VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistrinis darbas

Maršrutų optimizavimo algoritmai

Atliko:
Marius Budrauskas parašas

Vadovas:
prof. dr. Tadas Meškauskas

Vilnius
2019

Turinys

Sutartinis terminų žodynas

1. TSP - (Travel Sales Problem) Keliaujančio pirklio problema.
2. MTSP - (Multiple Travel Sales Problem) Kelių keliaujančių pirklių problema.
3. VRP - (Vehicle Route problem) Transporto priemonės maršruto problema.
4. CVRP - (Capacity Vehicle Route Problem) riboto dydžio talpos mar6ruto problema.

Santrauka

Šiuo metu vis didesnę svarbą turinčioje transporto srityje vis dar nėra galutinai išspręsta optimalaus maršruto sudarymo problema, todėl jaučiamas poreikis tobulinti esamus maršrutų optimizacijos algoritmus. Dėl šios priežasties darbe siekiama apžvelgti pagrindinius maršrutų optimizacijos algoritmus, juos išanalizuoti, išbandyti bei palyginti tarpusavyje. Remiantis atlikta analize buvo parašytas visų tirtų algoritmų programinis kodas ir atliekamas eksperimentinis tyrimas su bandomaisiais duomenimis. Kadangi darbe nagrinėjamos trys problemos, egzistuojančios nuo senų laikų kaip neišsprendžiami uždaviniai, pirmiausiai buvo atlikta teorinė uždavinių analizė, skirta bendram problematikos suvokimui. Atlikus algoritmo tyrimus ir gautus rezultatus buvo išsiaiškinta, kad „Modeliuojamo atkaitinimo" algoritmas yra optimaliausias. Šio tyrimo metu algoritmai buvo skirti pilnam grafui. Taip pat, pagal laiką geriausias algoritmas yra „Artimiausio kaimyno", tačiau jis pagal kokybę yra prasčiausias. Programos kūrimui buvo pasirinktas „Genetinis algoritmas", kuris yra pritaikytas (CVRP) talpos transporto maršruto problemai spręsti. Jis pritaikytas taip, kad atsižvelgtu į taškų transporto talpos kiekį, kad būtų galima pagal tai sudaryti maršrutą. Šis hibridinis algoritmas sudaro maršrutą pagal du kriterijus. Pirmas pagal atstumą, o antra pagal taško transporto iškrovimo talpą skirtam taškui. Šis sprendimas yra pritaikytas nepilnam grafui, o būtent transporto logistikos žemėlapui. Šiai programai reikia pateikti taškų koordinates remiantis „Google" žemėlapiu platumą ir ilgumą. Reikia pateikti duomenis transporto išvykimo, pasikrovimo tašką ir nuruodyti, kokia yra transporto galima talpą. Privaloma sukelti duomenių užklausas, kokią iškrovimo talpą sudaro taškas. Programa atvaizduoja sudarytus maršrutus ir sudaro nuoradas skirtaa žemėlapui.

Summary

Route optimization algorithms

Nowadays, the problem of optimal route creation is still not fully resolved in an increasingly important transport field, and there is a need to improve existing route optimization algorithms. Therefore, this paper seeks to overview the main algorithms of route optimization, analyze them, test and compare them with each other. On the basis of the analysis, the programming code of all algorithms was written and performed as an experimental study with the experimental data. As the work deals with three problems that existed from old times as unsolvable tasks, the analysis of theoretical tasks aimed at general perception of the problem was firstly performed. The results of the algorithm research and the results obtained revealed that the algorithm of ”simulated annealing” is the most optimal. In this study, algorithms were designed for full graph. Also, according to time, the best algorithm is "Nearest neighbor", but it is the worst quality. "Genetic Algorithm", which is adapted to address the (CVRP) capacity transport route problem, was chosen for program development. It is adapted to take into account the amount of points in the transport container to allow it to make a route. This hybrid algorithm forms a route according by two criteria. First by distance, and second by point for unloading capacity of point transport. This solution is adapted to the subgraph, namely for the transport logistics map. This program requires point coordinates based on the latitude and longitude of the Google map. It is necessary to provide data on the departure, the loading point of the transport and to indicate the available capacity of the transport. Require data requests for the landing capacity of the landing point. The program displays the drawn routes and makes a reference to the map.

Įvadas

Šiuolaikinėje visuomenėje vis didesnę svarbą kasdieniniame gyvenime bei įvairiose rinkose sudaro transportas. Greitėjant gyvenimo tempui, nuolat didėja transporto poreikis bei galimybė nusigauti į reikiamą vietą kuo greičiau. Siekiant užtikrinti optimalią kelionės trukmę, įvairios organizacijos kuria įrankius optimalaus maršruto sudarymui. Šių įrankių pagrindinis tikslas - sudaryti trumpiausią maršrutą nuo pažymėto pradžios taško iki nurodyto pabaigos taško per kuo mažesnį laiką. Deja, maršruto sudarymas vis dar priskiriamas NP klasės uždaviniui, kuriam reikalingi neriboti ištekliai.

Trumpiausio maršruto problema žinoma nuo pat senovės, kai buvo suformuotos optimalaus maršruto radimo problemos: Keliaujančio pirklio problema bei Transporto priemonės maršruto problema. Šioms problemoms išspręsti nuolat kuriami vis nauji algoritmai, sudarantys arba optimizuojantys jau sudarytą kelionės maršrutą. Deja, dažniausiai algoritmai veikia ženkliai per ilgai, todėl jų negalima realiai naudoti rinkoje. Kita pagrindinė problema - greitai veikiantys algoritmai ne visada suranda optimalų maršrutą, todėl ne iki galo įgyvendinamas pirminis algoritmo tikslas - surasti trumpiausią maršrutą. Buvo atltikti algoritmų tyrimas ir pagal gautą maršruto atstumą ir kokybę buvo sudarytas turnyrinė lentelę (žr. 3.3 skyrelį.). Optimaliausias algoritmas „Modeliuojamo atkaitinimo" algoritmas, o greičiausias algoritmas „Artimiausio kaimyno" algoritmas, tačiau jo sudaryto maršruto kokybė yra prasčiausias. Yra daug susijusių darbų, kurie sprendžia šias algoritmo problemas, tačiau visi pritaikyti yra pilname grafe. Šis hibridinis algoritmas skirtas (CVRP) talpos transporto maršruto problemai spręsti yra tuo išskirtinis, kad pritaikyas nepilnam grafui ir sudaro maršrutą pagal atstuma ir iškrovimo talpą. Tačiau algoritmų analizei buvo pritaikyti algoritmai pilname grafui ir sprendžia tik (TSP) transporto pirklio problema. Ir iš gautų rezultatų sudarytas turnyrinę lentelių rezutltati pagal sudarytą maršruto atstuma ir algoritmo laiką.

Darbo tikslas - suformuoti hibridinį optimalaus maršruto radimo algoritmą, pagal išanalizuotus trumpiausio maršruto radimo algoritmus. Siekint pagrindinio darbo tikslo, suformuoti uždaviniai:

1. Atlikti TSP ir VRP problemų analizę;
2. Išanalizuoti maršrutų optimizacijos algoritmus;
3. Išbandyti išanalizuotus algoritmus ir palyginti gautus rezultatus;
4. Suformuoti naują hibridinį maršrutų optimizavimui skirtą algoritmą skirtą spręsti (CVRP) talpos transporto maršruto problemai, remiantis atliktais bandymais ir analize.
5. Pateikti išvadas ir rezultatus.

Sukurtas hibridinis algoritmas, kuris sprendžia (CVRP) talpos transporto maršruto problemą. Šis sukurtas įrankis gali būti naudojamas įvariuose logistikos įmonėse, nes šis sprendimas pritaikytas nepilnam grafui ir skirtas žemėlapui. Taip pat, galima nuruodyti transporto galimą talpa ir pateikti užklausas, koki talpa reikia iškrauti būtent tame taške. Šis algoritmas pirma sprendžia atstuma, o vėliau tikrina ar įmanoma iškrauti transporto talpa skirtam taške. Jei taškas viršija viso maršruto talpa, algoritmas tada grįžta į pradinę tašką pakrovimo tašką. Šis hibridinis algoritmas pritaikytas būtent nepilnam grafui.

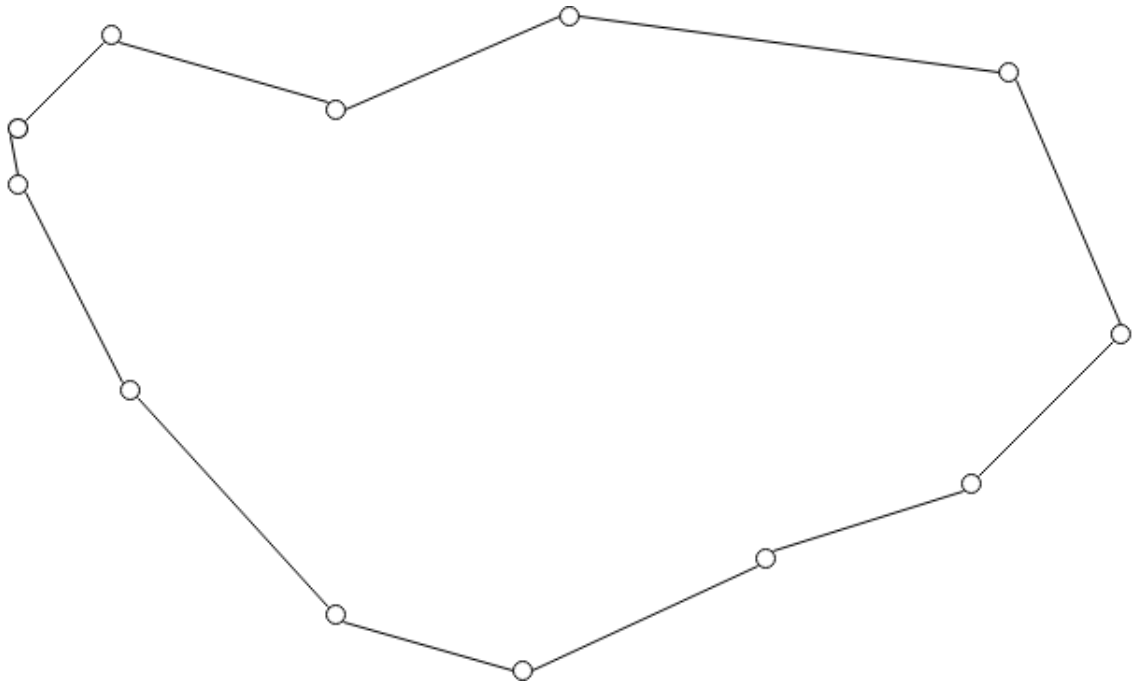
1. Sprendžiamos problemos

Šiame skyriuje aprašomos pagrindinės problemos, kurias darbe planuojama toliau nagrinėti ir spręsti. Išskiriamos šios pagrindinės maršrutų problemos: TSP ir VRP. Kiekviena iš problemų aprašoma, siekiant išvengti dviprasmybių bei netinkamo problematikos suvokimo.

1.1. TSP - keliaujančio pirklio problema

Keliaujančio pirklio problema (toliau TSP) yra gerai žinoma skaičiavimo problema, kuriai daugelį metų skiriama daug dėmesio, sprendžiant įvairius optimizacijos uždavinius. TSP problema buvo suformuluota pirmą kartą 1930 m. ir tai yra viena iš labiausiai analizuojamų problemų. Trumpiausias maršrutas yra sunkiai apskaičiuojamas, nors žinoma daugybė euristinių ir tikslių jų algoritmų. TSP yra suformuluota kaip problema, kurioje pirklys siekia aplankyti miestų rinkinį kuo mažesnėmis sąnaudomis, kiekvieną miestą aplankant tiksliai vieną kartą ir maršruto pabaigoje sugrįžtant į pradinį miestą, nuo kurio buvo pradėtas maršrutas [8]. Šis metodas gali būti taikomas daugeliui skirtingų problemų spręsti: transporto priemonių optimalaus maršruto radimui, skylių gręžimo į PCB plokščių gamybą tvarkai nustatyti, pluoštinių optinių kabelių optimaliam tinklo išdėstymui ir pan.

