VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Metaeuristinių algoritmų tyrimas ir jų taikymas transporto maršrutų optimizavime

Metaheuristics Algorithms Research and its Application for Vehicle Routing Problem Optimization

Magistro baigiamasis darbas

Atliko:	Dmitrij Mamajev	(parašas)
Darbo vadovas:	prof. habil. dr. Antanas Žilinskas	(parašas)
Recenzentas:	prof. dr. Rimantas Vaicekauskas	(parašas)

Vilnius – 2017

Santrauka

Greitas ir ekonomiškas prekių pristatymas yra logistikos iššūkis. Pastaruoju metu daug dėmesio susilaukė technologinė galimybė prekes pristatyti bepiločiais orlaiviais – dronais. Vienas iš variantų yra autotransporto priemonę sujungti su vienu arba keliais bepiločiais orlaiviais, kad prekės būtų pristatomos ir automobiliais, ir bepiločiais orlaiviais. Kilo visiškai naujas uždavinys, kurį galima pavadinti transporto maršrutų sudarymo uždaviniu (angl. Vehicle Routing Problem; VRP) su bepiločiais orlaiviais, arba dronais.

Kadangi tai palyginti naujas uždavinys, jam spręsti reikalingi nauji algoritmai, kurie padėtų geriau suprasti tokio tipo uždavinius ir leistų konstruoti optimalius maršrutus, automobilius derinant su bepiločiais orlaiviais. Savo darbe VRP su bepiločiais orlaiviais spręsti pritaikiau modeliuojamo atkaitinimo algoritmą. Šis algoritmas tinka įvairių tipų VRP uždaviniams spręsti. Palyginsiu gautus rezultatus, kai prekės pristatomos automobiliais su dronais ir be jų, naudodamas modeliuojamo atkaitinimo algoritmą įvertinsiu, kiek efektyvu prekėms pristatyti naudoti bepiločius orlaivius kartu su automobiliais.

Raktiniai žodžiai: transporto uždavinys, meta-euristika, modeliuojamo atkaitinimo algoritmas, transporto uždavinys su dronais, euristinis algoritmas.

Summary

The fast and cost-efficient delivery of goods is challenging problem of todays logistics. The companies are looking for new ways to deliver goods to their customers. One technology-enabled opportunity that recently has received much attention is the use of a drone in conjunction with trucks to support deliveries. This type of delivery in which a truck collaborates with a drone to make deliveries gives rise to a new variant of the Vehicle Routing Problem (VRP) that can be called the VRP with drones (VRP-D).

In my master thesis, I will create simulated annealing algorithm to solve this new VRP-D. I will compare results when goods are delivered by truck-only versus combined delivery of truck and drone. I will provide conclusion on what type of delivery is more cost-effective.

Keywords: vrp, vehicle routing problem, metaheuristics, simulated annealing algorithm, vehicle routing problem with drones

Turinys

Įvadas

Greitas ir ekonomiškas prekių pristatymas yra logistikos iššūkis. Daugelis įmonių ieško naujų būdų, kaip optimizuoti paskutinį, vadinamąjį „paskutinės mylios“ (angl. last-mile delivery [MD14]), pristatymo etapą iki gavėjo. Pastaruoju metu daug dėmesio susilaukė galimybė panaudoti bepiločius orlaivius prekėms gabenti. Pagal inovatyvią pristatymo koncepciją autotransporto priemonė sujungiama su vienu arba keliais bepiločiais orlaiviais prekių pristatymo tikslais. Atsiradus tokiai galimybei, atsirado ir visiškai naujas transporto maršrutų sudarymo uždavinio (angl. Vehicle Routing Problem; VRP) variantas, kurį galima pavadinti VRP su dronais, arba VRP-D.

Palyginti su įprastomis transporto priemonėmis, svarbus bepiločio orlaivio privalumas tas, kad jį galima eksploatuoti be brangiai kainuojančio žmogaus piloto darbo. Kitas privalumas tas, kad dronas yra greitas, gali skristi virš perpildytų kelių arba pristatyti krovinius į vietas, kurių kitos transporto priemonės negalėtų pasiekti, pavyzdžiui, kalnuose.

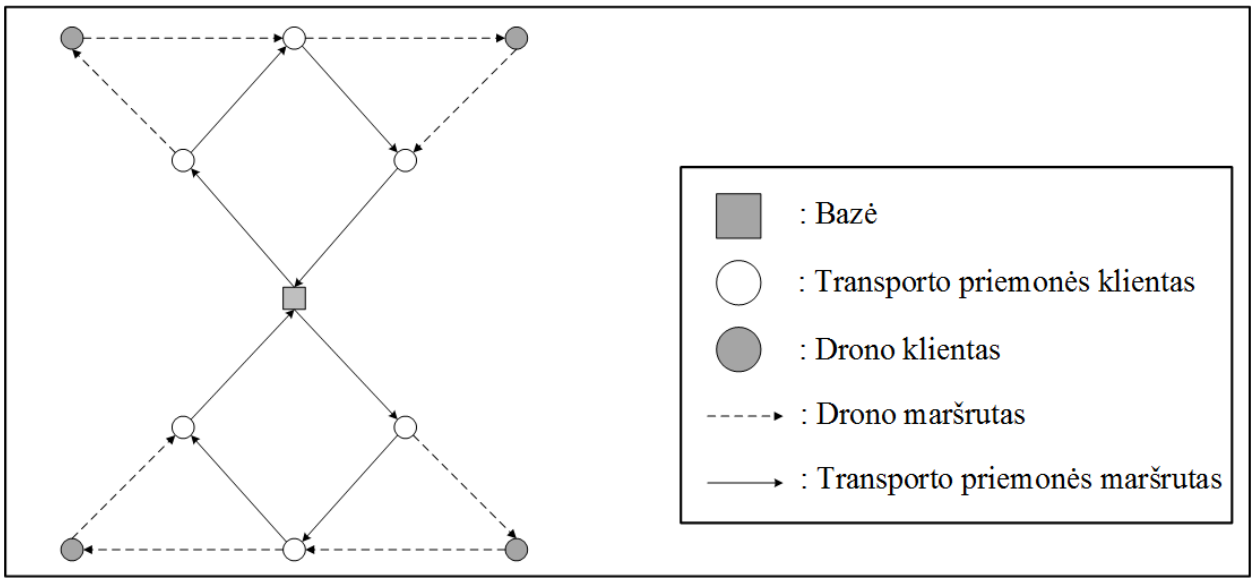
Kelios bendrovės, įskaitant „Amazon“, „Alibaba“ ir „Google“, šiuo metu atlieka praktinius bandymus ir tiria dronų naudojimo siuntiniams pristatyti galimybes [Pop13]. Šiems bandymams dažniausiai naudojami bepiločiai orlaiviai su keliais propeleriais; tokie orlaiviai gali pristatyti iki 30 kilogramų sveriančius siuntinius iki 30 kilometrų spinduliu. Yra dronų, kurie jau naudojami prekių pristatymo praktikoje, kol kas tik ne miesto aplinkoje. Pavyzdžiui, kompanija DHL neseniai pradėjo teikti pristatymo dronais paslaugą vaistams ir kitoms skubiai reikalingoms prekėms pristatyti į vieną iš Vokietijos salų Šiaurės jūroje [Her14]. Šiuo atveju dronas skrenda savarankiškai, tačiau jį vis tiek reikia nuolat stebėti. Aeronautikos specialistai tikisi, kad, atsižvelgiant į sparčiai tobulinamą kliūčių nustatymo ir vengimo technologiją, po kelerių metų dronai sugebės savarankiškai ir saugiai skristi miesto aplinkoje [JG15].

Nors dronas yra greita ir santykinai nebrangi transporto priemonė vertinant pagal vienam kilometrui tenkančias sąnaudas, tačiau jis turi ir tam tikrų apribojimų. Pavyzdžiui, maksimalus siuntinių kiekis, kurį dronas gali gabenti, priklauso nuo jo dydžio. Paprastai tai reiškia, kad po kiekvieno pristatymo dronas turėtų grįžti į bazę ir paimti kitą krovinį, o tai nėra labai efektyvu. Kitas apribojimas, kurį reikėtų paminėti, yra energijos šaltinis, arba akumuliatorius; tikėtina, kad drono aprėpiamas nuotolis išliks ribotas, palyginti su įprastinėmis transporto priemonėmis, kurios varomos degalais. Kita vertus, įprastai pristatymams naudojamos autotransporto priemonės gali aprėpti didelę teritoriją ir vienu metu gabenti daug siuntinių. Lentelėje Nr. 1 pateiktos kelios autotransporto priemonės ir bepiločio orlaivio charakteristikos.

1 lentelė. Bepiločio orlaivio ir autotransporto priemonės charakteristikos

	Krovinių svoris	Krovinių kiekis	Nuotolis
Bepilotis orlaivis	Lengvas	Vienas	Trumpas
Autotransporto priemonė	Sunkus	Daug	Ilgas

Vienas iš būdų, kaip būtų galima išplėsti drono veikimo nuotolį ir pajėgumus, yra sudaryti galimybę jam veikti kartu su autotransporto priemone. JAV bendrovė „AMP Electric Vehicles“



1 pav. VRP-D sprendinio pavyzdys

bendradarbiaudama su JAV Cincinnati universiteto Aerokosmonautikos inžinerijos katedra sukūrė droną, kuris yra montuojamas elektra varomuose sunkvežimiuose. Toks drono ir sunkvežimio derinys aptarnauja visus klientus ir padeda įvykdyti daugiau pristatymų [Woh14]. Sunkvežimis pristato prekes vienai klientų grupei, dronas tuo pačiu metu vieną po kito aptarnauja kitus klientus, po kiekvieno pristatymo grįžta prie sunkvežimio ir paima kitą siuntinį.

Vertinant iš transportavimo planavimo perspektyvos, ši inovatyvi koncepcija iškelia keletą svarbių planavimo uždavinių. Net ir vieno sunkvežimio ir vieno drono atveju šis uždavinys apima sprendimus dėl paskyrimo ir sprendimus dėl maršruto sudarymo. Priimant sprendimus dėl paskyrimo nusprendžiama, kuri transporto priemonė, dronas ar sunkvežimis, aptarnaus konkrečius klientus, o priimant sprendimus dėl maršruto sudarymo nusprendžiama, kokia seka bus lankomi klientai, priskirti kiekvienai transporto priemonei.

1 paveikslėlyje pateikiama mažo pavyzdžio iliustracija, kai iš bazės reikia aptarnauti dešimt klientų. Kaip matome, bepiločio orlaivio naudojimas keturiems klientams aptarnauti sumažina sunkvežimio nuvažiuojamą atstumą. Taikant tokį skirtingų pristatymo užduočių gretinimą, atsiranda galimybė sutrumpinti bendrą visų klientų aptarnavimo laiką.

Kadangi tai yra santykinai naujas reiškinys, reikalingi nauji modeliai ir algoritmai, kurie padėtų geriau suprasti susijusius planavimo uždavinius ir galimą naudą.

Mano magistro darbo tikslas yra sukurti ir pritaikyti modeliuojamo atkaitinimo algoritmą VRP ir VRP-D uždaviniams spręsti.

Šiam tikslui pasiekti yra iškelti tokie uždaviniai:

1. Išnagrinėti egzistuojančius algoritmus, skirtus transporto maršrutų sudarymo uždaviniams spręsti.

2. Sukurti ir pritaikyti modeliuojamo atkaitinimo algoritmą, skirtą VRP ir VRP-D uždaviniams spręsti.

3. Įvertinti sukurtą algoritmą, testuojant jį su moksliniuose tyrimuose naudojamais testiniais

duomenimis.

4. Palyginti VRP ir VRP-D gautus rezultatus, pateikti išvadas ir siūlymus dėl bepiločių orlaivių naudojimo kartu su autotransporto priemonėmis prekėms pristatyti.

1. Literatūros apžvalga

Transporto maršrutų sudarymo uždaviniai yra plačiai nagrinėjama kombinatorinė problema, kuria domisi jau egzistuojančių bei naujai kuriamų algoritmų tyrėjai. Paprastai VRP uždavinys formuluojamas kaip vieno krovinių paskirstymo punkto, aibės klientų bei kelių transporto priemonių uždavinys, kurio tikslas – kiekvienai transporto priemonei parinkti arba sukonstruoti tokį maršrutą, kuris optimizuotų pristatymo kaštus. Transporto maršrutų sudarymo uždaviniuose galimi apribojimai. Literatūroje nagrinėjami įvairūs VRP tipai, kur uždaviniai sugrupuoti priklausomai nuo įtraukiamų apribojimų. Dažniausiai pasitaikantys transporto maršrutų sudarymo uždavinių tipai yra šie:

- CVRP – VRP su talpos apribojimais, kur transporto priemonių talpa yra ribota;
- VRPTW – VRP su laiko langais, kur klientai gali būti aptarnaujami tik nustatytu laikotarpiu ar laikotarpiais;
- MDVRP – VRP su keliais krovinių paskirstymo punktais, iš kurių klientams gali būti pristatytos prekės;
- VRPPD – VRP su surinkimu ir pristatymu, kur apibrėžiamos sąlygos surinkti prekes iš vienų vietų ir pristatyti į kitas vietas.

Minėtiems uždaviniams spręsti literatūroje siūloma įvairių būdų, tarp jų ir metaeuristiniai.

Pastaruoju metu VRP uždaviniai pritraukia daug dėmesio dėl populiarėjančių įvairių geografinių sistemų, naujų technologijų atsiradimo ir jų naudojimo logistikos ir transportavimo srityje. Vis daugiau logistikos kompanijų stengiasi pasinaudoti šiomis technologijomis tam, kad galėtų geriau organizuoti prekių gabenimą. Tai gali būti įvairios logistikos sistemos, sujungtos su plačiai naudojamomis pozicionavimo sistemomis. Sėkmingas transporto maršrutų sudarymo uždavinių sprendimas yra vienas iš transportavimo išlaidų mažinimo būdų. Pavyzdžiui, tinkamesnis įvairiose srityse (paštui pristatyti, prekėms pristatyti į parduotuves, degalams gabenti į degalines ir t. t.) naudojamų transporto priemonių maršrutų planavimas gali padėti sutaupyti degalų, pinigų ir (arba) laiko, kuris galėtų būti skirtas naujiems klientams aptarnauti. Maršrutų organizavimas gali verslui padėti mažinti taršą ir daryti įtaką ekologiniams aspektams, kurie šiandieniniame pasaulyje plačiai aptarinėjami.

1.1. VRP uždavinio apibrėžimas

Klasikinis VRP uždavinys yra apibrėžiamas kaip grafas $G = (E,V)$, kuris susideda iš mazgų aibės $V = \{v_0, v_1, \ldots, v_k\}$, žyminčios krovinių paskirstymo punktą v_0 bei reikalingas aplankyti vietas $V \setminus \{v_0\}$, ir aibės $E = \{e_{ij}\}$, žyminčios trumpiausius kelius tarp šių mazgų.

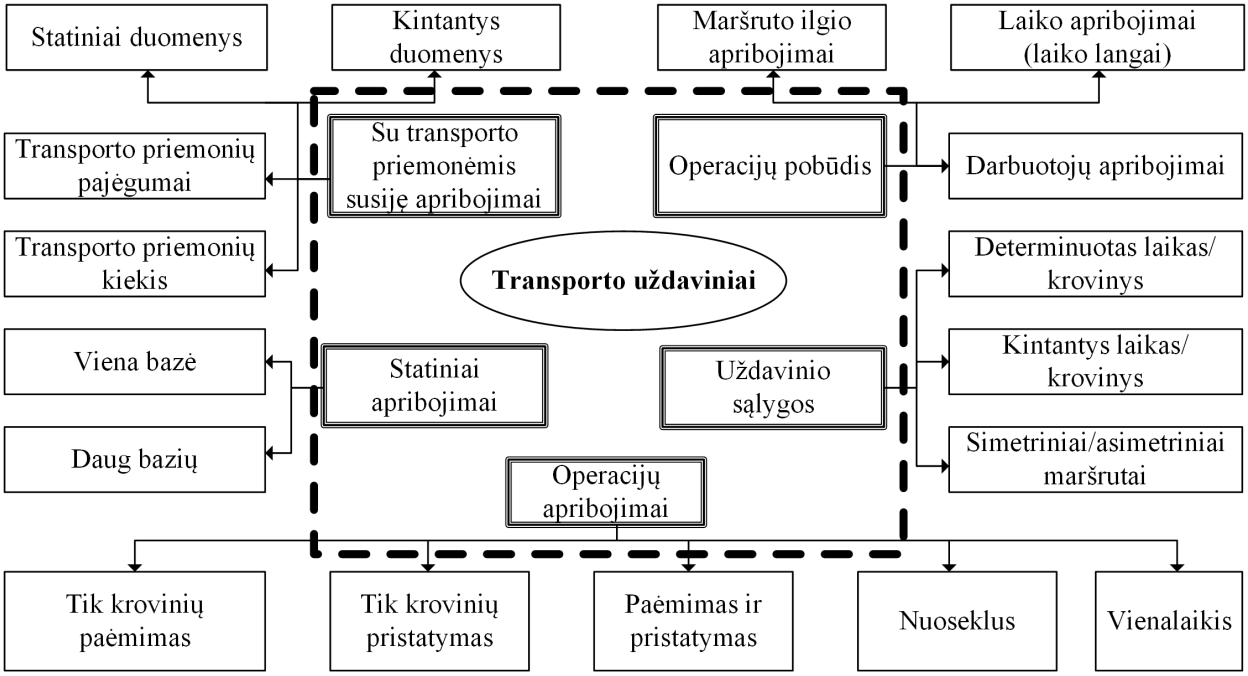
VRP uždaviniai skirti optimaliems maršrutams nustatyti duotam transporto priemonių parkui, atsižvelgiant į tam tikrus apribojimus. VRP uždavinių sprendiniai gali būti panaudoti įvairiose srityse: įvairių prekių tiekimo grandinės planavimui, kurjerių paslaugoms, viešajam transportui, šiukšlių surinkimui ir išvežimui ir kt. Kiekvienas uždavinys turi konkrečius apribojimus, paremtus

užduoties skubumu, transporto priemonės galimybėmis, asimetriniais atstumas ir t.t. Didžiausias iššūkis – surinkti duomenis, juos analizuoti ir rasti optimalų sprendimą, kurį būtų galima pritaikyti realioje situacijoje.

1.2. VRP uždavinių klasifikacija

Autoriai [AGM14] VRP uždavinius suskirstė į penkias klases pagal šias ypatybes:

1. Statiniai apribojimai
2. Su transporto priemonėmis susiję apribojimai
3. Operacijų pobūdis
4. Uždavinio sąlygos
5. Operacijų apribojimai



2 pav. VRP uždavinių klasifikacija pagal [AGM14]

1.2.1. VRP su statiniais apribojimais

- VRP su vienu paskirstymo punktu.

  Tokio tipo VRP uždavinys yra vienas iš standartinių VRP uždavinių, kurio pagrindinis reikalavimas yra rasti tokius pristatymo maršrutus, kad transporto priemonės pradėtų pristatymus iš vieno paskirstymo punkto ir atlikusios visas užduotis grįžtų atgal į jį, kad būtų aptarnauti visi taškai, o bendras nuvažiuotas atstumas – kuo mažesnis.

- VRP su keliais paskirstymo punktais.

  Uždavinyje su keliais paskirstymo punktais transporto priemonės gali būti sutelkiamos viename iš paskirstymo punktų arba keliuose paskirstymo punktuose. Kiekviena transporto

priemonė turi išvažiuoti ir grįžti į tą patį paskirstymo punktą. Taip pat taikytini visi standartinio VRP apribojimai. Tipinis tokio uždavinio pavyzdys yra picų pristatymas į namus. Net ir mažus kelių bazių uždavinius optimaliai išspręsti gana sudėtinga.

1.2.2. Su transporto priemonėmis susiję apribojimai

- Vienodos talpos transporto priemonių VRP. Šis maršrutų sudarymo uždavinys susijęs su fiksuotu arba kintamu transporto priemonių, kurių talpa yra vienoda, skaičiumi.
- Įvairios talpos transporto priemonių VRP. Šiame uždavinyje transporto priemonių talpa skiriasi, o sąnaudos gali būti vienodos arba skirtingos.
- Nevienodas fiksuotas transporto priemonių parkas. Tokio tipo uždavinyje transporto priemonių parke yra fiksuotas skaičius daugiau kaip vieno tipo (ir skirtingos talpos) transporto priemonių.
- Mišrus transporto priemonių parkas. Į mišraus parko VRP įeina tam tikras skaičius skirtingų transporto priemonių. Šiuo atveju kiekvieno tipo transporto priemonės pasižymi tam tikra talpa ir sąnaudomis (fiksuotomis ir kintamomis). Visi transporto priemonių maršrutai turi prasidėti bazėje, pasiekti reikiamus taškus ir vėl grįžti į bazę. Tikslas – kuo mažesnėmis sąnaudomis pasiekti visus taškus.

1.2.3. Operacijų pobūdis

VRP uždaviniai pagal prekių paėmimo ir (arba) pristatymo operacijas priklauso vienai iš šių keturių kategorijų:

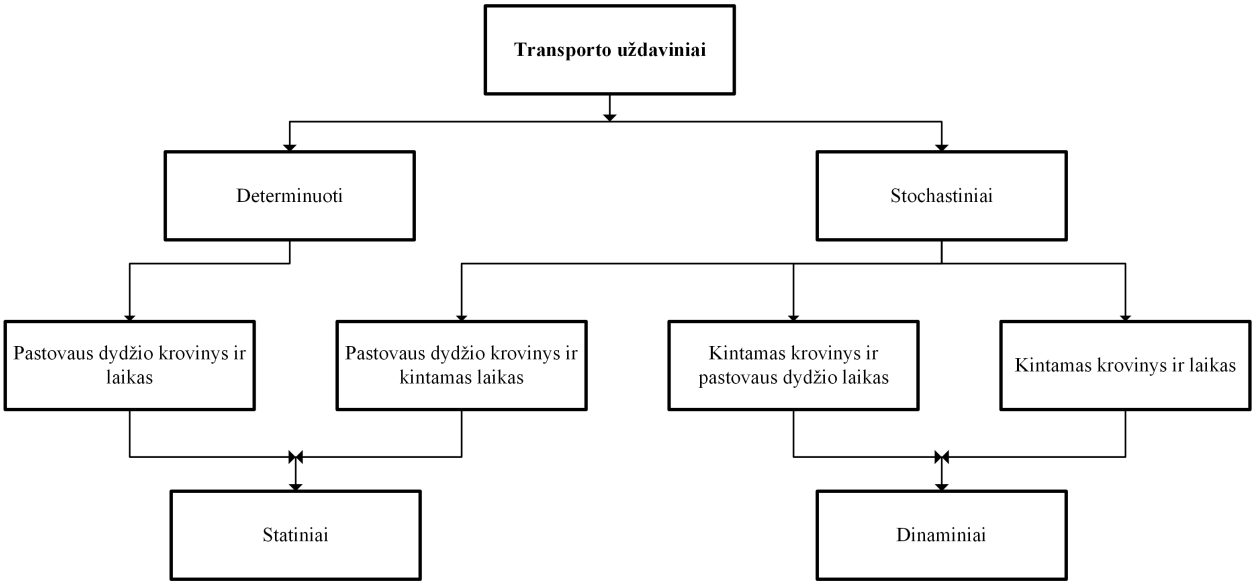
1. Vien paėmimas arba vien pristatymas
2. Paėmimas, paskui pristatymas
3. Pristatymas, paskui paėmimas
4. Mišrus pristatymas ir paėmimas

[CGW88] apžvelgė uždavinį, kai iš pradžių atliekamas prekių pristatymas, o kai visos prekės būna pristatytos, konstruojami maršrutai prekėms paimti iš klientų. Daugelyje tokio tipo uždavinių prekių paėmimo operacijų skaičius būna gerokai mažesnis už prekių pristatymo operacijų skaičių. [RBL02] nagrinėjo uždavinį, kai pirmiau atliekamas paėmimas, tik paskui pristatymas. Tokiu atveju paėmimo taške reikia apsilankyti pirmiau nei pristatymo taške. Autorius [Min89] analizavo atvejį, kai maršrutai konstruojami mišriai vykdant paėmimo ir pristatymo operacijas.

Tokio tipo uždaviniai įdomūs, nes galimas vis kitoks paėmimo ir pristatymo užduočių santykis bei paskirstymas. Šiuos uždavinius dar galima skirstyti, pavyzdžiui, į tokias kategorijas: „nedaug daugeliui“ (mažai paėmimo vietų, bet kroviniai pristatomi į daug didesnį pristatymo vietų skaičių) arba „daug nedaugeliui“ ir „daug daugeliui“.

1.2.4. Uždavinio sąlygos

VRP galime laikyti determinuotu, jei kuriant transporto priemonių maršrutus žinomi visi reikalingi duomenys, ir tikimybiniu, jei yra priešingai. Be to, VRP uždavinys, kurio sąlygos nepriklauso nuo laiko, vadinamas statiniu, priešingu atveju – dinaminiu (**??** pav.).



3 pav. VRP klasifikacija pagal [GNN07]

Determinuotais laikomi VRP uždaviniai, kai kroviniai ir laikas yra konstantos, nekintančios reikšmės. Visi duomenys žinomi iš anksto ir į laiką galime neatsižvelgti.

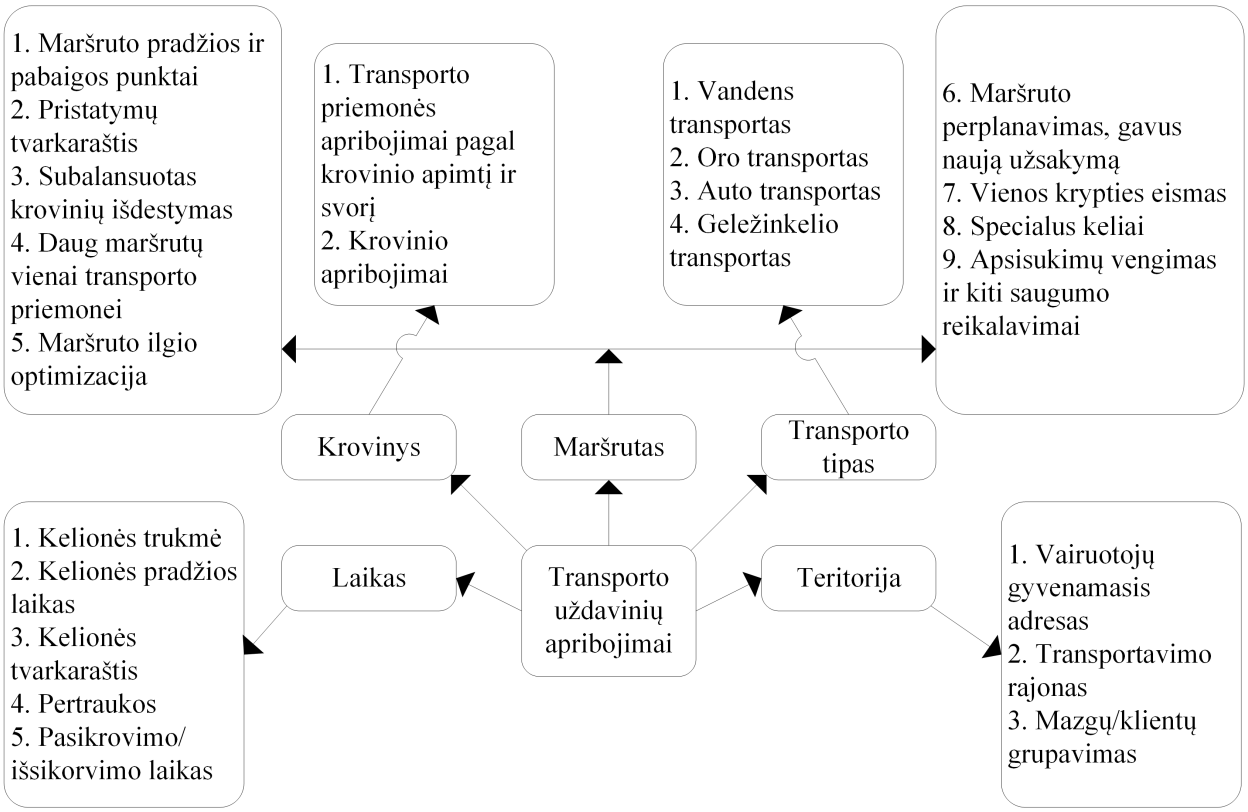
Tikimybiniai VRP atvejai. Patogumo dėlei VRP uždaviniai skirstomi į tris tipus:

- pastovaus krovinio ir kintamo laiko;
- kintamo krovinio ir pastovaus laiko;
- kintamo krovinio ir kintamo laiko.

