ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Linas Matakas
Informatikos specialybės IV kurso dieninio skyriaus studentas

Dirbtinės bičių kolonijos algoritmai ir jų taikymai skirstymo uždaviniams spręsti

Artificial Bee Colony Algorithms and their Application to Assignment Problems

BAKALAURO DARBAS

Darbo vadovas:
Doc. dr. G. Felinskas
Recenzentas:

Šiauliai, 2013

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autoriaus _____

(vardas, pavardė, parašas)

Darbo tikslai ir uždaviniai

Tikslas:

- Išanalizuoti dirbtinės bičių kolonijos algoritmus ir sukurti programinę realizaciją skirstymo uždaviniams spręsti.

Uždaviniai:

- Susipažinti su dalelių spiečių sistemų (Swarm systems) algoritmais.
- Susipažinti su dirbtinės bičių kolonijos (ArtificialBeeColony, ABC) algoritmo paradigmomis bei taikymais.
- Susipažinti su skirstymo (assigment) uždavinių įvairiomis formuluotėmis, interpretacijomis ir taikymais.
- Suprojektuoti ABC algoritmo funkcijas ir šio algoritmo procedūrų praktines interpretacijas.
- Sukurti programinę realizaciją su ABC algoritmo taikymais skirstymo uždaviniams spręsti.
- Atlikti sukurtos programinės įrangos testavimą, pateikti skaičiavimų rezultatus.

Darbo vadovas _____

(vardas, pavardė, parašas)

Turinys

Įvadas 5
1 Teorinė dalis 6
1.1 Dalelių spiečių sistemos 6
1.1.1 Dalelių spiečiaus optimizavimas 6
1.1.2 Skruzdėlių kolonijos sistema 6
1.1.3 Bakterijų maitinimosi optimizacijos algoritmas 7
1.2 Dirbtinės bičių kolonijos algoritmai 7
1.2.1 Operacijos 8
1.2.2 Dirbtinės bičių kolonijos algoritmų struktūra 9
1.2.3 Taikymai 10
1.3 Skirstymo uždaviniai 10
1.3.1 Skirstymo uždavinių formuluotės 11
1.3.2 Skirstymo uždavinių praktinės inerpretacijos 11
2. Projektinė dalis 12
2.1 Įrankių ir priemonių pasirinkimas 12
2.2 Darbų grafikas 13
2.3 Pradinis projektas 14
3. Realizacinė dalis 16
3.1 Darbų ataskaita 16
3.2 Galutinis projektas 17
3.3 Problemų sąrašas 20
3.4 Darbo rezultatų analizė 20
3.5 Rekomendacijos 25
Išvados 26
Literatūros sąrašas 27
Anotacija 29
Summary 29
Priedai 30
1. CD turinys: 30

Įvadas

Tobulėjant kompiuterinėms technologijoms, kompiuteriai yra naudojami vis didesniam kiekiui uždavinių spręsti. Kai kurie šių uždaviniai pasižymi labai dideliu duomenų kiekiu ar laiko jiems išspręsti ribojimu, todėl juos spręsti reikia pasitelkti sudėtingus algoritmus, nors jie ir nebūtinai ras optimalų sprendinį, bet jų sprendinys bus bent pakankamai geras ar statiškai vienas geriausių palyginus su kitais algoritmais. Paskutiniais dešimtmečiais šie algoritmai yra aktyviai kuriami stebint gamtoje sutinkamus reiškinius ar kitų gyvybės formų elgseną. Vienas iš reiškinių yra bičių kolonijos darbas renkant žiedadulkes ir pagal jį sukurtas dirbtinės bičių kolonijos algoritmas, kuris yra naudojamas spręsti optimizavimo uždaviniams.

Darbo tikslas - sukurti programinę realizaciją su dirbtinės bičių kolonijos algoritmo taikymais. Pagrindiniai darbo uždaviniai - susipažinti su dalelių spiečių sistemų algoritmais, dirbtinės bičių kolonijos algoritmo paradigmomis bei taikymais, skirstymo uždavinių įvairiomis formuluotėmis bei praktinėmis interpretacijomis ir pritaikyti dirbtinės bičių kolonijos algoritmą šiems uždaviniams spręsti.

1 Teorinė dalis

1.1 Dalelių spiečių sistemos

Dalelių spiečių sistemos yra bendro funkcionavimo, be centrinio valdymo, savarankiškai organizuotos sistemos. Jos gali būti dirbtinės arba natūralios: skruzdėlių kolonijos, paukščių pulkai, žuvų būriai, bičių kolonijos. Gamtoje šios sistemos yra naudojamos maisto šaltinių paieškai, kolonijų padėties keitimui. Informacija šiuose sistemose yra dažniausiai perduodama per vienarūšius agentus arba yra laikoma ar nurodoma aplinkoje: skruzdėlių paliekamuose feromonuose, bičių šokiuose, paukščių tarpusavio atstumuose.[1], [5]

Algoritmai:

- Dalelių spiečiaus optimizavimas
- Skruzdėlių kolonijos sistema
- Bakterijų maitinimosi optimizacijos algoritmas
- Bičių algoritmai

1.1.1 Dalelių spiečiaus optimizavimas

Šis algoritmas buvo įkvėptas paukščių ir žuvų maisto šaltinių paieškos metodika. Dalelėms būna paskirtos atsitiktinės pradinės vietos ir maži pradiniai greičiai. Dalelės juda po aplinka ir kas kažkiek laiko būna užfiksuojamas jų vietos tinkamumas. Po kiek laiko dalelės juda po aplinką sekdamos tinkamesnius spiečiaus narius ir kreipdamos savo judėjimą link istoriškai geresnių aplinkos vietų ar vietos. [2], [5]

1.1.2 Skruzdėlių kolonijos sistema

Šis algoritmas buvo įkvėptas skruzdėlių maisto šaltinių paieškos metodika, ypač jų komunikavimas aplinkoje feromonais nurodant kelią tarp kolonijos ir maisto šaltinio. Iš pradžių skruzdėlės juda po aplinka atsitiktinai. Kai būna randamas maisto šaltinis skruzdėlė keliauja nuo maisto šaltinio iki kolonijos palikdama feromonus savo kely. Jei tas kelias yra sekamas kitų skruzdėlių ir ten yra maisto tada būna paliekama daugiau feromonų. Kadangi feromonai aplinkoje nyksta todėl seni keliai pasidaro nebesekami. Kai kitos skruzdėlės atranda tą patį maisto šaltinį, tai padidina feromonų kiekį tarp kolonijos ir gerų maisto šaltinių. Teigiamas grįžtamasis ryšys pasiunčia vis daugiau skruzdžių link geriausių žinomų sprendinių, kurie palaipsniui yra gerinami. Sprendiniai yra gerinami po vieną žingsnį atsižvelgiant skaičiuojant tikimybes.[3], [5]

1.1.3 Bakterijų maitinimosi optimizacijos algoritmas

Šis algoritmas buvo sukurtas stebint bakterijų elgesį ieškant maisto šaltinių, nes jos supranta ir pajunta tam tikrus cheminius junginius pagal kuriuos nusprendžia ar judėti link jų ar tolyn. Taip pat jos gali išskirti tam tikrus cheminius junginius, taip „bendraudamos“ jos motyvuoja kitas ląsteles judėti link geresnių sprendinių ir/ar nekeliauti ten, kur jau kažkas buvo. [4]