Atsižvelgiant į miestų išsidėstymą ir atstumus tarp kiekvieno miesto TSP problemoje keliamas pagrindinis klausimas - kaip sudaryti trumpiausią maršrutą, aplankant visus miestus?



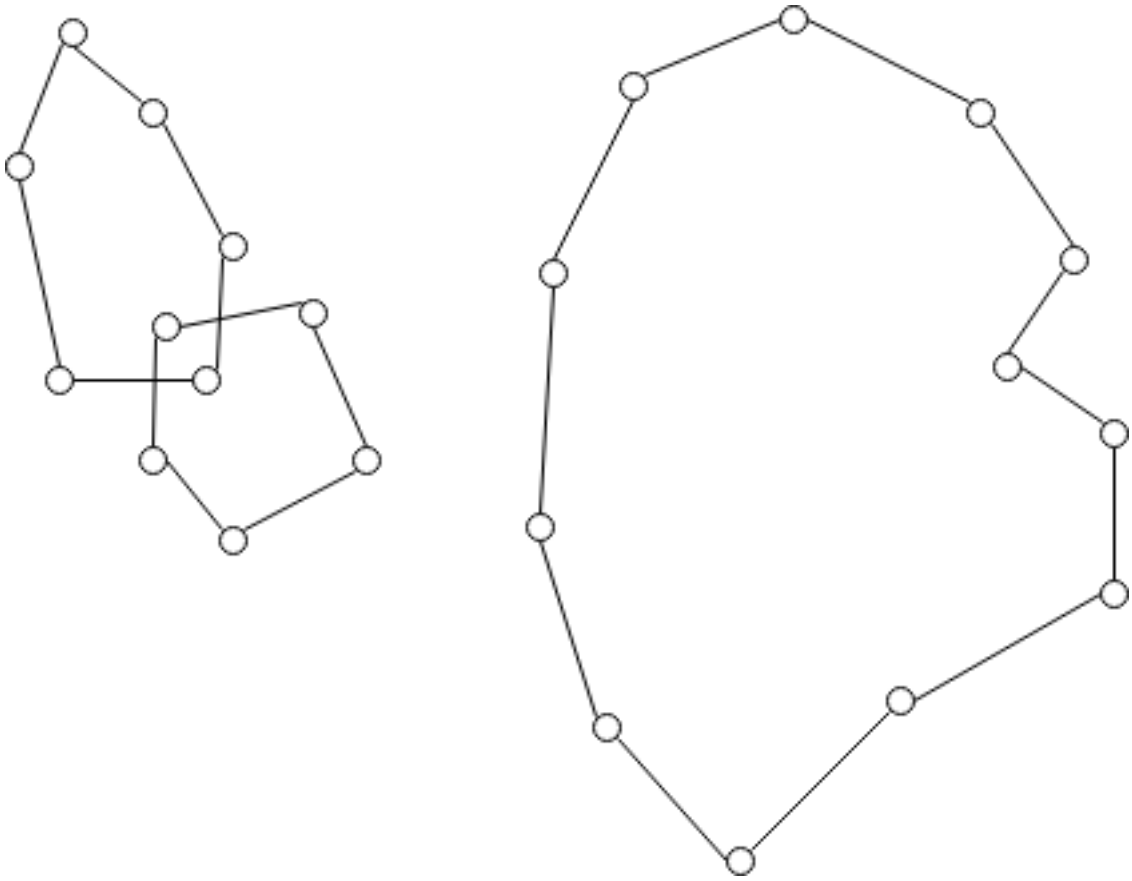
1 pav. TSP problema

TSP problemai išspręsti buvo pasiūlyta daugybė euristinių ir aproksimacijos algoritmų, leidžiančių sukurti naudingus TSP sprendimus, tokius kaip neuroniniai tinklai, imituotas kaitinimas, genetinis algoritmas ir daugybė kitų. Šie metodai gali sukurti gerus TSP sprendimus, bet jie vis dar naudoja daug resursų didelėje TSP, o skaičiavimo trukmė yra per didelė. [6]

Keliaujančio pirklio problema buvo apibendrinta ir suformuluota kelių keliaujančių pirklių problema (toliau MTSP). Kelių keliaujančių pirklių problemoje maršrute dalyvauja daugiau nei vienas pirklys. MTSP tikslas yra nustatyti maršrutų rinkinį m pirkliams, siekiant minimizuoti visų

maršrutų m sąnaudas. Metrinė kaina šiuo atveju gali atspindėti kainą, atstumą arba laiką. Pagrindiniai reikalavimai, keliami MTSP:

- Visi maršrutai turi prasidėti ir pasibaigti tuo pačiu miestu;
- Kiekvienas miestas turi būti aplankomas tik vieną kartą ir tik vieno pirklio.



2 pav. MTSP problema

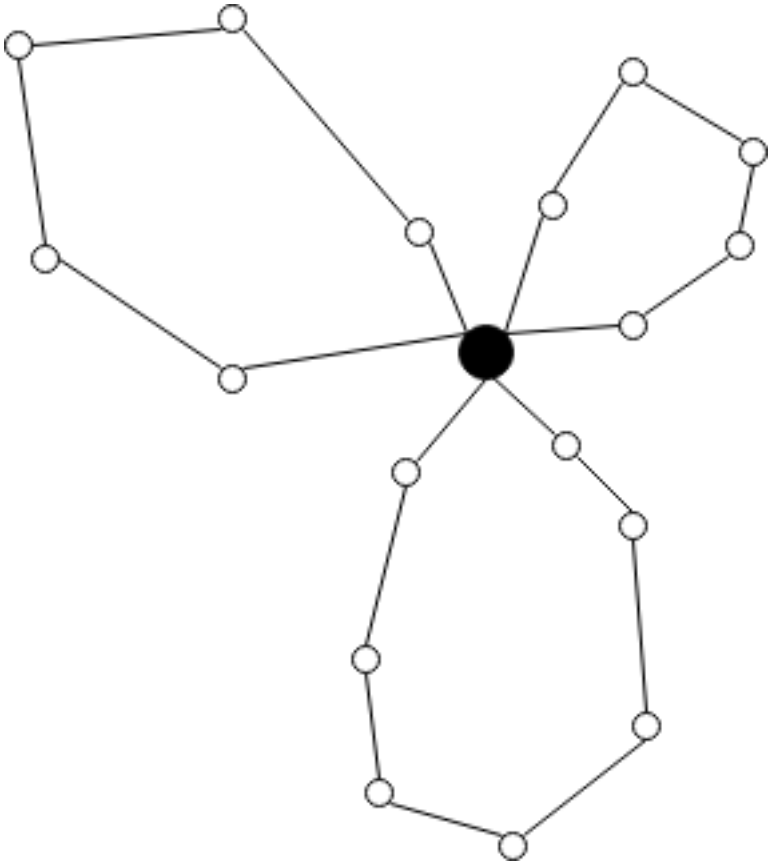
Viena iš pagrindinių TSP ir MTSP sprendimo problemų yra uždavinio sudėtingumas matematiniu požiūriu. Yra daugybė algoritmų, sprendžiančių šią problemą, tačiau, algoritmams pateikus daug duomenų, jie akivaizdžiai sulėtėja ir nebesugeba išspręsti uždavinio per pakankamai trumpą laiką. Pavyzdžiui, Dijkstra algoritmas geba spręsti tokio pobūdžio uždavinius, tačiau, pateikus didesnį duomenų kiekį, jis akivaizdžiai sulėtėja ir nesugeba pateikti sprendimo per protingą laiko tarpą. Nagrinėjant daug duomenų viename maršrute ženkliai didėja tikimybė, kad rastas maršrutas nebus optimalus.

1.2. VRP - Transporto priemonės maršruto problema

Transporto maršruto problema (toliau VRP) pastebima siekiant sukurti optimalių maršrutų rinkinį, kurių metu reikia aplankyti keletą skirtingose geografinėse vietose esančių miestų, pradedant nuo konkretaus vieno nurodyto miesto. VRP ypač ryškiai pastebima fizinio pasiskirstymo ir logistikos srityse. Ši problema kelia pagrindinius klausimus:

- Kaip būtų galima sudaryti daug optimalių maršrutų iš vieno centrinio miesto?
- Koks turėtų būti sprendimas norint aplankyti visus miestus, sutaupant laiką?

George'as Dantzigas ir Johnas Ramseris 1959 m. buvo pirmieji, kurių parašytas algoritmas VRP sprendimui rasti buvo pradėtas taikyti kuro pristatymui į degalines.



3 pav. VRP problema [12]

VRP iš dalies apibendrina TSP problemą. Vienintelis skirtumas tarp jų - TSP tikslas sudaryti vieną maršrutą, o VRP siekia sudaryti daug ir trumpų maršrutų iš to pačio miesto. Taip pat išskiriamos pagrindinės VRP variacijos [14]:

- Riboto dydžio talpos maršruto problema (CVRP). Pagrindinis šios specializacijos bruožas - transporto priemonė turi riboto dydžio krovinį, kurį reikia pristatyti optimaliu maršrutu.
- Atvira transporto priemonių maršruto problema (OVRP): Transporto priemonei nereikia grįžti į centrinį miestą.

- Kelių saugyklų problema. Tai reiškia, kad vienas maršrutas prasideda viename centriniame mieste, o pasibaigia kitame. Iš to centrinio miesto turi prasidėti kitas maršrutas, pasibaigti dar kitame ir t. t. Taip pat gali būti daug centrinių miestų su skirtingais arba tais pačiais maršrutais.

- Daugelio maršrutų periodinio laiko VRP. Pavyzdžiui, jei transporto atvykimo plano laikotarpis yra ilgesnis nei buvo numatytas, problema yra vadinamas periodiniu VRP. Periodinėje transporto priemonių maršruto problemoje tikslas yra rasti optimalų dienos laikotarpio maršrutą.

- Transporto priemonių maršruto problema su prekės paėmimu ir pristatymu (VRPPD). Daugelį prekių reikia pervežti iš vienų miestų į kitus. Šios problemos tikslas yra rasti optimalius maršrutus transporto priemonių parkui, kad būtų galima aplankyti visus prekių pakrovimo ir iškrovimo miestus.

- Transporto priemonių maršruto problema su laiko langais (VRPTW): pristatymo vietose yra laiko langai, kurių metu transporto priemonė turi atvykti į miestą.

- Transporto priemonių maršruto problema su keliais važiavimais (VRPMT). Pagrindinis iššūkis - transporto priemonės gali važiuoti daugiau nei vienu maršrutu.

Siekiant išspręsti didelio masto transporto priemonių maršrutų problemas, buvo skiriamas didelis dėmesys meta-euristiniams algoritmams, kurie yra skirti kompleksinių problemų sprendimui, nes įprasti algoritmai geba spręsti tik po vieną problemą.

