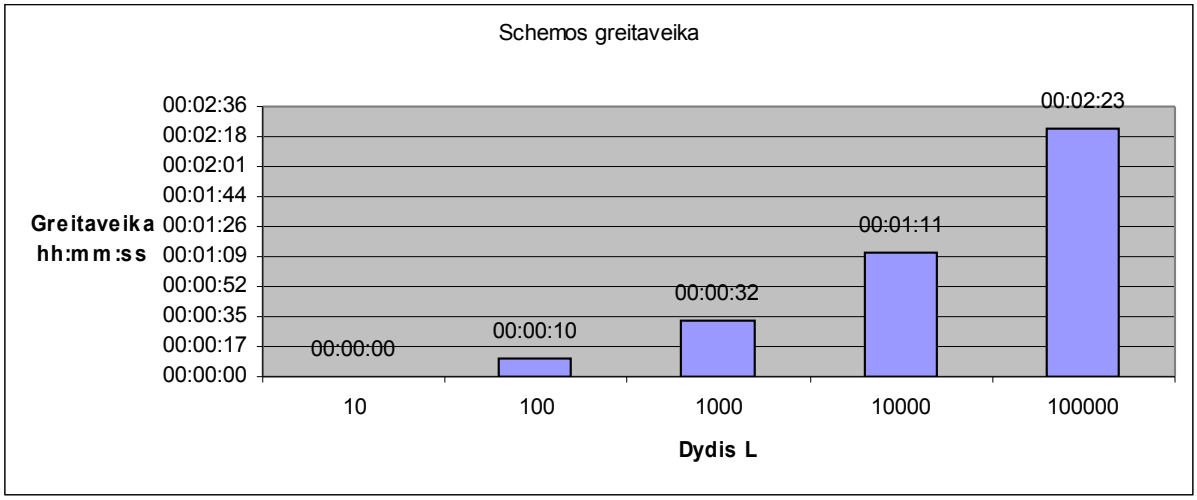


21. pav. b03 schemos greitaveika.

6.1.1.b04

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b04	414	1257	1291	10	0 sec.
2	b04	1332	2020	2013	100	10 sec.
3	b04	1890	2337	2346	1000	32 sec.
4	b04	2156	2454	2468	10000	1 min.11 sec.
5	b04	2156	2454	2468	100000	2 min.23 sec.

7. Lentelė. b04 schemos rezultatai.

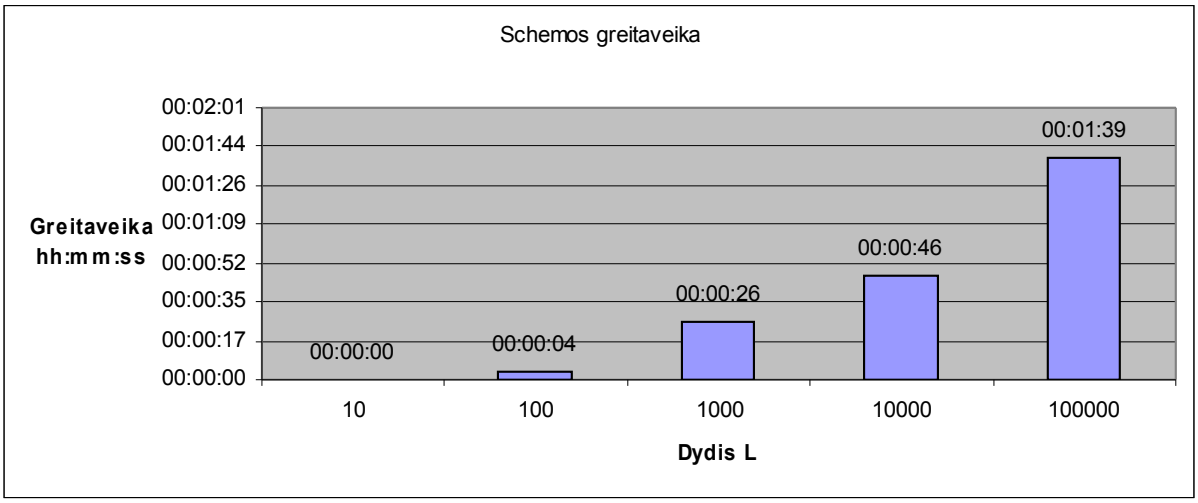


22. pav. b04 schemos greitaveika.

6.1.1.b05

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b05	482	1242	1359	10	0 sec.
2	b05	1220	1788	1861	100	4 sec.
3	b05	1738	2048	2111	1000	26 sec.
4	b05	1838	2095	2157	10000	46 sec.
5	b05	1838	2095	2157	100000	1 min.39 sec.

8. Lentelė. b05 schemos rezultatai.

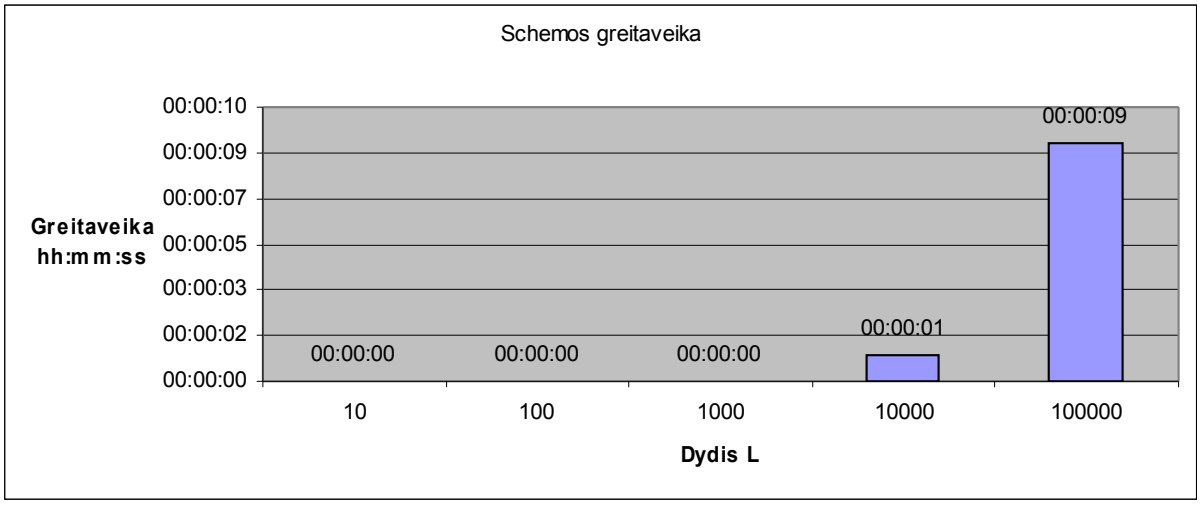


23. pav. b05 schemos greitaveika.

6.1.1.b06

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b06	112	82	86	10	0 sec.
2	b06	124	88	89	100	0 sec.
3	b06	132	90	91	1000	0 sec.
4	b06	132	90	91	10000	1 sec.
5	b06	132	90	91	100000	9 sec.

9. Lentelė. b06 schemos rezultatai.

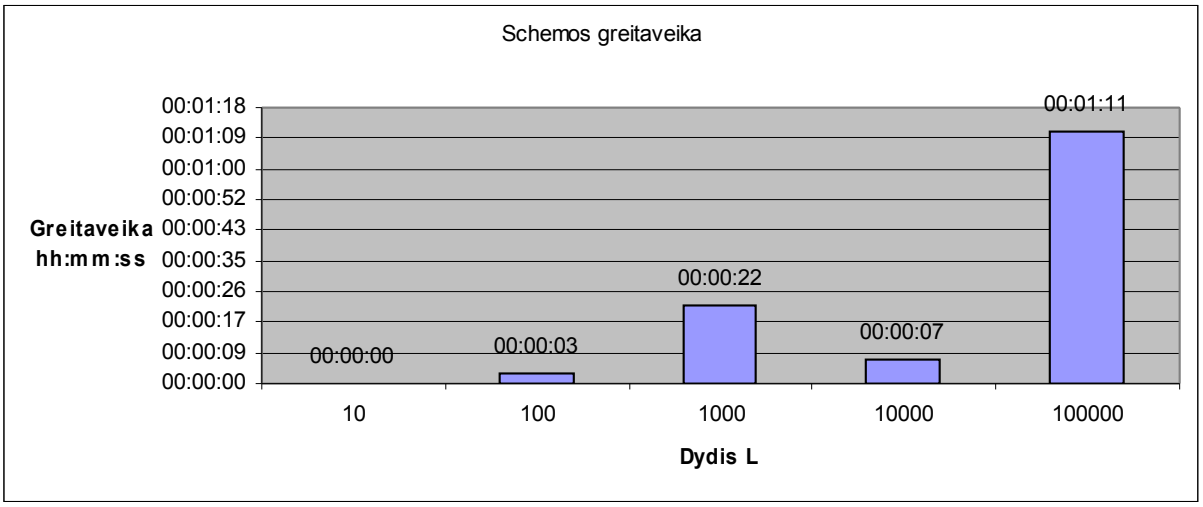


24. pav. b01 schemos greitaveika.