1.2 Dirbtinės bičių kolonijos algoritmai

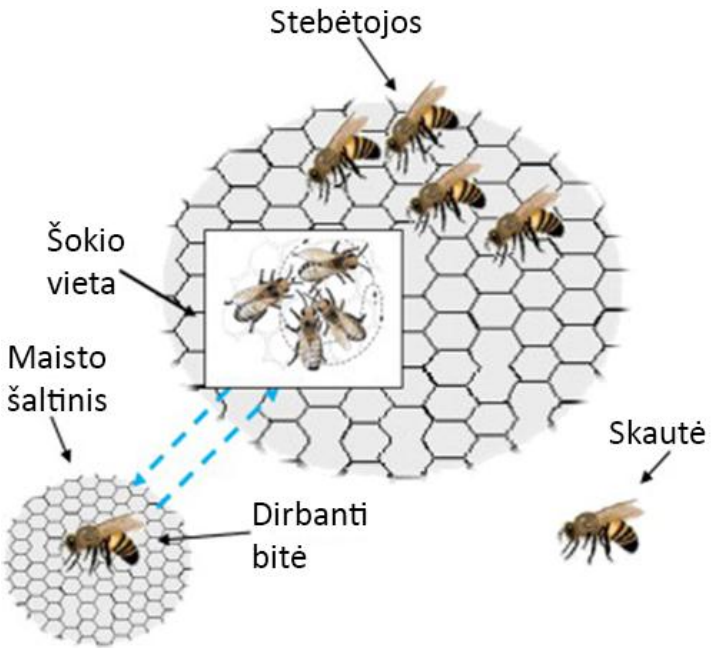
Dirbtinės bičių kolonijos algoritmai (Artificial Bee Colony, ABC) yra meta-euristiniai (iteracijomis bandantys pagerinti jau turimus sprendinius atsižvelgiant į reikiamos kokybės koeficientą), spiečiaus veikimu paremti algoritmai, kurie yra skirti optimizuoti skaičiavimo uždavinius ir buvo pristatyti Dervio Karabogos (Dervis Karaboga) 2005 metais. Šie algoritmai buvo sukurti stebint bičių kolonijos bičių, kurios renka žiedadulkes, darbo pasiskirstymą ir eigą. Šiuose algoritmuose dažniausiai yra trijų rūšių bitės: dirbančios (employed), stebinčios (onlookers) ir skautės (scouts) (Pav. 1). Stebinčios bitės ir skautės bitės taip pat gali būti vadinamos nedirbančiomis bitėmis. Dirbančios bitės keliauja į jų darbo vietas (maisto šaltinius), tada grįžta atgal į avilį ir sušoka šokį savo darbo vietai. Kai jų maisto šaltinis išsenka, jos tampa skautėmis arba nedirbančiomis bitėmis. Stebėtojos žiūri dirbančių bičių šokius ir pagal juos atsirenka į kurį maisto šaltinį keliauti. Dirbančios bitės pasirenka maisto šaltinius remdamosis savo patirtimi, stebinčios bitės jas pasirenka nusižiūrėdamos dirbančias bites, o skautės nesinaudoja patirtimi ir ieško maisto šaltinių remdamosis atsitiktinumu. Jei naujai atrastas maisto šaltinis yra didesnis nei praėjęs didžiausias, tai senasis yra pamirštamas ir prisimenamas tik didžiausias. Pradiniai maisto šaltiniai yra randami tik skaučių. ABC algoritmai apjungia lokalius paieškos metodus, atliekamus dirbančių ir stebinčių bičių, ir globalius paieškos metodus, atliekamus stebinčių ir skaučių bičių, bandant subalansuoti tyrimo ir panaudojimo procesus. [6], [7], [8]

Skautės bitės sprendinius gauna atsitiktinai arba specifiniam uždaviniui tinkančiais metodais. Pirmuoju atveju skautės bitės gali būti panaudotos bet kurioje sprendimo stadijoje. Antruoju atveju jos dažniausiai būna panaudojamos tik pradiniam sprendiniui gauti. Sprendžiant skirstymo uždavinius atsitiktinis skaučių bičių sprendinys dažniausiai yra naudojamas, kai agentų ir užduočių skaičius sutampa ar yra panašus ar uždavinys turi bent porą ribojimų kaip agentų

resursų limitas, kiekvienas agentas turi atlikti bent vieną užduotį, kai kurios užduotys gali būti atliktos tik po to, kai atliktos kitos užduotys. [6]

Kai skautės suranda sprendinį, jos tampa dirbančiomis bitėmis. Dirbančios bitės sprendinius keičia tik tuo atveju, jei žino ar randa geresnį. Jos saugo geriausius sprendinius ir dalis dirbančių bičių, kurios turi blogiausius sprendinius, tampa skautėmis ar stebėtojomis. Tam, kad gauti geresnius sprendinius, dirbančios bitės bando perkelti užduotis arba jas sukeisti vietomis, jei gaunamas sprendinys yra blogesnis, jis iškart atmetamas. [6]

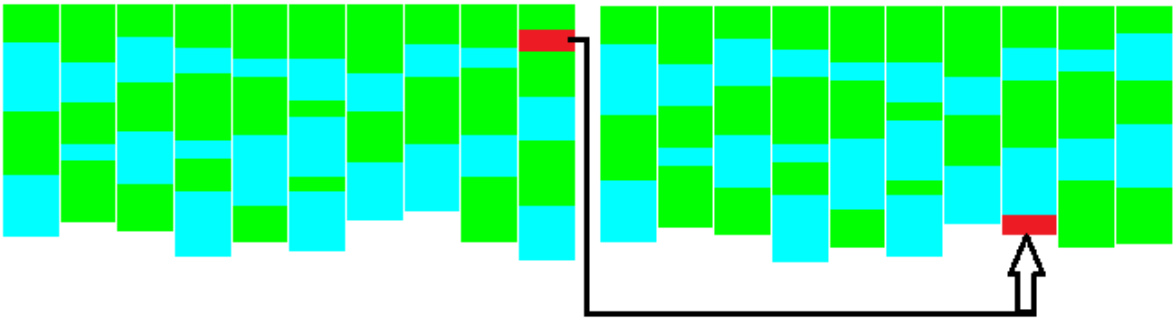
Kai dirbančios bitės grįžta atgal į avilį, jos sušoka šokį, pagal kurį stebinčios bitės išsirenka, kuriuos sprendinius nusižiūrėti ir atlikus grandinės operaciją juos pakeičia. Tada jos tampa dirbančiomis bitėmis. [6]



Pav. 1 Bičių kolonija gamtoje

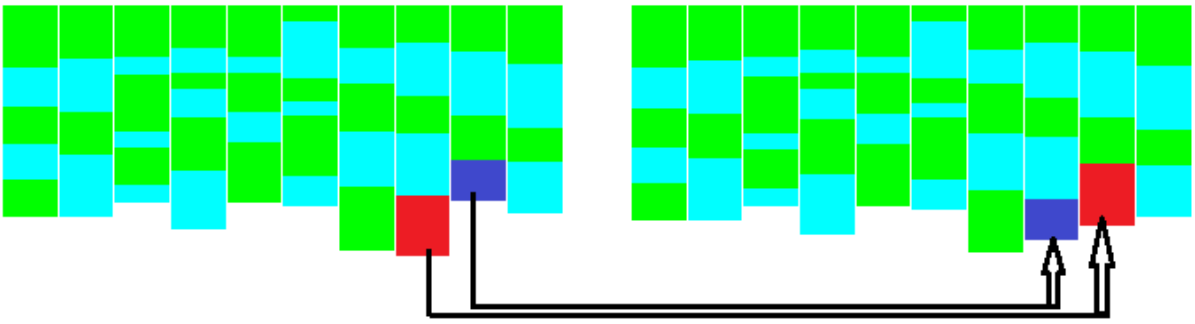
1.2.1 Operacijos

- Perkėlimas (Pav. 2)



Pav. 2 Perkėlimo operacija

- Apkeitimas (Pav. 3)



Pav. 3 Apkeitimo operacija

- Grandinė - išmetamas vienas sprendinys, atliekama perkėlimų eilė ir bandomas įdėti išmestas sprendinys, tikrinant ar pagerėjo sprendinys įdėjus atgal. Perkėlimų kiekis būna ribotas.

1.2.2 Dirbtinės bičių kolonijos algoritmų struktūra

- Pradinių duomenų inicijavimas - gaunamas pradinis sprendinys ar sprendiniai.
- Kartojama
    - Dirbančių bičių fazė - dirbančios bitės bando rasti geresnį sprendinį, naudodamos perkėlimo ir apkeitimo operacijas.
    - Nusižiūrinčių bičių fazė - kiek nusižiūrinčių bičių bus priskirta atskiroms dirbančioms bitėms yra nustatoma pagal dirbančių bičių tinkamumo įvertinimus. Nusižiūrinčios bitės bando pagerinti dirbančių bičių sprendinius, naudodamos pritaikytais uždaviniui metodais.

