



VILNIAUS GEDIMINO
TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

Albinas JARUŠEVIČIUS

**GAMTA GRĮSTŲ METAEURISTINIŲ ALGORITMŲ VAIZDAMS
APDOROTI TYRIMAS**

**INVESTIGATION OF NATURE-INSPIRED METAHEURISTIC
ALGORITHMS FOR IMAGE PROCESSING**

Magistro baigiamasis darbas

Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

Atvirojo kodo sistemų specializacija

Informatikos inžinerijos studijų kryptis

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

(parašas)

Prof. dr. D. Navakauskas

2019 m. _____ mėn. ____ d.

Albinas JARUŠEVIČIUS

**GAMTA GRĮSTŲ METAEURISTINIŲ ALGORITMŲ VAIZDAMS
APDOROTI TYRIMAS**

**INVESTIGATION OF NATURE-INSPIRED METAHEURISTIC
ALGORITHMS FOR IMAGE PROCESSING**

Magistro baigiamasis darbas

Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

Atvirojo kodo sistemų specializacija

Informatikos inžinerijos studijų kryptis

Vadovas

Prof. dr. D. Navakauskas

(pedag. vardas, moksl. laipsnis, vardas, pavardė) (parašas) (data)

Lietuvių kalbos konsultantas

lekt. dr. A. Gaidienė

(pedag. vardas, moksl. laipsnis, vardas, pavardė) (parašas) (data)

Vilniaus Gedimino technikos universitetas Elektronikos fakultetas Elektroninių sistemų katedra		ISBN ISSN Egz. sk. Data-.....-.....
Antrosios pakopos studijų Informacinių elektroninių sistemų programos magistro baigiamasis darbas Pavadinimas Gamta grįstų metaeuristinių algoritmų vaizdams apdoroti tyrimai Autorius Albinas Jaruševičius Vadovas Dalius Navakauskas		
Kalba: lietuvių		
Anotacija Baigiamajo darbo tikslas – atrinkti ir ištirti gamta grįstus metaeuristinius algoritmus ir jų taikymus bei pasiūlyti rekomendacijas. Tam pasiekti atlikama metaeuristinių algoritmų ir jų modifikacijų analitinė literatūros apžvalga, atrenkant perspektyviausius algoritmus. Iš šių algoritmų sukuriamas sistema vaizdams segmentuoti. Šiomis sistemomis atliekamas greیتaveikos, tikslumo ir lygiagretumo tyrimas. Tyrimo metu nustatyta, kad jonvabalių algoritmas sparčiausias, o uodų taikinių paieškos algoritmas tiksliausias iš metaeuristinių algoritmų. Gamta grįstų metaeuristinių algoritmų vaizdams apdoroti tyrimas. Magistro baigiamasis darbas informatikos inžinerijos laipsniui. Vilniaus Gedimino technikos universitetas. Vilnius, 2019, 56 p., 4 iliustr., 9 lent., 69 bibl., 2 priedai.		
Prasminiai žodžiai: metaeuristinių algoritmų tyrimas, skruzdžių kolonijų algoritmas, jonvabalių algoritmas, uodų taikinių paieškos algoritmas, medicininių vaizdų segmentavimas.		

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Electronic Systems

ISBN ISSN
Copies No.
Date-.....-.....

Master Degree Studies **Information Electronics Systems** study programme Master Graduation Thesis

Title **Investigation of Nature-Inspired Metaheuristic Algorithms for Image Processing**

Author **Albinas Jaruševičius**

Academic supervisor **Dalius Navakas**

Thesis language: Lithuanian

Annotation

The aim of the final thesis is to select and investigate nature-based metaheuristic algorithms and their applications and to propose recommendations. To achieve this, an analytical review of literature on metaheuristic algorithms and their modifications was performed, selecting the most promising algorithms. These algorithms are used to create systems for segmenting images. These systems are used for investigation to test speed, accuracy and parallelism of algorithms. The study found that firefly algorithm is the fastest, and the mosquito host seeking algorithm is the most accurate of metaheuristic algorithms.

Investigation of Nature-Inspired Metaheuristic Algorithms for Image Processing. Master's Thesis for Informatics Engineering. Vilnius Gediminas Technical University. Vilnius, 2019, 56p., 4 pictures, 9 tables, 69 bibliographical entries.

Keywords: Metaheuristic algorithms investigation, ant colony algorithm, firefly algorithm, mosquito host seeking algorithm, segmentation of medical images.

TURINYS

Ivadas	10
Darbo aktualumas ir tikslas	10
Darbo uždaviniai	10
Naudoti tyrimo ir analizės metodai	10
Darbo naujumas ir praktinė nauda.....	11
Darbo struktūra.....	11
1. Metaheuristinių algoritimų analitinė literatūros apžvalga	12
1.1. Skruzdžių kolonijų algoritmai	13
1.1.1. MAX-MIN skruzdžių sistema.....	16
1.1.2. Elitinė skruzdžių kolonija	19
1.2. Jonvabalių algoritmai	21
1.3. Uodų taikinių paieškos algoritmas	25
1.4. Skyriaus apibendrinimas	27
2. Parinktų metodų veikimų principai ir pritaikymas	28
2.1. Algoritimų veikimo principai	28
2.1.1. Skruzdžių kolonijos optimizavimo veikimo principas.....	28
2.1.2. Jonvabalių veikimo principas	29
2.1.3. Uodų taikinių paieškos algoritmas.....	31
2.2. Algoritimų veikimo skirtumai	31
2.3. Algoritimų pritaikymas	33
2.3.1. Atvirojo kodo licencija	33
2.3.2. Skruzdžių kolonijų algoritmo pritaikymas.....	34
2.3.3. Jonvabalių algoritmo pritaikymas.....	36
2.3.4. Uodų taikinių paieškos algoritmo pritaikymas	37
2.4. Skyriaus apibendrinimas	37
3. Duomenų apdorojimo metodika ir vaizdų atranka	39
3.1. Biomedicininį vaizdų atranka.....	39
3.2. Vaizdų apdorojimas.....	40
3.3. Tyrimo metodika	41
3.3.1. Tikslumo vertinimas	41
3.3.2. Greitaveikos ir lygiagretumo vertinimas	44
3.4. Skyriaus apibendrinimas	44
4. Metodų tyrimo rezultatai	46
4.1. Tyrimo rezultatai	46
4.1.1. Tikslumo vertinimo rezultatai.....	46
4.1.2. Greitaveikos ir lygiagretumo vertinimo rezultatai	50
4.2. Metodų rekomendacijos	51
4.3. Skyriaus apibendrinimas	52
Išvados	53
Literatūra	54
PRIEDAI	58
A priedas. Tyrimo rezultatų duomenys	58
B priedas. Pranešimo XXII-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga...	65

ŽYMENYS IR SANTRUMPOS

Žymenys

α	slopinimo koeficientas
β	slopinimo koeficientas
$D_E(q, p)$	Euklidinis atstumas
FN	taškai, kurie nepažymėti rezultate
FP	taškai, kurių neturėtų būti rezultate
$f(x)$	jonvabalių tikslo funkcija
$I(r)$	šviesos intensyvumo funkcija
p	feromono išlaikymo vertė
$P(t)$	uodų tarpusavio traukos funkcija
$Q(t)$	uodų bendradarbiavimo funkcija
TN	teisingai atmesti taškai
TP	teisingai segmentuoti taškai
$\tau_{i,j}$	skruzdės vieta
$\tau(t)$	feromono atnaujinimo funkcija
$u(t)$	uodų ir taikinių atstumų funkcija

Santrumpos

<i>ABC</i> (angl. <i>Artificial Bee Colony</i>)	dirbtinių bičių kolonija
<i>ACO</i> (angl. <i>Ant Colony Optimization</i>)	skruzdžių kolonijos optimizavimas
<i>AIS</i> (angl. <i>Artificial Immune System</i>)	dirbtinė imuninė sistema
<i>BSD</i> (angl. <i>Berkeley Software Distribution</i>)	Berklio programinės įrangos platinimo licencija
<i>CEP</i> (angl. <i>Classification Error Percentage</i>)	klasifikavimo klaidų procentas
<i>FA</i> (angl. <i>Firefly Algorithm</i>)	jonvabalių algoritmas
<i>FSSP</i> (angl. <i>Flow shop scheduling problem</i>)	srauto darbų planavimo problema
<i>GBAS</i> (angl. <i>Graph based ant system</i>)	grafais grįsta skruzdžių sistema
<i>GBS</i> (angl. <i>Global best solution</i>)	visuotinai geriausias sprendinys
<i>GVF</i> (angl. <i>Gradient Vector Flow Snake</i>)	gradiento vektorių srauto kontūrų metodas
<i>JSP</i> (angl. <i>Job Sheduling Problem</i>)	darbo planavimo problema
<i>MHSA</i> (angl. <i>Mosquito Host Seeking Algorithm</i>)	uodų taikinių paieškos algoritmas
<i>MMAS</i> (angl. <i>MAX-MIN Ant System</i>)	MAX-MIN skruzdžių sistema
<i>PDF</i> (angl. <i>Probability Density Function</i>)	tikimybių tankumo funkcija
<i>PSO</i> (angl. <i>Particle Swarm Optimization</i>)	dalelių spiečių optimizavimas
<i>PTSP</i> (angl. <i>Probablistic TSP</i>)	stochastinio „keliaujančio prekeivio“ problema
<i>QAP</i> (angl. <i>Quadratic Assignment Problem</i>)	kvadratinio priskyrimo problema
<i>RDE</i> (angl. <i>Relative Distance Error</i>)	santykinė nuotolio paklaida
<i>ROC</i> (angl. <i>Receiver Operating Characteristic</i>)	imtuvo veikimo charakteristika
<i>SA</i> (angl. <i>Simulated Annealing</i>)	imituojamas atkaitinimas
<i>SPEA</i> (angl. <i>Strength Pareto Evolutionary Algorithm</i>)	stiprusis Pareto evoliucinis algoritmas
<i>SVM</i> (angl. <i>Support Vector Machines</i>)	atraminių vektorių mašina
<i>TSP</i> (angl. <i>Traveling Salesman Problem</i>)	„keliaujančio prekeivio“ problema
<i>WBC</i> (angl. <i>White Blood Cell</i>)	baltasis kraujo kūnelis

IVADAS

Darbo aktualumas ir tikslas

Žmonių aplinkoje ypač daug informacijos, su kuria ne visada pavyksta susitvarkyti. Bandymas spręsti šią problemą skatino žmones išrasti vis sudėtingesnes priemones ir optimizavimo metodus informacijai apdoroti. Dar palyginti neseniai tai buvo daroma lėtais nuosekliais algoritmais, kurie sėkmingai sprendavo santykinai sudėtingus optimizavimo uždavinius. Ieškodami veiksmingesnių optimizavimo sprendimų informatikos inžinerijos mokslininkai pradėjo analizuoti gamtą, kuri juos įkvėpė sukurti populiacija grįstus algoritmus. Šio tipo algoritmai leidžia rasti pakankamai gerą sprendinį aukojant greitaveiką ar tikslumą ypač, kai ribojamas skaičiavimų kiekis. Tarp daugelio metaeuristinių algoritmų paminėtini šie: genetinis algoritmas (Forrest, 1993), skruzdžių kolonijos optimizavimas (Dorigo *et al.*, 1996), dalelių spiečių optimizavimas (Blackwell *et al.*, 2008), jonvabalių algoritmas (Yang, 2010), uodų taikinių paieškos algoritmas (Feng *et al.*, 2013) ir kt. Šie metaeuristiniai algoritmai dažniausiai lyginami „keliaujančio pardavėjo“ pardavėjo problemos rezultatais, bet šios problemos geri rezultatai ne visuomet reiškia gerus rezultatus realiose problemose tokiose kaip: duomenų klasifikavimas, daugelio industrinių tikslų optimizavimas, semantiniai tinklai, robotika, vaizdų apdorojimas. Vis gi, nors visi minėti algoritmai, išskyrus uodų taikinių paieškos algoritmą, yra pritaikyti vaizdų segmentavimui, jie retai buvo išbandyti medicininių vaizdų segmentavimui. Taigi šio **darbo tikslas** – atrinkti ir ištirti gamta grįstus metaeuristinius algoritmus ir jų taikymo biomediciniams vaizdams apdoroti galimybes.

Darbo uždaviniai

1. Atlikti metaeuristinių metodų analitinę literatūros analizę, siekiant atrinkti perspektyviausius algoritmus.
2. Sudaryti atrinktų metaeuristinių metodų veikimo principus, atskleidžiančius detaliuosius algoritmus ir jų pagrindu sukurti sistemą vaizdams segmentuoti.
3. Parengti imitacinių tyrimų metodiką, ištirti vaizdų segmentavimą metaeuristiniais algoritmais bei parengti jų taikymo rekomendacijas.

Naudoti tyrimo ir analizės metodai

Suformuluotiems uždaviniams darbe yra naudojami skirtingi klasifikatoriai, kurie paremti feromonų, šviesos intensyvumo, uodų elgesio, aktyvaus kontūro ir spredimo plokštumų atskirimo modeliais. Taip pat jonvabalių algoritmo klasifikatoriui pritaikyti naudojamas Otsu metodo kryžminimas.

Vaizdo pirminei analizei naudojamas gradientas x ir y ašimis bei vaizdo keitimas į pilkus pustonius. Tyrimo analizei atlikti naudojami statistiniai duomenys tokie kaip vidurkis ir standartinis nuokrypis bei naudojama ROC erdvės analizė.

Darbo naujumas ir praktinė nauda

Baigiamojo darbo *naujumas*:

1. Pirmą kartą MHSA pritaikytas segmentavimo problemai spręsti.
2. Atliktas baltųjų kraujų kūnelių segmentavimo eksperimentinis tyrimas ir atskleisti FA, ACO, MHSA algoritmų privalumai ir trūkumai sprendžiant vaizdų segmentavimo uždavinį.
3. Parengtos metaheuristinių algoritmų ir tyrimo metodikos rekomendacijos, įgalinančios tolimesnius tyrimus.

Šio darbo *praktinė nauda* – išgautos taikymui svarbios žinios apie metaheuristinių algoritmų savybes:

- FA tinkamai apima kontūru didelį kiekį baltojo kraujo kūnelio taškų, bet netinkamai į šį kontūrą įtraukia foną,
- MHSA dažniausiai gerai atskiria foną, bet neįtraukia pakankamai taškų į baltojo kraujo kūnelio kontūrą;
- FA spartesnis nei SVM, priskiriamas apmokamų algoritmų klasei.

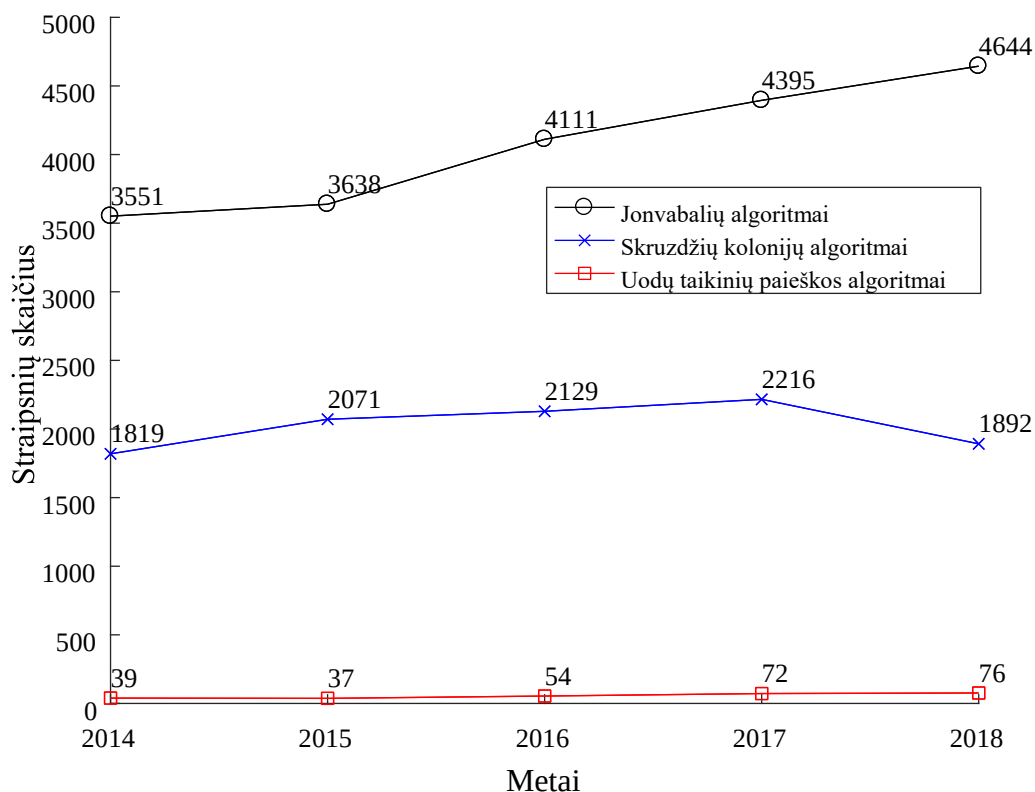
Darbo struktūra

Darbą sudaro keturi skyriai. Pirmame skyriuje atlikta analitinė literatūros apžvalga, išskiriant geriausius algoritmus. Antrame skyriuje aptarti išrinkti algoritmai, jų veikimo principai, skirtumai ir jų pritaikymas vaizdų apdorojimui. Trečiame skyriuje apžvelgtas šabloninis duomenų rinkinys, iš kurio sudarytas šiame darbe naudojamas duomenų rinkinys. Taip pat skyriuje aptartos su mediciniais vaizdais atliekamos operacijos bei šio tyrimo metodika. Paskutiniame skyriuje apžvelgti rezultatai ir pateikiamos algoritmų rekomendacijų. Galiausiai pateiktas darbo apibendrinimas išvadose.

1. METAEURISTINIŲ ALGORITMŲ ANALITINĖ LITERATŪROS APŽVALGA

Metaeuristiniai algoritmai nėra nauji. Pirmieji algoritmai pasiūlyti apie XX amžiaus vidurį. Šiame darbe analizuojama skruzdžių kolonijos algoritmų šeima pirmą kartą pasiūlyta Dorigo 1993 metais (Dorigo *et al.*, 1996) yra gerai ištirtinėta ir pritaikyta daugeliui problemų. Tuo tarpu jonvabalių algoritmų šeima X. S. Yang buvo pasiūlyta 2007 metais, nors jų veikimo efektyvumas yra įrodytas teoriškai, tik dabar mokslininkai stengiasi šiuos algoritmus pritaikyti konkrečioms problemoms spręsti (Sarbazfard *et al.*, 2017). Galiausiai uodų taikinių paieškos algoritmai yra vieni iš naujausių algoritmų, o jų veikimo efektyvumas įrodytas tik 2013 metas (Feng *et al.*, 2013). Šiuo metu šie algoritmai yra pritaikyti tik „keliaujančio prekeivio“ uždaviniui spręsti.

Šiame skyriuje apžvelgti algoritmai yra įvairūs ir vis dar aktualūs. Apie tai liudija iš „Web of Science“ duomenų bazės pagal raktažodžius atrinktų publikacijų skaičiaus priklausomybė nuo jų išleidimo metų (žr. 1 pav.). Matyti, kad visų trijų tipų algoritmai yra aktyviai tyrinėjami. Kadangi uodų taikinių paieškos algoritmas yra pakankamai naujas, su juo susijusių mokslinių darbų yra žymiai mažiau.



1 pav. Atrinktų pagal raktažodžius „Web of Science“ duomenų bazės straipsnių skaičius.

Paieškos rinkiniai: a) ant colony algorithm OR ACA OR ant colony optimization OR ACO; b) firefly algorithm OR FA OR firefly optimization algorithm; c) mosquito algorithm OR mosquito host seeking algorithm OR mhsa OR mosquito optimization

1.1. Skruzdžių kolonijų algoritmai

ACO kaip ir visi metaeuristiniai algoritmai ieško pakankamai gero uždavinio sprendinio, balansuojant greitaveiką ir tikslumą. Šis algoritmas pirmą kartą pasiūlytas 1993 metais, todėl dauguma klausimų, tokių kaip ir kodėl algoritmas veikia jau yra išspręsti ir įrodyti (Dorigo *et al.*, 2005). Naujesni straipsniai siekia gauti geresnių rezultatų negu standartinis skruzdžių algoritmas modifikuojant feromono atnaujinimą (Stützle *et al.*, 2000). Taip pat tiriami nauji algoritmo pritaikymai sudėtingoms problemoms (Dorigo *et al.*, 2006), pavyzdžiui, daugelio tikslų problemoms, dinamiui duomenų manipuliavimui ir pan.

Pirmoji nagrinėjama teorinė problema buvo susijusi su algoritmo konvergavimu: ar konkretus skruzdžių kolonijos algoritmas suras optimalų sprendimą, kai bus suteikta pakankamai išteklių? Skruzdžių kolonijų algoritmai remiasi atsitiktinės paieškos procedūra, kurioje feromono atnaujinimas gali užkirsti kelią jiems visuomet pasiekti optimalų sprendimą, todėl reikia įvertinti du galimus konvergavimo tipus: vertės konvergavimą ir sprendimo konvergavimą. Nagrinėdami vertės konvergavimą, reikia įvertinti tikimybę, kad algoritmas generuos optimalų sprendimą bent kartą. Priešingai, nagrinėjant sprendimo konvergavimą, reikia įvertinti tikimybę, kad algoritmas pasiekia tokį patį optimalų sprendinį generuojančią būseną. Doerner *et al.* (2004) pateikė vertės konvergavimo įrodymus. Jie įrodė su tikimybe $1 - \varepsilon$ konvergavimą link optimalaus sprendinio ir apskritai bet kokiai problemai optimalų sprendinį GBAS. Nepaisant teorinės svarbos, pagrindinis šio darbo trūkumas tas, kad grafais grįsta skruzdžių sistema skiriasi viekimo principu nuo skruzdžių kolonijos algoritmo ir todėl neatskleistas šio algoritmo veikimo principas. Dėl to teorinis darbas buvo nukreiptas į skruzdžių kolonijų algoritmų klasę, kurie yra praktiškai įrodyti kaip geriausi. Ypač gerai pasirodė tokios kolonijos, kuriuose priskirta minimali riba visiems feromonų dydžiams.

Skruzdžių kolonijos pasižymi lygiagretumu, nes didelė dalis sprendimo erdvės yra ištiriama ir įvertinama dideliu skruzdžių skaičiumi. Skruzdės tarpusavyje bendrauja pasitelkiant feromonus, kurių vertės nustatomos tikslo funkcija. Tai reiškia, kad geri sprendimai dėl feromonų veikimo pritraukia kitas skruzdes apieškoti kaimyninę erdvę siekiant pagerinti skruzdės sprendinį. Įprastai skruzdžių kolonijos algoritmu problema sprendžiama atliekant du žingsnius:

- skruzdės formuoja sprendinius pagal feromonų modelį ir atsitiktinį judėjimą apieškant sprendinių erdvę;
- skruzdžių suformuoti sprendiniai naudojami modifikuojant feromonų vertes modelyje taip, kad kita iteracija skruzdės judėtų link geresnės kokybės sprendinių (Dorigo *et al.*, 2005).

Vis dėlto šis algoritmas veikia tik tada, kai galima konstruoti sprendinį dalimis, todėl sprendžiant problemą šiuo algoritmu reikia atsižvelgti į tai, kad problema būtų skaldoma į dalis. Algoritmą

sudėtinga analizuoti, nes galimos atsitiktinių sprendinių sekos, kai geresni sprendiniai nerandami. Kadangi algoritmo kokybė ypač jautri sprendžiamam uždaviniui, sudėtinga spręsti iš anksto ar algoritmas bus tinkamas uždaviniui ar ne. Iš to taip pat kyla dilema, kuomet neaišku kada sustabdyti algoritmą, nors tai dažnai išsprendžiama tiesiog nustatant maksimalų iteracijų skaičių (Dorigo *et al.*, 2005).

Skrudžių kolonijos optimizavimas daug kur pritaikytas, pvz.: sprendžiant „keliaujančio prekeivio“, priskyrimo ar planavimo problemas (Dorigo *et al.*, 2005). Skrudžių kolonijos gali išspręsti kombinatorikos uždavinius. Norint taikyti algoritmą kombinatorinei problemai, reikia apibrėžti:

1. Tinkamą grafų atvaizdavimą.
2. Teigiamą grįžtamojo ryšio procesą (tikslo funkciją).
3. Euristiką, kuri konstruktyviai apibrėžia sprendimus.
4. Ribų ar pasitenkinimo metodą (pavyzdžiui tabu sąrašą).

