KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Mindaugas Dobilas

Genetinių algoritmų taikymas imituojant sistemas aprašytas agregatiniu metodu

Magistro darbas

Darbo vadovas
doc. dr. Dalius Makackas

Kaunas
2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Mindaugas Dobilas

Genetinių algoritmų taikymas imituojant sistemas aprašytas agregatiniu metodu

Magistro darbas

Recenzentas
doc. dr. Gytis Vilutis

Darbo vadovas
doc. dr. Dalius Makackas

Kaunas
2010

SANTRAUKA ANGLŲ KALBA (SUMMARY)

Research area - genetic algorithms approach to aggregate and use modeling complex systems.

Work objective - the application of genetic algorithms in formal methods, systems imitation modeling, to find optimal settings.

This work proposed new usage of aggregate method and genetic algorithm to describe system models. The system parameters of the model are used as the individual's chromosomes in genetic algorithm, and the system model used as a utility function of genetic algorithm. It also proposed other aggregate approach, the genetic algorithm used for the transition operator, followed by the population structure. This allows the simulation of biological, agent, self-regulating systems.

The first section contained aggregate method, agents and agent systems. The second chapter written about methods of agents' behavior: neural networks, genetic algorithms and other techniques. There are description of methods operation, advantages and disadvantages and applications. The third section displayed three models, their specifications and results of the genetic algorithm and the aggregate method.

Combining genetic algorithms and aggregate method opens up new possibilities for simulation system performance assessment. These results confirm that combining genetic algorithm and the aggregate method to obtain the modeling systems characteristics, it is possible to model of biological, agent, self-regulating systems.

A formal system description method DEVS[1] dependent events control systems class models, as well as aggregate method, the work of the genetic algorithm used is appropriate and DEVS.

[1] Discrete event system specification [1]

Turinys

ĮVADAS

Genetiniai algoritmai yra tam tikra evoliucinių algoritmų klasė, naudojanti gamtoje egzistuojančius gyvybės evoliucinius mechanizmus: paveldėjimą, mutaciją, natūraliąją atranką ir rekombinaciją. Genetiniai algoritmai yra metodas analizuoti duomenis, jų dėka galima rasti apytikslį užduoties sprendimą. Apie 1985-sius Ilinojaus universitete įvyko pirmoji tarptautinė konferencija skirta genetiniams algoritmams, iki tol genetinių algoritmų tyrimai buvo daugiausia tik teoriniai [2].

Agregatinis metodas skirtas sistemoms aprašyti formalia kalba. Projektuojant sudėtingas sistemas dažnai padaroma klaidų. Pradiniuose projektavimo etapuose padarytos klaidos gali būti pastebėtos tik galutinai realizavus sistemą, o tuomet ištaisyti klaidą yra labai brangu arba neracionalu. Šiuo metu sudėtingų sistemų projektavimui keliami dideli saugumo ir našumo reikalavimai, todėl vis plačiau taikomi formalieji specifikavimo metodai [3].

Agentas yra programinis objektas, turintis apibrėžtą tikslą. Agentas apibūdinamas konkrečiomis autonominėmis savybėmis, orientacija į tikslą, reakcija. Agentai retai būna savarankiškos sistemos, dažniausiai jie gyvuoja ir sąveikauja su vieni su kitais. Daugiaagentės sistemos pastaruoju metu labai išpopuliarėjo [4].

Mokslinių tyrimų sritis – genetinių algoritmų ir agregatinio metodo panaudojimas modeliuojant sudėtingas sistemas.

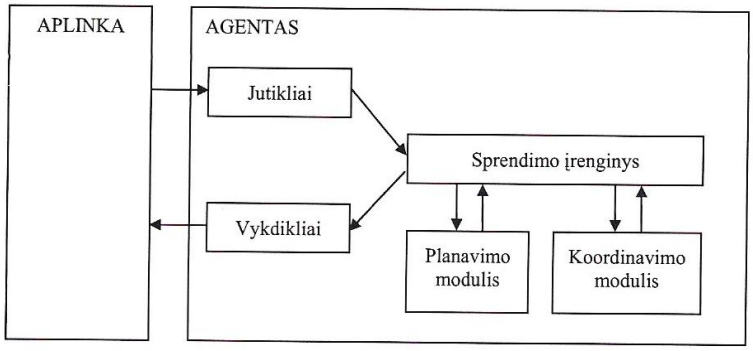
Darbo tikslas – genetinių algoritmų taikymas formaliuose sistemų aprašymo metoduose, sistemų imitaciniame modeliavime, sistemų parametrams nustatyti.

Mokslinis naujumas. Šiame darbe siūlomas naujas genetinio algoritmo ir agregatinio metodo taikymas sistemos modeliams aprašyti. Sistemos modelio parametrai genetiniame algoritme laikomi kaip individo chromosomos, o sistemos modelis tai naudingumo funkcija genetiniame algoritme. Padarytos prielaidos leidžia nustatyti sistemos parametrų optimalias reikšmes, kad sistema efektyviai dirbtų. Kitas siūlomas taikymo atvejis, kai genetinis algoritmas naudojamas perėjimo operatoriuje, nustatyti sekančios populiacijos struktūrai. Tai leidžia imituoti biologines, agentines, savireguliuojančias sistemas.

Darbo struktūra: pirmajame skyriuje išdėstytas agregatinis metodas ir agentai bei agentinės sistemos. Antrajame skyriuje aptariama agentų elgsenos aprašymo metodai: neuroniniai tinklai, genetiniai algoritmai kiti metodai. Aprašyta metodų veikimas, privalumai ir trūkumai, taikymai. Trečiajame skyriuje demonstruojami sudaryti modeliai, jų specifikacijos ir rezultatai gauti apjungus genetinį algoritmą ir agregatinį metodą.

1. AGENTINĖS SISTEMOS

Intelektualūs agentai yra apibrėžiami, kaip programinės įrangos komponentai, kurie gali vykdyti specifines užduotis ir valdyti intelekto laipsnį, kuris leidžia bendrauti su aplinka ir atlikti užduočių dalis_automatiškai. Informacijos agentų paskirtis- informacijos paieška paskirstytose sistemose arba tinkluose. Bendradarbiavimo agentų paskirtis - išspręsti sudėtingas problemas naudojant tam tikrus bendravimo ir bendradarbiavimo mechanizmus. Transakcijų agentų paskirtis - vykdyti ir tikrinti transakcijas [4].



1 pav. Klasikinė programinio agento architektūra

Intelektualiųjų agentų savybės gali būti grupuojamos į dvi dideles grupes: vidinės savybės ir išorinės savybės. Vidinės savybės formuoja agento "vidų", tai yra apibrėžia veiksmą agento viduje. Vidinės savybės: galimybė mokytis, reaguojamumas, autonomiškumas ir tikslingumas[2]. Išorinėms savybėms priklauso visos charakteristikos, kurios įtakoja bendravimą kelių agentų (tos savybės kartais vadinamas socialumu).

Reaguojamumas - agentas privalo atitinkamai reaguoti į daromą poveikį arba informaciją iš savo aplinkos.

Iniciatyvumas/tikslingumas - jeigu intelektualusis agentas nereaguoja į aplinkos pasikeitimus, bet imasi iniciatyvos atsižvelgiant į aplinkybes, tai yra vadinama iniciatyviu elgesiu.

Samprotavimas/Mokymas - agentas elgiasi racionaliai (protingai), kada jis dirba arti pasitenkinimo savo tikslais arba vienu iš dalinių tikslų. Be to galimybė išmokti iš ankstesnės patirties ir sėkmingai pritaikyti savo elgesį aplinkoje yra svarbi intelektualiam agentų elgesiui.

[2] angl. goal-orientedness.

Autonomiškumas - Vienas iš svarbiausių skirtumų tarp agentų ir tradicinių programinės įrangos programų yra agentų galimybė siekti savo tikslų autonomiškai, be bendravimo arba komandų iš aplinkos.

Mobilumas - apibrėžia agento galimybę judėti[3] elektroniniuose bendravimo[4] tinkluose.

Bendravimas/Bendradarbiavimas - agentas dažnai reikalauja bendravimo su savo aplinka, kad pilnai atliktų savo užduotis. Kai bendradarbiauja keli agentai, sudėtingos užduotys sprendžiamos greičiau ir geriau.

Charakteris - jeigu agentas atrodo savo vartotojui, kaip virtuali asmenybė. Tada jis turi turėti tam tikras žmogaus charakteristikas, kad pateisintų šią vaidmenį. Agento svarbiausios charakteristikos yra sąžiningumas, pasikliaujamumas, patikimumas [4].

Agentų sistemos klasifikuojamos pagal tris kriterijus: intelektas, mobilumas ir agentų skaičius.

Intelektas

- Paprastas agentas turi ribotą intelekto laipsnį.
- Sudėtingas agentas turi didesnį intelekto laipsnį.

Mobilumas

- Stacionariniai.
- Mobilūs.

Agentų kiekis

- Pavienis[5] agentas keliauja į aplinką, kuri neturi kitų agentų.
- Daugelių agentų[6] sistemos susideda iš kelių agentų, kurie gali bendrauti arba netgi bendradarbiauti vienas su kitu.

Stacionarūs agentai turi galimybę siųsti pranešimus nutolusiems objektams, tačiau negali patekti į kitą aplinką, t.y. kompiuterį. Mobilūs agentai gali laisvai judėti – migruoti – elektroniniu tinklu ir kompiuteriais. Taip pat turi galimybę bendrauti su aplinkos objektais, pvz., informacijos šaltiniais arba kitais agentais. Stacionarūs agentai bendrauti naudodami nutolusių procedūrų

[3] angl. navigate.

[4] angl. communication.

[5] angl. single.

[6] angl. multi-agent.

iššaukimą[7], o mobilūs agentai bendrauja nuotolinio programavimo[8] būdu nuo RPC skiriasi tuo, kad čia nevyksta užklausų ir atsakymų mainai. Vietoj to, klientas iškviečia procedūrą, kuri įvykdoma serveryje. Vietoj užklausų ir atsakymų RP sudaro pranešimai, talpinantys procedūrą ir su ja susijusias duomenų struktūras. Tokie pranešimai atitinka mobilių agentų elgseną. RP pasižymi didesniu lankstumu ir mažiau apkrauna tinklą.

Mobilių agentų privalumai

- Sumažinta tinklo apkrova.
- Sumažintas kliento resursų panaudojimas.
- Asinchroninis veikimas.
- Perkonfigūravimo galimybės.
- Aktyvus funkcionavimas.
- Decentralizuota struktūra.

Mobilių agentų trūkumai tai techninės kliūtys, neleidžiančios pašalinti saugumo problemos:

- Transportavimas/migravimas.
- Efektyvumas.
- Standartai/suderinamumas.
- Ataskaitų ruošimo sistemos.

Daugelio-agentų sistema – tai sistema, kurią sudaro tam tikras kiekis nepriklausomų, savarankiškų agentų. Joje kiekvienas agentas gali bendrauti/bendradarbiauti su kitais agentais, ieškant informacijos arba sprendžiant tam tikras individualias užduotis (problemas) [5].

Daugelio-agentų sistemos privalumai:

- Egzistuojančių agentų integracija į didesnę sistemą.
- Nebūtina kurti naujus agentus, tam kad išspręsti vieną specifinę problemą.
- Bendradarbiaudami agentai tarpusavyje gali užduotį gali atlikti greičiau ir geriau.

Agentų taikymai:

[7] angl. RPC remote procedure call.

[8] angl. RP remote programming.

- Paprasti E-pašto filtrai.
- Mobilūs taikymai.Intelektuali pagalba (asistavimas).
- Sudėtingų kritinių sistemų valdymas (oro erdvės valdymas).
- E-mokymas.

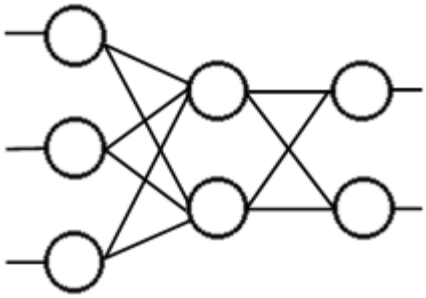
2. AGENTŲ ELGSENOS APRAŠYMO METODAI

Agentų elgsena gali būti aprašoma įvairiais metodais. Šiame skyriuje aprašyta genetiniai algoritmai, jų tipai ir taikymai, taip pat kiti metodai.

2.1 Neuroniniai tinklai

Neuroniniai tinklai buvo sukurti remiantis panašiais veikimo principais kaip ir žmogaus smegenys. Abiejų veikimas yra grindžiamas neuronų, pačių mažiausių sudedamųjų dalių, tarpusavio sąveika. Nors jų funkcijos apsiriboja tik signalų perdavimu, tačiau dideli tinklai sudaryti iš daugybės neuronų gali greitai ir tiksliai apdoroti nemažus kiekius informacijos. Taip pat jie toleruoja klaidas ir gali interpretuoti netikslią informaciją arba prisitaikyti prie naujų situacijų.

Kaip jau buvo minėta neuroninių tinklų pagrindas yra neuronai. Jie yra tarpusavyje sujungti jungtimis, kurių pagalba yra perduodami signalai (2 pav.). Kiekviena jungtis turi svorį, kuris nurodo jungties stipruma ir kaip modifikuojamas perduodamas signalas. Pats neuronas yra sudarytas iš dviejų dalių: sumatoriaus ir aktyvavimo funkcijos. Sumatorius sumuoja visus ateinančius signalus iš kitų neuronų, prieš tai juos padauginęs iš jungčių svorių, o aktyvavimo funkcija transformuoja gautą rezultatą. Jai dažniausiai yra naudojama sigmoidinė funkcija $\sigma(x) = \frac{1}{1+e^{-x}}$, tačiau gali būti pasirinktos ir kitos, pvz.: hiperbolinis tangentas. Gauname, kad kiekvienas neuronas yra apskaičiuojamas pagal tokią formulę: $o = \sigma(\sum_n w_n o_n)$, kur n – prieš tai esančio sluoksnio neuronų skaičius; w – jungties svoris; o – neurono išvedimas [6].



2 pav. Neuroninis tinklas

Dažniausiai neuronai yra grupuojami į sluoksnius ir kiekvienas sluoksnio neuronas yra sujungiamas su gretimų sluoksnių visais neuronais. Pirmasis sluoksnis vadinamas įvedimo, o visi kiti - paslėptaisiais, išskyrus paskutinįjį, kuris vadinamas išvedimo. Kaip galima spręsti iš pavadinimo, į pirmąjį yra perduodami parametrai, kuriuos reikia apdoroti. Tada tinklas yra įvykdomas, ir apskaičiuotas rezultatas gaunamas išvedimo sluoksnyje.

Neuroninių tinklų apmokymas

Tinklas yra mokomas keičiant jungčių tarp neuronų svorius. Šitaip yra imituojamas jungčių sudarymas ir nutraukimas biologinėse smegenyse. Standartinis ir dažniausiai naudojamas tinklams apmokyti „atgalinio sklidimo“ algoritmas[9] yra, kurio metodas remiasi svorių keitimu pagal lokalius nuolydžius. Norint išmokyti tinklą pirmiausia reikia turėti apmokymo aibę, su įvedimo ir rezultatų duomenimis. Yra situacijų kai tokia aibė neegzistuoja (pvz. klasifikavimo uždaviniuose), tada yra taikomi kitokie algoritmai, kurių metu tinklai mokosi atskirti tam tikrus požymius.

Prieš keičiant svorius pirmiausia tinklą reikia įvykdyti ir gautus rezultatus sulyginti su norimais. Pagal juos yra apskaičiuojama klaidos funkcija. „Atgalinio sklidimo“ algoritmas stengiasi minimizuoti šią funkciją apskaičiuodamas dalines išvestines kiekvienam svoriui ir atitinkamai juos pakeisdamas. Deja, toks algoritmas remiasi tik lokaliais nuolydžiais, todėl neretai jis gali įstrigti lokalaus minimumo taškuose, ir tinklas nesugebės pasiekti norimų rezultatų.

Taikymas

Dirbtiniai neuroniniai tinklai taikomi šiose srityse [7]:

[9] angl. back-propagation.