1.3. CVRP - Riboto dydžio talpos maršruto problema

Riboto dydžio talpos maršruto problema (toliau - CVRP) yra VRP tipo problemos išplėtimas. CVRP tipo problemoje transporto priemonėms pridedamas papildomas talpos ribojimas. Tai reiškia, kad kiekviena transporto priemonė turi tam tikrą riboto dydžio talpumą, todėl į ją gali būti įkrautas ribotas prekių kiekis, neviršijantis maksimalios transporto priemonės talpos. Šios problemos sprendimo tikslas yra rasti optimalų kelią su minimaliomis transportavimo išlaidomis, pasiekiant maksimalų kliento pasitenkinimą, atitinkant nustatytus talpos apribojimus. Pastaruosius tris dešimtmečius šio tipo problema buvo plačiai nagrinėjama, daugelis tyrimų buvo atliekami siekiant ją išspręsti. Keletas metodų buvo sukurti, remiantis paieška tiesioginiame medyje pagal šakų ir skilčių (angl. "Branch and Bound") (Christofides ir Eilon, 1969), kolonų generavimo algoritmus ir metauristiką. Visi šie metodai geba pasiūlyti kokybiškus sprendimus šio tipo problemai. Apibendrinant, pagrindinis CVRP tipo problemos išskirtinumas - transporto priemonė geba pervežti riboto dydžio krovinį, kurį reikia pristatyti optimaliu maršrutu. [10]

1.3.1. Transporto užklausos

Transporto užklausų pateikimas yra ganėtinai svarbi maršruto problemų ypatybė. Egzistuoja du pagrindiniai transporto užklausų pateikimo būdai - dinaminis ir statinis. Dinaminis būdas remiasi reikalavimų žinojimumaršruto vykdymo metu. Statinio transporto užklausų pateikimo būdu visos uxklausos yra žinomi prieš marštutų sudarymą. Laikysime, kad CVRP tipo problemoms visos transporto užklausos yra pateikiamos statiniu būdu, t. y. žinomos iki maršrutų sudarymo.

1.3.2. Maršrutų optimizavimo galimybės

Šiame skyriuje trumpai apžvelgiamos dažniausiai siūlomos maršrutų optimizavimo galimybės, analizuojant maršrutų optimizavimą iš skirtingų perspektyvų.

Transporto priemonių mažinimas: Transporto priemonių kiekis nustato, kiek priemonių kiekvieną dieną vežioja krovinius iš centrinio mazgo. Akivaizdu, kad maršruto mazgai, transporto priemonės bei jų vairuotojai kainuoja daugiausiai krovinių transportavimo kaštų. Mažesnis transporto priemonių kiekis atitinkamai prisideda prie maršruto optimizavimo, nes maršrutai planuojami efektyviau.

Maršruto ilgio mažinimas: Maršruto ilgis - bendras atstumas, kurį nukeliauja transporto priemonė, gabendama krovinį nuo pirminio iki galutinio mazgo kartu su grįžimu į pirminį mazgą. Maršruto ilgis turėtų daryti didelę įtaką kelionės kainai, tačiau didesnę įtaką tam daro bendra eismo situacija, į kurią dažniausiai neatsižvelgiama kelionės kainos ir sistemos veiksmingumo skaičiavimuose.

Maršruto trukmės trumpinimas: Maršruto trukmę sudaro krovinio vežimo laikas, pertraukos ir krovinio pakrovimo bei iškrovimo laukimo laikas. Transporto priemonės išvykimo iš centrinio mazgo laikas yra pradinis maršruto laikas, o maršruto pabaigos laikas fiksuojamas tuomet, kai transporto priemonė grįžta į centrinį mazgą, įvykdžiusi visą maršrutą. Sumažinus maršruto trukmę, sutaupomi kaštai tiek transporto priemonės naudojimo, tiek žmogiškųjų išteklių, t. y. vairuotojų, naudojimo atžvilgiu. Tokiu atveju, trumpesnė maršruto trukmė leidžia efektyviau išnaudoti transporto priemones bei žmogiškuosius išteklius, dalį jų perskirstant kitų maršrutų vykdymui.

Kelionės laiko mažinimas: Maršruto kelionės laiką sudaro kelių maršrutų bendras laikas, papildomai įskaičiuojant krovinių iškrovimo ir pakrovimo laiką. Pavyzdžiui, maisto banko veikloje visų vairuotojų atlyginimai yra fiksuoti, o vienas vairuotojas gali dirbti ne daugiau nei 11 valandų

per parą. Krovinių iškrovimo ir pakrovimo laikai skirtinguose mazguose yra taip pat fiksuotos trukmės, o krovinys prieš atvykstant transporto priemonei yra iš anksto paruošiamas pakrovimui. Šioje situacijoje, bendros kelionės trukmės sumažinimas yra logiškesnis sprendimas nei pavienių maršrutų trumpinimas, nes transporto priemonė per vieną kelionę aplanko kelis krovinio iškrovimo ir pakrovimo taškus, todėl yra labai svarbu minimizuoti laiką, praleidžiamą juose - tokiu būdu efektyviau išnaudojami žmogiškieji ištekliai bei transporto priemonių panaudojamumas.

Šiame tyrime yra gaunama tikroji kelionės laiko matrica, kadangi į ją yra įskaičiuojamas ir visas transporto priemonės stovėjimo laikas. Tikrosios kelionės trukmės skaičiavimas šio tyrimo rezultatus daro realistiškesnius ir paprasčiau pritaikomus praktikoje.

1.3.3. Problemos formuluotė

CVRP problemos sprendimas yra mišrusis skaičius su minimaliu transporto priemonių kiekiu ir optimalia kelionės trukme. Kadangi krovinių pervežimo paslaugų sektoriuje didžiausią kaštų dalį sudaro transporto priemonės ir žmogiškieji ištekliai, jų kiekis yra svarbesnis už bendrą kelionės trukmę.

Problemai išspręsti siekiama užtikrinti didesnį transporto priemonių kiekį, kurio pakaktų užtikrinti krovinių pervežimą reikalingais maršrutais. Tikslas - suformuoti maršrutą, pagal kurį turimas transporto priemonių kiekis gebėtų išvežioti visus krovinius. Atsižvelgiant į krovinių ir transporto priemonių kiekį, galimas maršrutų kiekio padidėjimas dėl transporto priemonės pervežamo krovinio talpos ribojimų. Svarbu pabrėžti, jog šiuo atveju problemos sprendimo laikas nėra svarbus.

2. Analizė

2.1. Maršrutų algoritmai

Šiame skyriuje aprašomi pagrindiniai algoritmai, naudojami tiriamajame darbe. Pasirinkti pagrindiniai optimizavimo algoritmai:

- Taillard's;
- Christofides;
- Artimiausio kaimyno;
- Modeliuojamo atkaitinimo;
- Dviejų pasirinktųjų apsikeitimo;
- Trijų pasirinktųjų apsikeitimo;
- Skruzdžių kolonijos algoritmas
- Genetiniai.

Šie algoritmai analizei buvo pasirinkti dėl skirtingo požiūrio, parametrizavimo galimybių bei veikimo principo. Išbandžius skirtingus algoritmus galima lengviau įžvelgti jų trūkumus bei formuoti hibridinį algoritmą, siekiant eliminuoti pagrindines klaidas.

2.1.1. Taillard's algoritmas

Taillard's algoritmas[17] akcentuoja mainų generavimo mechanizmą (Osman, 1993). Algoritmas individualius maršrutus pakartotinai atnaujina naudojant optimizavimo algoritmą (Volgenant ir Jonker, 1993). Vienas iš pagrindinių Talliard's algoritmo bruožų yra pagrindinių problemų skaidymas į smulkesnes, mažesnio lygmens problemas.[17]

Planarinėse problemose šios mažesnio lygmens problemos yra gaunamos iš pradžių padalijus viršūnes į sektorius, kurie yra centruojami į sandėlį, ir vėliau kiekvieną sektorių dalinant į koncentrinius regionus. Kiekviena paprogramė gali būti išspręsta savarankiškai, tačiau reikia periodiškų viršūnių perskirstymo į gretimus sektorius. Tai prasminga, kai sistema yra centralizuota ir viršūnės yra tolygiai paskirstytos plokštumoje.

Dėl neplanarinių problemų ir planarinių problemų, neturinčių šių savybių, siūlomas kitoks pasiskirstymo metodas, pagrįstas trumpiausiomis, į saugyklą įsišaknijusiaomis arborescencijomis. Šis problemų skaidymo metodas yra ypač tinkamas lygiagrečiam maršruto įgyvendinimui, nes įvairūs procesoriai gali būti paskirstyti skirtingų mažesnio lygmens problemų sprendimui.

2.1.2. Christofides algoritmas

Christofides algoritmas yra skirtas apytikslių keliaujančio pirklio problemos sprendimų paieškai, kai atstumai tarp maršruto skirtingų taškų sudaro metrinę erdvę (jie yra simetriški ir paklūsta trikampio nelygybei). Tai netikslus algoritmas, garantuojantis, kad jo sprendimai bus 3/2 optimalaus ilgio ir yra pavadintas Nicos Christofides, kuris jį paskelbė 1976 m., vardu. Nuo 2017 m. šis algoritmas laikomas geriausiu apytiksliu santykiu, įrodytu keliaujančio pirklio problemai spręsti bendrose metrinėse erdvėse, nors kai kuriems ypatingiems atvejams yra žinomi geresni algoritmai, leidžiantys tiksliau išspręsti problemą.[13]

Kaip ir Taillard's algoritmo atveju, Christofides algoritmo pagrindinė prielaida yra metrika - keliaujančio pirklio problema. Šis heuristinis algoritmas gali pateikti keliaujančio pirklio problemos sprendimą, kuris yra blogiausiu atveju 1.5 optimalaus sprendimo (algoritmo) kartotinis.

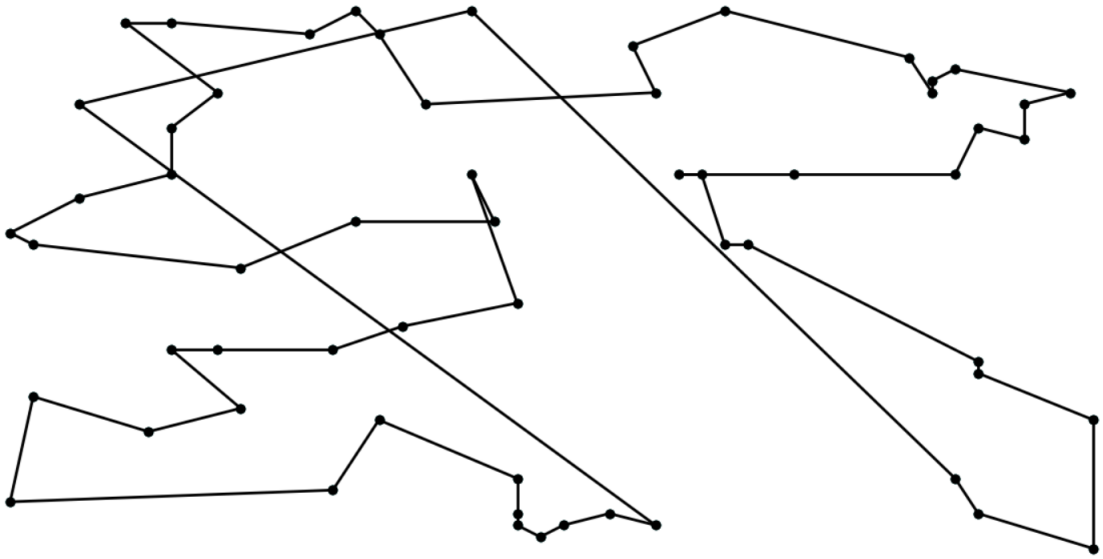
Keliaujančio pirklio problemos sprendimo procedūra, naudojant Christofides algoritmą:

1. Sukurti minimalų medį;
2. Rasti visas viršūnes, kurios turi nelyginį gretimų viršūnių skaičių;
3. Sukurti minimalų, tobulą suderinimą visame grafike nuo antrojo lygio viršūnių;
4. Pridėti papildomų briaunų nuo minimalaus svorio, idealiai atitinkančių minimalų medį;
5. Rasti Eulerijos kelią;

2.1.3. Artimiausio kaimyno algoritmas

Artimiausio kaimyno algoritmas yra euristinis algoritmas, savyje neturintis specifinių parametrų, kurie gali nulemti uždavinio sprendimą ir galutinį rezultatą. Algoritmas siekia pereiti per visus taškus ir surasti trumpiausią kelią nuo pradinio iki galutinio taško. Deja, algoritmas ne visada suranda trumpiausią kelią, nes algoritmas negali įvertinti visų taškų, esančių maršrute, iškart. Jei sekantis taškas yra labiau nutolęs nuo pradinio taško nei kiti, tačiau dar tolimesni maršruto taškai yra žymiai arčiau kelio pabaigos ir galutinis maršrutas gaunasi trumpesnis, šis taškas yra geriausias pasirinkimas. Tačiau, naudojant šį algoritmą, bus pasirenkamas tuo momentu arčiausiai esantis taškas, neatsižvelgiant į sekančius atstumus tarp taškų [9]. Algoritmas susideda iš kelių etapų:

1. Pasirenkamas pradinis taškas;
2. Surandami taškai, tiesiogiai prieinami iš pradinio taško;
3. Apskaičiuojami atstumai iki kiekvieno tiesiogiai prieinamo taško;
4. Parenkamas trumpiausias atstumas;
5. Taškas, nutolęs trumpiausiu atstumu pasirenkamas kaip sekantis pradinis taškas;
6. Kartojamas ciklas nuo 2 punkto tol, kol pasiekiamas paskutinis taškas;
7. Grįžtama į pradinį tašką, nuo kurio buvo pradėtas maršruto skaičiavimas.