Tai atlikta dviem gerai žinomoms kombinatorinio optimizavimo problemoms – kvadratinio priskyrimo ir darbų planavimo, kiekvieną kartą gaunant pritaikytą skrudžių sistemos versiją, kuri galėtų veiksmingai tvarkyti problemą. Sunkiausios užduotys, su kuriomis susiduriama taikant skrudžių kolonijas, yra suformuoti tinkamą grafinį vaizdą sprendžiamai problemai ir tikslo funkciją, kuri privestu prie geresnio sprendinio.

Skrudžių kolonijos buvo pritaikytos daugeliui spęstų problemų, kurias galima priskirti vienai iš šių kategorijų: maršruto nustatymo problemoms; priskyrimo problemoms, kai elementų (objektų, veiklų ir pan.) rinkinys turi būti priskirtas tam tikram išteklių skaičiui (vietoms, agentams ir pan.), atsižvelgiant į tam tikrus apribojimus; planavimo problemoms, kurios plačiaja prasme yra susijusios su ribotų išteklių paskirstymu užduotims laikui bėgant; grupavimo problemoms, kai problemos sprendimas laikomas turimų elementų pogrupio pasirinkimu. Taip pat skrudžių kolonijos buvo sėkmingai pritaikytos ir kitoms problemoms, iškylančioms tokiose srityse, kaip mašinų mokymasis (Dorigo *et al.*, 2006).

Skrudžių kolonijų algoritmai parodė, kad yra labai veiksmingi telekomunikacijų tinklų problemose, kai sistemos savybės, tokios kaip sąsajų naudojimo sąnaudos arba mazgų prieinamumas, laikui bėgant kinta. Skrudžių kolonijų algoritmai pirmiausia buvo taikomi maršrutizavimo problemoms grandinių komutaciniuose tinkluose (Schoonderwoerd *et al.*, 1996), kurie tuo metu parodė geriausią rezultatą iš lygintų algoritmų, o vėliau ir paketiniams tinklams tokiems, kaip vietiniai tinklai ar internetas (Dorigo *et al.*, 1998). Gerai žinomas pavyzdys yra skrudžių tinklas (angl. *Ant Net*) (Dorigo *et al.*, 1998). Skrudžių tinklas buvo plačiai išbandytas, jį taikant įvairiuose tinkluose ir skirtinguose maršruto parinkimo modeliuose. Lyginant su naujausiais maršrutizavimo algoritmais gauta, kad daugumoje aptariamų situacijų skrudžių tinklas lankstumu ir efektyvumu pranoksta savo

konkurentus. Skruzdukių kolonijų veikimo principais buvo sukurta keletas kitų maršrutizavimo algoritmų, kurie didina našumą laidiniame tinkle (Sim *et al.*, 2004). Neseniai buvo įrodyta, kad ACO algoritmas, sukurtas sudėtingoms mobiliųjų tinklų klasėms, yra konkurencingas su naujais maršrutizavimo algoritmais (Ducatelle *et al.*, 2005), (Caro *et al.*, 2005), ir tuo pačiu metu siūlo geresnį praplečiamumą.

Akademinių problemų sprendimo sėkmė atkreipė įmonių dėmesį ir jos pradėjo naudoti skruzdukių kolonijų algoritmus realioms problemoms spręsti. Tarp pirmųjų, naudojančių skruzdukių kolonijų metaheuristinius algoritmus yra „EuroBios“ kompanija. Ji taiko ACO daugeliui skirtingų planavimo problemų, pvz., nuolatinės srauto parduotuvės problemos su baigtiniais ištekliais (Dorigo *et al.*, 2005). Modeliuojami uždaviniai apima įvairius realaus pasaulio apribojimus, tokius kaip sąrankos laikai, pajėgumų apribojimai, išteklių suderinamumas ir techninės priežiūros kalendoriai. „AntOptima“ dar viena kompanija, kuri atliko ir vis dar vaidina labai svarbų vaidmenį skatinant skruzdukių kolonijų taikymą. „AntOptima“ mokslininkai sukūrė įrankių rinkinį, skirtą transporto priemonių maršruto problemų sprendimui, kurių algoritmai pagrįsti ACO (Dorigo *et al.*, 2005).

Vienas iš „AntOptima“ darbų pagrindu veikiančios programinės įrangos pavyzdžių yra „DyvOil“, skirtos šildymo alyvos paskirstymui valdyti ir optimizuoti planuojant sunkvežimių darbą. Šią įrangą pirmą kartą naudojo įmonė „Pina Petroli“ Šveicarijoje. Kitas programinės įrangos pavyzdys „AntRoute“, skirtas šimtų sunkvežimių valdymui bendrovių tokių, kaip Šveicarijos prekybos centrų „Migros“ arba pagrindinių Italijos makaronų gamintojos „Barilla“ maršrutams. Dar vieną transporto priemonės maršruto parinkimo programą „BiosGroup“ sukūrė Prancūzijos kompanijai „Air Liquide“. Kitos įdomios skruzdukių kolonijų taikomosios programos: gamybos tvarkaraščių sudarymas aliuminio liejimo centre (Gravel *et al.*, 2002), apkrovos linijos balansavimas su keliais tikslais ir apribojimais užduotyje (Bautista *et al.*, 2002).

Taip pat 2008 m. skruzdukių kolonijos optimizavimas buvo pritaikytas aptikti kraštams. Siūlomas ACO algoritmas straipsnyje (Jing *et al.*, 2008) buvo pritaikytas krašto aptikimui ir norint palyginti skirtingas feromonų matricos atnaujinimo funkcijas. Feromonų matrica atspindi krašto informaciją, pateiktą kiekvieno vaizdo taško padėtyje, atsižvelgiant į geriausių skruzdukių judesius, kurie atnaujinami skruzdekms judant vaizdu. Be to, šių skruzdukių judesius lemia vietinio vaizdo intensyvumo verčių svyravimai naudojant kaimyninius taškus. Darbe pateikiami eksperimentiniai rezultatai, parodantys aukščiausią siūlomo metodo efektyvumą (Jing *et al.*, 2008), todėl būtų idomu pritaikyti metodą medicininiam vaizdui segmentuoti.

Stochastinių optimizavimo problemų atveju kai kurie kintamieji yra tikimybiniai. Be tinklo maršrutizavimo problemų, dėl kurių pagrindinis dėmesys buvo skiriamas jų dinaminiam pobūdžiui, stochastinio keliaujančio prekeivio problema buvo pirmoji stochastinė problema, su kuria susidūrė skruzdukių kolonijų algoritmai. PTSP atveju kiekvienas miestas turi tam tikrą aplankymo tikimybę, o

tikslas yra surasti tikėtinai minimalaus ilgio kelią aplankant visus miestus. Pirmąjį šią problemą sprendžiantį skruzdžių kolonijos algoritmą pasiūlė Bianchi *et al.* (2002).

Kelis uždavinius dažnai galima išspręsti juos rūšiuojant arba pasveriant pagal jų santykinę svarbą. Dviejų skruzdžių kolonijų algoritme, skirtame transporto priemonės maršruto problemai spręsti, tai atliekama taikant laiko suvaržymus (Gambardella *et al.*, 1999). MAX-MIN skruzdžių kolonijoje, kuri sprendžia keleto tikslų srauto darbų planavimo problemą, kuri yra specialus darbų planavimo problemos atvejis (Doerner *et al.*, 2003), daugiafunkcinio optimizavimo problema sprendžiama rūšiuojant tikslus. Doerner *et al.* (2003) taikė skruzdžių koloniją kelių tikslų transporto problemai ir tikslus suderino taikant svertinę sumą. Antrą vertus, jei prioritetų ar svorių negalima suteikti *a priori*, tikslinga ieškoti tokio sprendinio, kurio negalima pagerinti neaukojant vieno iš tikslo verčių. Pirmąjį skruzdžių kolonijų algoritmą nereglamentuojamiems sprendimams rasti pasiūlė Iredi *et al.* (2001) ir jį taikė dviejų tikslų planavimo problemai spręsti. Kiti taikymai apima vertybinių popierių portfelio optimizavimą (Doerner *et al.*, 2006) ir jau minėtą kvadratinio paskirstymo problemą (Ibanzez *et al.*, 2004).

ACO teorijos vystymas prasidėjo nuo skruzdžių sistemos (angl. *Ant System*, AS). Algoritmas paprastai atnaujindavo feromonus pasitelkdamas iteracijoje visus skruzdžių sprendinius. Vėliau algoritmas buvo patobulintas pakeičiant feromonų vertes tik pagal geriausio sprendinio skruzdę. Taip pat feromono vertė turi tam tikras ribas, kurios nustatomos pagal sprendžiamą uždavinį. Šis patobulintas algoritmas literatūroje vadinamas MAX-MIN skruzdžių sistema. Be to buvo pasiūlyta ACO versija, kuri praplečia MAX-MIN skruzdžių sistemą, naudojant lokalų feromonų atnaujinimą bei papildomai mažinant praėjusios iteracijos feromonų įtaką pasitelkiant slopinimą. Taip pat pridėta papildoma taisyklė, kuri judina skruzdes atsitiktinai (Dorigo *et al.*, 2006).

1.1.1. MAX-MIN skruzdžių sistema

MAX-MIN skruzdžių sistema (angl. *MAX-MIN ant system*, *MMAS*) remiasi paieškos istorija, leidžiant tik geriausius sprendinius turinčioms skruzdėms palikti feromonus atnaujinimo būsenoje. Be to, naudojant gana paprastą mechanizmą, veiksmingai išvengiama ankstyvaus paieškos konvergavimo. Kai visos skruzdės baigia judėjimo procesą iteracijoje, atnaujinamos feromonų reikšmės. Tai daroma pirmiausia, mažinant feromonų reikšmes pastoviu dydžiu (atliekamas garinimas), o paskui leidžiama skruzdėms palikti feromonus vietose, kurias jos aplankė. Feromono atnaujinimas vykdomas pagal šią formulę:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \sum_{k=1}^m \Delta \tau_{ij}^k(t), \quad (1)$$

čia ρ (su $0 \leq \rho < 1$) – feromono išlaikymo vertė ($1 - \rho$ reikštų feromonų garavimo greitį); $\Delta\tau_{ij}^k$ – k -tosios skruzdės paliktas feromono kiekis. Garinimo mechanizmas padeda išvengti neriboto feromonų pėdsako kaupimosi. Jei skruzdės nesirenka konkrečios kelio atkarpos, su ja susijęs feromonų pėdsakas mažėja eksponentiškai ir tai laikui bėgant leidžia algoritmui užmiršti netinkamus sprendinius. Algoritmo sprendžiančio „keliaujančio prekeivio“ problemą $\Delta\tau_{ij}^k(t)$ yra apibrėžiamas taip:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{1}{L_k}, & \text{jei } k \text{ skruzdė judėjo } t \text{ iteracijoje keliu;} \\ 0, & \text{jei keliu nejudėjo,} \end{cases} \quad (2)$$

čia $L_k(t)$ – k -tosios skruzdės judėjimo ilgis. Remiantis (2) formule, kuo trumpesnis skruzdės nueitas kelias, tuo daugiau ji palieka feromonų. Įprastai daugelio skruzdžių lankomos vietos, kurios yra trumpesniuose kelio atkarpose, gaus daugiau feromonų, todėl greičiausiai bus pasirinktos skruzdžių būsimose algoritmo kartojimo iteracijose. Šiuo požiūriu feromonų $\tau_{ij}^k(t)$ kiekis reiškia išmoktą skruzdžių norą pasirinkti miestą j , kai skruzdės yra mieste i (Stützle *et al.*, 2000).

ACO algoritmas buvo palygintas su kitomis bendrojo pobūdžio euristinėmis priemonėmis kai kuriuose santykinai nedideliuose (iki 75 miestų) TSP atvejuose. Pradiniai rezultatai buvo gana neblogi ir parodė šio metodo gyvybingumą. Pavyzdžiui, galima įrodyti, kad skruzdžių kolonija gali pasiekti geresnes euristines savybes nei kiti gamtos įkvėpti algoritmai, pvz., imituotas atkaitinimas ar genetiniai algoritmai (Dorigo *et al.*, 1996).

Tačiau sudėtingesniems TSP atvejams, kai miestų skaičius didelis, ACO algoritmo rezultatai, palyginti su naujausiais algoritmais, prastėja. Vienas iš pasiūlytų patobulinimų yra elitinė skruzdžių kolonija, dar vadinama elitine strategija (Dorigo *et al.*, 1996). Ji papildomai sustiprina feromoną, kuris buvo parinktas algoritmo pradžioje, geriausio kelio sprendinio komponentams. Šis sprendinys įvardinamas kaip visuotinai geriausias sprendinys (angl. *Global best solution*, *GBS*). Tai įgyvendinama pridedant $e/f(\text{GBS})$ kiekį, čia e – elitinių skruzdžių skaičius; $f(\text{GBS})$ – geriausias iki šiol algoritmo rastas sprendimas, naudojamas po kiekvienos eilinės iteracijos. Dorigo *et al.* (1996) pateikti rezultatai rodo, kad naudojant elitinę strategiją su tinkamu elitinių skruzdžių skaičiumi, skruzdžių sistema gali rasti trumpesnius kelius ir anksčiau nei analogai. Tačiau, jei naudojama per daug elitinių skruzdžių, paieška ankstyvoje stadijoje susiduria su neoptimaliais sprendiniais, todėl algoritmas per anksti konverguoja link šio neoptimalaus sprendimo. Paieškos stagnacija apibrėžiama (Dorigo *et al.*, 1996) kaip situacija, kai visos skruzdės eina tuo pačiu keliu, todėl negali rasti geresnių sprendinių.

Tiek ACO algoritmo, tiek ir MMAS paieškos metu rastiems geriausiems sprendimams feromonus leidžiama atnaujinti tik po vieną kartą kiekvieną iteraciją. O rangų grįstoje skruzdžių

sistemoje – fiksuotas dabartinės iteracijos skruzdžių skaičius, išrūšiuotas pagal sprendinio gerumą, atnaujinama feromonų matricą pagal svorį, kuris priklauso nuo sprendinio gerumo. Taip geriausio sprendinio skruzdė gali turėti didžiausią poveikį atnaujinant feromonų matricą, bet neprarandama ir kitų skruzdžių daroma įtaka.

Visi patobulinti skruzdžių kolonijų algoritmai turi vieną svarbų bendrą požymį: jie naudoja geriausius paieškos metu rastus sprendinius daug daugiau nei tai daro standartinis ACO algoritmas. Be to, jie naudoja vietinę paiešką, kad patobulintų skruzdžių sprendinius. Tas faktas, kad papildomas geriausių sprendinių naudojimas priveda prie optimalaus sprendinio yra raktas į geresnį našumą ir yra susijęs su daugelio kombinatorinių optimizavimo problemų paieškos erdve (Stützle *et al.*, 2000). Visgi taip nėra su visomis problemų klasėmis.

Stützle *et al.* (2000) norėdami parodyti, kad skruzdžių kolonijų algoritmai yra geros alternatyvos esamiems algoritmams, naudojamiems sudėtingoms daugelio tikslų optimizavimo problemoms, atliko tyrimą, kurie užtikrina geresnius rezultatus nei standartinis modelis. ACO tyrimai parodė, kad geresnį našumą galima gauti išnaudojant geriausius sprendinius, nustatytus atliekant paiešką. Tačiau naudojant gobšesnę paiešką padidėja ankstyvos paieškos stagnacijos problema, kai paieška susitelkia apie per anksti rastą gerą, bet ne geriausią rezultatą. Todėl išeitis, norint pasiekti geriausius skruzdžių algoritmų rezultatus, yra derinti geriausių paieškos metu rastų sprendimų išnaudojimo metodą su mechanizmu, kuris skirtas išvengti ankstyvos paieškos stagnacijos, naudojimą (Stützle *et al.*, 2000).

1 lentelė. MAX-MIN skruzdžių algoritmų palyginimas sprendžiant TSP problemą naudojant šablonus iš TSPLIB (Stützle *et al.*, 2000)

Šablonas	Optimali vertė	MMAS	Standartinis ACO
eil51	426	427,6	437,3
kroA100	21282	21320,3	22471,4
d198	15780	15972,5	16702,1
ry48p	14422	14553,2	15296,4
ft70	38673	39040,2	39596,3
kro124p	36230	36773,5	38733,1
ft170	2755	2828,8	3154,5

MMAS, specialiai sukurta siekiant patenkinti šiuos reikalavimus:

- Išnaudoti geriausius sprendimus, nustatytus atliekant algoritmą, po kiekvienos iteracijos arba per algoritmo paleidimą pridedant tik vienos skruzdės feromonus. Ši skruzdė gali būti ta, kuri nustatė geriausią dabartinės iteracijos (iteracijoje geriausios skruzdės) ar iki šiol geriausio sprendimo variantą (visų iteracijų geriausia skruzdė).
- Siekiant išvengti paieškos stagnacijos, kiekvieno sprendimo komponento galimų feromonų takų diapazonas yra ribotas intervale $[\tau_{\min}, \tau_{\max}]$.

- Be to, sąmoningai inicijuojame feromonų takus iki $\tau_{\max}$, tokiu būdu priverčiant išsiterinti, kuo didesnę erdvę algoritmo pradžioje.

1.1.2. Elitinė skruzdžių kolonija

Skruzdžių kolonijų algoritmuose, sprendimai konstruojami kiekvienoje iteracijoje statybos etape atrinkus diskrečių tikimybių pasiskirstymą, kuris gaunamas naudojant feromonų matricą. Sprendimo komponento pasirinkimas atliekamas tikimybiškai kiekvienos iteracijos etapo metu. Tikslios tikimybinio sprendimo komponentų pasirinkimo taisyklės skiriasi nuo kitų skruzdžių kolonijų variantų. Loginis algoritmo pritaikymas būtų pereiti nuo diskretinio tikimybinio pasiskirstymo į nepertraukiamą – tikimybių tankumo. Norėdami apskaičiuoti informaciją apie feromonus, galima panaudoti įvairias tikimybių tankumo funkcijas ir naudojimo būdus (Afshar *et al.*, 2006; Liao *et al.*, 2014).

Pagerintas elitinis skruzdžių algoritmas leidžia naudoti tik skruzdę, kuri pateikia geriausią iteracijos sprendimą, kuri naudojama atnaujinti feromonų svorius. Pagrindinis elitinių skruzdžių algoritmo pranašumas: lengvai apieškoma sprendimo erdvė, tačiau jis taip pat turi tam tikrų trūkumų. Vienintelė įprasta tikimybių tankumo funkcija negali apibūdinti situacijos, kai yra keli tikslai, kurie sudaro neatsiejamas paieškos erdvės sritis. Todėl daugelio tikslų problemų elitinis skruzdžių algoritmas gali ir neišspręsti.

Apskritai, skruzdžių kolonijų algoritmai veikia geriau nei kiti euristinės paieškos būdai, įskaitant genetinį algoritmą, mažos apimties problemoms. Tačiau metodo vykdymas blogėja dėl augančių skaičiavimo sąnaudų (Socha *et al.*, 2008). Taip atsitinka dėl teigiamos feromonų atnaujinimo taisyklės. Tikimybės tankumo funkcijos charakteristikų pakeitimas, kuris gali būti laikomas feromono informacijos atnaujinimo taisyklėmis, sumažina PDF įtaka feromonui ir tuo pačiu kitai iteracijai, jei geriausias iteracijos sprendimas gerėja toliau kartojant iteracijas. Tais atvejais, kai geriausias iteracijos sprendimas yra blogesnis nei ankstesnės iteracijos sprendimas, standartinis nuokrypis gali padidėti ir PDF įtaka padidinama, suteikiant daugiau galimybių kitiems paieškos srities regionams. Kitaip sakant, skruzdės pasirenks variantą, kurio stiprinimo įtaka žymiai sumažina šių standartinio nuokrypio padidėjimo tikimybę. Ilgalaikis siūlomos problemos poveikis nuosekliai sumažina paieškos erdvę, daugiausia dėmesio skiriant palyginti nedidelei sprendimų erdvei. Jei kyla problemų dėl mažos paieškos erdvės, labiau tikėtina, kad algoritmas nustatys optimalų sprendimą, iki kol ši sąlyga įvyks. Tačiau, norint rasti didelių problemų sprendimus, metodas gali susidurti su svyravimais ar konvergavimu, kol nebus surastas optimalus sprendimas (Afshar *et al.*, 2006).

Ši strategija garantuos, kad geriausias iteracijos sprendimas visada pagerėja, palyginti su ankstesnėmis iteracijomis, ir todėl užtikrina algoritmo konvergenciją ir lemia jos taikymą didelio masto ir sudėtingose optimizavimo problemose (Ashtiani *et al.*, 2011).

Kalbant apie elitinių skruzdžių skaičių, nustatyta, kad skruzdžių sistemos rezultatai yra labai geri kai $\delta = n = m$, t. y., kai yra tiek elitinių skruzdžių, kiek yra problemoje yra miestų.

Elito sąvoką taip pat galima rasti genetiniuose algoritmuose. Apskritai genetiniame algoritme taikant genetinius operatorius, geriausios kartos individas (geriausias iteracijos sprendimas) su teigiama tikimybe nebus įtrauktas į naująją kartą. Tokiu atveju šio sprendimo genetinė informacija būtų prarasta. Todėl elito idėja yra išsaugoti gerą kartos individą. Dėl to vietinės paieškos aspektai (išnaudojimas) tampa vis svarbesni, o visuotinės paieškos aspektai (tyrinėjimas) tampa mažiau svarbūs (Afshar *et al.*, 2006). Šis galimas trūkumas taip pat galioja ir skruzdžių modifikacijai.

Kita vertus, skruzdžių kolonijų sistemos algoritmo feromonų takai atnaujinami remiantis visų skruzdėlių takais. Įnašo į šią visuotinę informaciją lygis priklauso nuo sukurto sprendimo kokybės. Tai yra dar vienas panašumas lyginant su genetiniu algoritmu, kai tikimybė, kad jie bus atrinkami reprodukcijai, priklauso nuo individo sprendimo kokybės, t. y. geresni sprendimai labiau prisideda prie būsimų sprendimų.

Šios procedūros trūkumas yra tas, kad jei per evoliuciją išaugs bendra sprendimų kokybė ir individų skirtumai sumažės, taip pat sumažėja atrankos tikimybių skirtumai, taigi išnaudojimas nėra toks didelis, kaip norima. Ta pačia savybe pasižymi ir standartinė skruzdžių sistema. Trumpo kelio pabrėžimo poveikis mažėja, kai skruzdės artėja viena prie kitos, ypač kai daug skruzdžių juda gerais, bet nepakankamai optimaliais keliais.

Galima atrankos schema, leidžianti išspręsti šią problemą palaikant vadinamąjį atrankos slėgį genetiniuose algoritmuose, yra rangas: pirmoji populiacija yra išrūšiuojama pagal rangą, o pasirinkimo tikimybė priklauso nuo skruzdės rango. Ši atranka vykdoma, kai visos skruzdės sudaro kelius ir jos surūšiuojamos pagal judėjimo ilgį, o skruzdžių įnašas į feromono lygio atnaujinimą yra įvertintas pagal skruzdžių rangą. Be to, svarstomos tik geriausios skruzdės. Taigi galima išvengti pernelyg pabrėžtų feromonų takų, kuriuos sukelia daugelis skruzdžių keliaujančių neoptimaliais takais, pavojų (Bullnheimer *et al.*, 1999).

Kalbant apie iki šiol surasto geriausio kelionės ilgio svorį, jo negalima viršyti jokių kitu svoriu. Dėl šios priežasties nuspręsta panaudoti svorį $(\sigma - \mu)$ μ -tos skruzdės ir nustatyti $\omega = \sigma - 1$, tai reiškia, kad skruzdžių, kurios apsvarstytos, yra nurungiamos elitinių skruzdėlių per vienetą. Naujas feromonų su elitinių ir reitingais grįstų skruzdžių takų lygių atnaujinimas atliekamas pagal šią formulę:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta \tau_{ij} + \Delta \tau_{ij}^*. \quad (3)$$

Pateikta atnaujinimo procedūra yra geras kompromisas. Iš šių tyrimo rezultatų matyti, kad išnaudojimas (pabrėžiant gerus kelius) ir žvalgymas (per daug dėmesio skiriant kelioms geroms

skruzdėlėms) yra žymiai kokybiškesni ir gerai subalansuoti. Nors aptartame darbe nebuvo optimizuoti parametrai problemai (Bullnheimer *et al.*, 1999).