- Grafinių vaizdų atpažinimas;
- Bioinformatikoje – dalinai kintančių, tačiau biologiškai reikšmingų nukleotidų bei aminorūgščių sekų fragmentų paieška. Sistema apmokoma naudojant žinomų sekų rinkinį ir randa naujas, panašiai atrodančias sekas.
- Finansuose - analizuojant, prognozuojant akcijų kitimo kursus;
- Procesams modeliuoti ir valdyti: fizinės sistemos neuroninių tinklų modelis taikomas geriausiems valdymo parametrams nustatyti;
- Mašinų diagnostikai: stebi mašiną ir jai sugedus įspėja sistemą;
- Taikiniui atpažinti: karinėse programose padeda apdoroti paprastą arba infraraudonųjų spindulių vaizdą priešo taikiniui nustatyti.
- Medicininei diagnozei: analizuodami simptomus ir vaizdo duomenis, pavyzdžiui, rentgeno nuotraukas, tinklai padeda gydytojams nustatyti diagnozę [8].

2.2 Genetiniai algoritmai

Genetiniai algoritmai (GA) yra tam tikra evoliucinių algoritmų klasė, naudojanti gamtoje egzistuojančius gyvybės evoliucinius mechanizmus: paveldėjimą, mutaciją, natūraliąją atranką ir rekombinaciją. Genetiniai algoritmai yra metodas analizuoti duomenis, jų dėka galima rasti apytikslį užduoties sprendimą. Sprendimas randamas naudojant evoliucinį ciklą, veikiantį gamtoje. Genetiniai algoritmai tinka ne visur, tačiau jais galima rasti palyginti neblogus sprendinius užduočių, kurių tikslaus sprendimo algoritmai nežinomi. Dažniausiai GA greitai lokalizuoja gerą sprendimą, net ir sudėtinguose uždaviniuose.

Pasirinkus pradinę populiaciją, evoliucija prasideda nuo visiškai atsitiktinių kitimų. Gavus naują populiacijos kartą (kandidatus) įvertinamas jos tinkamumas, atrenkamas tam tikras naujos kartos individų skaičius, pagal atrankos kriterijų. Atrinktieji individai pakeičiami darant mutacijas arba rekombinacijas ir sukuriama nauja populiacija. Vėliau viskas kartojama, atrenkant naujus tinkamiausius individus, sukuriant naują populiaciją. Ciklas kartojamas, kol gaunamas užduotį tenkinantis sprendimą.

- Pasirenkama pradinė populiacija;
- Ciklo pradžia:
- Pagal atrankos kriterijų įvertinamas tam tikros dydžio populiacijos individų tinkamumas;

- Atrenkami geriausi kandidatai, iš kurių palikuonių bus sudaryta naujoji populiacija;
- Atliekama reprodukcija, daromi pakeitimai (mutacijos ir rekombinacijos), gaunama nauja populiacija;
- Ciklas kartojamas, jeigu netenkinama nutraukimo sąlyga.

Genetiniai algoritmai kilo studijuojant ląstelių mechanizmą. Tyrimus vykdė *John Holland* vadovaujama grupė iš JAV Mičigano universiteto. Apie 1985-sius Ilinojaus universitete įvyko Pirmoji tarptautinė konferencija skirta genetiniams algoritmams, iki tol genetinių algoritmų tyrimai buvo daugiausia tik teoriniai. Augant akademiniam susidomėjimui bei vystantis kompiuterijai atsirado galimybė praktiniam genetinių algoritmų įgyvendinimui. 1989 m. JAV laikraštis The New York Times išspausdino rašytojo John Markoff straipsnį apie pirmą komercinę genetinį algoritmą naudojančią programą Evolver. Vėliau atsirado įvairios kompiuterinės programos skirtos įvairioms sritims. Dabartiniu metu programas naudojančias genetinius algoritmus naudoja didžioji dalis Fortune *500* kompanijų, išspręsti tvarkaraščių, duomenų talpinimo, biudžeto paskirstymo ir kitas svarbias užduotis, kur galima pritaikyti kombinatorinį optimizavimą.

Pradžioje daug individualių sprendimų yra atsitiktinai generuojami suformuojant atsitiktinę pradinę populiaciją. Populiacijos dydis pasirenkamas pagal užduoties sudėtingumą. Dažniausiai populiaciją sudaro keletas šimtų ar net tūkstančių galimų sprendimų. Tradiciškai populiacija generuojama atsitiktinai, apimant didelį sprendinių paieškos diapazoną. Kartais nubrėžiamos tam tikros ribos, kuriose sprendinys tikėtinai turėtų būti, tokiu būdu sutrumpinamas paieškos laikas.

2.2.1 Atrankos metodai

Iš populiacijos atrenkama dalis tinkamiausių sprendinių (sėkmingiausių „palikuonių“), iš kurių bus kuriama naujoji populiacija. Atrankos kriterijų aprašo tinkamumo funkcija, kuri ir lemia atranką. Dažniausiai taikomas metodas, kai iš visos populiacijos atrenkami labiausiai tinkami sprendiniai. Tačiau yra ir kitas atrankos metodas, kurio metu yra vertinami tik atsitiktinai populiacijos sprendiniai. Šis, antrasis metodas, yra mažiau efektyvus nei pirmasis, kadangi šiuo metodu tiksliausio sprendimo paieška trunka žymiai ilgiau.

Dauguma atrankos funkcijų yra tikimybinės ir sukurtos taip, kad atrinktų tinkamais ir nedidelę dalį sprendinių, kurie yra mažiau tinkama. Tokiu būdu užtikrinimą, kad išliktų įvairovė

ir išvengiama pirmalaikės konvergencijos bei nukrypimo į netinkamus sprendinius. Dažniausi ir daugiausiai išstudijuoti atrankos metodai: ruletės ir turnyrinė atranka.

Ruletės metodas

Ruletės rato metode kiekviena chromosoma yra renkama tiesiogiai pagal jos tinkamumo reikšmę – kuo didesnis tinkamumas, tuo didesnė tikimybė. Kiekviena ruletės dalis yra proporciška jai atitinkančios chromosomos tinkamumui. Darant selekciją ruletė yra "išsukama" ir pozicija ties kuria ji sustoja atitiks vieną chromosoma. Toks algoritmas galėtų būti realizuotas taip:

1. Susumuoti visų chromosomų tinkamumo reikšmes – S, S = t1 + ... + tn.
2. Sugeneruoti atsitiktinį skaičių – r, iš intervalo (0; S).
3. Pereiti per populiaciją sumuojant tinkamumus – s. Jei sumuojant s > r, grąžinti pridedamą chromosomą.

Deja toks algoritmas turės problemų kai vienos (ar kelių) chromosomų tinkamumo reikšmė sudarys didelę dalį pačios ruletės. Tokiu atveju mažo tinkamumo sprendimai turės labai mažai šansų būti pasirinktais. Šiam reiškiniui ištaisyti yra naudojamas rangų metodas. Jo metu kiekvienam sprendimui yra priskiriamas eilės numeris nuo 1 iki n, pradedant nuo blogiausią tinkamumą turinčių sprendimų ir užbaigiant geriausiu. Tačiau čia geresnės ir blogesnės chromosomos neturi tokio didelio skirtumo kas gali atsiliepti konvergavimo greičiui.

Turnyrinė atranka

Du individai parenkami atsitiktinai. Tėvu tampa vienas iš jų, turintis geresnę tinkamumo funkcijos reikšmę. Tai garantuoja, kad geresni individai tėvais taps dažniau nei kiti. Tačiau yra tikimybė, kad bus prarasti tokie individai kurie turėdami gerą tinkamumo funkcijos reikšmę bus atmetami, kadangi bus nugalėti kito geresnio individo.

Elito atranka

Ignoruojantis įprastą eigą, tačiau labai efektyvus yra variantas, kai konstruojant naują populiaciją, leidžiama keliems sėkmingiausiems organizmams (sprendiniams) pereiti į naują populiacijai visai nepakitus. Ši strategija vadinama elito atranka[10].

Kita labai panaši metodika yra išlaikyti seną populiaciją tiesiog pakeičiant blogesnius tinkamumus turinčius individus naujai sukurtais, tokiu būdu išsaugant geriausius sprendimus.

[10] angl. elitist selection.

2.2.2 Populiacijos atnaujinimas

Antrosios ir kitų populiacijų kūrimą sudaro genetinių mechanizmų: rekombinacijos ir mutacijos – pritaikymas. Rekombinaciją ir mutacija yra biologinių mechanizmų analogai.

Kiekvieno naujo sprendinio sukūrimui iš populiacijos imama pora sprendinių „tėvų“, kurie sukryžminami ir mutacijos ar rekombinacijos būdu gaunamas sprendinys „vaikas“. Toliau imama kita „tėvų“ pora ir vėl sukuriamas „vaikas“ ir taip tęsiama kol pasiekiamas tam tikras naujos kartos individų skaičius, sukuriama pilna nauja sprendinių populiacija.

Galiausiai po šių procesų naująsias kartas sudarys individai, kurių chromosomos (sandara) yra visiškai skirtinga nei pradinės populiacijos. Bendras populiacijos tinkamumo vidurkis taip pat pakils, kadangi išliks tik geriausi individai (sprendiniai) ir dalis mažiau tinkamų individų .

Mutacija

Mutacija yra būdas genetiniuose algoritmuose išlaikyti genetinę įvairovę. Mutacija naudojama GA yra biologinės mutacijos analogas, kadangi jos mechanizmas buvo kuriamas pagal biologinę mutaciją. Mutacijos padeda išvengti populiacijos chromosomų supanašėjimo (lokalaus konvergavimo), o tai vestų į evoliucijos sulėtėjimą ar net visišką stagnaciją. Tai taip pat paaiškina, kodėl GA sistemos vengia naudoti tokį atrinkimo būdą, kai atrenkami vien tik geriausi.

Klasikinį mutacijų operatoriaus veikimą sudaro atsitiktinai generuojami skaičiai, kurie nurodo įvyks ar neįvyks mutaciją tam tikrai duomenų daliai.

Rekombinacija

Rekombinacija yra būdas genetiniuose algoritmuose perteikti vienos chromosomos ar kelių chromosomų sandarą į kitą kartą. Ji yra biologinės rekombinacijos analogas, kadangi būtent biologiniu procesu ir paremta.

Yra nemažai rekombinacijos būdų, kurie skirtingai perteikia biologinės rekombinacijos esmę:

Rekombinacija naudojant 1 tašką

- Su vienu rekombinacijos tašku. Pasirenkamas rekombinacijos taškas tėvų duomenyse. Vėliau visi „tėvų“ duomenys, buvę už taško, apkeičiami vietomis. Rezultatas – „vaikai“, kurie turi dalį vieno ir kito tėvo (genetinių) duomenų.

Rekombinacija naudojant 2 taškus

- Su dviem rekombinacijos taškais. Pasirenkami du rekombinacijos taškai tėvų duomenyse. Vėliau visi „tėvų“ duomenys, esantys tarp šių taškų sukeičiami vietomis. Rezultatas –„vaikai“, kurie turi dalį vieno ir kito tėvo duomenų.

Pjaustymas

- „Pjaustymas“ – rekombinavimo technika, kai keičiamas „vaikų“ duomenų ilgis. Skirtumas nuo ankščiau minėtų technikų yra tas, kad čia tėvų duomenyse pasirenkami taškai skirtingose vietose.
- Vienoda ir pusiau vienoda rekombinacija. Abiejų technikų atvejais „tėvų“ duomenų pagrindu sukuriami du nauji „vaikai“. Pirmuoju atveju „tėvų“ duomenys sulyginami tarpusavyje ir sukeičiami vietomis su 50 proc. tikimybe. Pusiau vienodos rekombinacijos atveju sukeičiama vietomis pusė nesutampančių tėvų duomenų.

Mutacija ar rekombinacija?

Atranka yra pats svarbiausias veiksnys, tačiau yra dvi vyraujančios nuomonės dėl to kas svarbiau mutacija ar rekombinacija. Rekombinaciją palaikantieji teigia, kad ji svarbiausia, o mutacija tik užtikrinanti, kad nebūtų prarastas sprendimo potencialas. Kiti teigia, kad rekombinacija reikalinga tik tam, kad paskleistų naujoves, sukurtas mutacijų. Ir tam kad, nepastoviose populiacijose rekombinacija yra tapati didelei mutacijai (kuri dažniausiai būna katastrofiška).

2.2.3 Proceso nutraukimas

Bendras sprendinių evoliucijos procesas yra ciklišk ai kartojamas kol pasiekiama nutraukimo sąlyga.

Dažniausios nutraukimo sąlygos:

- Sprendinys tenkina minimalų kriterijų;
- Pasiektas nustatytas generacijų skaičius;
- Tam skirtas biudžetas (laikas/pinigai) išnaudotas;
- Pasiekta stabili būsena, kai nėra gaunama geresnių sprendinių;
- Rankinis sustabdymas, atliekama rezultatų patikrinimas;
- Aukščiau minėtų priežasčių kombinacijos.

2.2.4 GA Problemos

Dauguma GA problemų kyla dėl didelio jų sudėtingumo.

Lokalus konvergavimas

GA gali turėti tendenciją konverguoti link lokalaus (riboto) sprendimo, vietoje globalaus (visa apimančio) tinkamiausio sprendimo. Šios problemos tikėtinumas priklauso nuo architektūrinės tinkamumo formos. Tam tikrų problemų sprendimai lengviau krypsta link globalaus sprendinio, kitoms funkcijos lengviau rasti vietinį tinkamiausią sprendinį.

Ją sumažinti ar net visai išspręsti gali skirtingos atrankos funkcijos, arba metodai naudojami išlaikyti kuo įvairiapusiškesnę sprendinių populiaciją.

Ankstyvas konvergavimas

Sunkumų iškyla dirbant su dinaminiais duomenų rinkiniais, kai genomai pradeda anksti konverguoti, tokiu būdu nelieka reikalingų duomenų, iš jų sekančių sprendinių kūrimui.

Šiai problema spręsti variantai:

- galima padidinti genetinį įvairumą, tokiu būdu bus išvengta ankstyvos konvergencijos,
- galima padidinti mutacijos stiprumą, sukeliant vadinamas hipermutacijas (tačiau nukenčia kokybė),
- galima retkarčiais įtraukti visiškai naujus, atsitiktinai generuotus, genų fondo elementus (atsitiktiniai imigrantai).

Genetinių algoritmų klasifikacija

Genetinius algoritmus galima klasifikuoti pagal tai kaip jie atrenka individus tolesniam populiacijos vystymui. Dažniausiai naudojamos anksčiau aprašytos metodikos: ruletės atranka ir turnyrinė atranka. Taip pat kaip modifikacija gali būti naudojama elito atranka. Taip pat galima naudoti skirtingus rekombinacijos metodus.

2.2.5 Dažniausi pritaikymai

Ypatingai tinkamos genetinių algoritmų panaudojimui užduotys yra tvarkaraščių ir grafikų sudarymas, todėl daugybė šios srities programinės įrangos paketų yra paremta genetiniais algoritmais. Genetiniai algoritmai naudojami inžinerijoje bei spręsti globalias optimizavimo problemas.

Genetiniai algoritmai panaudojami ir ten, kur tradiciniai laiptiniai algoritmai[11] gali nerasti sprendinio. Genetiniai algoritmai gali būti naudojami ten, kur užduoties struktūra ganėtinai

[11] angl. hill climbing algorithms

sudėtinga, dėl rekombinacijos jie nuo minimumo gali judėti toliau link teisingo sprendinio, ko nesugeba laiptiniai algoritmai.

2.3 Gimininga metodikos

Genetinis programavimas[12] – naudojamas medžio tipo duomenų struktūrose, vaizduojant kompiuterio programų adaptaciją, vietoje sąrašo ar masyvo, kurį dažniausiai naudoja genetiniai algoritmai. Genetinio programavimo algoritmai dažniausiai reikalauja ilgesnio veikimo laiko, tačiau jų didesnis galingumas. Jie gali būti pritaikomi spręsti tuos uždavinius, kuriuos spręsti sunkiai pavyksta su genetiniais algoritmais.

Sąveikaujantys genetiniai algoritmai[13] – genetiniai algoritmai, kurie naudoja žmogaus įvertinimą. Jie naudojami srityse, kur sunku aprašyti atrankos funkciją. Pavyzdžiui, evoliucionuojantys vaizdai, muzika, kitos meninės formos, kurios priklauso nuo naudotojų estetinio pasirinkimo.

Atkaitinimas[14] (SA) – siejami su globaliais optimizavimo metodais, kurie keliauja paieškos erdve, bandydami įvairias mutacijas ir individualius sprendimus. Priimama ta mutacija kuri padidina veikimo efektyvumą. Mutacija, kuri mažina efektyvumą priimama tikimybiškai priklausomai nuo tinkamumo pasiskirstymo, dažniausiai mažinant temperatūros parametrą. Egzistuoja skirtingi prioritetų vystymo keliai: pagal vieną siekiama suvartoti kuo mažiau energijos, pagal kitą siekiama didžiausio sprendimo tinkamumo. SA gali būti naudojami GA viduje, paprasčiausiai pradedama naudojant didesnį mutacijų dažnį, kuris vėliau pagal grafiką mažinamas.