4 pav. Artimiausio kaimyno algoritmas

Kadangi artimiausio kaimyno algoritmas neatsižvelgia į vėliau esančius atstumus tarp taškų, gaunami rezultatai gali reikšmingai skirtis, keičiantis pradiniam taškui. Siekiant ištaisyti šią spragą, buvo suformuluota tikslesnė algoritmo versija, pavadinta Pasikartojančio artimiausio kaimyno algoritmu. Ši algoritmo versija skiriasi tuo, jog pirmiausiai surandami trumpiausi maršrutai, imant vis kitą tašką kaip pradinį. Turint visų maršrutų ilgius, parenkamas trumpiausias iš jų, kartu nurodant parinktą tinkamiausią pradžios tašką.

Artimiausio kaimyno pseudo kodas[5]:

1. Išsaugokite visus treniruočių pavyzdžius;
2. Klasifikuoti naują pavyzdį x surandant mokymą surandant mokymą pavyzdys (xi,yi) , kuri yra arčiausiai artimiausia x pagal euklido atstumą:

$$\|\mathbf{x} - \mathbf{x}_i\| = \sqrt{\sum_j (x_j - x_{ij})^2}$$

5 pav. Artimiausio kaimyno psuedo kodas.

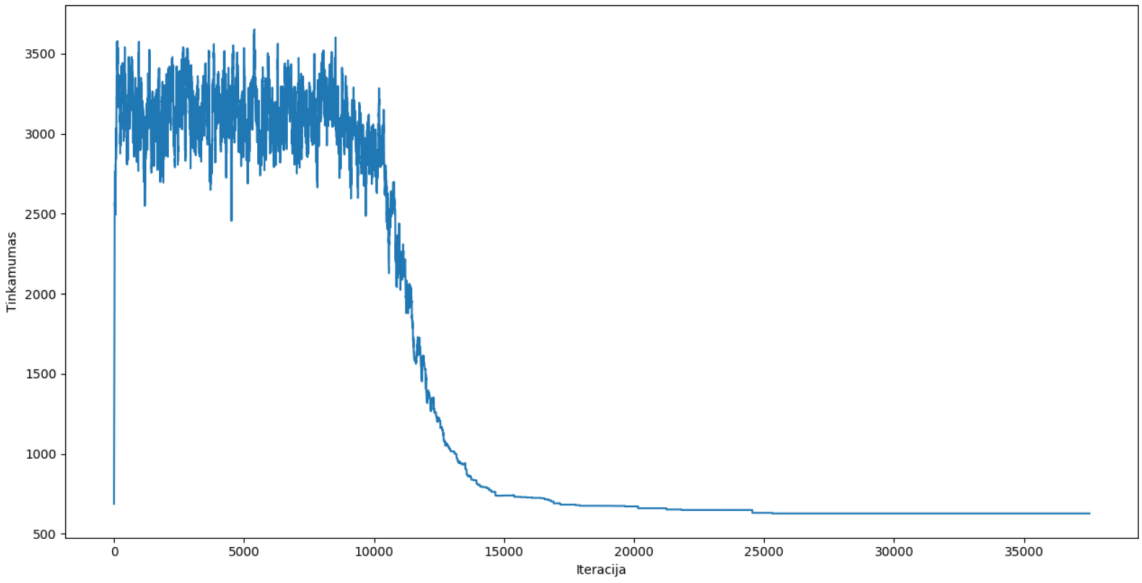
3. kvadratinis Euklido atstumas Efektyvumo triukas: kvadratinis Euklido atstumas suteikia tą patį atsakymą, tačiau vengia kvadrato suteikti tą patį atsakymą, tačiau vengia kvadrato šaknies skaičiavimas:

$$\|\mathbf{x} - \mathbf{x}_i\|^2 = \sum_j (x_j - x_{ij})^2$$

6 pav. Artimiausio kaimyno psuedo kodas su kvadratiniu euklidu.

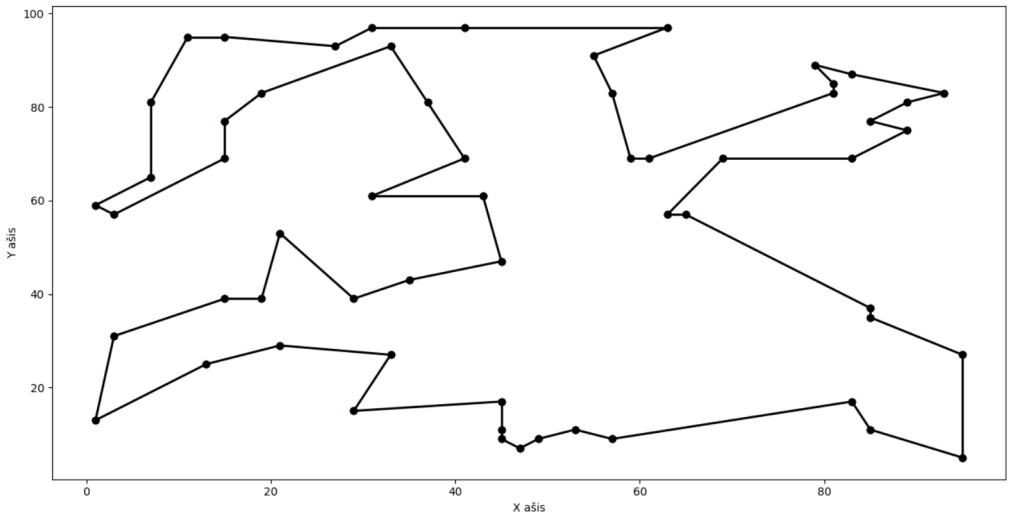
2.1.4. Modeliuojamo atkaitinimo algoritmas

Modeliuojamas atkaitinimas yra fizinio atkaitinimo proceso skaičiavimo modelis. Fizikiniame kontekste yra prielaida, kad metalai dažnai gali pasiekti didesnę galutinę energijos būseną, jei jie aušinimo metu periodiškai „perkaista". Panaudojus šį principą atskirai optimizacijai, maršrutų paieškos algoritmai dažnai gali gauti geresnių galutinių objektyvių kriterijų verčių, jei jiems leidžiama periodiškai priimti sprendimus su blogesnėmis kriterijų vertėmis. Modeliuojamas atkaitinimas yra meta-euristinis algoritmas, padedantis atrinkti tinkamiausius kintamuosius. Algoritmas plačiai naudojamas optimizacijos uždavinių sprendimui. Modeliuojamo atkaitinimo algoritmo veiksmingumą neretai gali nulemti algoritmo projektavimo ypatybės bei aplinkinių pogrupių bandomųjų potinklių generavimas. Taip pat algoritmas turi pagrindinius parametrus nuo kurių gali priklausyti galutinis rezultatas: minimali ir maksimali temperatūros ir aušinimo schema [11]. Pagrindinis modeliuojamo atkaitinimo algoritmas palaiko tiek būseną, tiek skaičiavimo temperatūrą, kuri iš pradžių yra didelė ir vykdymo metu sumažinama iki nulio. Sprendimų kokybė įvertinama naudojant specialią įvertinimo funkciją, kurios metu iš dviejų galimų variantų pasirenkama nauja būsena. Žemiau esančiame paveiksle pateiktas algoritmo funkcijos įvertinimo pavyzdys, išgautas parašius algoritmą ir atlikus pirmuosius bandymus su juo. Šiam algoritmui sukurti naudojaus kodu iš šios svetainės.[4]



7 pav. Modeliuojamo atkaitinimo algoritmo funkcijos įvertinimo pavyzdys

Labiausiai algoritmo veikimas priklauso nuo pradinės temperatūros ir ciklų kiekio. Modeliuojamo atkaitinimo algoritmas naudoja paieškos procesą, kuriame naudojami bandomieji pogrupiai, pabloginantys tikslinės funkcijos priimtinumą ir išreiškiantys jį tikimybe. Tikimybę kontroliuoja tikslinės funkcijos pablogėjimo mastas ir esama temperatūra. Temperatūra, vykdant algoritmą, periodiškai mažinama. Didinant temperatūrą ir sukimųsi kiekį, didėja tikimybė, kad rastas maršrutas bus optimalus. Kadangi algoritmo metu atliekama daug ciklų, rezultatas gaunamas per ganėtinai ilgą laikotarpį, todėl galima teigti, kad algoritmas veikia pakankamai lėtai. Žemiau esančiame paveiksle pateiktas gautas rezultatas pagal atliktus bandymus, naudojant modeliuojamo atkaitinimo algoritmą.



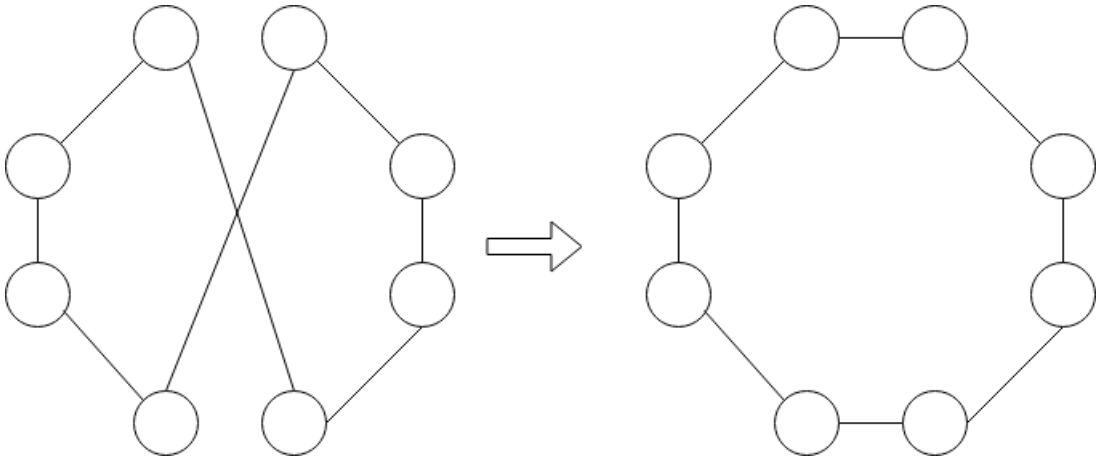
8 pav. Modeliuojamo atkaitinimo algoritmo rezultato pavyzdys

Svarbu paminėti, kad modeliuojamo atkaitinimo algoritmo parametrai paprastai yra naudotojo valdomi sprendimai, todėl išlieka nemaža tikimybė, kad algoritmo išgautas maršrutas priklausys nuo naudotojo priimtų sprendimų ir nustatytų parametrų.