1.2. Jonvabalių algoritmai

Jonvabalių algoritmas yra vienas iš naujai sukurtų (Yang, 2008) žvalgybos metodų. Tai gamtos (jonvabalių šviesos žiburių) įkvėptas metaeuristinis algoritmas, kuris tinka sunkiausioms optimizavimo problemoms spręsti. FA priskiriamas stochastiniams algoritmams. Tai reiškia, kad ieškant sprendinių rinkinio, naudojamas tam tikras atsitiktinės atrankos būdas. Euristika reiškia rasti arba atrasti sprendimus bandymų ir klaidų keliu (Yang, 2008). Iš tiesų negarantuojama, kad optimalus sprendinys bus surastas per duotą ar leistiną laiką. FA – populiacija grįstas algoritmas. Jame taikoma šviesos intensyvumo I formulė, kai intensyvumas mažėja didėjant atstumo kvadratu, r^2 . Šie reiškiniai gali būti susieti su tikslo funkcija, kurią reikia optimizuoti siekiant išspręsti konkrečią problemą. Kai kurios jonvabalių savybės ir charakteristikos yra idealizuotos, formuluojant FA:

- Visi jonvabaliai yra tos pačios lyties.
- Jonvabalių patrauklumas yra proporcingas jų šviesos intensyvumui.
- Šviesos intensyvumą veikia arba nustato tikslo funkcija.

Siekiant tinkamai parengti FA, reikia apibrėžti du svarbius klausimus: šviesos intensyvumo kitimą ir patrauklumo formulavimą. Šie du klausimai leidžia kūrėjams pritaikyti skirtingus FA taip, kad jie būtų tinkamiausi sprendžiamų problemų reikmėms. Standartinio FA šviesos intensyvumas I , atitinkantis sprendimą, yra proporcingas tikslo funkcijos vertei, o šviesos intensyvumas $I(r)$ apskaičiuojamas pagal tokią lygtį:

$$I(r) = I_0 e^{-\gamma r^2}, \quad (4)$$

čia I_0 žymi šviesos šaltinio intensyvumą, o šviesos absorbcija apytiksliai nustatoma naudojant fiksuotą šviesos sugerties koeficientą γ .

Jonvabalių β patrauklumas yra proporcingas jų šviesos intensyvumui $I(r)$. Apibendrinant, FA yra kontroliuojamas trimis parametrais: atsitiktinių imčių parametru, patrauklumo ir absorbcijos koeficientai. Pagal parametrų nustatymą, FA pasižymi dvejomis elgsenomis. Pirmoji gaunama, kai $\gamma \rightarrow 0$ ir pastaroji, kai $\gamma \rightarrow \infty$. Jei $\gamma \rightarrow 0$, patrauklumas tampa $\beta = \beta_0$. Tai yra, patrauklumas yra pastovus bet kurioje paieškos erdvės vietoje. Šis elgesys yra ypatingas dalelių spiečiaus optimizavimo atvejis. Jei $\gamma \rightarrow \infty$, judėjimas tampa atsitiktine eiga, kuri iš esmės yra lygiagreti imituojamui atkaitinimui. Iš tikrųjų kiekvienas jonvabalių įgyvendinimas gali būti tarp šių dviejų elgsenų.

Pagal tai, kas modifikuota FA gali būti klasifikuojami pagal jų sudedamąsias dalis ar savybes, pvz:

- jonvabalių verčių naudojimą (dvejetainių, realių verčių);

- populiacijos schemą (spiečio, daugelio spiečių);
- tikslo funkcijos vertinimą;
- geriausio sprendimo (ne elitis, elitinis) nustatymą;
- jonvabalių judėjimą (pagal Gausą, pagal „Levy“ skrydžius, pagal tolygųjį pasiskirstymą).

FA pirmiausia priklauso nuo šviesos intensyvumo ir patrauklumo formulavimo. Abu veiksniai suteikia didelę reikšmę algoritmų tobulinimui. Pavyzdžiui, Tilahun *et al.* (2012) pakeitė atsitiktinį ryškiausio jonvabalio judesį, kuris kai kuriose kartose, kai dabartinė geriausia padėtis nepagerėja, gali mažinti šviesos intensyvumą siekiant apieškoti didesnę sprendimo erdvės dalį. Siūloma modifikacija stengiamasi pagerinti ryškiausios šviesos padėtį, generuojant atsitiktines vertes ir judinant jonvabalį geriausia veikimo kryptimi. Jei tokia kryptis neegzistuoja, ryškiausias jonvabalis lieka dabartinėje padėtyje. Šiuo atveju ryškiausias jonvabalis taip pat yra elitinis sprendimas, nes jis niekada nepakeičiamas blogesniais rasta dabartinės kartos sprendimais, kurių tikslo funkcijos reikšmės yra mažesnės. Eksperimentai, atlikti optimizuojant septynias etalonų funkcijas, parodė, kad šis modifikuotas FA pranašesnis už savo klasikinį pirmtaką (Yang *et al.*, 2010).

Dvejietinis FA taip pat pritaikytas daugelyje sričių. Pavyzdžiui Palit *et al.* (2011) pasiūlė dvejetainę jonvabalių kriptanalizę, norėdamas iškoduoti tekstą iš šifro, naudodamas Merkle-Hellmano šifro algoritmą (Behrouz, 2007). Pakeitimai beveik visuose dvejetainio jonvabalio elementuose turi būti atliekami dėl naujo jonvabalių veikimo. Algoritmo rezultatai buvo palyginti su genetinio algoritmo rezultatais. Remiantis rezultatais matyti, kad siūlomas dvejetainis FA buvo daug efektyvesnis už genetinį algoritmą, kai jis naudojamas tam pačiam tikslui. Kitas dvejetainio FA įgyvendinimas buvo sukurtas Falcon *et al.* (2011), kuris naudoja dvejetainį sprendimų kandidatų kodavimą, adaptyvų šviesos absorbcijos koeficientą, skirtą pagreitinti paiešką, ir konkrečioms problemoms būdingas žinias, skirtas tvarkyti neįmanomus sprendinius. Empirinė analizė atlikta sprendžiant sistemos lygio gedimų diagnostiką, kuri yra kombinatorinė optimizavimo problema. Iš siūlomo algoritmo rezultato, kurie palyginti su dirbtinės imuninės sistemos algoritmu (Yang *et al.*, 2008) ir PSO (Falcon *et al.*, 2010), matyti, kad jis gerokai viršija esamas technologijas, atsižvelgiant į konvergavimo greitį ir atminties reikalavimus. Chandrasekaran *et al.* (2012) pasiūlė dvejetainį FA skirtą elektros tinklo efektyvumo ir patikimumo išipareigojimo problemai spręsti, kurią iškėlė Carrion *et al.* (2006), nesilaikant tinklo saugumo apribojimų. Siūlomo algoritmo rezultatai buvo perspektyvūs lyginant su kitais pateiktais metodais.

Siekiant stabilizuoti FA, Farahani *et al.* (2011) suformulavo naują jonvabalių algoritmą, didinantį konvergavimo greitį. Tam buvo naudojamas Gauso skirstinys, siekiant kiekviena iteracija perkelti visus jonvabalius geriausio sprendinio link. Nepaisant fiksuoto atsitiktinės atrankos parametro (taip pat žingsnio dydžio), šis parametras siūlomame algoritme pritaikytas. Šis algoritmas išbandytas

penkioms standartinėmis funkcijomis. Iš eksperimentinių rezultatų matyti, kad FA su gauso skirstiniu spartesnis ir tikslesnis nei klasikinis FA (Fister *et al.*, 2013).

Ankstyvaisiais etapais FA rado norimus sprendimus nuolatinėms optimizavimo problemoms. Sunkumai kilo, kai bandyta pritaikyti sudėtingesnioms optimizavimo problemoms, kuomet algoritmas neranda tinkamo sprendimo. Siekiant išspręsti tokias problemas, optimizavimo algoritmas buvo kryžmintas. Atsižvelgiant į tai, FA buvo kryžmintas su kitais optimizavimo algoritmais, mašinų mokymosi metodais, euristika ir t. t. Kryžminimas gali būti įterptas beveik visuose FA komponentuose, pavyzdžiui, inicijavimo procedūroje, vertinimo ar judėjimo funkcijose ir pan. (Fister *et al.*, 2013).

Siekiant susidaryti išsamų šios srities vaizdą, Gandomi *et al.* (2011) peržiūrėjo, kur FA buvo naudojama sprendžiant mišrią nepertraukiamą struktūrinio optimizavimo problemą. Literatūroje rasta, kad FA išspręsta suvirinto pluošto konstrukciją, slėginių indų konstrukciją, sraigtinį suspaudimo spyruoklinį dizainą, gelžbetoninio pluošto konstrukcijas, pakopinę sijos konstrukciją ir automobilio smūgio dizainą ir kitas inžinerines problemas. Iš optimizavimo rezultatų matyti, kad FA yra efektyvesnis už kitus metaeuristinius algoritmus, tokius kaip PSO, genetiniai algoritmai, imituotas atkaitinimas ir diferencinė evoliucija. Nors FA buvo labai efektyvūs, pastebėta, kad paieškos procesas artėjant prie optimalaus sprendimo pradeda svyruoti, t. y. atsitiktinai ieškoma aplink optimalų sprendimą. Šis jonvabalių elgesys galėtų būti pagerintas, palaipsniui mažinant atsitiktinės atrankos parametro įtaką iteracijoms didėjant.

Yang (2013) suformulavo naują daugiafunkcinį FA, kuris išplėtė jonvabalių algoritmą tam, kad būtų gaunamas Pareto frontas, t. y. toks sprendimas, kurio negalima pagerinti nepabloginant vieno iš tikslų kriterijų. Šis algoritmas buvo išbandytas daugiafunkcinių funkcijų pogrupyje su išgaubtais, neišgaubtais ir nepertraukiamais Pareto frontais, paimtais iš literatūros, ir tuomet buvo pritaikytas projektavimo optimizavimo etalonams pramonės inžinerijoje. Rezultatai buvo lyginami su kitais algoritmais daugiafunkciniam optimizavimui, pvz., vektorių vertinimo genetiniu algoritmu (angl. *Vector Evaluated Genetic Algorithm*, VEGA) (Schaffer, 1985), be svarbiausio rūšiavimo genetiniu algoritmu-II (angl. *Non-dominated Sorting Genetic Algorithm II*, NSGA-II) (Deb *et al.*, 2002), daugiafunkcine diferencine evoliucija (angl. *Multi-Objective Differential evolution*, MODE) (Babu *et al.*, 2007), diferencinė evoliucija daugiafunkciniam optimizavimui (angl. *Differential Evolution for Multiobjective Optimization*, DEMO) (Tusar *et al.*, 2005), daugiafunkcine bičių algoritmu (Falcon *et al.*, 2007) bei stipriuoju Pareto evoliuciniu algoritmu (angl. *Strength Pareto Evolutionary Algorithm*, SPEA) (Zitzler *et al.*, 1999) ir gauta, kad siūlomas algoritmas yra efektyvus daugiafunkcinis optimizatorius.

FA (Abedinia *et al.*, 2012) taip pat pritaikytas aplinkosaugos ir ekonominės galios išsiuntimo balanso (angl. *Environmental/Economic power Dispatch*, EED) problemai. Ši problema buvo

suformuluota kaip netiesinė suvaržyta daugiafunkcinė problema, susijusi su tikslais optimizuojant kuro sąnaudas, išmetamų dujų kiekį ir sistemos nuostolius. Rezultatai buvo lyginami su kitais žinomais daugiafunkciniais algoritmais iš literatūros. Iš rezultatų matyti, kad algoritmui įmanoma išspręsti daugiafunkcinę problemą, bet rezultatai neoptimalūs.

Šiek tiek kitokia problema buvo išspręsta Niknam *et al.* (2012), kur autoriai pasiūlė daugiafunkcinį FA, kad pasiektų nereglamentuojamų (Pareto optimalių) sprendimų rinkinį. Siekiant pagerinti bendrą algoritmo veikimą, buvo naudojamas atsitiktinės atrankos ir absorbcijos koeficientų derinimas, pagrįstas chaotiškų žemėlapių naudojimu ir savireguliuojančiomis tikimybinėmis mutacijų strategijomis. Šis algoritmas išsprendė dinamiškos ekonominės emisijos išsiuntimo (angl. *Dynamic Economic Emission Dispatch, DEED*) problemą, kuri sujungė šilumos ir elektros energijos įrenginius vadinamosiose kombinuotose šilumos ir elektros energijos sistemose. Problema buvo apibrėžiama kaip daugiafunkcinė ir siūlomo algoritmo našumas buvo sėkmingai patvirtintas naudojant skaitines simuliacijas.

Senthilnath *et al.* (2011) taikė FA, kad sugrupuotų duomenų objektus į grupes pagal jų atributų vertes. FA rezultatai buvo lyginami su kitų gamtos įkvėptų algoritmų, pvz., ABC ir PSO, ir kitų devynių algoritmų, naudojamų literatūroje, rezultatais. Palyginimui naudotas efektyvumo matas buvo klasifikavimo klaidų procentas (angl. *Classification Error Percentage, CEP*), kuris apibrėžiamas kaip santykis tarp klaidingai klasifikuojamų mėginių skaičiaus bandymų duomenų rinkinyje ir bendro mėginių skaičiaus bandymo duomenų rinkinyje. Remiantis gautais rezultatais, autoriai padarė išvadą, kad FA buvo efektyvus grupavimo metodas.

Yang *et al.* (2010) suformulavo naują metaeuristinį algoritmą pasitelkdamas „Levy“ skrydžių judėjimo strategiją. Iš skaitmeninių tyrimų, matyti, kad siūlomas „Levy“ skrydžio FA yra geresnis už PSO ir genetinius algoritmus efektyvumo ir sėkmės rodiklių atžvilgiu. Yang (2012) analizavo su metaeuristika susijusią konvergenciją ir efektyvumą, pvz., spiečiaus protinius gebėjimus, atsitiktinius skrydžius, Levy skrydžius ir erelio strategiją. Nors autorius bandė atrasti keletą matematinių pagrindimų metaeuristiniam elgesiui, jis padarė išvadą, kad nepaisant to, kad naujai sukurtas gamtos įkvėptas metaeuristinis algoritmas veikia gerai, matematinis jų supratimas iš dalies išlieka neatskleistas.

Konkrečios euristikos yra skirtos tam tikroms problemoms spręsti ir paprastai tobulina problemų sprendimo priemonių rezultatus, t. y. euristikos, naudojasi specifinėmis žiniomis apie konkrečias dalykines sritis. Iš tiesų, tokia euristika taip pat gali būti įtraukta į FA. Toks kryžminimas FA apskritai pagerina rezultatus sprendžiant šią problemą.

FA taip pat gali būti naudojamas, kaip etaloninis kryžminimas su kitais bendraisiais problemų sprendėjais dėl jo savybių, t. y. daugiaryšio ryšio ir greitesnio konvergavimo. Kaip pirmasis FA kryžminimas, Yang *et al.* (2010) suformulavo naują metaeuristinį paieškos algoritmą, pavadintą

erelio strategiją (angl. *Eagle Strategy*). Jis apjungia „Levy“ skrydžio paiešką su FA. Erelio strategiją įkvėpė erelių elgesys. Ereliai skraido laisvai per savo teritoriją atsitiktine tvarka panašiai, kaip „Levy“ skrydžių strategijoje. Kai grobis yra matomas, erelis bando sugauti jį kuo efektyviau. Algoritminiu požiūriu erelio strategija susideda iš dviejų komponentų: atsitiktinės paieškos pagal „Levy“ skrydį ir intensyvios vietinės paieškos. Įdomu tai, kad FA buvo naudojamas vietinei paieškai. Šis kryžmintas metaeuristinis algoritmas panaudojo „Ackley“ funkciją su Gauso triukšmu. Iš modeliavimo rezultatų matyti, kad erelio strategija gerokai pralenkia PSO algoritmą efektyvumo ir sėkmės rodikliais (Fister *et al.*, 2013).

1.3. Uodų taikinių paieškos algoritmas

Uodų taikinių paieškos algoritmą motyvuoja tam tikri pastebėjimai dėl uodų elgesio, tai kinematikos ir dinamikos uodų elgesyje matomo lygiagretumo, vietos sąveikos ir savarankiško organizavimo savybės. Tai paskatino mokslininkus mėgdžioti taikinių ieškančių uodų elgesį sukuriant protingų skaičiavimų modelį. Šis naujas modelis gali įveikti kai kuriuos esamų populiarių metaeuristinių algoritmų trūkumus. Algoritmas turi šiuos privalumus (Feng *et al.*, 2013):

- gali atlikti didelio masto paskirstytą lygiagrečią optimizaciją;
- gali konverguoti;
- parametrai nepriklauso nuo algoritmų mokymosi ar ankstesnių žinių;
- gali apibūdinti sudėtingą elgesį ir dinamiką;
- turi visapusiškas optimizavimo galimybes keliems tikslams;
- yra efektyvus, iš esmės nepriklauso nuo pradinių sąlygų, problemos dydžio ir nedidelių parametru pokyčių;
- turi galingą apdorojimo pajėgumą sudėtingoje ir dinamiškoje realiojo laiku besikeičiančioje aplinkoje;
- yra lankstus ir lengvai pritaikomas įvairiems optimizavimo uždaviniams spręsti.

Apskritai jis daugeliu atžvilgių iš esmės skiriasi nuo kitų populiarių algoritmų, įskaitant optimizavimo mechanizmą, jo elementais ir jų būsenomis, biologiniu modeliu, matematiniu modeliu ir teoriniu pagrindu, kuriuo jis grindžiamas (Feng *et al.*, 2013).

Šiame modelyje kiekvienas skaičiavimo elementas yra traktuojamas kaip dirbtinis uodas m_{ij} , kuris pakeičia n miestų *TSP* sprendimo procesą į taikinio paieškos procesą, kuriame veikia $n \times n$ dirbtinių uodų spiečius. Visi dirbtiniai uodai yra tolygiai paskirstyti vienodame atstume, supančiame taikinį. Kiekviena patelė, atitinka gyvą skaičiavimo elementą ir yra pritraukiama prie taikinio aptinkant anglies dioksidą, kvapus ir spinduliuojamą šilumą. Spindulio nuotolis (angl. *Radial Distance*) tarp dirbtinio uodo ir taikinio atitinka dirbtinių uodų naudingumą – tai dirbtinių uodų sėkmės rodiklis. Šio rodiklio reikšmę galima būtų prilyginti tikslo funkcijai kitose metaeuristiniuose

algoritmuose. Kuo didesnė anglies dioksido ir kvapų koncentracija, tuo greičiau dirbtiniai uodai bando judėti link savo taikinio. Kai visi dirbtiniai uodai nustoja judėti, būdami pusiausvyros būsenoje, skaičiavimo procesas pasiekia optimalų keliaujančio prekeivio sprendimą (Feng *et al.*, 2013).

Išsamiau, kiekvienas dirbtinis uodas m_{ij} turi pilkumo skalės reikšmę r_{ij} , kuri uodams judant nuolat kinta intervale $[0, 1]$. Kai visi uodai yra pusiausvyros būsenoje, visi r_{ij} įgis 1 arba 0 vertes. Būsena $r_{ij} = 1$ reiškia, kad dirbtinis uodas puola taikinį ir kelio p_{ij} vertė 0 t. y. kelias praeinamas. Atvirkštinė būsena $r_{ij} = 0$ reiškia, kad uodas m_{ij} nepuola taikinio.

MHSA įgyvendinimas gali pasižymėti aukštu skaičiavimų lygiagretumu ir yra lengvai plečiamas. Visi skaičiavimai ląstelių dinamikos tiek tame pačiame masyve, tiek skirtinguose masyvuose yra vykdomi vienu metu. Ląstelinė struktūra, ląstelių dinamika ir algoritmas nepriklauso nuo problemos skalės (Feng *et al.*, 2013).

2 lentelė. Gramacy & Lee, Ackley, Rastrigin, Rosenbrock, Griewank ir Schwefel šabloninių funkcijų rezultatai įgyvendinant MHSA (Alauddin, 2016)

Funkcija	Erdvės matmuo	Geriausia vertė	Vidutinė vertė	Standartinis nuokrypis
Gramacy ir Lee	1	$6,12 \times 10^{-16}$	$3,4 \times 10^{-14}$	$4,9304 \times 10^{-31}$
Griewank	5	$2,3368 \times 10^{-5}$	$4,34 \times 10^{-4}$	$3,4695 \times 10^{-18}$
Rastrigin	5	$2,55 \times 10^{-2}$	1,04	2×10^{-15}
Rosenbrock	2	$1,68 \times 10^{-4}$	0,00186	$4,33 \times 10^{-3}$
Ackley	5	$1,3223 \times 10^{-2}$	0,09435	$1,3878 \times 10^{-17}$
Schwefel	5	$4,54 \times 10^{-4}$	0,00187	$3,85 \times 10^{-4}$

Yra ir kitokių uodais grįstų algoritmų, pvz. uodų skraidymo optimizavimas. Šis algoritmas yra pakankamai naujas, imituojantis uodų elgesį randant skylę tinkluose (jei tokia yra). Uodų skraidymo optimizavimas buvo išbandytas su šešiais šablonais (Dieterich *et al.*, 2012; Jamil *et al.*, 2013), išbandančiais skirtingas algoritmo savybes. Šis algoritmas įgyvendintas optimizuojant Gramacy & Lee, Ackley, Rastrigin, Rosenbrock, Griewank ir Schwefel šablonines funkcijas, kurių rezultatai yra sudaromi antroje lentelėje. Iš lentelės matyti, kad algoritmas yra labai efektyvus siekiant optimizuoti įvairaus pobūdžio problemas. Taip pat veikia pakankamai greitai, ypač ties mažesnėmis problemomis (Alauddin, 2016).

MHSA buvo pritaikytas daugelio sandėlių transporto maršruto problemoje (Wang *et al.*, 2017). Sandėliuose yra transporto priemonės, skirtos paslaugų teikimui klientams, o kelių eismo srauto duomenis gaunami per eismo srauto detektorius. Pateikiamas kiekvieno kliento poreikis, kiekvienos transporto priemonės nuvažiuoto nuotolio apribojimas, klientų buvimo vieta ir klientų paslaugų priėmimo laiko langai, o įkrovimo laikas nėra svarstomas. Be to, kiekvienos transporto priemonės fiksuotos ir kelionės išlaidos yra žinomos iš anksto, siekiant suplanuoti efektyvius maršrutus taip, kad transporto priemonės iš kiekvieno sandėlio, galų gale užbaigtų paskirstymo užduotį ir grįžtų į savo

pradinį sandėlį, o bendri tikslai (transporto išlaidos, degalų sąnaudos, laiko langų baudos, vairuotojai ir kt.) yra minimalūs.

Problemos sprendimas MHSA buvo pasiektas problemą suprastinant iki TSP ir kitus tikslus įgyvendinant per įvestus apribojimus tokius, kaip transporto išlaidos, degalų sąnaudos ir kt. Algoritmas buvo lyginimas su tabu paieška ir ACO algoritmu. Taip tyrimais buvo parodyta, kad algoritmas buvo tikslesnis negu minėti algoritmai ir taip pat rado geresnius sprendimus problemai (Wang *et al.*, 2017).

1.4. Skyriaus apibendrinimas

Magistro baigiamajame darbe išnagrinėtos trys algoritmų šeimos: skruzdžių kolonijų optimizacijos, jonvabalių ir uodų taikinių paieškos. Algoritmai pasirinkti dėl skirtingų priežasčių. Pavyzdžiui, skruzdžių kolonijos optimizavimas yra pakankamai gerai ištirtas, t. y. teoriškai ir praktiškai algoritmo veikimas yra įrodytas. Naujausioje mokslinėje literatūroje algoritmas pritaikomas specifinėms problemoms arba tyrinėjami nauji kryžminti ar modifikuoti algoritmai. Nors neaišku, kaip algoritmas veiks su medicininiais vaizdais, pirminiai testai su šabloninių vaizdų briaunų radimu rodo, kad potencialias yra. Nustatyta, kad algoritmas nėra lygintas su kitais šiame darbe taikomais algoritmais segmentavimo problemoje. Kadangi ACO yra gerai ištirtas, tai jį taikysime ir naudosime, kaip bazinį algoritmą palyginimui su kitais.

FA neturi daug modifikacijų ir yra mažiau ištirtas nei ACO algoritmas. FA yra pakankamai naujas ir turintis teorinį pagrindimą. Pagrindinis šio algoritmo privalumas yra greitas sprendimo konvergavimas. Deja tai pasireiškia ne visose problemose, todėl dažnai šis metodas yra kryžminamas. Bet stebint TSP rezultatus matyti, kad algoritmas yra efektyvesnis negu ACO algoritmas. Nes palyginti šio algoritmo skirtingose problemose negalima, todėl reiktų šį algoritmą palyginti tokioje pačioje problemoje ypač atsižvelgiant, kad algoritmo kokybė kinta keičiantis problemoms.