Pastovaus krovinio ir kintamo laiko uždavinyje kroviniai, kuriuos reikia pristatyti arba paimti, žinomi iš anksto, tačiau reikia atsižvelgti ir į laiką (pvz., VRP su laiko langais). Kintamo krovinio ir pastovaus laiko uždavinyje pačioje planavimo pradžioje nustatomi transporto priemonių maršrutai, nors kroviniai dar nebūna žinomi. Viskas priklauso nuo aptarnavimo užklausų ir naudotojų poreikių. Kintamo krovinio ir kintamo laiko uždaviniuose nežinomi nei kelionės laikas, nei pristatytini ar paimtini kroviniai. Naudotojų prašymai perduodami vairuotojams realiuoju laiku.

1.2.5. Operacijų apribojimai

Įvairūs operacijų apribojimai paprastai yra susiję su transporto priemonėmis, vairuotojų tvarkaraščiais, mazgų pageidavimais ar regiono, į kurį reikia vykti, vietovės apribojimais. Dalis šių apribojimų išvardyta **??** paveikslėlyje.



4 pav. VRP apribojimų klasifikacija pagal [GNN07]

Paprastai laikoma, kad sprendinio maršrutas prasideda ir baigiasi bazėje. Tačiau kelių bazių atveju arba, pavyzdžiui, kai tam tikros transporto priemonės maršrutas prasideda arba baigiasi vairuotojo gyvenamojoje vietoje, optimalaus sprendinio radimas tampa sudėtingesnis. Kai uždaviniuose reikia rasti maršruto pradinį ir pabaigos taškus, kurie gali sutapti, o gali ir nesutapti, ir kartu rasti optimalų maršrutą, tokiu atveju sprendinys tampa kompleksinis ir susideda iš kelių kintamųjų. Todėl ieškant sprendinio būtina atsižvelgti ir į apribojimus. [LLM89] apžvelgė keletą tokio uždavinio taikymo pavyzdžių.

Norint sudaryti vairuotojo maršruto tvarkaraštį reikia nustatyti jo darbo pradžios laiką ir trukmę, taip pat atsižvelgti į krovimo taškus bei įvertinti darbo pertraukas. Kai kuriuose uždaviniuose pradžios laiko kintamajam reikia skirti daugiau dėmesio. Pavyzdžiui, laikraščių pristatymo uždaviniuose maršrutų pradžios laikas priklauso nuo spaustuvių gamybos greičio. Tai nulemia, kada transporto priemonės gali būti pakrautos.

Sudarant maršrutus, kurie tęsiasi daugiau kaip vieną darbo dieną, reikia atsižvelgti į papildomus apribojimus, susijusius su vairuotojų darbo saugos taisyklėmis. Pagal šias taisykles neleidžiama vairuoti ilgiau kaip 10 val. per parą, apskritai darbo laikas negali viršyti 15 val. per parą ir 60 val. per savaitę. Be to, į tvarkaraštį reikia įtraukti bent 8 miego valandas per parą. Tokio tipo veiklos apribojimus minėjo autoriai [FAJ80] ir [CCL+87].

Antroji maršrutų apribojimų grupė yra susijusi su transporto priemonių talpa ir apkrova. Atsižvelgiant į skirtingas prekes, praktikoje gali reikėti atsižvelgti į svorio ir (arba) tūrio apribojimus. Pavyzdžiui, naftos produktų gabenimo atveju svorio ir tūrio santykis dar priklauso ir nuo aplinkos temperatūros [BEG+87]. Todėl kartais norint nustatyti svorio apribojimus reikia

atsižvelgti ir į oro sąlygas, aplinkos temperatūrą ir kitus veiksnius, nes apribojimai gali priklausyti ir nuo vietovių, kuriomis važiuojama.

Taip pat apribojimai gali būti taikomi transporto priemonėms, važiuojančioms tam tikromis kelio atkarpomis ar tiltais. Dėl leidžiamo transporto priemonių svorio gali būti ir vietinių reikalavimų. Suprantama, kai taškų poreikiai, palyginti su maksimalia transporto priemonės talpa, yra gana dideli ir kai kelionės į tuos taškus trukmė nėra labai didelė, prireiks važiuoti kelis kartus arba pristatymą padalyti į kelias dalis.

1.3. VRP sudėtingumas

Formuluojant ir sprendžiant VRP svarbiu veiksniu laikomas uždavinio sudėtingumas. Dauguma maršrutų sudarymo uždavinių suformuluojami kaip grafo teorijos uždaviniai. Uždavinio dydį nusako grafo briaunų ir viršūnių skaičius. Kartais net ir nelabai didelio uždavinio sprendimas reikalauja tokių milžiniškų resursų, kad ieškoti optimalaus sprendimo tampa neprasminga. Nors pageidautina, kad VRP uždaviniai būtų sprendžiami taikant polinominio sudėtingumo algoritmus, paprastai jie patenka į NP uždavinių klasę. Autoriai [PS82] įrodė, kad VRP priklauso NP-sunkių uždavinių klasei. Todėl norint rasti beveik optimalius sprendinius per protingą laiką, paprastai tokiems uždaviniams reikia naudoti euristinius arba metaeursitinius algoritmus.

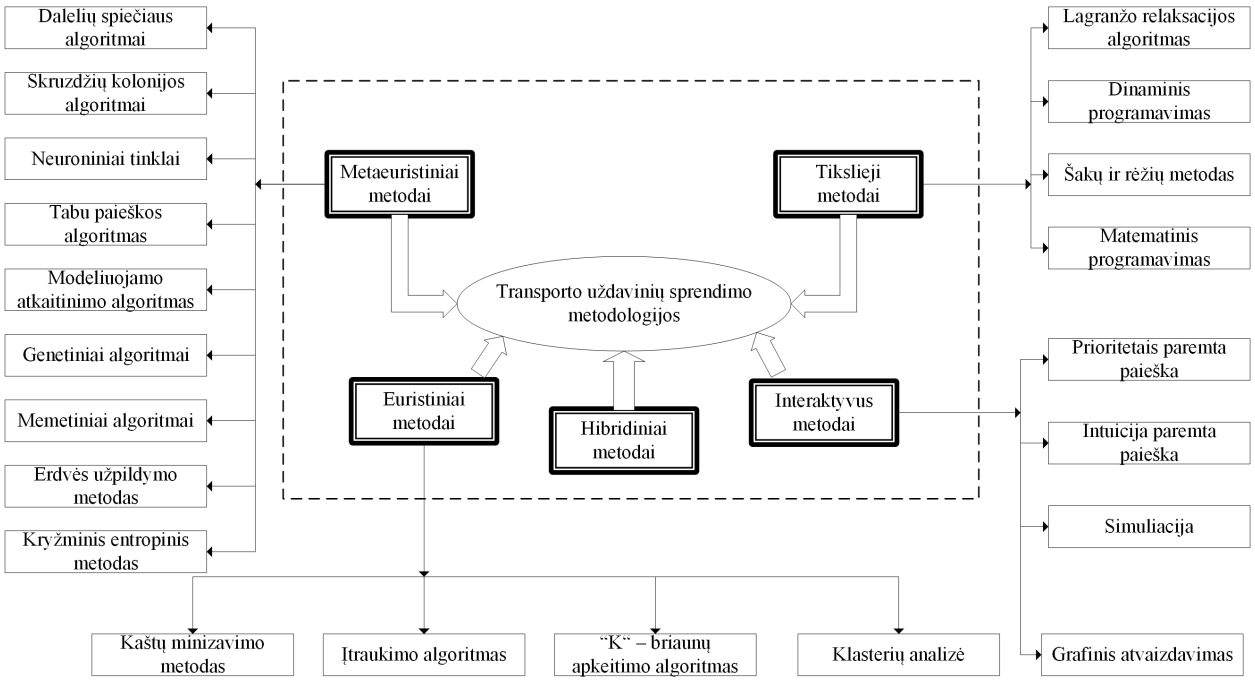
2. VRP uždavinių sprendimo būdai

Kadangi VRP uždaviniai priklauso NP sudėtingumo uždavinių klasei, jiems spręsti kuriami ir tiriami įvairūs tikslūs, euristiniai ir metaeuristiniai algoritmai. Nors tikslūs algoritmai randa optimalų sprendinį, didėjant uždavinio apimčiai skaičiavimo laikas žymiai pailgėja. NP sudėtingumo uždaviniams spręsti egzistuoja įvairūs metaeuristiniai metodai, tačiau lokalios paieškos ir metaeuristinių algoritmų efektyvumas gali priklausyti nuo uždavinio duomenų, pavyzdžiui, apribojimų, arba gali reikėti papildomų duomenų algoritmui apmokyti. Siekiant didesnio efektyvumo skaičiavimuose nagrinėjami ir tiriami įvairūs hibridiniai algoritmai, susidedantys iš kelių algoritmų.

Sprendžiant VRP uždavinį, nagrinėtoje literatūroje yra siūlomi keli skirtingi sprendimo būdai, kurie yra suskirstyti pagal algoritmų klases:

1. Tikslieji algoritmai.
2. Euristiniai algoritmai.
3. Metaeuristiniai algoritmai.
4. Hibridiniai algoritmai.

Autorių [AGM14] schemoje pavaizduota, kokiais algoritmais sprendžiami VRP uždaviniai ir kokiai kategorijai tie algoritmai priklauso.



5 pav. VRP uždavinių sprendimo būdai

2.1. Tikslieji algoritmai

Tikslieji algoritmai garantuoja uždavinio optimalaus sprendinio radimą. Tačiau bendrai paėmus beveik visų kombinatorinio optimizavimo, tarp jų, žinoma, ir VRP uždavinių, tiksliųjų

algoritmų vykdymo laikas auga eksponentiškai, didėjant uždavinio apimčiai. Jei uždaviniui spręsti neegzistuoja tikslus algoritmas, toks uždavinys priskiriamas NP–pilnajai uždavinių sudėtingumo klasei. Dažniausiai visos kombinatoriniam optimizavimui priskiriamos problemos priklauso NP–pilnajai klasei. Laikoma, kad uždavinys yra išsprendžiamas kompiuteriu, jeigu jo algoritmo vykdymo laiko funkcija polinominė, ir atvirkščiai, uždavinys yra kompiuteriu neišsprendžiamas, jei jo algoritmo vykdymo laiko funkcija auga eksponentiškai. Todėl per priimtiną skaičiavimų laiką tiksliai išspręsti pavyksta tik nedidelės apimties kombinatorinio optimizavimo uždavinius. Tai būtų bene vienintelis, tačiau reikšmingas tiksliųjų algoritmų trūkumas.

Tiksliuosius algoritmus, skirtus spręsti VRP uždavinius, ypač VRP su talpos apribojimais, sudaro šakų ir ribų, šakų ir rėžių bei šakų ir kainų algoritmai.

Bendras tiksliųjų algoritmų veikimo principas yra toks: perrenkamos visos įmanomos uždavinio sprendinių kombinacijos ir išrenkama geriausia. Toks sprendimo būdas tinka maršrutams su nedaug klientų-taškų, kuriems reikia pristatyti prekes, ir transporto priemonių, kuriomis reikia aplankyti visus klientus-taškus, kadangi uždavinio sudėtingumas yra $O(n + m - 1)!$, kur n yra aplankytinų taškų skaičius, o m yra transporto priemonių skaičius, ir $n > m$. Tai yra labai didelis skaičius kombinacijų net ir šiuolaikiniams kompiuteriams, kadangi turėdami bent 10 aplankytinų taškų ir 4 autotransporto priemones mes gauname 13, kas yra lygu: 6227020800 palyginimo kombinacijų.

Algoritmo laikas auga eksponentiškai su kiekvienu nauju tašku arba pridėjus naują transporto priemonę, taigi praktiškai tokie algoritmai nėra taikomi, nes per normalų laiką apskaičiuoti uždavinį su 20-30 klientų-taškų tampa praktiškai neįmanoma.

2.2. Euristiniai algoritmai

Euristiniu laikomas toks optimizavimo uždavinio sprendimo metodas, kuriuo siekiama rasti aukštos kokybės, bet nebūtinai optimalų sprendinį per priimtiną skaičiavimų laiką. Tuo euristiniai algoritmai iš esmės skiriasi nuo anksčiau minėtų tikslių metodų. Šios rūšies algoritmai remiasi tam tikrų taisyklių, kurios vadinamos euristikomis, pritaikymu sprendžiant konkretų kombinatorinio optimizavimo uždavinį. Tokios taisyklės šiuo atveju nurodo sprendimą, kurį reikia priimti konkrečiose situacijose, o šio sprendimo pasirinkimas gali padėti gauti geresnį sprendinį. Euristiniai algoritmai pritaikomi tik tam tikriems konkretiems uždaviniams su tam tikrais apribojimais ar prielaidomis.

Euristiniai algoritmai sprendžia uždavinius mokymosi, atradimo būdais ir žymiai pagreitina sprendimo procesą, tačiau jie nėra tikslieji algoritmai, todėl sprendžia uždavinius su tam tikromis paklaidomis. Pagrindinis klausimas – kokia ta paklaida: ar ji nėra labai didelė ir ar didinant tikslumą nėra prarandama daug laiko, kuris reikalingas sprendimui rasti. Euristiniams algoritmams tenka balansuoti tarp tikslumo ir laiko, reikalingo uždaviniui išspręsti.

Sukurta daug euristinių metodų VRP uždaviniams spręsti. Iš pradžių tai buvo maršrutų konstravimo metodai, vėliau atsirado ir kitos euristikos, kurios padėjo spręsti VRP uždavinius. Galima paminėti, kad beveik visi euristiniai metodai yra pritaikyti spręsti VRP uždavinius su nustatytų autotransporto priemonių skaičiumi m, jie gali būti pritaikyti spręsti uždavinius, esant

bet kokiam skaičiui $0 < m < \infty$.

Toliau apžvelgsiu keletą euristinių algoritmų, kuriais naudojantis sprendžiami įvairių tipų VRP uždaviniai.

2.2.1. Artimiausio kaimyno algoritmas

Artimiausio kaimyno algoritmas (angl. Nearest Neighbour algorithm) – pats paprasčiausias iš euristinių algoritmų. Jo pagrindinė mintis – pradėti nuo tam tikro maršruto taško, sekančiu maršruto tašku pasirinkti artimiausią esantį tašką ir juos sujungti. Prijungtam taškui surandamas jo artimiausias taškas, bet jis negali būti iš jau sujungtų taškų aibės. Taip kartojama tol, kol sujungiami visi taškai.

2.2.2. Įtraukimo algoritmas

Įtraukimo algoritmas (angl. Insertion Heuristics) [Mer99] irgi yra vienas iš paprasčiausių euristinių algoritmų. Jo veikimo principas toks: iš karto sujungiami labiausiai nutolę taškai – taip formuojamas pradinis maršrutas. Toliau į maršrutą likę nesujungti taškai įtraukiami 2 būdais:

1. Ieškant artimiausio taško, nutolusio nuo pradinės kelionės. Iteracijos kartojamos tol, kol visi taškai būna sujungti; per kiekvieną iteraciją mažinamas pradinės kelionės atstumas.

2. 2. Ieškant tolimiausio taško, nutolusio nuo pradinės kelionės. Iteracijos kartojamos tol, kol visi taškai būna sujungti. Pasirenkamas būdas, kurio pradinėmis iteracijomis gaunamas mažiausias atstumas.

2.2.3. 2-jų briaunų apkeitimo algoritmas

2-iejų briaunų apkeitimo algoritmas (angl. 2-opt algorithm) yra gana paprastas ir efektyvus. Pradinė kelionė gali būti sudaroma dviem būdais:

- Pradinė kelionė sudaroma pasinaudojant kitu euristiniu algoritmu, dažniausiai pasirenkamas artimiausio kaimyno principas, nes jis paprasčiausias ir greičiausias, o dviejų briaunų apkeitimo algoritmas tiesiog atlieka patobulinimus, taip sutrumpindamas kelionę.

- Pradinė kelionė sudaroma atsitiktiniu būdu sujungiant taškus. Toks būdas naudojamas rečiau, nes reikalauja daugiau apkeitimo operacijų.

Briaunų sujungimo principas: pasirenkamos 2 poros viršūnių, bandoma jas sujungti kitu būdu, nei jos jau yra sujungtos, ir tikrinama, ar kelionės atstumas sumažėja. Jei sumažėja, pakeitimas paliekamas, pasirenkamos sekančios dvi poros ir atliekama ta pati operacija. Iteracijos atliekamos tol, kol egzistuoja poros, kurios dar nebuvo keičiamos.

2.2.4. k-jų briaunų apkeitimo algoritmas

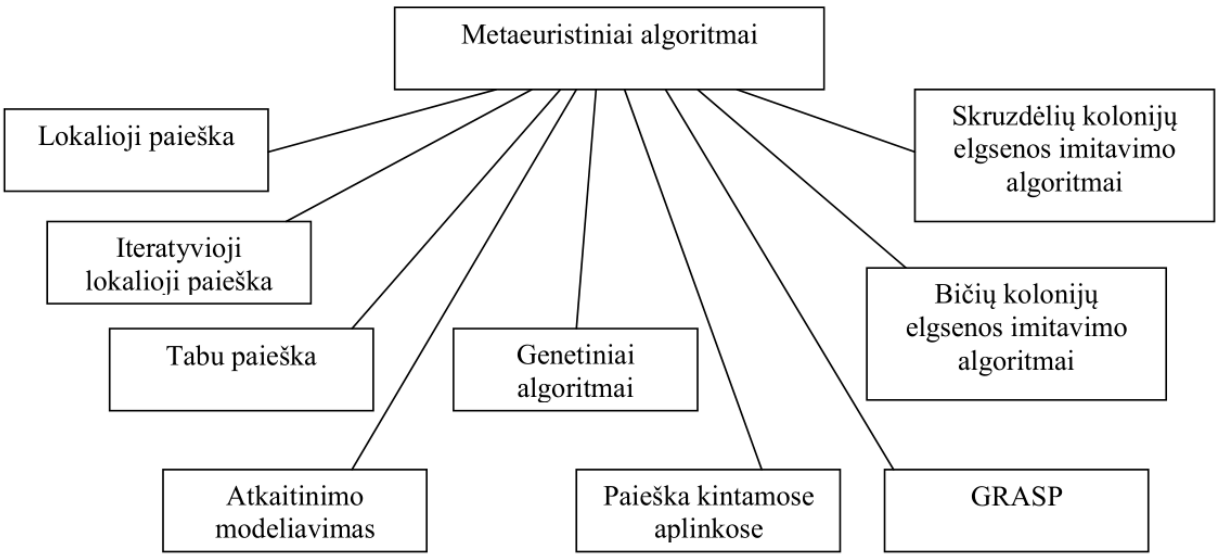
Algoritmo veikimo principas lygiai toks pats kaip ir dviejų briaunų, tik yra pasirenkamos ne dvi, o „k“ briaunų, bandant jas sujungti skirtingais būdais, taip mažinant atstumą. Skaičiaus „k“ didinimas didina kelionės tikslumą, bet tai yra brangi operacija, kadangi artinant skaičių prie turimų taškų skaičius yra artinamas prie pradinio tikslaus algoritmo – visų įmanomų kombinacijų perrinkimo. Dažniausiai praktikoje naudojamas dviejų arba trijų briaunų apkeitimo principas.

2.2.5. Lin-Kernighan algoritmas

Algoritmas paremtas ankščiau išvardytų „k“ – briaunų sujungimo principu [LK73]. Vienintelis skirtumas tas, kad šis algoritmas adaptyvus, ir kiekviename žingsnyje parenkamas vis kitas skaičius „k“. Šis algoritmas laikomas vienu iš efektyviausių euristinių algoritmų su gana maža paklaida ir trumpu vykdymo laiku.

2.3. Metaeuristiniai algoritmai

Metaeuristinius metodus apibrėšime kaip tam tikro aukšto abstrakcijos lygio nurodymų rinkinius. Priešingai nei euristinių algoritmų, tokių nurodymų paskirtis yra formaliai aprašyti kurios nors klasės uždavinių sprendimo idėją, principą. Taigi sąvoka „metaeuristinis algoritmas“ yra talpesnė nei „euristinis metodas“. Kombinatorinio optimizavimo uždaviniams spręsti naudojami šie metaeuristiniai algoritmai: lokalioji paieška, iteratyvioji lokalioji paieška, tabu paieška, genetiniai algoritmai, godžiosios randomizuotos adaptyvios paieškos procedūros (GRASP), atkaitinimo modeliavimas, skruzdėlių kolonijų algoritmai, paieška kintamose aplinkose, bičių kolonijų algoritmai ir kt.



6 pav. Metaeuristiniai algoritmai

Toliau apžvelgsime keletą metaeuristiniu algoritmų, naudojamų VRP uždaviniams spręsti.

2.3.1. Modeliuojamo atkaitinimo metodas

Modeliuojamojo atkaitinimo algoritmas (angl. Simulated annealing) paremtas fizikinio proceso, kuris vyksta grūdinant metalą, analogija. Šis algoritmas buvo sukurtas remiantis analogijomis iš statistinės mechanikos, modeliuojant procesus sistemose, sudarytose iš didelio skaičiaus mažų diskrečiųjų dalelių. Modeliuojant tokios sistemos „atkaitinimą“, pradžioje visa sistema yra pakaitinama iki aukštos temperatūros, vėliau sistemos temperatūra laipsniškai mažinama. Temperatūra mažinama tol, kol sistema „užsigrūdina“, t. y. pereina į optimalią būseną – pusiausvyros būseną.

Šio algoritmo pradininkais laikomi [MRR+53]. Jie pasinaudojo Bolcmano (eksponentiniu) pasiskirstymo dėsniu apibrėždami tikimybę, kad sistema, įvykus joje tam tikrai perturbacijai-pasikeitimui, pereis iš vieno energijos lygio (E_1) į kitą (E_2), kai temperatūra lygi t:

$$P = \begin{cases} 1 & \text{,kai } \Delta E > 0 \\ e^{-\Delta E/C_B t} & \text{,kai } \Delta E \leq 0 \end{cases}$$

, čia $\Delta E = E2 – E1$, o C_B – Bolcmano konstanta.

Vėliau autoriai [KGV83] pirmieji panaudojo atkaitinimo modeliavimo metodą spręsdami kombinatorinio optimizavimo uždavinius.

Svarbus šio algoritmo parametras yra temperatūra, mažėjanti su kiekvienu atkaitinimo proceso imitavimo žingsniu. Šis parametras reguliuojamas įvedant specialią temperatūros atnaujinimo (aušinimo) funkciją. Šiame algoritme įvedamas specialus nuo temperatūros priklausantis tikimybinis patvirtinimo kriterijus, kuriuo remiantis kiekviename žingsnyje priimamas surastas geresnis sprendinys nei esamas ir su nedidele tikimybe gali būti priimamas blogesnis sprendinys. Ši savybė leidžia „ištrūkti“ iš lokalių optimumo taškų, ieškant globalaus optimumo.

Atkaitinimo modeliavimo procese temperatūra laipsniškai mažinama, pradedant nuo tam tikros pradinės reikšmės t_0. Ši reikšmė neturėtų būti nei per didelė, nei per maža. Iš tikrųjų, jei pradinė temperatūra būtų pernelyg aukšta, atkaitinimo procesas užsitęstų labai ilgai, nereikalingai nagrinėjant daug „blogų“ sprendinių. Kita vertus, per žema pradinė temperatūra gali lemti patekimą į nebūtinai gero lokaliojo optimumo zoną. Kartais naudojamas pusiausvyros testas temperatūros mažinimo momentui nustatyti. Pusiausvyra apibūdinama stabilios proceso būsenos kriterijumi.

Kombinatorinių uždavinių atveju pusiausvyros kriterijus galėtų būti, pavyzdžiui, tikslo funkcijos reikšmių svyravimų amplitudė (jeigu ji nedidelė, laikoma, jog pusiausvyra pasiekta). Atsižvelgiant į pusiausvyros testą, skiriami du atkaitinimo tipai: homogeninis ir nehomogeninis. Pirmuoju atveju bandymai atliekami su ta pačia fiksuota temperatūra, kol pasiekiama pusiausvyra; tada temperatūra mažinama, ir procesas kartojamas. Antruoju atveju temperatūra mažinama po kiekvieno bandymo. Pusiausvyros testas neatliekamas. Kai kuriose atkaitinimo modeliavimo algoritmų versijose temperatūra mažinama ir didinama periodiškai. Tyrimų, atliktų dar 1989 m., rezultatai rodo, kad tikslinga taikyti „atkaitinimų“ ir „kaitinimų“ seką, vadinamą pakartotiniu atkaitinimu (angl. ReAnnealing), nes tai suteikia papildomų lankstesnių galimybių kuriant atkaitinimo modeliavimo algoritmus. Modeliuojant atkaitinimą svarbu parinkti tinkamą

„aušinimo“ funkciją. Galimos įvairios aušinimo funkcijos.

Atkaitinimo procesas teoriškai turėtų būti tęsiamas tol, kol galutinė temperatūra taps lygi 0. Tačiau atkaitinimą galima baigti ir anksčiau, kai tampa mažai tikėtina, jog bus pasiektas pagerėjimas, pavyzdžiui, kai tikslo funkcijos reikšmė nemažėja gana ilgą laiką arba kai, atlikus pakankamai daug bandymų, priimtų teigiamų sprendimų (t. y. perėjimų iš vieno sprendinio į kitą) skaičius tampa mažesnis už tam tikrą slenkstį. Dažnai kaip baigimo sąlyga tarnauja fiksuotas atliekamų bandymų skaičius.