Tabu paieška [15](TS) – panašūs į SA, abiejuose ieškoma sprendimo keliaujant paieškos erdve ir bandomos įvairios mutacijas bei individualūs sprendimai. Kai tuo tarpu SA generuoja vieną mutavusį sprendimą, tuo tarpu TS generuoja daugybę mutavusių sprendinių, bet ima mažiausia tinkamumą (sveikumą) pademonstravusį sprendinį. Tam kad būtų išvengta cikliškumo užtikrinama didesnė judėjimo laisvė sprendinių erdvėje. Tabu sąrašą sudaro daliniai arba pilni

[12] angl. genetic programming.

[13] angl. interactive genetic algorithms.

[14] angl. simulated annealing.

[15] angl. tabu search.

sprendiniai. Yra draudžiama imti sprendinį iš tabu sąrašo, kuris atnaujinamas vykstant sprendinio paieškai.

Skruzdėlių kolonijos optimizavimas [16]naudoja daug skruzdėlių (agentų), kurios keliauja sprendimų erdvėje ir ieško produktyviausių vietų. Skruzdėlių kolonijos optimizavimas gali būti naudojamas spręsti uždaviniams, kurie nėra globalūs ar neturi naujausios informacijos, kurios reikia kitiems metodams, todėl gali būti pritaikytas ten kur kiti negali veikti.

Memetitinis algoritmas[17] (MA) – terminas, kurį naudoja mokslininkai įvardindami gentinių algoritmus, kurie yra kombinuoti su kitomis lokalių paieškų formomis, tokiomis kaip SA. Kai kurie mokslininkai juos įvardija kaip genetinių algoritmų ir paralelinių genetinių algoritmų hibridus. MA yra efektyvesni už genetinius algoritmus ieškant sprendimo kai kuriose srityse.

3. GENETINIŲ ALGORITMŲ TAIKYMAS SISTEMŲ IMITACINIAME MODELIAVIME

3.1 Agregatinio metodo apžvalga

Agregatinis metodas skirtas sistemoms aprašyti formalia kalba. Taip aprašytas sistemas galima nagrinėti, verifikuoti, patikrinti visas galimas būsenas, įvykių eiliškumą ir pan. [3]. Aprašant sistemą agregatiniu metodu, sistema laikoma kaip tarpusavyje sąveikaujančių PLA [18]agregatų junginys. PLA laikomas kaip objektas, nusakomas būsenų aibe Z, įėjimo signalų aibe X ir išėjimo signalų aibe Y. Agregatas funkcionuoja laiko momentų aibėje $t \in T$. Būsena $z \in Z$, įėjimo signalai $x \in X$ ir išėjimo signalai $y \in Y$ laikomi laiko funkcijomis. Taip pat dar yra perėjimo *H* ir *G* operatorių aibės [9].

$X = \bigcup_{i=1}^{N} X_i$, įėjimų aibę X sudaro N poaibių, kai yra apjungta N agregatų;

$Y = \bigcup_{l=1}^{M} X_l$, išėjimų aibę Y sudaro M poaibių, kai yra apjungta M agregatų;

[16] angl. ant colony optimization.

[17] angl. memetic algorithm.

[18] PLA – piece-linear aggregate (atkarpomis tiesiniai agregatai)

Kiekvienas išėjimas taip pat turi turėti išėjimo operatorių, o kiekvienas įėjimas turi išorinį įvykį. Kiekvienas vidinis įvykis turi savo valdančią seką, kuri nusako laiko momentą kada tas įvykis įvyks [9].

Atkarpomis tiesinio agregato būsena $z \in Z$ yra tas pats kas atkarpomis tiesinio Markovo proceso būsena:

$$z(t) = (\upsilon(t), z_{\upsilon}(t)),$$

čia: $\upsilon(t)$ yra diskrečioji būsenos komponentė, o $z_{\upsilon}(t)$ yra tolydžioji būsenos dedamoji kurią sudaro $z_{\upsilon 1}(t), z_{\upsilon 2}(t),..., z_{\upsilon k}(t)$ koordinatės [9].

Kai modeliuojamoje sistemoje tolydinės koordinatės $z_{\upsilon i}(t)$ aprašo tik laiko kitimą, tai

$\frac{dz_{\upsilon 1}(t)}{dt} = -1$, arba lygu 0.

Agregato būsena gali keistis tik dviem atvejais: atėjus įėjimo signalui arba tolydinė komponentė įgyja begalinę reikšmę.

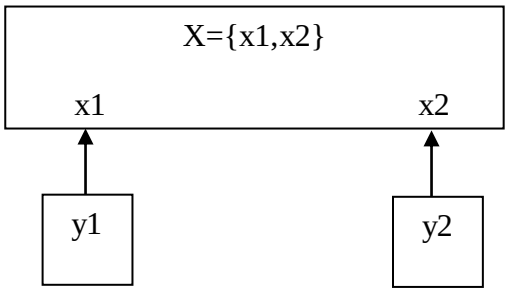
Kaip pavyzdį imkime vieno kanalo aptarnavimo sistemą kaip atkarpomis tiesinį agregatą.

Agregato būsena $z(t) = (\upsilon(t), z_{\upsilon 1}(t), z_{\upsilon 2}(t))$, kur

$\upsilon(t)$ užklausų kiekis sistemoje, laiko momentu *t*, o $z_{\upsilon 1}(t)$ - laiko momentas, kada nauja užklausa ateis į sistemą.

$$z_{\upsilon 2}(t) = \begin{cases} neapibrėeta, jei\ \upsilon(t) = 0; \\ > 0, jei\ \upsilon(t) \neq 0; \end{cases}$$

$z_{\upsilon 2}(t)$ - laikas kada baigsis aptarnavimas.



3 pav. agregato schema

Turime agregatinį modelį sudarytą iš trijų sąveikaujančių agregatų, du agregatai veikia trečiąjį, todėl pastarasis turi du įėjimus(tai reiškia turi du išorinius įvykius). Kitu du agregatai turi tik po vieną išėjimą, tai reiškia jog taip pat turi ir po vieną vidinį įvykį, bei valdančią seką.

y1 agregatas:

1. Išorinių įėjimų aibė: $X = \varnothing$;
2. Išorinių išėjimų aibė: $Y = \{y_1\}$;
3. Išorinių įvykių aibė: $E' = \varnothing$;
4. Vidinių įvykių aibė: $E'' = \{e_1''\}$;
5. Valdančios sekos: $e_1'' \mapsto \xi_0^1, \xi_1^1, \xi_2^1, \ldots$;
6. Diskretinė būsenos dedamoji: $\upsilon(t) = \varnothing$;
7. Tolydinė būsenos dedamoji: $Z_\upsilon(t) = \{\omega(e_1'', t)\}$;
8. Pradinė būsena: $\omega(e_1'', t) = t_0 + \xi_0^1$;
9. Perėjimo operatorius: $H(e_1'')$:

   $\omega(e_1'', t_m) = t_m + \xi_m^1$;

   Išėjimo operatorius: $G(e_1'')$:

   $Y = \{y_1\}$;

y2 agregatas:

1. Išorinių įėjimų aibė: $X = \varnothing$;
2. Išorinių išėjimų aibė: $Y = \{y_2\}$;
3. Išorinių įvykių aibė: $E' = \varnothing$;
4. Vidinių įvykių aibė: $E'' = \{e_2''\}$;
5. Valdančios sekos: $e_2'' \mapsto \xi_0^2, \xi_1^2, \xi_2^2, \ldots$;
6. Diskretinė būsenos dedamoji: $\upsilon(t) = \varnothing$;
7. Tolydinė būsenos dedamoji: $Z_\upsilon(t) = \{\omega(e_2'', t)\}$;
8. Pradinė būsena: $\omega(e_2'', t) = t_0 + \xi_0^2$;
9. Perėjimo operatorius: $H(e_2'')$:

   $\omega(e_2'', t_m) = t_m + \xi_m^2$;

   Išėjimo operatorius: $G(e_2'')$:

   $Y = \{y_2\}$;

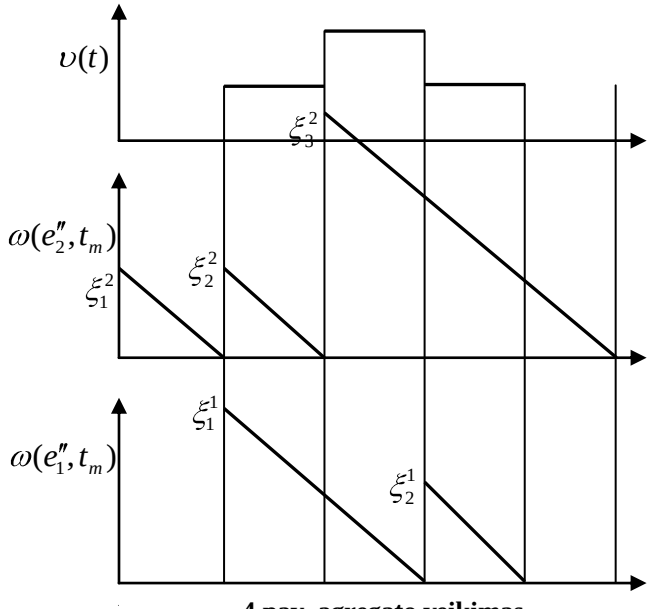
X agregatas:

1. Išorinių įėjimų aibė: $X = \{x_1, x_2\}$;
2. Išorinių išėjimų aibė: $Y = \varnothing$;
3. Išorinių įvykių aibė: $E' = \{e'_1, e'_2\}$;
4. Vidinių įvykių aibė: $E'' = \varnothing$;
5. Valdančios sekos: $\varnothing$;
6. Diskretinė būsenos dedamoji: $\upsilon(t) = \{\upsilon^*(t)\}$;
7. Tolydinė būsenos dedamoji: $Z_\upsilon(t) = \varnothing$;
8. Pradinė būsena: $\upsilon^*(t) = 0$;
9. Perėjimo operatoriai: $H(e'_1)$:

$$\upsilon(t) = \upsilon(t) - 1;$$

$H(e'_2)$:

$$\upsilon(t) = \upsilon(t) + 1;$$

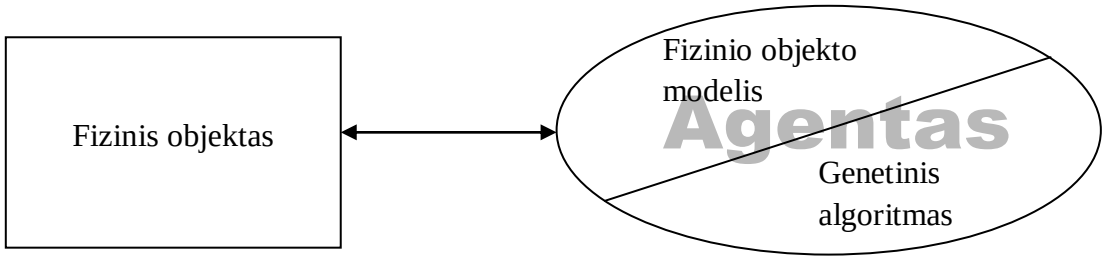


4 pav. agregato veikimas

4 pav. Paveiksle pavaizduotas grafiškai, atkarpomis tiesinio agregato funkcionavimas, įvykus įvykiui e'_1, $\upsilon(t)$ reikšmė sumažėja vienetu, ir atvirkščiai, padidėja kai įvyksta įvykis e'_2. Taip pat matome kad vidiniai įvykiai įvyksta tam tikru laiko momentu pagal valdančių sekų reikšmes.

3.2 Genetinio algoritmo ir agregatinio metodo taikymo pavyzdžiai

Sistemos valdymas yra labai svarbus, kadangi nuo to priklauso sistemos darbas. Realaus laiko sistemos privalo reaguoti į konkrečius įvykius veiksmais, kurių kiekvienas turi būti atliktas tam tikruose laiko apribojimuose [10]. Tokias sistemas ir jas veikiančią aplinką reikia nuolat stebėti ir reikalui esant reaguoti į pokyčius. Tokia veiksena pasižymi agentai.



5 pav. Agento architektūra

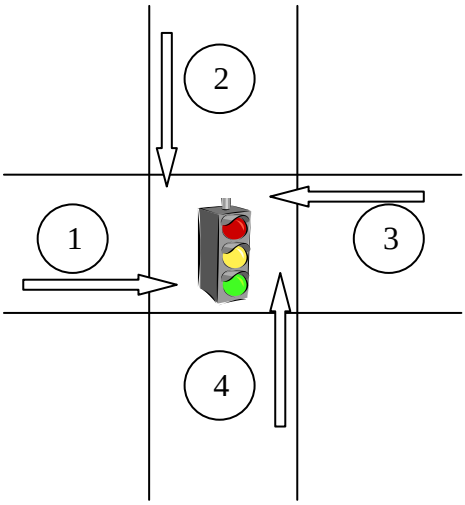
Agentas stebi ir kontroliuoja fizinį objektą. Agento sudaro fizinio objekto modelis, ir genetinis algoritmas. Agentas stebi objektą ir aplinką, pagal tai nustato reikiamus parametrus modeliui ir naudodamas genetinį algoritmą atlieka skaičiavimus – daug bandymų kaitaliojant modelio parametrus. Stebima kaip elgiasi modelis ir atrenkama geriausia parametrų kombinacija. Genetinio algoritmo gauti parametrai nustatomi fiziniam objektui ir toliau stebima ar jo darbo efektyvumas pagerėjo.

3.2.1 Šviesoforo uždavinys

Tokio agento pritaikymo pavyzdys galėtų būti šviesoforo reguliavimas. Šviesoforas – tai fizinis objektas, agentas stebi sankryžą ir šviesoforą. Gautus stebėjimo parametrus – automobilių intensyvumą keliuose agentas nustato turimam sankryžos modeliui. Tuomet paleidžiamas genetinis algoritmas, kuris kaitalioja šviesoforo šviesų degimo laikus ir atlieka daug bandymų su turimu modeliu, taip ieškoma geriausio laikų rinkinio. Šviesoforo laikų rinkinio gerumas matuojamas vidutiniu automobilių laukimo laiku eilėse. Geriausio rasto laikų rinkinio laikai nustatomi šviesoforui. Taip šviesoforas dirbtų optimaliau, automobiliai greičiau pravažiuotų sankryžą, susidarytų mažesnės eilės keliuose. Svarbiausia turėti kuo tikslesnį, labiau atitinkantį tikrovę sankryžos modelį.

Formali sankryžos modelio specifikacija:

Modeliuojama nesudėtinga simetrinė sankryža ir automobiliai važiuoja tik tiesiai.