Remiantis moksline literatūra matyti, kad MHSA yra pats jauniausias iš analizuojamų algoritmų, nepakankamai ištirtas ir taikytas tik TSP spręsti. Taigi tikslinga šį algoritmą ištirti detaliau. Teoriškai algoritmas turi mažas lygiagretaus skaičiavimo sąnaudas, kas reikštų, kad skaičiavimai geriau išnaudoja kompiuterio gijas. Taip pat algoritmo konvergavimas yra teoriškai įrodytas, be to parodyta, kad jis gerai veikia sprendžiant daugiamates problemas.

2. PARINKTŲ METODŲ VEIKIMŲ PRINCIPAI IR PRITAIKYMAS

Šiame skyriuje aptariami algoritmų veikimo principai, kurie atrinkti literatūros apžvalgos metu. Toliau metaheuristiniai algoritmai lyginami apžvelgiant jų privalumus ir trūkumus. Galiausiai aptariami algoritmų pritaikymai medicininiais vaizdams apdoroti.

2.1. Algoritmų veikimo principai

Poskyriuje detalai išnagrinėjami standartiniai skruzdžių kolonijos optimizavimo, jonvabalių ir uodų taikinių paieškos algoritmų veikimo principai.

2.1.1. Skruzdžių kolonijos optimizavimo veikimo principas

Skruzdžių kolonijų algoritmai išnaudoja populiaciją, kolektyviai sprendžiant nagrinėjamą optimizavimo problemą. Skruzdžių surinkta informacija paieškos proceso metu saugoma feromonų takeliuose r_{ij} , susietuose su jungtimis l_{ij} . Feromonų takai koduoja ilgalaikę atmintį apie visą skruzdžių paieškos procesą. Priklausomai nuo pasirinktos problemos, feromonų takai gali būti susieti su visais problemos takais arba tik kai kuriais iš jų. Keliai taip pat gali turėti susijusią euristinę vertę q_{ij} , reiškiančią *a priori* informaciją apie problemos apibrėžimą arba paleisties laiką, kurį pateikia išorinis šaltinis. ACO algoritmas pasižymi šiomis savybėmis:

- Skruzdė ieško minimaliomis sąnaudomis pagrįstų sprendimų – $J_\varphi = \min \varphi$, kai $J_\varphi(L, t)$.
- Skruzdė k turi atmintį M_k , kurią ji gali naudoti, kad išsaugotų informaciją apie praeitus kelius. Atmintis gali būti naudojama kuriant naujus sprendimus, įvertinant rastą sprendimą ir atsekti kelią atgal.
- Skruzdė būsenoje $s_t = (s_{t-l}, i)$ gali judėti į bet kurį j mazgą skruzdės kaiminystėje.
- Skruzdei k gali būti priskirta pradinė būseną s_s^k ir viena ar daugiau nutraukimo sąlygų e^k . Įprastai pradinė būseną išreiškiama kaip bloko ilgio seka.
- Skruzdės juda nuo pradinės būsenos ir pereina prie galimų kaimyninių būsenų, formuodamas sprendimą laipsniškai. Procedūra sustabdoma, kai bent viena skruzdė k įvykdo bent vieną iš nutraukimo sąlygų e^k .
- Skruzdė k gali judėti nuo i mazgo į j mazgą N_i^k sprendimų erdvėje. Šis žingsnis pasirenkamas taikant tikimybinio sprendimo taisyklę.
- Skruzdžių judėjimo taisyklė yra funkcija gaunama iš mazgo vietinės duomenų struktūros saugomų verčių, skruzdėlės asmeninės atminties, kurioje saugoma praeities istorija, ir problemos suvaržymų.

- Skruzdei judant iš i mazgo prie kaimyninio j mazgo, ji gali atnaujinti feromono trajektoriją τ_{ij} .
- Kai suformuojamas sprendimas, skruzdė gali atsekti tą patį kelią judėdama atgal ir atnaujinti feromonų vertes. Tai literatūroje vadinama uždelstu feromono atnaujinimu.
- Skruzdei sukūrus sprendimą ir grįžus nueitu keliu atgal į šaltinio mazgą, ji „miršta“, t. y. atlaisvina visus priskirtus kompiuterio išteklius.

Nors kiekviena kolonijos skruzdė yra pakankamai sudėtinga, kad rastų problemos sprendimą, geros kokybės sprendiniai gali atsirasti tik kaip kolektyvinės sąveikos rezultatas. Kiekviena skruzdė naudoja tik privačią informaciją ir informaciją, esančią vietiniame mazge, kurį ji lanko. Todėl skruzdžių bendravimas yra netiesioginis tarpininkavimas ir informacijos keitimasis, atliekamas per saugomas feromono takų vertes.

Neoficialiai skruzdžių elgesys skruzdžių kolonijų algoritme gali būti apibendrintas taip: skruzdžių kolonija tuo pačiu metu ir asinchroniškai pereina per gretimas problemos būsenas, judėdama per kaimyninius G mazgus. Judėdamos skruzdės palaipsniui sukuria optimizavimo problemos sprendinius. Skruzdė kurdama sprendinį, vertina dalinį sprendimą ir kaupia informaciją apie jo gerumą feromonų keliuose, kuriais judėjo sprendimo linkme. Ši feromono informacija nukreipia būsimų skruzdžių paiešką.

Be skruzdžių veiklos, skruzdžių kolonijos algoritmas apima dar dvi procedūras: feromonų garavimo ir paslaugų teikimo sistemos veiksmus. Feromono garavimas yra procesas, kurio metu automatiškai sumažėja feromonų trajektorijos intensyvumas jungtyse. Praktiniu požiūriu feromono išgarinimas yra būtinas, kad būtų išvengta pernelyg greito algoritmo konvergavimo į netinkamą optimalų regioną. Taip įgyvendinamas užmiršimas ir skatinama ieškoti naujų paieškos erdvės sričių.

Paslaugų teikimo sistemos (angl. *Daemon*) veiksmai nėra privalomi skruzdžių kolonijų metaeuristikos komponentai, bet gali būti naudojami centralizuotiems veiksmams įgyvendinti, kurių negali atlikti viena skruzdė. Pavyzdžiui, vietinės optimizavimo procedūros įjungimas arba visuotinės informacijos rinkimas, kurį galima naudoti ACO algoritme norint nuspręsti, ar yra naudinga išsaugoti papildomą feromono reikšmę, kad paieškos procesas būtų nukreiptas visuotiniu požiūriu. Kaip praktinį pavyzdį, paslaugų teikimo sistema gali stebėti kiekvienos skruzdės kelią ir gali pasirinkti papildomai padidinti feromono kelią, kurį naudojo skruzdė su trumpiausiu keliu. Toks paslaugų teikimo sistemos naudojimas atliekant feromonų atnaujinimus vadinamas neprijungtais feromonų atnaujinimais (Dorigo *et al.*, 1999).

2.1.2. Jonvabalių veikimo principas

Jonvabaliai yra švytintys vabalai, kurie šviečia pasitelkdami bioluminescenciją. Paprastai apibūdinant jonvabalių algoritmą, yra naudojamos trys idealizuotos taisyklės:

1. Visi jonvabaliai traukia vienas kitą nepaisant jų lyties.
2. Jonvabaliai švyti tik tam, kad pritrauktų grobį ir kitus jonvabalius, kad pasidalininti maistu.
3. Patrauklumas yra proporcingas jų ryškumui, todėl kiekvienas jonvabalis juda prie kaimyno, kuris šviečia ryškiau (Senthilnath *et al.*, 2011).

Jonvabalių algoritmas yra populiacija grįstas algoritmas, skirtas rasti optimalų tikslo funkcijų sprendinį. Jonvabaliai yra atsitiktinai paskirstomi paieškos erdvėje. Įsivaizduojama, kad programiniai agentai yra lyg jonvabaliai, turintys liuminescencijos savybę, kuri skleidžia tikslo funkcijos vertei proporcingą šviesą. Kiekvieną jonvabalį pritraukia šviesesnis kitų kaimyninių jonvabalių švytėjimas. Patrauklumas mažėja, didėjant atstumui. Jei nėra ryškesnio jonvabalio, jis pasirenka judėti atsitiktine kryptimi. Taikant jonvabalių algoritmus grupavimui (angl. *Clustering*), sprendimų kintamieji yra grupių centrai. Tikslo funkcija yra susijusi su suma visuose mokymuose, nustatytuose Euklido atstumu n -dimensijos erdvėje.

Remiantis šia tikslo funkcija, iš pradžių visi jonvabaliai yra atsitiktinai išsklaidyti per paieškos erdvę. Du jonvabalių algoritmo etapai yra tokie:

Šviesos intensyvumo kitimas: Šviesos intensyvumas yra susijęs su tikslo funkcijos vertėmis. Taigi norint spręsti maksimizavimo (minimizavimo) problemą reikia, kad aukšto (žemo) intensyvumo jonvabaliai pritrauktų kitus jonvabalius su aukštu (mažu) intensyvumu. Pavyzdžiui egzistuoja n jonvabalių, x_i – i -tojo jonvabalio sprendimas, o $f(x_i)$ reikštų jo sprendimo tinkamumo vertę. Čia jonvabalio šviesumas I nustatomas taip, kad atspindėtų jonvabalio dabartinę padėtį tikslo funkcijoje $f(x)$, kuri nurodo sprendimo kokybę.

Judėjimas link patrauklaus jonvabalio: Jonvabalio patrauklumas yra proporcingas šviesos intensyvumui, kurį mato gretimi jonvabaliai. Kiekvienas jonvabalis turi savo išskirtinį patrauklumą β , kurio reikšmė nustato, kiek jis pritraukia kitų narių. Tačiau patrauklumas β yra santykinis, nes kinta priklausomai nuo atstumo r_{ij} tarp dviejų jonvabalių i ir j , atitinkamai x_i ir x_j vietose ir yra apskaičiuojamas pagal formulę:

$$r_{ij} = \|x_i - x_j\|. \quad (1)$$

Tuomet jonvabalio patrauklumo funkcija $\beta(r)$ apskaičiuojama pagal formulę:

$$\beta(r) = \beta_0 e^{-\gamma r^2}. \quad (2)$$

Kaip matyti jonvabalių atstumas mažina patrauklumą eksponentiniu dėsniu. Jonvabalio i judėjimas iš x_i vietos link kito patrauklesnio jonvabalio j vietoje x_j tolesnėje iteracijoje nustatomas pagal formulę:

$$x_i(t+1) = x_i(t) + \beta_0 e^{-\gamma r^2} (x_j - x_i). \quad (3)$$

2.1.3. Uodų taikinių paieškos algoritmas

MHSA kiekvieną „keliaujančio prekeivio“ problemos miestą matricoje $M(t)$ laiko, kaip dirbtinį uodą m_{ij} , kur problema tarp n miestų paverčiama taikinio paieška uodų grupėje.

Pavyzdžiui c_{ij} yra svoris tarp miestų porų c_i ir c_j . Uodų taikinių paieškos modelyje c_{ij} taip pat yra dirbtinio uodo m_{ij} trauka, t. y. kuo didesnis svoris c_{ij} , tuo geresnė uodo paieška ir tuo lengviau randamas taikinis ir gaunamas optimizacijos sprendinys. Jei uodo pilkumo vertė yra r_{ij} , tuomet c_{ij} ir r_{ij} plėtojamas su taikinio paieška kiekvienoje iteracijoje. Tuomet $u_{ij}(t)$ reiškia atstumą tarp dirbtinių uodų m_{ij} ir taikinio laiku t :

$$u_{ij}(t) = \exp(-c_{ij}(t)r_{ij}(t)x_{ij}(t)). \quad (4)$$

Laiku t traukos funkcija, sukurta taikinio, yra apibrėžiama formule:

$$P(t) = \varepsilon^2 \ln \sum_i^n \sum_j^n \exp[-u_{ij}^2(t)/2\varepsilon^2] - \varepsilon^2 \ln nn, \quad (5)$$

čia $0 < \varepsilon < 1$, kuo didesnis $P(t)$, tuo greičiau dirbtiniai uodai randa taikinį ir tuo greičiau randamas problemos optimizavimo sprendimas. Trauka, kuriama taikinio, verčia dirbtinį uodą, ypač tą uodą, kurio naudingumas mažiausias, judėti link taikinio, kad pagerintų individualią naudą.

Iteracijos laiku t , bendradarbiavimo funkcijos $Q(t)$ formulė aprašoma taip:

$$Q(t) = \sum_i^n \sum_j^n r_{ij}(t)x_{ij}(t) - 2 - \sum_{i,j} \int_0^{u_{ij}} \left\{ [1 + \exp(-10x)]^{-1} - 0,5 \right\} dx. \quad (6)$$

Dirbtiniai uodai gali judėti keliu iki taikinio dėl kelių veiksnių: dėl individualaus taikinio paieškos elgesio; visos populiacijos taikinio paieškos elgesio; paties taikinio traukos; sąveikos tarp dirbtinių uodų. Šie keturi veiksniai verčia dirbtinius uodus judėti link taikinio. Tam, kad galima būtų kontroliuoti šiuos veiksnius yra įvedami įtakos koeficientai, kurių pagalba gauname hibridinę traukos funkciją apskaičiuojama formule:

$$E_{ij}(t) = -\lambda_1 u_{ij}(t) - \lambda_2 J(t) - \lambda_3 P(t) - \lambda_4 Q(t), \quad (7)$$

čia $\lambda_1, \lambda_2, \lambda_3$ ir λ_4 – įtakos koeficientai, esantys intervale $[0, 1]$. Ši hibridinė traukos funkcija toliau naudojama atnaujinti viso sprendimo kintamąjį $r_{ij}(t)$ ir tarp miestinių kelių svorius $c_{ij}(t)$.

2.2. Algoritmų veikimo skirtumai

Pasirinkti trys algoritmai turi daug panašumų, pavyzdžiui, judėjimas pagal vietinių taškų ir visuotinį populiacijos įvertinimą bei išnaudojami atsitiktiniai judėjimai, skirti paskatinti agentus ištirti visą sprendimų erdvę. Vis dėlto šis visuotinas įvertinimas įgyvendintas skirtingais būdais.

Pavyzdžiui, skruzdžių kolonijos išnaudoja feromonus, kuriuose yra informacija apie prieš tai ieškotas vietas. Jonvabalių ir MHSA tokios savybės neturi. Vis dėlto toks informacijos laikymas yra trūkumas, nes pakeitus įvestį dinamiškai, skruzdžių kolonija turi mažą tikimybę iš naujo apieškoti ankščiau ištirtas vietas. Tuo tarpu FA ir MHSA prisitaikys prie naujos įvesties. Visgi dėl šios priežasties skruzdžių kolonija pakankamai gerą sprendimą pasiekia greičiau nei kiti algoritmai, nes atsitiktiniu judėjimu skruzdės negrįžta į jau aplankytas vietas. Tai taip pat reiškia, kad skruzdžių kolonijos algoritmas yra santykinai sudėtingesnis nei FA ir MHSA.

3 lentelė. Metaeuristinių algoritmų savybių palyginimas veikimo principo, greitaveikos, lankstumo ir vaizdų segmentavimo atžvilgiais

Algoritmo pavadinimas	Pagrindiniai komponentai	Lygiagrečių skaičiavimų naudojimas	Dinaminis prisitaikymas prie problemos	Daugiamatį ir netiesinių problemų	Prisitaikymas prie užduoties pokyčių	Informacijos keitimosi išlaidos	Kryžminimas segmentavimui
Uodų taikinių paieška	Hibridinės traukos funkcijos	A	+	⁺ (teor.)	A	Ž	–
Genetinis algoritmas	Chromosomos	Ž	–	–	V	V	–
Atkaitinimo algoritmas	Energijos funkcijos	Ž	±	⁺ (dalin.)	A	Ž	–
Skruzdžių kolonijos optimizavimas	Feromonai	A	–	–	Ž	Ž	–
Dalelių judėjimas	Greičio ir koordinatų modelis	A	–	–	V	V	+
Jonvabalių algoritmas	Šviesio traukos funkcija	A	±	–	V	Ž	+

Žymėjimai: A – aukštas; V – vidutinis; Ž – žemas.

Kaip matyti iš 3 lentelės, kurios duomenys gauti iš (Feng *et al.*, 2013), tik MHSA sugeba susidoroti su daugiamatėmis ir netiesinėmis problemomis. Vis dėlto nagrinėjant vaizdų segmentavimą daugumai algoritmų nereikia kryžminimo (išskyrus jonvabalių ir dalelių spiečių algoritmą). FA pagal veikimo principą grąžina vieną geriausią sprendimo variantą, nebent algoritmas stabdomas iki kol šis yra gaunamas, todėl vaizdų pritaikymui reikalingas kryžminimas (Sharma *et al.*, 2016).

Vienintelis MHSA teoriškai turi galimybę išspręsti daugiamates ir netiesines problemas. Bet algoritmas tokio tipo problemoms nebuvo įgyvendintas, todėl neaišku ar rezultatai bus pakankamai geri (Wang *et al.*, 2017).

Nors FA, ACO ir MHSA gali būti skaičiuojami lygiagrečiai įterpiant tam tikrus pakeitimus algoritme, tai lengviausia padaryti jonvabalių algoritme, nes jonvabaliai bendravimui naudoja tik šviesos intensyvumo duomenis pagrindiniame cikle (Yang, 2010).

2.3. Algoritmų pritaikymas

Šiame poskyriuje aptariame naudojamo atvirojo kodo licencija ir atliktus pakeitimus standartiniams metaeuristiniams algoritmams pritaikant vaizdų segmentavimo užduočiai bei pateikiame jų pseudokodus.

2.3.1. Atvirojo kodo licencija

Kadangi atliekant analitinę literatūros apžvalgą rastas ACO pritaikymas segmentavimo problemai (Jing *et al.*, 2008), naudosime prie straipsnio pridėtą pirminį kodą su BSD-2 sąlygų licencija. Taip pat, modifikuosime bazinį FA ir MHSA, kurie licencijuoti ta pačia BSD-2 sąlygų licencija. Kad nebūtų pažeistos autorių teisių, reikia išsiaiškinti, kokias teises suteikia pirminiame kode patalpinta atvirojo kodo licencija.

Ši programinė įranga apsaugota BSD-2 sąlygų licencijos. BSD licencijų šeima yra labai paprastų ir mažai įpareigojančių kompiuterinės programinės įrangos licencijų klasė, kuri iš pradžių buvo sukurta Berkelio universitete Kalifornijoje. Nors licencija sukurta Kalifornijos universiteto licencija yra viešo naudojimo, todėl ją galima modifikuoti. Ji pirmą kartą buvo naudota 1980 m. BSD UNIX operacinei sistemai licencijuoti. BSD licencijos grupuojamos pagal tai, kiek sąlygų yra licencijoje (Lin *et al.*, 2006).

BSD-2 sąlygų licencijoje reikalaujama, kad būtų perkopijuotas licencijos tekstas ir autoriaus atsisakymas dėl atsakomybės, jei kodas platinamas pirminio kodo forma. Jei kodas bus kompiliuojamas ir platinamas kaip programinė įranga, tuomet tekstas ir atsakomybės atsisakymas įdedamas dokumentacijoje ar kituose šaltiniuose, kuriuos gauna asmuo. Tai reiškia, kad platinant programinę įrangą nebūtina pateikti pirminio kodo (Lin *et al.*, 2006).

Apibendrintant BSD-2 sąlygų licencija reikalauja, kad jei asmenys platina programinę įrangą saugoma licencijos bet kokia forma su pakeitimais ar be jų, platindami turi: įtraukti pirminį autorių teisių įspėjimą, atsakomybės už tinkamą programinės įrangos veikimą atsisakymą ir du paprastus apribojimus. Šie apribojimai yra: nereikėtų teigti, kad asmenys parašė programinės įrangos kodą, jei jo neparašė, ir kūrėjas neatsako, jei programinė įranga veikia ne taip, kaip tikėtasi ar neveikia kaip pageidaujama. Nėra jokių apribojimų apmokestinant išvestinę programinę įrangą ar naudojant BSD-2 sąlygų licenciją su nuosavybine licencija (Lin *et al.*, 2006).

Pagrindinis išvestinių darbų apibrėžimas yra produktas, kurio pagrindas yra vienas ar keli jau esami darbai. Išvestinę programinę įrangą galima nustatyti iš pirminio kodo, net jei jis ir modifikuotas, įskaitant tobulinimą, išplėtimą, pertvarkymą ar vertimą į kitą programavimo kalbą (Lin *et al.*, 2006).

Kadangi ši licencija neturi sąlygų apie pirminio kodo naudojimą kitose licencijose, BSD-2 sąlygų licencija galima jungti su kitomis suderinamomis licencijomis. Taip pat BSD licencija yra pakankamai lanksti ir į ją galima pridėti tam tikrų teisių ar pareigų derinant ją prie tam tikros licencijos.

2.3.2. Skruzdžių kolonijų algoritmo pritaikymas

Nors ir naudojame duomenų rinkinį su apkarpytais vaizdais, juos papildomai modifikuojame programoje. Visuose algoritmuose nuskaitomas vaizdas paverčiamas pilkų tonų vaizdu panaudojant `rgb2gray.m` funkciją. Tuomet nustatomos vaizdo ribos priklausomai nuo vaizdo dydžio, kurios duomenų rinkinio atveju yra nuo 0 iki 120.

Skruzdžių kolonijų algoritme vaizdo vertės toliau normalizuojamos taip, kad būtų intervale nuo 0 iki 1 priklausomai nuo vaizdo intensyvumo ir keičiamos į *double* kintamojo tipą „Matlab“ terpėje. Kadangi vaizdai yra 8 bitų spalvų gilio kiekvieną vaizdo tašką daliname iš 255. Galiausiai ši matrica naudojama, kaip įvestis skruzdžių kolonijų algoritme.

Siekiant segmentuoti vaizdą skruzdžių kolonijų algoritmas skaidomas į keturis žingsnius: inicilizavimo, konstravimo, atnaujinimo ir sprendimo.

A. Inicilizavimo procesas

k skruzdžių atsitiktinai priskiriamos vaizdo I , kurio dydis yra $M \times N$, taškams. Parenkama pradinė feromono matricos vertės $\tau(0)$ konstanta.

B. Konstravimo procesas

Konstravimo žingsnyje viena skruzdė parenkama iš visos skruzdžių kolonijos n -osios iteracijos metu, judėjimui po vaizdą L žingsnių. Skruzdė judės nuo vieno mazgo prie kaimyninio mazgo pagal perėjimo tikimybės formulę:

$$p_{i,j}^n = \frac{(\tau_{i,j}^{n-1})^\alpha \eta_{i,j}^\beta}{\sum_{i,j \in \Omega(l,m)} (\tau_{i,j}^{n-1})^\alpha \eta_{i,j}^\beta}, \quad (8)$$

čia τ_{ij}^{n-1} – feromono reikšmė (i, j) mazge; $\Omega(l, m)$ – kaimyniniai mazgai; $\eta_{i,j}$ – (i, j) mazgo euristinė informacija. Konstantos α ir β nustato feromono matricos ir euristinės matricos įtaką.

Euristinė informacija algoritme nustatoma iš normalizavimo metu gautos lokalios statistikos. Ši informacija renkama iš 8 vaizdo taškų aplink skruzdės padėtį:

$$n_{ij} = \frac{1}{Z} \lambda I_{ij}, \quad (9)$$

čia Z – visų aplinkinių taškų intensyvumo suma arba normalizavimo faktorius; I_{ij} intensyvumo vertė (i, j) vaizdo taške; λ – įtakos konstanta.

C. Atnaujinimo procesas

Feromono matrica yra atnaujinama dviem būdais:

- Pirmasis atnaujinimas atliekamas po skruzdžių judėjimo kiekviename konstravimo žingsnyje. Kiekvienas feromono matricos komponentas atnaujinamas pagal feromonų matricą, jei skruzdė nebuvo aplankiusi k -tojo vaizdo taško pagal formulę:

$$\tau_{i,j}^{n-1} = \begin{cases} (1-\rho) \tau_{i,j}^{n-1} + \rho \Delta_{i,j}^k, & \text{jei } k \text{ skruzdė aplankė tašką } i, j; \\ \tau_{i,j}^{n-1}, & \text{jei neaplanė.} \end{cases} \quad (10)$$

- Antrasis atnaujinimas yra atliekamas pagal visų skruzdžių judėjimą kiekvienos iteracijos konstravimo žingsnyje pagal formulę:

$$\tau^{(n)} = (1-\psi) \cdot \tau^{(n-1)} + \psi \cdot \tau^{(0)}. \quad (4)$$

D. Sprendimo priėmimo procesas

Šiame etape kiekviename vaizdo taške priimamas dvejetainis sprendimas apie tai ar ši vieta yra kraštas ar ne. Tam naudojamas slenkstis T pritaikomas galutinei feromonų matricai. Šis slenkstis gaunamas dinamiškai su `func_seperate_two_class` funkcija. Jei slenksčio vertė didesnė nei feromono (i, j) vaizdo taške vertė laikoma, kad taškas yra kraštas.