2.3.2. Tabu paieška

Tabu paieškos (angl. Tabu search) algoritmas remiasi tam tikrų sprendinių uždraudimo principu, siekiant „ištrūkti“ iš lokaliųjų optimumų zonų. Pagrindinis šio metodo parametras – tabu sąrašas (angl. Tabu list), kuris pradžioje būna tuščias, po to yra papildomas ir modifikuojamas. Į šį sąrašą atliekant optimizavimą įtraukiami jau nagrinėti sprendiniai arba jų aplinkos. Tad šiame metode draudžiama tam tikrą laiką grįžti prie jau nagrinėtų sprendinių arba jų aplinkų, kol draudimas bus išstumtas iš fiksuoto ilgio ir nuolat kintančio tabu sąrašo. Tabu paieškos metodas remiasi išplėsta lokaliąja paieška. Skirtingai negu lokaliosios paieškos procedūros, kurios apsiriboja lokaliai optimalaus sprendinio (lokaliojo optimumo) suradimu nagrinėjamo sprendinio aplinkoje (kaimyninių sprendinių aibėje), tabu paieška pagrįsti algoritmai tęsia paiešką ir tuo atveju, kai surandamas lokalinis optimumas, t. y. kai duoto sprendinio aplinkoje neįmanoma rasti geresnio sprendinio.

Tabu paieška pradedama nuo pradinio sprendinio $s^{(0)}$ iš galimų sprendinių aibės S. Pradinis sprendinys gali būti sugeneruotas tiesiog atsitiktiniu būdu. Algoritmo vykdymo metu analizuojama sprendinio $s \in S$ aplinkos (kaimyninių) sprendinių aibė $N(s)$. Baigus analizę, pereinama į tą sprendinį s' iš $N(s)$, kuriam tikslo funkcijos f reikšmė yra mažiausia. Perėjimas atliekamas ir tuo atveju, kai tikslo funkcijos pokytis yra teigiamas (t. y., tikslo funkcijos reikšmė ”pablogėja”) – taip galima pereiti nuo vieno lokalaus optimalaus sprendinio prie kito. Grįžimas į anksčiau nagrinėtą sprendinį turi būti uždraudžiamas tam tikram iteracijų skaičiui. Taigi nagrinėtieji sprendiniai yra įtraukiami į tabu sąrašą. Tokiu būdu perėjimas į sprendinį $s' \in N(s)$ yra draudžiamas, jeigu tas sprendinys (ar tam tikras požymis, susijęs su tuo sprendiniu) duotu metu yra tabu sąraše.

Besąlyginis sprendinių draudimas gali apriboti paiešką kai kuriose sprendinių aibės S srityse. Todėl kartu su tabu sąrašu naudojamas aspiracijos kriterijus. Naudojantis šiuo kriterijumi galima anuliuoti ”tabu būseną”, esant tam tikroms sąlygoms. Vienas iš standartinių aspiracijos kriterijų yra toks: ”perėjimas” iš sprendinio s į sprendinį s' – nors jis ir yra ”tabu” – leidžiamas, jeigu tenkinama sąlyga $f(s') < f(s^*)$, kur s^* yra geriausias iki šiol per paiešką surastas sprendinys.

Tabu paieškos algoritmo lankstumui padidinti tabu sąrašas, tiksliau, tabu sąrašo ilgis (dydis), vykdant algoritmą gali būti kartkartėmis atnaujinamas. Paieška tabu algoritmu užbaigiama, atlikus iš anksto nustatytą paieškos iteracijų (bandymų) skaičių. Tačiau galima įvesti ir kitokias algoritmo stabdymo sąlygas. Tabu paieškos algoritmo rezultatas yra geriausias surastas sprendinys, kuris nebūtinai sutampa su sprendiniu, gautu paskutinėje paieškos iteracijoje, nes geriausias rastas sprendinys įsimenamas tuo momentu, kai jis surandamas.

Tabu paieškos algoritmai skiriasi tabu sąrašo tvarkymu, aspiracijos kriterijumi bei kitokiais veiksniais. Taip pat reikėtų paminėti, jog tabu paieškos algoritmą galima įtraukti į kitas metaeuristines procedūras, pavyzdžiui, genetinius algoritmus. Tokiu būdu gautuose hibridiniuose genetiniuose algoritmuose tabu paieškos procedūros pritaikomos generuoti pradinėms aukštos kokybės sprendinių „populiacijoms“, arba geriausiems sprendiniams atrinkti.

Šio algoritmo pradininku laikomas Fred Glover, pasiūlęs šią lokalios paieškos schemą. Tabu paieškos metodas sėkmingai išbandytas kai kuriems kombinatorinio optimizavimo uždaviniams, tarp jų ir kvadratinio paskirstymo uždaviniui.

Vieni pirmųjų autoriai [CS93] pasiūlė tabu paieškos algoritmą keliaujančio pirklio uždaviniui spręsti. Šiame uždavinyje sprendinį sudaro viršūnių, kurios turi būti apeinamos nuoseklia tvarka, sąrašas. Algoritme panaudotas fiksuoto ilgio (dydžio) tabu sąrašas, kurį sudaro jau perstatytos viršūnės. Vykdant algoritmą tas sąrašas tvarkomas pagal taisyklę „pirmasis įėjo – pirmasis išėjo“ (angl. first in – first out; FIFO), t. y. sprendinys (ar to sprendinio požymis), kuris anksčiausiai įtraukiamas į sąrašą, anksčiausiai iš to sąrašo ir pašalinamas. Paieškos procedūra pagrįsta tokia strategija: duotame sprendinyje neleidžiama sukeisti (perstatyti) elementų, priklausančių tabu sąrašui (tai reiškia, jog tie elementai buvo sukeisti ne anksčiau kaip prieš i iteracijų), nebent sukeitus tuos elementus būtų gautas geresnis sprendinys negu geriausias iki tol surastas. Perėjus prie naujo sprendinio (atlikus elementų sukeitimą), sukeistieji elementai įtraukiami į tabu sąrašą, tuo pačiu pašalinant iš to sąrašo anksčiausiai į jį įtrauktą elementų porą – tai porai anuliuojama „tabu būsena“.

Autorius [Tai91] pasiūlė randomizuotos tabu paieškos algoritmą. Kitaip negu [CS93], [Tai91] algoritme buvo panaudotas dinamiškai (atsitiktiniu būdu) kintančio ilgio (dydžio) tabu sąrašas. Šiame algoritme yra modifikuota sprendinių elementų įtraukimo į tabu sąrašą ir jų nagrinėjimo tvarka ir panaudotas aspiracijos kriterijus: jeigu kuri nors elementų pora nebuvo sukeista per gana didelį iteracijų skaičių, tai tos poros elementai yra sukeičiami ir pereinama į naują sprendinį, nepriklausomai nuo „tabu būsenos“ ir tikslo funkcijos reikšmės. Svarbus [Tai91] algoritmo ypatumas, jog einamojo sprendinio (perstatymo) s aplinka $N(s)$ išnagrinėjama panaudojant $O(n^2)$ operacijų vietoje $O(n^3)$ operacijų [CS93] algoritme. Panaudojant šį algoritmą, gauti geri keliaujančio pirklio uždavinio sprendimo rezultatai.

Sukurta ir kitų tabu paieška besiremiančių algoritmų. Autoriai [BT94] sukūrė reaktyviosios tabu paieškos algoritmą, kuris efektyviau sprendė kai kuriuos keliaujančio pirklio uždavinio testinius pavyzdžius, tačiau šio metodo įgyvendinimas yra sudėtingesnis.

2.3.3. Genetiniai algoritmai

Genetinius algoritmus (angl. Genetic Algorithms) ir jų sudarymo principus XX amžiaus aštuntąjį dešimtmetį pasiūlė Hollandas [Hol92]. Pradedant Hollando pirmuoju darbu, genetiniai algoritmai buvo sėkmingai išbandyti sprendžiant įvairius optimizavimo uždavinius, pavyzdžiui, tokius kaip elektroninių komponentų išdėstymas, operacijų planavimas ir kt. Genetinių algoritmų veikimas pagrįstas evoliucijos, vykstančios gyvojoje gamtoje, t. y. natūralios atrankos, proceso imitavimu. Pagrindinės sąvokos, vartojamos modeliuojant biologinės evoliucijos procesus, yra „individas“ ir „populiacija“. Individas yra tam tikras elementarus, neskaidomas vienetas, objektas.

Pseudokodas 1 Tabu paiėškos algoritmas [Bro11]

```
Input: TabuList_size
Output: S_best
S_best ← ConstructInitialSolution()
TabuList ← ∅
While (¬ StopCondition())
  CandidateList ← ∅
  For (S_candidate ∈ Sbest_neighborhood)
    If (¬ ContainsAnyFeatures(S_candidate, TabuList))
      CandidateList ← S_candidate
    End
  End
  S_candidate ← LocateBestCandidate(CandidateList)
  If (Cost(S_candidate) ≤ Cost(S_best))
    S_best ← S_candidate
    TabuList ← FeatureDifferences(S_candidate, S_best)
    While (TabuList > TabuList_size)
      DeleteFeature(TabuList)
    End
  End
End
Return (S_best)
```

Didesnė ar mažesnė individų grupė sudaro populiaciją. Dar vienas svarbus dalykas yra vadinamasis individo tinkamumas – savotiška individo vertė. Individo vertę galima traktuoti kaip individo gebėjimo sėkmingai prisitaikyti (prie aplinkos), išlikti ir reprodukcijos lygį. Šia prasme vertingesnis (grynai biologiškai) yra tas individas, kuris sugeba geriausiai prisitaikyti (yra stipresnis už kitus) ir, galbūt, palikti daugiau palikuonių („vaikų“).

Optimizavimo procese vietoje sąvokų „individas“, „populiacija“, „individo vertė“ vartojamos tradicinės, įprastos sąvokos: „individą“ atitinka atskiras sprendinys, „populiaciją“ – sprendinių aibė (grupė, rinkinys), o „individo vertė“ asocijuojama su tikslo funkcijos reikšme duotajam sprendiniui. Taip, kaip gyvosios gamtos evoliucijoje išlieka tik vertingiausi (stipriausi) individai, taip ir optimizavimo procese siekiama gauti kuo geresnį sprendinį, t. y. sprendinį su kuo mažesne (ar didesne) – nelygu, koks optimizavimo uždavinys (minimizavimo ar maksimizavimo) sprendžiamas, – tikslo funkcijos reikšme.

Genetiniai algoritmai priklauso metaeuristinių metodų klasei, kuriai būdingos tokios savybės:

1. Operuojama su viena ar daugiau leistinų sprendinių ”populiacijų”.

2. Atskiri sprendiniai iš vienos ar kelių populiacijų parenkami (panaudojant kurią nors euristiką) tolesniam nagrinėjimui, apdorojimui, teikiant pirmenybę tiems sprendiniams, kurie turi geresnes tikslo funkcijos reikšmes.

3. Naudojamas tam tikras mechanizmas, skirtas formuoti naujiems leistiniems sprendiniams, kombinuojant (derinant) anksčiau gautus sprendinius ar kurias nors jų savybes,

charakteristikas.

4. Esant reikalui, naudojamas papildomas mechanizmas naujiems galimiems sprendiniams iš atskirų anksčiau gautų sprendinių generuoti (atliekant atsitiktinius tų sprendinių elementų perstatymus (perturbacijas)).

5. Atlikus (2)–(4) žingsnius, esama (-os) populiacija (-os) atnaujinama, pašalinant iš jos (jų) tam tikrus sprendinius (paprastai blogesnius) bei paliekant joje (jose) geresniuosius sprendinius, tarp jų naujai suformuotus (sugeneruotus) (jeigu jie pakankamai geri).

Formalizuojant genetinių algoritmų aprašymą, pirmiau minėtos bendrosios savybės, mechanizmai susiejami su atitinkamomis procedūromis. Taip (2) savybė susiejama su vadinamąja atrinkimo procedūra, (3) mechanizmas tradiciškai vadinamas krosoveriu, (4) mechanizmas – mutavimo procedūra (mutatoriumi). (5) etape (žingsnyje) atliekamas populiacijos atnaujinimas (savotiškas filtravimas, išgryninimas).

Apibendrintas genetinio algoritmo pseudokodas pateikiamas toliau:

1. sukurti pradinę (atsitiktinių) leistinų sprendinių populiaciją (arba kelias populiacijas);

2. kartoti :

   (a) parinkti sprendinius-tėvus;

   (b) vykdyti krosoverį parinktiems sprendiniams-tėvams, gaunant sprendinį-palikuonį, kuris įtraukiamas į populiaciją;

   (c) vykdyti mutavimo procedūrą tam tikriems duotosios populiacijos sprendiniams;

   (d) atnaujinti esamą populiaciją;

3. baigti, kai patenkinama baigimo sąlyga.

Egzistuoja daugybė įvairių būdų, kaip konkrečiai realizuoti sprendinių-tėvų parinkimo, krosoverio, mutavimo bei populiacijos sprendinių atnaujinimo procedūras. Pavyzdžiui, galime operuoti viena didele sprendinių populiacija, tačiau galime turėti ir kelias mažesnes (lygiagrečias) populiacijas (subpopuliacijas). Toliau galime pasirinkti tėvus, atsižvelgdami į jų vertingumą (tinkamumą), t. y. absoliutinę tikslo funkcijos reikšmę, arba į kokius nors kitus įverčius, charakteristikas ar kriterijus. Galima įvairiai realizuoti krosoverio bei mutavimo procedūras, be to, galima nuspręsti į vieną procedūrą sujungti krosoverį ir mutavimą arba tai daryti atskiromis nepriklausomomis procedūromis. Pagaliau galime apsispręsti pakeisti visus be išimties einamosios populiacijos narius (sprendinius) naujai gautais palikuonimis po kiekvieno algoritmo ciklo iteracijos įvykdymo. Dar reikia pastebėti, kad genetinio algoritmo realizacija gali priklausyti nuo to, kaip yra vaizduojami, pateikiami sprendiniai, kitaip tariant, nuo sprendinio kodavimo pobūdžio.

Pseudokodas 2 Genetinis algoritmas [Bro11]

```
Input: Population_size, Problem_size, P_crossover, P_mutation
Output: S_best
Population ← InitializePopulation(Population_size, Problem_size)
EvaluatePopulation(Population)
S_best ← GetBestSolution(Population)
While (¬StopCondition())
  Parents ← SelectParents(Population, Population_size)
  Children ← ∅
  For (Parent_1, Parent_2 ∈ Parents)
    Child_1, Child_2 ← Crossover(Parent_1, Parent_2, P_crossover)
    Children ← Mutate(Child_1, P_mutation)
    Children ← Mutate(Child_2, P_mutation)
  End
  EvaluatePopulation(Children)
  S_best ← GetBestSolution(Children)
  Population ← Replace(Population, Children)
End
Return (S_best)
```

2.3.4. Išbarstytoji paieška

Išbarstytoji paieška (angl. Scatter Search) – tai evoliucinis metodas, naudojamas kombinatorinio ir netiesinio optimizavimo uždaviniams spręsti [RC06]. Išbarstytoji paieška operuoja sprendinių rinkiniu, vadinamu atraminiu. Kombinuojant to rinkinio sprendinius, gaunami nauji sprendiniai. Sprendiniai į atraminį rinkinį atrenkami naudojant procedūras, panašias į taikomas genetiniuose algoritmuose. Skirtingai nuo genetinių algoritmų, išbarstytoji paieška operuoja su žymiai mažesniu sprendinių rinkiniu, nes genetiniuose algoritmuose nauji sprendiniai gaunami kombinuojant nagrinėjamos populiacijos elementų poras. O kombinuojant sprendinius išbarstytoje paieškoje, kai jie imami iš atraminių sprendinių rinkinio įvairiais poaibiais, galima gauti gana daug įvairių kombinacijų. Todėl pakanka operuoti nedideliu sprendinių rinkiniu. Reikia pažymėti, kad per išbarstytąją paiešką gali būti nagrinėjami neleistini arba nepasiekiami sprendiniai. Nepasiekiamais sprendiniais vadinami sugeneruoti sprendiniai, kurie nepriklauso uždavinio galimų sprendinių aibei. Pavyzdžiui, per sveikaskaičio optimizavimo uždavinio sprendinio išbarstytąją paiešką gali būti sugeneruotas sprendinys su neigiamomis ar trupmeninėmis reikšmėmis. Toks sprendinys vadinamas nepasiekiamu. Šis sprendinys gali būti paverčiamas galimu sprendiniu, pavyzdžiui, jį suapvalinus.

Išbarstytosios paieškos algoritmas realizuojamas keliais etapais, vadovaujantis tokiomis taisyklėmis [Fel07].

1. Paieška pradedama pradinio sprendinių rinkinio generavimu, užtikrinant jo įvairovės lygį ir gerinimo galimybes.

2. Nauji sprendiniai yra generuojami, naudojant esamo atraminio sprendinių rinkinio poaibių iškilą apvalkalą. Reikia pastebėti, kad nauji sprendiniai gali būti nebūtinai leistini ir

pasiekiami.

3. Sprendiniai, gauti antrame žingsnyje, yra modifikuojami, dalį jų paverčiant leistinais ir pasiekiamais.

4. Iš generuotų ir po to modifikuotų sprendinių, yra atrenkami „geriausi“, kurie yra priskiriami atraminiam rinkiniui. Sprendinio vertė priklauso nuo jį atitinkančios funkcijos reikšmės bei nuo galimybės gauti efektyvius sprendinius tolesnės evoliucijos metu. Išbarstytoji paieška yra stabdoma atlikus nustatytą iteracijų skaičių.

Pseudokodas 3 Išbarstytosios paieškos algoritmas [Bro11]

```
Input: DiverseSet_size, ReferenceSet_size
Output: ReferenceSet
InitialSet ← ConstructInitialSolution(DiverseSet_size)
RefinedSet ← ∅
For (S_i ∈ InitialSet)
  RefinedSet ← LocalSearch(S_i)
End
ReferenceSet ← SelectInitialReferenceSet(ReferenceSet_size)
While (¬ StopCondition())
  Subsets ← SelectSubset(ReferenceSet)
  CandidateSet ← ∅
  For (Subset_i ∈ Subsets)
    RecombinedCandidates ← RecombineMembers(Subset_i)
    For (S_i ∈ RecombinedCandidates)
      CandidateSet ← LocalSearch(S_i)
    End
  End
  ReferenceSet ← Select(ReferenceSet, CandidateSet, ReferenceSet_size)
End
Return (ReferenceSet)
```

2.3.5. Skruzdžių kolonijos algoritmas

Daug metų buvo stebima skruzdžių kolonijų elgsena ir jų gebėjimas rasti trumpiausią maršrutą iki maisto šaltinio. Nustatyta, kad skruzdės, eidamos link maisto šaltinio, palieka chemines medžiagas – feromonus. Jei maršrutas ilgas, juo keliauja mažiau skruzdžių, todėl feromonai jame išgaruoja greičiau nei trumpame maršrute. Taip ilgesnis maršrutas skruzdėms tampa mažiau patrauklus ir pasirenkamas trumpesnis maršrutas, turintis daugiau feromonų. Toks skruzdžių elgsenos imitavimas buvo panaudotas optimizavimo uždaviniams spręsti.

Skruzdžių kolonijų optimizavimo (ACO) principai pirmą kartą aprašyti Marco Dorigo darbe 1992 metais [DS03]. Paprastai skruzdžių kolonijų optimizavimo algoritmas taikomas kombinatoriniams optimizavimo uždaviniams, tokiems kaip keliaujančio pirklio uždavinys, tvarkaraščių sudarymo ir kiti, spręsti.

Bendra algoritmo veikimo paradigma yra tokia:

1. Nustatomi pradiniai algoritmo parametrai;
2. Sukuriamas pasirinktas skaičius skruzdžių;
3. Kiekviena skruzdė konstruoja sprendinį – konstruojant sprendinį, sprendinio elementai pasirenkami tikimybiškai. Tikimybė, kad bus pasirinktas tam tikras elementas, priklauso nuo jam priskirto feromonų kiekio;
4. Priklausomai nuo to, kokios kokybės sprendinys buvo gautas, kiekvienam sprendinio elementui atnaujinamas feromonų kiekis;
5. Visiems galimiems elementams sumažinamas tam tikras feromonų kiekis (garavimas). Taip išvengiama lokalių optimumų;
6. Tikrinama, ar pasiektas tikslas, jei ne, grįžtama į antrą žingsnį.

Pseudokodas 4 Skruzdžių kolonijos algoritmas [Bro11]

```
Input: ProblemSize, Population_size, m, ρ, α, β
Output: P_best
P_best ← CreateHeuristicSolution(ProblemSize)
Pbest_cost ← Cost(S_h)
Pheromone ← InitializePheromone(Pbest_cost)
While (¬StopCondition())
  Candidates ← ∅
  For (i = 1 To m)
    S_i ← ProbabilisticStepwiseConstruction(Pheromone, ProblemSize, α, β)
    Si_cost ← Cost(S_i)
    If (Si_cost ≤ Pbest_cost)
      Pbest_cost ← Si_cost
      P_best ← S_i
    End
    Candidates ← S_i
  End
  DecayPheromone(Pheromone, ρ)
  For (S_i ∈ Candidates)
    UpdatePheromone(Pheromone, S_i, Si_cost)
  End
End
Return (P_best)
```

2.3.6. Metaeuristinių algoritmų apibendrinimas

Lokaliosios paieškos algoritmai remiasi teiginiu, kad turimas pradinis uždavinio sprendinys gali būti pagerintas atlikus mažus to sprendinio pakeitimus. Šiuose algoritmuose atliekami tik

tokie nedideli pradinio sprendinio pakeitimai, kuriuos atlikus sumažėja pasirinktos tikslo funkcijos reikšmė. Lokaliosios paieškos algoritmo gautas sprendinys bus tik lokalusis optimumas tikslo funkcijos prasme toje aplinkoje, į kurią pataikė pradinis sprendinys. Šiuo atveju bet kokio pradinio sprendinio aplinka galėtume pavadinti tokią aibę kitų sprendinių, kurie gaunami atlikus visus galimus nedidelius pradinio sprendinio pakeitimus. Lokaliosios paieškos algoritmų sprendinių kokybė smarkiai priklauso nuo pradinio sprendinio.

Atkaitinimo modeliavimo algoritmai remiasi fizikinio proceso – atkaitinimo (grūdinimo) idėja. Vienas iš pagrindinių parametrų – temperatūra T, kuri su kiekvienu šio proceso imitavimo žingsniu nuolat mažėja. Įvedama speciali temperatūros atnaujinimo (aušinimo) funkcija, taip pat specialus nuo temperatūros priklausomas tikimybinis patvirtinimo kriterijus. Jo esmė – kiekviename šio metodo žingsnyje patvirtinti rastus geresnius nei esamas sprendinius ir su nedidele tikimybe patvirtinti blogesnius sprendinius. Tai suteikia galimybę ištrūkti iš lokalių optimumo taškų ieškant globalaus optimumo.

Tabu paieškos metodas remiasi tam tikrų sprendinių uždraudimo idėja. Pagrindinis jo parametras – tabu sąrašas, kuris iš pradžių yra tuščias. Į jį vykstant procesui vis įtraukiami tie sprendiniai (ar jų aplinkos), kuriuos mes jau aplankėme. Tokiu būdu šis metodas kuriam laikui uždraudžia grįžimą prie jau buvusių sprendinių (jų aplinkų), kol iš fiksuoto ilgio ir nuolat kintančio tabu sąrašo bus išstumtas draudimas. Taigi, tabu paieškos algoritmuose, priešingai nei likusiuose, saugoma dalis paieškos proceso praeities, kas teoriškai turėtų suteikti ir suteikia pranašumą prieš kitus algoritmus. Tabu paieškos algoritmų įvairios modifikacijos yra vieni iš efektyviausių metodų, naudojamų kombinatorinio optimizavimo uždaviniams spręsti.

Esminis genetinių algoritmų skirtumas nuo anksčiau aptartų algoritmų tas, kad čia operuojama ne vienu sprendiniu, o sprendinių populiacija – chromosomų rinkiniu. Prie tokių metodų priskiriami genetiniai algoritmai. Šie algoritmai remiasi genetikos mokslo idėjomis. Nauji sprendiniai iš turimų gaunami pritaikius genetines operacijas – kryžminimą, mutaciją, atranką. Taip gaunamos naujos sprendinių populiacijos su „geresnėmis“ savybėmis, kurios turi įtakos apibrėžtos tikslo funkcijos gerėjimui.

Godžioji randomizuota adaptyvi paieškos procedūra (GRASP) yra konstruktyvios euristikos ir lokaliosios paieškos derinys. Kiekviena GRASP iteracija susideda iš dviejų dalių: sprendinio generavimo ir sprendinio gerinimo naudojant lokaliąją paiešką. Paieškos kintamose aplinkose algoritmuose vieną iteraciją paprastai sudaro dvi fazės: suvirpinimo procedūra ir lokalioji paieška. Suvirpinimo procedūroje atliekami atsitiktinių turimo sprendinio perstatymai, kurie gali ir pagerinti, ir pabloginti sprendinį. Tuo siekiama, kad algoritmu gaunami sprendiniai neįstrigtų kurioje nors lokalaus minimumo aplinkoje. Po suvirpinimo procedūros seka lokaliosios paieškos fazė.

Pagrindines metaeuristinių algoritmų savybes:

- Metaeuristiniuose algoritmuose derinamos globaliosios ir lokaliosios paieškos procedūros, siekiant ištrūkti iš lokaliųjų optimumų traukos zonų ir tikslinant sprendinį lokaliosios paieškos euristinėmis procedūromis aptikus globaliojo ekstremumo traukos zoną.
- Metaeuristinis algoritmas turi garantuoti principinę galimybę perrinkti visus galimus

sprendinius, išvengiant žymaus tų pačių sprendinių kartotinio nagrinėjimo.

- Metaeuristiniai algoritmai gali būti klasifikuojami priklausomai nuo sprendinių populiacijos, nagrinėjamos vienoje iteracijoje, dydžio, sprendinio aplinkos tyrimo intensyvumo bei galimybės išsaugoti ir pritaikyti informaciją, gautą per optimizavimo procesą.

2.3.7. Metaeuristinių algoritmų taikymas VRP uždaviniams spręsti

Visi pirmiau minėti algoritmai taikomi VRP uždaviniams spręsti. Tai yra pagrindinis metaeuristikų pranašumas prieš kokiam nors vienam konkrečiam uždaviniui spręsti tinkamus euristinius algoritmus. Norint rasti geresnės kokybės sprendinius, dėl metaeuristinių metodų universalumo dažnai galime sujungti kelių skirtingų rūšių metaeuristinių algoritmų idėjas. Todėl šiuo metu metaeuristiniai algoritmai ir įvairios jų ir kelių metaeuristinių metodų idėjų kombinacijos, pavyzdžiui, genetiniai algoritmai ir lokalioji paieška, tabu paieška, atkaitinimo modeliavimas ir kt., yra pagrindiniai tyrinėjimų, kuriais siekiama sukurti algoritmus, kuo efektyviau sprendžiančius kombinatorinio optimizavimo uždavinius, tarp jų ir VRP, objektai.

3. VRP-D uždavinio sprendimas

3.1. VRP-D uždavinio formulavimas

VRP uždavinys yra apibrėžiamas kaip grafas $G = (V, E)$, kuris susideda iš mazgų aibės $V = \{0,\ldots,n\}$ ir briaunų aibės $E = \{e_{ij}\}$. Kiekviena viršūnė $i \in V \setminus \{0\}$ atitinka klientą, kuriam priskiriamas neneigiamas poreikis q_i, tuo tarpu viršūnė 0 atitinka bazę. Su kiekviena grafo briauna $e \in E = \{(i,j) : i,j \in V, i < j\}$ yra siejamos kelionės išlaidos c_e arba c_{ij}. Bazėje yra fiksuotas identiškų transporto priemonių parkas m, o kiekvienos iš jų pajėgumas yra lygus Q. Simetrinio VRP atveju, reikia surasti m maršrutų, kurių bendra kelionės išlaidų suma būtų mažiausia, ir kurie atitiktų tam tikrus apribojimus ir sąlygas:

1. Kiekvienas klientas yra aplankomas tik vieną kartą viename maršrute.
2. Kiekvienas maršrutas prasideda ir baigiasi bazėje.
3. Bendri maršruto eigoje aptarnaujamų klientų poreikiai neviršija transporto priemonės pajėgumų Q.
4. Kiekvieno maršruto ilgis neviršija iš anksto nustatytos ribos L.