6 pav. Sankryžos schema

1. Išorinė įėjimų aibė:

$X = \varnothing;$

2. Išorinių išėjimų aibė:

$Y = \varnothing;$

3. Išoriniai įvykiai:

$E' = \varnothing;$

4. Vidiniai įvykiai:

$E'' = \{e''_{11}, e''_{12}, e''_{13}, e''_{14}, e''_{2}, e''_{31}, e''_{32}, e''_{33}, e''_{34}\};$

čia:

e''_{1i} - naujas automobilis atvažiuoja į sankryžą *i*-tuoju keliu;

e''_{2} - šviesoforo šviesos pasikeitimas, čia naudojamas simetriškas šviesoforas taigi kai 1,3 keliuose užsidega žalia šviesa 2,4 keliuose užsidega raudona ir atvirkščiai. Geltona šviesa užsidega visuose keliuose vienu metu;

e''_{3i} - automobilio pravažiavimo laikas per sankryžą kelyje *i*;

5. Valdančios sekos:

$e''_{1i} \mapsto \{\xi^i_0, \xi^i_1, \xi^i_2, ...\}$, čia ξ^i_k - laikas po kurio atvažiuos *k+1*-asis automobilis *i* kelyje, tai atsitiktinis dydis pasiskirstęs pagal eksponentinį dėsnį ;

$e''_{2} \mapsto \{\eta^0, \eta^1, \eta^2\};$

čia:

$\eta^0 = \{\eta_0^0, \eta_1^0, \eta_2^0, ...\}$ Žalios šviesos užsidegimo momentas;

$\eta^1 = \{\eta_0^1, \eta_1^1, \eta_2^1, ...\}$ Geltonos šviesos užsidegimo momentas;

$\eta^2 = \{\eta_0^2, \eta_1^2, \eta_2^2, ...\}$ Raudonos šviesos užsidegimo momentas;

$e''_{3i} \mapsto \{\zeta_0^i, \zeta_1^i, \zeta_2^i, ...\}$;

čia ζ_k^i - yra *k*-tojo automobilio pravažiavimo per sankryžą laikas, kuris priklauso nuo to kelintas eilėje jis stovėjo užsidegus žaliai šviesoforo šviesai;

6. Būsenos diskrečioji dedamoji:

$\upsilon(t) = \{Q_1(t), Q_2(t)Q_3(t), Q_4(t), \chi_0(t), \chi_1(t)\}$;

čia: $Q_k(t)$ - tai eilė *k*-tajame kelyje laiko momentu *t*, *k*=1,2,3,4;

$$\mathrm{X}_0(t) = \mathrm{X}_1(t) = \begin{cases} 0, \textit{geltona}, \\ 1, \textit{žalia}, \\ 2, \textit{geltona}, \\ 3, \textit{raudona}, \end{cases}$$

Nusako degančią šviesoforo šviesą $\chi_0(t)$ - 2-ame ir 4-ame keliuose, o $\chi_1(t)$ - 1-ame ir 3-ame keliuose;

7. Būsenos tolydžioji dedamoji:

$Z_\upsilon(t) = \{\omega(e''_{11}, t), \omega(e''_{12}, t), \omega(e''_{13}, t), \omega(e''_{14}, t), \omega(e''_{2}, t), \omega(e''_{31}, t), \omega(e''_{32}, t), \omega(e''_{33}, t), \omega(e''_{34}, t)\}$;

8. Pradinė būsena laiko momentu t_0:

$\chi_0(t_0) = 0$, $\chi_1(t_0) = 0$;Dega geltona šviesoforo šviesa;

$\#Q_1(t_0) = 0$, $\#Q_2(t_0) = 0$, $\#Q_3(t_0) = 0$, $\#Q_4(t_0) = 0$;

Visuose keliuose eilės yra tuščios;

$\omega(e''_{11}, t_0) = t_0 + \xi_0^1, \omega(e''_{12}, t_0) = t_0 + \xi_0^2, \omega(e''_{13}, t_0) = t_0 + \xi_0^3, \omega(e''_{14}, t_0) = t_0 + \xi_0^4$;

$\omega(e''_{2}, t_0) = t_0 + \eta_0^1$;

$\omega(e''_{31}, t_0) = t_0 + \zeta_0^1, \omega(e''_{32}, t_0) = t_0 + \zeta_0^2, \omega(e''_{33}, t_0) = t_0 + \zeta_0^3, \omega(e''_{34}, t_0) = t_0 + \zeta_0^4$;

9. Perėjimų operatoriai:

$H(e''_{1i})$:

$$Q_i = \begin{cases} enq(Q_i(t_{m-1}), t_m), \#Q_i(t_{m-1}) > 0 \vee \neg \chi_{i \bmod 2}(t_{m-1}) = 1; \\ 0, kitu\ atveju; \end{cases}$$

Atvažiavus naujam automobiliui prie sankryžos, jai eilė netuščia arba dega ne žalia šviesa, tai naujas automobilis statomas į eilę, kitų atveju pravažiuoja per sankryžą, o eilė pažymima tuščia.

$\omega(e''_{1i}, t_m) = t_m + \xi^i_m$;

Naujas automobilis atvažiuos po atsitiktinio laiko tarpo ξ^i_m;

$$\omega(e''_{3i}, t_m) = \begin{cases} t_m + \zeta^i_m, \omega(e''_{3i}, t_{m-1}) = \infty \wedge \chi_{i \bmod 2}(t_m) = 1; \\ \omega(e''_{3i}, t_{m-1}), kitu\ atveju; \end{cases}$$

Sekantis automobilis pravažiuos per sankryža po laiko tarpo ζ^i_m, jei laiko momentu t_{m-1} automobilis nepravažiavo per sankryžą ir dega žalia šviesoforo šviesa;

$H(e''_2)$:

$\chi_0(t_m) = (\chi_0(t_{m-1}) + 1) \bmod 4$;

Perjungiama sekanti šviesoforo šviesa 2-ame ir 4-ame keliuose;

$\chi_1(t_m) = (\chi_1(t_{m-1}) + 1) \bmod 4$;

Perjungiama sekanti šviesoforo šviesa 1-ame ir 3-ame keliuose;

$\omega(e''_2, t_m) = t_m + \eta_m^{\chi_0(t_{m-1}) \bmod 3 + \left[\frac{\chi_0(t_{m-1})}{3}\right]}$;

Nustatomas sekantis šviesoforo šviesos perjungimo momentas;

$$\omega(e''_{3i}, t_m) = \begin{cases} t_m + \zeta^i_m + \Delta(t_{m-1}), \#Q_i(t_{m-1}) > 0 \wedge \chi_{i \bmod 2}(t_m) = 1; \\ \omega(e''_{3i}, t_{m-1}), kitu\ atveju; \end{cases}$$

Sekantis automobilis pravažiuos per sankryža po laiko tarpo $\zeta^i_m + \Delta(t_{m-1})$, jei laiko momentu t_{m-1} eilė buvo netuščia ir dega žalia šviesoforo šviesa;

$$Q_i(t_m) = \begin{cases} deq(Q_i(t_{m-1})), \#Q_i(t_{m-1}) > 0 \wedge \chi_{i \bmod 2}(t_m) = 1; \\ Q_i(t_{m-1}), kitu\ atveju; \end{cases}$$

Jei eilė netuščia ir dega žalia šviesa, tai išimamas vienas automobilis iš eilės, kitu atveju eilė nesikeičia;

$$n_i(t_m) = \begin{cases} 1, \#Q_i(t_{m-1}) > 0 \wedge \chi_{i \bmod 2}(t_m) = 1; \\ 0, kitu\ atveju; \end{cases}$$

Jei eilė netuščia ir dega žalia šviesa, tai automobilio numeris eilėje yra 1, kitu atveju 0;

$H(e''_{3i})$:

$$Q_i(t_m) = \begin{cases} deq(Q_i(t_{m-1})), \#Q_i(t_{m-1}) > 0; \\ Q_i(t_{m-1}), kitu\ atveju; \end{cases}$$

Automobiliui pravažiavus per sankryžą, jei eilė netuščia, tai išimamas vienas automobilis iš eilės, kitu atveju eilė nesikeičia;

$$n_i(t_m) = \begin{cases} n_i(t_{m-1}) + 1, \#Q_i(t_{m-1}) > 0; \\ 0, kitu\ atveju; \end{cases}$$

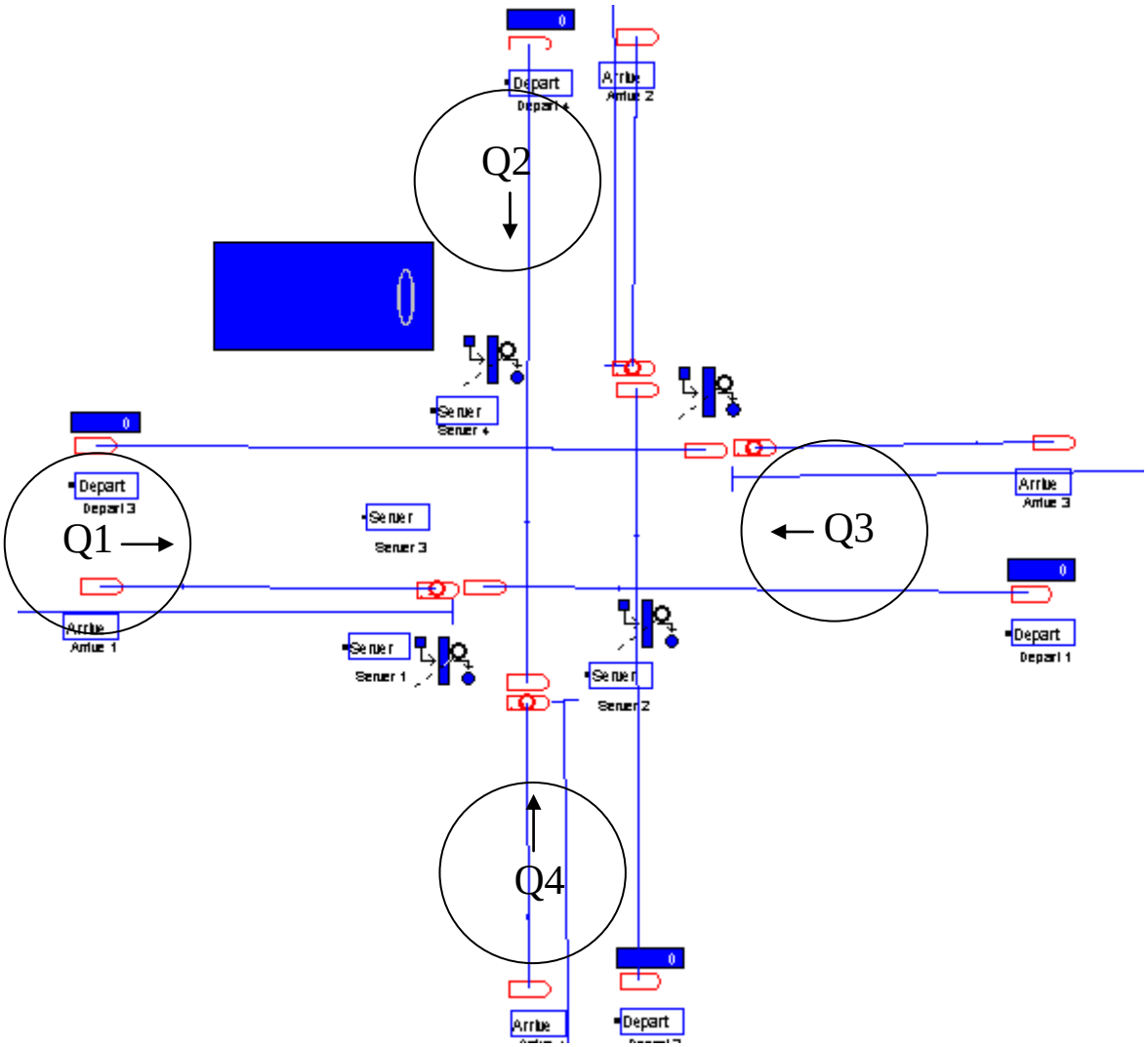
Jei eilė netuščia, tai automobilio numeris eilėje padidinamas 1, kitu atveju nustatoma 0;

$$\omega(e''_3, t_m) = \begin{cases} t_m + \zeta_m^i + \Delta(t_{m-1}), \#Q_i(t_{m-1}) > 0; \\ \omega(e''_{3i}, t_{m-1}), kitu\ atveju; \end{cases}$$

Sekantis automobilis pravažiuos per sankryža po laiko tarpo $\zeta_m^i + \Delta(t_{m-1})$, jei laiko momentu t_{m-1} eilė buvo netuščia;

čia $\Delta(k)$ - laiko tarpas, kiek užtrunka *k*-tasis automobilis eilėje privažiuoti iki sankryžos.

Pagal agregatinę specifikaciją buvo sudarytas sankryžos valdomos šviesoforu modelis, naudojant modeliavimo įrankį „Arena 3.0“.



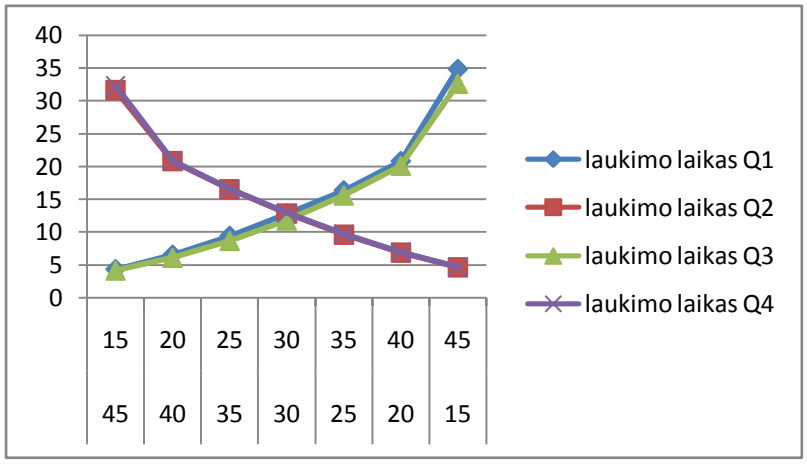
7 pav. Sankryžos – šviesoforo modelis sudarytas naudojant modeliavimo įrankį Arena

Naudojant sudarytą modelį buvo atlikta bandymų nustatyti ar galima taikyti genetinį algoritmą ieškant geriausių šviesoforo parametrų. Q1..Q4 yra sankryžoje susikertantys keliai, ir automobilių eilė. Kiekvieno bandymo modeliavimo laikas 10000 laiko vienetų. Visuose bandymuose geltonos spalvos degimo laikas 3 laiko vienetai. 1,3 eilių žalios šviesos degimo laikas – α 2,4 eilių žalios šviesos degimo laikas – β.

Šiame bandyme naujo automobilio atvažiavimo dažnis yra ρ1.. ρ4=exp(5). Šiame bandyme bandoma pademonstruoti parametrų paieška naudojant pilną perrinkimą, tam kad gautą rezultatą būtų galima palyginti su rezultatu gautu naudojant genetinį algoritmą.

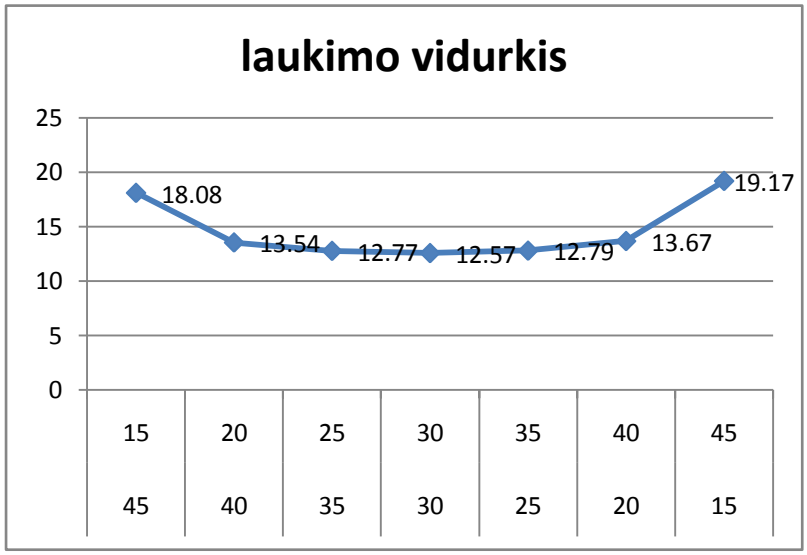
Lentelė Nr.1 Pirmojo etapo bandymų duomenys

α	45	40	35	30	25	20	15
β	15	20	25	30	35	40	45
laukimo laikas Q1	4.3132	6.5176	9.4059	12.81	16.339	20.809	34.848
laukimo laikas Q2	31.584	20.743	16.504	12.81	9.5733	6.8684	4.6168
laukimo laikas Q3	4.1268	6.0641	8.6382	11.748	15.563	20.087	32.572
laukimo laikas Q4	32.279	20.851	16.541	12.894	9.6673	6.9078	4.6358
vidurkis	18.08	13.54	12.77	12.57	12.79	13.67	19.17



8 pav. Laukimo laiko priklausomybė nuo šviesos degimo laiko pilno perrinkimo etapas 1

α ir β reikšmės buvo keičiamos (didinamos ir mažinamos) naudojant žingsnelį 5. Kadangi automobilių srautai visuose keliuose vienodi, todėl geriausias rezultatas gautas kreivių susikirtimo taške kai α=β.