Funkcija `func_seperate_two_class` apskaičiuoja globalų slenkstį, kurį galima naudoti su `im2bw` funkcija konvertuoti vaizdo intensyvumą į dvejetainį vaizdą. Gautas slenkstis yra normalizuota intensyvumo vertė, esanti intervale $[0, 1]$. Funkcijoje taikomas Ridler'io ir Calvard'o sukurtas iteracinio slenksčio parinkimo metodas.

Funkcija veikia naudojant histogramą, kuri iš pradžių yra suskirstoma į dvi dalis, naudojant pradinę ribinę vertę gautą iš pusės didžiausios vaizdo reikšmės. Apskaičiuojamas reikšmių, susijusių su objekto vaizdo taškais vidurkis ir reikšmių, susijusių su fono taškais vidurkis. Tada apskaičiuojama nauja slenkstinė vertė, kaip šių dviejų reikšmių vidurkis. Procesas kartojamas, atsižvelgiant į naują slenkstį, iki kol slenkstinė vertė nustoja kisti. Tuomet funkcija grąžina slenksčio reikšmę.

1 programos pseudo kodas. Vaizdų segmentavimas skruzdžių kolonija

```

Nustatomas vaizdo dydis
Nustatomas parametrų koeficientus
While iteracijų skaičius nepasiekia maksimumo
    For nuo 1 iki skruzdžių kiekio
        Įrašomas į atmintį skruzdės kelias
        Pagal kaukę apskaičiuojama tikimybė pereiti į kaimyninį langelį
        If visi kaimyniai langeliai atmintyje
            Skruzdė juda atsitiktinai
        EndIf
        Generuojamas atsitiktinis skaičius
        If skaičius patenka į vietinės tikimybės judėjimo intervalą
            Skruzdė juda pagal vietinę informaciją
        Else
            Skruzdė juda pagal feromonų matricą
    
```

```

EndIf
    Įrašomas į atmintį skruzdės nauja vieta
EndFor
    Atnaujinama feromonu matrica
EndWhile
Naudojama func_seperate_two_class su feromonų matrica
Feromonų matricai atliekamas binarizavimas

```

2.3.3. Jonvabalių algoritmo pritaikymas

Kadangi jonvabalių algoritmui reikalingas kryžminimas jį pritaikant vaizdo segmentavimo problemai naudojama Otsu metodo principo dalis. Pradinės Otsu metodo reikšmės yra labai jautrios ir dėl to parinktos pradinės slenksčio vertės turi didelę įtaką vaizdo segmentavimui. Todėl kryžminant šį metodą jonvabalių algoritmu išsprendžiama lokalaus pakankamai gero, bet ne visuotinai gero, sprendimo problema, nes jonvabalių algoritmas yra globalios paieškos metodas. Pagrindiniai algoritmo žingsniai yra:

Inicilizuojame pradinis parametrus: Nustatomas jonvabalių skaičių $n = 20$, maksimalų traukos koeficientą $\beta_0 = 0,1$, šviesos intensyvumo koeficientą $\gamma = 0,1$ ir kompensacijos faktorių $\alpha = 0,5$. Tikslų funkcija:

$$f(x) = \omega_0(t)\omega_1(t)(\mu_0(t) - \mu_1(t))^2, \quad (11)$$

čia ω_1 ir ω_2 yra tikimybės parodančios kiek tikėtina, kad paimitas taškas atitiks klasę pagal intensyvumą, o μ_1 ir μ_2 yra klasių vidutiniai intensyvumai. Paskaičiavus tikslo funkcijos reikšmę gaunama slenksčiu segmentuoto vaizdo entropija. Pagal gautas tikslo funkcijos reikšmes apskaičiuojamos naujos jonvabalių vietos. Algoritmas kartojamas kol pasiekama mažiausia entropija, kurios metu funkcijos reikšmė pasiekia minimumą. Pagal tikslo funkciją jonvabaliai juda link mažiausios entropijos reikšmės. Jonvabalių patrauklumo ir jonvabalių vietų atnaujinimo formulės jau buvo aptartos ankstesniame poskyriuje apie bendrą algoritmo veikimo principą.

Jei pasiekiamas paieškos užsibrežtas tikslumas arba maksimalus iteracijų skaičius $N_{\max} = 50$ stabdome jonvabalių algoritmą. Jonvabalis, kurio tikslo funkcijos vertė paskutinėje iteracijoje yra mažiausia ir bus geriausia algoritmo vertė. Šią vertę naudojame im2bw metode slenksčiui nustatyti.

2 programos pseudo kodas. Vaizdų segmentavimas jonvabalių algoritmu

```

Nustatomi parametrai ir generuojamos pradinės jonvabalių vietos
Sudaroma vaizdo intensyvumo histograma
While iteracijų skaičius nepasiekia maksimumo
    Apskaičiuojamos jonvabalių klasių tikimybės ir vidutinės vertės
    Apskaičiuojama jonvabalių tikslo funkcija
    Pagal tikslo funkcijos vertes rūšiuojami jonvabaliai
    Atnaujinamos jonvabalių vietos
    If pasiektas užsibrežtas tikslumas
        Break išeiti iš ciklo
    EndIf
EndWhile
Atrenkama mažiausia tikslo funkcijos vertė

```

Naudojama `im2bw` funkcija su gauta tikslo funkcijos verte

2.3.4. Uodų taikinių paieškos algoritmo pritaikymas

Šiam algoritmui sudėtingų modifikacijų nereikia. Pagrindiniai pakeitimai nuo standartinio algoritmo skiriasi tuo, kad standartiniam algoritmui reikia pakeisti įvestį ir apdoroti išvestį.

Įvesties vaizdai iš pradžių sumažinamas mastelis su funkcija `imresize`, kuria sumažinamas įvesties vaizdas keturiais kartais. Tuomet apskaičiuojamas šio vaizdo gradientas funkcija `imgradient`, norint suformuoti informaciją apie kraštus. Šis gradientas normuojamas pagal maksimumą ir sudaroma c matrica. Jei taškai i ir j sutampa pagal algoritmą į matricą įrašome reikšmę 10. Pasitelkiant c matricą algoritmas bandys apjungti kraštus į kontūrus, o neatitinkančius kraštų taškus ir triukšmą slopins.

Kaip minėta pagrindinis algoritmas, kurį jau aptarėme 2.1.3 skirsnyje, nemodifikuojamas, bet norint vizualiai atvaizduoti kitimą iteracijų metu išrenkame maksimalias c matricos stulpelių reikšmes ir pakeičiame matricą į 30×30 . Tą pačią operaciją atliksime ir pasibaigus pagrindiniam ciklui ir pakeisime mastelį į 120×120 ir pritaikysime funkciją `func_seperate_two_class` norint gauti išvesties rezultatą.

3 programos pseudo kodas. Vaizdų segmentavimas uodų taikinių paieškos algoritmu

```
Nustatomi parametrai
Vaizdo dydis sumažinamas keturiais kartais
Apskaičiuojamas vaizdo gradientas
Normuojamas vaizdo gradientas
While iteracijų skaičius nepasiekia maksimumo
    Apskaičiuojama hibridinė traukos funkcija
    Atnaujinamos uodų vietos
    Atnaujinamos svorių matricos
    If pasiektas užsibrėžtas tikslumas
        Break Išeiti iš ciklo
    EndIf
EndWhile
Išrenkamos didžiausios svorių matricos stulpelių reikšmės
Svorių matricos dydis padidinamas keturiais kartais
Naudojama func_seperate_two_class su svoriu matrica
Svorių matricai atliekamas binarizavimas
```

2.4. Skyriaus apibendrinimas

Skyriaus pradžioje išnagrinėti standartinių metaeuristinių algoritmų veikimo principai, siekiant atskleisti veikimo skirtumus. Algoritmų veikimo principai turi panašumų, tačiau pasižymi šiomis skirtingomis savybėmis:

- ACO algoritmas sudėtingesnis nei MSHA ir FA;
- ACO algoritmas pasižymi didesniu atsparumu užstrigti ties vietiniu sprendimu nei kiti algoritmai;

- ACO algoritmas ir MHSA gali dirbti su segmentavimo problemomis nekeičiant veikimo principo, o FA reikalingas kryžminimas norint spręsti segmentavimo uždavinius.
- FA teoriškai turėtų geriausiai išnaudoti išteklius lygiagrečiams skaičiavimams.

Vėliau aptarti pakeitimai atlikti metaeuristiniams algoritmams pritaikant vaizdų segmentavimui. ACO algoritmui pritaikoma įvestie norint segmentuoti tinkamo dydžio vaizdus. FA papildomai atliktas kryžminimas su Otsu metodu. MHSA panaudojamas vaizdo mastelio keitimas ir funkcija `func_seperate_two_class` norint išskirti kontūrus.

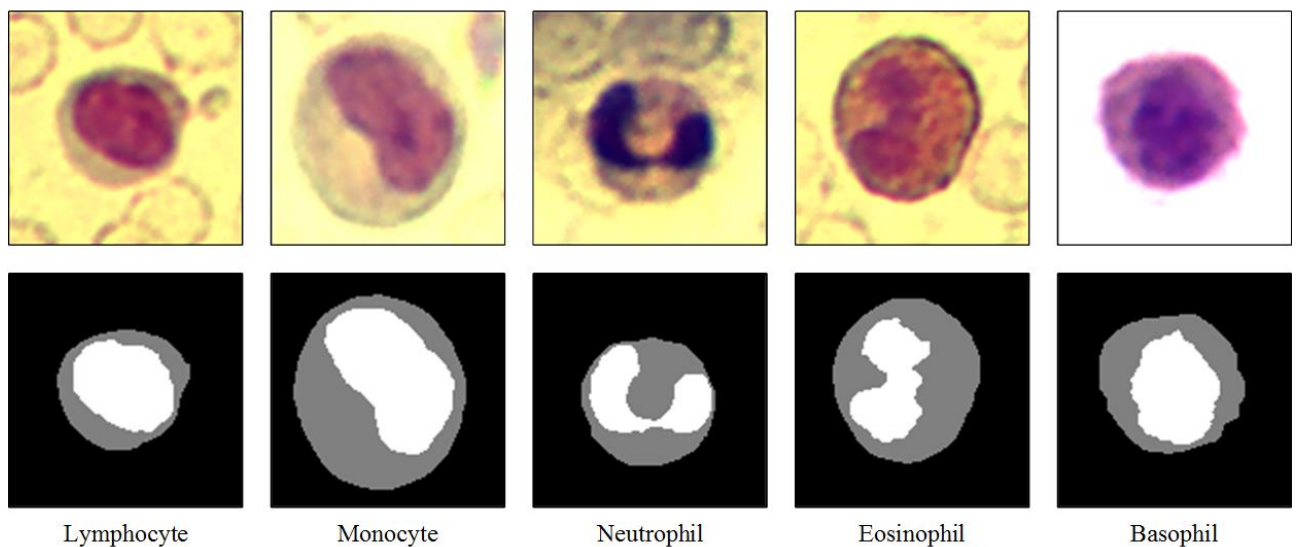
3. DUOMENŲ APDOROJIMO METODIKA IR VAIZDŲ ATRANKA

Skyriaus pradžioje aptariama biomedicininį vaizdų rinkinio atranka. Toliau pristatomos su medicininiais vaizdais atliktos operacijos, taikytos prieš vaizdus paduodant metaeuristiniams algoritams. Taip pat aptariami etalono apdorojimo, norint gauti tyrimui analizuoti skirtus duomenis, klausimai. Pabaigoje pristatoma tyrimo metodika ir rezultatų vertinimo parametrai.

3.1. Biomedicininį vaizdų atranka

Darbe taikomas vienas iš baltųjų kraujo kūnelių vaizdų rinkinių, naudotas „Fast and robust segmentation of white blood cell images“ straipsnyje (Zadeh, 2011). Straipsnio vaizdų rinkinys gautas iš „Jiangxi Tecom Science“ kompanijos. Šis rinkinys apdorotas taikant atraminių vektorių mašinas (Zheng *et al.*, 2018) ir gradiento vektorių srauto kontūrų metodu (Ko *et al.*, 2011), todėl straipsnyje gauti rezultatai gali būti palyginti su šiame darbe taikomų metaeuristinių algoritmų efektyvumu.

Vaizdų rinkinio etalonai (sričių kontūrai) suformuoti rankiniu būdu, remiantis dalykinės srities ekspertų (hematologų) nuomone. Etalone išskiriami trijų sričių kontūrai: branduolio, citoplazmos ir fono žymint atitinkamai balta, pilka ir juoda spalvomis. Vaizdų rinkinyje vaizdai suklasifikuoti į 5 grupes: limfocitus, monocitus, neutrofilus, eozonofilus ir bazofilus, tačiau šiame tyrime į tai neatsižvelgta, t. y. vykdyta atsitiktinė atranka iš viso vaizdų rinkinio. Rinkinio vaizdai gauti naudojant „Motic Moticom Pro 252A“ optinį mikroskopą ir N800-D kamera su automatine fokusavimo funkcija. Tepinėliai apdoroti naudojant hematologinį reagentą baltųjų kraujo kūnelių dažymui.



2 pav. Baltųjų kraujo kūnelių vaizdų ir jų etalonų pavyzdžiai (Zadeh, 2011)

2 pav. pateikti vaizdų rinkinio ir jų etalonų pavyzdžiai. Vaizdų dydis – 120×120 vaizdo taškų su 24 b RGB spalvų gyliu, saugomi .bmp bylos formate. Šis vaizdų rinkinys sudarytas išskarpant

80 didesnių vaizdų. Etalono vaizdai – tokio pat dydžio tik 8 b pilkų pustonių spalvų gylis, saugomi .png bylos formatu. Straipsnio vaizdų rinkinyje iš viso yra 300 baltųjų kraujo kūnelių vaizdų.

Siekiant palyginti metaeuristinius algoritmus, iš šio rinkinio atsitiktinai (su sąlyga, kad bus bent vienas vaizdas iš kiekvienos baltųjų kraujo kūnelių grupės) išrinkta 100 vaizdų.

3.2. Vaizdų apdorojimas

Vaizdams nuskaityti ir perduoti naudojamas palyginimo skriptas, kuriame paduodami vaizdų pavadinimai funkcijoms, kuriose įgyvendinti metaeuristiniai algoritmai. Nors ir naudojamas duomenų rinkinys su apkarpytais vaizdais, jie papildomai modifikuoti skripte. Visuose algoritmuose nuskaitytas vaizdas paverčiamas pilkų tonų vaizdu panaudojant „Matlab“ `rgb2gray.m` funkciją. Tuomet nustatomos vaizdo ribos priklausomai nuo vaizdo dydžio, nors rinkinyje yra tik 120×120 .

Vaizdo vertės toliau normalizuojamos taip, kad būtų intervale nuo 0 iki 1 priklausomai nuo vaizdo intensyvumo ir keičiamos į *double* kintamojo tipą „Matlab“ terpėje. Kadangi vaizdai yra 8 b spalvų gilio, tam, kad būtų pasiektas normalizavimas, kiekvieną vaizdo tašką daliname iš 255. Galiausiai ši matrica naudojama, kaip įvestis aptartiems metaeuristiniams algoritmams.

Kadangi MHSA algoritmo pagrindinis ciklas dirba su $14\,400 \times 14\,400$ matrica, iš jos išrenkama maksimalios vertės kiekviename stulpelyje ir pertvarkomos į 120×120 matricą. ACO ir MHSA algoritmų pagrindiniai ciklai grąžina 120×120 matricas, bet jos nenaudojamos, kaip galutinis segmentavimo rezultatas, nes vertės yra *double* tipo ir turi taškų su nedideliu triukšmu. Todėl papildomai naudota funkcija `func_seperate_two_class`, kuria apskaičiuojama globalus slenkstis. Šiuo globaliu slenksčiu apdorojame abiejų algoritmų pagrindinio ciklo rezultatus panaudojant `im2bw` funkciją konvertuoti vaizdo intensyvumą į dvejetainį vaizdą.

Vos tik baigiamas darbas metaeuristinių algoritmų funkcijose, skripte nuskaitytas etalonas, kuris apdorojamas skaičiuojant gradientus pagal x ir y ašis. Tai reikia atlikti norint išgauti kraštus, nes ACO algoritmas grąžina, kaip atsakymą kraštų kontūrus. Šių etalonų gradientus pagal x ir y ašis vektoriškai sumuojami ir taip gauname lyginimui reikalingą etaloną.

Iš funkcijų gautų išvesčių ir etalono gradiento ašių sumos išskaičiuojami tikrasis teigiamas (angl. *True Positive*, *TP*), klaidingai teigiamas (angl. *False Positive*, *FP*), klaidingai neigiamas (angl. *False Negative*, *FN*) ir tikrasis neigiamas (angl. *True Negative*, *TN*) vertės. Tikrosios teigiamos vertės tai taškų skaičius, kuriuose etalone ir algoritmo rezultate žymimas baltasis kraujo kūnelis. Klaidingai teigiamų vertės yra taškų skaičius, kuriuose turėtų būti fonas, o algoritmo rezultate taškas blogai nustatytas, kaip baltasis kraujo kūnelis. Klaidingai neigiamos vertės tai taškų skaičius, kuriuose etalone baltasis kraujo kūnelis, bet algoritmo rezultate yra fonas. Tikrosios neigiamos vertės tai toks taškų skaičius, kuriuose taškai etalone ir rezultate yra pažymėti kaip fonas.

3.3. Tyrimo metodika

Pagrindinis šio tyrimo tikslas išsiaiškinti algoritmų tikslumą ir greitaveiką bei lygiagrečių skaičiavimų pajėgumą bei juos palyginti, kad būtų galima pateikti rekomendacijas. Šiame poskyryje aptarta, kaip įvertinami rezultatai ir kodėl pasirinkti tam tikri palyginimo parametrai.

3.3.1. Tikslumo vertinimas

Tikslumui įvertinti naudotas jau aptartas duomenų rinkinys su 100 vaizdų ir tiek pat etalonų. Tam sukurtas skriptas `palyginimast.m`, kuris paleistas kompiuteryje su „AMD“ Ryzen r5 1600 ir 16 GB operatyviosios atminties. Naudotas tik vienas kompiuteris, nes tikslumas neturi keistis esant skirtingam kompiuterio našumui.

Tikslumui įvertinti pasirinkti aštuoni parametrai. Keturi iš jų yra įgyvendintos duomenų rinkinyje su baltųjų kraujo kūnelių atvaizdais (Zadeh, 2011) straipsnyje. Kad galėtume palyginti, įvertintos ir straipsnyje naudotas segmentavimo kokybės funkcijas. Nors straipsnyje yra keli algoritmai, palyginimui naudosime SVM ir GVF (Ko *et al.*, 2011). SVM parinktas dėl to, kad buvo geriausias algoritmas tarp lygintų straipsnyje (Zheng *et al.*, 2018), o GVF parinktas, nes yra geriausias iš apmokymo nenaudojančių algoritmų.

Šio segmentavimo kokybę įvertinta pagal tris sritimis grįstus parametrus: per daug segmentuoto vaizdo paklaidą (angl. *Over-segmentation Rate*, OR_p), nepakankamai segmentuoto vaizdo paklaidą (angl. *Under-segmentation Rate*, UR_p) ir bendrą klaidų lygį (angl. *Overall error Rate*, ER_p). Šiomis sritimis grįsti parametrai apskaičiuojami pagal formules:

$$OR_p = \frac{FN}{FN + TP}; \quad (1)$$

$$UR_p = \frac{FP}{FP + TP}; \quad (2)$$

$$ER_p = \frac{FN + FP}{TP + TN + FP + FN}, \quad (3)$$

Taip pat minėtame straipsnyje (Zheng *et al.*, 2018) apskaičiuota santykinė nuotolio paklaida (angl. *Relative Distance Error*, RDE). Santykinę nuotolio paklaidą galime įvertinti, kiek taškų vidutiniškai algoritmai nukrypsta nuo etalono. Sakykime, kad $e_1, e_2, \dots, e_n$ bei $t_1, t_2, \dots, t_n$ yra taškai iš gautų algoritmo kontūro taškų E ir tikslo kontūro taškų T matricių, o n_e ir n_t taškų skaičius E ir T matriciose. Tuomet santykinė nuotolio paklaida gaunama pagal formules:

$$RDE = \frac{1}{2} \left(\frac{1}{n_e} \sum_{i=1}^{n_e} d_{\min}(e_i, t) + \frac{1}{n_t} \sum_{i=1}^{n_t} d_{\min}(t_i, e) \right), \quad (4)$$

$$d_{\min}(o, r) = \min \sum_{i=1} d_E(o, r_i), \quad (5)$$

čia $d_{\min}$ yra minimalus Euklidinis atstumas tarp taško ir taškų aibės, o e_i bei t_i – i -tasis etalono ir segmentuoto vaizdo taškas. Euklidinis atstumas d_E apskaičiuotas pagal formulę:

$$d_E(q, p) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2}. \quad (6)$$

Kuo mažesnės keturių segmentavimo klaidų įvertinimo funkcijų vertės, tuo geriau algoritmas segmentuoja vaizdą lyginant su etalonu. Šias funkcijas panaudojome lygindami rezultatus su etalono rinkiniu.

Be aptartų parametrų taip pat apskaičiuoti jautrumą (angl. *Sensitivity*), specifiškumą (angl. *Specificity*), taiklumo (angl. *Accuracy*) ir neigiamos nuspėjamosios vertės (angl. *Negative Predictive Values*, *NPV*) parametrai. Šie parametrai apskaičiuojami pagal formules:

$$SENS = \frac{TP}{TP + FN}; \quad (7)$$

$$SPEC = \frac{TN}{TN + FP}; \quad (8)$$

$$ACC = \frac{TP + TN}{P + N}; \quad (9)$$

$$NPV = \frac{TN}{TN + FN}, \quad (10)$$

čia P yra teisingai ir neteisingai nustatytų baltųjų kraujo kūnelių taškų suma, o N teisingai ir neteisingai nustatytų fono taškų suma.

Jautrumas pasirinktas norint įvertinti teisingai ir klaidingai nustatytų kraštų santykį. Specifiškumas pasirinktas vertinant algoritmo gebėjimą atmesti klaidingus taškus, t. y. taškus, kurie yra fonas medicininame vaizde. Taiklumas pasirinktas tam, kad būtų galima įvertinti, kiek toli nuo etalono yra gauti algoritmo rezultatai, o neigiama nuspėjamoji vertė – įvertinti, kaip gerai algoritmai atskiria fono taškus nuo baltojo kraujo kūnelio.

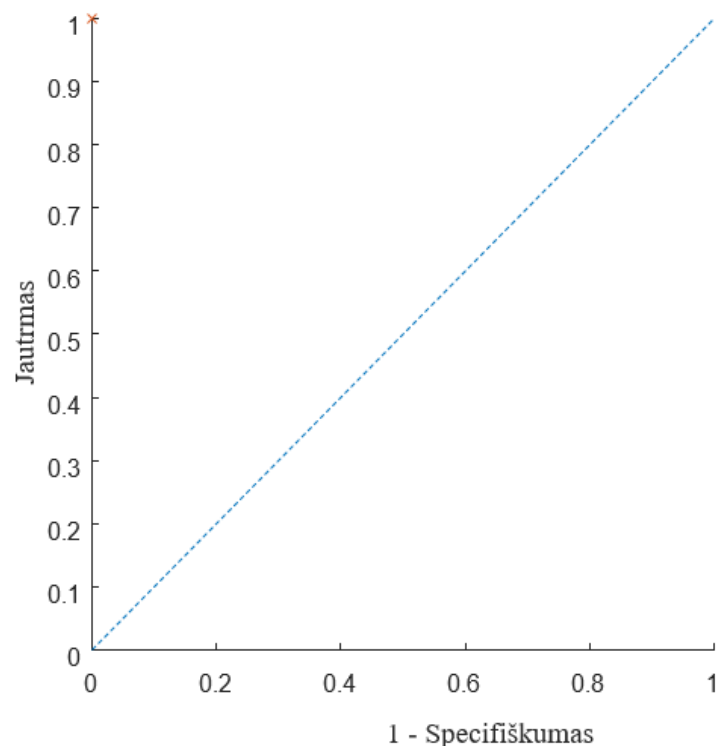
Kadangi skriptas skaičiuoja didelį kiekį duomenų iš gautų rezultatų sudarysime statistikas naudodami vidurkį ir standartinę nuokrypį kiekvienam parametrai. Standartinę nuokrypį skaičiuosime pagal formulę:

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2}, \quad (11)$$

čia x_i – i -tasis parametro rezultatas; $\bar{x}$ – vidutinė parametro vertė; N – segmentuotų vaizdų skaičius naudotas įvertinti algoritmams.