- Skaučių bičių fazė - dirbančios bitės, kurios negalėjo pagerinti savo sprendinių, tampa skautėmis arba nedirbančiomis bitėmis.
- Įsimenamas geriausias sprendinys

- Tol, kol pasiektas maksimalus ciklų skaičius [6], [7], [8]

1.2.3 Taikymai

- Keliaujančio pirklio uždavinys
- Parduotuvės darbo tvarkaraštis
- Mobiliojo ryšio tinklų maršrutizavimo protokolas
- Spręsti „sudoku“ dėliones
- Skaitmeninė ir inžinierinė optimizacija
- Apibendrintas skirstymo uždavinys
- Patarimų sistema
- Skirstymo uždaviniai
- Maršrutizavimo uždaviniai

1.3 Skirstymo uždaviniai

Skirstymo uždaviniai dažniausiai sprendžia kaip priskirti n kiekiui objektų m kiekį kitų objektų taip, kad jie būtų priskirti geriausiu būdu. Objektai n dažniausiai būna darbuotojai, konteineriai, talpyklos, mašinos ar žmonės, o objektai m dažniausiai būna užduotys, darbai ar daiktai. Objektai n yra vadinami agentais ir objektai m vadinami užduotimis. [10], [11]

Pagrindiniai skirstymo uždavinių skirtumai:

- Agentų ir užduočių santykiu
  - Agentų skaičius lygus užduočių skaičiui
  - Agentų skaičius mažesnis už užduočių skaičių
- Agentų savybėmis
  - Visi agentai vienodi ar skiriasi jų efektyvumas
  - Agentas gali turi ribotus resursus
- Užduočių savybėmis
  - Užduotys atliekamos bet kokia tvarka ar kai kurios užduotys turi būti atlikos anksčiau
  - Keliems agentams galima priskirti užduotį: vienam ar daugiau.

- Agentų ir užduočių sąsaja
    - Agentai tas pačias užduotis atlieka per skirtinga laiką ar per tą patį laiką (nepriklausomai nuo agentų asmeninio efektyvumo)

1.3.1 Skirstymo uždavinių formuluotės

1. Tiesinis skirstymo uždavinys: turimas n kiekis agentų ir n kiekis užduočių, agentai tas pačias užduotis atlieka per skirtingą laiką. Reikia priskirti vieną agentą vienai užduočiai ir vieną užduotį vienam agentui. Dažniausiai ieškomas minimalus maksimumas. [10], [11]
2. Apibendrintas skirstymo uždavinys: turimas n kiekis agentų ir m kiekis užduočių. Bet kiek užduočių gali būti priskirta vienam agentui, bet viena užduotis negali būti priskirta keliems agentams. Agentai gali būti susiję su užduotimis (nauda ir resursų išnaudojimu) ir turi ribotus resursus, užduotys turi būti daromos viena po kitos. Dažniausiai ieškomas maksimalus pelnas. [10]
3. Kuprinės uždavinys: turimas vienas agentas (kuprinė) ir n kiekis užduočių (daiktų). Daiktai turi vertę ir svorį, kuprinė turi svorio limitą. Reikia sudėti daiktus taip, kad nebūtų viršytas kuprinės limitas ir surinkta kuo didesnė visų daiktų vertė.
4. Nesubalansuotas laiko minimizavimo skirstymo uždavinys arba silpnosios grandies skirstymo uždavinys: turimas n kiekis agentų ir m (>n) kiekis užduočių. Bet kiek užduočių gali būti priskirta vienam agentui, bet viena užduotis negali būti priskirta keliems agentams, užduotys turi būti daromos viena po kitos. Agentai tas pačias užduotis gali atlikti per skirtingą laiką ar kai kurios užduotys turi būti įvykdytos vėliau nei kai kurios kitos pabaigtos. Tikslas yra paskirstyti visas užduotis taip, kad vėliausiai atlikta užduotis būtu atlikta kuo anksčiau, t.y. randamas minimalus maksimumas. [12], [13]

1.3.2 Skirstymo uždavinių praktinės interpretacijos

1. Tiesinis skirstymo uždavinys: turimas N kiekis mašinų ir N kiekis darbų. Sudaroma matrica NxN su kiekvienos mašinos atlikimo laiku kiekvienam darbui.

90	75	75	80
35	85	55	65
125	95	90	105
45	110	95	115

Lentelė 1 Tiesinio skirstymo uždavinio pavyzdys

Reikia išrinkti iš kiekvienos eilutės ir stulpelio po vieną skaičių taip, kad maksimumas būtu mažiausias.

2. Nesubalansuotas laiko minimizavimo skirstymo uždavinys arba silpnosios grandies skirstymo uždavinys: turimas n kiekis darbuotojų ir m (>n) kiekis darbų. Reikia paskirstyti darbus taip, kad visi darbai būtu baigti kuo anksčiau.
3. Nesubalansuotas laiko minimizavimo skirstymo uždavinys arba silpnosios grandies skirstymo uždavinys: turimas n kiekis žmonių ir m (>n) kiekis daiktų. Daiktai turi svorius. Reikia paskirstyti daiktus taip, kad maksimalus daiktų svoris vienam žmogui būtu kuo mažesnis.
4. Nesubalansuotas laiko minimizavimo skirstymo uždavinys arba silpnosios grandies skirstymo uždavinys: turimas n kiekis spausdintuvų ir m (>n) kiekis spausdinimo užduočių. Reikia paskirstyti užduotis taip, kad visos užduotys būtu baigtos kuo anksčiau.

2. Projektinė dalis

Uždavinys, kuriam buvo pritaikytas algoritmas yra nesubalansuotas laiko minimizavimo skirstymo uždavinys arba silpnosios grandies skirstymo uždavinys, kai kiekvienas agentas tas pačias užduotis atlieka per skirtingą laiką, užduočių yra daugiau nei agentų, užduotys yra atliekamos viena po kitos ir reikia užduotis paskirstyti taip, kad visos užduotys būtų baigtos kuo anksčiau.

Šiame uždavinyje

- skautės bitės ras pradinį sprendinį, t.y. paskirstys visas užduotis agentams.
- dirbančios bitės saugos sprendinius ir kiekvienos iteracijos pradžioje mėgins pagerinti sprendinį pasinaudodamos perkėlimo ir apkeitimo operacijomis, o kiekvienos iteracijos pabaigoje atmes prasčiausius sprendinius.
- nusižiūrinčios bitės nukopijuos kiekvienos dirbančios bitės sprendinius ir atliks grandinės operaciją, taip jos taps dirbančiomis bitėmis.

2.1 Įrankių ir priemonių pasirinkimas

Šiai programai įgyvendinti buvo norima pasirinkti įrankį su kurio pagalba būtų galima nesunkiai stebėti algoritmo veikimą vykdymo metu testavimo stadijoje. Taip pat buvo norima pasirinkti įrankį su kuriuo būtu lengva įgyvendinti vartotojo sąsaja algoritmo eksperimentavimui. Buvo renkamasi iš Qt(C++), Visual Studio(C++,C#), Rhino, JavaScript. Qt ir Visual Studio buvo nepasirinkti todėl, kad būtu sunku eksperimentuoti algoritmą žingsnis po žingsnio, ypač jei

norima vykdyti kodą vartotojo sąsajoje. Rhino ir JavaScript turi galimybę vykdyti kodą vartotojo sąsajoje. Kadangi Rhino veikia Java aplinkoje, o JavaScript naršyklėse, buvo pasirinkta JavaScript dėl to, kad ji turi daug bibliotekų, kurios palengviną darbą su elementais ir objektais, naršyklės turi konsoles, elementų struktūrinę peržiūrą, tai leidžia nesunkiai apjungti tekstinę ir vizualią informaciją testavimams. Taip pat šios savybės leidžia nesunkiai keisti vizualų pateikimą, todėl nereikia rūpintis dėl įvairių kiekių duomenų pateikimo prieš atliekant skaičiavimus. Be to, daugiau vartotojų gali naudotis naršyklėmis nei Java aplikacijomis.