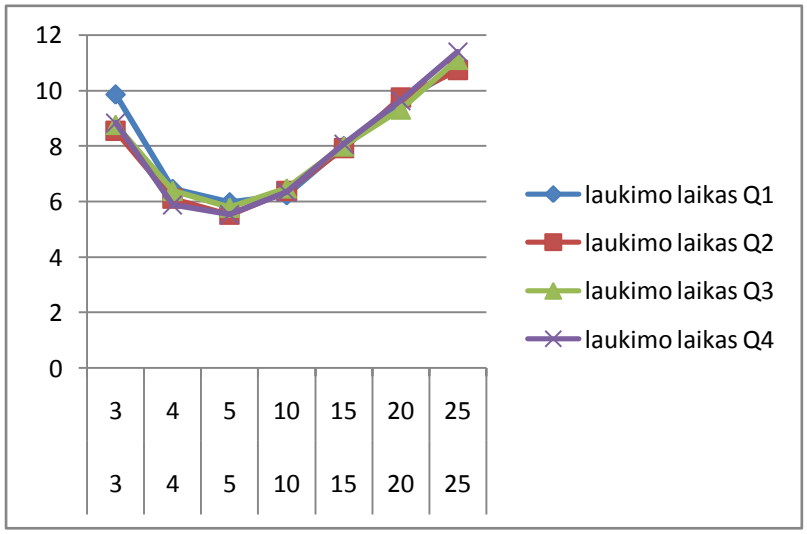
9 pav. Laukimo laiko vidurkio kitimas pilno perrinkimo etapas 1

Geriausias rezultatas gautas kai α=β=30 laukimo laiko vidurkis 12.57. Kadangi šiuo bandymu norime rasti geriausią rezultatą, reikia atlikti dar vieną bandymą išbandyti daugiau variantų kai α=β.

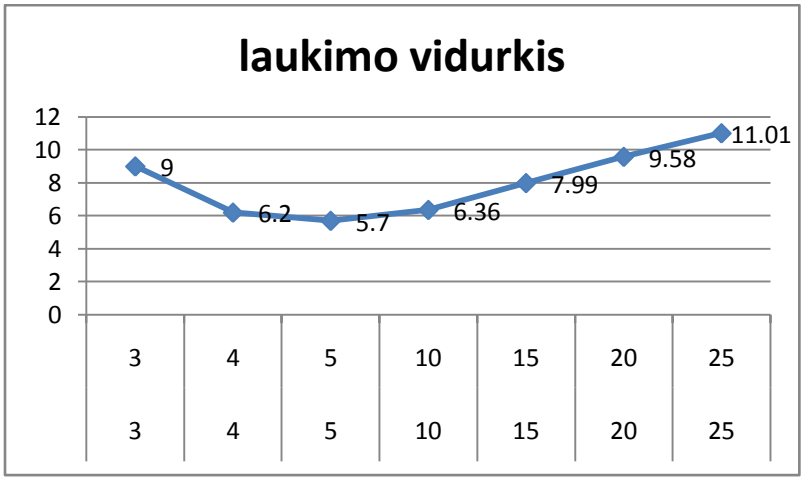
Antrasis pilnojo perrinkimo etapas

Lentelė Nr.2 Antrojo etapo bandymo duomenys

α	3	4	5	10	15	20	25
β	3	4	5	10	15	20	25
laukimo laikas Q1	9.853	6.4528	5.9753	6.2334	7.9931	9.6017	10.8
laukimo laikas Q2	8.5349	6.097	5.52	6.3762	7.918	9.7376	10.731
laukimo laikas Q3	8.7664	6.3835	5.7796	6.4752	7.9872	9.319	11.114
laukimo laikas Q4	8.8356	5.8803	5.5355	6.344	8.0655	9.6434	11.384
vidurkis	9	6.2	5.7	6.36	7.99	9.58	11.01



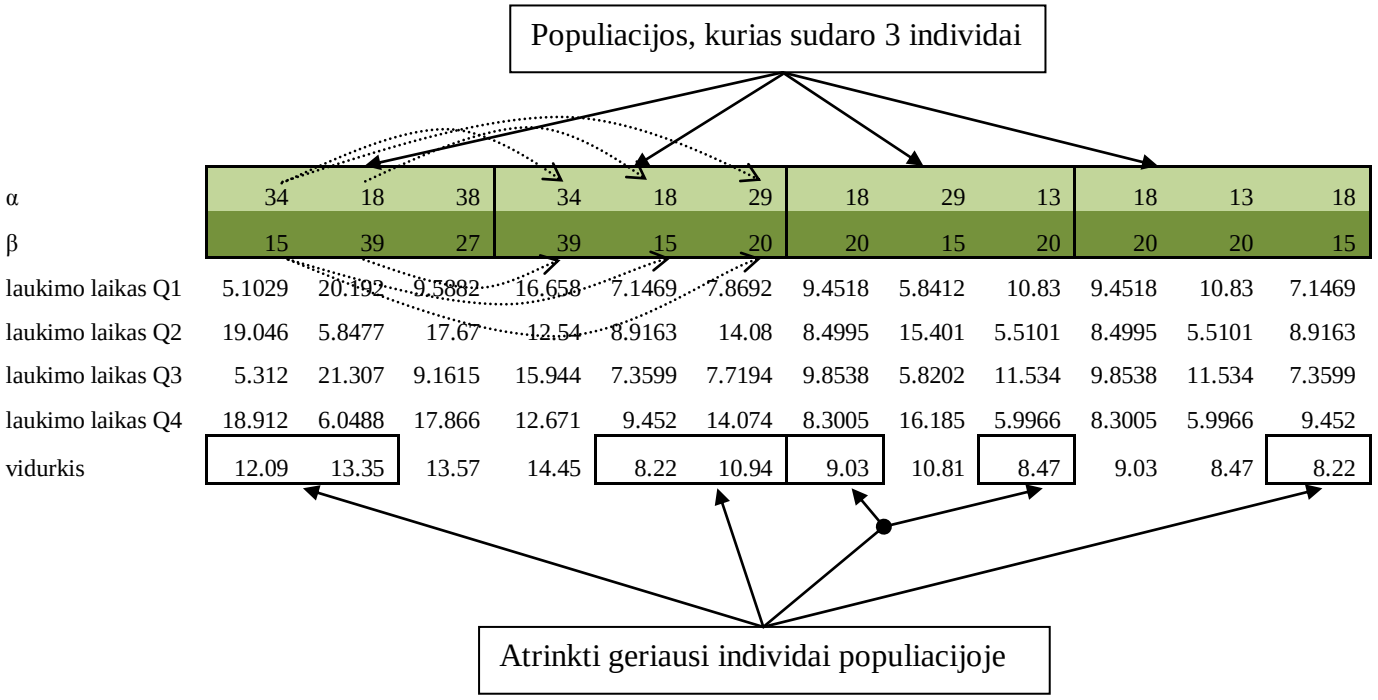
10 pav. Laukimo laiko priklausomybė nuo šviesos degimo laiko pilno perrinkimo etapas 2



11 pav. Laukimo laiko vidurkio kitimas pilno perrinkimo etapas 2

Pilno perrinkimo metodu gautas geriausias rezultatas kai α=β=5, tuomet vidutinis laukimo laikas yra tik 5.7 sekundės.

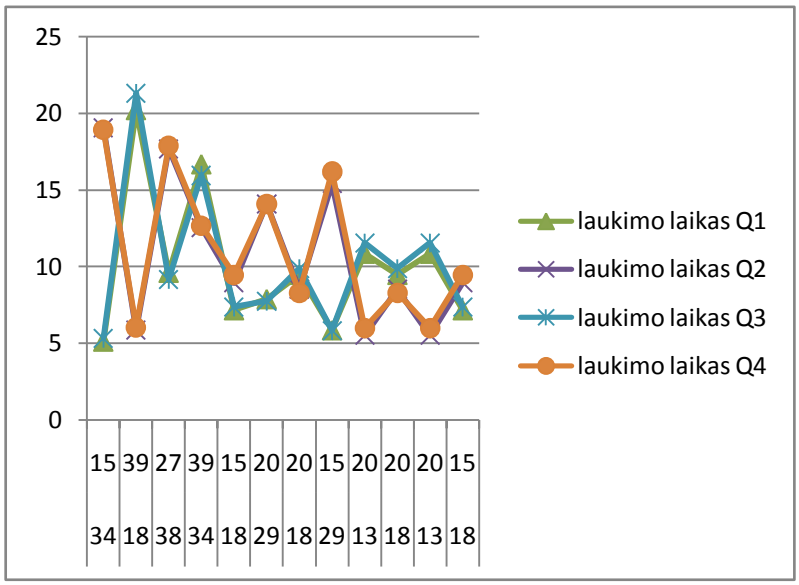
Dabar demonstruojamas genetinio algoritmo veikimo principu atliktas bandymas:



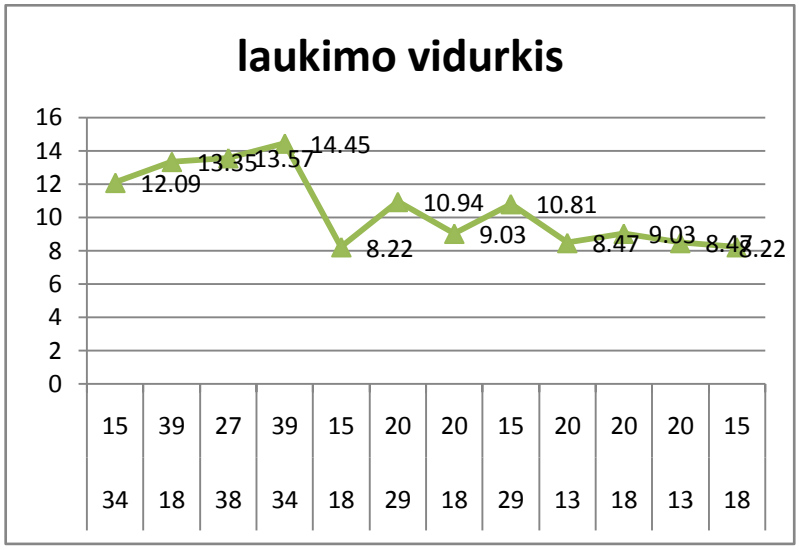
12 pav. Genetinio algoritmo veikimo demonstracija

Pirmajame bandymo žingsnyje atsitiktinai sugeneruojama pirmoji populiacija kurią sudaro 3 individai. Kiekvieną individą sudaro 2 skaičiai – 2 chromosomos, pvz. pirmasis

individas yra α=34 β=15, antrasis individas α=18 β=39 ir t.t. Panaudojus turimą modelį, įvertinama gerumo funkcija – automobiliu laukimo laiko vidurkis. Pagal gerumo funkciją atrenkami 2 geriausia individai populiacijoje, kurių gerumo funkcijos reikšmė mažiausia. Taigi iš pirmosios populiacijos atrenkami pirmas ir antras nariai (apibrėžtos laukimo laiko vidurkio reikšmės). Iš atrinktų geriausių narių sudaroma nauja populiacija. Sudarant naujus narius pritaikyta rekombinacija ir mutacija. Pvz. antrosios populiacijos pirmas narys 34 39 sudarytas iš abiejų geriausiu pirmosios populiacijos individu. 34 yra iš pirmojo individo, o 39 iš antrojo. Taip pat sudarytas ir antrasis antrosios populiacijos narys. Trečiasis narys 29 20 gautas mutavus pirmajam pirmos populiacijos nariui, 29=34-5 ir 20=15+5. Vadovaujantis tais pačiais principais sudarytos ir kitos populiacijos.



13 pav. Laukimo laiko priklausomybė nuo šviesos degimo laiko kai vienodi intensyvumai



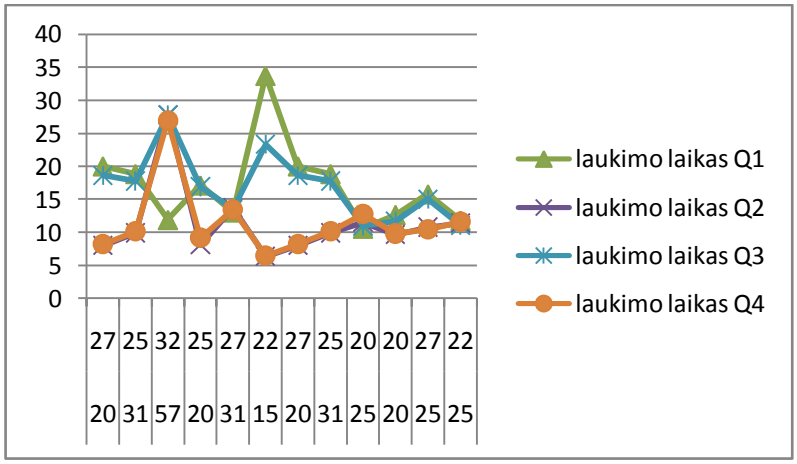
14 pav. Laukimo laiko vidurkio kitimas kai vienodi intensyvumai

Matome, jog pradžioje grafikas smarkiai svyruoja, tačiau tau trečioje iteracijoje pradeda nusistovėti ir laukimo vidurkis mažėja. Atlikus daugiau iteracijų ir naudojant didesnes populiacijas (šiuo atveju populiacijos dydis galėtų būti apie 50 individų) galėtume panaudoti daugiau skirtingų mutacijų, šiuo atvejų tikslingiau naudoti mutaciją kadangi turime tik 2 chromosomas, todėl naudojant rekombinaciją galime niekada nepriartėti prie sprendinio. Todėl geriausia naudoti mišrią strategiją: mutaciją ir rekombinaciją, taip galėtume gauti žymiai geresnį sprendinį, artimą reikšmę gautą pilnojo perrinkimo metodu.

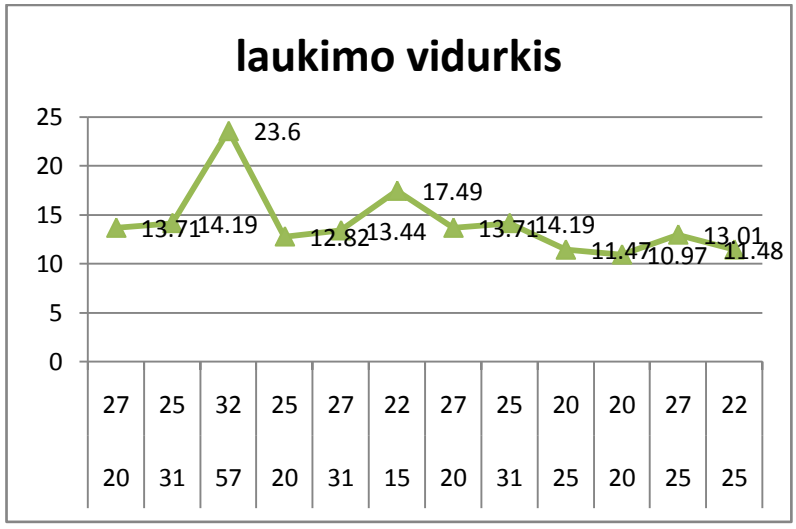
Buvo pademonstruotas paprastas atvejis, kai visuose keliuose vienodas intensyvumas. Sekantis pavyzdys kai su skirtingais eismo intensyvumais keliuose: ρ2, ρ4=exp(5) o ρ1, ρ3=exp(3). Šiame bandyme taip pat kaip ir anksčiau aprašytame populiacijos sudaromos tokiu pat metodu.

Lentelė Nr.3 Genetinio algoritmoveikimo demonstracijos duomenys, kai skirtingi intensyvumai

α	20	31	57	20	31	15	20	31	25	20	25	25
β	27	25	32	25	27	22	27	25	20	20	27	22
laukimo laikas Q1	20.044	18.883	11.922	17.151	13.064	33.759	20.044	18.883	10.605	12.687	15.804	11.817
laukimo laikas Q2	7.9837	9.8922	27.759	8.0576	13.721	6.3575	7.9837	9.8922	11.516	9.6417	10.755	11.389
laukimo laikas Q3	18.618	17.795	27.759	16.911	13.503	23.348	18.618	17.795	10.99	11.708	15.024	11.089
laukimo laikas Q4	8.2078	10.187	26.951	9.1509	13.458	6.4775	8.2078	10.187	12.766	9.8264	10.472	11.616
vidurkis	13.71	14.19	23.6	12.82	13.44	17.49	13.71	14.19	11.47	10.97	13.01	11.48



15 pav. Laukimo laiko priklausomybė nuo šviesos degimo laiko kai skirtingi intensyvumai



16 pav. Laukimo laiko vidurkio kitimas kai skirtingi intensyvumai

Matome jog šiuo atveju kaip ir buvo tikėtas geriausias rezultatas gautas su skirtingais žalios šviesos degimo laikais, tai reiškia, kad šis modelis gali prisitaikyti prie aplinkos.