VRP-D sprendinį galima apibrėžti kaip grupę m ciklų, turinčių bendrą viršūnę bazėje.

Straipsnio [ABS16] autoriai atliko kruopščią keliaujančio pirklio uždavinio su dronu analizę. Remiantis šių autorių atliktu darbu ir atsižvelgiant į tai, kad transporto maršrutų sudarymo uždavinys yra keliaujančio pirklio uždavinio apibendrinimas, galima daryti šias prielaidas:

- Kiekvieną VRP-D maršrutą galima pateikti kaip atskirą keliaujančio pirklio su dronu uždavinio sprendinį.
- Dronas gali gabenti tik vieną siuntą/krovinį.
- Dronas po kiekvieno pristatymo turi grįžti prie sunkvežimio.
- Siuntinių paėmimas iš sunkvežimio gali vykti tik klientų viršūnėse arba bazėje. Tai yra, dronas gali nusileisti ir pakilti nuo sunkvežimio tik jam stovint kliento buvimo vietoje ar bazėje.
- Norint supaprastinti žymėjimą, mes darome prielaidą, kad gali būti neatsižvelgiama į drono akumuliatorių įkrovimo laiką, pavyzdžiui, keičiant akumuliatorius.

3.2. Modeliuojamo atkaitinimo algoritmo charakteristikos

Konkrečios atkaitinimo modeliavimo algoritmų realizacijos skiriasi viena nuo kitos šiais veiksniais (faktoriais): aplinkos funkcija, atkaitinimo („atšaldymo“) schema ir baigimo sąlyga. Formuojant atkaitinimo schemą, yra svarbūs tokie veiksniais:

- pradinės (galutinės) temperatūros parinkimas;

- pusiausvyros testas;
- temperatūros mažinimo formulė.

Modeliuojant atkaitinimą temperatūra nėra nekintamas dydis, ji turi būti laipsniškai mažinama, pradedant nuo tam tikros pradinės reikšmės t_0. Ši reikšmė neturėtų būti nei per didelė, nei per maža. Iš tikrųjų, jei pradinė temperatūra būtų pernelyg aukšta, atkaitinimo procesas užsitęstų labai ilgai, nereikalingai nagrinėjant daug „blogų“ sprendinių. Kita vertus, per daug žema pradinė temperatūra galėtų lemti greitą „įkritimą“ į nebūtinai gero lokaliojo optimumo „duobę“. Apibrėžiant konkrečią reikšmę t_0, ji galėtų būti prilyginama, pavyzdžiui, Δf_{max}, čia Δf_{max} – didžiausias (teigiamas) tikslo funkcijos pokytis, gautas dviems ”sprendiniams kaimynams” atlikus tam tikrą skaičių bandymų.

Atkaitinimo modeliavimo algoritmuose naudojamas vadinamasis pusiausvyros testas siekiant nustatyti, kuriuo momentu turi būti mažinama temperatūra. Pusiausvyra trumpai gali būti charakterizuojama kaip tam tikra stabili, be žymesnių fluktuacijų proceso būsena. Kombinatorinių uždavinių atveju pusiausvyros kriterijus galėtų būti, pavyzdžiui, tikslo funkcijos reikšmių svyravimų amplitudė (jeigu ji nedidelė, konstatuojama, kad pusiausvyros būsena pasiekta). Atsižvelgiant į tai, kaip realizuojamas pusiausvyros testas, skiriami du atkaitinimo tipai: homogeninis ir nehomogeninis atkaitinimas. Pirmuoju atveju atliekama daug bandymų esant tai pačiai fiksuotai temperatūrai; tai tęsiama tol, kol pasiekiama pusiausvyra, – tada temperatūra sumažinama, ir procesas kartojamas. Antruoju atveju temperatūra mažinama po kiekvieno įvykdyto bandymo; galima sakyti, jog pusiausvyros testas iš viso neatliekamas.

Modeliuojant atkaitinimą svarbu parinkti tinkamą temperatūros mažinimo formulę. Praktiškai taikomuose atkaitinimo algoritmuose plačiausiai naudojamos yra: geometrinė formulė ($t_k = \alpha t_{k-1}$; t_k – einamoji temperatūros reikšmė; $k = 1, 2, ...; t_0 = const$; $0.8 \leq \alpha \leq 0.99$) [KGV83] ir Lundy-Mees formulė ($t_k = t_{k-1}/(1 + \beta t_{k-1})$; $k = 1, 2, ...; t_0 = const; \beta \ll t_0$)[AK89]. Pastaruoju metų naujausiose atkaitinimo modeliavimo algoritmų versijose temperatūra greičiau keičiama periodiškai nei monotoniškai mažinama.

Baigimo sąlyga. Atkaitinimo procesas teoriškai turėtų būti tęsiamas tol, kol galutinė temperatūra t_g tampa lygi 0. Tačiau praktiškai atkaitinimą galima baigti ir anksčiau – kai tampa mažai tikėtina, jog bus pasiektas pagerinimas, pavyzdžiui, kai tikslo funkcijos reikšmė nemažėja gana ilgą laiką arba kai, atlikus gana daug bandymų, priimtų teigiamų sprendimų (t. y. perėjimų iš vieno sprendinio į kitą) skaičius tampa mažesnis už tam tikrą slenkstį. Dažnai kaip baigimo sąlyga tarnauja a priori fiksuotas atliekamų bandymų skaičius, t. y. atkaitinimo schemos „ilgis“.

Kaip veikia modeliuojamo atkaitinimo algoritmas:

- Pirmiausia turime nustatyti pradinę temperatūrą ir sukurti atsitiktinį pradinį uždavinio sprendinį.
- Tada pradedame iteruoti tol, kol bus pasiekta stabdymo sąlyga. Paprastai sustojama kai, sistema pakankamai ataušta arba randamas pakankamai geras sprendinys.
- Kiekvienos iteracijos metu, generuojamas kaimyninis sprendinys, nedaug pakeičiant pradinį sprendinį.

- Tada nusprendžiame, ar pereiti prie kaimyninio sprendinio.
- Galiausiai sumažiname temperatūrą ir tęsiame iteracijas.

Pseudokodas 5 Modeliuojamo atkaitinimo algoritmas [Bro11]

```
Input: ProblemSize, iterations_max, temp_max
Output: S_best
S_current ← CreateInitialSolution(ProblemSize)
S_best ← S_current
For (i = 1 To iterations_max)
  S_i ← CreateNeighborSolution(S_current)
  temp_curr ← CalculateTemperature(i, temp_max)
  If (Cost(S_i) ≤ Cost(S_current))
    S_current ← S_i
    If (Cost(S_i) ≤ Cost(S_best))
      S_best ← S_i
    End
  ElseIf (Exp((CostS_current − CostS_i) / temp_curr) > Rand())
    S_current ← S_i
  End
End
Return (S_best)
```

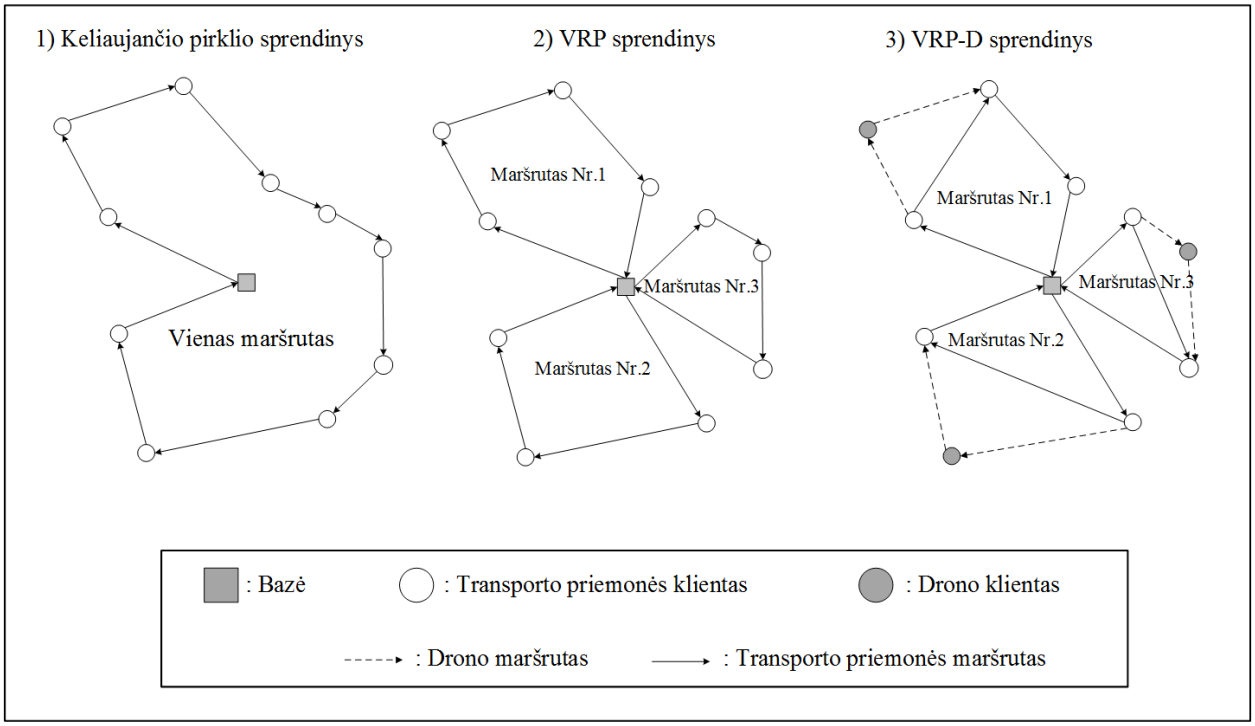
3.3. VRP-D sprendinio konstravimas

VRP-D sprendinys sudaromas trimis etapais:

1. Naudojant minėtąjį modeliuojamo atkaitinimo algoritmą sudaromas geriausias optimalus (trumpiausias) keliaujančio pirklio maršrutas.
2. Toliau keliaujančio pirklio maršrutas padalijamas į atskirus maršrutus taip, kad kiekvienas atskiras maršrutas neviršytų vieno sunkvežimio pajėgumų.
3. Sudarius vieno sunkvežimio maršrutą, sudaromas sunkvežimio su dronu maršrutas, atsižvelgiant į šiuos apribojimus:
   - m sunkvežimių, kuriuose yra k dronų, aptarnauja n klientų;
   - kiekvienam klientui reikia pristatyti vieną siuntinį, tam naudojant arba sunkvežimį, arba droną;
   - kiekviename sunkvežimyje telpa C siuntinių, dronas oru gali gabenti daugiausiai vieną siuntinį;
   - dronų akumuliatorių eksploatavimo laikotarpis yra neribotas;

- droną galima paleisti iš sunkvežimio ir jį gali priimti tas pats sunkvežimis, bet tik kai sunkvežimis yra kliento buvimo vietoje arba bazėje;
- išsiuntus droną, sunkvežimis gali toliau aptarnauti kitus klientus ir priimti droną kitoje grafo viršūnėje (kliento taške);
- sunkvežimis arba dronas gali atvykti į kliento tašką (grafo mazgą) ir vienas iš jų gali palaukti kito ;
- pristatymui nėra taikomas aptarnavimo laikas;
- tikslas yra sumažinti kiekvieno sunkvežimio maršruto atstumą.

Grafiškai VRP-D sprendinio konstravimas atrodo taip:



7 pav. VRP-D sprendinio konstravimas

4. Tyrimo dalis

4.1. Tyrimo duomenys

Tiriant sudaryto atkaitinimo modeliavimo algoritmo efektyvumą, atlikti išsamūs eksperimentiniai tyrimai su VRP uždavinio testiniais pavyzdžiais (duomenimis) iš viešosios elektroninės VRP testinių pavyzdžių bibliotekos CVRPLIB [UPP+17].

CVRPLIB duomenų bibliotekoje galima rasti įvairių uždavinių ir jų sprendinių VRP uždaviniui spręsti. Tyrinėtojai gali parsisiųsti duomenų bylas algoritmams testuoti ir savo rezultatus palyginti su geriausiais rastais rezultatais. Kiekvieną duomenų bylą sudaro duomenys ir jų aprašymas. Aprašomoji dalis susideda iš duomenų formato apibūdinimo ir komentarų.

Duomenų formato aprašymą sudaro tokie tyrimui reikalingi atributai:

- ”Name” – duomenų bylos pavadinimas;
- ”Type” – duomenų tipas. Tyrime bus naudojami simetrinio tipo VRP duomenys (t.y. lankas ij yra tokios pat vertės kaip ir lankas ji);
- ”Comment” – komentaras apie duomenis;
- ”Dimension” – miestų skaičius;
- ”Edge_Weight_Type” – nusako, kokiu formatu pateikiamos miestų koordinatės. Galimas GEO formatas (kai koordinatės pateikiamos kaip geografinė ilguma ir platuma) arba EUC_2D formatas (kai koordinatės pateikiamos paprastomis koordinatėmis ir skaičiuojamas euklidinis atstumas).

Kaip minėta, VRP tipo miestų duomenys pateikiami paprastomis koordinatėmis arba geografinėmis koordinatėmis: geografine ilguma ir platuma. Kai duomenys pateikiami paprastomis koordinatėmis, skaičiuojamas euklidinis atstumas. Savo darbe tyrimui naudosiu VRP testinius duomenis, kuriuose naudojamos tik paprastos koordinatės, ir bus skaičiuojamas euklidinis atstumas.

Yra kelios minėtųjų testinių duomenų rinkinių pasirinkimo priežastys:

1. Šiuo metu nepavyko rasti jokių etaloninių transporto maršrutų sudarymo uždavinių su dronais duomenų rinkinių, taigi turėjau rinktis iš esamų transporto maršrutų sudarymo uždavinių pristatymams sunkvežimiais duomenų rinkinių.
2. Pasirinkau šiuos vienuolika duomenų rinkinių norėdamas patikrinti taikyto algoritmo efektyvumą su skirtingu grafo mazgų skaičiumi.
3. Skirtingas grafo mazgų išdėstymas erdvėje.
4. Kiekviename pasirinktame duomenų rinkinyje yra žinomas optimalus prekių pristatymo sunkvežimiais sprendinys. Todėl norėčiau palyginti gautus prekių pristatymo sunkvežimiais ir dronais rezultatus bei pateikti išvadas ir rekomendacijas

2 lentelė. Testiniai duomenys

Nr.	Testinio pavyzdžio pavadinimas	Miestų/klientų skaičius	Optimalus automobilių skaičius	Optimali tikslo funkcijos reikšmė
1	E-n22-k4	21	4	375
2	E-n23-k3	22	3	569
3	E-n30-k3	29	3	534
4	E-n33-k4	32	4	835
5	E-n51-k5	50	5	521
6	E-n76-k7	75	7	682
7	E-n76-k8	75	8	735
8	E-n76-k10	75	10	830
9	E-n76-k14	75	14	1021
10	E-n101-k8	100	8	815
11	E-n101-k14	100	14	1067

4.2. Sukurto modeliuojamo atkaitinimo algoritmo tyrimas

Iš viso buvo eksperimentuota su vienuolika skirtingų testinių pavyzdžių.

Pirmiausia kaip kriterijus algoritmo efektyvumui įvertinti buvo pasirinktas santykinės gaunamų sprendinių kokybės rodiklis, tiksliau, gaunamų tikslo funkcijos reikšmių, t. y. maršrutų ilgių, vidutinis santykinis nuokrypis nuo tikėtinos optimalios tikslo funkcijos reikšmės (minimalaus galimo maršruto ilgio). Kuo šis nuokrypis mažesnis, tuo efektyvumas didesnis. Vidutinis santykinis nuokrypis (jį pažymėsime δ) apibrėžiamas pagal formulę :

$$\delta = \frac{\left(\bar{C}_{auto} - C_{opt}\right)}{C_{opt}} \times 100\%,$$

kur $\bar{C}_{auto}$ yra gautų tikslo funkcijos reikšmių (tik autotransporto maršrutų ilgių) vidurkis, kuris apskaičiuojamas, atlikus 50 algoritmo pakartotinių vykdymų, o C_{opt} yra optimali tikslo funkcijos reikšmė (šios reikšmės pateikiamos bibliotekoje CVRPLIB [UPP+17]).

Vienas iš pagrindinių eksperimentinių tyrimų tikslas — nustatyti geriausias sukurto algoritmo parametrų reikšmes. Tuo pačiu siekta išsiaiškinti, kurie būtent algoritmo valdymo parametrai turi didžiausią įtaką algoritmo efektyvumui (t.y. gaunamų sprendinių kokybei, įvertinamai kriterijumi δ). Optimalaus valdymo parametrų reikšmių rinkinio apskaičiavimas yra atskiras sudėtingas uždavinys. Atliekant šiuos eksperimentus pasinaudota supaprastinta parametrų tyrimo metodika. Metodikos esmė – suformuoti kiek įmanoma priimtinesnį (naudojamo efektyvumo kriterijaus atžvilgiu) parametrų reikšmių poaibį, vykdant nuoseklius savarankiškus eksperimentus su atskirais parametrais.

Reikėtų pažymėti, kad tyrimas buvo pradedamas, turint jau iš anksto sudarytą tam tikrą preliminarią pradinę parametrų reikšmių konfigūraciją. Pradinis parametrų reikšmių rinkinys

gali žymiai paveikti eksperimentuojant gaunamas reikšmes, todėl labai gerai, jeigu preliminarų parametrų reikšmių rinkinį sudaro optimizavimo srities specialistas ar kitas ekspertas, remdamasis kuriomis nors išankstinėmis teorinėmis prielaidomis.

Eksperimentuojant su vienu iš duotų parametrų, visų kitų rinkinio parametrų reikšmės buvo fiksuotos (nekintančios). Eksperimentuodami su atskiromis parametrų reikšmėmis, bandome eksperimentuoti su skirtingomis algoritmo modifikacijomis. Palyginę duotos modifikacijos rezultatus (sprendinių kokybę) su kitų modifikacijų rezultatais, galime vertinti tos modifikacijos tinkamumą ir apsispręsti, ar pasirinkto parametro reikšmė yra gera, ar ją reikėtų atmesti. Tokiu būdu galima arba atmesti duotą modifikaciją (parametro reikšmę) kaip neefektyvią, arba laikyti ją atskaitos tašku tolimesniems eksperimentams. Tokia „bandymų ir klaidų“ metodika paremtu eksperimentavimu išsiaiškinama geriausia modifikacija, t. y. randama tokia tiriamo parametro reikšmė, kuri įgalina gauti geriausius sprendinius – mažiausią vidutinį nuokrypį δ). Taip patikrinami visi algoritmo veikimą veikiantys parametrai. Atlikus tokius eksperimentus su visais parametrais, gaunamas daugiau ar mažiau patenkinamas parametrų reikšmių rinkinys.

Sukurto algoritmo pradinių parametrų reikšmių rinkinys pateiktas toliau:

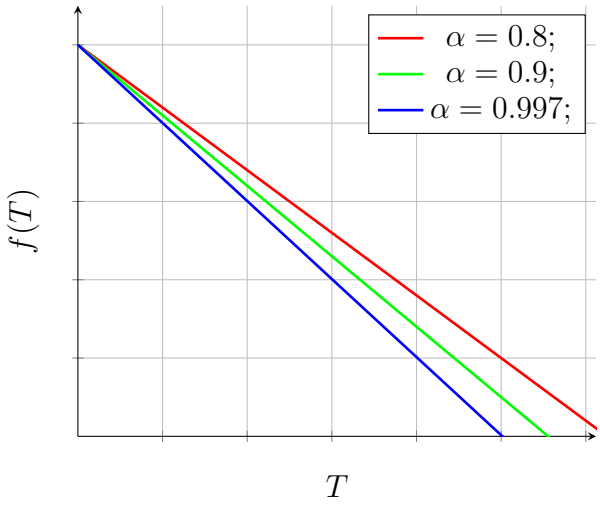
1. Pradinės temperatūros (T_0) reikšmės: 1000, 10000, 100000;
2. Aušinimo planas:
   - Tiesinis pagal [KGV83], temperatūra mažinama tiesiškai, žr. paveikslėlį Nr.8:

$$T_{i+1} = \alpha \times T_i$$

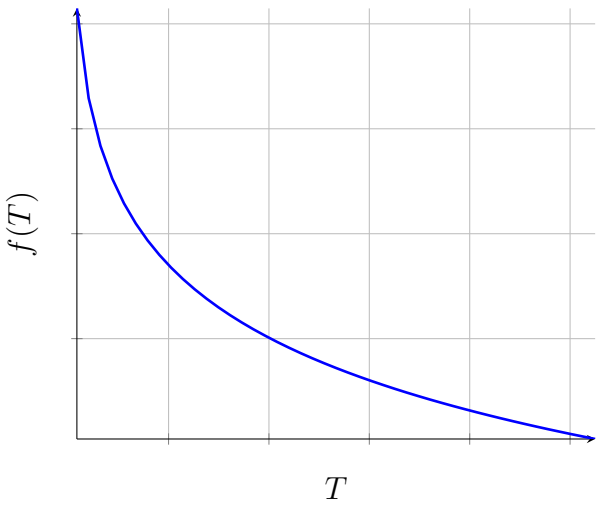
   - Logaritminis pagal [AK89], temperatūra mažinama logaritmiškai, žr. paveikslėlį Nr.9:

$$T_{i+1} = \frac{T_i}{1 + Ln(1+i)}$$

3. Itėracijų skaičiaus (i) reikšmės: 1000, 10000, 100000;



8 pav. Tiesinis aušinimo planas, $f(T) = \alpha \times T$



9 pav. Logaritminis aušinimo planas, $T_{i+1} = \frac{T_i}{1+Ln(1+i)}$

Pasirinkus pradinių parametrų reikšmių rinkinius, buvo eksperimentuojama su testiniais pavyzdžiais iš lentelės Nr.2. Eksperimentams su algoritmo modifikacijomis kaip kriterijus algoritmo efektyvumui įvertinti pasirinktas ne tik santykinės gaunamų sprendinių kokybės rodiklis, bet ir algoritmo vykdymo laikas t.

Geriausi rezultatai sprendžiant VRP uždavinį, taikant tiesinį aušinimo planą pateikti lentelėje Nr. 3.

3 lentelė. Geriausi rezultatai, taikant tiesinį aušinimo planą

Nr.	**Testinio pavyzdžio pavadinimas**	**Optimali tikslo funkcijos reikšmė,** C_{opt}	**Gautos tikslo funkcijos vidurkis,** $\bar{C}_{auto}$	**Geriausia gauta tikslo funkcijos reikšmė,** C_{best}	**Vidutinis santykinis nuokrypis,** δ, %	**Pradinės temperatūros reikšmė,** T_0	**Iteracijų skaičius,** i	α **koeficiento reikšmė**	**Vidutinis algoritmo vykdymo laikas,** s
1	E-n22-k4	375	430,86	389,07	14,89	1000	100000	0,997	20,479
2	E-n23-k3	569	635,48	571,4	11,68	10000	10000	0,8	8,3444
3	E-n30-k3	534	601,07	558,71	12,55	100000	10000	0,8	14,1373
4	E-n33-k4	835	928,34	883,57	11,17	1000	10000	0,8	15,6681
5	E-n51-k5	521	643,79	595,87	23,56	100000	100000	0,8	132,2988
6	E-n76-k7	682	834,16	773,85	22,31	100000	1000	0,8	233,5556
7	E-n76-k8	735	882,18	840,27	20,02	100000	10000	0,8	155,56
8	E-n76-k10	830	985,35	930,21	18,71	100000	100000	0,8	156,2153
9	E-n76-k14	1021	1167,75	1141	14,37	100000	1000	0,997	271,5531
10	E-n101-k8	815	1014,15	962,61	24,43	100000	10000	0,997	199,7876
11	E-n101-k14	1067	1288,06	1238,54	20,71	100000	100000	0,8	219,4961
	Vidurkis:	**725,82**	**855,56**	**807,74**	**17,68**				

Kaip matome geriausias rezultatas buvo rastas testiniam pavyzdžiui ”E-n33-K4”, kur δ nuokrypis yra mažiausias. Taip pat galima pastebėti tendenciją, kad didėjant miestų skaičiui, vidutinio santykinio nuokrypio vertė pradėjo siekti 20% ir daugiau, nors mažesniam miestų skaičiaus atvejų nuokrypio vertė siekia 11-14 %. Daugiausiai geriausių rezultatų algoritmas surado, kai pradinės temperatūros reikšmė siekė 100 000, iteracijų skaičius 10 000, o $\alpha = 0{,}8$.

Dėl temperatūros, reikia pažymėti kad tai yra vienas svarbių modeliuojamo atkaitinimo algoritmo parametrų, kuris nulemia atkaitinimo proceso charakterį, tuo pačiu ir gaunamus galutinius rezultatus. Eksperimentų rezultatai patvirtina hipotetines prielaidas apie pradinės temperatūros reikšmės įtaką gaunamų sprendinių kokybei. Kaip ir spėta, jei pradinė temperatūra per daug aukšta, tai konvergavimo į geresnės kokybės sprendinius procesas gali užsitęsti pernelyg ilgai.

Kita vertus, per daug žema pradinė temperatūra ne visada užtikrina pakankamai geros kokybės sprendinių radimo. Nors parinkti optimalią pradinės temperatūros reikšmę nebuvo paprasta, galima pastebėti, kad geresni rezultatai gauti temperatūrai esant aukščiausiai.

Toliau lentelėje Nr. 4 pateikiami geriausi rezultatai sprendžiant VRP uždavinį, taikant logaritminį aušinimo planą.

4 lentelė. Geriausi rezultatai, taikant logaritminį aušinimo planą

Nr.	**Testinio pavyzdžio pavadinimas**	**Optimali tikslo funkcijos reikšmė,** C_{opt}	**Gautos tikslo funkcijos vidurkis,** $\bar{C}_{auto}$	**Geriausia gauta tikslo funkcijos reikšmė,** C_{best}	**Vidutinis santykinis nuokrypis,** δ, %	**Pradinės temperatūros reikšmė,** T_0	**Iteracijų skaičius,** i	**Vidutinis algoritmo vykdymo laikas,** s
1	E-n22-k4	375	418,64	383,84	11,64	10000	10000	3,2491
2	E-n23-k3	569	649,05	599,17	14,07	10000	1000	0,1681
3	E-n30-k3	534	603,07	566,26	12,93	1000	10000	3,9118
4	E-n33-k4	835	944,46	884,54	13,11	1000	100000	39,8654
5	E-n51-k5	521	644,94	611,95	23,79	10000	100000	60,1134
6	E-n76-k7	682	848,44	803,64	24,40	1000	100000	95,8089
7	E-n76-k8	735	894,44	852,9	21,69	1000	100000	81,6222
8	E-n76-k10	830	1001,06	973,08	20,61	1000	100000	74,9687
9	E-n76-k14	1021	1165,41	1126,67	14,14	1000	100000	78,4096
10	E-n101-k8	815	1013,05	962,84	24,30	100000	100000	116,6947
11	E-n101-k14	1067	1288,23	1245,51	20,73	100000	100000	120,9306
	Vidurkis:	**725,82**	**860,98**	**819,13**	**18,31**			

Geriausias rezultatas, artimiausias C_{opt} buvo rastas testiniam pavyzdžiui "E-n22-k4", kur δ nuokrypis yra mažiausias. Taip pat pritaikius logaritminį aušinimo planą pastebima tendencija, kad, kaip ir tiesinio aušinimo plano atveju, didėjant uždavinio miestų skaičiui, algoritmo efektyvumas mažėja. Taip pat matyti, kad geresni rezultatai buvo randami, kai $T_0 = 1000$. Iš to išplaukia, kad logaritminio aušinimo atveju, norint rasti geresnį sprendinį, pradinė temperatūra gali būti ir nelabai aukšta.