2.1.5. Dviejų pasirinktųjų apsikeitimo algoritmas

Dviejų pasirinktųjų apsikeitimo algoritmas yra vienas iš geriausių ir populiariausių euristinių algoritmų, naudojamų maršrutų optimizacijos srityje. Dviejų pasirinktųjų apsikeitimo algoritmas atsirado iš paprasto pastebėjimo, kad, jei kelionės maršrute esantys keliai tarp dviejų taškų bent kartą susikerta, tada kelionės maršrutas gali būti nesunkiai sutrumpintas. Norint sutrumpinti tokį maršrutą, reikia pašalinti susikertančius maršrutus tarp dviejų taškų ir, vietoje jų, sudaryti naujas, nesusikertančias kryptis tarp dviejų taškų.

Algoritmo veikimo principas - panaikinti dvi briaunas ir jas sujungti kitokiu būdu tikintis, kad naujai sujungus briaunas bus gaunamas trumpesnis maršrutas. Jei panaikintų briaunų svorių suma yra didesnė už naujai sujungtų briaunų svorių sumą, tuomet naujai sujungtos briaunos visam laikui pakeičia seniau buvusias. Svarbu paminėti, kad visada yra tik vienas būdas perjungti lankus, kurie įeina į maršrutą, kad išliktų ciklas. Žemiau esančiame paveiksle pateikiamas iliuostruotas pavyzdys, kuriame vaizduojama, kaip sutrumpinti maršrutą, kai keli atstumai tarp taškų susikerta tarpusavyje [15]. Šiam algoritmui sukurti naudojaus kodu iš šios svetainės.[1]

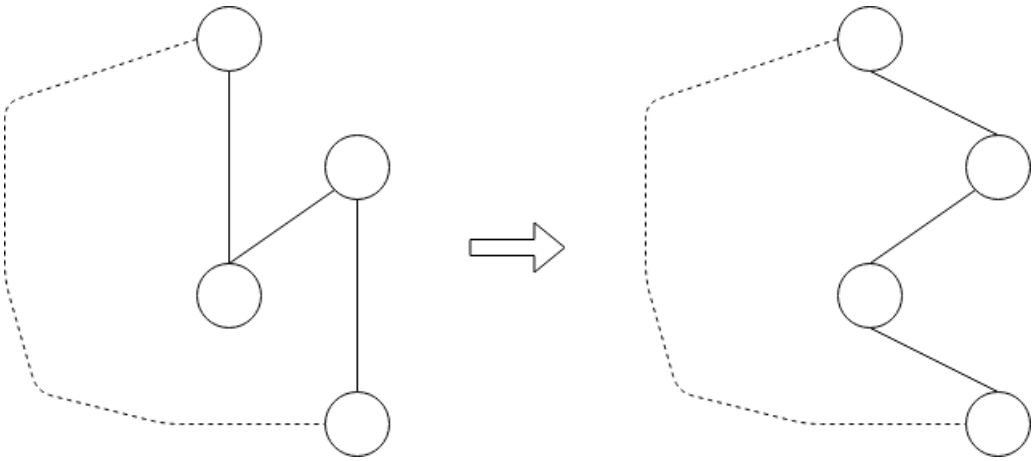


9 pav. Lankų apsikeitimas įstrižai

Dviejų pasirinktųjų apsikeitimo algoritmas susideda iš šių pagrindinių žingsnių:

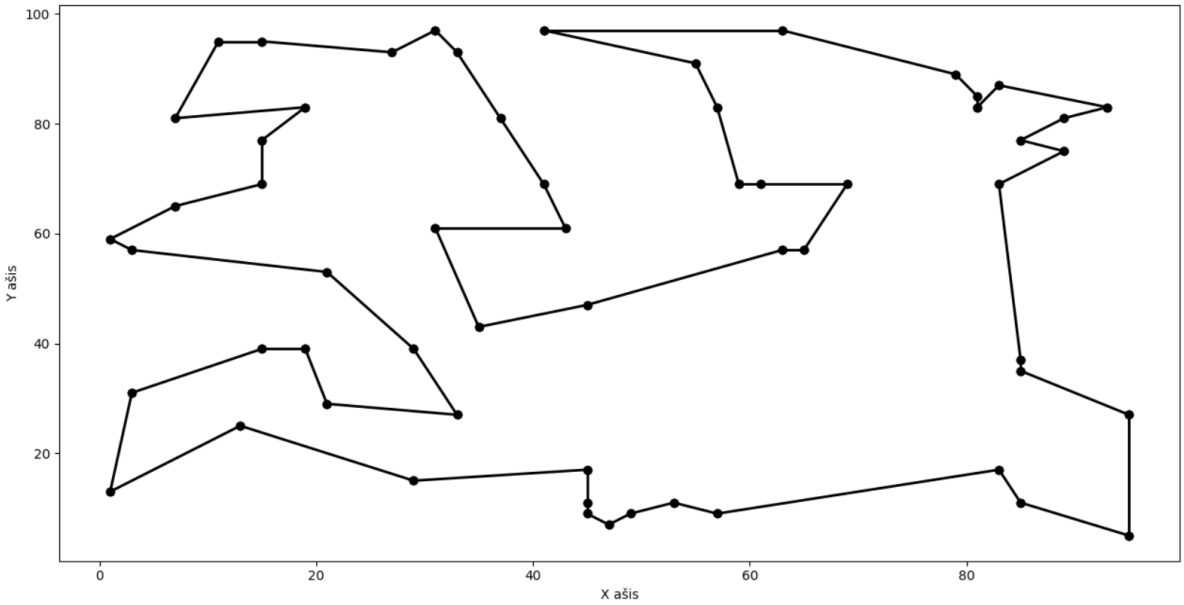
1. Surandama tobulintina lankų pora;
2. Surandamos visi galimi lankų poros sujungimo variantai;
3. Parenkama trumpiausio maršruto lankų pora;
4. Kartojamas 1 etapas tol, kol nebelieka porų, kurias būtų galima sukeisti.

Žemiau esančiame paveiksle pateiktas lankų apsikeitimo pavyzdys, trumpinant maršrutą.



10 pav. Lankų apsikeitimas sutrumpinant maršrutą

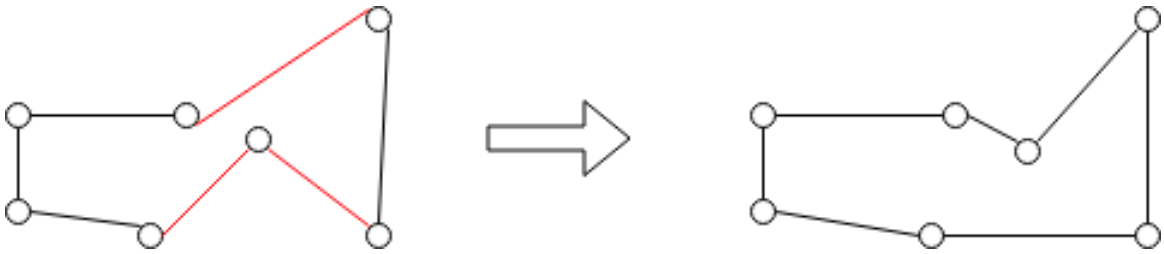
Šitas algoritmas yra $O(N^2)$ sunkumo ir greitai veikia tik su mažu duomenų kiekiu. Deja, jei optimizacijai pateikiamas didelės apimties maršrutas, algoritmo greitis spačiai lėtėja. Atkreiptinas dėmesys į tai, jog dviejų pasirinktųjų apsikeitimo algoritmas tik optimizuoja lankus tarp mazgų, todėl svarbu žinoti, kad algoritmas reikalaujaiš anksto sudaryto pilno maršruto bei gali jį tik optimizuoti.



11 pav. Gauto rezultato iš dviejų pasirinktųjų apsikeitimo algoritmo pavyzdys

2.1.6. Trijų pasirinktųjų apsikeitimo algoritmas

Trijų pasirinktųjų apsikeitimo algoritmas veikia panašiai kaip ir dviejų pasirinktųjų apsikeitimo algoritmas, bet šio algoritmo metu šalinami trys lankai. Trijų lankų judėjimas gali būti vertinamas ir kaip dviejų lankų. Žemiau esančiame paveiksle pateiktas trijų pasirinktųjų apsikeitimo veikimo principas.



12 pav. Trijų pasirinktųjų apsikeitimo įstrižai pavyzdys

Šis algoritmas yra $O(N^3)$ sunkumo ir pakankamai greitai veikia tik su labai mažu duomenų kiekiu. Kadangi trijų pasirinktųjų algoritme trumpinant maršrutą dalyvauja ne du, o trys lankai, dėl to dar labiau kenčia greitaveika (galimų lankų sujungimo kombinacijų yra žymiai daugiau nei jungiant du lankus). Dėl šios priežasties trijų pasirinktųjų apsikeitimo algoritmas geba per protingai trumpą laiką apdoroti dar mažiau duomenų nei dviejų pasirinktųjų apsikeitimo algoritmas [16]. Nepaisant to, trijų pasirinktųjų apsikeitimo algoritmas yra žymiai efektyvesnis už dviejų pasirinktųjų apsikeitimo algoritmą, nes vienu metu vertinama daugiau lankų, todėl optimizacija vykdoma platesniu mastu.

2.1.7. Skruzdžių kolonijų optimizavimo algoritmas

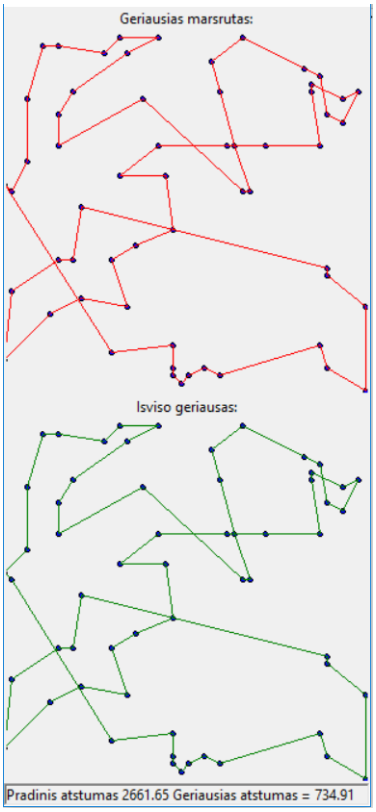
Skruzdžių kolonijų optimizavimas (toliau - ACO) yra spąstų žvalgybos technika ir metauristinis algoritmas, įkvėptas realių skruzdžių, ieškančių maisto, elgesio moduliu. Koloniją sudaro paprasti, nepriklausomi ir asinchroniniai tarpusavyje bendradarbiaujantys agentai, iešaktys tinkamo problemos sprendimo. ACO skruzdės yra stochastinės sprendimų kūrimo procedūros, kurios iteratyviai sukuria išsamius sprendimus mažesnių komponentų problemoms spręsti. Būtent tokio pobūdžio bendradarbiavimas kolonijoje suteikia algoritmui išskirtinumą.[18]

ACO algoritmai yra priskiriami sparčiojo intelekto algoritmams. Šį algoritmą įkvėpė realių skruzdžių elgesys, ieškantis tinkamų ir gausių maisto šaltinių. Pirmą kartą algoritmas buvo pristatytas dešimtojo dešimtmečio pradžioje. Iš pradžių jis buvo naudojamas skaičiavimo problemoms, susijusioms su grafikų judėjimu, išspręsti. ACO metauristinis algoritmas buvo apibrėžtas taip, kad būtų sukurtas bendras naujos algoritmų klasės apibūdinimas ir pamatinė sistema, skirta kurti naujiems ACO algoritmų atvejams. ACO algoritmai tapo gana populiarūs - yra specialios tinklavietės, kurios jungia mokslininkų ir tyrėjų bendruomenę, kurie nuolat dalinasi naujausia informacija, įvykiais, diskutuoja apie ACO algoritmo teorines savybes. Taip pat, du kartus per metus rengiama konferencija „Skruzdžių kolonija“. Apibendrinant galima teigti, kad ACO metauristinė veikla jau daugelį metų yra gerai žinoma ir diskutuotina mokslinių tyrimų tema.