6.1.1.b07

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b07	692	1143	1153	10	0 sec.
2	b07	1228	1416	1474	100	3 sec.
3	b07	1854	2033	2032	1000	22 sec.
4	b07	1854	2033	2032	10000	7 sec.
5	b07	1854	2033	2032	100000	1 min.11 sec.

10. Lentelė. b07 schemos rezultatai.



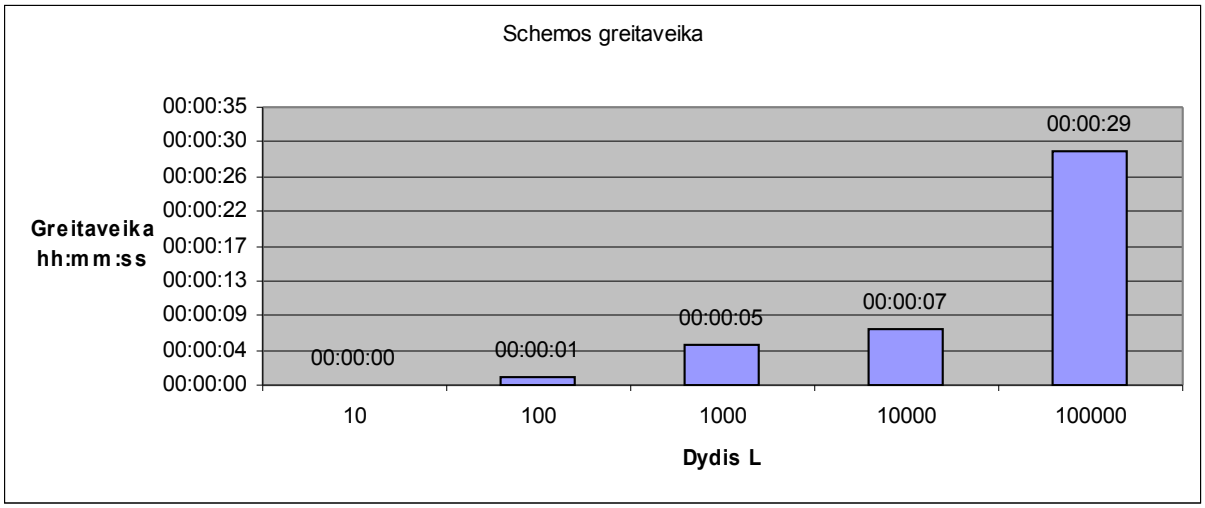
25. pav. b07 schemos greitaveika.

6.1.1.b08

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b08	228	228	226	10	0 sec.
2	b08	500	339	358	100	1 sec.

3	b08	638	395	405	1000	5 sec.
4	b08	680	417	417	10000	7 sec.
5	b08	680	417	417	100000	29 sec.

11. Lentelė. b08 schemos rezultatai.

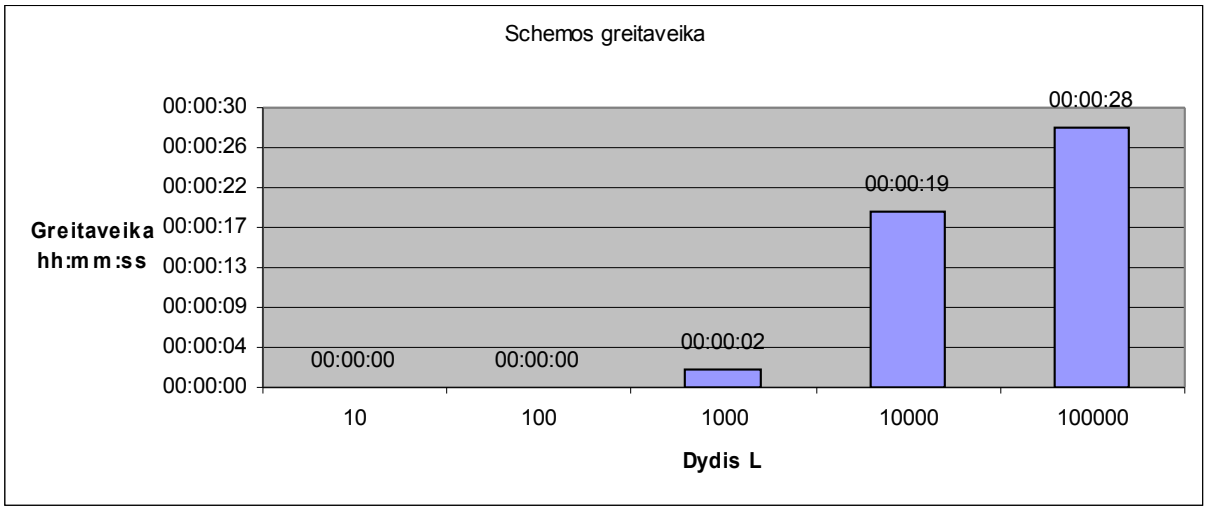


26. pav. b08 schemos greitaveika.

6.1.1.b09

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b09	224	341	344	10	0 sec.
2	b09	250	347	353	100	0 sec.
3	b09	326	499	514	1000	2 sec.
4	b09	630	908	923	10000	19 sec.
5	b09	630	908	923	100000	28 sec.

12. Lentelė. b09 schemos rezultatai.

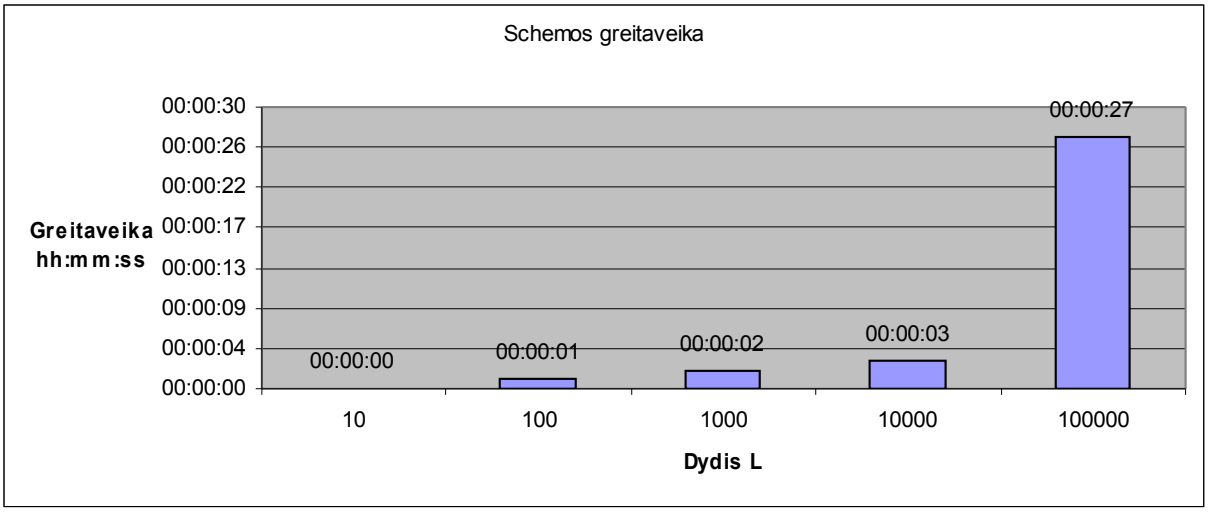


27. pav. b09 schemos greitaveika.