Lentelėje Nr. 5 palyginami tiesinio ir logaritminio aušinimo planų rezultatai.

5 lentelė. Geriausi tiesinio ir logaritminio aušinimo planu rezultatai

Nr.	**Testinio pavyzdžio pavadinimas**	**Optimali tikslo funkcijos reikšmė,** C_{opt}	**Gautos tikslo funkcijos vidurkis,** $\bar{C}_{auto}$	**Geriausia gauta tikslo funkcijos reikšmė,** C_{best}	**Vidutinis santykinis nuokrypis,** δ, %	**Aušinimo planas**
1	E-n22-k4	375	418,64	383,84	11,64	logaritminis
2	E-n23-k3	569	635,48	571,4	11,68	tiesinis
3	E-n30-k3	534	601,07	558,71	12,56	tiesinis
4	E-n33-k4	835	928,34	883,57	11,18	tiesinis
5	E-n51-k5	521	643,79	595,87	23,57	tiesinis
6	E-n76-k7	682	834,16	773,85	22,31	tiesinis
7	E-n76-k8	735	882,18	840,27	20,02	tiesinis
8	E-n76-k10	830	985,35	930,21	18,72	tiesinis
9	E-n76-k14	1021	1165,41	1126,67	14,14	logaritminis
10	E-n101-k8	815	1013,05	962,84	24,30	logaritminis
11	E-n101-k14	1067	1288,06	1238,54	20,72	logaritminis
	Vidurkis:	**725,82**	**854,14**	**805,98**	**17,35**	

Apibendrinus gautus rezultatus matyti, kad logaritminio aušinimo plano rezultatai buvo geresni, kai uždavinio miestų skaičius buvo iki 22 miestų ir daugiau nei 76 miestai. Tiesinio

aušinimo plano atveju geresni rezultatai gauti, kai miestų skaičius buvo nuo 23 iki 76. Reikia paminėti, kad logaritminio ir tiesinio aušinimo plano atvejais, kai miestų skaičius siekė 76 ir daugiau, vidutinis santykinis nuokrypio skirtumas siekė 1%. Todėl galima daryti išvadą, kad taikant tiesinį aušinimo planą gauti geresni rezultatai.

Kita vertus, palyginus vidutinius algoritmo vykdymo laikus „t“ galima pastebėti, kad logaritminio aušinimo atveju algoritmo vykdymo laikas beveik 30–50% trumpesnis, todėl jeigu vykdymo laikas yra svarbus, geriau pasirinkti logaritminį aušinimo planą.

4.3. VRP-D sprendimas

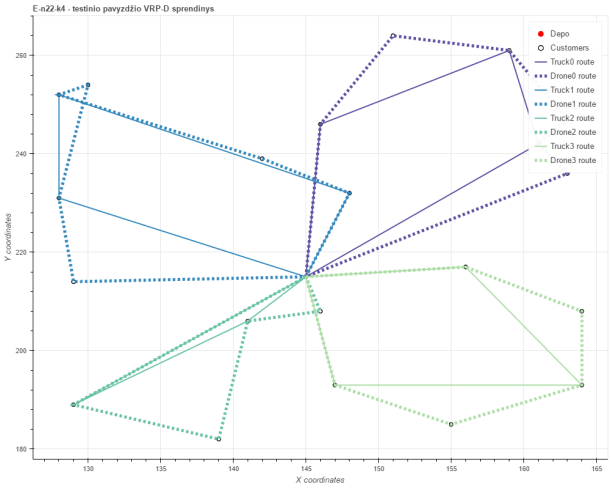
Po algoritmo tyrimo buvo atrinkti geriausi sukurto algoritmo parametrai, kurie bus panaudoti VRP-D uždaviniams spręsti. Kadangi VRP-D testinių duomenų nepavyko rasti, algoritmas bus testuojamas lentelėje Nr.2 VRP pateiktiems uždaviniams spręsti. Tik krovinių pristatymas bus vykdomas ne tik automobiliais, bet automobiliais kartu su dronais, t. y. bus sprendžiamas ne VRP, bet VRP-D uždavinys. Sukurtas algoritmas bus patobulintas VRP-D uždaviniams spręsti. Grafiškai sprendinio konstravimas pavaizduotas paveikslėlyje Nr.7.

Dabar po algoritmo tyrimo buvo atrinkti geriausi sukurto algoritmo parametrai, kurie bus panaudoti spręsti VRP-D uždaviniams. Kadangi VRP-D testinių duomenų nepavyko surasti, todėl algoritmas bus testuojamas spręsti lentelėje Nr. 2 VRP uždavinius. Tik pristatymas bus vykdomas ne tik automobiliais, bet automobiliais kartu su dronais, t.y. bus sprendžiamas ne VRP, bet VRP-D uždavinys. Pats sukurtas algoritmas bus patobulintas, VRP-D uždaviniams spręsti. Grafiškai sprendinio konstravimas yra pavaizduotas paveikslėlyje Nr. 7. Lentelėje Nr. 6 pateikiami geriausi gauti VRP-D rezultatai.

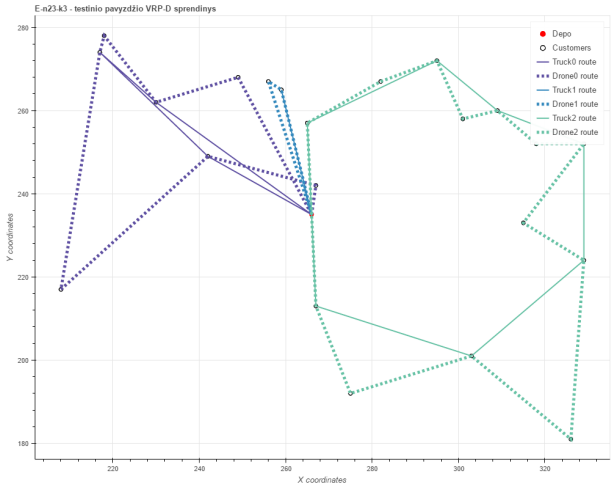
6 lentelė. VRP-D geriausi rezultatai

Nr.	**Testinio pavyzdžio pavadinimas**	**VRP optimali tikslo funkcijos reikšmė,** C_{VRP}	**VRP-D drono geriausia gauta tikslo funkcijos reikšmė,** C_{drone}	**VRP-D automobilio geriausia gauta tikslo funkcijos reikšmė,** C_{auto}	**VRP-D (bendra) geriausia gauta tikslo funkcijos reikšmė,** C_{VRP-D}	**Aušinimo planas**
1	E-n22-k4	375	142,65	328,00	470,66	logaritminis
2	E-n23-k3	569	322,08	405,67	727,75	tiesinis
3	E-n30-k3	534	131,98	512,84	644,82	tiesinis
4	E-n33-k4	835	245,97	739,99	985,96	tiesinis
5	E-n51-k5	521	299,99	458,18	758,18	tiesinis
6	E-n76-k7	682	378,59	597,48	976,07	tiesinis
7	E-n76-k8	735	366,27	659,23	1025,50	tiesinis
8	E-n76-k10	830	356,94	700,96	1057,89	tiesinis
9	E-n76-k14	1021	318,04	879,58	1197,62	tiesinis
10	E-n101-k8	815	465,17	799,03	1264,20	tiesinis
11	E-n101-k14	1067	422,95	1031,56	1454,51	tiesinis
	Vidurkis:	**725,82**	**313,69**	**646,59**	**960,29**	

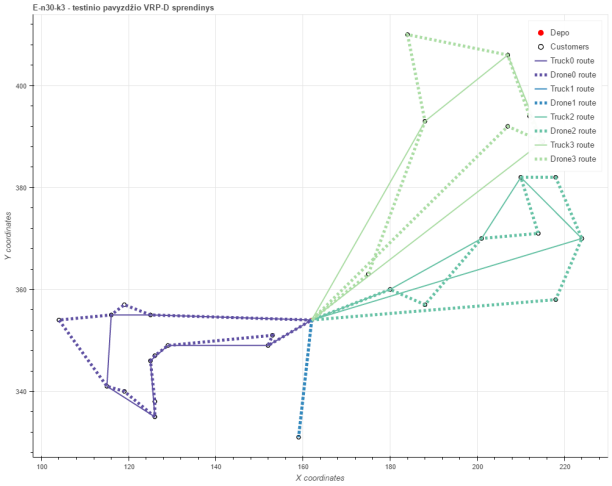
Pirmų keturių testinių pavyzdžių VRP-D sprendinių maršrutai pavaizduoti 15-18 paveikslėliuose.



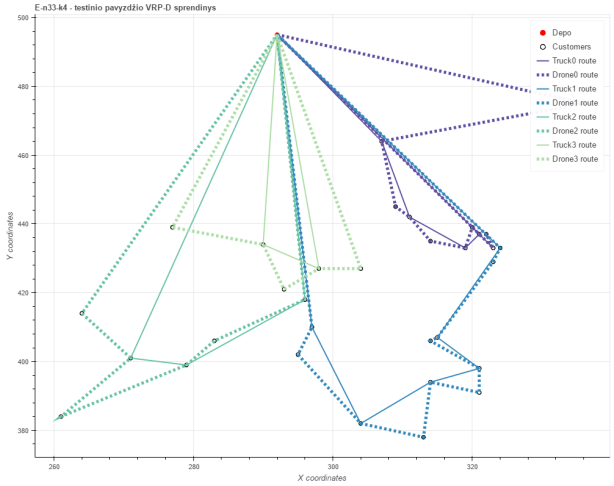
10 pav. E-n22-k4 VRP-D sprendinys



11 pav. E-n23-k3 VRP-D sprendinys



12 pav. E-n30-k3 VRP-D sprendinys



13 pav. E-n33-k4 VRP-D sprendinys

7 lentelė. VRP-D C_{auto} geriausių gautu rezultatų palyginimas su C_{VRP}

Nr.	Testinio pavyzdžio pavadinimas	VRP optimali tikslo funkcijos reikšmė, C_{VRP}	VRP-D automobilio geriausia gauta tikslo funkcijos reikšmė	Santykinė paklaida, %	Absoliuti paklaida
1	E-n22-k4	375	328,00	12,53	47,00
2	E-n23-k3	569	405,67	28,70	163,33
3	E-n30-k3	534	512,84	3,96	21,16
4	E-n33-k4	835	739,99	11,38	95,01
5	E-n51-k5	521	458,18	12,06	62,82
6	E-n76-k7	682	597,48	12,39	84,52
7	E-n76-k8	735	659,23	10,31	75,77
8	E-n76-k10	830	700,96	15,55	129,04
9	E-n76-k14	1021	879,58	13,85	141,42
10	E-n101-k8	815	799,03	1,96	15,97
11	E-n101-k14	1067	1031,56	3,32	35,44
	Vidurkis:	**725,82**	**646,59**		

Iš lentelės Nr. 6rezultatų matyti, kad VRP-D atveju geresnių rezultatų pasiekta naudojant tiesinį aušinimo planą. Lentelėje Nr. 7 lyginamos C_{VRP} ir VRP-D C_{auto} reikšmės. Matome, kad VRP-D atveju sukurtas algoritmo efektyvumas mažėja didėjant miestų skaičiui. Testinių duomenų E-n101-k8 ir E-n101-k14 atvejais santykinė paklaida siekia vos 1,96–3,32%, visais kitais atvejais automobilio nuvažiuotas kelias (tikslo funkcijos reikšmės) VRP-D beveik apie 10–15% trumpesnis už optimalias VRP uždavinio tikslo funkcijos reikšmes.

4.4. VRP ir VRP-D kaštai

Pagal duomenis, pateiktus įvairiuose šaltiniuose [Kha16], [Kee15],[Wor17], [ABo16], galima apskaičiuoti preliminarius kaštus, kai prekės pristatomos automobiliais ir dronais. Apibendrinęs pirmiau minėtų straipsnių informaciją, toliau pateiksiu transportavimo vieną mylią (tai yra 1,609344 km) kaštus JAV doleriais, krovinius gabenant automobiliu ir bepiločiu orlaiviu:

1. Dyzelinis automobilis - apie 1,00 USD už 1 mylią;
2. Elektrinis automobilis - apie 0,45 USD už 1 mylią;
3. Bepilotis orlaivis - apie 0,03 USD už 1 mylią [Wor17].

Į šiuos kaštus įtrauktos tik degalų sąnaudos, bepiločio orlaivio ir elektrinio automobilio atveju apskaičiuotos elektros sąnaudos.

Kaip matyti, pristatymo bepiločiais orlaiviais kaštai siekia tik 3% pristatymo dyzeliniais automobiliais kaštų, todėl tai yra labai perspektyvus, ekonomiškas ir ekologiškas prekių pristatymo būdas.

Toliau pasistengsiu įvertinti, kiek sukurtas algoritmas būtų naudingas pritaikius pirmiau minėtą informaciją ir kiek leistų sutaupyti VRP-D atveju.

8 lentelė. VRP ir VRP-D kaštai

Nr.	**Testinio pavyzdžio pavadinimas**	**VRP transportavimo išlaidos, USD**	**VRP-D bendros (drono ir automobilio) transportavimo išlaidos, USD**	**Santykinė paklaida, %**
1	E-n22-k4	375	330,86	11,77
2	E-n23-k3	569	412,11	27,57
3	E-n30-k3	534	515,47	3,47
4	E-n33-k4	835	744,90	10,79
5	E-n51-k5	521	464,18	10,91
6	E-n76-k7	682	605,05	11,28
7	E-n76-k8	735	666,55	9,31
8	E-n76-k10	830	708,10	14,69
9	E-n76-k14	1021	885,94	13,23
10	E-n101-k8	815	808,34	0,82
11	E-n101-k14	1067	1040,02	2,53
	Vidurkis:	**725,82**	**652,87**	**10,05**

Pagal lentelės Nr. 8 duomenis galime daryti išvadą, kad drono ir automobilio derinys suvartoja mažiau degalų vienai myliai. Įvertinus vidutinę santykinę paklaidą galima daryti išvadą, kad, pritaikius sukurtą algoritmą, kai prekėms pristatyti naudojamas drono ir automobilio derinys, vidutiniškai galima būtų sutaupyti apie 10% degalų.

Rezultatai ir išvados

Pagrindinė šio magistro darbo išvada: taikant modeliuojamo atkaitinimo algoritmą galima sukurti efektyvų algoritmą VRP ir naujo tipo uždaviniams, tokiems kaip VRP-D uždaviniai, spręsti.

Šiame darbe išnagrinėti VRP uždaviniai, apžvelgta jų klasifikacija pagal skirtingus parametrus ir charakteristikas. Taip pat išnagrinėti VRP uždavinių sprendimo būdai, tarp kurių yra tikslieji, euristiniai ir metaeuristiniai metodai, skirti VRP uždaviniams spręsti.

Aprašyta problema ir suformuluotas maršrutų sudarymo uždavinys su bepiločiais orlaiviais (VRP-D).

VRP ir VRP-D uždaviniams spręsti sukurtas modeliuojamo atkaitinimo algoritmas. Sukūrus modeliuojamo atkaitinimo algoritmą VRP ir VRP-D uždaviniams spręsti, buvo nustatyta:

1. Bene didžiausią įtaką gaunamų rezultatų kokybei turi atkaitinimo proceso pradinės temperatūros parametras. Panaši ir iteracijų skaičiaus įtaka. Svarbų vaidmenį taip pat vaidina pasirinktas aušinimo planas, kuris irgi turi įtakos galutiniams rezultatams. Kai kuriais atvejais koreliacija tarp atskirų parametrų reikšmių ir gaunamų sprendinių kokybės yra visiškai akivaizdi; vis tik įžvelgti aiškią bendrą tendenciją ir dėsningumą visais atvejais nėra paprasta.

2. Eksperimentuojant su sukurtu modeliuojamo atkaitinimo algoritmu išbandyti du dviejų tipų aušinimo planai, tiesinis ir logaritminis. Pritaikius tiesinį aušinimo planą, rezultatai buvo geresni už logaritminį, nors ir ne visais atvejais. Kas dėl logaritminio aušinimo plano, verta paminėti, kad algoritmo vykdymo laikas buvo beveik dvigubai trumpesnis už tiesinį aušinimo planą.

3. Pastebėta, kad didinant VRP apimtis sukurtas modeliuojamo atkaitinimo algoritmas keičia savo veikimo tendencijas. Esant didesniam VRP miestų skaičiui algoritmo efektyvumas mažėjo.

4. Realizuotas algoritmas bei jo paveiktų parametrų analizė leido pasiekti daug žadančių rezultatų šiame darbe panaudotiems VRP testiniams pavyzdžiams. Nepavyko pritaikius šį algoritmą rasti optimalių sprendinių su pasirinktais parametrais.

Literatūra

[ABo16] A.Boyle. One companys vision for electric trucks, delivery drones — and a hybrid flying car. `https://www.geekwire.com/2016/workhorse-electric-trucks-drones-flying-cars/`, 2016. žiūrėta 2017-01-09.

[ABS16] N. Agatz, P. Bouman ir M. Schmidt. Optimization approaches for the traveling salesman problem with drone, 2016.

[AGM14] P. Anbuudayasankar, K. Ganesh, and S. Mohapatra. *Models for Practical Routing Problems in Logistics*. Springer International Publishing, 2014. 13 p.

[AK89] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, Inc., 1989.

[BEG+87] G. Brown, C. Ellis, G. Graves ir D. Ronen. Real-time, wide area dispatch of mobil tank trucks. *Interfaces*, 17:107–120, 1987.

[Bro11] J. Brownlee. *Clever Algorithms: Nature-inspired Programming Recipes*. Lulu.com, 2011.

[BT94] R. Battiti ir G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6:126–140, 1994.

[CCL+87] C.Yano, T. Chan, L.Richter, T.Cutler, K. Murty ir D. McGettigan. Vehicle routing at quality stores. *Interfaces*, 17:52–63, 1987.

[CGW88] D. Casco, Bl. Golden, and E. Wasil. Vehicle routing with backhauls: models, algorithms, and case studies. *Vehicle routing: methods and studies.*:127–147, 1988.

[CS93] J. Chakrapani ir J. Skorin-Kapov. Connection machine implementation of a tabu search algorithm for the traveling salesman problem. *Journal of Computing and Information Technology*, 1:29–36, 1993.

[DS03] M. Dorigo ir T. Stutzle. *The ant colony optimization metaheuristic: Algorithms, applications, and advances*. Springer, 2003, p.p. 250–285.

[FAJ80] M. Fisher, A.Greenfield ir R. Jaikumar. *VERGIN: A decision support system for vehicle scheduling*. Division of Research, Graduate School of Business Administration, Harvard University, 1980.

[Fel07] G. Felinskas. *EURISTINIŲ METODŲ TYRIMAS IR TAIKYMAS RIBOTŲ IŠTEKLIŲ TVARKARAŠČIAMS OPTIMIZUOTI*. Disertacija, VYTAUTO DIDŽIOJO UNIVERSITETAS, 2007.

[GNN07] K. Ganesh, A.S. Nallathambi ir T.T. Narendran. Variants, solution approaches and applications for vehicle routing problems in supply chain: agile framework and comprehensive review. *International Journal of Agile Systems and Management (IJASM)*, 2:50–75, 2007.

[Her14] A. Hern. Dhl launches first commercial drone 'parcelcopter' delivery service. https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first-commercial-drone-delivery-service, 2014. žiūrėta 2016-09-15.

[Hol92] J. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[JG15] J.Nicas ir G.Bensinger. Delivery drones hit bumps on path to doorstep. http://www.wsj.com/articles/technical-hurdles-delay-drone-deliveries-1426867441, 2015. žiūrėta 2016-09-15.

[Kee15] T. Keeney. How can amazon charge 1usd for drone delivery. https://ark-invest.com/research/drone-delivery-amazon, 2015. žiūrėta 2017-01-09.

[KGV83] S. Kirkpatrick, C.D. Gelatt ir M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[Kha16] A. Kharpal. This firm beat amazon to drone deliveries by launching it from the roof of a truck. http://www.cnbc.com/2016/08/18/this-firm-beat-amazon-to-drone-deliveries-by-launching-it-from-the-roof-of-a-truck.html, 2016. žiūrėta 2017-01-09.

[LK73] S. Lin ir B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[LLM89] G. Laporte, F. Louveaux ir H. Mercure. Models and exact solutions for a class of stochastic location-routing problems. *European Journal of Operational Research*, 39:71–78, 1989.

[MD14] E. Morganti and L. Dablanc. *Non-technological Innovations for Sustainable Transport: Four Transport Case Studies*. Springer International Publishing, 2014. pp. 27-45.

[Mer99] S. Merthens. Insertion heuristics. http://www-e.uni-magdeburg.de/mertens/TSP/node2.html, 1999. žiūrėta 2017-01-07.

[Min89] H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23:377–386, 1989.

[MRR+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller ir E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[Pop13] B. Popper. Ups researching delivery drones that could compete with amazon's prime air. http://www.theverge.com/2013/12/3/5169878/ups-is-researching-its-own-delivery-drones-to-compete-with-amazons, 2013. žiūrėta 2016-09-15.

[PS82] Ch. Papadimitriou and K. Steiglitz. *Combinatorial optimization*. Englewood Cliffs, N.J.: Prentice Hall, 1982.

[RBL02] J. Renaud, F. Boctor ir G. Laporte. Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers and Operations Research*, 29:1129–1141, 2002.

[RC06] R. Russell ir W. Chiang. Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, 169:606–622, 2006.

[Tai91] E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[UPP+17] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal ir A. Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257:845–858, 2017.

[Woh14] M. Wohlsen. The next big thing you missed: amazon's delivery drones could work. they just need trucks. `https://www.wired.com/2014/06/the-next-big-thing-you-missed-delivery-drones-launched-from-trucks-are-the-future-of-shipping`, 2014. žiūrėta 2016-09-15.

[Wor17] Workhorse Group Inc. Horsefly - autonomous drone delivery system. `http://workhorse.com/aerospace`, 2017. žiūrėta 2017-01-09.

Santrumpos

VRP (angl. Vehicle Routing Problem) - transporto maršrutų sudarymo uždavinys.

VRP-D (angl. Vehicle Routing Problem with Drones) - transporto maršrutų sudarymo uždavinys su bepiločiais orlaiviais.

CVRP (angl., capacitated VRP (CVRP)) - transporto maršrutų sudarymo uždavinys su talpos apribojimais.

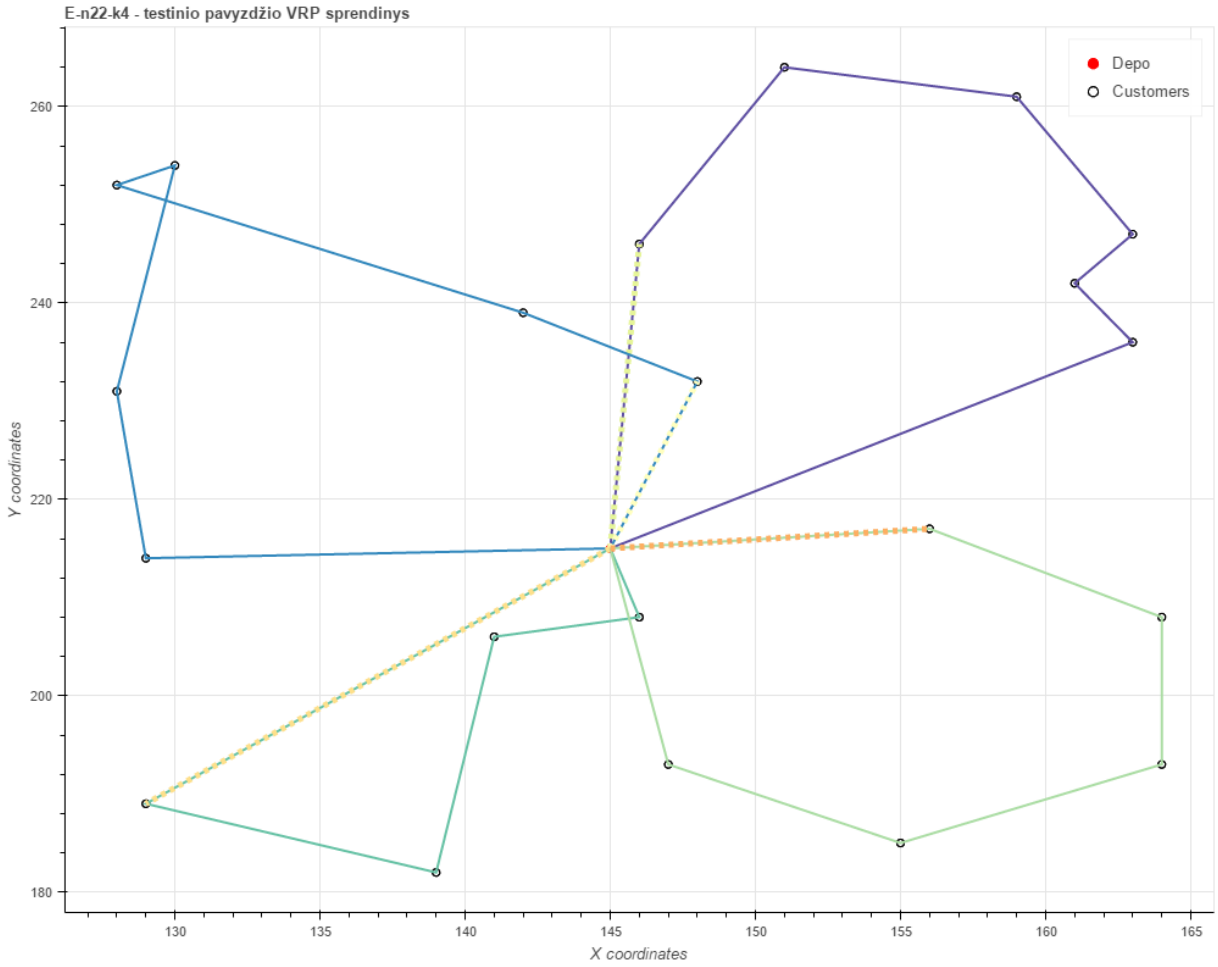
VRPTW (angl., VRP with time windows (VRPTW) - transporto maršrutų sudarymo uždavinys su laiko langais.

MDVRP (angl., VRP with multiple depots (MDVRP) - transporto maršrutų sudarymo uždavinys su keliais krovinių paskirstymo punktais, iš kurių klientams gali būti pristatytos prekės.