ACO algoritmų taikymo sritis yra labai plati. Šie algoritmai yra taikomi dešimtims skirtingų NP klasės skaičiavimo problemoms, kurios dažnai pasiekia prastus sprendimo rezultatus, spręsti. Pirmasis ACO algoritmas yra „Ant“ sistema - ji buvo pritaikyta keliaujančio pirklio problemai. Nuo tada ACO algoritmai taikomi optimalaus maršruto parinkimo ir planavimo uždaviniams, problemų ir jų pogrupių, mašinų mokymosi, baltymų sulankstymo, DNR sekos nustatymo ir paketų nukreipimo telekomunikacijų tinkluose uždaviniams spręsti. ACO algoritmai gali būti ypač perspektyvūs netinkamos struktūros arba labai dinamiškų problemų sprendimui. Bendrai, ACO algoritmų klasė yra perspektyvios ir veiksmingos priemonės, padedančios išspręsti sunkias skaičiavimo problemas, dažniausiai priskiriamas NP klasės uždaviniams.

2.1.8. Genetinis algoritmas

Genetinis algoritmas yra euristinis algoritmas, imituojantis evoliucijos biologijos principus, ieškant sprendimų sudėtingoms problemoms. Genetinių algoritmų veikimo principas yra pagrįstas gyvosios gamtos evoliucijos imitavimu, todėl pagrindinės sąvokos susijusios su evoliucija: individas, populiacija ir individo tinkamumas. Individas yra elementarus, pats smulkiausias vienetas. Populiacija susideda iš daugybės individų. Individo tinkamumu laikomas tam tikras matas, pagal kurį vertinamas individas dėl savo tam tikrų savybių. Šie specifiniai parametrai gali paveikti gautus rezultatus, jei vienas iš jų nustato pradinę populiaciją. Optimizacijos atveju individas atitinka uždavinio sprendinį, populiacija laikoma sprendinių poaibis, o individo tinkamumą atitinka tikslinės funkcijos reikšmė [7]. Šiam algoritmui sukurti naudojaus kodu iš šios svetainės.[3]



13 pav. Genetinio algoritmo pavyzdys

Genetiniai algoritmai susideda iš šių pagrindinių etapų:

1. Sudaroma pradinė populiacija;
2. Vykdomas populiacijos pogrupių atrinkimas;
3. Atrinkti populiacijos pogrupiai sukryžminami;
4. Sukryžminti pogrupiai mutuoja;
5. Atnaujinama populiacija, prijungiant mutavusį pogrupį;
6. Sprendžiama, ar gauta populiacija optimali. Jeigu ne, ciklas kartojamas nuo 2 punkto.

Sprendžiant optimizacijos uždavinius, paprastai sprendinio optimizavimas vykdomas po kryžminimo operacijos, nors galima optimizuoti ir kituose genetinio algoritmo etapuose. Dažnai euristinės procedūros naudojamos kaip aukštos kokybės pradinių sprendinių populiacijų generatoriai. Tokiu atveju, genetiniai algoritmai gauna jau optimizuotą pradinę populiaciją - dėl šios priežasties optimalių sprendinių paieška tampa daug efektyvesnė nei jau turimų neefektyvių sprendinių tobulinimas.

2.2. Ateities tyrimų kryptys

Ateities darbams lieka neišspręstos dvi pagrindinės problemos. Pirmoji yra tiesiogiai susijusi su techniniu algoritmo įgyvendinimu - parašyti greičiau veikiantį hibridinį algoritmą skirtą spręsti (CVRP) problemai, nepriklausomai nuo turimų kompiuterio išteklių. Antroji - gauto maršruto atvaizdavimas žemėlapyje. Galutinai nėra aišku, kaip teisingai atvaizduoti gautus rezultatus žemėlapyje. Apibendrinant, pagrinidniai ateities darbai ir jų planas, gairės tolimesniems darbams:

- Parašyti hibridinį algoritmą skirtą spręsti (CVRP) problemai, naudojantį mažiau kompiuterio išteklių.
- Sukurti programą, kurioje būtų techniškai įgyvendinamas šis hibridinis algoritmas.
- Pritaikyti algoritmo rezultatus tvarkingam maršruto atvaizdavimui žemėlapyje.

3. Tyrimas

Analizuojant TSP ir VRP problemas pastebėta, kad yra sukurta labai daug algoritmų, kurie pilnai ar bent dalinai sprendžia šias problemas. Darbe išanalizuoti ir aprašyti tik keli, vieni iš dažniausiai naudojamų algoritmų, sprendžiantys TSP ir VRP problemą. Esamų sprendimų analizė padeda suvokti esminius algoritmų trūkumus, kuriuos bus siekiama eliminuoti. Šiame skyriuje aprašytas atliktas tyrimas, naudojant 3.8 apžvelgtus bei aprašytus algoritmus. Pirmiausiai apžvelgiami techniniai sprendimai bei turimi ištekliai, kuriais naudojantis bus vykdomi algoritmai. Taip pat pateikiamas tyrimo planas, kurio laikantis bus siekiama ištirti kiekvieno algoritmo trūkumus ir privalumus - ši informacija bus naudojama hibridinio algoritmo formavimo etape.

3.1. Tyrimo planas

Kiekvienas aprašytas algoritmas bus tiriamas atskirai, keičiant jų parametrus ir naudojant skirtingus bandomuosius duomenis. Tokiu būdu bus siekiama išsiaiškinti algoritmų parametrizavimo svarbą ir parametrų daromą įtaką algoritmo rezultatams. Atlikus algoritmų veiksmingumo tyrimą, bus formuojamas naujas hibridinis algoritmas. Tyrimo planas:

1. Sukurti duomenų struktūrą, kuri bus naudojama algoritmų efektyvumo tyrimui;
2. Parašyti programinį algoritmų kodą;
3. Išbandyti analizuotus algoritmus naudojant tuos pačius bandomuosius duomenis ir skirtingus algoritmų parametrus;
4. Palyginti algoritmų veiklos trukmes ir priimtų sprendimų kokybes;
5. Išrinkti geriausią algoritmą ir visų išbandytų, kuriuo bus remiamasi formuojant hibridinį algoritmą.

3.2. Techninė informacija

Algoritmų greičiui daro didelę įtaką turimi ištekliai bei techniniai parametrai, todėl pakankamai svarbu išskirti pagrindinius techninius parametrus bei nustatyti turimus išteklius, kad vėliau būtų galima objektyviau įvertinti gautus tyrimo rezultats. Šiam tyrimui bus reikalingi ištekliai: kompiuteris ir bandomieji maršrutų taškų duomenys. Algoritmai parašyti „Python" programavimo kalba ir bus bandomi naudojant „Windows 10" operacinę sistemą bei 64-bit kompiuterio architektūrą. Tyrimo atlikimo metu kompiuterio laikinoji atmintis pajungta. Pilna sisteminė informacija:

- Operacinė sistema - Windows 10;
- Kompiuterio architektūra - 64 bit;
- Programos interpretatorius - Python 3.6;
- Operatyvinė atmintis (RAM) - 8 GB;
- Procesorius - Intel i7-6700HQ CPU, 2.6 GHz;
- Vaizdo plokštė - Nvidia GTX950M;

3.3. Duomenų struktūra tyrimiems algoritmams

Taip pat labai svarbu tinkamai parengti bandomuosius duomenis, nes jie skirti algoritmų efektyvumui patikrinti. Bandymams su skirtingais algoritmais atlikti buvo suformuoti 60 taškų duomenys dvimatėje erdvėje, nurodytos konkrečios taškų koordinatės. Taško svoris yra skaičiuojamas pagal atstumą nuo taško A iki taško B. Žemiau pateikta bandomųjų duomenų struktūra:

```
<node id="1" type="0">
    <cx>27.0</cx>
    <cy>93.0</cy>
</node>
<node id="2" type="1">
    <cx>33.0</cx>
    <cy>27.0</cy>
</node>
<node id="3" type="1">
    <cx>29.0</cx>
    <cy>39.0</cy>
</node>
```

14 pav. Bandomųjų duomenų struktūra

3.4. Maršrutų optimizavimo algoritmų bandymo rezultatai

Atliekant algoritmų efektyvumo tyrimą buvo analizuojami algoritmai pagal kiekvieno jų savybes ir parametrus. Tyrimo metu buvo bandoma keisti kiekvieno algoritmo parametrus ir savybes, siekiant išsiaiškinti algoritmų parametrizavimo svarbą ir parametrų daromą įtaką algoritmo rezultatams. Algoritmai parašyti „Python" programavimo kalba ir bandomi naudojant „Windows 10 64-bit" architektūrą. Tyrimo metu buvo naudojama laikinoji atmintis. Tyrimui atlikti buvo suformuoti 60 skirtingų taškų duomenys, naudojant x ir y ašių koordinates. Žemiau esančiame sąraše pateikiami pagrindiniai algoritmų parametrai, naudoti bandymams:

1. Modeliuojamo atkaitinimo parametrai:

    1 bandymas: Pradinė temperatūra laipsniais - 200000;
    Vėsimo temperatūra laipsniais - 0.9999;
    Sukimųsi kartai - 200000.

    2 bandymas: Pradinė temperatūra laipsniais - 20000;
    Vėsimo temperatūra laipsniais - 0.999;
    Sukimųsi kartai - 20000.

    3 bandymas: Pradinė temperatūra laipsniais - 2000;
    Vėsimo temperatūra laipsniais - 0.99;
    Sukimųsi kartai - 2000.

2. Dviejų pasirinktųjų apsikeitimo algoritme naudojami pradiniai taškai:

    1 bandymas: Pradinis taškas - 55.

    2 bandymas: Pradinis taškas - 40.

    3 bandymas: Pradinis taškas - 54.

3. Genetinio algoritmo parametrai:

    1 bandymas: Populiacija - 10;
    Generacija - 40.

    2 bandymas: Populiacija - 100;
    Generacija - 80.

    3 bandymas: Populiacija - 1000;
    Generacija - 120.

Tyrimo metu buvo analizuojami keturi maršrutų optimizavimo algoritmai: artimiausio kaimyno, dviejų pasirinktųjų apsikeitimo, modeliuojamo atkaitinimo bei genetinis. Visi algoritmai buvo

parašyti „Python" programavimo kalba bei išbandyti, naudojant suformuotus bandomuosius duomenis. Pagrindiniai stebimi kriterijai tyrimo metu buvo optimalaus maršruto atstumo gavimas bei laikas, per kurį algoritmas apskaičiavo trumpiausią atstumą.
Žemiau esančioje lentelėje pateikiami trijų bandymų rezultatai. Kiekvienas bandymas buvo atliekamas su visais algoritmais ir tais pačiais bandomaisiais duomenimis, kurie buvo keičiami kiekvienam bandymui.