Rašyti JavaScript galima naudotis bet kokiu teksto redaktoriumi, bet įgyvendinti vartotojo sąsajai taip pat reikia HTML ir CSS, norint, kad programos kodas būtų tvarkingas reikia rašyti CSS, HTML, JavaScript į skirtingus failus. Naudojantis programavimo aplinkomis kaip Qt ar Visual Studio susiję projekto failai būna nesunkiai pasiekiami, t.y. nereikia jų visų atskirai atsidarinėti. Tam galima naudoti Notepad++, nes jis gali prisiminti kokie failai buvo atidaryti paskutinį kartą arba Adobe Dreamweaver, kuris atidaręs HTML failą automatiškai sukuria susijusių failų nuorodas ir leidžia taisyti lokalius failus, be atskiro jų prijungimo. Taip pat Adobe Dreamwaver duoda objektų metodų užuominas. Todėl programai rašyti buvo pasirinkta išbandomoji Adobe Dreamweaver versiją.

2.2 Darbų grafikas

		Vasaris				Kovas				Balandis				Gegužė
		Savaitės												
Nr.	Valandų sk.	1	2	3	4	5	6	7	8	9	10	11	12	13
1	100													
2	30													
3	70													
4	70													
5	50													
Iš viso:	320													

Lentelė 2 Darbų grafikas

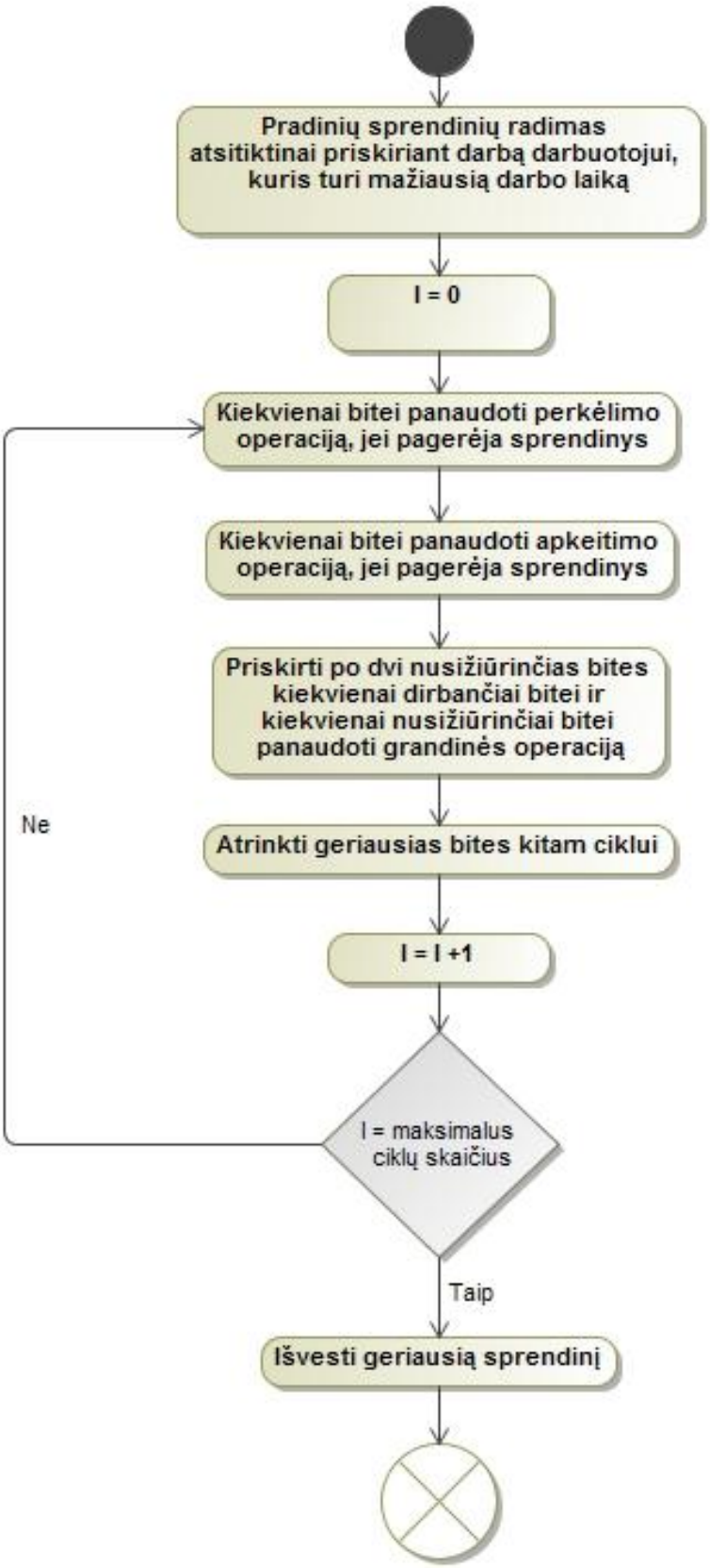
Darbų paaiškinimas:

1. Informacijos paieška ir analizė.

2. Aptarimas su darbo vadovu.
3. Programinės įrangos kūrimas.
4. Programinės įrangos testavimas.
5. Darbo aprašymas.

2.3 Pradinis projektas

1. Pradinių duomenų aprašymas kode.
2. Parašyti pradinio sprendinio funkciją.
3. Parašyti sprendinio piešimo funkciją.
4. Nustatyti sprendinio tinkamumo parametrus ir kada juos apskaičiuoti.
5. Parašyti perkėlimo operacijos funkciją.
6. Parašyti apkeitimo operacijos funkciją.
7. Parašyti grandinės operacijos funkciją.
8. Pritaikyti dirbtinės bičių kolonijos algoritmą (Pav. 4) panaudojant visus prieš tai išvardintus punktus.
9. Rasti rekomenduotinus parametrus vidinių ciklų kiekiams ir užtikrinti, kad nebus prarandami geresni sprendiniai.
10. Sukurti pradinių duomenų įvedimą vartotojo sąsajoje.



Pav. 4 Dirbtinės bičių kolonijos algoritmas

3. Realizacinė dalis

3.1 Darbų ataskaita

		Vasaris				Kovas				Balandis				Gegužė
		Savaitės												
Nr.	Valandų sk.	1	2	3	4	5	6	7	8	9	10	11	12	13
1	120													
2	25													
3	65													
4	60													
5	50													
Iš viso:	320													