6.1.1.b10

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b10	398	356	350	10	0 sec.
2	b10	630	447	445	100	1 sec.
3	b10	704	467	468	1000	2 sec.
4	b10	708	469	468	10000	3 sec.
5	b10	708	469	468	100000	27 sec.

13. Lentelė. b10 schemos rezultatai.

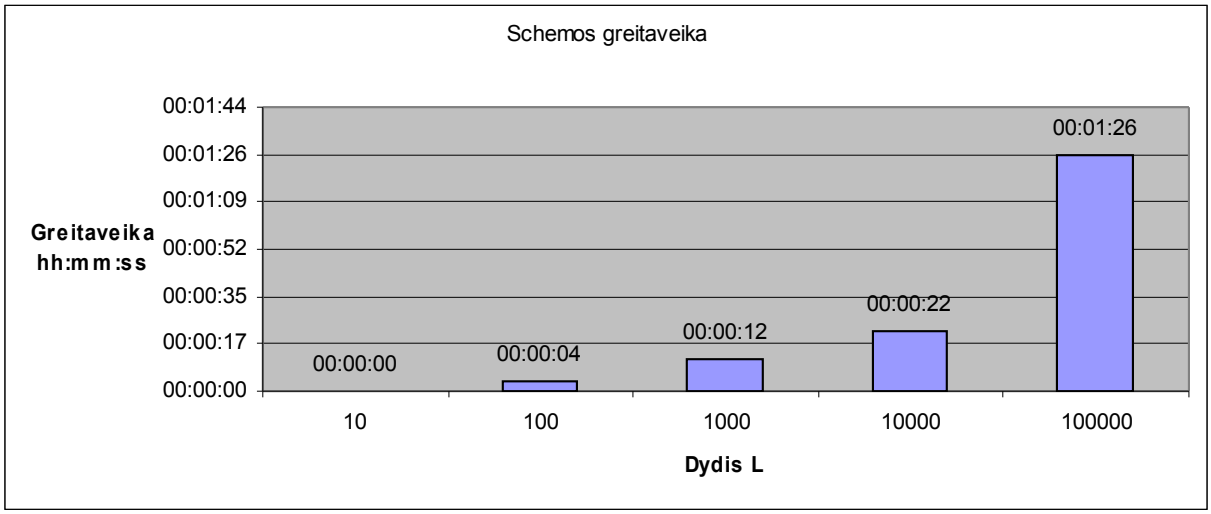


28. pav. b10 schemos greitaveika.

6.1.1.b11

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b11	632	840	850	10	0 sec.
2	b11	1142	1167	1210	100	4 sec.
3	b11	1452	1352	1338	1000	12 sec.
4	b11	1524	1374	1366	10000	22 sec.
5	b11	1524	1374	1366	100000	1 min.26 sec.

14. Lentelė. b11 schemos rezultatai.

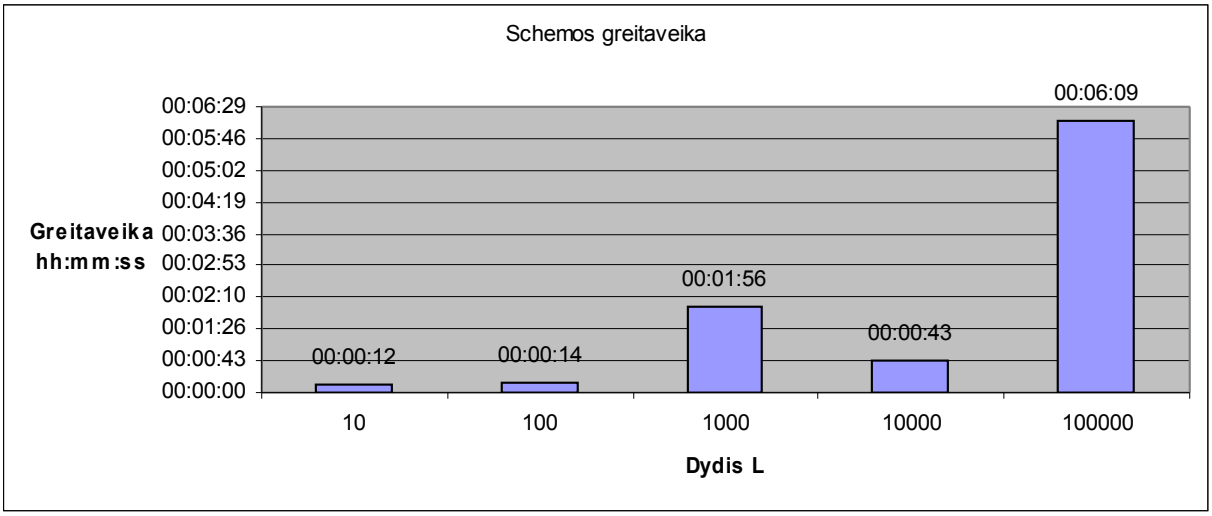


29. pav. b11 schemos greitaveika.

6.1.1.b12

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b12	2502	3008	3012	10	12 sec.
2	b12	3178	3321	3352	100	14 sec.
3	b12	3956	3834	3866	1000	1 min.56 sec.
4	b12	3956	3834	3866	10000	43 sec.
5	b12	3956	3834	3866	100000	6 min.9 sec.

15. Lentelė. b12 schemos rezultatai.



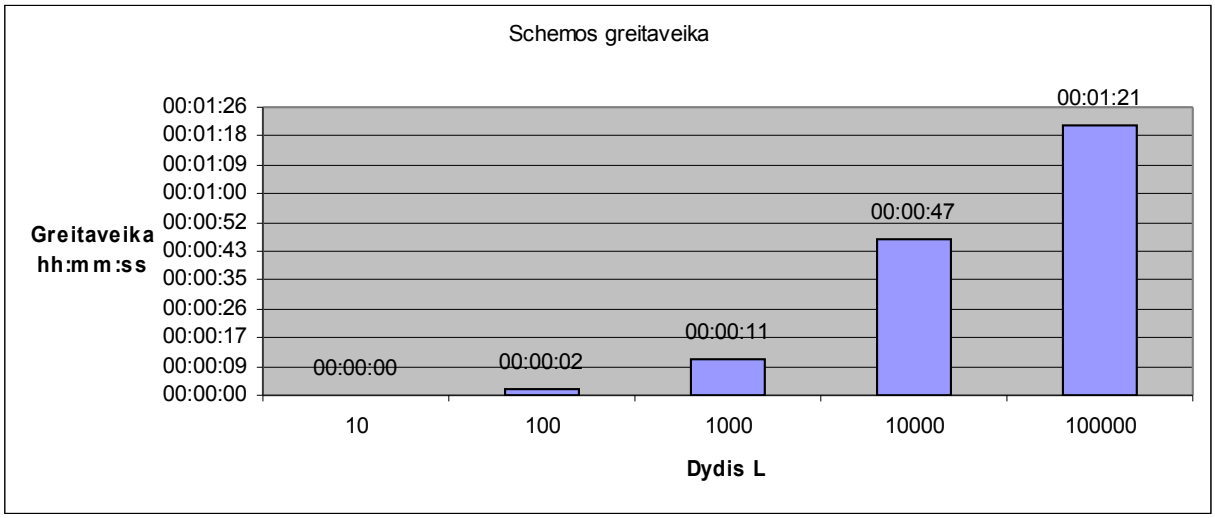
30. pav. b12 schemos greitaveika.

6.1.1.b13

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b13	380	701	704	10	0 sec.
2	b13	680	834	849	100	2 sec.

3	b13	924	962	957	1000	11 sec.
4	b13	1028	997	1004	10000	47 sec.
5	b13	1028	997	1004	100000	1 min.21 sec.