1 lentelė. Bandymo rezultatai

Bandymas	Algoritmas	Atstumas	Laikas
1	Artimiausio kaimyno	770.73	0.01
1	Dviejų pasirinktųjų apsikeitimo	617.94	3.84
1	Modeliuojamo atkaitinimo	583.18	4.54
1	Genetinis	2446.99	0.7
2	Artimiausio kaimyno	676.32	0.01
2	Dviejų pasirinktųjų apsikeitimo	632.52	3.4
2	Modeliuojamo atkaitinimo	615.68	0.47
2	Genetinis	1234.53	5.9
3	Artimiausio kaimyno	743.08	0.01
3	Dviejų pasirinktųjų apsikeitimo	611.65	2.68
3	Modeliuojamo atkaitinimo	761.43	0.05
3	Genetinis	675.56	73.5

Atlikus bandymus, gauti rezultatai buvo vertinami lyginamuoju būdu. Siekiant objektyvumo, algoritmai vertinami suteikiant jiems balus už pasiektą trumpiausią atstumą bei veikimo trukmę, lyginami tarp skirtingų algoritmų. Pirmiausiai, algoritmai buvo lyginami tarpusavyje pagal apskaičiuotą trumpiausią maršruto atstumą. Žemiau esančioje lentelėje pateikiami palyginimo tarp skirtingų algoritmų rezultatai bei balų skaičius, skiriamas kiekvienam algoritmui pagal gautus rezultatus.

2 lentelė. Algoritmų palyginimas pagal rastą trumpiausią atstumą

Vieta	Algoritmas	Atstumas	Balai
1	Modeliuojamo atkaitinimo	583.18	4
2	Dviejų pasirinktųjų apsikeitimo	611.65	3
3	Genetinis	675.56	2
4	Artimiausio kaimyno	676.32	1

Maršrutų optimizavim algoritmai taip pat buvo lyginami tarpusavyje greitaveikos atžvilgiu. Žemiau esančioje lentelėje pateikiamas algoritmų greitaveikos palyginimas bei balų skaičius, skiriamas kiekvienam algoritmui pagal gautus rezultatus. Algoritmų greitaveika buvo apskaičiuota išvedant matematinį vidurkį iš visų bandymų trukmių.

3 lentelė. Algoritmų palyginimas pagal vidutinę veikimo trukmę

Vieta	Algoritmas	Laikas	Balai
1	Artimiausio kaimyno	0.01	4
2	Modeliuojamo atkaitinimo	1.68	3
3	Dviejų pasirinktųjų apsikeitimo	3.30	2
4	Genetinis	26.70	1

Galutiniam algoritmų vertinimui buvo susumuojami balai, skirti už greitaveiką bei trumpiausio maršruto radimą. Žemiau esančioje lentelėje pateikiama suminė balų lentelė, kurioje matomas greitaveikos ir sprendimo kokybės santykinis palyginimas tarp skirtingų maršrutų optimizavimo algoritmų.

4 lentelė. Suminis algoritmų palyginimas

Vieta	Algoritmas	Balų suma
1	Modeliuojamo atkaitinimo	7
2-3	Dviejų pasirinktųjų apsikeitimo	5
2-3	Artimiausio kaimyno	5
4	Genetinis	3

Taigi, galima teigti, jog geriausias greitaveikos bei sprendimo kokybės santykis pastebėtas modeliuojamo atkaitinimo algoritme. Kadangi atlikto bandymo rezultatais nustatyta, jog šis algoritmas geriausiai atitinka keliamus tikslus, tolimesniame darbe planuojama remtis šio algoritmo pagrindais.

3.5. Algoritmų tyrimo išvados ir rezultatai

- Bandymo metu nustatyta, jog greičiausiai veikiantis algoritmas - artimiausio kaimyno. Nepaisant to, algoritmo metu surasto maršruto kokybė yra prasčiausia, todėl remtis šiuo algoritmu nepatartina, nes algoritmo sprendimo kokybė yra prasčiausia, lyginant su kitais bandytais algoritmais.
- Efektyviausias algoritmas, atsižvelgiant į sprendimo kokybės ir algoritmo veikimo trukmės santykį, yra genetinis. Hibridinio algoritmo kūrimo metu buvo remiamasi šio algoritmo pagrindiniais principais, nes lyginant su kitų algoritmų rezultatais, genetinio algoritmo sprendimo kokybė geriausia.
- Maršrutų optimizavimo algoritmų analizė bei bandymai padeda suvokti algoritmų neigiamus aspektus ir trūkumus. Ši informacija yra labai svarbi naujo algoritmo kūrimui, nes iš anksto yra žinomos pagrindinės problemos, kurias siekiama išspręsti.

3.6. Hibridinis algoritmas

Hibridinio algoritmo kūrimui buvo pasirinkta dalykinė sritis, siekiant tikslingai įvertinti visus pašalinius veiksmus, galinčius daryti įtaką algoritmo efektyvumui. Pasirinkta dalykinė sritis - maisto banke paaukotų produktų išvežiojimas į maisto dalijimo punktus. Sunkvežimiai aplanko šimtus maisto dalijimo punktų, kuriuos turi pasiekti paaukoti maisto produktai.

Tokio pobūdžio problemos sprendimui yra sudėtinga rasti tikslų algoritmą, kuris apskaičiuotų maršrutą per protingą terminą. Yra keletas sudėtingų sąlygų, kurios apsunkina problemos sprendimą, tokių kaip ribota sunkvežimio talpa, laiko langai tarp iškrovimo ir pakrovimo ir maršruto ilgio ribojimai. Nepaisant to, tokio pobūdžio problemos sprendimui yra sėkmingai pritaikytų heuristinių algoritmų, kurie yra pagrįsti įvairių draudimų ir ribojimų paieška. Šių ribojimų paieškai moksliniuose tyrimuose nuolat skiriama daug dėmesio, kadangi tai yra pagrindinės priežastys, apsunkinančios problemos sprendimo radimą (Chiang W.C. ir Russell R.A., 1997). Lee at al. (2003) palygino ribojimų ir konfliktų paiešką su kitomis konkuruojančiomis heuristikomis ir nustatė, kad ribojimų paieškos metodas geba pasiūlyti geresnius sprendimus nei kiti algoritmai. Atsižvelgiant į šio tyrimo rezultatus, darbe aprašomas tyrimas yra orientuotas į vietinių paieškos metodų taikymą, sustiprintą ribojimų ir konfliktų paieška. Tokiu būdu siekiama optimaliai išspręsti CVRP tipo problemą.

Kuriamo hibridinio algoritmo pagrindu buvo pasirinktas genetinis algoritmas dėl savo dinamiškumo. Pats algoritmas buvo pritaikytas kiekvieno sunkvežimio talpai bei turimų sunkvežimių kiekiui. Algoritmas geba atrinkti transporto priemones ir priskirti joms optimalius maršrutus. Maršruto optimizavimui buvo atrinkti geriausi genetinio algoritmo parametrai.

Sukurtas algoritmas buvo išsamiai ištestuotas su skirtingais parametrais, o efektyviausi parametrai, nustatyti pagal gautus bandymų rezultatus, buvo laikomi geriausiais - juos nuspręsta naudoti toliau. Šiam algoritmui sukurti naudojaus kodu iš šios svetainės.[2]

3.6.1. Algoritmo įvestys ir išvestys

Algoritmui reikalingų duomenų surinkimui buvo naudojamos „OSRM" žemėlapio koordinatės. Žemiau esančiame 15 paveiksle pateikta maršrūto taškų koordinačių struktūra.

```
<node id="1" type="0">
    <cx>55.2533635</cx>
    <cy>22.2903395</cy>
</node>
<node id="2" type="1">
    <cx>55.7344985</cx>
    <cy>24.3578055</cy>
</node>
<node id="3" type="1">
    <cx>54.8982139</cx>
    <cy>23.9044817</cy>
</node>
<node id="4" type="1">
    <cx>55.7127529</cx>
    <cy>21.1350469</cy>
</node>
<node id="5" type="1">
    <cx>54.6870458</cx>
    <cy>25.2829111</cy>
</node>
```

15 pav. Maršrūto taškų koordinatės

Visų koordinačių struktūrą sudaro:

- „node" - elementas, kurio parametrai nusako taško koordinates. Parametras „id" identifikuoja tašką ir užtikrina jo unikalumą, parametras „type" gali būti: 0 - nurodo, kad tai pradinis maršruto taškas; 1 - nurodo, kad tai tarpinis maršruto taškas.
- „cx" - koordinatės platuma.
- „cy" - koordinatės ilguma.

Kiekviena transporto priemonė taip pat aprašoma struktūrizuotai. Žemiau esančiame 16 paveiksle pateiktas pavyzdinis transporto priemonės aprašymas.

```
<vehicle_profile type="0">
    <departure_node>1</departure_node>
    <arrival_node>1</arrival_node>
    <capacity>4.0</capacity>
</vehicle_profile>
```

16 pav. Transporto priemonės aprašymo struktūra

Visų transporto priemonių struktūrą sudaro:

- „vehicle profile" - identifikuoja transporto priemonės aprašymo pradžią bei pabaigą. Parametras „type" susieja transporto priemonę su koordinate, turinčia tokį patį tipą.
- „departure node" - pradinio maršruto taško identifikacinis numeris.
- „arrival node"- galutinio maršruto taško identifikacinis numeris.
- „capacity" - transporto priemonės maksimali talpa.

Kiekviena koordinatė algoritmo metu gauna užklausas. Koordinačių užklausos yra aprašomos elemente „requests”. Žemiau esančiame 17 paveiksle pateikiamas pavyzdinis koordinačių užklausų sąrašas.

```
<requests>
    <request id="1" node="1">
        <quantity>0.0</quantity>
    </request>
    <request id="2" node="2">
        <quantity>1.0</quantity>
    </request>
    <request id="3" node="3">
        <quantity>2.0</quantity>
    </request>
    <request id="4" node="4">
        <quantity>1.0</quantity>
    </request>
    <request id="5" node="5">
        <quantity>1.0</quantity>
    </request>
</requests>
```

17 pav. Koordinačių užklausų struktūra

Kiekvieną užklausą sudaro:

- „request” - nurodo užklausos koordinatei pradžią ir pabaigą. Parametrai „id” - identifikuoja užklausą ir užtikrina jos unikalumą; „node” - nurodo, į kurią koordinatę užklausa yra nukreipta.
- „quantity” - nurodo turimą transporto priemonės svorį konkrečiai koordinatei.

Siekiant automatizuoto algoritmo testavimo, buvo parašyta programa, paremta sukurtu algoritmu, ieškanti optimalaus maršruto, ribojant transporto priemonių kiekį, maršrutų ilgius ir transporto priemonių talpas.

3.6.2. Algoritmo veikimo principai

Pagrindinis programos funkcija yra genetininio algoritmo įgyvendinimas, kurio rezultatas pateikiamas nepilname grafe. Nepilnam grafui skaičiuoti atstumus tarp taškų yra pritaikytas OSRM (angl. Open Source Route Machine) serveris ir žemėlapis. Pats algoritmas yra specializuotas CVRP problemai spręsti.

Genetinį algoritmą, kuris gali išspręsti daugelį praktinių problemų, sudaro trys pagrindinės operacijos:

- Reproduckija.
- Kryžminimas.
- Mutacija.