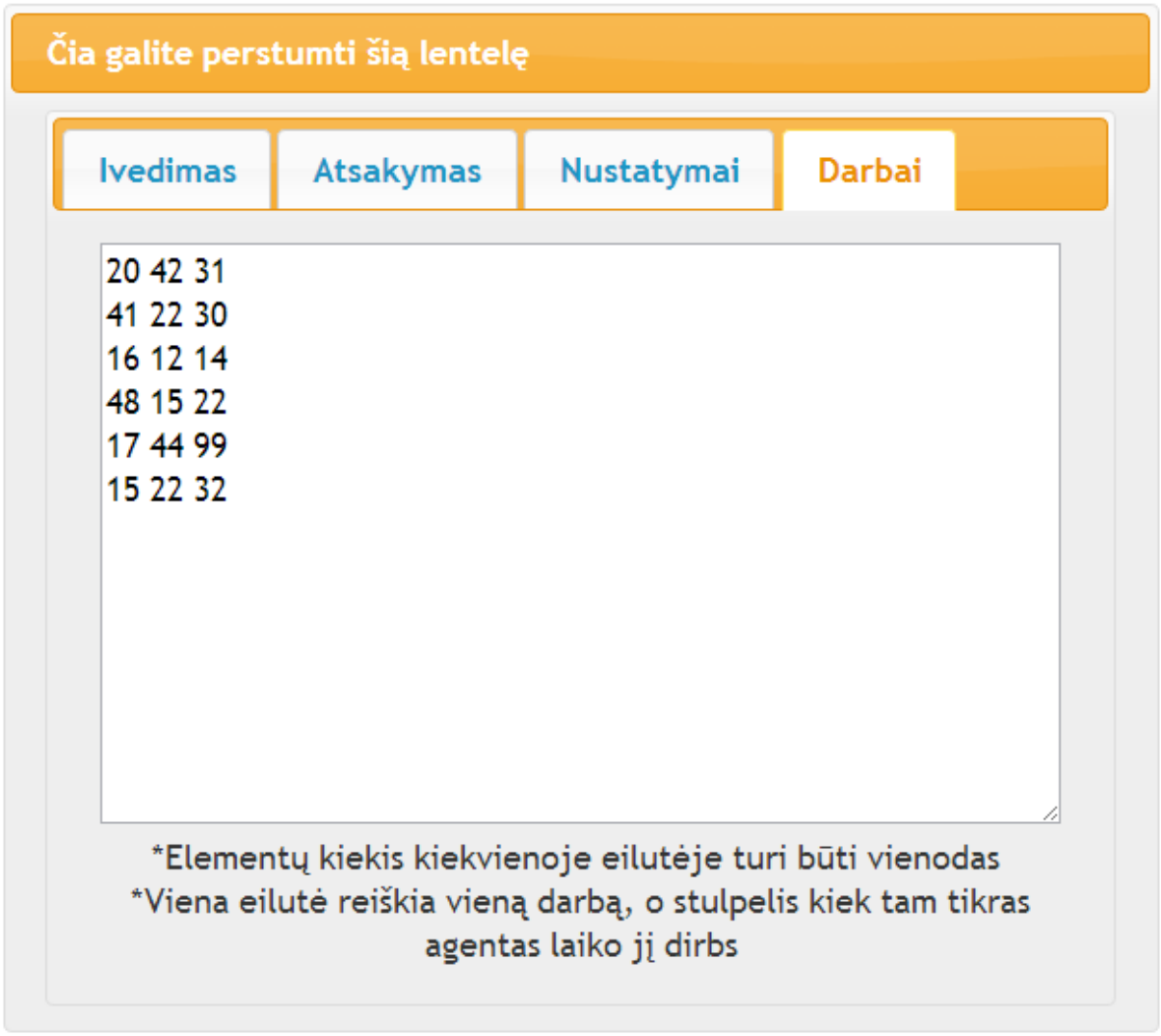
Lentelė 3 Darbų ataskaitos grafikas

Darbų paaiškinimas:

1. Informacijos paieška ir analizė.
2. Aptarimas su darbo vadovu.
3. Programinės įrangos kūrimas.
4. Programinės įrangos testavimas.
5. Darbo aprašymas.

3.2 Galutinis projektas

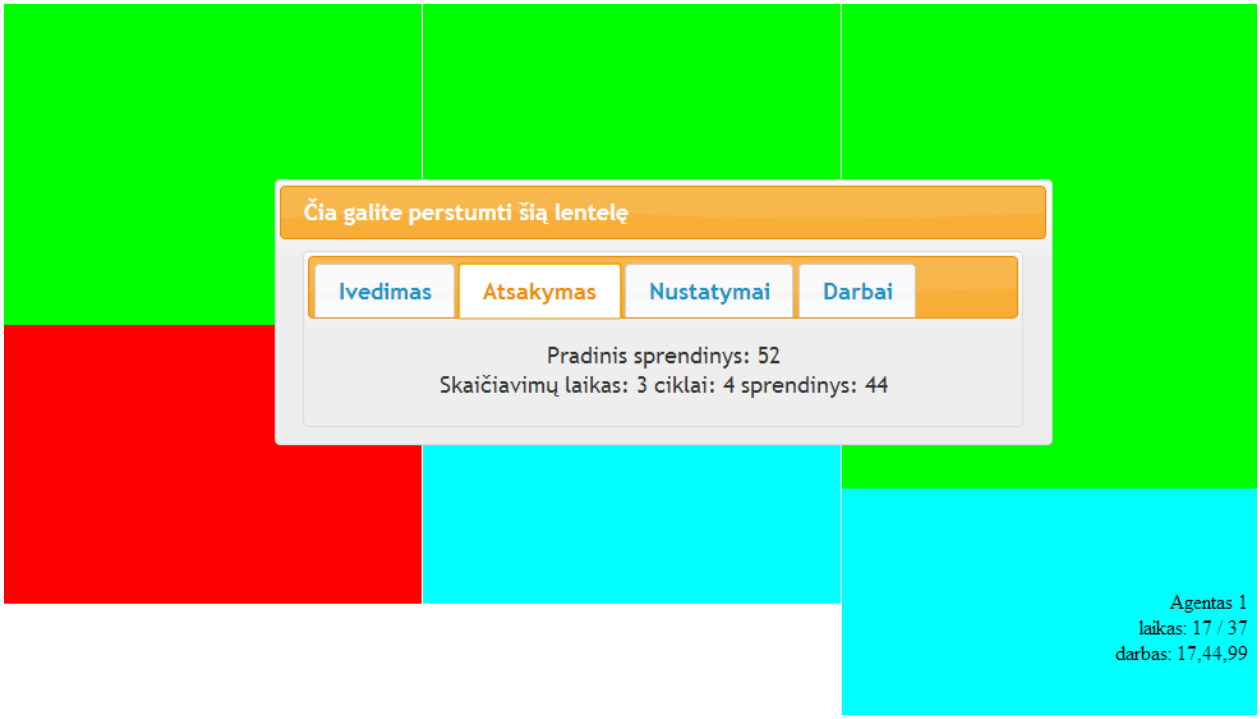
Šiuo metu yra sukurta dirbtinės bičių kolonijos algoritmo realizacija pasirinktam uždaviniui spręsti. Buvo patobulinta pradinio sprendinio paieška, patobulintas dirbtinės bičių kolonijos algoritmo pritaikymas (Pav. 8). Sukurta vartotojo sąsaja pradiniams duomenims įvesti (Pav. 5, Pav. 6) ir gauto sprendinio vizualizacija (Pav. 7). Šią programą galima nesunkiai pertvarkyti tiesiniui skirstymo uždaviniui spręsti.



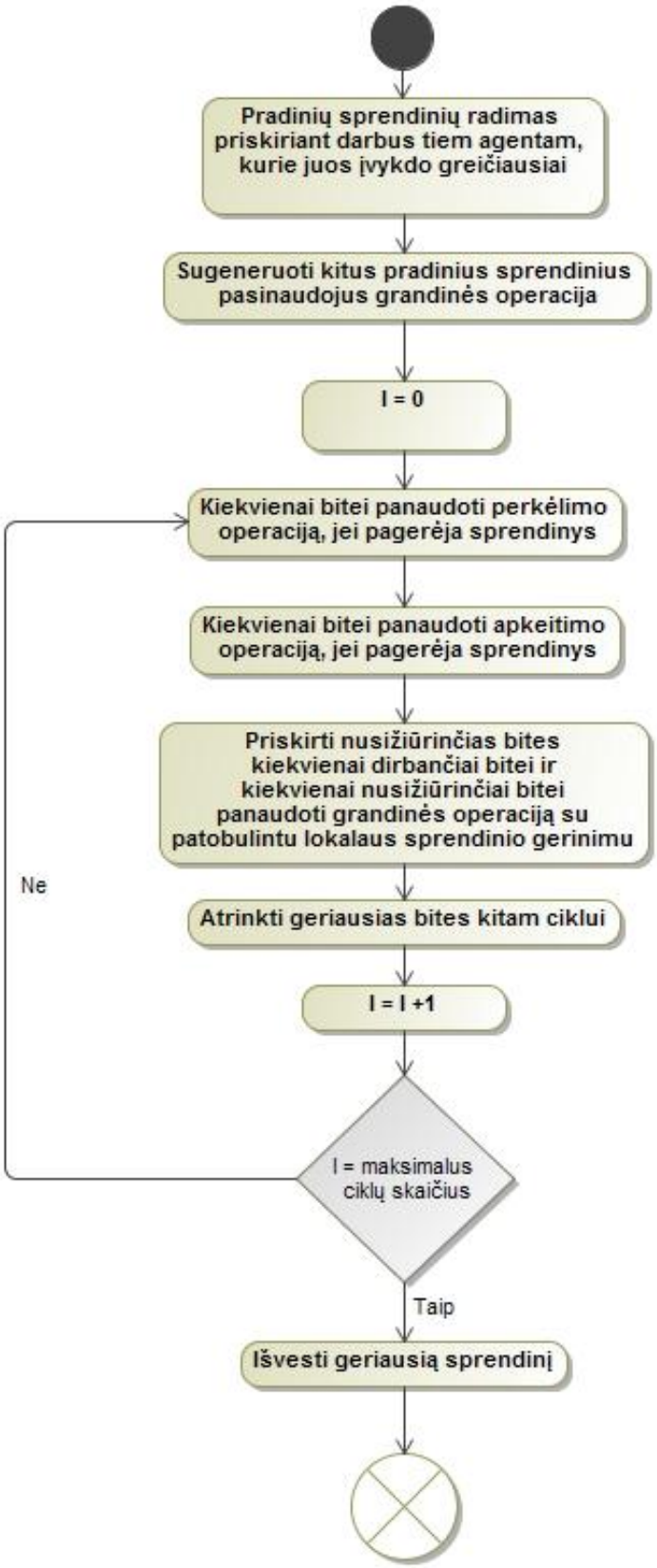
Pav. 5 Darbų sąrašo įvedimas



Pav. 6 Skaičiavimo parametrų įvedimas



Pav. 7 Gauto sprendinio vizualizacija



Pav. 8 Patobulintas dirbtinės bičių kolonijos algorimas

3.3 Problemų sąrašas

Kuriant programą buvo susidurta su šiomis problemomis:

1. Atliekant grandinės operaciją buvo būtina klonuoti bites operacijos pradžioje tam, kad vėliau būtu galima jas grąžinti pakeistas į pagrindinę funkciją. Bitės buvo aprašomos kelių dimensijų masyvais, o javaScript neturi tiesioginio metodo klonuoti visą masyvą, o tik vienos dimensijos indeksuotus narius. Tam, kad išspręsti šią problemą reikėjo pasinaudoti "map" metodu, kuris grąžina naują masyvo elementą vietoj kiekvieno sutikto masyvo elemento. Naują masyvo elementą priskiriant to elemento dimensijos indeksuotam nariams.
2. Darbus skirstant atsitiktinai pradinis sprendinys būdavo per toli nuo siekiamo rezultato, dėl to skaičiavimai užtrukdavo per ilgai dėl bereikalingo grandinės operacijos naudojimo. Šiai problemai išspręsti buvo panaudotas naujas pradinių sprendinių gavimo metodas: randamas vienas sprendinys darbus skirstant tiems agentams, kurie juos atlieka greičiausiai ir panaudojant grandinės operaciją gauti likusius pradinius sprendinius.
3. Sprendžiant šį uždavinį ir naudojant grandinės operaciją, atsitiktinai kilnojant užduotis rezultatai, retai pagerėdavo, todėl reikėjo patobulinti grandinės operaciją taip, kad darbus kilnotų šiek tiek atsižvelgiant į tai kur darbas bus keliamas ir iš kur išimtas.

3.4 Darbo rezultatų analizė

1. Buvo atlikti testavimai su mažais duomenų kiekiais (2-3 agentai ir iki 15 užduočių) tam, kad įsitikinti, jog programa veikia be klaidų.

20	42	31
41	22	30
16	12	14
48	15	22
17	44	99
15	22	32

Lentelė 4 Vienas is uždavinių, kuris buvo sprendžiamas testuojant

Atliekant uždavinio (Lentelė 4) skaičiavimus su parametrais (bitės - 10, nusižiūrinčios bitės - 4, ciklai - 5) buvo gaunami rezultatai (Lentelės 5-7).

20 42 31	48 **15** 22	41 22 **30**
17 44 99	15 **22** 32	16 12 **14**
37	**37**	**44**

Lentelė 5

17 44 99	41 **22** 30	20 42 **31**
15 22 32	48 **15** 22	16 12 **14**
32	**37**	**45**

Lentelė 6

20 42 31	41 **22** 30	16 12 **14**
17 44 99	15 **22** 32	48 15 **22**
37	**44**	**36**

Lentelė 7

Atliekant testavimus su mažos apimties uždaviniais buvo pastebėta, kad gautus sprendinius yra labai sunku ar neįmanoma pagerinti skaičiuojant pačiam (Lentelės 4-7). Pateikto mažos apimties uždavinio (Lentelė 4) gaunami sprendiniai dažniausiai yra optimumas.

2. Buvo sugeneruota tūkstantis užduočių dešimčiai agentų. Vienos užduoties laikas atsitiktinai tarp 5 ir 104 įskaitant galus. Buvo keičiamas darbų skaičius, ciklų skaičius ir bičių skaičius testams atlikti (Lentelė 8). Testavimams atlikti naudota Google Chrome interneto naršyklė.

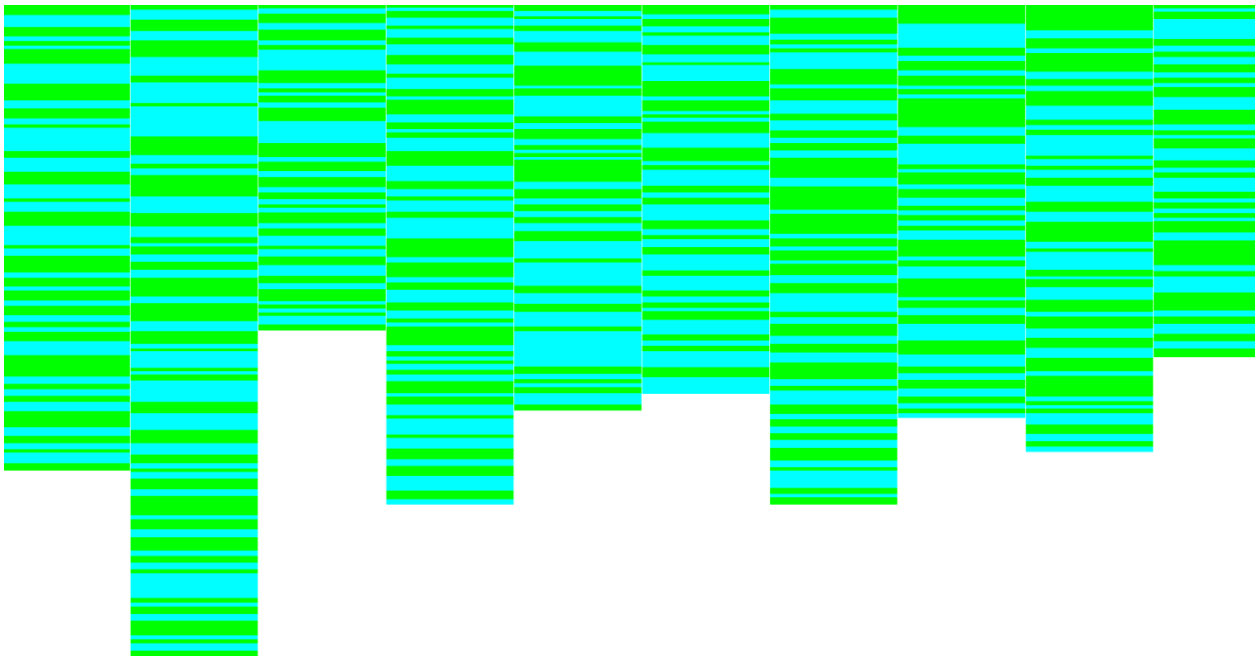
Darbų skaičius	Ciklų skaičius	Dirbančių Bičių skaičius	Nusižiūrinčių bičių skaičius	Pradinis sprendinys	Skaičiavimų laikų intervalas(ms)	Gautų sprendinių intervalas
15	10	2	2	52	1-2	40-52
15	10	10	2	52	11-21	40
15	100	2	2	52	11-14	40-43
100	10	2	2	263	2-6	157-182
100	100	2	2	263	18-35	148-162
100	100	3	2	263	29-49	148-157
100	1000	3	2	263	284-325	148-158
500	10	3	2	1006	8-27	758-772
500	100	3	2	1006	114-144	704-718
500	100	3	10	1006	765-882	702-711
500	100	12	2	1006	556-733	702-711
500	100	12	5	1006	1619-1900	700-709
500	100	20	2	1006	1100-1205	700-712
500	100	40	1	1006	1264-1475	701-713
500	1000	3	2	1006	1212-1310	703-718
500	1000	6	2	1006	3131-3398	702-717
500	1000	12	2	1006	6675-7315	702-708
500	1000	5	5	1006	6255-7951	700-709
500	10000	3	2	1006	8422-9544	701-713
1000	10	3	2	1889	24-40	1585-1666
1000	10	10	2	1889	68-109	1534-1578
1000	100	3	2	1889	259-304	1423-1441
1000	100	10	2	1889	815-1001	1411-1425
1000	100	20	2	1889	1796-2004	1408-1422