16. Lentelė. b13 schemos rezultatai.

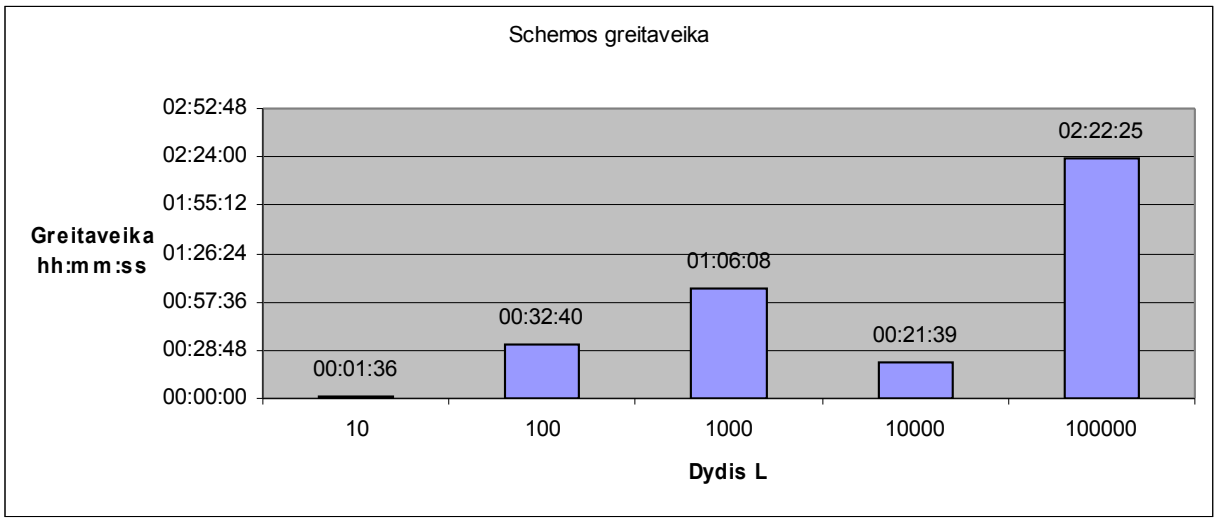


31. pav. b13 schemos greitaveika.

6.1.1.b14

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b14	1342	12479	12371	10	1 min.36 sec.
2	b14	7482	24735	24653	100	32 min.40 sec.
3	b14	12254	28744	28741	1000	1 val.6 min.8 sec.
4	b14	12254	28744	28741	10000	21 min.39 sec.
5	b14	12254	28744	28741	100000	2 val.22 min.25 sec.

17. Lentelė. b14 schemos rezultatai.

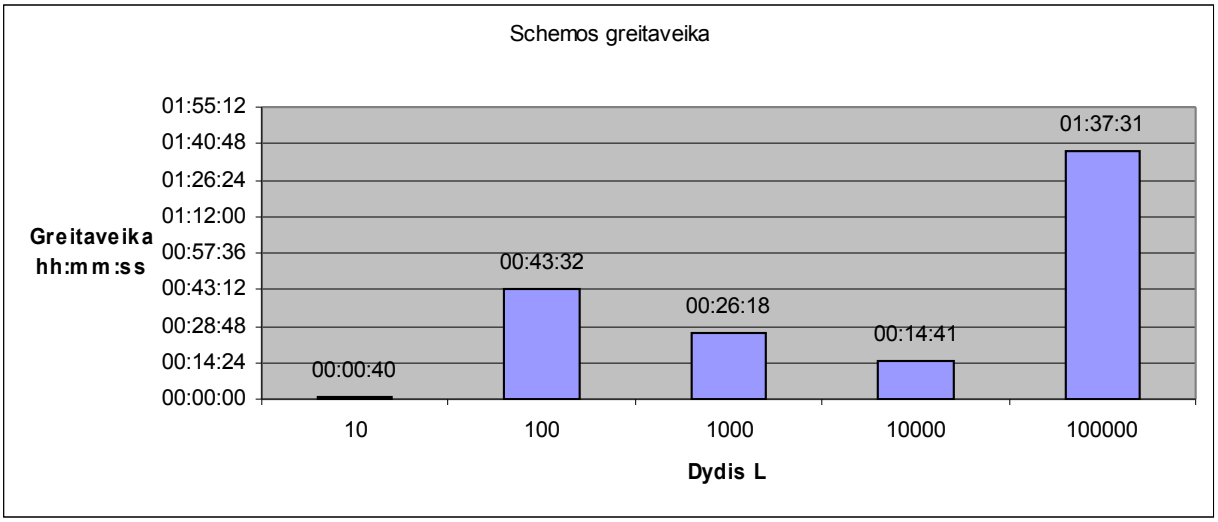


32. pav. b14 schemos greitaveika.

6.1.1.b14_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b14_1	848	9203	9549	10	40 sec.
2	b14_1	10048	25538	25656	100	43 min.32 sec.
3	b14_1	12004	27015	27083	1000	26 min.18 sec.
4	b14_1	12004	27015	27083	10000	14 min.41 sec.
5	b14_1	12004	27015	27083	100000	1 val.37 min.31 sec.

18. Lentelė. b14_1 schemos rezultatai.

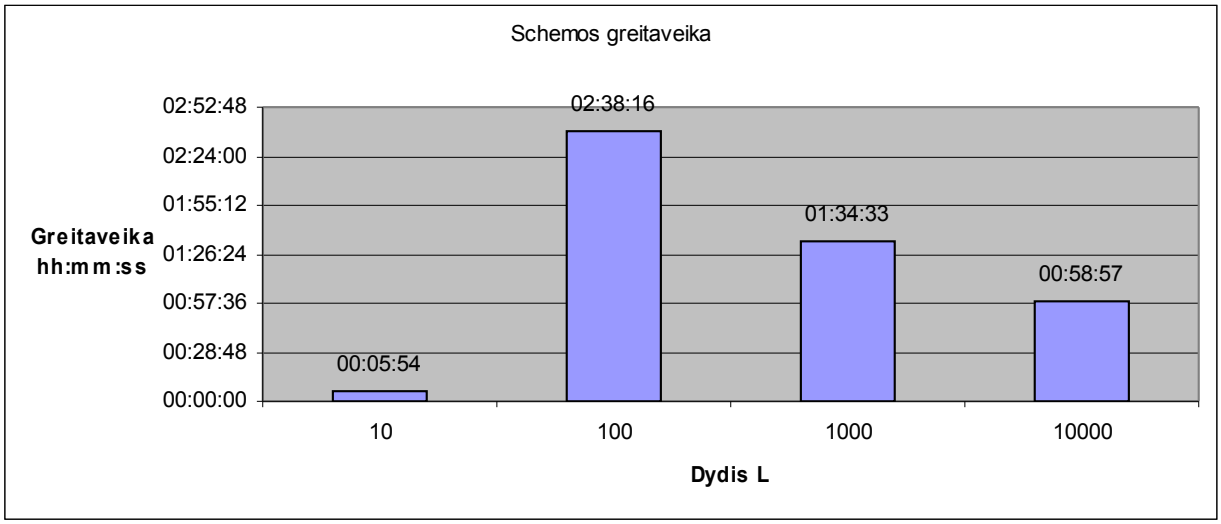


33. pav. b14_1 schemos greitaveika.

6.1.1.b15_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b15_1	2396	11344	11351	10	5 min.54 sec.
2	b15_1	15084	30946	31037	100	2 val.38 min.16 sec.
3	b15_1	16644	34739	34973	1000	1 val.34 min.33 sec.
4	b15_1	16644	34739	34973	10000	58 min.57 sec.

19. Lentelė. b15_1 schemos rezultatai.

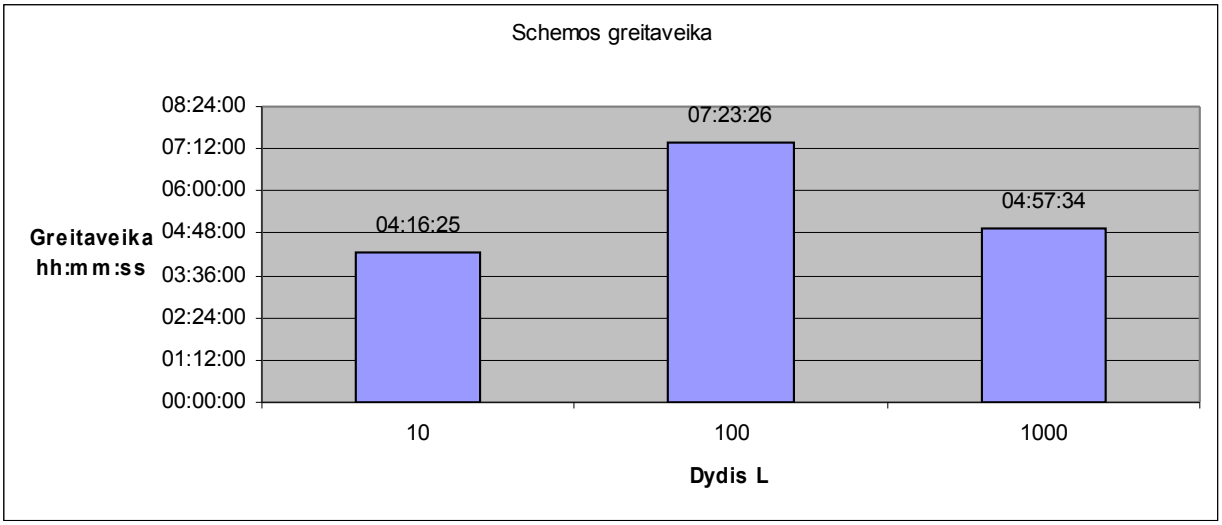


34. pav. b15_1 schemos greitaveika.