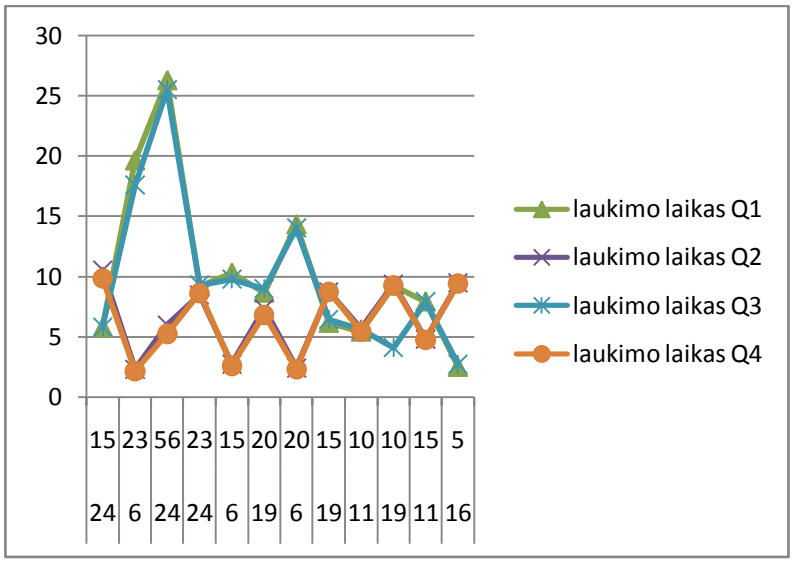
Nors tai tik demonstracija ir buvo sudaryta labai maža populiacija ir atliktos tik 3 iteracijos galima matyti jog genetinis algoritmas tinka spręsti šiam uždaviniui. Matome jog laukimo vidurkio kreivė mažėja, taip artėjama prie geresnio sprendinio. Naudojant didesnį populiacijos kiekį, įvairias mutacijas ir daugiau iteracijų galėtume gauti žymiai geresnį rezultatą.

Toliau panagrinėsime sudėtingesnį modelį, kuriame įvertinama tai jog eilėje stovintys automobiliai ilgiau užtrunka kol pajuda iš vietos ir pravažiuoja sankryžą. Atliekant bandymus su šiuo modeliu, teko sumažinti intensyvumus, kadangi eilės susidarydavo tokios ilgos, jog

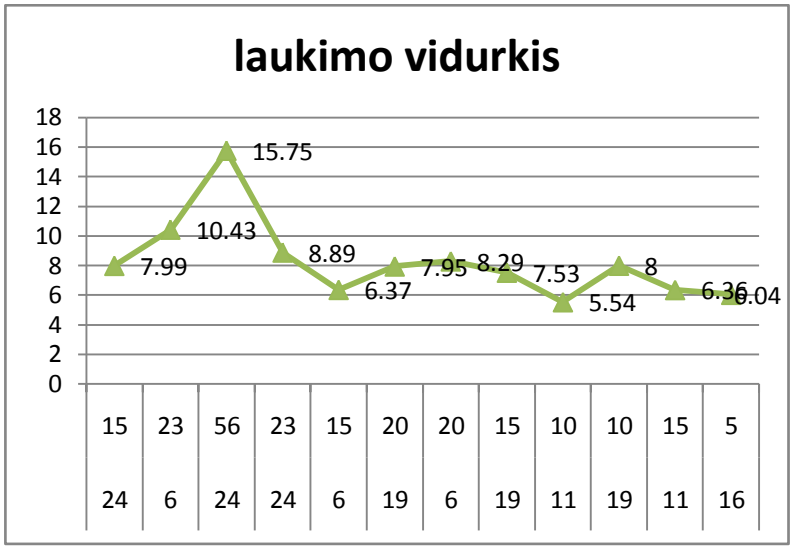
persipildydavo eilės. Taigi modeliuota buvo 10000 laiko vienetų, intensyvumai: ρ2, ρ4=exp(20) o ρ1, ρ3=exp(10).

Lentelė Nr.4 Genetinio algoritmoveikimo demonstracijos duomenys kai intensyvumai 10 ir 20

α	24	6	24	24	6	19	6	19	11	19	11	16
β	15	23	56	23	15	20	20	15	10	10	15	5
laukimo laikas Q1	5.7731	19.635	26.291	9.1784	10.321	8.6621	14.341	6.1641	5.4308	9.261	7.9315	2.5028
laukimo laikas Q2	10.53	2.3203	5.9631	8.4736	2.7579	7.3375	2.4337	8.7505	5.6505	9.3852	4.827	9.4953
laukimo laikas Q3	5.7853	17.593	25.491	9.2813	9.7966	8.9746	14.026	6.4506	5.6114	4.0999	7.9113	2.7158
laukimo laikas Q4	9.8549	2.164	5.2667	8.644	2.615	6.8143	2.3406	8.736	5.4483	9.261	4.7582	9.4474
vidurkis	7.99	10.43	15.75	8.89	6.37	7.95	8.29	7.53	5.54	8	6.36	6.04



17 pav. Laukimo laiko priklausomybė nuo šviesos degimo laiko kai intensyvumai 10 ir 20



18 pav. Laukimo laiko vidurkio kitimas kai intensyvumai 10 ir 20

Matome jog genetinis algoritmas elgiasi labai panašiai atliekant bandymus su šiuo modeliu, nors čia modeliuojama su smarkiai mažesniu automobilių srautu.

3.2.2 Dvikovos uždavinys

Dvikovoje dalyvauja du objektai. Jie iš pradinės pozicijos juda vienas link kito ir šauna. Pataikymo tikimybe priklauso nuo atstumo tarp objektų. Abu objektai turi tik po vieną galimybę šauti. Jai vienas objektas šovė tačiau nepataikė tai antrasis objektas iššaus tik pasiekus pusiaukelę kai atstumas yra 0, o tikimybė pataikyti 1.

Tarkime yra du objektai A ir B. Pradinis atstumas tarp jų yra D. Laiko momentu t objektas A šauna, jo pataikymo tikimybė priklauso nuo laiko p(t). Jai objektas A nepataikė, tai abu objektai juda pirmyn kol susitinka ir tuomet objektas B šauna ir jo pataikymo tikimybė yra 1, kadangi atstumas tarp objektų lygus 0. Dėl paprastumo tarkime jog objektai negali būti abu nušauti t.y. kulka skriejimo laikas yra 0.

Pataikymo tikimybė skaičiuojama pagal formulę:

$p(t) = 1 - \frac{d(t)^2}{D^2}$; čia $d(t)$ - tai atstumas tarp objektų laiko momentu t.

Dvikovos uždavinio formali specifikacija:

1. Išorinė įėjimų aibė:

$X = \emptyset$;

2. Išorinių išėjimų aibė:

$Y = \text{Ø};$

3. Išoriniai įvykiai:

$E' = \text{Ø};$

4. Vidiniai įvykiai:

$E'' = \{e_1'' , e_2'' \};$

čia: e_1'' - dvikovos pradžios momentas;

5. Valdančios sekos:

$e_1'' \mapsto \{\tau_0, \tau_1, \tau_2, ...\};$

6. Būsena:

$\upsilon = \{d_1(t), d_2(t), d_3(t), ...d_N(t)\};$

čia: $d_i(t)$ - i-tojo dvikovininko savybė (iš kokio atstumo jis šauna), N – tai kiekis kiek įvyko dvikovų populiacijoje;

$Z_\upsilon(t) = \{\omega(e_1'' , t), \omega(e_2'' , t)\};$

$n(t)$ - skaičius kiek įvyko dvikovų einamoje populiacijoje;

7. Pradinė būsena laiko momentu t_0:

$d_i(t) = \xi_i,$ čia ξ_i - atsitiktinis dydis iš intervalo [0,1];

8. Perėjimų ir išėjimų operatoriai:

$H(e_1''):$

$\alpha, \beta \in i \{ d_i(t_{m-1}) \neq \infty, 1 \le i \le N \};$

Dvikovoje dalyvauja du objektai α ir β;

$$d_\alpha(t_m) = \begin{cases} \infty, ((\xi \le 1 - d_\alpha^2(t_{m-1})) \wedge (\eta \le 1 - d_\beta^2(t_{m-1})) \wedge (d_\alpha(t_{m-1}) < d_\beta(t_{m-1}))) \vee \\ ((\xi > 1 - d_\alpha^2(t_{m-1})) \wedge (\eta \le d_\alpha^2(t_{m-1}))); \\ d_\alpha(t_{m-1}), \textit{kitu atveju}; \end{cases}$$

Objektas α pralaimi dvikova, jei objektas β iššauna pirmas ir pataiko, arba kai objektas α nepataiko;

$$d_\beta(t_m) = \begin{cases} \infty, ((\eta \le 1 - d_\beta^2(t_{m-1})) \wedge (\xi \le 1 - d_\alpha^2(t_{m-1})) \wedge (d_\beta(t_{m-1}) < d_\alpha(t_{m-1}))) \vee \\ ((\eta > 1 - d_\beta^2(t_{m-1})) \wedge (\xi \le d_\beta^2(t_{m-1}))); \\ d_\beta(t_{m-1}), \textit{kitu atveju}; \end{cases}$$

Objektas β pralaimi dvikova, jei objektas α iššauna pirmas ir pataiko, arba kai objektas β nepataiko;

čia: η, ξ - atsitiktiniai dydžiai iš intervalo [0;1];

$$n(t_m) = \begin{cases} n(t_{m-1}) + 1, d_\alpha(t_m) + d_\beta(t_m) = \infty; \\ n(t_{m-1}), kitu\ atveju; \end{cases}$$

Padidinamas įvykusių dvikovų kiekis;

$\omega(e_1'', t_m) = t_m + \tau_m$;

$$\omega(e_2'', t_m) = \begin{cases} t_{m-1} + \frac{\tau_m}{2}, n(t_m) = \frac{N}{2}; \\ \omega(e_2'', t_m), kitu\ atveju; \end{cases}$$

$H(e_1''):$

$n(t_m) = 0$;

Turime $d_1(t_{m-1}), d_2(t_{m-1}), d_3(t_{m-1}), ...d_N(t_{m-1})$ pažymėkime

$d_{i_1}(t_{m-1}), d_{i_2}(t_{m-1}), d_{i_3}(t_{m-1}), ...d_{i_N}(t_{m-1})$ taip, kad

$d_{i_1}(t_{m-1}) \le d_{i_2}(t_{m-1}) \le d_{i_3}(t_{m-1}) ... \le d_{i_N}(t_{m-1})$

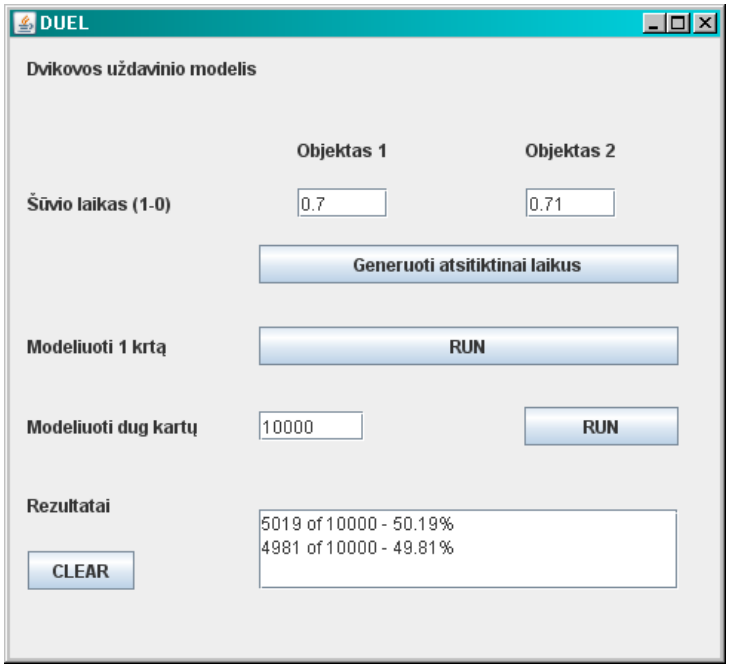
tuomet $d_{1_{\frac{N}{2}+j}}(t_m) = f(d_{i_j}(t_{m-1})), 1 \le j \le \frac{N}{2}$;

Atkuriama populiacija;

čia $f(d_{i_j}(t_{m-1}))$ - funkcija kuri iš turimų individų sukuria naujus individus, tai gali būti rekombinacija ir/arba mutacija.

Šiame modelyje realizuotas mišrus atrankos metodas, apjungiant ruletės ir elito atrankos metodus.

Pagal specifikaciją buvo sudarytas uždavinio modelis, ir atlikti bandymai. Buvo naudojama tokia pat metodika kaip ir su sankryžos modeliu, gauti bandymų rezultatai lyginami ir atrenkami geriausi individai. Šiuo atveju imama laikų pora, ir laikas kuris buvo geresnis (objektas šaudamas tuo laiku laimėjo daugiau kartų) laikomas geresniu. Taip iš pradinės laikų imties išlieka tik pusė individų. Sekančios genetinio algoritmo iteracijos metu iš turimos pusės imtis sudaroma kita pusė laikų. Šiuo atveju naudojama tiktai mutacija, kadangi yra tiktai viena chromosoma kiekvieną individą nusako vienas skaičius.

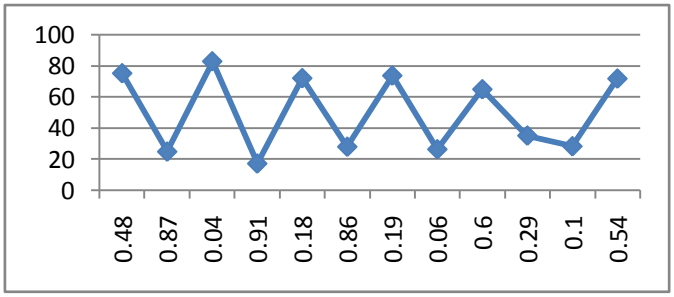


19 pav. Dvikovos modelio programos vartotojo sąsajos vaizdas

Pirmasis šūvio laikų rinkinys sugeneruotas atsitiktinai

Lentelė Nr.5 Dvikovos modelio bandymų duomenys, iteracija 1

šūvio laikas	0.48	0.87	0.04	0.91	0.18	0.86	0.19	0.06	0.6	0.29	0.1	0.54
pergalės procentas	75.1	24.9	82.73	17.27	71.95	28.05	73.58	26.42	64.84	35.16	28.33	71.67

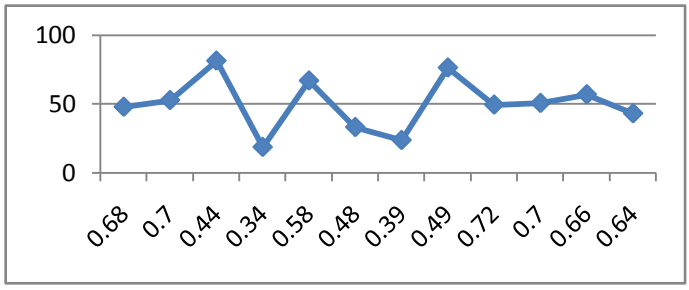


20 pav. Pataikymo procentų pasiskirstymas

Iš atrinktų individų (paryškinti laikai) sudaroma nauja populiacija naudojant mutaciją ir elito atrankos metodą. Taigi penktoje iteracijoje (visų iteracijų rezultatus galima rasti pirmame priede) gauname:

Lentelė Nr.6 Dvikovos modelio bandymų duomenys, iteracija 5

šūvio laikas	0.68	0.7	0.44	0.34	0.58	0.48	0.39	0.49	0.72	0.7	0.66	0.64
pergalės procentas	47.83	52.7	81.41	18.59	66.93	33.07	23.63	76.37	49.41	50.59	56.91	43.09

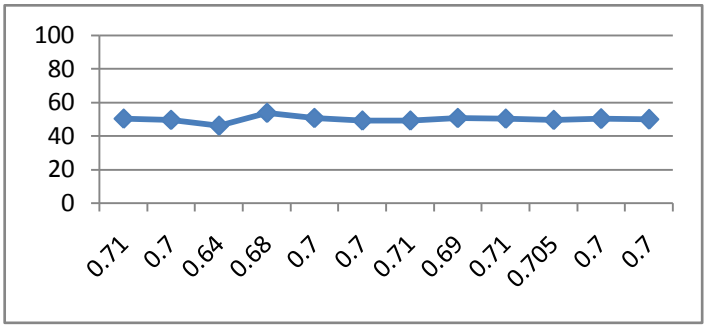


21 pav. Pataikymo procentų pasiskirstymas

Matome jog pataikymų procentai mažiau išsibarstę ir artėja link 50%. Tęsiant toliau devintoje iteracijoje gauname:

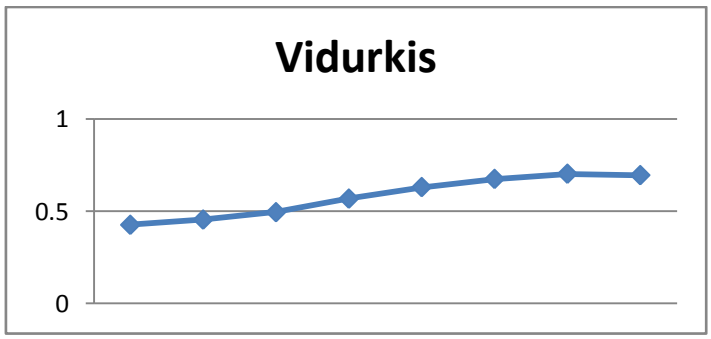
Lentelė Nr.7 Dvikovos modelio bandymų duomenys, iteracija 9

šūvio laikas	0.71	0.7	0.64	0.68	0.7	0.7	0.71	0.69	0.71	0.705	0.7	0.7
pergalės procentas	50.34	49.66	46.16	53.84	50.76	49.24	49.35	50.65	50.32	49.68	50.19	49.81



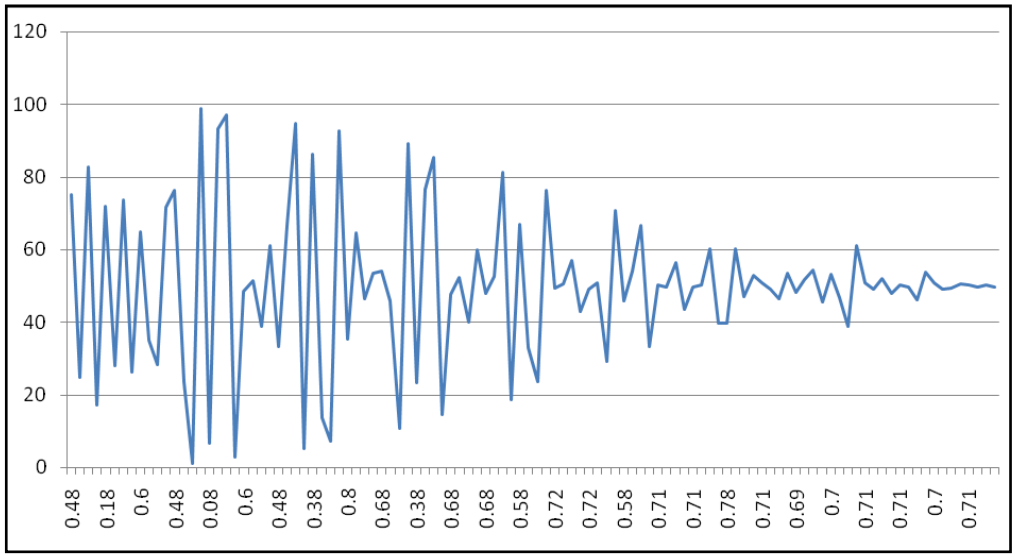
22 pav. Pataikymo procentų pasiskirstymas

Devintoje iteracijoje, pataikymo procentai visiškai nusistovėjo ties 50%, tai reiškia kad radome optimalų šūvio atstumą.



23 pav. Šūvio laiko vidurkio kitimas

Šūvio atstumo vidurkis apsistojo apytiksliai ties 0.7.



24 pav. Bendra pataikymo procentų pasiskirstymo kreivė

Ši kreivė parodo pataikymo procentų istoriją kaip kito visose iteracijose.

3.2.3 Viršvalandžių uždavinys

Uždavinio tikslas yra rasti maksimalų pelną, kurį galima gauti atlikus tam tikrą darbą per fiksuotą laiko tarpą. Už atliktą darbą gaunamas pelnas, jai vėluojama atlikti darbus skaičiuojama bauda. Iš gauto pelno dar reikia atskaičiuoti darbo užmokestį darbuotojams. Jai darbo yra daug, o skirto laiko nepakanka, tai darbuotojai gali dirbti viršvalandžius. Už viršvalandžius skaičiuojamas didesni atlyginimas, kuo daugiau viršvalandžių dirbama per dieną, tuo didesnis atlyginimas už darbą turi būti išmokėtas. Viršvalandžių darbo užmokestis valandai skaičiuojamas pagal formulę:

$$U = \frac{2 \cdot u + (k-1)^2 \cdot 0.5}{2};$$

čia: u – pradinis viršvalandžių valandos darbo užmokestis, k – viršvalandžių kiekis per dieną;

Jeigu skirtų dienų darbui atlikti (žymima δ) nepakanka tai bendras dienų kiekis skaičiuojamas taip:

$I = W/\kappa + k$; čia: κ - valandų kiekis per dieną, W - darbo kiekis valandomis;

Bendras užmokestis skaičiuojamas pagal formulę:

$C = U \cdot (I \cdot k) + \upsilon \cdot I \cdot \kappa$; čia: υ - darbo užmokestis už valandą ne viršvalandžių metu;

Bendra baudos suma skaičiuojama pagal formulę:

$B = V^2 \cdot b$; čia V – vėlavimas valandomis, b – bauda už valandą;

$V = W - (\delta \cdot \kappa + \delta \cdot k)$;

Taigi galutinis pelnas gaunamas:

$P = p - (B + C)$; čia: p – pradinis pelnas, kuris gaunamas taip:

$p = W \cdot \pi$; čia: π - pelnas gautas už atlikto darbo valandą;

Viršvalandžių uždavinio formali specifikacija:

1. Išorinė įėjimų aibė:

$X = \text{Ø}$;

2. Išorinių išėjimų aibė:

$Y = \text{Ø}$;

3. Išoriniai įvykiai:

$E' = \text{Ø}$;

4. Vidiniai įvykiai:

$E'' = \{e_1'' , e_2'' \}$;

čia: e_1'' - skaičiavimų pradžios momentas;

čia: e_2'' - skaičiavimų pabaigos momentas;

5. Valdančios sekos:

$e_1'' \mapsto \{\tau_0^1, \tau_1^1, \tau_2^1, ...\}$;

$e_2'' \mapsto \{\tau_0^2, \tau_1^2, \tau_2^2, ...\}$;

6. Būsena:

$\upsilon = d_1(t), d_2(t), d_3(t), ...d_N(t)$;

čia: $d_i(t)$ - *i*-tojo individo savybė (viršvalandžių kiekis per dieną), *N* – tai kiekis kiek įvyko skaičiavimų ir rezultatų palyginimų populiacijoje;

$Z_\upsilon(t) = \{\omega(e_1'', t), \omega(e_2'', t)\}$;

$n(t)$ - skaičius kiek įvyko lyginimų einamoje populiacijoje;

7. Pradinė būsena laiko momentu t_0:

$d_i(t) = \xi_i$, čia ξ_i - atsitiktinis dydis iš intervalo [0,8];

8. Perėjimų ir išėjimų operatoriai:

$H(e_1'')$:

$\alpha, \beta \in i \{ d_i(t_{m-1}) \neq \infty, 1 \leq i \leq N \}$;

Skaičiuojame galutinį pelną dviejų objektų α ir β, ir paliekame tik tą kurio pelnas didesnis;

$$d_\alpha(t_m) = \begin{cases} \infty, \varphi(d_\alpha(t_{m-1})) < \varphi(d_\beta(t_{m-1})); \\ d_\alpha(t_{m-1}), kitu\ atveju; \end{cases}$$

$$d_\beta(t_m) = \begin{cases} \infty, \varphi(d_\beta(t_{m-1})) < \varphi(d_\alpha(t_{m-1})); \\ d_\beta(t_{m-1}), kitu\ atveju; \end{cases}$$

čia: $\varphi(d(t_{m-1}))$ - funkcija suskaičiuojanti galutinį pelną pagal su nurodytu viršvalandžių kiekiu;

$$n(t_m) = \begin{cases} n(t_{m-1}) + 1, d_\alpha(t_m) + d_\beta(t_m) = \infty; \\ n(t_{m-1}), kitu\ atveju; \end{cases}$$ Padidinamas įvykusių lyginimų kiekis;

$\omega(e_1'', t_m) = t_m + \tau_m$;

$$\omega(e_2'', t_m) = \begin{cases} t_{m-1} + \frac{\tau_m}{2}, n(t_m) = \frac{N}{2}; \\ \omega(e_2'', t_m), kitu\ atveju; \end{cases}$$

$H(e_1'')$:

$n(t_m) = 0$;

Turime $d_1(t_{m-1}), d_2(t_{m-1}), d_3(t_{m-1}), \ldots d_N(t_{m-1})$ pažymėkime

$d_{i_1}(t_{m-1}), d_{i_2}(t_{m-1}), d_{i_3}(t_{m-1}), \ldots d_{i_N}(t_{m-1})$ taip, kad

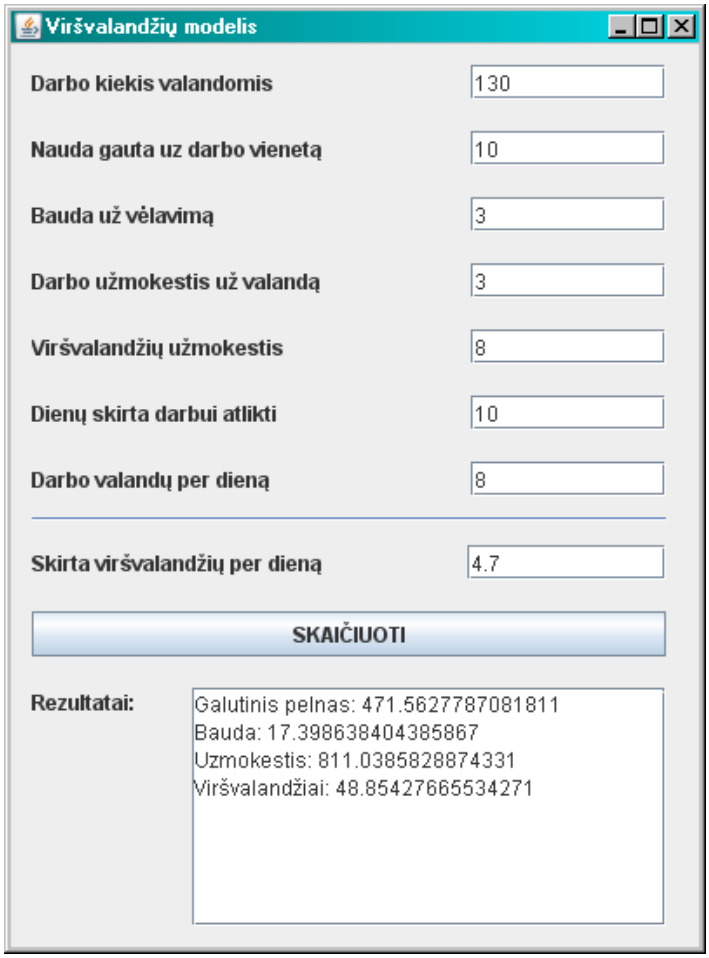
$d_{i_1}(t_{m-1}) \leq d_{i_2}(t_{m-1}) \leq d_{i_3}(t_{m-1}) \ldots \leq d_{i_N}(t_{m-1})$

Tuomet $d_{1_{\frac{N}{2}+j}}(t_m) = f(d_{i_j}(t_{m-1})), 1 \le j \le \frac{N}{2}$; Atkuriama populiacija;

čia $f(d_{i_j}(t_{m-1}))$ - funkcija kuri iš turimų individų sukuria naujus individus, tai gali būti rekombinacija ir/arba mutacija.

Šiame modelyje, genetiniame algoritme realizuojamas turnyrinės atrankos metodas, tai yra imami du individai, ir populiacijai atnaujinti imamas tik tas individas kurio pelnas didesnis.

Naudojant nurodytas formules skaičiavimuose, buvo sudarytas modelis – programa skaičiuojanti pelną pagal įvestus parametrus.



25 pav. Pelno skaičiavimo programos vartotojo sąsajos vaizdas

Buvo atliekami bandymai su šiuo modeliu ir taikant genetinio algoritmo principą ieškomas maksimalus pelnas, kai:

Darbo kiekis valandomis: 130

Pelnas gautas už atlikto darbo valandą: 10

Bauda už valandą (koeficientas *b*): 3

Darbo užmokestis už valandą: 3

Pradinis užmokestis už viršvalandžių valandą (koeficientas *u*): 8

Dienos skirtos darbui atlikti (laikas kol neskaičiuojama bauda): 10

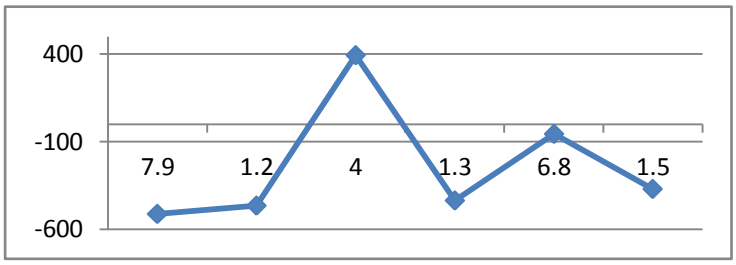
Darbo valandų per dieną: 8

Viršvalandžių per dieną galima skirti nuo 0 iki 8 valandų.

Pirmosios iteracijos viršvalandžių kiekiai buvo sugeneruoti atsitiktiniu būdu.

Lentelė 8 Pelno skaičiavimo bandymų rezultatai, iteracija 1

Viršvalandžiai per dieną	7.9	1.2	4	1.3	6.8	1.5
Galutinis pelnas	-512.3	-464.8	392.4	-433.5	-55.9	-368.9

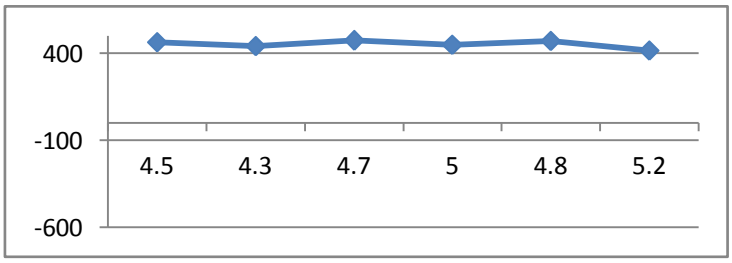


26 pav. Pelno pasiskirstymas, iteracija 1

Iš atrinktų geriausių egzempliorių (paryškinti) naudojant mutaciją sudaroma nauja populiacija. Taip tęsiant ketvirtoje iteracijoje (visų iteracijų rezultatus galima rasti antrajame priede) gauname:

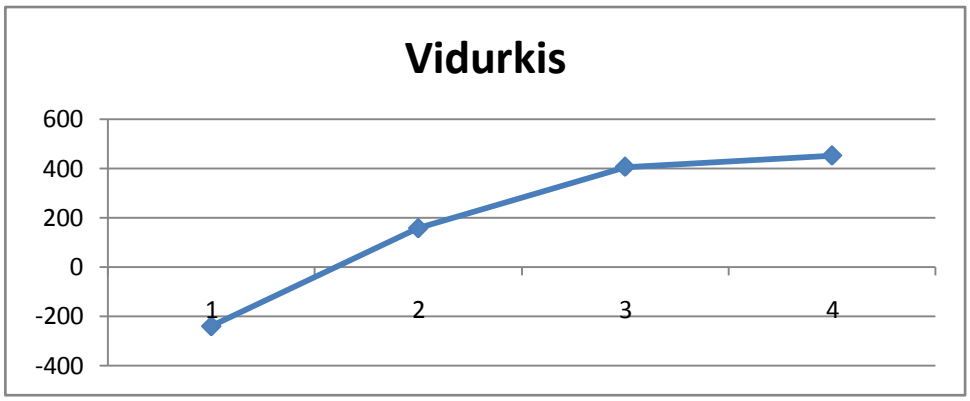
Lentelė Nr.9 Pelno skaičiavimo bandymų rezultatai, iteracija 4

Viršvalandžiai per dieną	4.5	4.3	4.7	5	4.8	5.2
Galutinis pelnas	463	441.9	471.6	448	469.8	414.7



27 pav. Pelno pasiskirstymas, iteraacija 4

Matome jog pelno kreive nusistovėjo virš 400, kai viršvalandžių kiekis per dieną yra tarp 4-5 valandų. Didžiausias pelnas gautas kai skiriama 4.7 valandos viršvalandžių per diena, tada pelnas 471.6.



28 pav. Pelno vidurkio kitimas visose iteracijose

Šiame ir dvikovos modeliuose agregatinis metodas buvo taikytas genetiniam algoritmui aprašyti, tiksliau buvo aprašyti individų atrankos metodai ir populiacijos atnaujinimas.