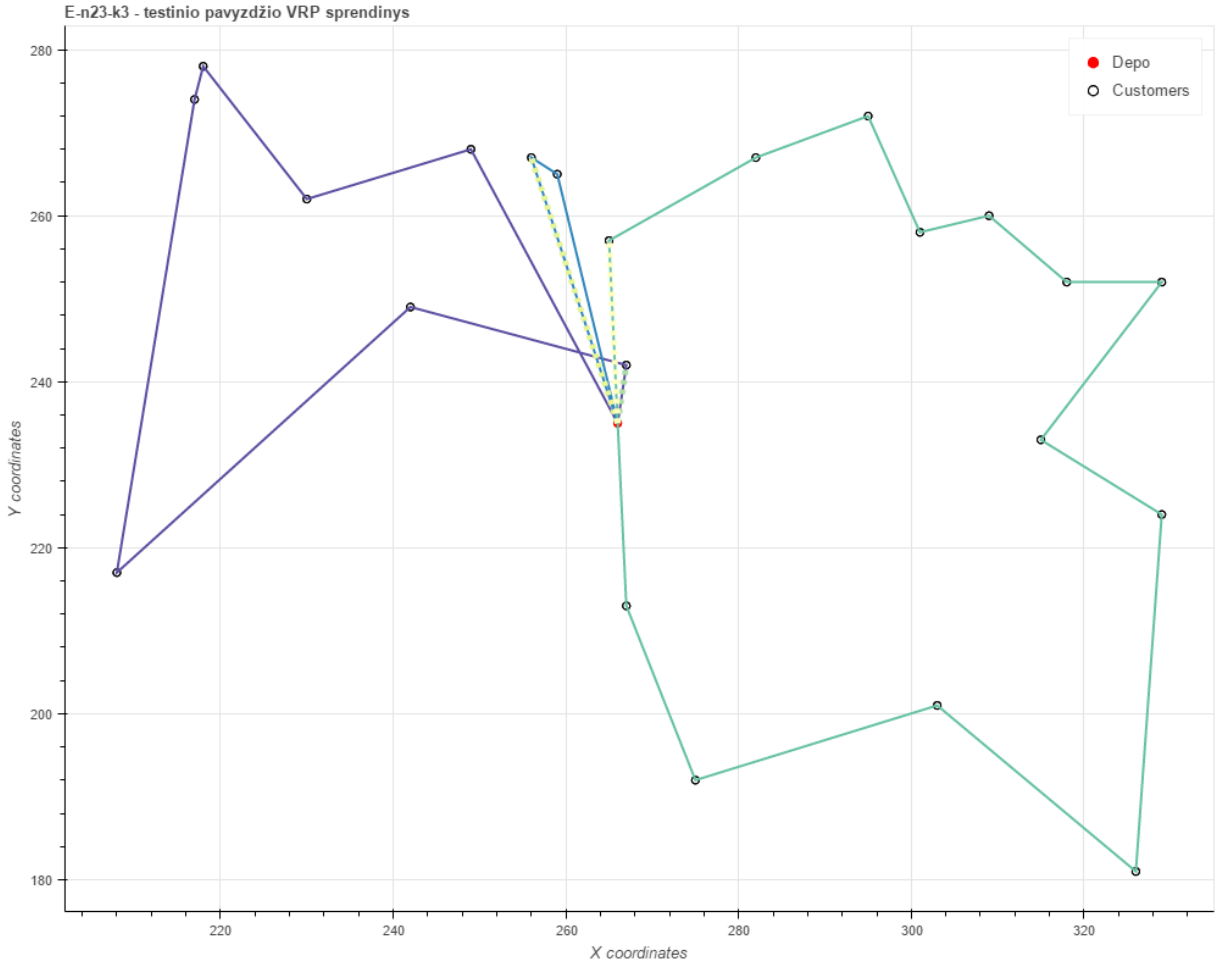
VRPPD (angl., VRP with pick-up and delivery (VRPPD) - transporto maršrutų sudarymo uždavinys su surinkimais ir pristatymais, kur yra apibrėžiamos sąlygos surinkti prekes iš vienų vietų bei pristatyti į kitas vietas

TSP (angl. Travelling Salesman Problem) - keliaujančio pirklio uždavinys

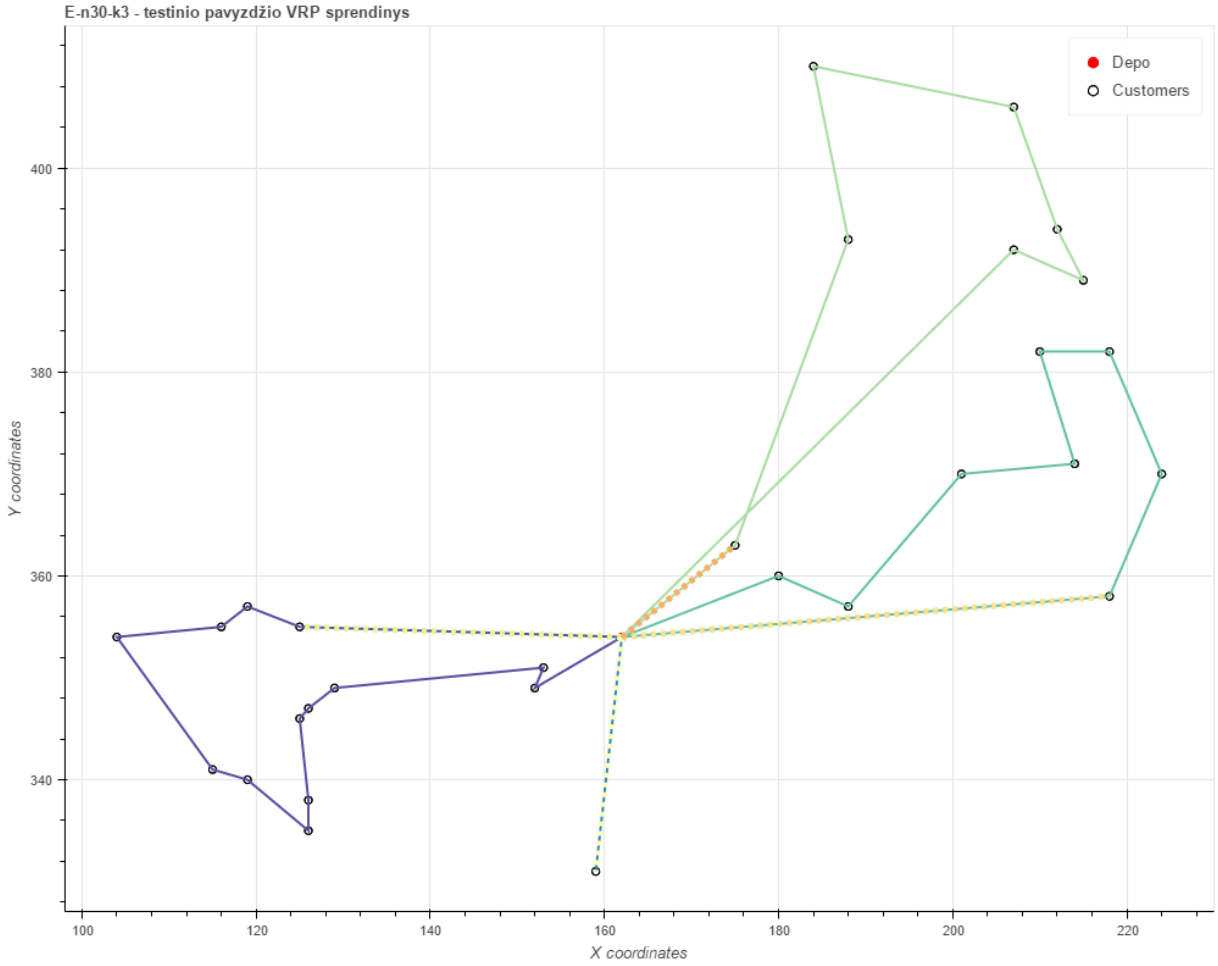
Priedas Nr. 1
VRP geriausių sprendinių maršrutai



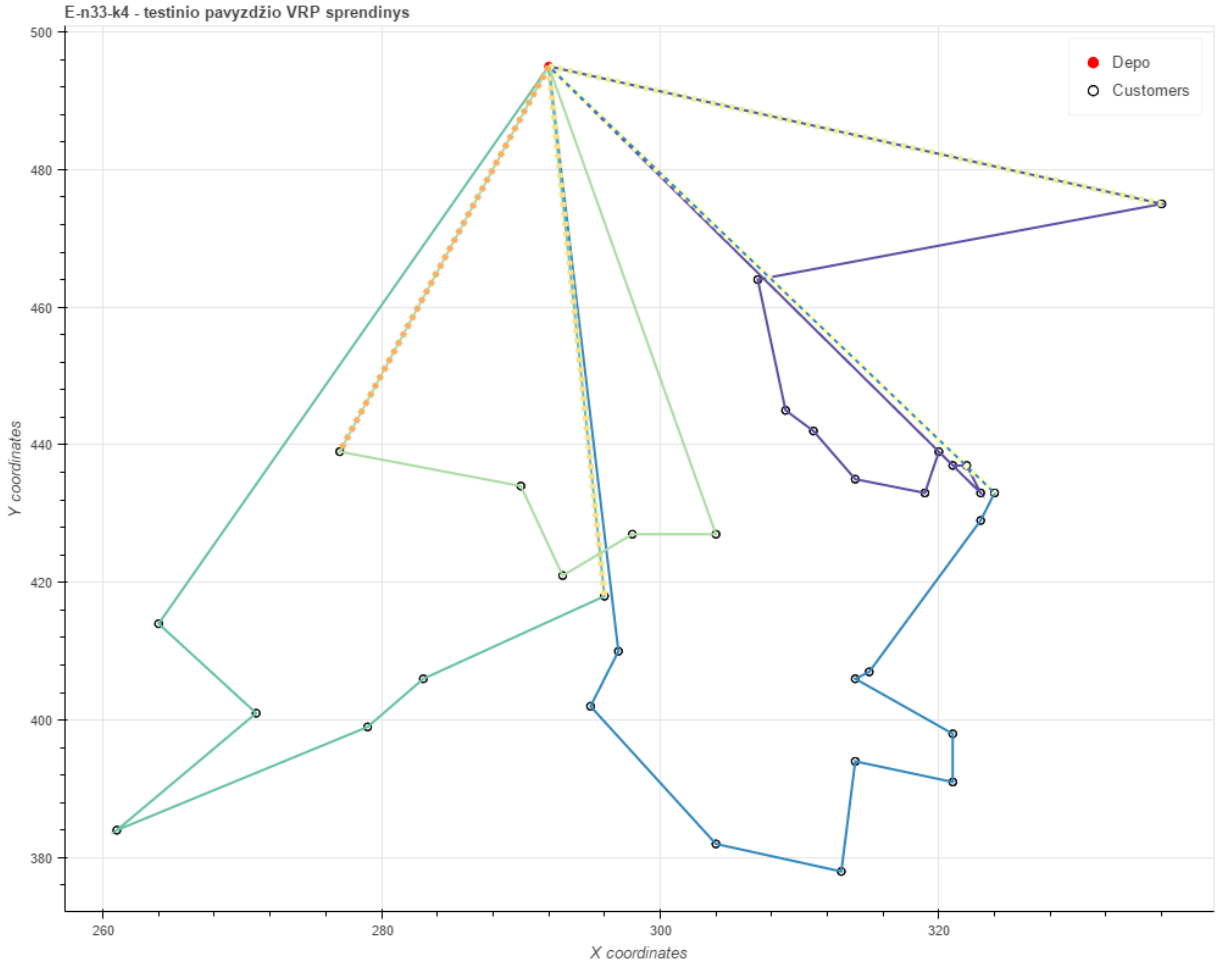
14 pav


E-n23-k3 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
280
260
240
220
200
180
220
240
260
280
300
320

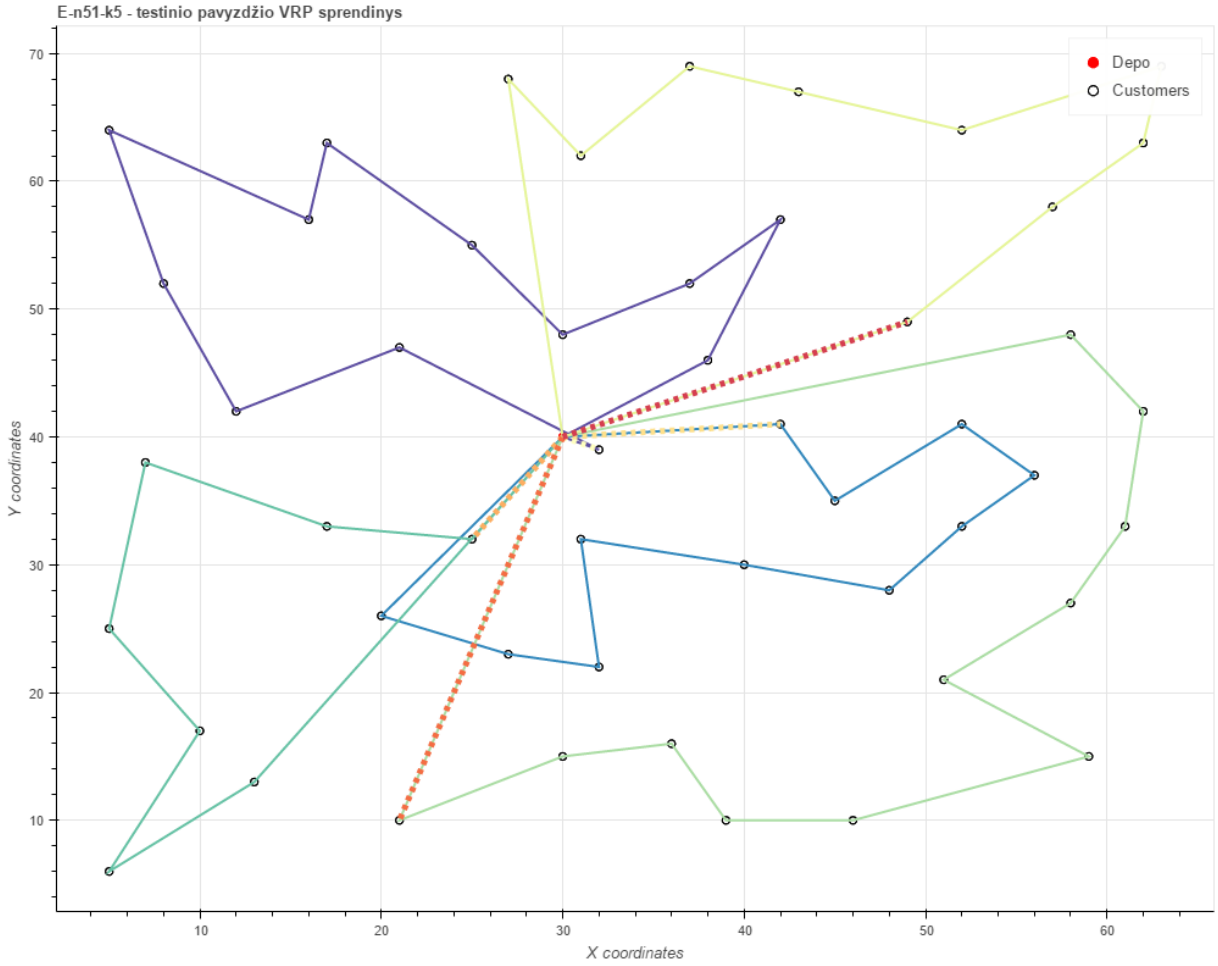
15 pav


E-n30-k3 - testinio pavyzdžio VRP sprendinys
Depo
Customers
400
380
360
340
Y coordinates
100
120
140
160
180
200
220
X coordinates

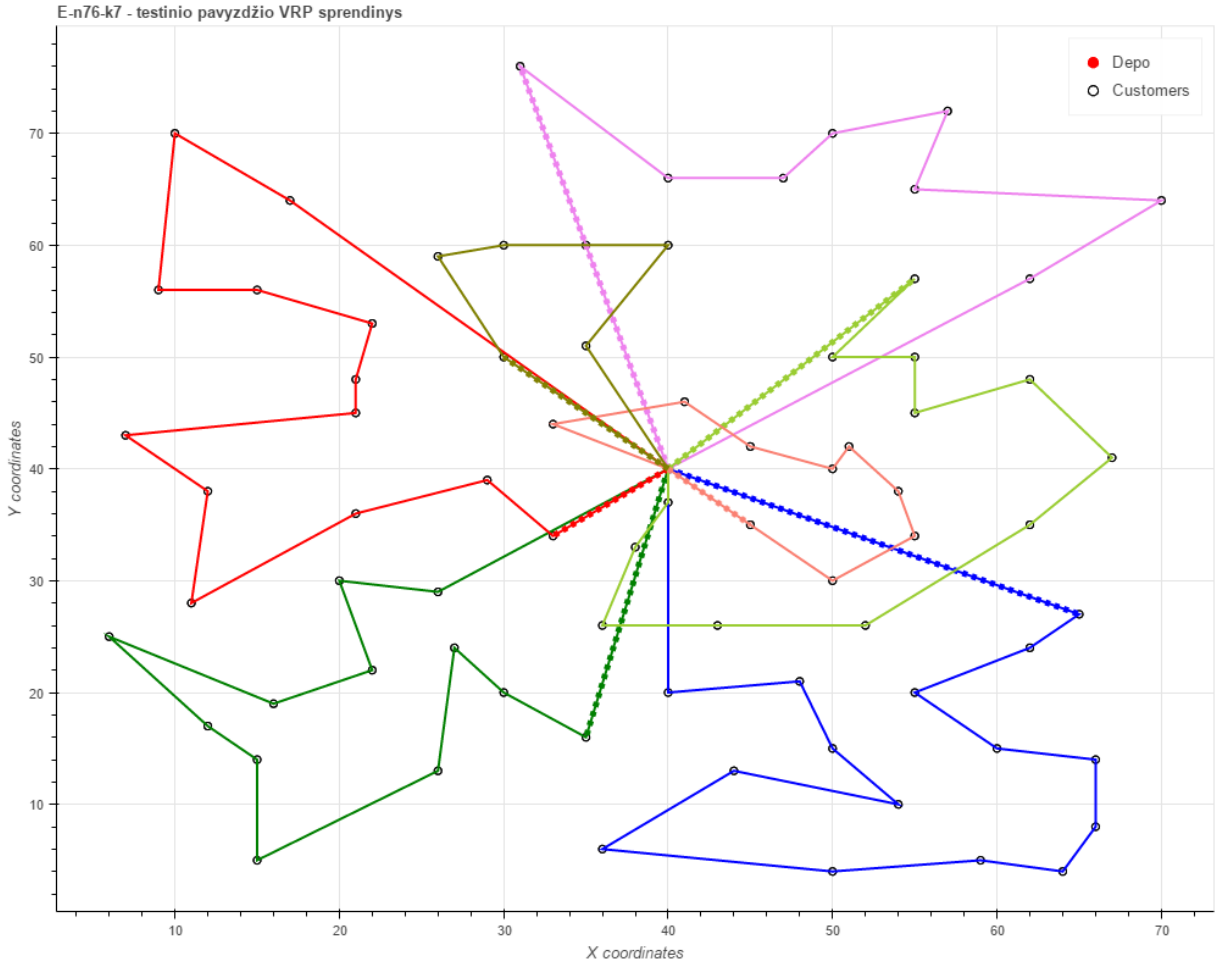
16 pav


E-n33-k4 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
500
480
460
440
420
400
380
260
280
300
320

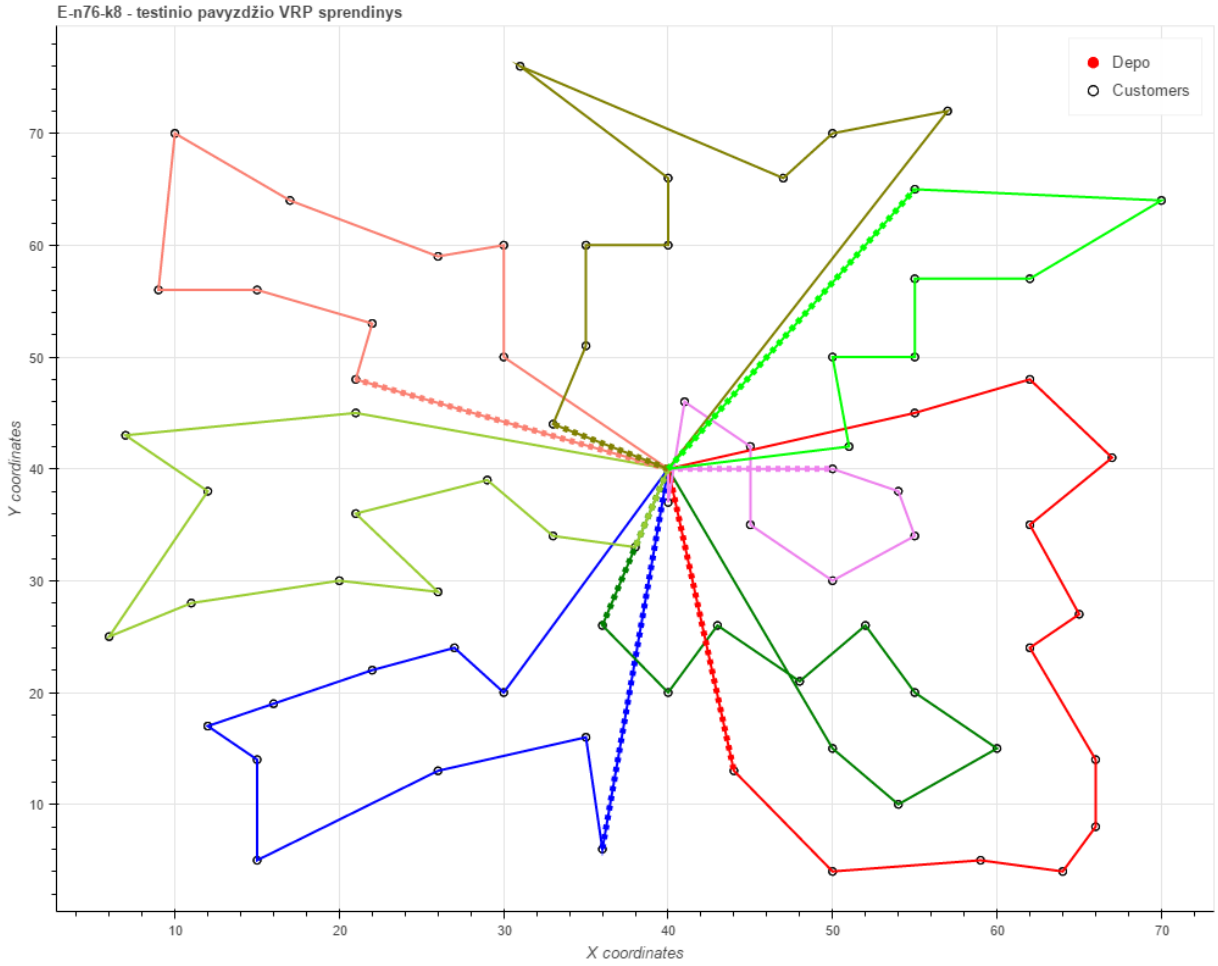
17 pav


E-n51-k5 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
70
60
50
40
30
20
10
10
20
30
40
50
60

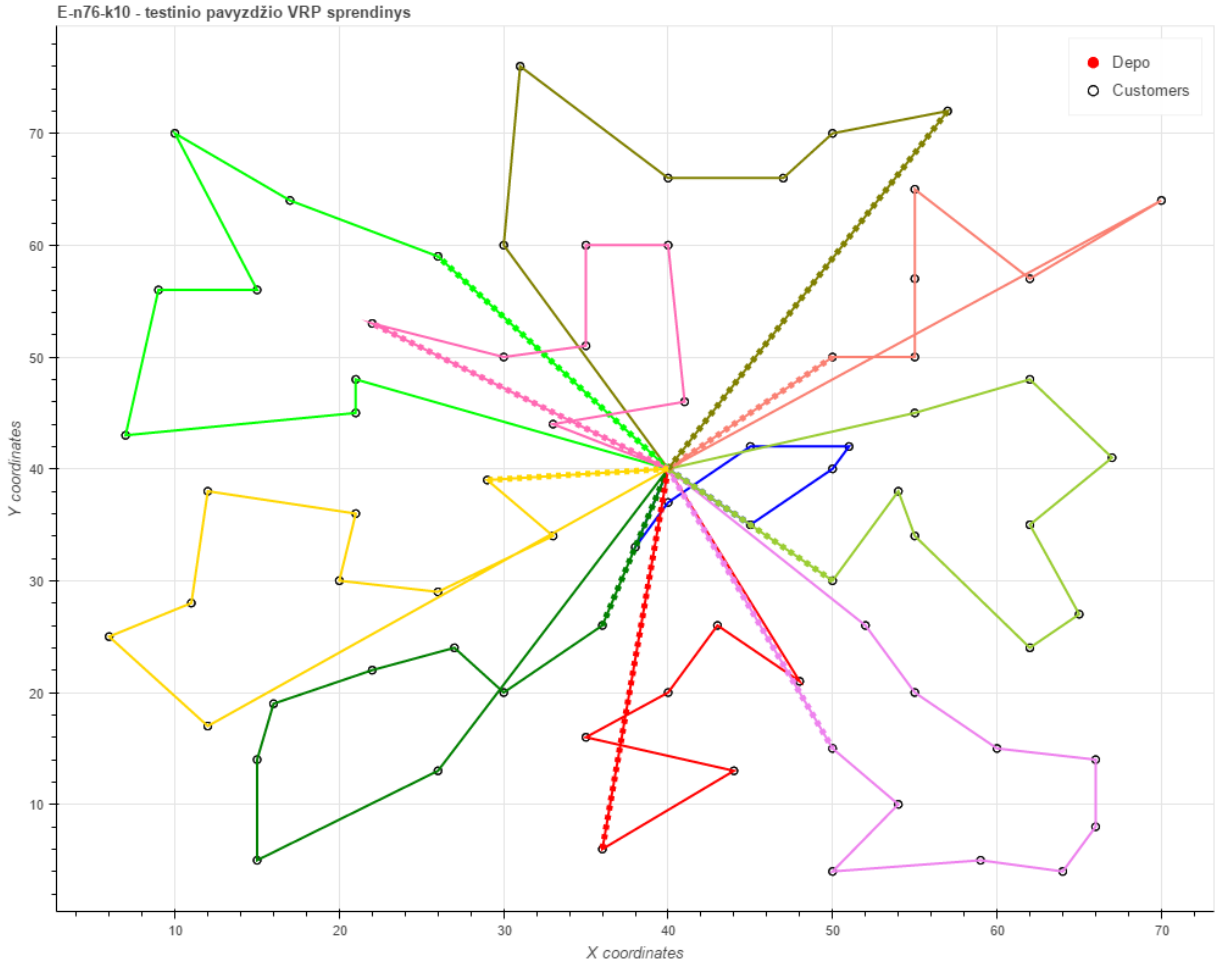
18 pav


E-n76-k7 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
10
20
30
40
50
60
70