6.1.1.b17

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b17	12718	59344	60280	10	4 val.16 min.25 sec.
2	b17	21162	74187	74965	100	7 val.23 min.26 sec.
3	b17	21162	74187	74965	1000	4 val.57 min.34 sec.

20. Lentelė. b17 schemos rezultatai.



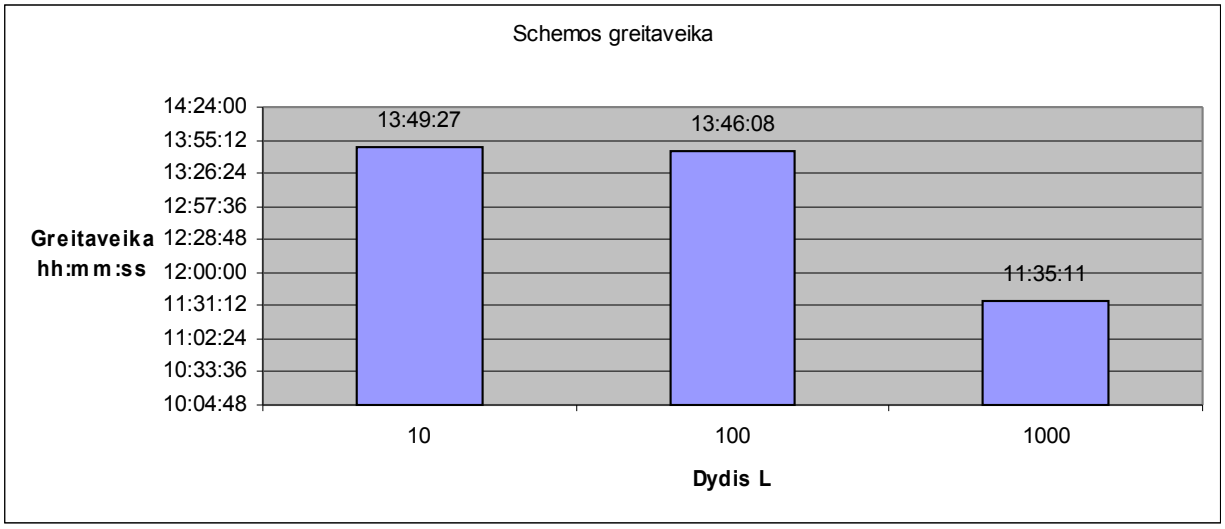
35. pav. b17 schemos greitaveika.

6.1.1.b17_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b17_1	16344	63734	63876	10	13 val.49 min.27 sec.

2	b17_1	20478	70866	70925	100	13 val.46 min.8 sec.
3	b17_1	20478	70866	70925	1000	11 val.35 min.11 sec.

21. Lentelė. b17_1 schemos rezultatai.

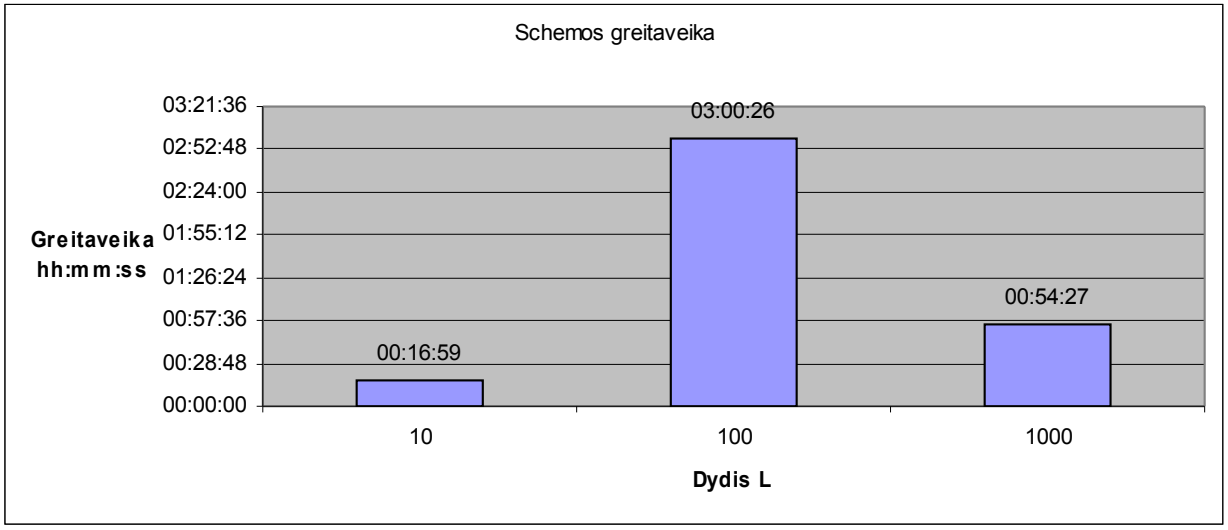


36. pav. b17_1 schemos greitaveika.

6.1.1.b20

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b20	4896	43008	42672	10	16 min.59 sec.
2	b20	19584	68934	67981	100	3 val.0 min.26 sec.
3	b20	19584	68934	67981	1000	54 min.27 sec.

22. Lentelė. b20 schemos rezultatai.



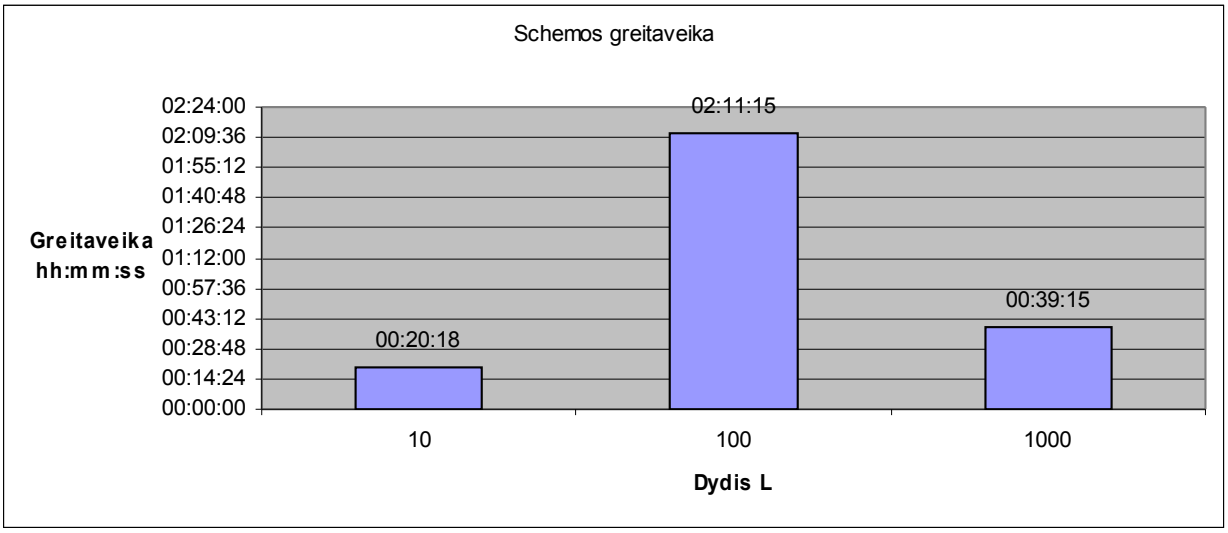
37. pav. b20 schemos greitaveika.