1000	1000	3	2	1889	2729-2888	1410-1427
1000	1000	10	2	1889	9279-10026	1404-1417
1000	1000	20	2	1889	20414-22542	1404-1414
1000	2000	20	2	1889	44003-46284	1401-1411
1000	2000	40	1	1889	46073-47262	1403-1406
1000	2000	10	4	1889	45815-49872	1404-1411
1000	10000	3	2	1889	27433-29888	1403-1414
1000	10000	10	2	1889	95385-99148	1402-1411

Lentelė 8 Testavimų rezultatai su atsitiktinai sugeneruotais darbais

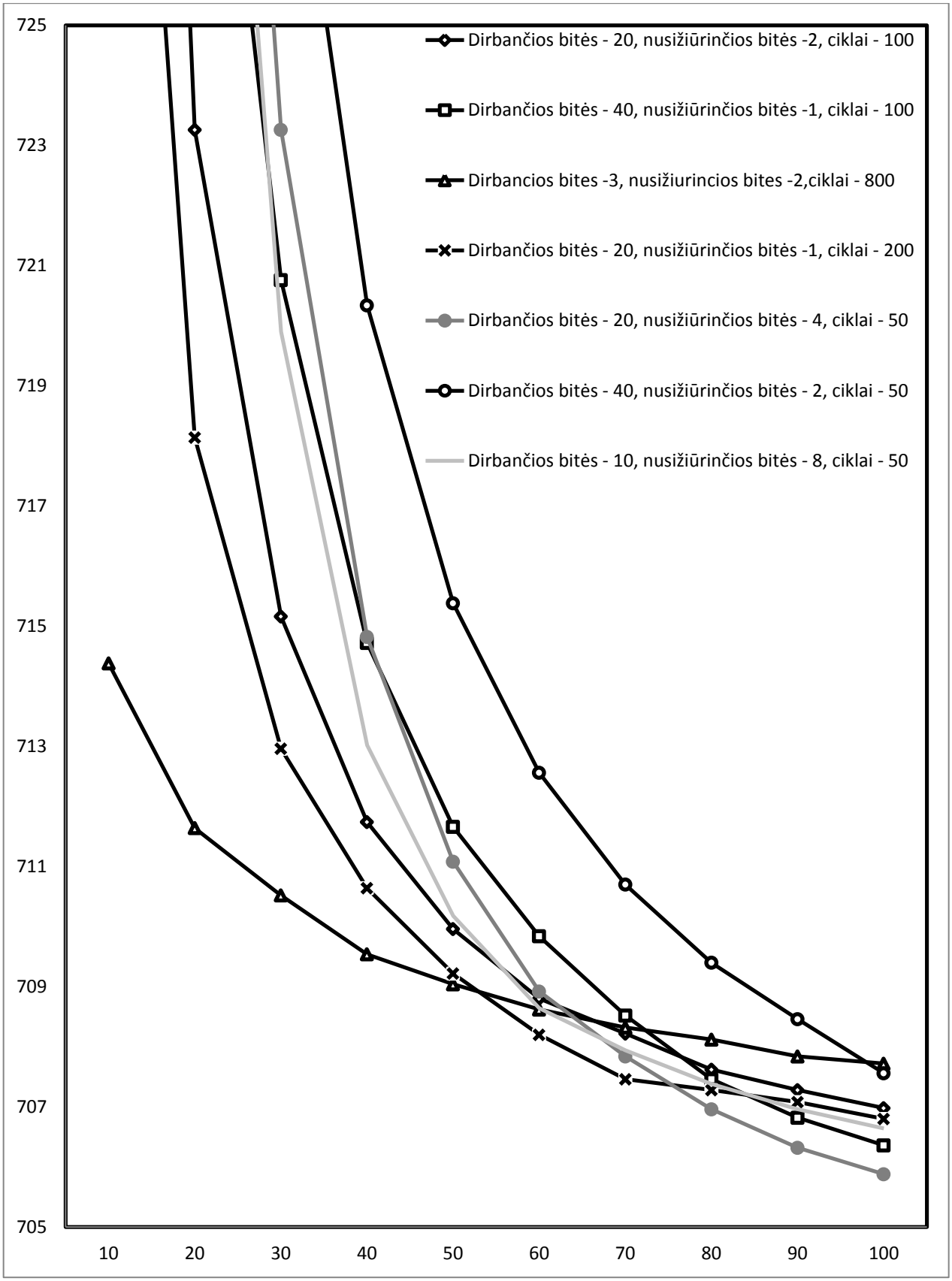
Keičiant parametrus buvo pastebėta, kad didesnis bičių kiekis su mažiau iteracijų gauna geresnius ar tokius pat rezultatus per truputį geresnį laiką nei mažesnis bičių kiekis su daugiau iteracijų. Taip pat testuojant buvo pastebėta, kad didinamas dirbančių bičių kiekis dažniausiai pagerina gaunamų sprendinių intervalą, o didinamas nusižiūrinčių bičių kiekis kartais pagerina geriausią gaunamą sprendinį, bet geriausi rezultatai pasiekiami balansuojant visus šiuos parametrus priklausomai nuo uždavinio apimties. Rekomenduotinas mažiausias nusižiūrinčių bičių kiekis yra 2.

3.



Pav. 9 Pradinis sprendinys su 500 užduočių ir 10 agentų

Su kiekvienu parametrų rinkiniu buvo atlikta po 50 bandymų optimizuojant pradinį sprendinį (Pav. 9) ir užfiksuoti jų geriausi sprendiniai kas apytiksliai vieną dešimtąją jų vykdymo laiko ir suskaičiuotas tų sprendinių vidurkis (Grafikas 1, Lentelė 9). Kiekvienas bandymas užtrukdavo panašiai laiko.



Grafikas 1 Sprendinio, su 500 užduočių (Pav .9), artėjimas link optimumo laiko atžvilgiu su skirtingais parametrais

Laikas (%)	Dirbančios bitės - 20, nusižiūrinčios bitės -2, ciklai - 100	Dirbančios bitės - 40, nusižiūrinčios bitės - 1, ciklai - 100	Dirbančios bites -3, nusižiūrinčios bites - 2,ciklai - 800	Dirbančios bitės - 20, nusižiūrinčios bitės -1, ciklai - 200	Dirbančios bitės - 20, nusižiūrinčios bitės - 4, ciklai - 50	Dirbančios bitės - 40, nusižiūrinčios bitės - 2, ciklai - 50
10	755.86	769.46	714.38	738.14	795.84	808.86
20	723.26	733.36	711.64	718.14	743.38	750.36
30	715.16	720.76	710.52	712.96	723.26	730.28
40	711.74	714.72	709.54	710.64	714.82	720.34
50	709.96	711.66	709.04	709.22	711.08	715.38
60	708.8	709.84	708.62	708.2	708.92	712.56
70	708.22	708.52	708.32	707.46	707.84	710.7
80	707.62	707.46	708.12	707.28	706.96	709.4
90	707.28	706.82	707.84	707.08	706.32	708.46
100	706.98	706.36	707.72	706.8	705.88	707.56

Lentelė 9 Sprendinio, su 500 užduočių (Pav .9), artėjimas link optimumo laiko atžvilgiu su skirtingais parametrais (visus bandymus galima rasti "testavimai.xlsx" faile)

Laiko atžvilgiu didelis ciklų skaičius ir mažas bičių kiekis greitai optimizuoja pradžioje, bet tai yra dėl to, kad pradinis sprendinys paskirsto užduotis taip, kad paskirstymas būtų kuo panašesnis į gaunamo ar optimalaus sprendinio paskirstymą. Todėl didelis ciklų skaičius greitai išmėto užduotis agentams tolygiau, bet taip būna nepakankamai atliekama globali paieška ir sprendinys po mažo laiko tarpo beveik nebeartėja link optimumo.