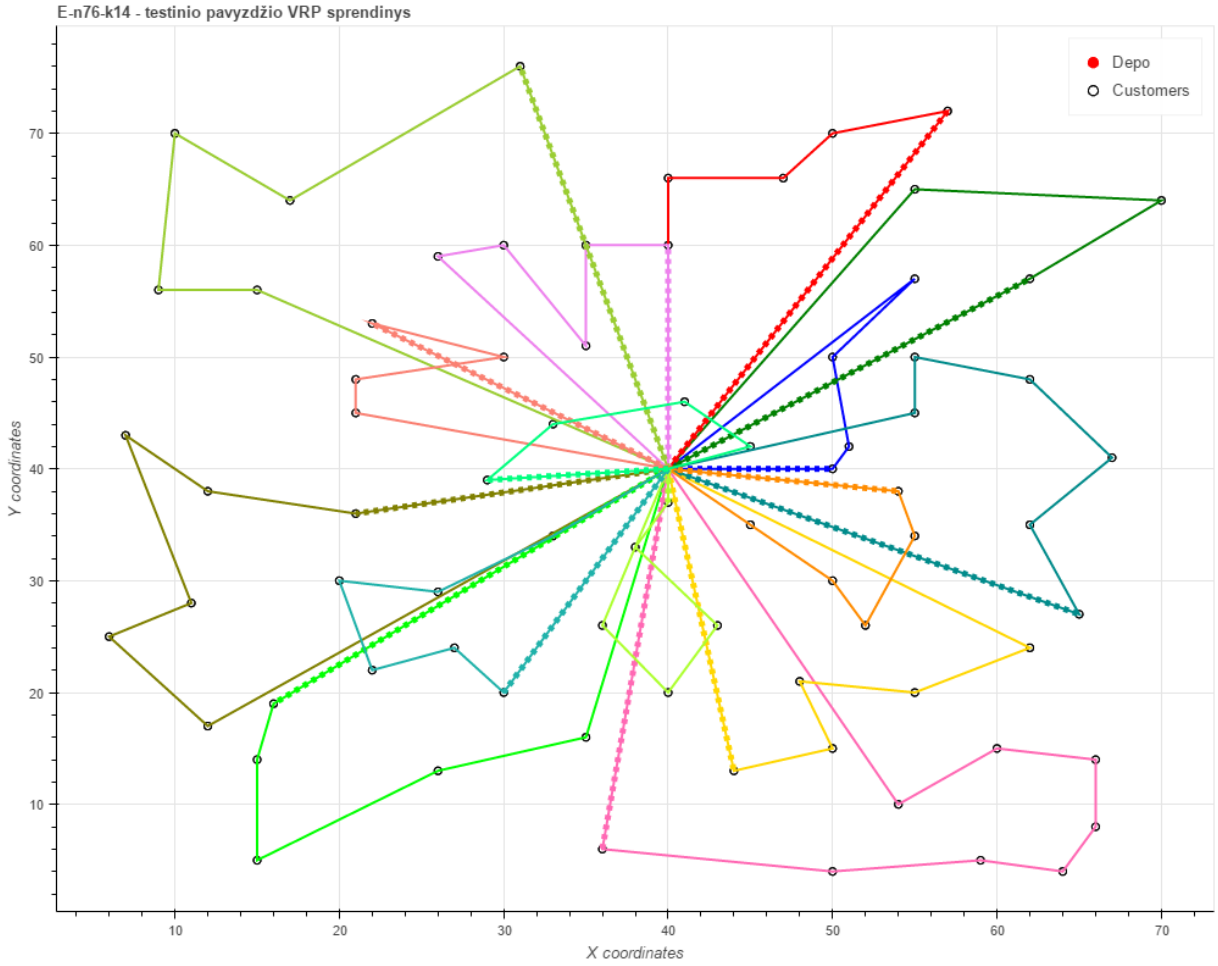
19 pav


E-n76-k8 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
10
20
30
40
50
60
70

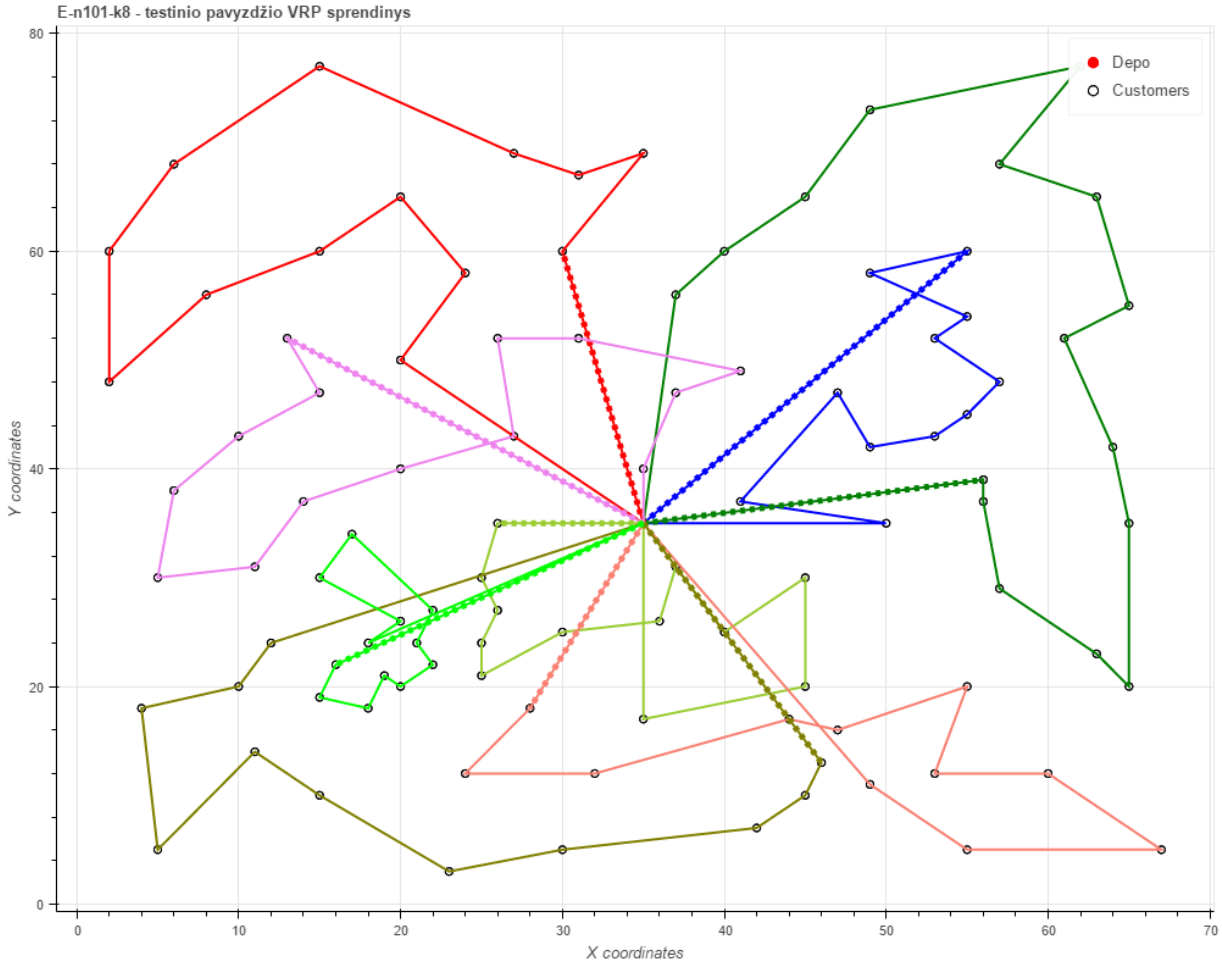
20 pav


E-n76-k10 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
10
20
30
40
50
60
70

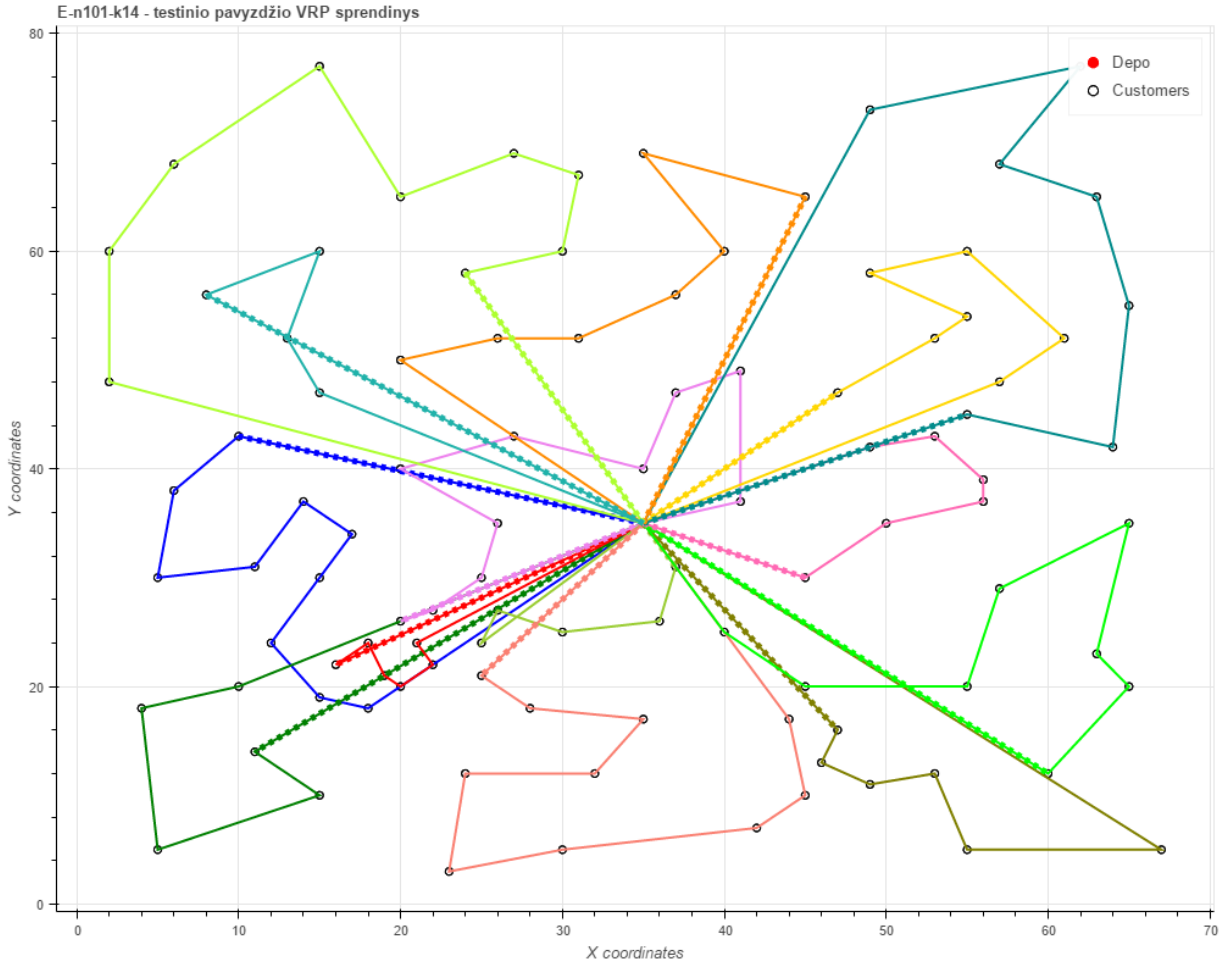
21 pav


E-n76-k14 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
10
20
30
40
50
60
70

22 pav


E-n101-k8 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
0
20
40
60
80
0
10
20
30
40
50
60
70

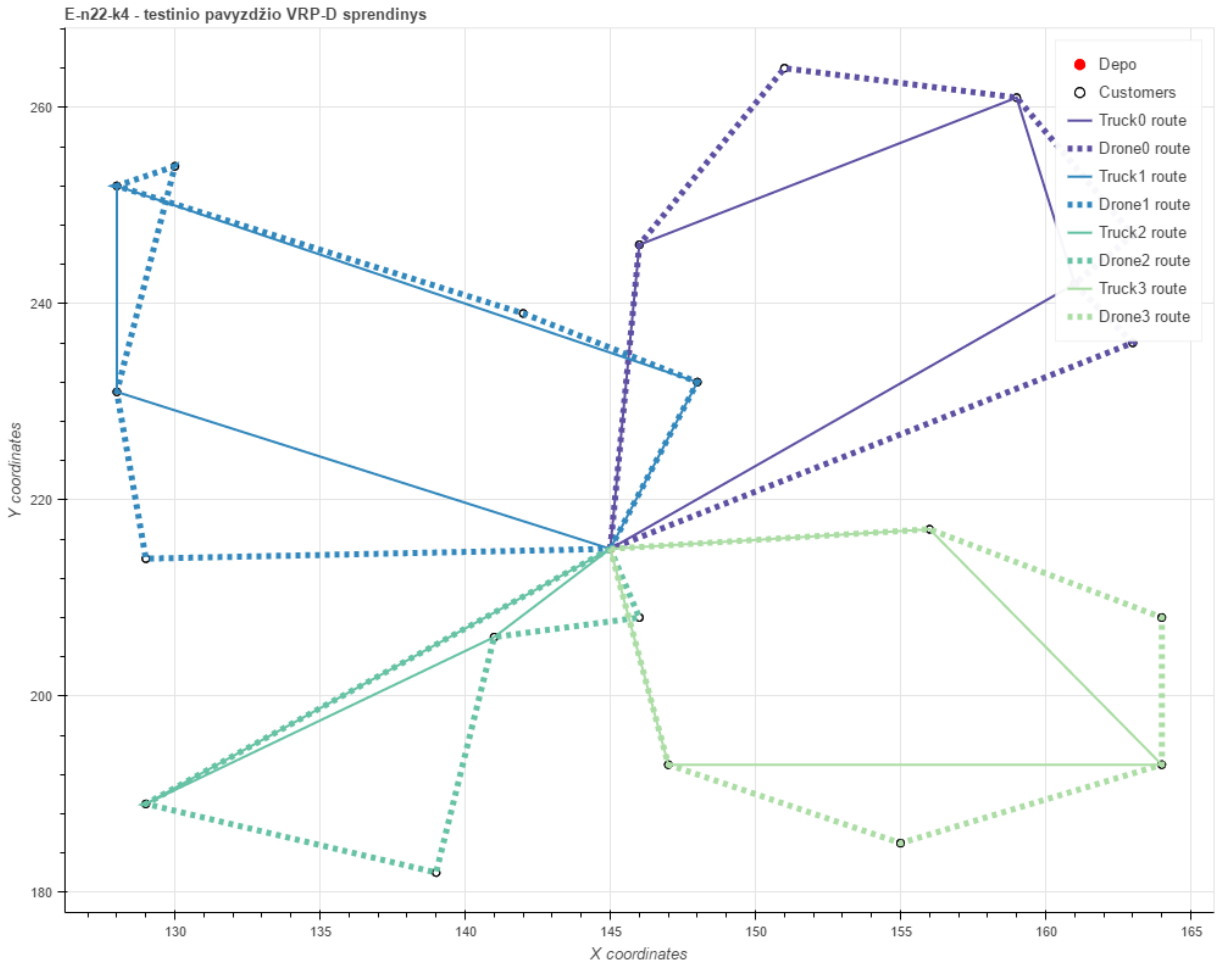
23 pav


E-n101-k14 - testinio pavyzdžio VRP sprendinys
Depo
Customers
Y coordinates
X coordinates
0
20
40
60
80
0
10
20
30
40
50
60
70

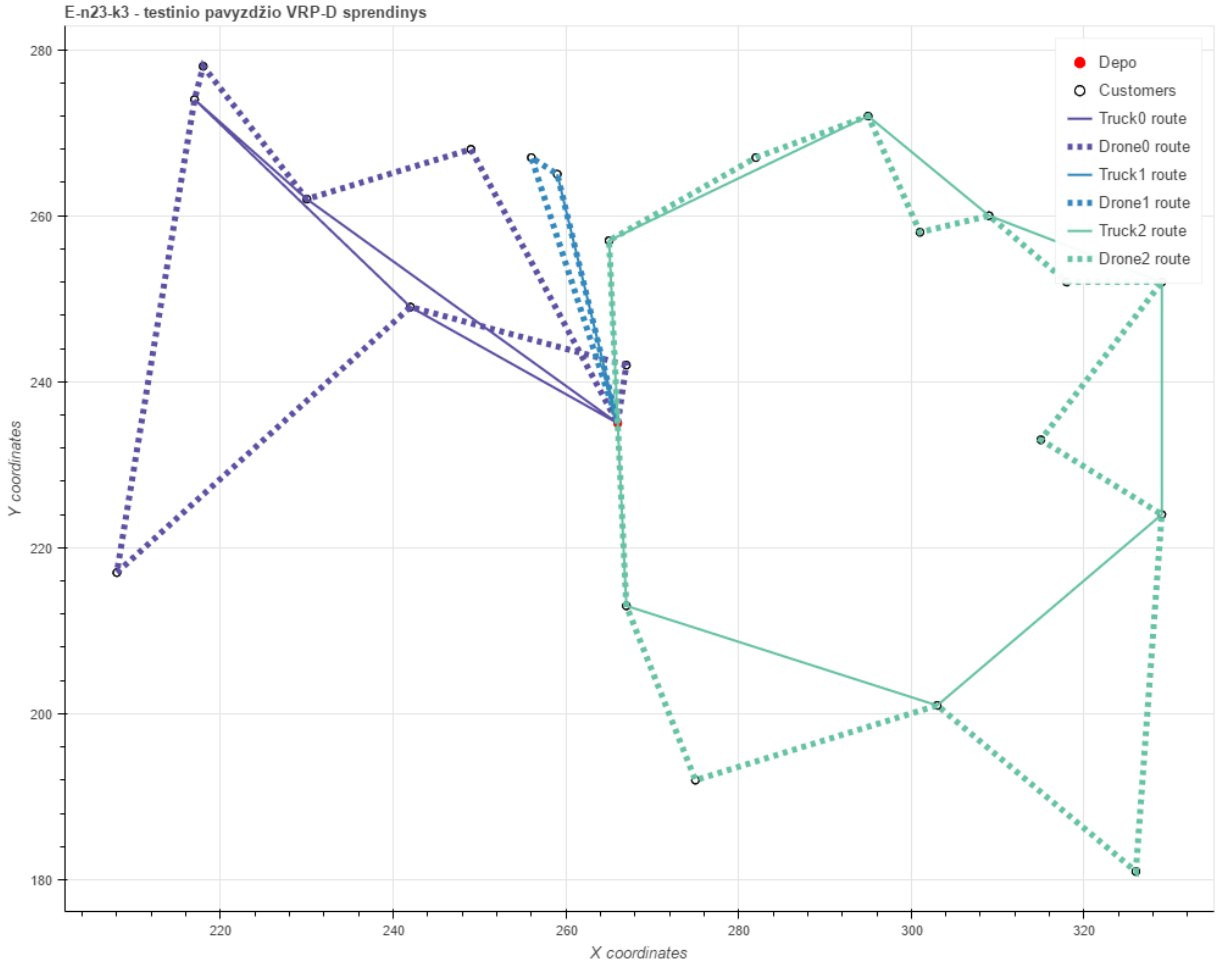
24 pav

Priedas Nr. 2

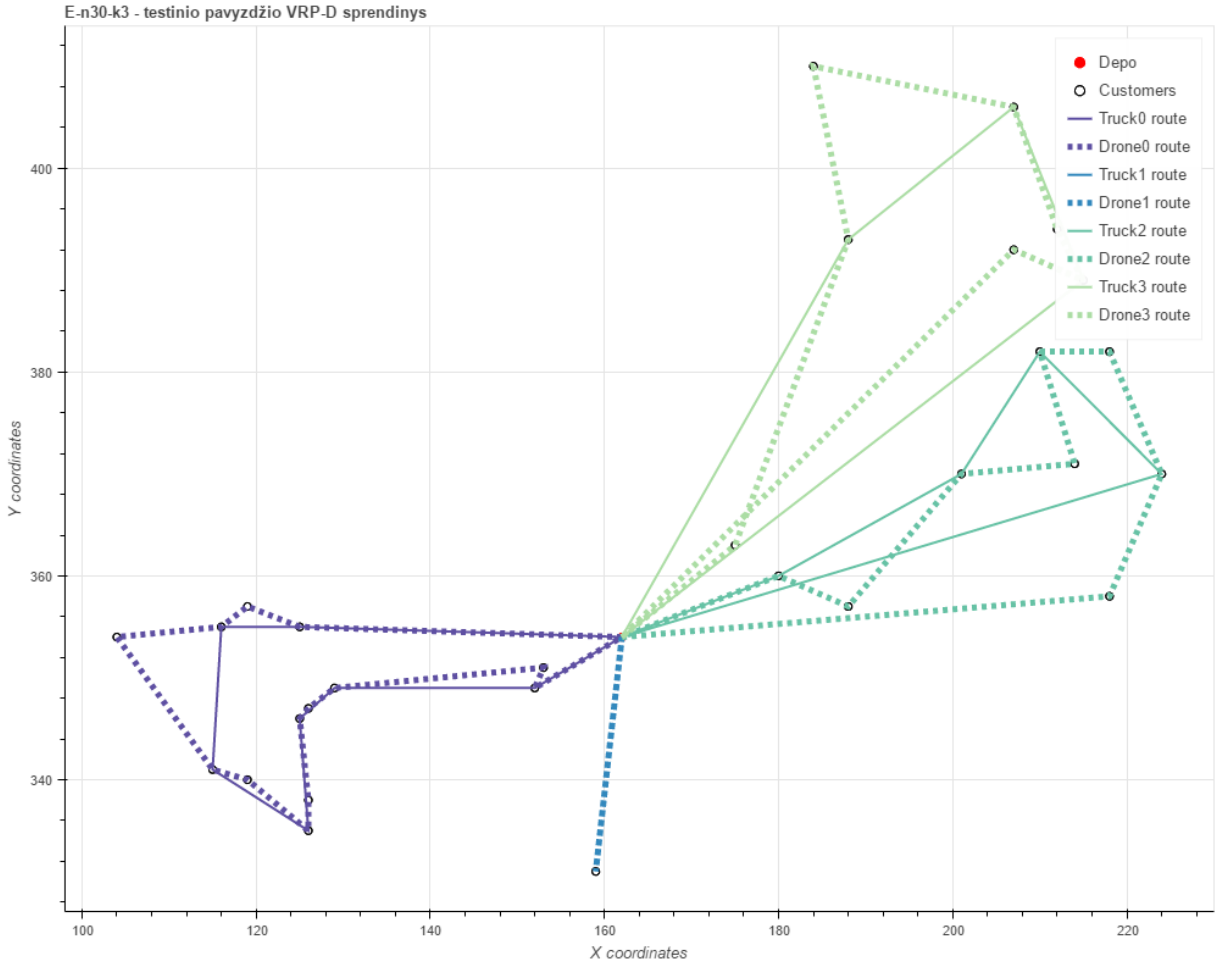
VRP-D geriausių sprendinių maršrutai



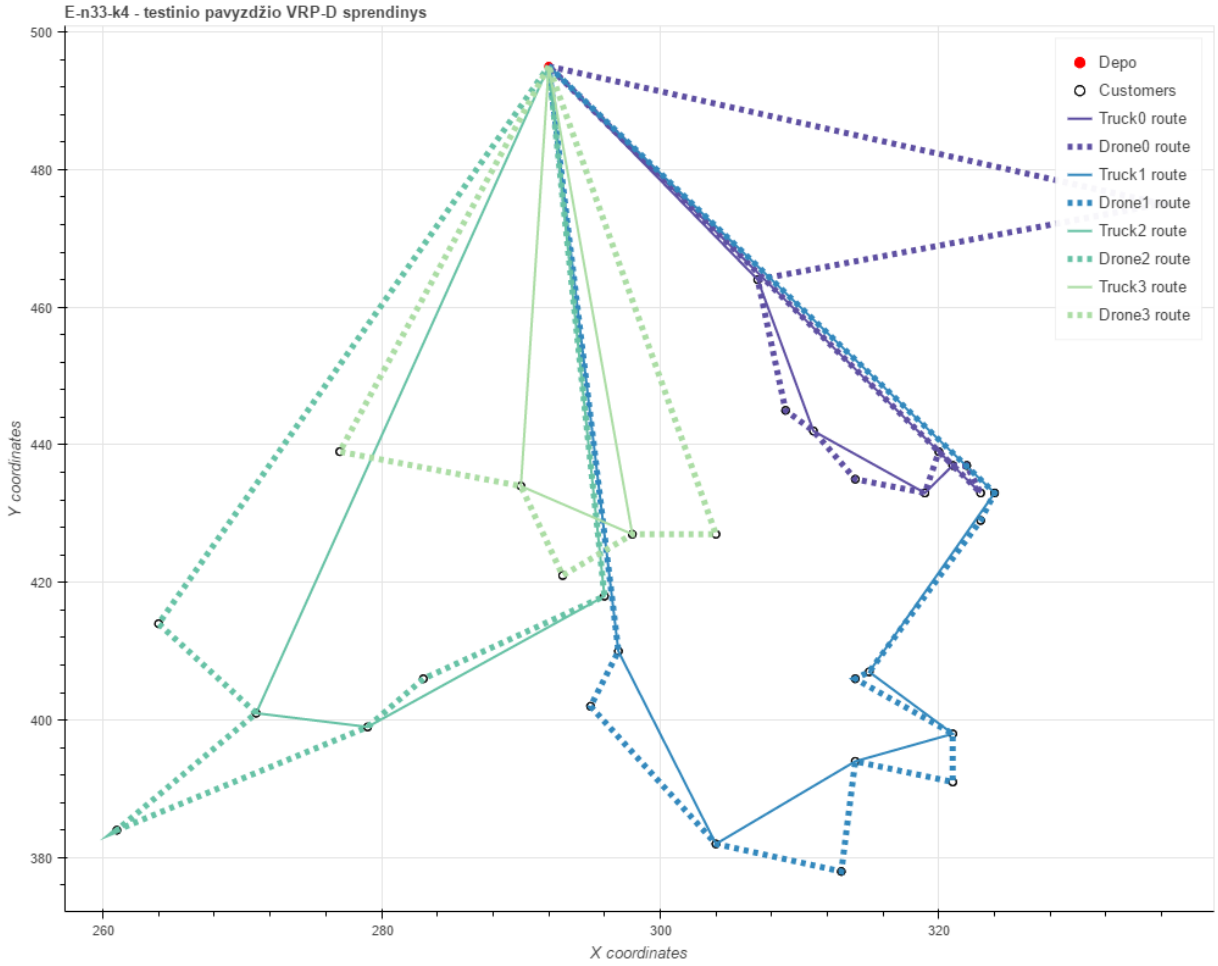
25 pav


E-n23-k3 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Y coordinates
X coordinates
280
260
240
220
200
180
220
240
260
280
300
320