6.1.1.b20_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė	L	Laikas

				įtaka		
1	b20_1	6768	46520	46471	10	20 min.18 sec.
2	b20_1	19806	65416	65280	100	2 val.11 min.15 sec.
3	b20_1	19806	65416	65280	1000	39 min.15 sec.

23. Lentelė. b20_1 schemos rezultatai.

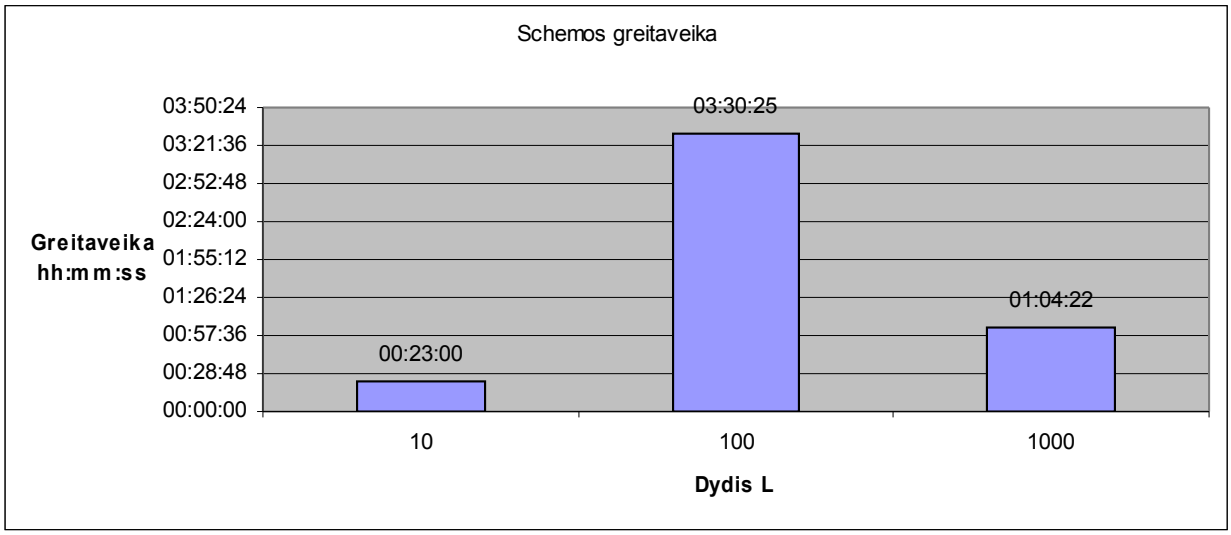


38. pav. b20_1 schemos greitaveika.

6.1.1.b21

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b21	5494	43320	43295	10	23 min.0 sec.
2	b21	19674	67544	67598	100	3 val.30 min.25 sec.
3	b21	19674	67544	67598	1000	1 val.4 min.22 sec.

24. Lentelė. b21 schemos rezultatai.

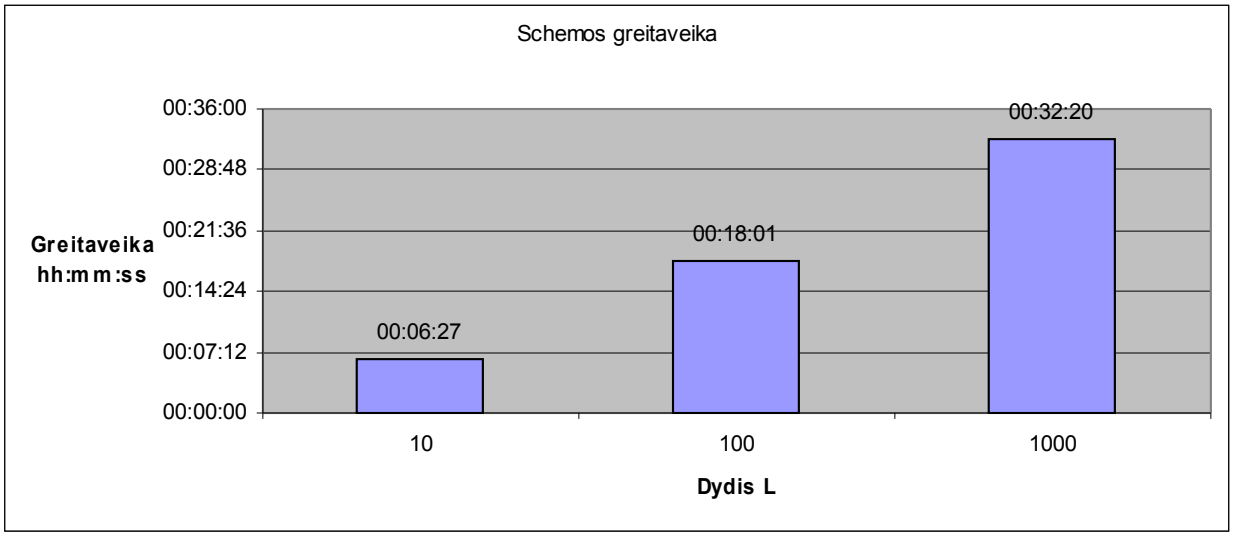


39. pav. b21 schemos greitaveika.

6.1.1.b21_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b21_1	6190	1007278	895076	10	6 min.27 sec.
2	b21_1	9490	1030921	908960	100	18 min.1 sec.
3	b21_1	10058	1031345	909406	1000	32 min.20 sec.

25. Lentelė. b21_1 schemos rezultatai.

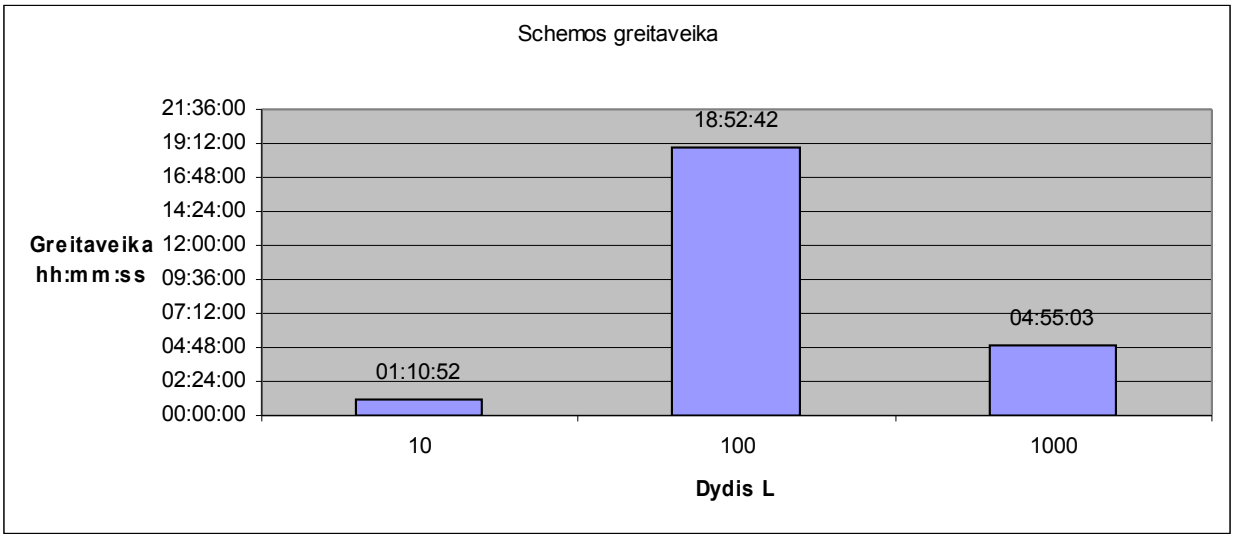


40. pav. b21_1 schemos greitaveika.

6.1.1.b22

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b22	7138	67685	68027	10	1 val.10 min.52 sec.
2	b22	38130	113989	113839	100	18 val.52 min.42 sec.
3	b22	38130	113989	113839	1000	4 val.55 min.3 sec.