REZULTATAI IR IŠVADOS

Darbe nagrinėjama genetinių algoritmų taikymo galimybės, imituojamų sistemų charakteristikoms nustatyti. Sudaryti trys modeliai:

1. Keturšalės sankryžos modelis, pagal automobilių srautus automatizuotai reguliuojasi šviesoforo šviesų degimo laikus, taip, kad automobilių laukimo laikas būtų kuo trumpesnis.
2. Dvikovos uždavinyje reikia rasti optimalų šūvio atstumą, šiame uždavinyje pataikymo tikimybė priklauso nuo atstumo. Iš kuo mažesnio atstumo šaunama tuo pataikymo tikimybė didesnė, tačiau kuo vėliau šaunama tuo didesnė tikimybė kad priešas iššaus pirmas ir pataikys.
3. Viršvalandžių uždavinyje reikia rasti didžiausią pelną, reikia subalansuoti viršvalandžių kiekį, kad bauda už vėlavimą butų mažesnė, tačiau jai viršvalandžių bus skirta per daug tuomet pelnas vėl nukentės todėl reikia rasti pusiausvirą.

Pirmasis modelis leidžia tvirtinti, kad agregatinius modelius galima naudoti kaip genetinio algoritmo naudingumo funkciją jų funkcionavimo charakteristikoms nustatyti. Tai demonstruoja keturšalės sankryžos modelis, kuris prisitaiko prie aplinkos ir subalansuoja šviesoforo šviesų degimo laikus mažinant automobilių eiles ir jų laukimo laikus.

Antras ir trečias modeliai parodo, kad genetinis algoritmas gali būti agregatinio modelio dalis, norit išreikšti modeliuojamų sistemų elgesio, reakcijos į aplinką savybes. Dvikovos uždavinyje realizuotas ruletės atrankos metodas, o viršvalandžių uždavinyje realizuotas turnyrinės atrankos metodas.

Genetinių algoritmų ir agregatinio metodo apjungimas atveria naujas galimybes modeliuojamų sistemų charakteristikų įvertinime. Gauti rezultatai patvirtina faktą, kad apjungiant genetinį algoritmą ir agregatinį metodą galima gauti tikslesnes modeliuojamų sistemų charakteristikas, modeliuoti biologines, agentines ir savireguliuojančias sistemas.

Formalus sistemos aprašymo metodas DEVS[19] priklauso įvykiais valdomų sistemų modelių klasei, kaip ir agregatinis metodas, todėl darbe taikomas genetinio algoritmo panaudojimo būdas gali būti taikomas ir DEVS sistemų modeliams.

[19]Diskrečių įvykių sistemos specifikavimas [1]

LITERATŪRA

[1] http://en.wikipedia.org/wiki/DEVS (2010 05 20).

[2] http://lt.wikipedia.org/wiki/Genetiniai_algoritmai (2010 05 20).

[3] Henrikas Pranevičius. Sudėtingų sistemų formalizavimas ir analizė, Mokslo aidai, 2008.

[4] Henrikas Pranevičius, Šarūnas Raudys, Algimantas Rudžionis, Vytautas Rudžionis, Kastytis Ratkevičius, Jūratė Sakalauskaitė, Dalius Makackas. Agentinių sistemų modeliai, Mokslo aidai, 2008.

[5] E. Vyšniauskas. Programiniai agentai. Prieiga per internetą: <http://kopustas.elen.ktu.lt/studentai/_media/ernestas_vysniauskas_-_pa.doc?... > (2010 05 20).

[6] J. Trinkūnas. Neuroniniai tinklai ir genetiniai algoritmai. Prieiga per internetą: <http://ik.su.lt/~grazvis/di/Ivairi%20medziaga,%20nuorodos/Neurotinklai%20ir%20Gen.algoritmai.doc> (2010 05 20).

[7] http://lt.wikipedia.org/wiki/Dirbtinis_neuroninis_tinklas (2010 05 20).

[8] http://en.wikipedia.org/wiki/Neural_network (2010 05 20).

[9] K. Wang, H. Pranevičius. Applications of AI to Product Engineering, KTU Press Technologija 1997, 380.

[10] V. Jukavičius. Realaus laiko sistemų verifikavimas panaudojant PLA modelį. Prieiga per internetą:

<http://www.stud.ktu.lt/~vaijuka/Dokumentai/Realaus%20laiko%20sistemu%20verifikavimas%20panaudojant%20PLA%20modeli.pdf> (2010 05 20).

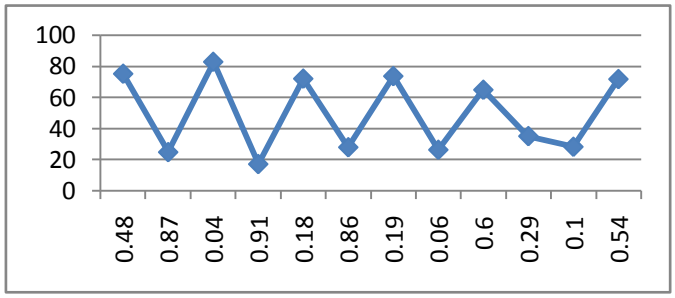
PRIEDAI

1 priedas. Dvikovos uždavinio modelio bandymų rezultatai

Iteracija 1

šūvio laikas	0.48	0.87	0.04	0.91	0.18	0.86	0.19	0.06	0.6	0.29	0.1	0.54
pergalės procentas	75.1	24.9	82.73	17.27	71.95	28.05	73.58	26.42	64.84	35.16	28.33	71.67

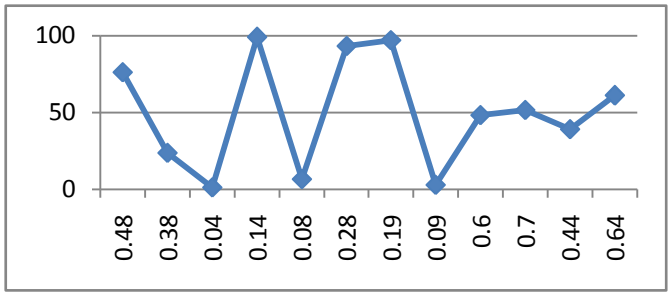
Vidurkis: 0.426667



Iteracija 2

šūvio laikas	0.48	0.38	0.04	0.14	0.08	0.28	0.19	0.09	0.6	0.7	0.44	0.64
pergalės procentas	76.37	23.63	1.13	98.87	6.67	93.33	96.98	3.02	48.43	51.57	38.99	61.01

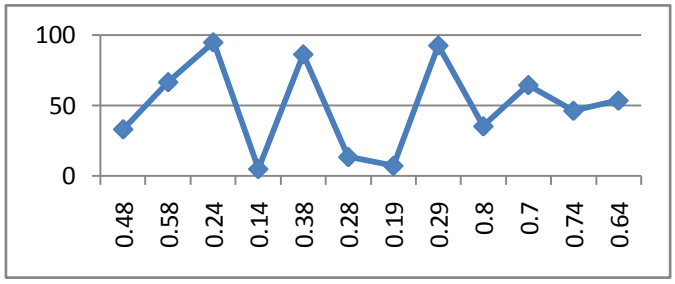
Vidurkis: 0.338333



Iteracija 3

šūvio laikas	0.48	0.58	0.24	0.14	0.38	0.28	0.19	0.29	0.8	0.7	0.74	0.64
pergalės procentas	33.33	66.67	94.8	5.2	86.33	13.67	7.41	92.59	35.42	64.58	46.39	53.61

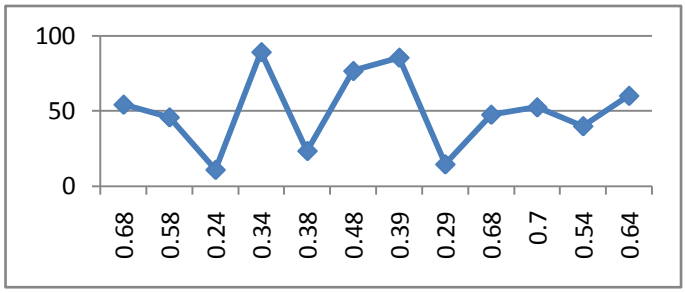
Vidurkis: 0.455



Iteracija 4

šūvio laikas	0.68	0.58	0.24	0.34	0.38	0.48	0.39	0.29	0.68	0.7	0.54	0.64
pergalės procentas	54.24	45.76	10.87	89.13	23.39	76.61	85.47	14.53	47.54	52.46	39.91	60.09

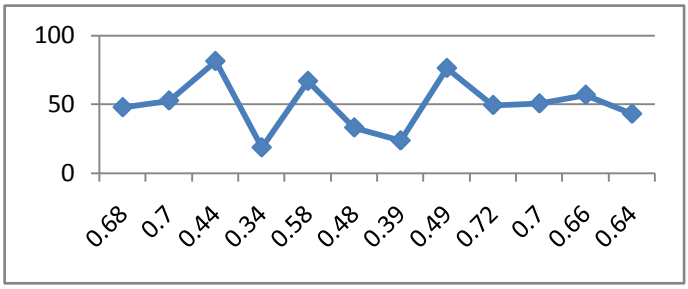
Vidurkis: 0.495



Iteracija 5

šūvio laikas	0.68	0.7	0.44	0.34	0.58	0.48	0.39	0.49	0.72	0.7	0.66	0.64
pergalės procentas	47.83	52.7	81.41	18.59	66.93	33.07	23.63	76.37	49.41	50.59	56.91	43.09

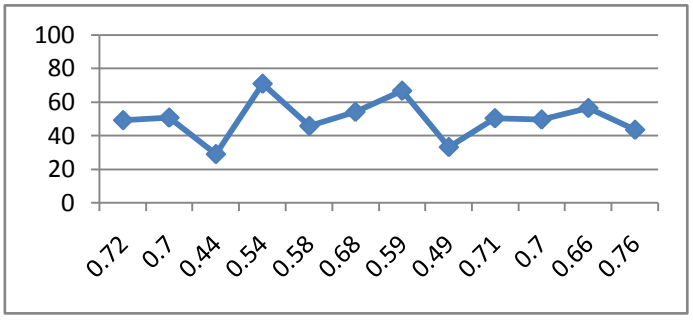
Vidurkis: 0.568333



Iteracija 6

šūvio laikas	0.72	0.7	0.44	0.54	0.58	0.68	0.59	0.49	0.71	0.7	0.66	0.76
pergalės procentas	49.09	50.91	29.08	70.92	45.76	54.24	66.64	33.36	50.25	49.75	56.57	43.43

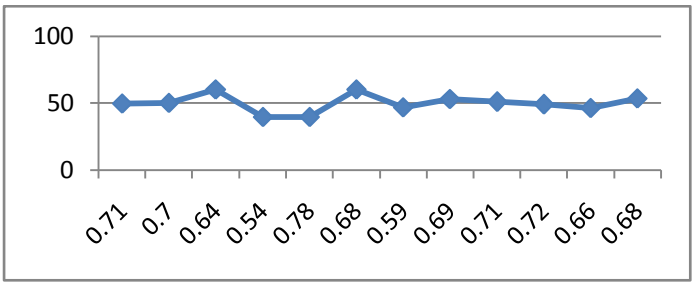
Vidurkis: 0.630833



Iteracija 7

šūvio laikas	0.71	0.7	0.64	0.54	0.78	0.68	0.59	0.69	0.71	0.72	0.66	0.68
pergalės procentas	49.73	50.27	60.16	39.84	39.71	60.29	46.94	53.06	50.96	49.04	46.48	53.52

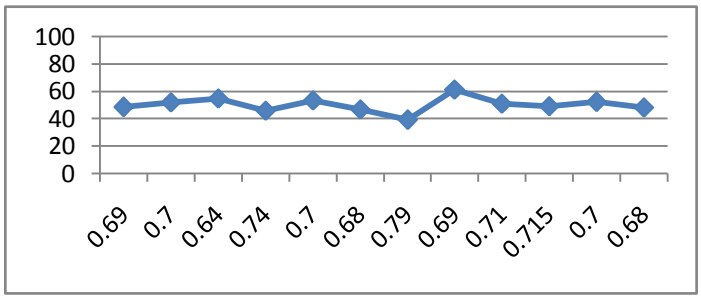
Vidurkis: 0.675



Iteracija 8

šūvio laikas	0.69	0.7	0.64	0.74	0.7	0.68	0.79	0.69	0.71	0.715	0.7	0.68
pergalės procentas	48.34	51.66	54.43	45.57	53.16	46.84	38.95	61.05	50.88	49.12	52.07	47.93

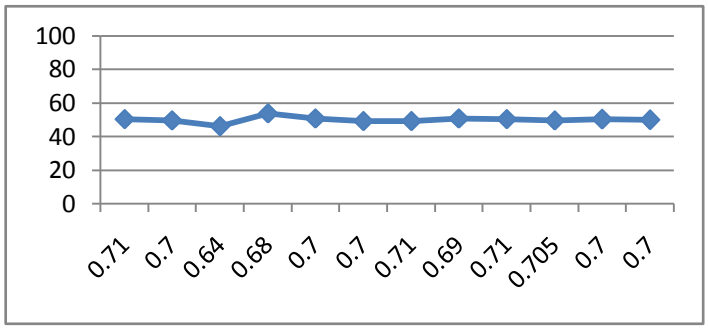
Vidurkis: 0.702917



Iteracija 9

šūvio laikas	0.71	0.7	0.64	0.68	0.7	0.7	0.71	0.69	0.71	0.705	0.7	0.7
pergalės procentas	50.34	49.66	46.16	53.84	50.76	49.24	49.35	50.65	50.32	49.68	50.19	49.81

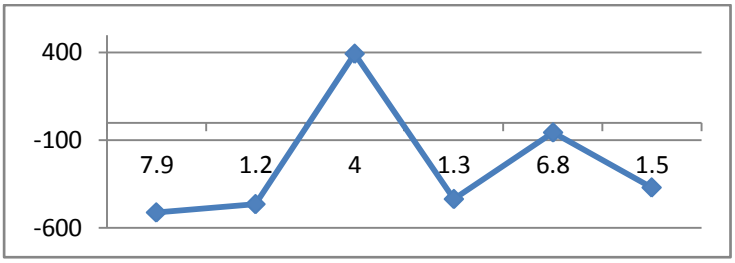
Vidurkis: 0.695417



2 priedas. Viršvalandžių uždavinio modelio bandymų rezultatai

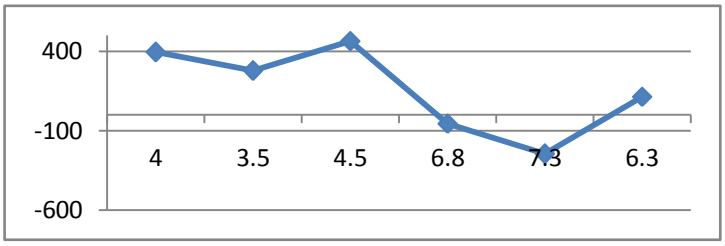
Iteracija 1

							Vidurkis
virsvalandziai per diena	7.9	1.2	4	1.3	6.8	1.5	3.783333
galutinis pelnas	-512.3	-464.8	392.4	-433.5	-55.9	-368.9	-240.5



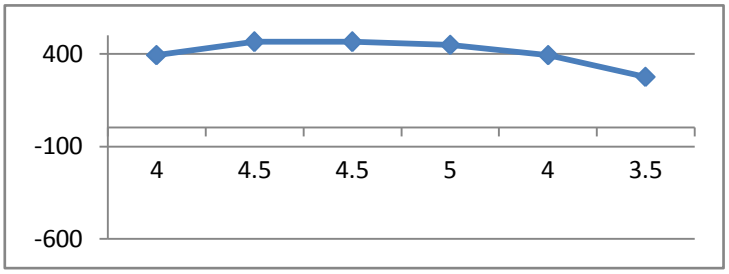
Iteracija 2

							Vidurkis
virsvalandziai per diena	4	3.5	4.5	6.8	7.3	6.3	5.4
galutinis pelnas	392.4	275.9	463	-55.9	-248.3	113.6	156.7833



Iteracija 3

							Vidurkis
virsvalandziai per diena	4	4.5	4.5	5	4	3.5	4.25
galutinis pelnas	392.4	463	463	448	392.4	275.9	405.7833



Iteracija 4

							Vidurkis
virsvalandziai per diena	4.5	4.3	4.7	5	4.8	5.2	4.75
galutinis pelnas	463	441.9	471.6	448	469.8	414.7	451.5