Parametrų pateikti turėsime dvi lenteles ir du paveikslėlius, kuriuose pateiksime standartinius nuokrypius. Vienoje iš lentelių pateiksime metaeuristinių algoritmų rezultatų palyginimą su SVM (Zheng *et al.*, 2018) ir GVF (Ko *et al.*, 2011). Lyginamos vidutinės OR_p , UR_p , ER_p ir RDE parametrų reikšmės ir jų standartinis nuokrypis. Parametrų reikšmės yra intervale nuo 0 iki 1 ir nuo mažesnės reikšmės tuo geresnis algoritmo rezultatas. Taip pat šioje lentelėje žaliu fonu išskiriama geriausia vidutinė parametro vertė. Kadangi lentelėje duomenis sunku analizuoti, pridėtas paveikslėlis su juostine diagrama. Šioje diagramoje vizualiai pavaizduoti skirtumai tarp minėtų parametrų, įtraukiant standartinį nuokrypį. Taip lengviau įvertinti, kuris algoritmas statistiškai nesiskiria tam tikrų parametrų atžvilgiu, nes matyti standartinio nuokrypio intervalo persidengimas.

Kitoje lentelėje tarpusavyje lyginti metaeuristiniai algoritmai su jautrumo, specifiškumo, taiklumo ir neigiamos nuspėjamos vertės parametrais. SVM ir GVS šioje lentelėje neįtrauktas, nes šių parametrų straipsnyje (Zheng *et al.*, 2018) neapskaičiuotos. Parametrų reikšmės yra intervale nuo 0 iki 1 ir kuo didesnės reikšmės tuo geresnis algoritmo rezultatas šių parametrų atžvilgiu. Lentelėje nurodomos vidutinės ir standartinio nuokrypio parametrų vertės bei geltonu fonu pažymima geriausios vidutinės vertės. Prie lentelės papildomai pridėdame ir paveikslėlį su dar viena juostine diagrama, kurioje vizualiai išreikštos parametrų vertės.



3 pav. ROC erdvės pavyzdys su kryžiuoku pavaizduotu tobulo algoritmo tašku ir atsitiktinį spėjimą atitinkanti įstrižai kertanti tiesė

Galiausiai pavaizduoti visi 100 rezultatų ROC erdvėje, kurios pavyzdys pavaizduotas 3 paveikslėlyje. ROC erdvės grafike parodoma jautrumo ir specifiškumo ryšys. Todėl vizualiai galima matyti ar algoritmas yra geresnis už atsitiktinį spėjimą. ROC erdvėje atsitiktinis spėjimas būtų

atvaizduojamas, kaip įstrižai grafiką kertanti tiesė. Todėl jei algoritmo taškai patenka į šią grafiko dalį, algoritmas veikia tokiu pat tikslumu, kaip ir atsitiktinis spėjimas. Bet jei algoritmo taškai patenka virš tiesės algoritmas yra geresnis nei atsitiktinis spėjimas, o jei žemiau blogesnis. Tobulai segmentuojantis algoritmas ROC erdvėje būtų pažymėtas taške (0,1).

Siekiant pašalinti pradinių verčių įtaka rezultatams, kiekvienam vaizdui kiekvienas algoritmas paleidžiamas po 3 kartus. Parametrus skaičiuoti išsaugomas geriausias iš 3 paleidimų rezultatas. Išrinkti geriausiam rezultatui naudojamas taiklumo parametras.

3.3.2. Greitaveikos ir lygiagretumo vertinimas

Greitaveikai įvertinti naudoti keli kompiuteriai ir „Matlab“ funkcijos laikui matuoti. Šių funkcijų skaičiavimai įgyvendinti „Matlab“ skripte `palyginimast.m`. Skirtingai negu tikslumo eksperimente, rinkinys šioje dalyje sumažintas iki 20 vaizdų.

„Matlab“ skriptą `palyginimasg.m` paleistas 3 kompiuteriuose, turinčiuose įvairią komplektaciją: „AMD“ Ryzen r5 1600 3,6 GHz 6 branduolių procesoriumi su 16 GB operatyviosios atminties, „Intel“ Core i5 2500 3,7 GHz 4 branduolių procesoriumi su 8 GB operatyviosios atminties ir „Intel“ Core i5 5200u 2,7 GHz 2 branduolių procesoriumi su 4 GB operatyviosios atminties. Vertinti specialiai pasirinkti skirtingi kompiuteriai su skirtingų gijų skaičiumi ir operatyviosios atminties dydžiu. Taip galima įvertinti gijų skaičių poveikį skaičiavimams, nes išvardyti procesoriai turi skirtingą branduolių skaičių.

„Matlab“ skripte `palyginimasg.m` skaičiavimų greitis yra gaunamas iš sukurtų metaeuristinių algoritmų funkcijų. Laiko skaičiavimas gaunamas algoritmų funkcijose pasitelkiant „Matlab“ `tic` ir `toc` funkcijas, kuriomis išmatuojamas laikas, skirtas atlikti vieno vaizdo segmentavimą. Šis laikas pradedamas, kai nuskaitomas ir apdorojamas vaizdas, ir baigiamas, kai algoritmas gauna galutinį rezultatą, todėl `tic` ir `toc` funkcijos įgyvendintos pačiose algoritmų funkcijose. Į šį laiką neitraukiamas parametras, reikalingų įvertinti algoritmų tikslumą, skaičiavimo laikas. Algoritmai baigs darbą, kai pasieks maksimalų iteracijų skaičių, tam kad nesusidurtume su per anksti gautu pakankamai geru sprendimo rezultatu. Iš gautų algoritmo veikimo reikšmių apskaičiuojamas vidurkis ir standartinis nuokrypis, kuris naudojamas greitaveikai įvertinti.

Rezultatus atvaizduosime lentelėje, kurio stulpeliuose metaeuristiniai algoritmai, o eilutės naudoti kompiuteriai. Segmentavimo laikas matuojamas sekundėmis ir išreiškiamas bandymų vidurkiu bei standartinio nuokrypio reikšmėmis.

3.4. Skyriaus apibendrinimas

Šiame skyriuje aptartas naudojamas baltųjų kraujo kūnelių vaizdų rinkinys. Rinkinys parinktas iš „Fast and robust segmentation of white blood cell images“ (Zadeh, 2011) straipsnio, nes turi gerą

etaloną ir yra įgyvendintas su keliais algoritmais. Iš šio rinkinio vaizdo atrinkti 100 vaizdų tikslumo eksperimentui, kurių apdorojimas aptartas antrame poskyriuje. Apdorojami vaizdai verčiami į pilkų pustonių vaizdus. Etalonas apdorojamas, kaip gradiento vaizdas norint palyginti jį su algoritmų gautomis išvestimis, kurios taip pat apdorotos slenksčiu.

Tuomet aptarta tyrimo metodika, kurioje apžvelgti aštuoni pasirinkti parametrai tikslumui nustatyti: per didelio segmentavimo, per mažo segmentavimo paklaidos, bendras klaidų lygis, santykinė nuotolio klaida, jautrumas, specifiškumas, taiklumas ir neigiama nuspėjamoji vertė. Taip pat apskaičiuojamos šių parametrų vidutinės ir standartinio nuokrypio vertės. Tikslumui nustatyti eksperimentas atliekamas su 100 vaizdų. Greitaveikai ir lygiagretumui įvertinti tarp algoritmų aprašytas atskiras eksperimentas su trimis skirtingais kompiuteriais. Šiame eksperimente algoritmai leidžiami vienodą iteracijų skaičių su 20 vaizdų ir matuojamas algoritmo vykdymo laikas.

4. METODŲ TYRIMO REZULTATAI

Skyriuje aptariami tikslumo, greitaveikos ir lygiagretumo eksperimentų rezultatai, norint iširti metaeuristinių algoritimų skirtumus. Vėliau pateikiamos rezultatais grįstos metodų rekomendacijos.

4.1. Tyrimo rezultatai

Šiame poskyryje aptariame gautus tyrimo rezultatus lygindami juos tarpusavyje. Įvertinami 8 tikslumo parametrai, ROC erdvė, greitaveika ir skaičiavimų lygiagretumas.

4.1.1. Tikslumo vertinimo rezultatai

Šiame poskyryje pateikiami, pagal 3.3.1 skirsnyje pateiktą tyrimo metodiką, gauti tikslumo vertinimo rezultatai. Taip pat šiuos rezultatus pateikiame lentelių ir grafikų formomis siekdami lengviau analizuoti rezultatus.

4 lentelė. Metaeuristinių, SVM ir GVF algoritimų tikslumo eksperimento rezultatai

Parametras	FA	ACO	MHSA	SVM	GVF
OR_p	$0,296 \pm 0,217$	$0,522 \pm 0,089$	$0,0657 \pm 0,0537$	$0,0187 \pm 0,01$	$0,0517 \pm 0,015$
UR_p	$0,0246 \pm 0,0614$	$0,416 \pm 0,072$	$0,169 \pm 0,108$	$0,0131 \pm 0,01$	$0,0731 \pm 0,015$
ER_p	$0,17 \pm 0,138$	$0,0779 \pm 0,0188$	$0,134 \pm 0,08$	$0,0318 \pm 0,04$	$0,141 \pm 0,3$
RDE	$11,3 \pm 2,6$	$15,6 \pm 2,7$	$13,6 \pm 3,1$	$1,79 \pm 0,5$	$6,74 \pm 2$

Fonu pažymėtos mažiausios atitinkamų parametrų vertės.

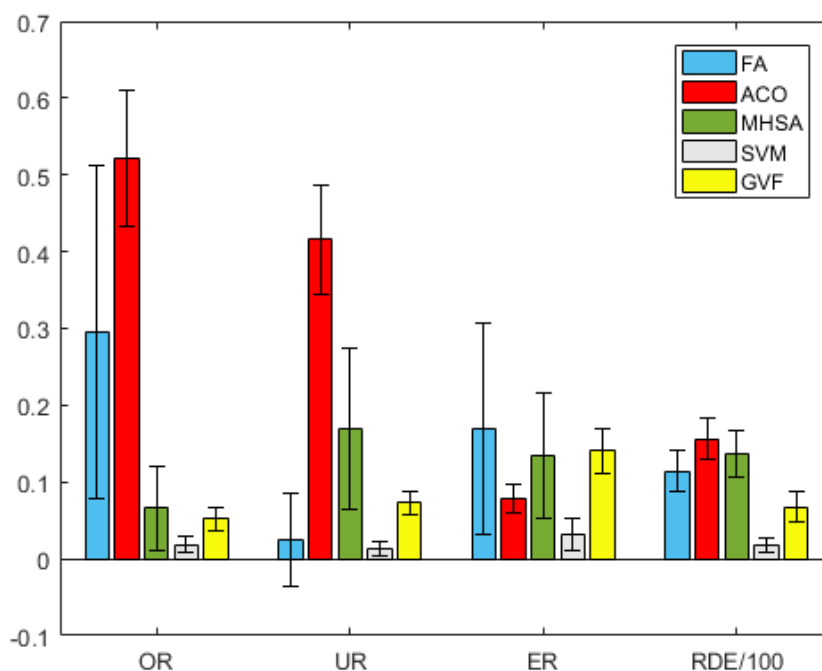
Kaip matyti iš 4 lentelės geriausius vidutinius OR_p rezultatus gavo SVM. Tai reiškia, kad algoritmas neteisingai į baltųjų kraujo kūnelių kontūrą įtraukia mažiausią skaičių fono taškų. Lyginant likusius algoritmus šiuo parametru, matyti, kad GVF ir MHSA yra panašūs. Tuo tarpu FA parodė prastą rezultatą, todėl, jei vaizdo segmentavime reikalingas, kuo mažesnis į baltojo kraujo kūnelio kontūrą įtrauktų fono taškų skaičius, FA ir ypač ACO netinka.

Žvelgiant į UR_p rezultatus, su SVM gauti geriausi rezultatai, bet FA vidutine verte yra arti šio apmokymo reikalaujančio algoritmo. Nors GVF gautas neblogas rezultatas FA šiuo parametro atžvilgiu pranašesnis, nors GVF turi geresnį balansą tarp OR_p ir UR_p parametrų. MHSA rezultatas rodo, kad algoritmas įtraukia daugiau fono taškų į baltųjų kraujo kūnelių taškų kontūrą, todėl jo OR_p geresnis nei UR_p . Su ACO gauti prasčiausias UR_p rezultatas.

Analizuojant ER_p vertes matyti, kad SVM pasirodė geriausiai. Visgi su ACO gauti pakankamai geri rezultatai. MHSA ir GVF algoritimų ER_p vertės yra panašios ir juos skiria tik pora procentų. FA rezultatas, nors ir yra pakankamai arti MHSA ir GVF, vis dėlto buvo prasčiausias.

RDE parametras rodo kiek arti etalono taškų yra segmentuotas baltojo kraujo kūnelio kontūras. Su SVM gauti 3 kartus geresnės šio parametro vertės lyginant su GVF. Metaeuristiniais algoritmais

gauti apie 2 kartus prastesni rezultatai negu su GVF. Taip yra todėl, kad metaeuristiniai algoritmai įtraukia per daug fono taškų į rezultatą arba neįtraukia pakankamai baltojo kraujo kūnelio taškų.



4 pav. Metaeuristinių, SVM ir GVF algoritmų tikslumo rezultatų palyginimas

Kaip matyti iš 4 pav. SVM algoritmas yra tikslesnis, ką ir rodo visi palyginti parametrai įvertinant standartinę nuokrypį. Vis dėlto šiam algoritmui reikalingas apmokymas, todėl toliau lyginsime algoritmus, kuriems apmokymas nereikalingas. Remiantis OR_p , MHSA kraujo kūnelio taškus atskiria panašiu tikslumu kaip ir GVF, bet atsižvelgiant į MHSA standartinę nuokrypį, rezultatas prastesnis. Kita vertus FA įtraukia daugiau negu reikia taškų į rezultatą remiantis UR_p vertėmis, nors vidurkio UR_p vertė yra geresnė už GVF standartinio nuokrypio reikšmę rodo, kad algoritmai panašūs, todėl nėra aišku, kuris algoritmas pranašesnis. Su ACO gauti rezultatai – prasti ir standartinis nuokrypis nėra didelis, nors ER_p ganėtinai žemas, OR_p ir UR_p vertės rodo, kad lieka daug taškų kurie turėtų būti atrinkti, bet nėra. RDE rodo, kad visi metaeuristiniai algoritmai nukrypsta vidutiniškai bent po 10 taškų nuo etalono. Net įvertinant GVF algoritmo standartinę nuokrypį, jis yra statistiškai pranašesnis.

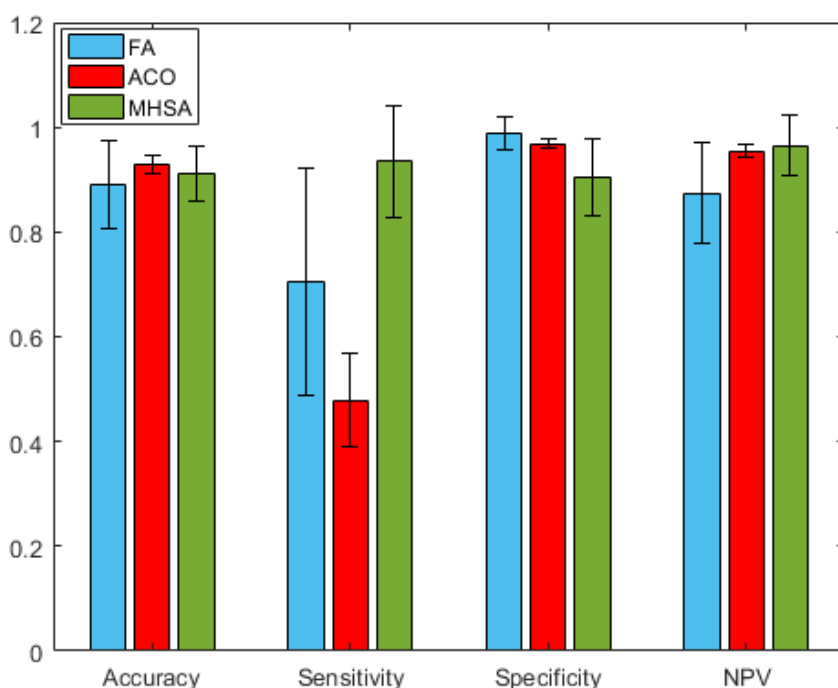
5 lentelė. Metaeuristinių algoritmų tikslumo vertinimo eksperimento rezultatai

Parametras	FA	ACO	MHSA
ACC	0,892±0,084	0,929±0,016	0,912±0,053
SENS	0,704±0,217	0,478±0,089	0,9343±0,108
SPEC	0,989±0,033	0,969±0,01	0,905±0,075
NPV	0,874±0,097	0,954±0,012	0,965±0,054

Fonu pažymėtos didžiausios atitinkamų parametrų vertės.

Rezultatai 5 lentelėje rodo, kad MHSA pasižymi geriausiu vidutiniu jautrumu. Šis parametras rodo teisingai ir klaidingai nustatytų baltųjų kraujo kūnelių taškų skaičiaus santykį. Ypač prasti rezultatai gauti ACO algoritmu, nes jie artėja prie atsitiktinio spėjimo jautrumo parametro atžvilgiu. FA gautas geriausias vidutinis specifiškumo rezultatas, todėl, kad algoritmas teisingai nustato fono taškus, kurie neturėtų būti kontūre.

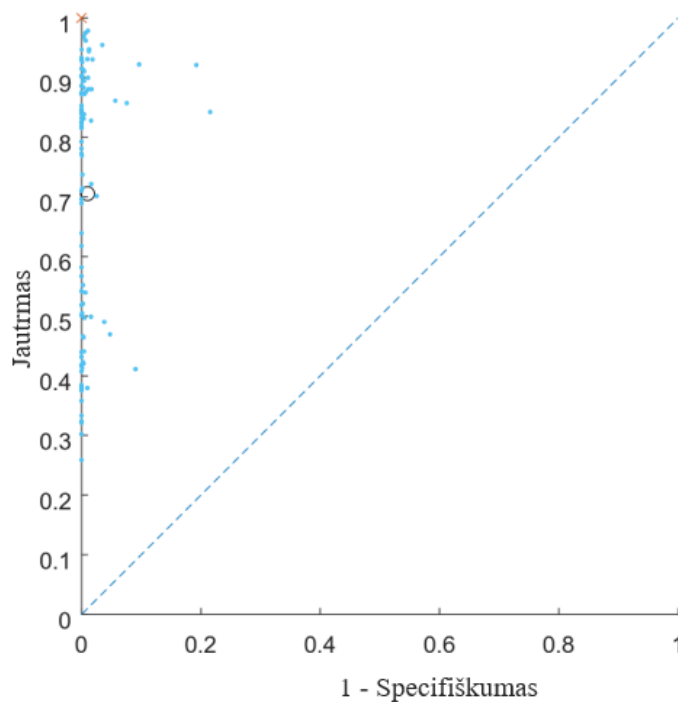
Taiklumas rodo segmentavimo rezultatų pasiskirstymą ir iš vidutinės vertės matyti, kad su ACO gautos vertės mažiausiai kinta prie skirtingų baltųjų kraujo kūnelių vaizdų. Iš NPV rezultato, matyti, kaip gerai algoritmas atskiria foną nuo kontūro. Su MHSA gauta vidutinė NPV vertė geriausia, o su FA – labai bloga. Tai reiškia, kad FA neatskiria fono etalone ir kai kuriais atvejais jį žymi, kaip baltojo kraujo kūnelio taškus.



5 pav. Metaeuristinių algoritmų tikslumo vertinimo eksperimento rezultatų palyginimas

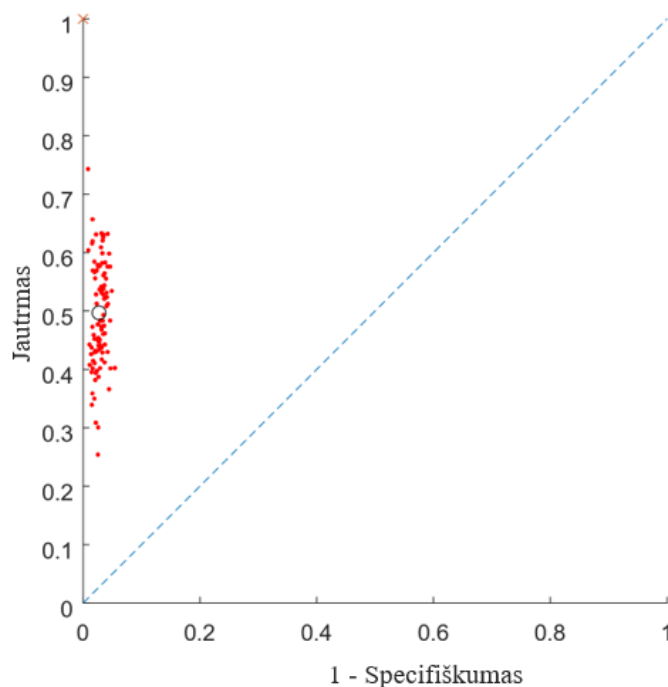
Kaip matyti iš 5 pav. sunku nustatyti, įvertinant standartinį nuokrypį, kuris algoritmas yra taiklesnis. Statistiškai MHSA gautas jautrumas yra geresnis nei ACO, bet FA rezultatą įvertinti sudėtinga, dėl didelio standartinio nuokrypio. Specifiškumo standartiniai nuokrypiai taip pat sudaro statistinę vertinimo problemą. FA šiuo atveju stipriai kito prie skirtingų vaizdų, kas reikštų, kad jo rezultatus būtų įmanoma patobulinti, keičiant koeficientų vertes.

Remiantis specifiškumu, ACO ir FA pakankamai gerai atskiria foną ir neturi tokio didelio triukšmo, kaip MHSA. NPV parametras rodo, kad MHSA yra itin tikslus. Tai tik patvirtina OR_p parametro rezultatą, o tai reiškia, kad MHSA atskiria foną, bet turi daug neteisingai fono taškų pažymėtų, kaip baltieji kraujo kūneliai. Vis dėlto šiuo parametru nėra įvertinamas triukšmas, todėl kiti algoritmai turėjo geresnius rezultatus negu OR_p .



6 pav. Jonvabalių algoritmo (FA) 100 rezultatų ROC erdvė

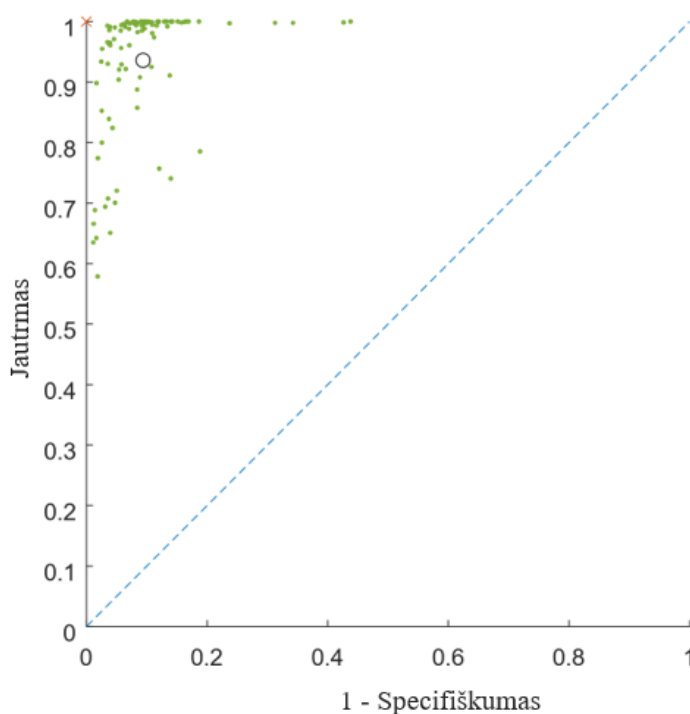
Iš 6 pav. matyti, kad FA nėra labai tikslus. Vis dėlto iš grafiko matoma ir algoritmo savybė, kurią jau atskleidė prieš tai minėti parametrai, t. y. algoritmas tinkamai kontūru apima didelį kiekį baltojo kraujo kūnelio taškų, bet netinkamai į šį kontūrą įtrauktą foną. Kompiuteriui pilnai segmentuoti duomenų rinkinį jonvabalių algoritmu prireikė 107,92 s.