26. Lentelė. b22 schemos rezultatai.

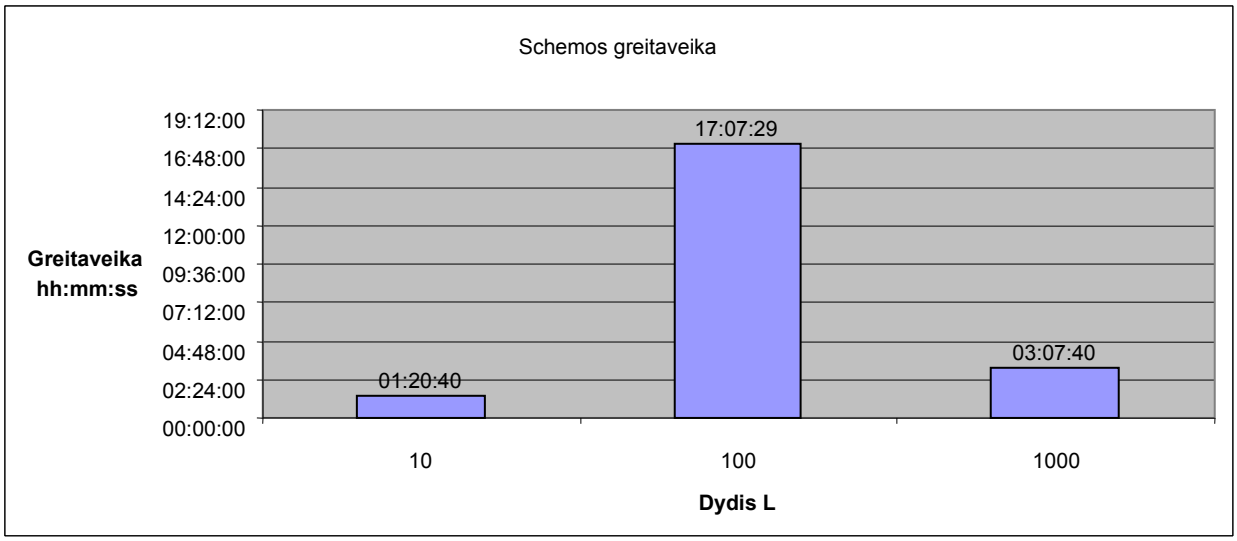


41. pav. b22 schemos greitaveika.

6.1.1.b22_1

Nr.	Schemos pav.	Rinkinių kiekis	Tiesioginė įtaka	Netiesioginė įtaka	L	Laikas
1	b22_1	10068	75184	75097	10	1 val.20 min.40 sec.
2	b22_1	37564	110269	110145	100	17 val.7 min.29 sec.
3	b22_1	37564	110269	110145	1000	3 val.7 min.40 sec.

27. Lentelė. b22_1 schemos rezultatai.



42. pav. b22_1 schemos greitaveika.

6.2. NAUDOTI ALGORITMAI

1. Funkcinių vėlinimo testų generavimo algoritmas be panašių generavimo
2. Nuskaitom iš failo funkcinį testą,nustatom rinkinių kiekį tk; (Nurodom L)
3. $X=||x_{i,j}:=0||_{2n \times 4m}$; tp:=1;
4. DO WHILE (tp≠tk)
5. //Testiniams rinkiniams paskaičiuojam matricą X
6. DO t:=tp, , tk, (kas 2) ;
7. $P^1:=P^t$; $R^1:=f(P^1)$; $P^2:=P^{t+1}$; $R^2:=f(P^2)$;
8. DO i: =1,2,3,...,n; P^3: $=P^2$; d:=1- p^1_i;
9. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=1- p^2_i$; $R^3=f(P^3)$;
10. DO j: =1,2,3,...,m; c:=3- r^1_j;
11. IF ($r^3_j \neq r^2_j$) THEN
12. IF($r^2_j \neq r^1_j$) THEN $x_{2i-d,4j-c}:=1$;
13. ELSE $x_{2i-d,4j-c+2}:=1$;
14. ENDIF;
15. ENDIF;
16. ENDDO;
17. ENDIF;
18. ENDDO;
19. ENDDO; tp:=tk;
20. // Nagrinėjam 2L atsitiktinai sugeneruotų rinkinių ir iš jų išrenkam geriausią porą pagal esamą užpildymą
21. h:=tk+1; SKM:=0;
22. DO k:=1,2,3, , L;
23. $P^2:= (p^2_1,p^2_2,...,p^2_i,...,p^2_n)$, p^2_i:=Random(0,1); $R^2:=f(P^2)$;
24. $P^1:= (p^1_1,p^1_2,...,p^1_i,...,p^1_n)$, p^1_i:=Random(0,1); $R^1:=f(P^1)$;
25. // Gali būti dar dvi alternatyvos: P^1 lygus invertuotam P^2 arba invertuotoms aktyvioms P^2 reikšmėms
26. SK1:=0;SK2:=0;
27. DO i: =1,2,3,...,n; P^3: $=P^2$; d:=1- p^1_i;
28. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=1- p^2_i$; $R^3=f(P^3)$;
29. DO j: =1,2,3,...,m; c:=3- r^1_j;
30. IF ($r^3_j \neq r^2_j$) THEN
31. IF($r^2_j \neq r^1_j$) THEN IF ($x_{2i-d,4j-c}=0$)
32. THEN SK1:= SK1+1;ENDIF;
33. ELSE IF ($x_{2i-d,4j-c+2}=0$)
34. THEN SK2:= SK2+1; ENDIF;
35. ENDIF;
36. ENDIF;
37. ENDDO;
38. ENDIF;
39. ENDDO;
40. IF ((SK1+SK2)>SKM) THEN SKM:=SK1+SK2; $P^{1M}:=P^1$;$P^{2M}:=P^2$; ENDIF;
41. ENDDO;
42. IF (SKM>0) THEN $P^h:=P^{1M}$; $P^{h+1}:=P^{2M}$;h:=h+2; tk:=h; ENDIF;
43. ENDWHILE;