Geriausi sprendiniai per 700-1100ms laiko tarpą buvo gauti naudojant didesnius nusižiūrinčių bičių kiekius ir mažus ciklų kiekius, vidutinius ciklų kiekius ir didesnius dirbančių bičių kiekius.

Geriausią patobulėjimą per paskutinius 20% skaičiavimų laiko parodė 1 ir 2 vietoje, po bandymų laiko, esantys parametrų rinkiniai (po 1,08 ir 1,1) ir priešpaskutinėje vietoje likęs parametrų rinkinys, kuris pagerėjo (1,84).

3.5 Rekomendacijos

Programai būtų galima:

- pridėti užduočių įvedimą iš .XLS (excel) failo.
- leisti užduotims suteikti vardus ir realizuoti jų paiešką, kad jas būtu lengviau surasti gautame sprendinyje.
- sukurti vizualiciją, kuri leistų efektyvų praktinį panaudojimą.
- sukurti tekstinį gauto sprendinio išvedimą, kurį būtų galima atspausdinti.
- suteikti galimybę vartotojui tęsti gauto sprendinio optimizavimą.

Išvados

1. Susipažinus su dalelių spiečių algoritmais paaiškėjo, kad šie algoritmai naudoja globalius ir lokalius paieškos metodus, dažniausiai taikomi optimizavimo uždaviniams spręsti.
2. Dirbtinės bičių kolonijos algoritmų analizė parodė, jog šie algoritmai yra gana efektyviai taikomi sudėtingiems optimizavimo uždaviniams spręsti, o taip pat yra tinkami spręsti įvairiems skirstymo uždaviniams.
3. Susipažinus su įvairiais skirstymo uždaviniais paaiškėjo, jog jie dažniausiai sutinkami sudarant darbo grafikus, logistikoje, versle, bei kitose srityse. Jie dažniausiai yra sudėtingi, nes negalima nuspėti rezultato, o galimų sprendinių kiekis yra labai didelis.
4. Programai kurti pasirinkta skriptų kalba buvo naudinga, nes galima peržiūrėti gautų masyvų reikšmes naršyklės konsolėje, o JavaScript bibliotekos padeda greitai sukurti vartotojo sąsają.
5. Atlikus testinius skaičiavimus su kontroliniais duomenimis paaiškėjo, jog programa veikia be klaidų, sugeba rasti sprendinius artimus optimumui ar net optimumą.
6. Atlikus testavimų analizę galima teigti, kad skirstymo uždaviniams vien lokalaus optimizavimo neužtenka, taip pat reikia ir globalios paieškos, kurią atlieka „bitės“, nes didinant bičių kiekį buvo gaunami geresni rezultatai, nei didinant ciklų skaičių.
7. Priklausomai nuo uždavinio apimties reikia pasirinkti tinkamus ciklų ir bičių kiekius, kad skaičiavimai būtų efektyvesni laiko atžvilgiu.

Literatūros sąrašas

1. Jason Brownlee, (2011). *Clever Algorithms: Nature-Inspired Programming Recipes.* [interaktyvus] [žiūrėta 2013-04-12]. Prieiga internete: http://www.cleveralgorithms.com/nature-inspired/swarm.html
2. Jason Brownlee, (2011). Clever Algorithms: Nature-Inspired Programming Recipes. [interaktyvus] [žiūrėta 2013-04-12]. Prieiga internete: http://www.cleveralgorithms.com/nature-inspired/swarm/pso.html
3. Jason Brownlee, (2011). Clever Algorithms: Nature-Inspired Programming Recipes. [interaktyvus] [žiūrėta 2013-04-12]. Prieiga internete: http://www.cleveralgorithms.com/nature-inspired/swarm/ant_colony_system.html
4. Jason Brownlee, (2011). Clever Algorithms: Nature-Inspired Programming Recipes. [interaktyvus] [žiūrėta 2013-04-12]. Prieiga internete: http://www.cleveralgorithms.com/nature-inspired/swarm/bfoa.html
5. Eberhart et al, Morgan Kaufmann. *Swarm Intelligence and Ant Colony Optimisation.* [interaktyvus] [žiūrėta 2013-04-12]. Prieiga internete: http://www0.cs.ucl.ac.uk/staff/p.bentley/teaching/L8_Swarms_and_ACO.pdf
6. Adil Baykasoùlu, Lale Özbakır and Pınar Tapkan. *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem.* [interaktyvus] [žiūrėta 2013-04-13]. Prieiga internete: http://natcomp.liacs.nl/SWI/papers/artificial.bee.colony.algorithm/abc.algorithm.and.its.application.to.generalized.assignment.problem.pdf
7. Dervis Karaboga, (2010). *Artificial Bee Colony (ABC) Algorithm*. . [interaktyvus] [žiūrėta 2013-04-13]. Prieiga internete: http://mf.erciyes.edu.tr/abc/
8. Dervis Karaboga, (2010). *Artificial Bee Colony (ABC) Algorithm*. . [interaktyvus] [žiūrėta 2013-04-13]. Prieiga internete: http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm
9. Hemant Nagpure, Rohit Raja. *The Applications Survey on Bee Colony Optimization*. [interaktyvus] [žiūrėta 2013-04-16]. Prieiga internete: http://www.ijcsit.com/docs/Volume%203/vol3issue5/ijcsit2012030539.pdf
10. Eranda Cela. *Assignment Problems.* [interaktyvus] [žiūrėta 2013-04-15]. Prieiga internete: http://www.opt.math.tu-graz.ac.at/~cela/papers/hao.pdf

11. Pavlo A. Krokhmal, Panos M. Pardalos, (2007). *Random Assignment Problems.* [interaktyvus] [žiūrėta 2013-04-16]. Prieiga internete: http://www.engineering.uiowa.edu/~krokhmal/pdf/random-assignment-problems.pdf
12. David W. Pentico, (2005). *Assignment problems: A golden anniversary survey*. [interaktyvus] [žiūrėta 2013-04-16]. Prieiga internete: http://math.tongji.edu.cn/model/docs/pentico.ejor176.pdf
13. Shalini Arora, M.C. Puri, (1997). *A variant of time minimizing assignment problem*. [interaktyvus] [žiūrėta 2013-04-16]. Prieiga internete: http://www.mcpuri.com/papers/tmap_shalini.pdf

Anotacija

Autorius: Linas Matakas

Tema: Dirbtinės bičių kolonijos algoritmai ir jų taikymai skirstymo uždaviniams spręsti.

Šiaulių universitetas 2013

Šiame darbe yra trumpai apžvelgiami dalelių spiečių sistemų algoritmai, skirstymo uždaviniai ir jų formuluotės, bei praktinės interpretacijos, plačiau apžvelgiami ir analizuojami dirbtinių bičių kolonijų algoritmai. Taip pat šiame darbe galima rasti dirbtinių bičių kolonijų algoritmo pritaikymą skirstymo uždaviniams spręsti, bei sukurtos programos skaičiavimo rezultatų analizę.

Summary

Author: Linas Matakas

Subject: Artificial Bee Colony Algorithms and their Application to Assigment Problems.

Šiauliai University, 2013

This paper consists of short descriptions of swarm systems algorithms, assigment problems and longer overview of artificial bee colony algorithms and it‘s analysis. Moreover, you can find an Artificial Bee Colony Algorithm's Application to one of an Assigment Problems and it's computational results analysis.

Priedai

1. CD turinys:

- Kataloge "Programa" yra HTML failas ("ABC.html"), funkcijų failas ("funkcijosABC1.js"), pakopinių stilių failas ("css.css", kuriuose yra dirbtinės bičių kolonijos algoritmo ir vartotojo sąsajos kodas.
- CD šakniniame kataloge yra elektroninės bakalauro darbo aprašymo versijos (rinkmenos „dirbtines biciu kolonijos algoritmai ir ju taikymai skirstymo uzdaviniams spresti.docx“, bei „dirbtines biciu kolonijos algoritmai ir ju taikymai skirstymo uzdaviniams spresti.pdf“) ir testavimo bandymų failas ("testavimai.xlsx").
- Kataloge "Programa" yra užduočių failas "duomenys.txt", kuriame yra 1000 atsitiktinai sugeneruotų užduočių dešimčiai agentų.