7 pav. Skruzdžių kolonijų optimizavimo (ACO) algoritmo 100 rezultatų ROC erdvė

Iš 7 pav. matyti, kad skruzdžių kolonijų algoritmas yra tikslesnis. Ir nors šis algoritmas taip pat daugiau įtraukia taškų į rezultatą, šis algoritmas nėra kokybiškesnis už FA. Taip pat matyti, kad iš

visų trijų algoritmų šis yra arčiausiai atsitiktinio spėjimo. Algoritmui pabaigti duomenų rinkinį prireikė 6 965 s.



8 pav. Uodų taikinių paieškos algoritmo (MSHA) 100 rezultatų ROC erdvė

Iš 8 pav. matyti, kad MSHA yra geriausias iš lygintų metaeuristinių algoritmų. Vis dėlto iš ROC matyti, kad yra didelis rezultatų pasiskirstymas, kurio nematyti lyginant vidutines reikšmes. Taip pat, nors gautos ganėtinai žemos OR_p ir UR_p vertės, algoritmas dažniausiai gerai atskiria foną, bet neįtraukia pakankamai taškų į kontūrą, iš ROC erdvės grafiko matyti, kad algoritmo rezultatai pakankamai pasiskirstę. Tai reiškia, kad retais atvejais su kai kuriais vaizdais algoritmas atlieka per didelį segmentavimą. Nors šis algoritmas geriausias iš lygintų metaeuristinių algoritmų, jo greitaveikos ir tikslumo kompromisas nėra geras, nes viso rinkinio segmentavimas užtruko ~6,77 paras.

4.1.2. Greitaveikos ir lygiagretumo vertinimo rezultatai

Iš 6 lentelės matyti, kad FA algoritmas yra greičiausias iš metaeuristinių algoritmų. ACO palyginti su praėjusiame poskyriuje nagrinėtu GVF, kuriam vienam vaizdui segmentuoti prireikė 56,8 s, buvo spartesnis. Vis dėlto tam įtaka galėjo padaryti skirtinga kompiuterių sandara. Taip pat SVM vidutinė greitaveika (Zheng *et al.*, 2018) buvo 0,639 s. Nors kompiuterių specifikacija ir skiriasi, FA greitaveika buvo nuo 3 iki 12 kartų didesnė priklausomai nuo to, kuriame kompiuteryje algoritmas veikė.

Geriausią lygiagrečių skaičiavimų rezultatą parodė FA metaeuristinis algoritmas. Skirtumas tarp „AMD“ r5 1600 ir „Intel“ i5 2500 sistemų rezultatų buvo beveik dvigubai didesnė sparta. Nors

„AMD“ procesorius turi 6 branduolius, o „Intel“ 4 branduolius, toks spartos skirtumas atsiranda dėl to, kad „AMD“ procesorius naudoja daugelio gijų technologija, kas leidžia pagerinti lygiagrečių skaičiavimų sparta, o „Intel“ procesorius šios galimybės neturi.

6 lentelė. Metaeuristinių algoritmų greitaveikos vertinimo eksperimento rezultatai

Kompiuteris	FA	ACO	MHSA
„AMD“ r5 1600 su 16 GB	0,0582±0,074	21,9±0,216	1921±4,28
„Intel“ i5 2500 su 8 GB	0,118±0,2	14,72±1,11	2852±15,3
„Intel“ i5 5200u su 4 GB	0,2029±0,475	22,1±1,22	3128±27,9

Fonu pažymėtos mažiausios atitinkamų parametrų vertės.

Įdomus rezultatas gautas ACO algoritme, kur „Intel“ i5 2500 dėl kažkurių priežasčių buvo greitesnis nei likusieji kompiuteriai, kurie algoritmą baigia beveik vienodu laiku. Tam įtaka galėjo padaryti aukštesnis „Intel“ procesoriaus taktinis dažnis. Taip pat ACO sprendžiant iš vidutinio segmentavimo laiko neišnaudoja daugelio branduolių procesorių.

MHSA atveju „AMD“ procesorius buvo sparčiausias, bet daug mažesniu procentu negu FA. Taip pat neaišku ar šio algoritmo spartai didžiausią įtaka darė didelis greitos operatyvinės atminties dydis ar didesnis branduolių kiekis. Bet atsižvelgiant į FA algoritmą didesnis branduolių kiekis didelės įtakos nedaro.

Dar vienas įdomus rezultatas yra standartiniai nuokrypiai, kurie žymiai mažesni „AMD“ kompiuteriui. Taip galėjo atsitikti dėl konteksto perjungimo, bet tam parodyti reikia atlikti daugiau tyrimų.

4.2. Metodų rekomendacijos

Šiame poskyriuje analizavome tyrimo metodiką ir jos gerumą, įvertinome statistinių parametrų informatyvumą ir pateikėme tyrimo rekomendacijas.

Atlikto tyrimo metode pasirinkti vertinimo parametrai ir charakteristikos pasiteisino, buvo gerai atskleisti algoritmų privalumai ir trūkumai. Ypač gerai tai parodė sudaryti ROC erdvės grafikai. Gautos *RDE* ir taiklumo vertės metaeuristinius algoritmus juos išreitingavo skirtingose skalėse, nes parametrai, nors ir skaičiuojami skirtingai atskleidžia panašią informaciją.

Gauti *ER_p*, *RDE*, specifiškumo ir *NPV* vidutiniai parametrai turi didelį standartinį nuokrypį, todėl sunku įvertinti, kuris iš vertintų algoritmų geresnis. Tai bandyta ištaisyti atliekant kiekvieno vaizdo segmentavimą 3 kartus ir atrenkant geriausią rezultatą. Tačiau ši atranka nepakankamai (tik ~1,5%) sumažino standartinius nuokrypius. Galimi du šios problemos sprendimo būdai: a) galima būtų dar labiau padidinti atrankos imtį, tačiau tai labai prailgintų eksperimento trukmę; b) galima būtų išrūšiuoti vaizdus, siekiant atmesti prasčiausiai apdorojamus atvejus, tačiau taip gali būti pažeisti bendrieji eksperimento principai – iškiltų pavojus prisiderinti prie duomenų.

ER_p parametras buvo ypač neinformatyvus ir tikėtina nepatikimas. Literatūroje parametras paaiškinamas, kaip bendras klaidų skaičius, tačiau ACO atveju jo vertė buvo mažiausia, kai visų kitų parametrų vertės rodė, kad algoritmas turėtų būti mažiausio tikslumo.

Greitaveikos vertinimo tyrimo metodika pasiteisino. Buvo tinkamai išskirti algoritmai pagal jų spartą. Bet lygiagrečių skaičiavimų galimybės įvertinti truko informacijos. Kai kuriais atvejais neaišku ar spartesnė atmintis ar gijų skaičius darė didesnę įtaką rezultatams. Eksperimentą reikėjo atlikti vienu kompiuteriu programiškai išjungiant tam tikrą gijų skaičių.

Eksperimentiniai tyrimai truko ilgai, ypač su MHSA, todėl rekomenduojama eksperimentą atlikti dalimis.

4.3. Skyriaus apibendrinimas

Šiame skyriuje pateikti eksperimentinio tyrimo rezultatai, kuriais remiantis galima teigti:

- SVM yra tiksliausias iš palygintų algoritmų. Visgi šiam algoritmui reikalingas apmokymas.
- FA vaizdus segmentuoja sparčiausiai.
- Nors iš tiesų metaeuristiniai algoritmai nėra geresni negu GVF algoritmas, jie turi savų pranašumų. Pavyzdžiui, FA apima kontūru didelį kiekį baltojo kraujo kūnelio taškų, bet netinkamai į šį kontūrą įtraukia foną, o MHSA dažniausiai gerai atskiria foną, bet neįtraukia pakankamai baltojo kraujo kūnelio taškų į kontūrą.
- ACO yra mažiausiai tikslus iš tiriamų metaeuristinių algoritmų.

Analizuojant taikytą tyrimo metodiką, suformuluotos šios rekomendacijos:

- Gauti ER_p , RDE , specifiškumo ir NPV vidutiniai parametrų standartiniai nuokrypiai galėtų būti sumažinti dviem būdais: a) galima būtų dar labiau padidinti atrankos imtį, tačiau tai labai prailgintų eksperimento trukmę; b) galima būtų išrūšiuoti vaizdus, siekiant atmesti prasčiausiai apdorojamus atvejus, tačiau taip gali būti pažeisti bendrieji eksperimento principai – iškiltų pavojus prisiderinti prie duomenų.
- ER_p parametro rezultatai neinformatyvūs, todėl jo naudojimas nerekomenduotinas.
- Ypač sunku nustatyti lygiagretumo ir operatyvinės atminties daroma įtaką rezultatams, todėl eksperimentą reikėtų kartoti vienu kompiuteriu programiškai atjungiant tam tikrą gijų skaičių.

IŠVADOS

Magistro baigiamojo darbo tikslas pasiektas – atrinkti trys gamta grįsti metaeuristiniai algoritmai: skruzdžių kolonijų optimizavimo (ACO), jonvabalių (FA) ir uodų taikinių paieškos (MHSA), ir ištirtos jų taikymo baltųjų kraujo kūnelių vaizdams segmentuoti galimybės.

ACO algoritmą pritaikius etalono vaizdų dydžiui ir spalvų gamai, kryžminant FA (naudojant Otsu algoritmo dalis) ir naujai pritaikius MHSA segmentavimo problemai spręsti, pagal sudarytą tyrimo metodiką atlikti eksperimentiniai tyrimai, kurių rezultatai leidžia teigti, kad:

1. Geriausiai baltųjų kraujo kūnelių vaizdus segmentuoja SVM algoritmas ($OR_p = 0,0187 \pm 0,01$, $UR_p = 0,0131 \pm 0,01$). Visgi šiam algoritmui reikalingas apmokymas.
2. Sparčiausiai baltųjų kraujo kūnelių vaizdus segmentuoja FA (0,0582 s), antras pagal spartą – SVM (0,639 s).
3. Nors tirti metaeuristiniai algoritmai nėra geresni negu GVF algoritmas, jie turi savų pranašumų. Pavyzdžiui, FA apima kontūrą didelį kiekį baltojo kraujo kūnelio taškų, bet netinkamai į šį kontūrą įtraukia fono taškų, o MHSA dažniausiai gerai atskiria foną, bet neįtraukia didelės dalies baltojo kraujo kūnelio taškų į kontūrą.
4. ACO mažiausiai tikslus iš tiriamų algoritmų remiantis OR_p ($0,522 \pm 0,0894$) ir UR_p ($0,416 \pm 0,0715$) parametrais.

Remiantis atliktais metaeuristinių algoritmų tyrimų rezultatais, rekomenduojama:

1. Siekiant vaizdus segmentuoti jonvabalių algoritmu jį būtina kryžminti. Kryžminimas buvo atliktas pridėdant Otsu metodo klasių dispersijos funkciją, kurią FA naudoja kaip tikslo funkciją.
2. Siekiant segmentuoti baltųjų kraujo kūnelių vaizdus uodų taikinių paieškos algoritmu, jo įvestį reikia apdoroti naudojant vaizdų gradientus.
3. Siekiant didžiausio baltųjų kraujo kūnelių vaizdų segmentavimo tikslumo, nepatartina taikyti ACO, FA ir MHSA, nes tikslesni rezultatai gaunami apmokomais algoritmais (SVM algoritmu).
4. Siekiant baltųjų kraujo kūnelių vaizdų segmentavimo spartos, rekomenduojama taikyti metaeuristinius algoritmus, konkrečiai – jonvabalių algoritmą.
5. Metaeuristinius algoritmus vertinti rekomenduojama pagal OR_p , UR_p , tikslumą, jautrumą ir ROC erdvę.

LITERATŪRA

- Abedinia, O.; Amjady, N.; Naderi, M. (2012). Multi-objective environmental/economic dispatch using firefly technique, in: *Environment and Electrical Engineering, 2012 11th International Conference on Environment and Electrical Engineering (EEEIC)*: 461–466.
- Afshar, M.; Ketabchi, H.; Rasa, E. (2006). *Elitist Continuous Ant Colony Optimization Algorithm: Application to Reservoir Operation Problems*. 274–285 p.
- Alauddin, M. (2016). Mosquito flying optimization (MFO) *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)* 3–5 March 2016, 79–84.
- Ashtiani, B.; Ketabchi, H. (2011). *Elitist Continuous Ant Colony Optimization Algorithm for Optimal Management of Coastal Aquifers*. 165–190 p.
- Babu, B.; Gujarathi, A. (2007). Multi-objective differential evolution (mode) for optimization of supply chain planning and management, *Evolutionary Computation*: 2732–2739.
- Bautista, J.; Pereira, J. (2002). Ant algorithms for assembly line balancing, *Springer Verlag* 2463: 65–75.
- Behrouz, A. (2007). *Cryptography and Network Security, Tata McGrawHill*.
- Bianchi, L.; Gambardella, L. M.; Dorigo, M. (2002). An ant colony optimization approach to the probabilistic traveling salesman problem, *Springer Verlag* 2439(883–892).
- Blackwell, T.; Branke, J.; Li, X. (2008). Particle swarms for dynamic optimization problems: 193–217.
- Bullnheimer, B.; Hartl, R.; Strauss, C. (1999). *A New Rank Based Version of the Ant System - A Computational Study*. 25–38 p.
- Caro, G. D.; Ducatelle, F.; Gambardella, L. M. (2005). AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks, *European Transactions on Telecommunications* 16(5): 443–455.
- Carrion, M.; Arroyo, J. M. (2006). A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem, *IEEE Transactions on Power Systems* 21(3): 1371–1378. ISSN 0885-8950. doi: 10.1109/TPWRS.2006.876672.
- Chandrasekaran, K.; Simon, S. P. (2012). Network and reliability constrained unit commitment problem using binary real coded firefly algorithm, *International Journal of Electrical Power & Energy Systems* 43(1): 921–932. ISSN 0142-0615. doi: <https://doi.org/10.1016/j.ijepes.2012.06.004>.
- Deb, K. *et al.* (2002). A fast and elitist multiobjective algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*: 182–197.
- Dieterich, J.; Hartke, B. (2012). *Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization*.
- Doerner, K. *et al.* (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection, *Annals of Operations Research* 131(1–4): 79–99. 1–4(131): 79–99.
- Doerner, K. *et al.* (2006). Pareto ant colony optimization in multiobjective project portfolio selection with ILP preprocessing, *European journal of operational research* 171(3): 830–841.
- Doerner, K.; Hartl, R. F.; Reimann, M. (2003). CompetAnts for problem solving—the case of full truckload transportation, *Central European Journal for Operations Research and Economics* 11(2): 115–141.

- Dorigo, M.; Birattari, M.; Stutzle, T. (2006). Ant colony optimization, *IEEE Computational Intelligence Magazine* 1(4): 28–39. ISSN 1556-603X. doi: 10.1109/MCI.2006.329691.
- Dorigo, M.; Blum, C. (2005). Ant colony optimization theory: A survey, *Theoretical Computer Science* 344(2): 243–278. ISSN 0304-3975. doi: <https://doi.org/10.1016/j.tcs.2005.05.020>.
- Dorigo, M.; Caro, G. D. (1998). AntNet: Distributed stigmergetic control for communications networks, *Journal of Artificial Intelligence Research* 9: 317–365.
- Dorigo, M.; Caro, G. D. (1999). Ant colony optimization: a new meta-heuristic *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* 1999, 1–1477 Vol. 2.
- Dorigo, M.; Maniezzo, V.; Coloni, A. (1996). Ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26(1): 29–41. ISSN 1083-4419. doi: 10.1109/3477.484436.
- Ducatelle, F.; Caro, G. D.; Gambardella, L. M. (2005). Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks, *International Journal of Computational Intelligence and Applications* 5: 169–184.
- Falcon, R.; Almeida, M.; Nayak, A. (2007). Multi-objective optimisation using the bees algorithm, *3rd international virtual conference on intelligent production machines and systems*: 1–5.
- Falcon, R.; Almeida, M.; Nayak, A. (2010). A binary Particle Swarm Optimization approach to fault diagnosis in parallel and distributed systems *IEEE Congress on Evolutionary Computation* 18–23 July 2010, 1–8. ISBN 1089-778X.
- Falcon, R.; Almeida, M.; Nayak, A. (2011). Fault identification with binary adaptive fireflies in parallel and distributed systems *2011 IEEE Congress of Evolutionary Computation (CEC)* 5–8 June 2011, 1359–1366. ISBN 1941-0026.
- Farahani, M. et al. (2011). *A Gaussian Firefly Algorithm*. 448–453 p.
- Feng, X.; Lau, F. C. M.; Yu, H. (2013). A novel bio-inspired approach based on the behavior of mosquitoes, *Information Sciences* 233(Supplement C): 87–108. ISSN 0020-0255. doi: <https://doi.org/10.1016/j.ins.2012.12.053>.
- Fister, I. et al. (2013). A comprehensive review of firefly algorithms, *Swarm and Evolutionary Computation* 13(Supplement C): 34–46. ISSN 2210-6502. doi: <https://doi.org/10.1016/j.swevo.2013.06.001>.
- Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation, *Science* 261(5123): 872–878. doi: 10.1126/science.8346439.
- Gambardella, L. M.; Taillard, E. D.; Agazzi, G. (1999). MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows, *New Ideas in Optimization*: 63–76.
- Gandomi, A. H.; Yang, X. S.; Alavi, A. H. (2011). Mixed variable structural optimization using Firefly Algorithm, *Computers & Structures* 89(23): 2325–2336. ISSN 0045-7949. doi: <https://doi.org/10.1016/j.compstruc.2011.08.002>.
- Gravel, M.; Price, W. L.; Gagn, C. (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristics, *European journal of operational research* 44(5): 443–455.
- Ibanz, M. L.; Paquete, L.; Stutzle, T. (2004). On the design of ACO for biobjective quadratic assignment problem, *Springer Verlag* 3172: 214–225.

- Iredi, S.; Merkle, D.; Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms in First International Conference on Evolutionary Multi-Criterion Optimization, *Springer Verlag* 1993: 359–372.
- Jamil, M.; Yang, X. S. (2013). *A Literature Survey of Benchmark Functions For Global Optimization Problems*.
- Jing, T.; Weiyu, Y.; Shengli, X. (2008). An ant colony optimization algorithm for image edge detection2008 *IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)* 1-6 June 2008, 751–756. ISBN 1089-778X.
- Ko, B. C.; Gim, J.-W.; Nam, J.-Y. (2011). Automatic white blood cell segmentation using stepwise merging rules and gradient vector flow snake, *Micron* 42(7): 695–705. ISSN 0968-4328. doi: <https://doi.org/10.1016/j.micron.2011.03.009>.
- Liao, T. et al. (2014). Ant Colony Optimization for Mixed-Variable Optimization Problems, *IEEE Transactions on Evolutionary Computation* 18(4): 503-518. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281531.
- Lin, Y. H. et al. (2006). *Open Source Licenses and the Creative Commons Framework: License Selection and Comparison*. 1–17 p.
- Niknam, T. et al. (2012). A new multiobjective reserve constrained combined heat and power dynamic economic emission dispatch, *Energy* 42: 530–545.
- Palit, S. et al. (2011). A cryptanalytic attack on the knapsack cryptosystem using binary Firefly algorithm2011 *2nd International Conference on Computer and Communication Technology (ICCCCT-2011)* 15–17 Sept. 2011, 428–432.
- Sarbafard, S.; Jafarian, A. (2017). A Hybrid Algorithm Based on Firefly Algorithm and Differential Evolution for Global Optimization, *Journal of Advances in Computer Research* 8(2): 21–38. ISSN 2345-606X.
- Schaffer, J. (1985). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. 93–100 p.
- Schoonderwoerd, R.; Holland, O.; Rothkrantz, J. B. L. (1996). Ant-based load balancing in telecommunications networks, *Adaptive Behavior* 5(2): 169–207.
- Senthilnath, J.; Omkar, S. N.; Mani, V. (2011). Clustering using firefly algorithm: Performance study, *Swarm and Evolutionary Computation* 1(3): 164–171. ISSN 2210-6502. doi: <https://doi.org/10.1016/j.swevo.2011.06.003>.
- Sharma, A.; Sehgal, S. (2016). Image segmentation using firefly algorithm2016 *International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds* 6–7 Oct. 2016, 99–102.
- Sim, K. M.; Sun, W. H. (2004). Ant colony optimization for routing and load-balancing: Survey and new directions, *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*.
- Socha, K.; Dorigo, M. (2008). Ant colony optimization for continuous domains, *European journal of operational research* 185(3): 1155–1173. ISSN 0377-2217. doi: <https://doi.org/10.1016/j.ejor.2006.06.046>.
- Stützle, T.; Hoos, H. H. (2000). MAX–MIN Ant System, *Future Generation Computer Systems* 16(8): 889–914. ISSN 0167-739X. doi: [https://doi.org/10.1016/S0167-739X\(00\)00043-1](https://doi.org/10.1016/S0167-739X(00)00043-1).
- Tilahun, S.; Ong, H. C. (2012). *Modified Firefly Algorithm*.
- Tusar, T.; Filipic, B. (2005). *DEMO: Differential Evolution for Multiobjective Optimization*. 520-533 p.

- Wang, G.; Lin, Z. (2017). Hybrid Mosquito Host-Seeking Algorithm for Multi-Depot Vehicle Routing Problem 2017 9th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA) 14–15 Jan. 2017, 182–186.
- Yang (2010). Firefly Algorithm. Engineering Optimization, John Wiley & Sons, Inc.: 221–230.
- Yang, H. *et al.* (2008). Network Fault Diagnosis: An Artificial Immune System Approach 2008 14th IEEE International Conference on Parallel and Distributed Systems 8–10 Dec. 2008, 463–469. ISBN 1521-9097.
- Yang, X. S. (2013). *Multiobjective Firefly Algorithm for Continuous Optimization*. 1–10 p.
- Yang, X. S.; Deb, S. (2010). *Eagle Strategy Using Lévy Walk and Firefly Algorithms for Stochastic Optimization*. 101–111 p.
- Yang, X. S. (2008). Firefly algorithm, *Nature-Inspired Metaheuristic Algorithms*: 79-90.
- Yang, X. S. (2012). Efficiency analysis of swarm intelligence and randomization techniques, *Journal of Computational and Theoretical Nanoscience* 9 (2) 189-198.
- Zadeh, S. H.; Rezaatofghi. H. S. (2011). Automatic recognition of five types of white blood cells in peripheral blood., *Computerized Medical Imaging and Graphics* 35(4).
- Zheng, X. *et al.* (2018). Fast and Robust Segmentation of White Blood Cell Images by Self-supervised Learning, *Micron* 107: 55–71. doi: <https://doi.org/10.1016/j.micron.2018.01.010>.
- Zitzler, E.; Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation* 3(4): 257–271. ISSN 1089-778X. doi: 10.1109/4235.797969.