1. Funkcinių vėlinimo testų generavimo algoritmas su panašių nagrinėjimu
2. Nuskaitom iš failo funkcinį testą,nustatom rinkinių kiekį tk; (Nurodom L)
3. $X=||x_{i,j}:=0||_{2n \times 4m}$; tp:=1;
4. DO WHILE (tp≠tk) tpp:=tp; PZ:=0; h:=tp;
5. //Testiniams rinkiniams paskaičiuojam matricą X
6. DO t:=tp,…, tk, (kas 2) ;
7. $P^1:=P^t$; $R^1:=f(P^1)$; $P^2:=P^{t+1}$; $R^2:=f(P^2)$;
8. DO i: =1,2,3,...,n; P^3: =P^2; d:=1- p^1_i;
9. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=p^1_i$; $R^3=f(P^3)$;
10. DO j: =1,2,3,...,m; c:=3- r^1_j;
11. IF ($r^3_j \neq r^2_j$) THEN
12. IF($r^2_j \neq r^1_j$) THEN IF ($x_{2i-d,4j-c}$=0
13. THEN PZ:=1; $x_{2i-d,4j-c}$:=1;ENDIF;
14. ELSE IF ($x_{2i-d,4j-c+2}$=0)
15. THEN PZ:=1; $x_{2i-d,4j-c+2}$:=1;ENDIF;
16. ENDIF;
17. ENDIF;
18. ENDDO;
19. ENDIF;
20. ENDDO;
21. IF (PZ=1) THEN $P^h:=P^1$; $P^{h+1}:=P^2$;h:=h+2; tk:=h-1; ENDIF;
22. ENDDO; tp:=tk;
23. // Panašių generavimas. Turėtų būti galimybė išjungti pagal požymį.
24. h:=tk+1;
25. DO ii: =1,2,3,...,n;
26. DO t:=tpp,…, tk, (kas 2) ;
27. $P^1:=P^t$; $R^1:=f(P^1)$; $P^2:=P^{t+1}$; $p^2_{ii}:=1- p^2_{ii}$; $R^2:=f(P^2)$; SK:=0;
28. DO i: =1,2,3,...,n; P^3: =P^2; d:=1- p^1_i;
29. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=p^1_i$; $R^3=f(P^3)$;
30. DO j: =1,2,3,...,m; c:=3- r^1_j;
31. IF ($r^3_j \neq r^2_j$) THEN
32. IF($r^2_j \neq r^1_j$) THEN IF ($x_{2i-d,4j-c}$=0
33. THEN SK:=SK+1;ENDIF;
34. ELSE IF ($x_{2i-d,4j-c+2}$=0
35. THEN SK:=SK+1;ENDIF;
36. ENDIF;
37. ENDIF;
38. ENDDO;
39. ENDIF;
40. ENDDO;
41. IF (SK>0) THEN $P^h:=P^1$; $P^{h+1}:=P^2$;h:=h+2; tk:=h-1; ENDIF;
42. ENDDO;
43. ENDDO;
44. // Nagrinėjam 2L atsitiktinai sugeneruotų rinkinių ir iš jų išrenkam geriausią porą
45. SKM:=0;
46. DO k:=1,2,3, , L;
47. $P^2:= (p^2_1,p^2_2,...,p^2_i,...,p^2_n)$, p^2_i:=Random(0,1); $R^2:=f(P^2)$;
48. $P^1:= (p^1_1,p^1_2,...,p^1_i,...,p^1_n)$, p^1_i:=Random(0,1); $R^1:=f(P^1)$;

```
49.                 // Gali būti dar dvi alternatyvos: P[1] lygus invertuotam P[2] arba invertuotoms
                        aktyvioms  P[2] reikšmėms
50.                 SK1:=0;SK2:=0;
51.                 DO i: =1,2,3,...,n; P[3]: =P[2]; d:=1- p[1]i;
52.                     IF (p[1]i≠p[2]i) THEN p[3]i:=p[1]i; R[3]=f(P[3]);
53.                         DO j: =1,2,3,...,m; c:=3- r[1]j;
54.                             IF (r[3]j≠r[2]j) THEN
55.                                 IF(r[2]j≠r[1]j) THEN IF (x2i-d,4j-c=0)
56.                                                         THEN SK1:= SK1+1;ENDIF;
57.                                     ELSE IF (x2i-d,4j-c+2=0)
58.                                             THEN SK2:= SK2+1; ENDIF;
59.                                 ENDIF;
60.                             ENDIF;
61.                         ENDDO;
62.                     ENDIF;
63.                 ENDDO;
64.                 IF ((SK1+SK2)>SKM) THEN SKM:=SK1+SK2; P[1M]:=P[1];P[2M]:=P[2]; ENDIF;
65.             // Imant SK1+SK2, SK1, arba SK2 gaunami trys algoritmo variantai
66.             ENDDO;
67.             IF (SKM>0) THEN P[h]:=P[1M]; P[h+1]:=P[2M]; h:=h+2; tk:=h-1; ENDIF;
68.     ENDWHILE;
```

1. Funkcinių vėlinimo testų generavimo algoritmas su panašių nagrinėjimu (Papildytas variantas)
2. Nuskaitom iš failo funkcinį testą,nustatom rinkinių kiekį tk; (Nurodom L)
3. $X=||x_{i,j}:=0||_{2n \times 4m}$; tp:=1;
4. DO WHILE (tp≠tk) tpp:=tp; PZ:=0; h:=tp;
5. //Testiniams rinkiniams paskaičiuojam matricą X
6. DO t:=tp,…, tk, (kas 2) ;
7. $P^1:=P^t$; $R^1:=f(P^1)$; $P^2:=P^{t+1}$; $R^2:=f(P^2)$;
8. DO i: =1,2,3,...,n; P^3: $=P^2$; d:=1- p^1_i;
9. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=p^1_i$; $R^3=f(P^3)$;
10. DO j: =1,2,3,...,m; c:=3- r^1_j;
11. IF ($r^3_j \neq r^2_j$) THEN
12. IF($r^2_j \neq r^1_j$) THEN IF ($x_{2i-d,4j-c}=0$) THEN $x_{2i-d,4j-c}:=1$; PZ:=1;
13. ELSE $x_{2i-d,4j-c}:= x_{2i-d,4j-c} +1/(x_{2i-d,4j-c}+1)$;
14. ENDIF;
15. ELSE IF ($x_{2i-d,4j-c+2}=0$) THEN $x_{2i-d,4j-c+2}:=1$; PZ:=1;
16. ELSE $x_{2i-d,4j-c+2}:= x_{2i-d,4j-c+2}+1/(x_{2i-d,4j-c+2}+1)$; ENDIF;
17. ENDIF;
18. ENDIF;
19. ENDDO;
20. ENDIF;
21. ENDDO;
22. IF (PZ=1) THEN $P^h:=P^1$; $P^{h+1}:=P^2$;h:=h+2; tk:=h-1; ENDIF;
23. ENDDO; tp:=tk+1;
24. // Panašių generavimas. Turėtų būti galimybė išjungti pagal požymį.
25. h:=tk+1;
26. DO ii: =1,2,3,...,n;
27. DO t:=tpp,…, tk, (kas 2) ;
28. $P^1:=P^t$; $R^1:=f(P^1)$; $P^2:=P^{t+1}$; $p^2_{ii}:=1- p^2_{ii}$; $R^2:=f(P^2)$; SK:=0;
29. DO i: =1,2,3,...,n; P^3: $=P^2$; d:=1- p^1_i;
30. IF ($p^1_i \neq p^2_i$) THEN $p^3_i:=p^1_i$; $R^3=f(P^3)$;
31. DO j: =1,2,3,...,m; c:=3- r^1_j;
32. IF ($r^3_j \neq r^2_j$) THEN
33. IF($r^2_j \neq r^1_j$) THEN IF ($x_{2i-d,4j-c}=0$
34. THEN SK:=SK+1;ENDIF;
35. ELSE IF ($x_{2i-d,4j-c+2}=0$
36. THEN SK:=SK+1;ENDIF;
37. ENDIF;
38. ENDIF;
39. ENDDO;
40. ENDIF;
41. ENDDO;
42. IF (SK>0) THEN $P^h:=P^1$; $P^{h+1}:=P^2$;h:=h+2; tk:=h-1; ENDIF;
43. ENDDO;
44. ENDDO;
45. // Nagrinėjam 2L atsitiktinai sugeneruotų rinkinių ir iš jų išrenkam geriausią porą
46. SKM:=0;
47. DO k:=1,2,3, , L;

```
48.         P²:= (p²1,p²2,...,p²i,...,p²n),  p²i:=Random(0,1); R²:=f(P²);
49.             P¹:= (p¹1,p¹2,...,p¹i,...,p¹n),  p¹i:=Random(0,1); R¹:=f(P¹);
50.             // Gali būti dar dvi alternatyvos: P¹ lygus invertuotam P² arba invertuotoms
                    aktyvioms  P² reikšmėms
51.             SK1:=0;SK2:=0; SK3:=0;SK4:=0;
52.             DO i: =1,2,3,...,n; P³: =P²; d:=1- p¹i;
53.                 IF (p¹i≠p²i) THEN p³i:=p¹i; R³=f(P³);
54.                     DO j: =1,2,3,...,m; c:=3- r¹j;
55.                         IF (r³j≠r²j) THEN
56.                             IF(r²j≠r¹j) THEN IF (x2i-d,4j-c=0)
57.                                 THEN SK1:= SK1+1;ENDIF;
58.                                 ELSE SK3:= SK3+ x2i-d,4j-c; ENDIF;
59.                                 ELSE IF (x2i-d,4j-c+2=0)
60.                                 THEN SK2:= SK2+1; ENDIF;
61.                                 ELSE SK4:= SK4+ x2i-d,4j-c+2; ENDIF;
62.                             ENDIF;
63.                         ENDIF;
64.                     ENDDO;
65.                 ENDIF;
66.             ENDDO;
67.             SK:=k1*SK1+ k2*SK2+ k3*SK3+ k4*SK4;
68.             IF (SK>SKM) THEN SKM:=SK; P1M:=P¹;P2M:=P²; ENDIF;
69.         // Imant skirtingus koeficientus k1,k2,k3,k4 gaunami įvairūs algoritmo variantai
70.         ENDDO;
71.         IF (SKM>0) THEN Ph:=P1M; Ph+1:=P2M; h:=h+2; tk:=h-1; ENDIF;
72. ENDWHILE;
```