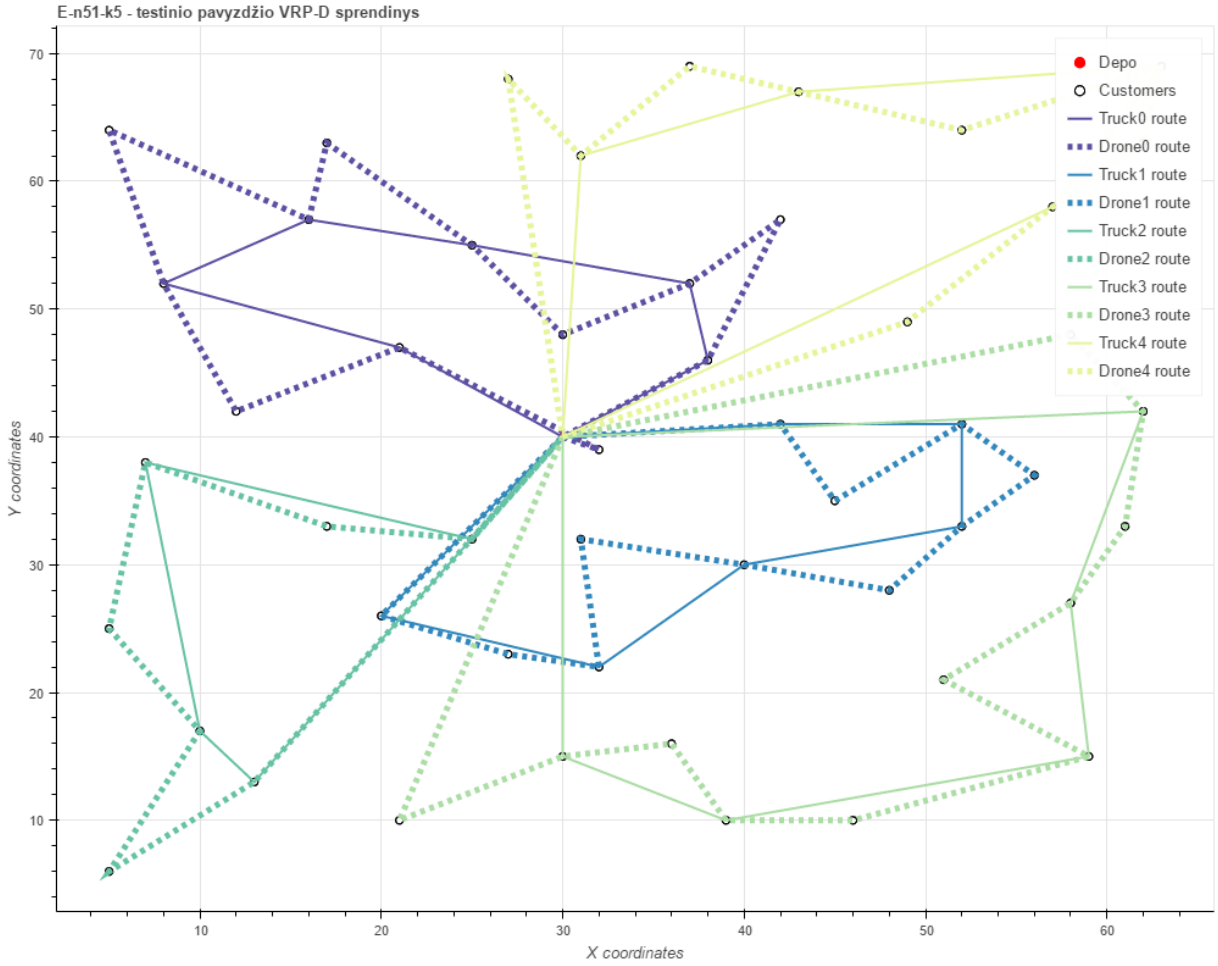
26 pav


E-n30-k3 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
400
380
360
340
Y coordinates
100
120
140
160
180
200
220
X coordinates

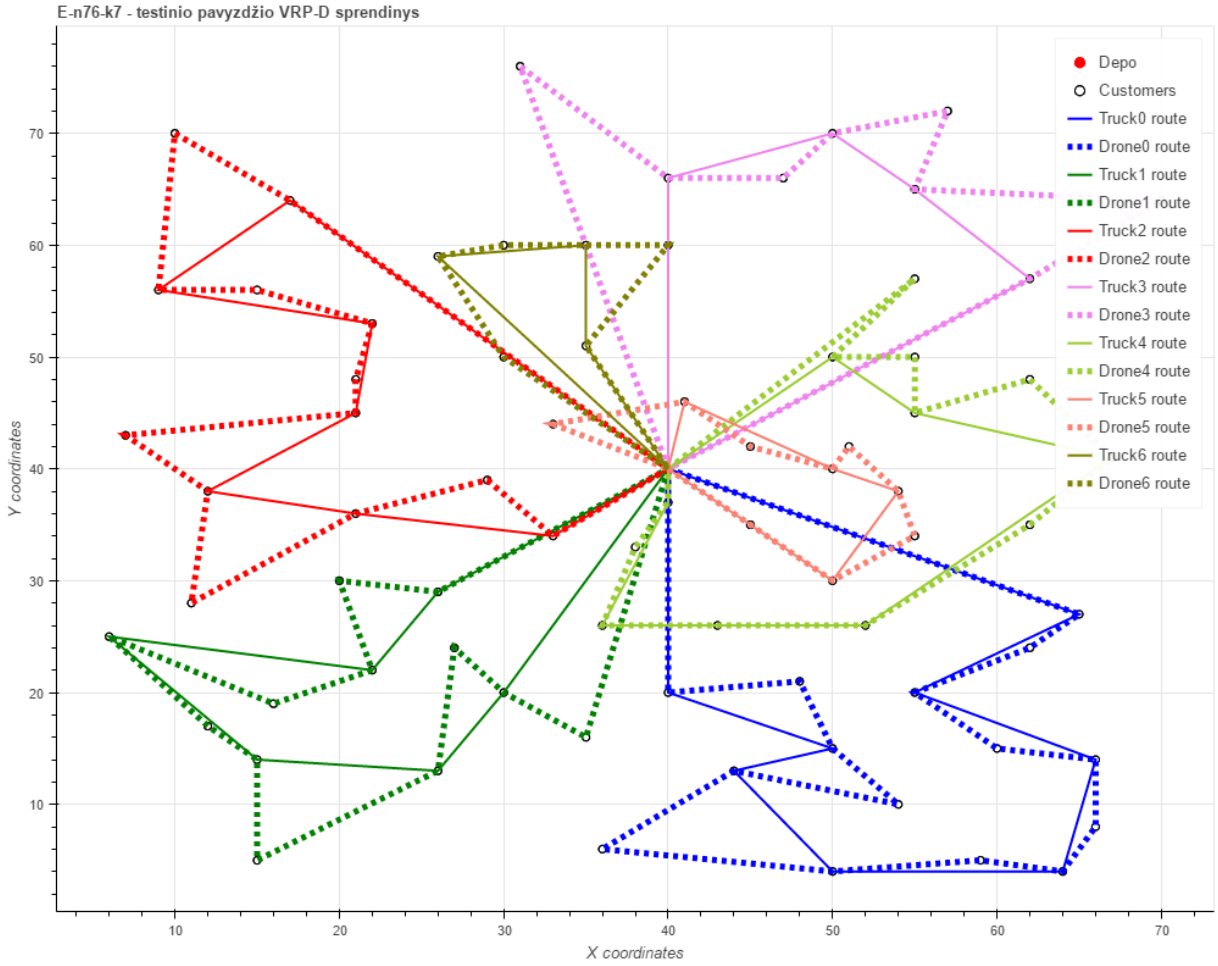
27 pav


E-n33-k4 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Y coordinates
X coordinates
500
480
460
440
420
400
380
260
280
300
320

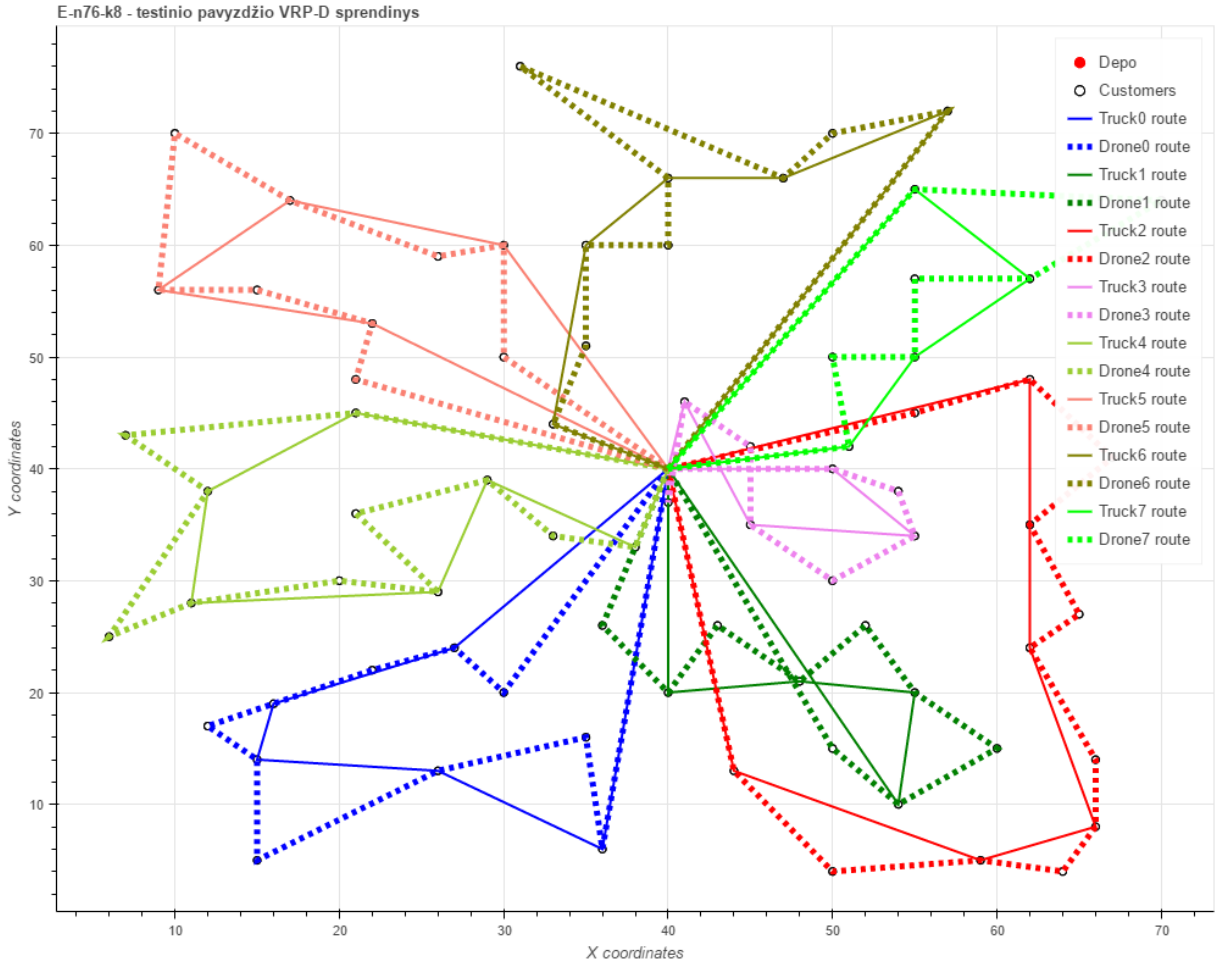
28 pav


E-n51-k5 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Y coordinates
X coordinates
70
60
50
40
30
20
10
10
20
30
40
50
60

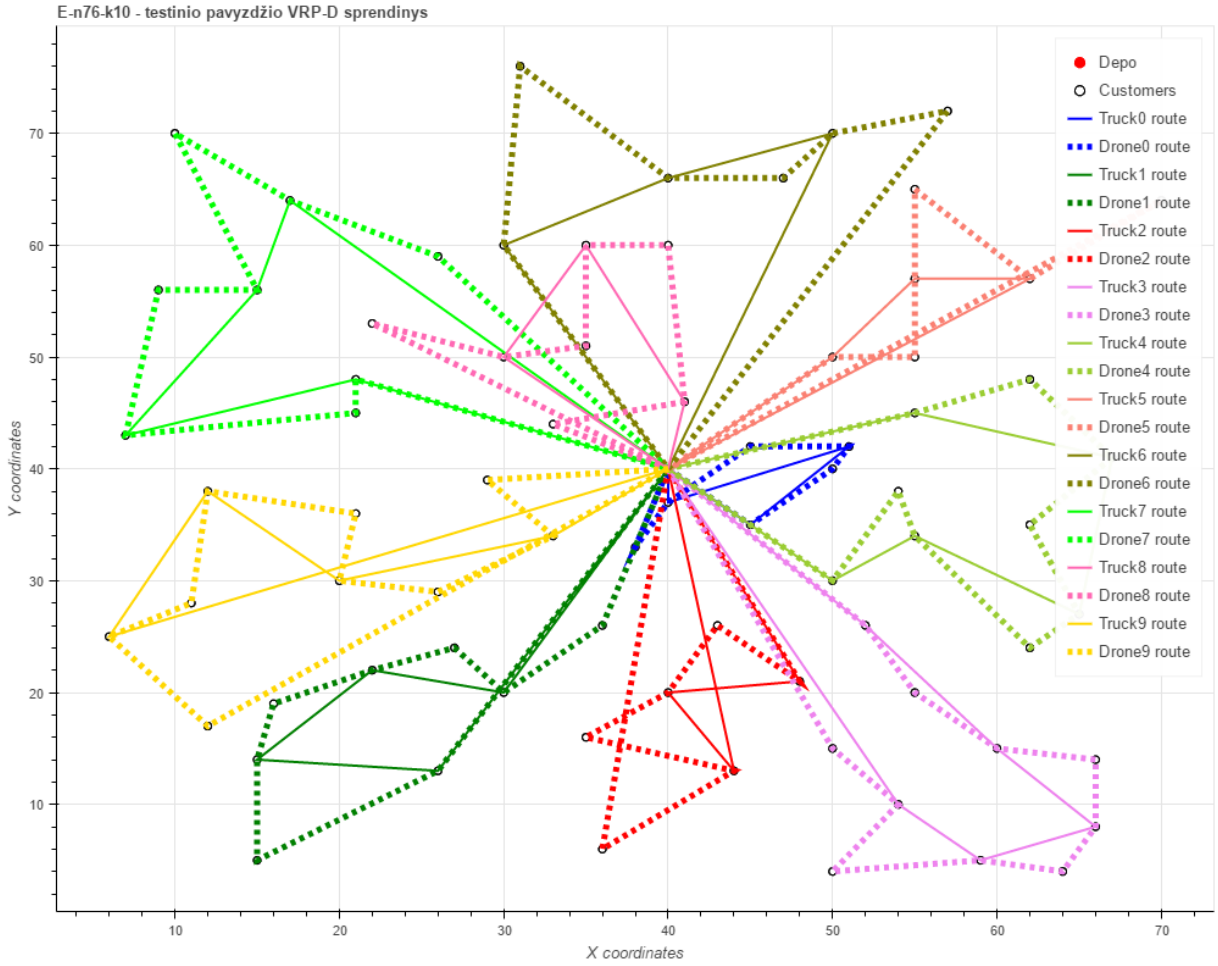
29 pav


E-n76-k7 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Y coordinates
X coordinates
10
20
30
40
50
60
70

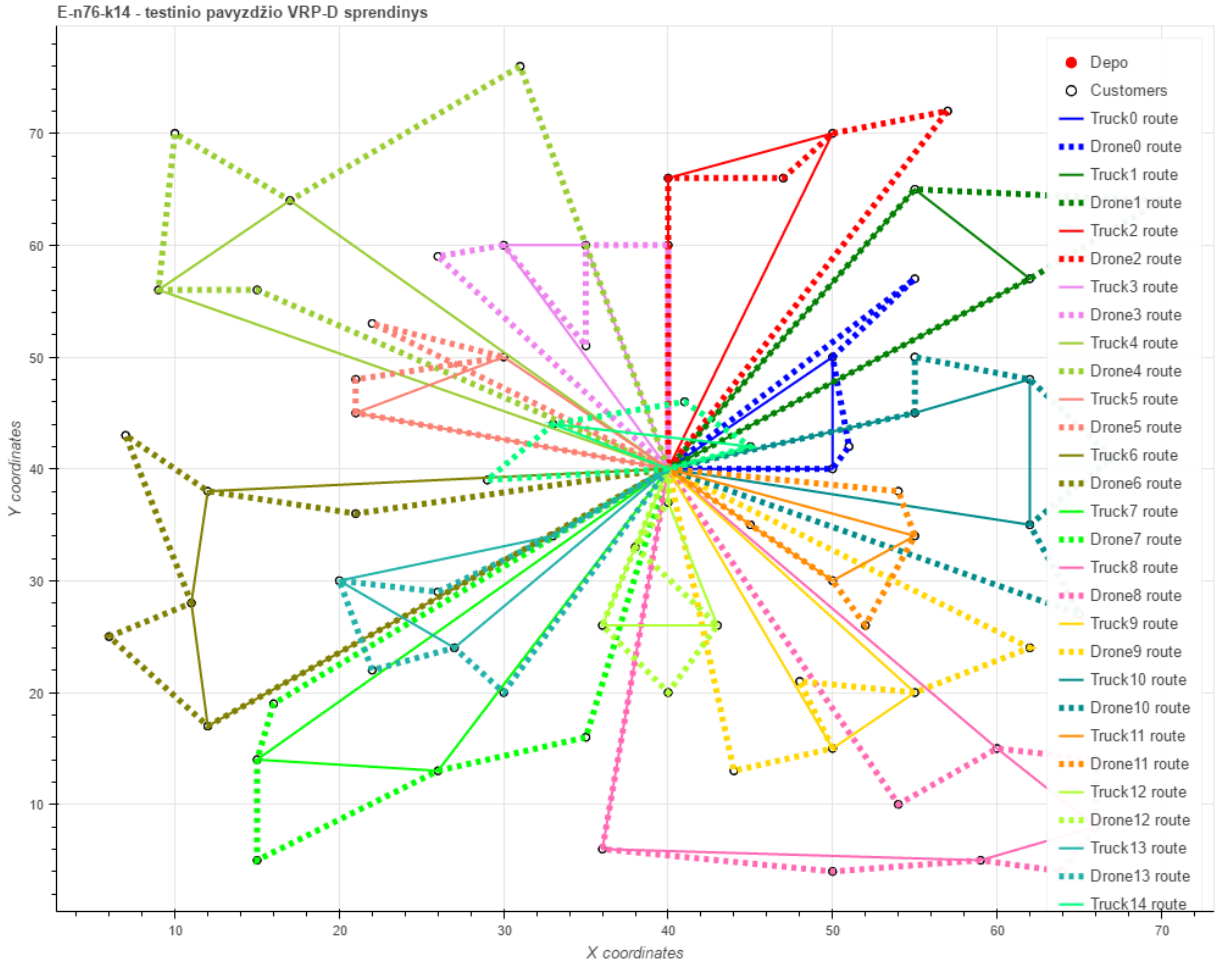
30 pav


E-n76-k8 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Truck7 route
Drone7 route
Y coordinates
X coordinates
10
20
30
40
50
60
70

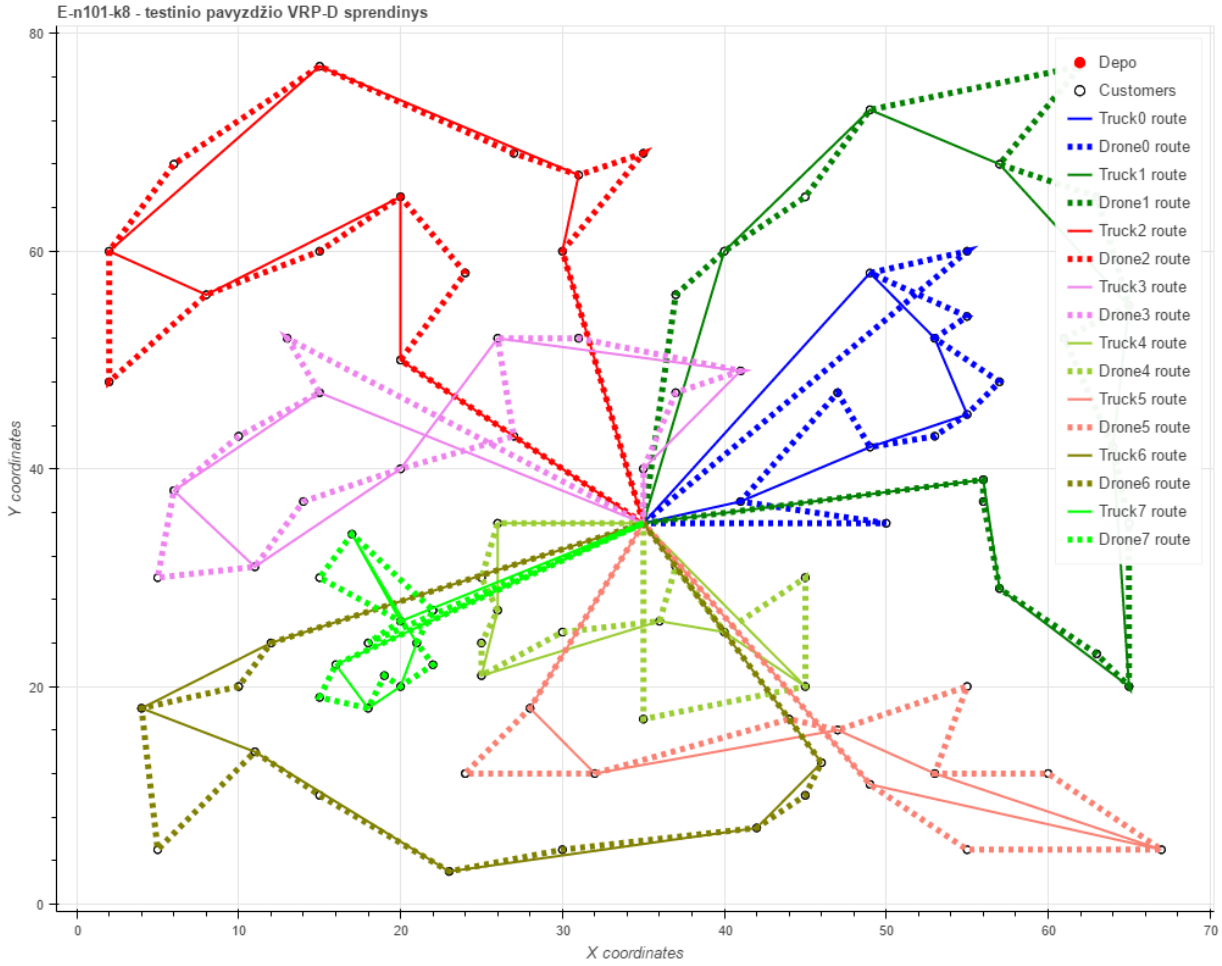
31 pav


E-n76-k10 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Truck7 route
Drone7 route
Truck8 route
Drone8 route
Truck9 route
Drone9 route
Y coordinates
X coordinates
10
20
30
40
50
60
70

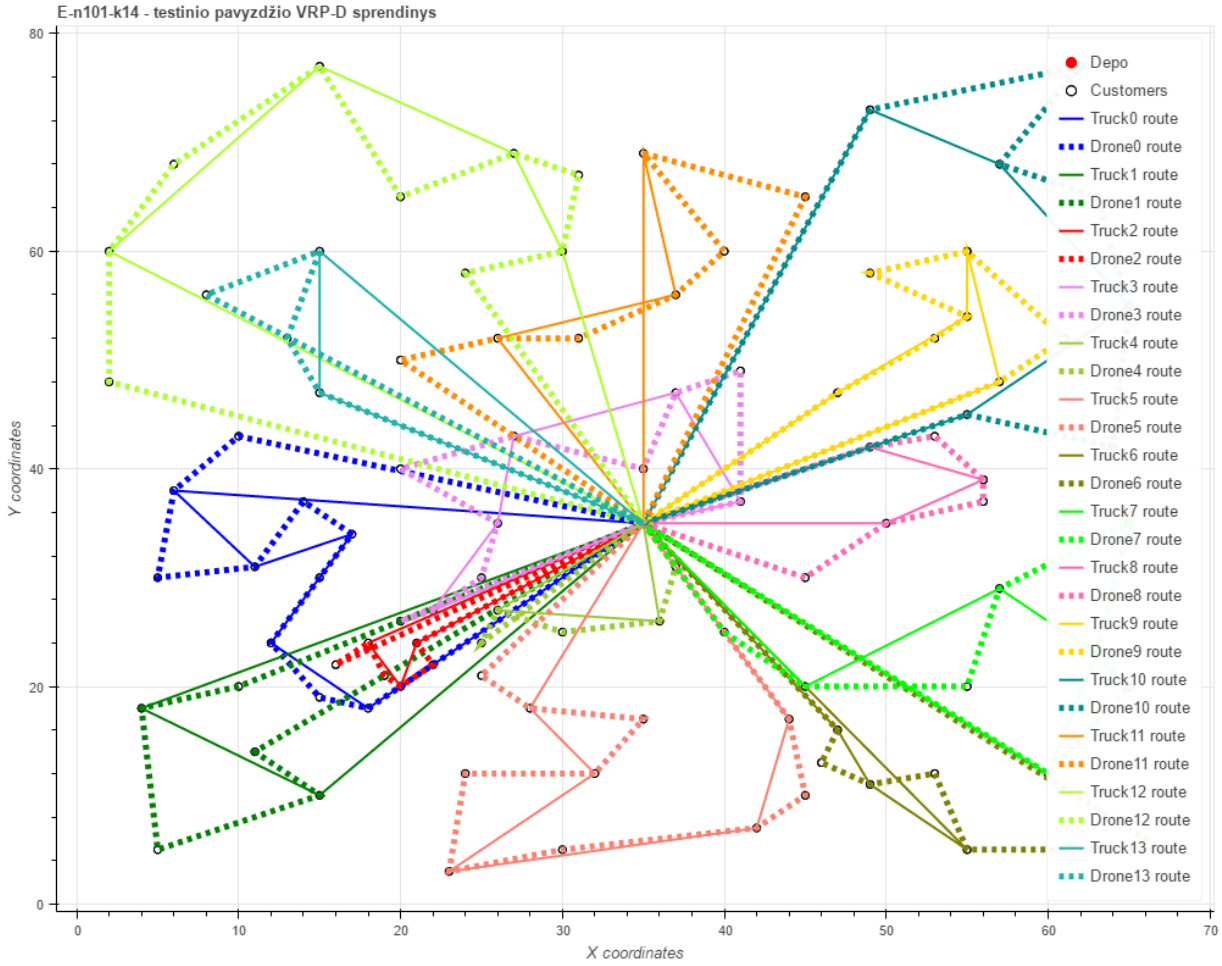
32 pav


E-n76-k14 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Truck7 route
Drone7 route
Truck8 route
Drone8 route
Truck9 route
Drone9 route
Truck10 route
Drone10 route
Truck11 route
Drone11 route
Truck12 route
Drone12 route
Truck13 route
Drone13 route
Truck14 route
Y coordinates
X coordinates
10
20
30
40
50
60
70

33 pav


E-n101-k8 - testinio pavyzdžio VRP-D sprendinys
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Truck7 route
Drone7 route
Y coordinates
X coordinates
0
20
40
60
80
0
10
20
30
40
50
60
70

34 pav


E-n101-k14 - testinio pavyzdžio VRP-D sprendinys
Y coordinates
X coordinates
0
20
40
60
80
0
10
20
30
40
50
60
70
Depo
Customers
Truck0 route
Drone0 route
Truck1 route
Drone1 route
Truck2 route
Drone2 route
Truck3 route
Drone3 route
Truck4 route
Drone4 route
Truck5 route
Drone5 route
Truck6 route
Drone6 route
Truck7 route
Drone7 route
Truck8 route
Drone8 route
Truck9 route
Drone9 route
Truck10 route
Drone10 route
Truck11 route
Drone11 route
Truck12 route
Drone12 route
Truck13 route
Drone13 route

35 pav