PRIEDAI

A priedas. Tyrimo rezultatų duomenys

7 lentelė. FA parametrų rezultatai 100 vaizdų

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	$SENS$	$SPEC$	NPV
1	0.174	0	0.033	6.92	0.972	0.826	1	0.968
2	0.0525	0.0181	0.0502	15.6	0.971	0.947	0.987	0.963
3	0.0695	0.0269	0.0679	14	0.96	0.93	0.982	0.952
4	0.433	0	0.224	8.85	0.853	0.567	1	0.817
5	0.158	0.278	0.321	21.1	0.808	0.842	0.784	0.882
6	0.0741	0.00186	0.043	10.9	0.973	0.926	0.999	0.96
7	0.382	0	0.127	8.07	0.905	0.618	1	0.887
8	0.119	0.0321	0.0839	11.9	0.946	0.881	0.983	0.936
9	0.0213	0.0128	0.0282	14.8	0.985	0.979	0.989	0.982
10	0.0846	0.00076	0.0849	12.6	0.957	0.915	0.999	0.922
11	0.0561	0.0227	0.0435	13.4	0.972	0.944	0.988	0.969
12	0.499	0	0.369	12.8	0.788	0.501	1	0.73
13	0.143	0.111	0.177	14.3	0.896	0.857	0.924	0.901
14	0.503	0.0181	0.274	9.59	0.822	0.497	0.995	0.787
15	0.069	0.0155	0.0554	12.6	0.967	0.931	0.99	0.956
16	0.106	0.00961	0.0287	9.13	0.977	0.894	0.998	0.974
17	0.286	0.00183	0.211	11.6	0.879	0.714	0.999	0.827
18	0.495	0.00247	0.315	11.9	0.807	0.505	0.999	0.761
19	0.155	0	0.0414	8.73	0.967	0.845	1	0.96
20	0.207	0	0.162	12.5	0.909	0.793	1	0.861
21	0.0379	0.0108	0.0317	13.1	0.981	0.962	0.993	0.976
22	0.124	0.0119	0.09	12	0.946	0.876	0.993	0.923
23	0.0986	0.00359	0.053	11.8	0.965	0.901	0.998	0.951
24	0.161	0.0065	0.126	12.8	0.928	0.839	0.996	0.891
25	0.231	0.00373	0.0475	9.77	0.96	0.769	0.999	0.955
26	0.299	0.0539	0.211	8.99	0.87	0.701	0.975	0.84
27	0.587	0.00326	0.411	9.51	0.758	0.413	0.999	0.709
28	0.584	0	0.435	12.2	0.751	0.416	1	0.697
29	0.0851	0	0.0236	9.49	0.982	0.915	1	0.977
30	0.172	0.0323	0.116	11.9	0.927	0.828	0.984	0.908
31	0.153	0	0.0314	8.3	0.974	0.847	1	0.97
32	0.0899	0.0184	0.0274	8.85	0.978	0.91	0.996	0.977
33	0.577	0.00823	0.32	13.5	0.794	0.423	0.998	0.758
34	0.479	0.00686	0.28	9.93	0.823	0.521	0.998	0.782
35	0.51	0.106	0.371	10.7	0.775	0.49	0.962	0.743
36	0.263	0.00746	0.0672	9.9	0.946	0.737	0.999	0.938
37	0.501	0.0477	0.332	15.2	0.797	0.499	0.984	0.757
38	0.169	0.0175	0.0398	8.13	0.967	0.831	0.997	0.965

Iteracija	OR _p	UR _p	ER _p	RDE	ACC	SENS	SPEC	NPV
39	0.676	0	0.42	10.8	0.741	0.324	1	0.704
40	0.448	0.00667	0.249	9.09	0.839	0.552	0.998	0.801
41	0.579	0.0117	0.352	8.67	0.78	0.421	0.997	0.741
42	0.642	0	0.462	9.69	0.731	0.358	1	0.684
43	0.531	0.154	0.343	15.9	0.78	0.469	0.952	0.763
44	0.589	0.251	0.476	19.4	0.712	0.411	0.91	0.702
45	0.279	0.0355	0.186	13	0.884	0.721	0.984	0.853
46	0.592	0	0.372	8.81	0.771	0.408	1	0.729
47	0.139	0.207	0.0914	11.7	0.927	0.861	0.943	0.964
48	0.559	0.0171	0.322	10.7	0.795	0.441	0.996	0.758
49	0.533	0.00983	0.265	7.59	0.822	0.467	0.998	0.792
50	0.622	0	0.419	9.74	0.75	0.378	1	0.705
51	0.117	0.0198	0.0256	8.99	0.978	0.883	0.997	0.978
52	0.568	0	0.399	10.5	0.765	0.432	1	0.715
53	0.0789	0.385	0.219	18.1	0.836	0.921	0.808	0.968
54	0.0881	0.00867	0.0422	11.4	0.971	0.912	0.997	0.963
55	0.481	0	0.221	9.2	0.849	0.519	1	0.819
56	0.168	0	0.0729	10.3	0.949	0.832	1	0.932
57	0.304	0	0.182	9.59	0.886	0.696	1	0.846
58	0.461	0.0227	0.241	10.2	0.84	0.539	0.994	0.809
59	0.618	0	0.335	12	0.783	0.382	1	0.749
60	0.311	0	0.208	13.3	0.875	0.689	1	0.828
61	0.536	0.0138	0.256	12.4	0.826	0.464	0.997	0.798
62	0.181	0	0.0347	7.36	0.971	0.819	1	0.967
63	0.0779	0.161	0.139	18.9	0.91	0.922	0.903	0.955
64	0.418	0	0.206	8.73	0.862	0.582	1	0.829
65	0.581	0	0.276	13.3	0.813	0.419	1	0.784
66	0.0317	0.00722	0.0213	12.5	0.986	0.968	0.996	0.983
67	0.128	0.0262	0.0313	9.9	0.974	0.872	0.995	0.974
68	0.625	0	0.465	11	0.733	0.375	1	0.682
69	0.114	0	0.0505	10.7	0.965	0.886	1	0.952
70	0.128	0	0.0293	8.62	0.976	0.872	1	0.972
71	0.1	0.024	0.06	11.8	0.96	0.9	0.989	0.953
72	0.0706	0	0.0349	12.6	0.977	0.929	1	0.966
73	0.184	0	0.0973	11.8	0.936	0.816	1	0.911
74	0.159	0	0.0801	10.9	0.947	0.841	1	0.926
75	0.053	0.000208	0.0289	12.9	0.981	0.947	1	0.972
76	0.56	0	0.239	10.4	0.833	0.44	1	0.807
77	0.698	0	0.381	11.7	0.754	0.302	1	0.724
78	0.741	0	0.355	14.4	0.76	0.259	1	0.738
79	0.679	0	0.244	8.37	0.82	0.321	1	0.804
80	0.615	0	0.194	10.6	0.852	0.385	1	0.837
81	0.621	0.0655	0.232	12.4	0.829	0.379	0.99	0.817
82	0.0253	0.0164	0.0159	11.9	0.988	0.975	0.994	0.99
83	0.105	0.00836	0.0633	11.3	0.959	0.895	0.996	0.944

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	$SENS$	$SPEC$	NPV
84	0.0452	0.0933	0.0506	13	0.963	0.955	0.965	0.984
85	0.0667	0	0.024	11	0.982	0.933	1	0.977
86	0.119	0.0222	0.0812	11.9	0.949	0.881	0.988	0.934
87	0.361	0	0.0858	8.79	0.931	0.639	1	0.921
88	0.219	0	0.0462	7.87	0.962	0.781	1	0.956
89	0.29	0	0.0912	11.3	0.931	0.71	1	0.916
90	0.176	0	0.0661	10.2	0.952	0.824	1	0.938
91	0.147	0	0.0605	9.79	0.957	0.853	1	0.943
92	0.125	0	0.0494	11.3	0.965	0.875	1	0.953
93	0.175	0	0.14	16.2	0.922	0.825	1	0.877
94	0.161	0	0.0779	10.8	0.948	0.839	1	0.928
95	0.0964	0	0.0579	15.2	0.964	0.904	1	0.945
96	0.667	0	0.471	12.6	0.724	0.333	1	0.68
97	0.0984	0	0.0492	10.2	0.967	0.902	1	0.953
98	0.458	0	0.27	12.9	0.83	0.542	1	0.787
99	0.227	0	0.076	9.79	0.943	0.773	1	0.929
100	0.156	0	0.0588	9.37	0.957	0.844	1	0.944

8 lentelė. ACO parametrų rezultatai 100 vaizdų

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	$SENS$	$SPEC$	NPV
1	0.313	0.243	0.0294	14.2	0.972	0.687	0.988	0.983
2	0.396	0.329	0.0735	17.8	0.934	0.604	0.969	0.958
3	0.406	0.394	0.0791	14.7	0.928	0.594	0.961	0.96
4	0.43	0.37	0.0809	14	0.927	0.57	0.965	0.955
5	0.509	0.532	0.0946	20.7	0.913	0.491	0.951	0.955
6	0.444	0.399	0.0859	13.1	0.922	0.556	0.961	0.953
7	0.423	0.407	0.059	15	0.945	0.577	0.971	0.97
8	0.533	0.419	0.0901	14.6	0.918	0.467	0.965	0.946
9	0.402	0.411	0.0927	18.3	0.917	0.598	0.953	0.954
10	0.498	0.438	0.109	16.1	0.903	0.502	0.952	0.94
11	0.382	0.398	0.0712	15.9	0.935	0.618	0.963	0.965
12	0.486	0.364	0.0885	15.5	0.921	0.514	0.967	0.946
13	0.533	0.508	0.0993	21.4	0.91	0.467	0.953	0.948
14	0.519	0.458	0.0954	15.3	0.914	0.481	0.958	0.947
15	0.466	0.426	0.0964	15.6	0.913	0.534	0.956	0.948
16	0.433	0.347	0.0478	14.6	0.955	0.567	0.98	0.972
17	0.477	0.491	0.0887	15	0.919	0.523	0.954	0.957
18	0.486	0.447	0.101	14.2	0.909	0.514	0.953	0.946
19	0.519	0.4	0.0559	14	0.948	0.481	0.979	0.966
20	0.517	0.34	0.0895	12.8	0.92	0.483	0.971	0.941
21	0.464	0.415	0.0791	18.5	0.928	0.536	0.964	0.957
22	0.417	0.335	0.0751	14.3	0.932	0.583	0.969	0.956
23	0.4	0.363	0.079	16.2	0.929	0.6	0.964	0.958
24	0.507	0.362	0.106	15.4	0.907	0.493	0.962	0.934
25	0.535	0.32	0.0443	12.2	0.958	0.465	0.987	0.969

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	SENS	SPEC	NPV
26	0.58	0.484	0.0982	13.1	0.911	0.42	0.96	0.943
27	0.503	0.405	0.0883	13.8	0.92	0.497	0.964	0.948
28	0.535	0.415	0.0853	15.2	0.922	0.465	0.967	0.948
29	0.572	0.387	0.058	15.3	0.946	0.428	0.981	0.961
30	0.664	0.511	0.0924	21.2	0.915	0.336	0.968	0.941
31	0.414	0.311	0.0398	13.2	0.962	0.586	0.984	0.976
32	0.398	0.518	0.068	14.8	0.936	0.602	0.958	0.974
33	0.607	0.486	0.0915	16.3	0.916	0.393	0.965	0.944
34	0.55	0.479	0.106	13.4	0.904	0.45	0.954	0.94
35	0.64	0.513	0.131	14.2	0.884	0.36	0.951	0.921
36	0.59	0.515	0.068	14.4	0.936	0.41	0.971	0.961
37	0.451	0.494	0.0969	20.6	0.912	0.549	0.947	0.955
38	0.436	0.247	0.0398	12.8	0.963	0.564	0.988	0.973
39	0.598	0.382	0.0825	13.2	0.925	0.402	0.976	0.944
40	0.572	0.432	0.101	13.9	0.909	0.428	0.963	0.937
41	0.609	0.444	0.0957	13.6	0.913	0.391	0.967	0.939
42	0.541	0.384	0.0915	12.6	0.918	0.459	0.968	0.942
43	0.534	0.486	0.102	19.3	0.908	0.466	0.954	0.945
44	0.616	0.557	0.117	22.1	0.894	0.384	0.948	0.935
45	0.464	0.495	0.0858	21.3	0.921	0.536	0.954	0.96
46	0.523	0.435	0.0943	11.3	0.915	0.477	0.961	0.945
47	0.516	0.563	0.0736	16.9	0.931	0.484	0.96	0.966
48	0.58	0.474	0.105	14	0.905	0.42	0.959	0.938
49	0.548	0.484	0.0914	14.4	0.916	0.452	0.96	0.949
50	0.526	0.399	0.0867	12.6	0.921	0.474	0.968	0.947
51	0.403	0.341	0.0396	15.5	0.963	0.597	0.983	0.978
52	0.541	0.404	0.0954	14.1	0.914	0.459	0.965	0.941
53	0.463	0.44	0.0719	13.2	0.934	0.537	0.966	0.963
54	0.493	0.49	0.0798	22.5	0.926	0.507	0.96	0.96
55	0.523	0.384	0.0732	13.3	0.933	0.477	0.973	0.954
56	0.443	0.415	0.0676	16.2	0.937	0.557	0.968	0.964
57	0.425	0.405	0.0819	12.5	0.926	0.575	0.961	0.957
58	0.424	0.41	0.0702	16.6	0.935	0.576	0.966	0.964
59	0.596	0.422	0.0816	14.9	0.925	0.404	0.973	0.947
60	0.579	0.48	0.088	17.8	0.919	0.421	0.965	0.948
61	0.588	0.538	0.0934	16.8	0.914	0.412	0.958	0.949
62	0.501	0.365	0.047	12.9	0.956	0.499	0.983	0.97
63	0.367	0.37	0.063	16.1	0.942	0.633	0.968	0.969
64	0.437	0.364	0.063	15.4	0.942	0.563	0.973	0.964
65	0.627	0.467	0.0882	13.2	0.919	0.373	0.97	0.944
66	0.626	0.401	0.0895	22.9	0.919	0.374	0.974	0.938
67	0.313	0.256	0.033	14.4	0.969	0.688	0.986	0.981
68	0.591	0.383	0.0879	13.2	0.92	0.409	0.974	0.941
69	0.447	0.461	0.0734	17.4	0.932	0.553	0.962	0.964
70	0.433	0.422	0.0511	14.1	0.952	0.567	0.975	0.974

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	$SENS$	$SPEC$	NPV
71	0.532	0.427	0.0805	18.8	0.926	0.468	0.968	0.952
72	0.461	0.248	0.0535	16.5	0.951	0.539	0.985	0.962
73	0.497	0.352	0.0642	14.1	0.941	0.503	0.977	0.959
74	0.475	0.357	0.0644	14	0.941	0.525	0.976	0.961
75	0.646	0.303	0.0769	16.2	0.93	0.354	0.985	0.941
76	0.555	0.333	0.0714	12.9	0.935	0.445	0.98	0.95
77	0.565	0.333	0.0765	13.9	0.93	0.435	0.979	0.947
78	0.641	0.437	0.0716	13.3	0.934	0.359	0.978	0.951
79	0.476	0.307	0.0551	12.8	0.949	0.524	0.982	0.964
80	0.663	0.397	0.0689	14.1	0.936	0.337	0.983	0.95
81	0.604	0.422	0.0732	14.3	0.932	0.396	0.976	0.952
82	0.421	0.337	0.0557	15.6	0.948	0.579	0.977	0.968
83	0.442	0.444	0.0917	16.3	0.917	0.558	0.954	0.954
84	0.428	0.499	0.0709	17	0.934	0.572	0.96	0.969
85	0.478	0.309	0.0628	13.6	0.942	0.522	0.979	0.959
86	0.498	0.375	0.0672	15	0.938	0.502	0.975	0.959
87	0.626	0.438	0.0583	14.8	0.945	0.374	0.981	0.961
88	0.588	0.357	0.0494	13.8	0.953	0.412	0.986	0.965
89	0.625	0.412	0.0633	17.5	0.941	0.375	0.981	0.957
90	0.625	0.493	0.0745	16.2	0.931	0.375	0.973	0.954
91	0.595	0.442	0.07	18.3	0.935	0.405	0.975	0.955
92	0.605	0.374	0.0666	18.5	0.938	0.395	0.981	0.953
93	0.747	0.508	0.0989	22	0.91	0.253	0.974	0.93
94	0.727	0.522	0.0843	19	0.922	0.273	0.975	0.942
95	0.613	0.388	0.0772	19.4	0.929	0.387	0.978	0.947
96	0.556	0.391	0.0723	21.8	0.933	0.444	0.975	0.953
97	0.709	0.497	0.0878	17.7	0.919	0.291	0.975	0.94
98	0.633	0.548	0.0944	15.2	0.913	0.367	0.961	0.945
99	0.648	0.487	0.0726	12.2	0.932	0.352	0.975	0.953
100	0.603	0.417	0.0735	12.9	0.932	0.397	0.976	0.951

9 lentelė. MSHA parametrų rezultatai 100 vaizdų

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	$SENS$	$SPEC$	NPV
1	0	0.387	0.12	14.9	0.899	1	0.88	1
2	0	0.165	0.142	17.6	0.917	1	0.858	1
3	0.000167	0.164	0.14	15.4	0.918	1	0.86	1
4	0.00653	0.151	0.0949	13.2	0.937	0.993	0.908	0.996
5	0.00208	0.32	0.314	21.9	0.812	0.998	0.687	0.998
6	0	0.201	0.143	13.6	0.909	1	0.857	1
7	0.00808	0.289	0.137	15	0.897	0.992	0.866	0.997
8	0.000385	0.224	0.163	15.8	0.896	1	0.838	1
9	0	0.135	0.129	15.7	0.93	1	0.871	1
10	0	0.132	0.151	15	0.924	1	0.849	1
11	0	0.233	0.169	17.3	0.891	1	0.831	1
12	0.00376	0.0805	0.0673	13.4	0.961	0.996	0.935	0.997

Iteracija	OR _p	UR _p	ER _p	RDE	ACC	SENS	SPEC	NPV
13	0.00267	0.251	0.239	16.9	0.86	0.997	0.763	0.998
14	0.00678	0.18	0.12	16.1	0.922	0.993	0.884	0.996
15	0.00105	0.132	0.101	14.2	0.939	0.999	0.9	0.999
16	0	0.266	0.0906	13.1	0.928	1	0.909	1
17	0	0.203	0.186	16.9	0.893	1	0.814	1
18	0	0.207	0.165	14.3	0.899	1	0.835	1
19	0	0.227	0.0782	11.4	0.938	1	0.922	1
20	0	0.104	0.0902	15.1	0.949	1	0.91	1
21	0	0.176	0.141	13.8	0.915	1	0.859	1
22	0.00156	0.128	0.0987	13.4	0.941	0.998	0.902	0.999
23	0	0.156	0.0965	13.3	0.937	1	0.903	1
24	0.000161	0.118	0.101	15	0.943	1	0.899	1
25	0.0148	0.315	0.0952	11.3	0.921	0.985	0.908	0.997
26	0.112	0.132	0.154	10.2	0.905	0.888	0.916	0.929
27	0.0709	0.0821	0.108	14.3	0.937	0.929	0.942	0.95
28	0.0963	0.0737	0.125	14.1	0.928	0.904	0.946	0.93
29	0.00064	0.275	0.105	11.8	0.918	0.999	0.895	1
30	0.0749	0.167	0.151	13.9	0.904	0.925	0.892	0.954
31	0	0.305	0.0901	10.8	0.925	1	0.91	1
32	0.00272	0.219	0.0725	10.7	0.942	0.997	0.928	0.999
33	0.3	0.109	0.213	14.6	0.863	0.7	0.953	0.852
34	0.147	0.0486	0.111	11.5	0.93	0.853	0.975	0.92
35	0.243	0.196	0.279	12.8	0.831	0.757	0.879	0.847
36	0.0783	0.22	0.0847	12.1	0.932	0.922	0.935	0.979
37	0.143	0.135	0.174	15.9	0.893	0.857	0.916	0.911
38	0.0039	0.307	0.0961	11.3	0.921	0.996	0.905	0.999
39	0.421	0.0491	0.28	8.9	0.827	0.579	0.981	0.79
40	0.2	0.0549	0.136	10.8	0.912	0.8	0.974	0.898
41	0.358	0.0405	0.232	9.23	0.855	0.642	0.984	0.82
42	0.365	0.0242	0.274	9.28	0.841	0.635	0.989	0.79
43	0.26	0.253	0.285	17.4	0.817	0.74	0.86	0.856
44	0.215	0.268	0.329	0	0.801	0.785	0.812	0.852
45	0.0922	0.138	0.145	15.7	0.91	0.908	0.911	0.942
46	0.349	0.088	0.259	9.17	0.841	0.651	0.961	0.814
47	0.019	0.307	0.114	13	0.909	0.981	0.891	0.995
48	0.161	0.0728	0.129	11.9	0.918	0.839	0.963	0.913
49	0.28	0.124	0.188	10.5	0.874	0.72	0.95	0.873
50	0.312	0.0294	0.224	10	0.866	0.688	0.986	0.824
51	0	0.333	0.0951	13	0.92	1	0.905	1
52	0.306	0.0602	0.246	10.9	0.855	0.694	0.969	0.818
53	0.00139	0.562	0.427	22.2	0.68	0.999	0.574	0.999
54	0.00319	0.161	0.0855	14.2	0.941	0.997	0.916	0.998
55	0.176	0.102	0.124	11.1	0.915	0.824	0.957	0.922
56	0.0363	0.0816	0.0529	11.3	0.963	0.964	0.963	0.984
57	0.0391	0.0643	0.0629	11.6	0.961	0.961	0.96	0.976

Iteracija	OR_p	UR_p	ER_p	RDE	ACC	SENS	SPEC	NPV
58	0.0892	0.229	0.184	16.6	0.878	0.911	0.862	0.95
59	0.226	0.0432	0.141	12.2	0.908	0.774	0.981	0.889
60	0.0697	0.0537	0.0821	14.6	0.951	0.93	0.965	0.954
61	0.293	0.0968	0.174	13.2	0.882	0.707	0.964	0.875
62	0.000866	0.302	0.0827	10.2	0.931	0.999	0.917	1
63	0.00216	0.386	0.344	23.5	0.778	0.998	0.657	0.998
64	0.029	0.0871	0.06	12.4	0.96	0.971	0.954	0.985
65	0.102	0.0375	0.0648	11.5	0.956	0.898	0.983	0.953
66	0	0.194	0.133	14.5	0.914	1	0.867	1
67	0	0.363	0.118	17.1	0.902	1	0.882	1
68	0.334	0.0236	0.261	10.1	0.85	0.666	0.988	0.799
69	0.00994	0.097	0.0516	11.6	0.964	0.99	0.953	0.995
70	0.00112	0.323	0.11	11.6	0.911	0.999	0.891	1
71	0.0175	0.149	0.0933	13.7	0.937	0.982	0.915	0.991
72	0.00168	0.132	0.0755	13.7	0.949	0.998	0.925	0.999
73	0.0119	0.12	0.0771	13	0.95	0.988	0.929	0.993
74	0.00581	0.103	0.0607	12.4	0.96	0.994	0.942	0.997
75	0.00158	0.225	0.158	15.4	0.897	0.998	0.842	0.999
76	0.0341	0.0776	0.0492	12.6	0.965	0.966	0.965	0.985
77	0.0661	0.046	0.0606	12.4	0.961	0.934	0.975	0.964
78	0.0347	0.0777	0.0556	14.1	0.962	0.965	0.961	0.983
79	0.0436	0.143	0.0732	12.1	0.946	0.956	0.943	0.984
80	0.0142	0.109	0.0426	10.2	0.968	0.986	0.962	0.995
81	0.0793	0.141	0.0825	13.2	0.939	0.921	0.946	0.971
82	0	0.252	0.13	15.3	0.906	1	0.87	1
83	0.00522	0.164	0.112	13.6	0.928	0.995	0.891	0.997
84	0	0.554	0.438	26.1	0.676	1	0.562	1
85	0	0.158	0.0675	11.8	0.95	1	0.933	1
86	0.026	0.165	0.127	14.6	0.92	0.974	0.888	0.983
87	0.00904	0.215	0.0665	10.2	0.946	0.991	0.936	0.998
88	0.0016	0.266	0.0764	9.88	0.937	0.998	0.924	1
89	0.0151	0.155	0.0616	12.4	0.953	0.985	0.943	0.995
90	0.00713	0.085	0.0372	11.4	0.973	0.993	0.965	0.997
91	0.00262	0.159	0.0786	11.4	0.944	0.997	0.922	0.999
92	0.000491	0.205	0.102	13.4	0.927	1	0.898	1
93	0.0392	0.0846	0.103	17.4	0.943	0.961	0.929	0.967
94	0.00597	0.146	0.0852	12.3	0.943	0.994	0.918	0.997
95	0.00185	0.143	0.101	16.6	0.937	0.998	0.9	0.999
96	0.0114	0.121	0.104	18.3	0.939	0.989	0.904	0.991
97	0.000625	0.194	0.12	12.7	0.92	0.999	0.88	1
98	0.0451	0.0438	0.0525	13.3	0.967	0.955	0.974	0.973
99	0.00887	0.106	0.0421	11.2	0.968	0.991	0.961	0.997
100	0.00431	0.141	0.0632	11.1	0.954	0.996	0.938	0.998

B priedas. Pranešimo XXII-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga



XXII-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“
ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA
Albinas Jaruševičius

Vaizdų technologijų (T111), Sistemų inžinerijos ir kompiuterių technologijų (T120), Signalų technologijų (T121), Aukštų dažnių technologijų (T191) jungtinėje sekcijoje perskaitė pranešimą

**GAMTA GRĮSTŲ METAEURISTINIŲ
ALGORITMŲ VAIZDAMS APDOROTI TYRIMAI**

Mokslo komiteto pirmininkas

prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas

dr. Tomyslav Sledevič

2019 m. kovo 15 d.
Vilnius