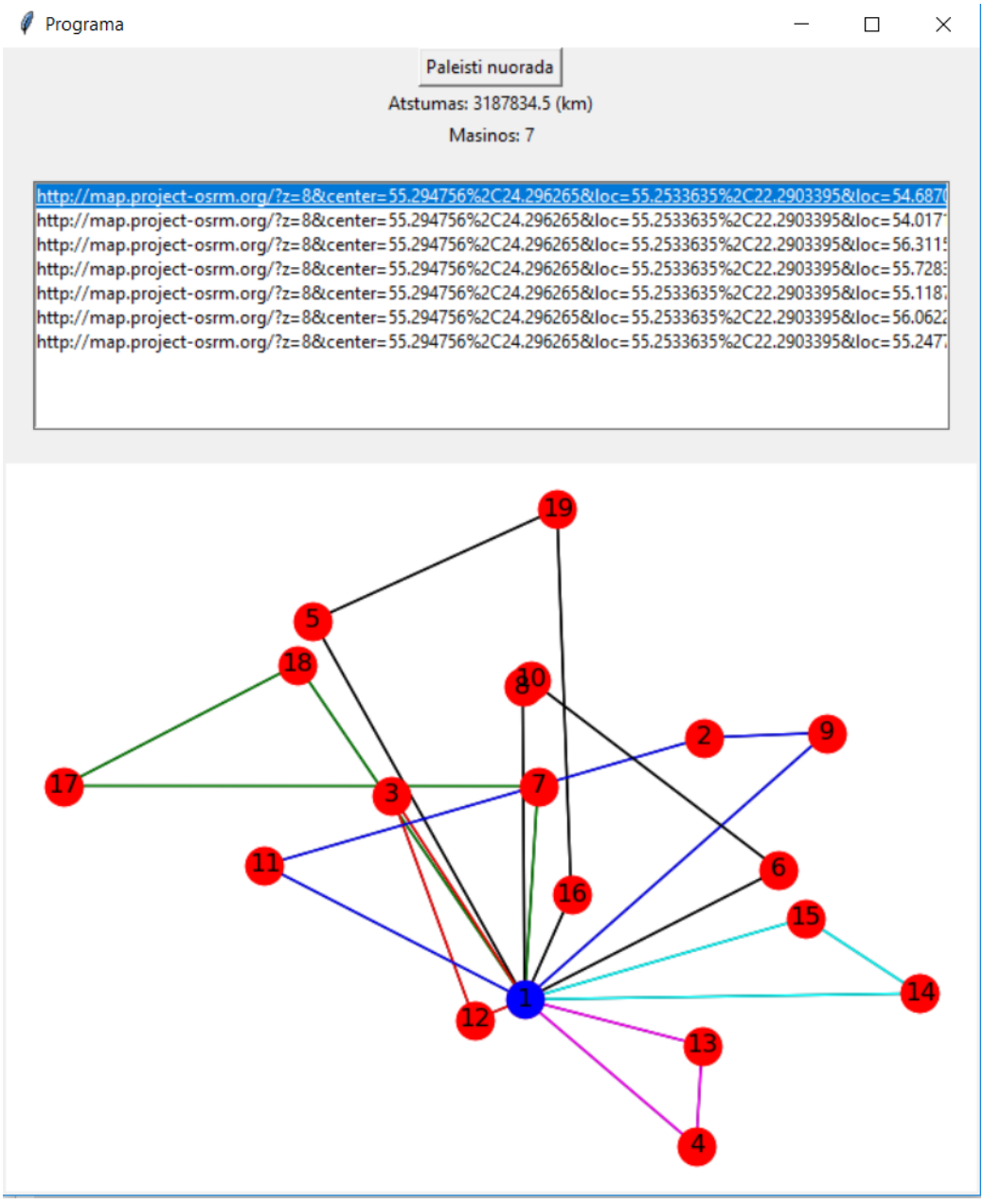
Reprodukcijos proceso metu siekiama, kad pakankamai tinkama genetinė informacija būtų išsaugota ir išliktų kitai kartai. Proceso metu kiekvienai populiacijos eilutei priskiriama tinkamumo reikšmė ir nustatomas tinkamumas objekto funkcijai. Ši reikšmė yra tikimybė, kad populiacijos eilutė bus pasirinkta tėvine naujos kartos atkūrimo procese.

Genetinio kryžminimas yra procesas, suskirstytas į segmentus, kurie keičiasi su vienu segmentu ir su kita populiacijos eilute. Šiais procesais sukuriamos dvi naujos populiacijos eilutės, kurios skiriasi nuo tų, kurios buvo sukurtos anksčiau. Būtina paminėti, kad iš eilės perbrauktų styginių pasirinkimas, pasirinktas ankstesniame reprodukcijos procese, yra atsitiktinis. Problemų optimizavimo požiūriu jis yra lygus parametrų erdvės ploto naudojimui.

Mutacijos procesas pasireiškia nedideliu individų genetinės eilutės pokyčiu. Dirbtinių genetinių stygų atveju mutacija yra lygi individo kodo elementariosios dalies (alelio) pokyčiui. Mutacija vyksta skirtingomis charakteristikomis, todėl individų charakteristikos, buvusios prieš mutacijos procesą, gali nebelikti populiacijoje arba gali atsirasti visai naujų, anksčiau nebuvusių charakteristikų. Problemų optimizavimo požiūriu mutacija yra lygi paieškos srities keitimui parametrų erdvėje.

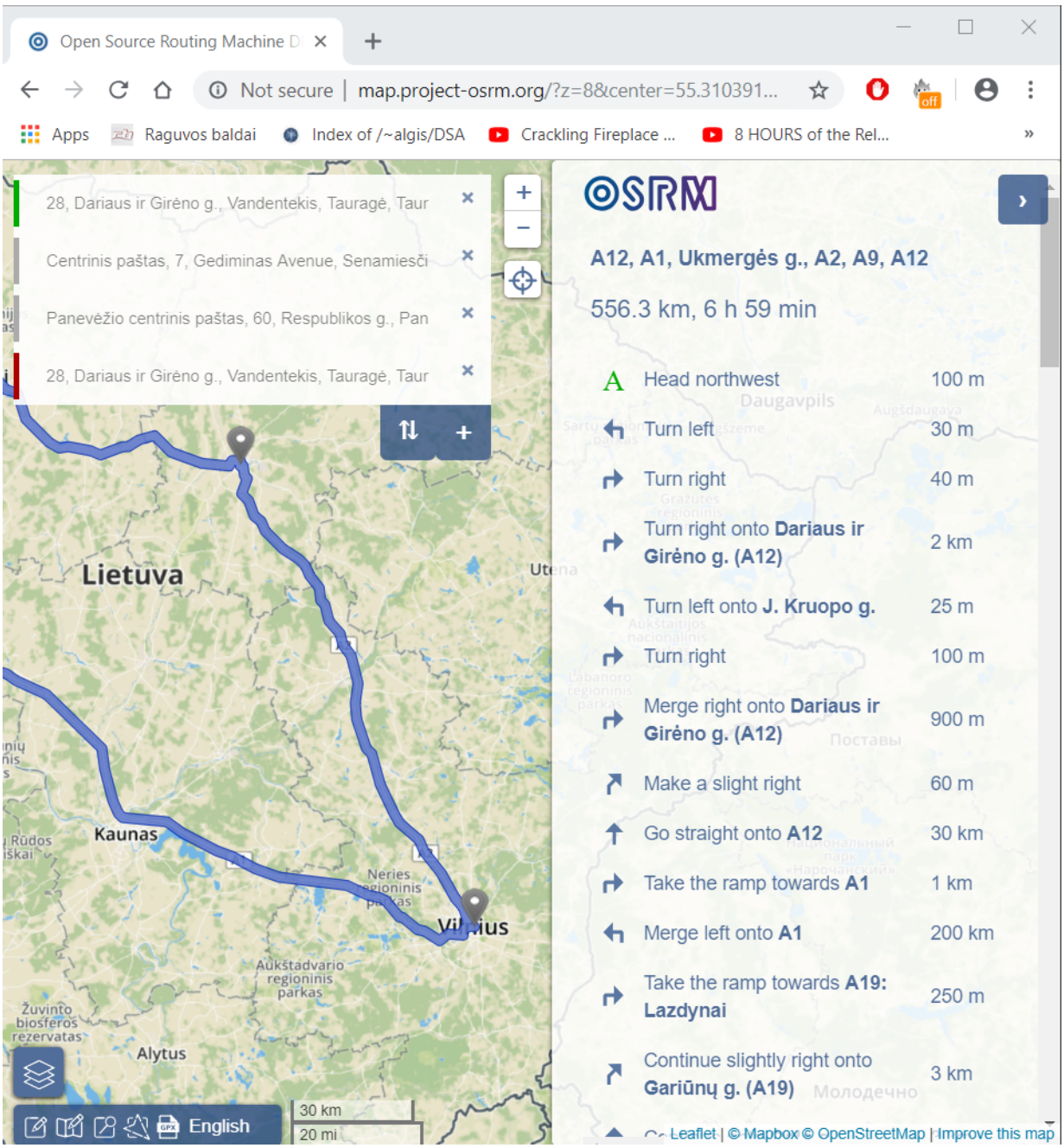
3.7. Algoritmo įgyvendinimas - programos kūrimas

Programoje pateikiamos įrašų nuorodos, kurias paspaudus atsidaroma internetinė svetainė su konkrečiu maršrutu. Kiekviena nuoroda vaizduoja po vieną maršrutą, naudotą nagrinėjamos problemos sprendimui. Bandymų rezultatai pateikiami grafuose. Deja, parašyta programa veikia pakankamai lėtai, kadangi dviejų taškų koordinatės yra skaičiuojamos pateikiant užklausą į OSRM serverį, kuris riboja užklausų kiekį kas 3-5 sekundes. 18 paveiksle pateikiama visų maršrutų vizualizacija pilname grafe.



18 pav. Visų maršrutų bendra vizualizacija parašytoje programoje ir nuoradų sąrašas skirtas vieno maršruto žemėlapui atidarymui.

Kaip matoma 18 paveiksle, programoje gautas rezultatas yra vizualizuojamas pilname grafe. Atidarius programoje nuorodą maršrutas vizualizuojamas nebe pilname, bet komplikuotame grafe, kuriame akivaizdžiai matomi įvairūs trukdžiai. Maršruto vizualizacijoje taip pat pateikiamas galutinis maršruto atstumas ir jam įveikti reikalingų transporto priemonių kiekis.



19 pav. Maršruto vizualizacija atidarius programoje pateiktą nuorodą

Atidarius nuorodą programoje atvaizduojamas vienas iš daugelio sugeneruotų maršrutų (žr. 19 paveikslą). Atidarytame puslapyje yra naudojamas OSRM žemėlapis, kuriame pateikiami maršruto taškai, bendras maršruto ilgis ir atstumai tarp atskirų maršruto taškų.

3.8. Programos testavimas

Testavimo metu svarbu išskirti pagrindinius techninius parametrus bei nustatyti turimus išteklius, kad būtų galima objektyviau įvertinti gautus rezultatus. Programos testavimui reikalingi ištekliai: kompiuteris ir bandomieji maršrutų taškų duomenys. Algoritmai, parašyti „Python" programavimo kalba, bandomi naudojant „Windows 10" operacinę sistemą bei 64-bit kompiuterio architektūrą. Bandymo metu kompiuterio laikinoji atmintis įjungta. Pilna sisteminė informacija:

- Operacinė sistema - Windows 10;
- Kompiuterio architektūra - 64 bit;
- Programos interpretatorius - Python 3.6;
- Operatyvinė atmintis (RAM) - 8 GB;
- Procesorius - Intel i7-6700HQ CPU, 2.6 GHz;
- Vaizdo plokštė - Nvidia GTX950M;
- OSRM - žemėlapis;

Bandymo metu buvo parinkta 20 skirtingų taškų visoje Lietuvoje. Kiekvieno bandymo metu buvo keičiami genetinio algoritmo parametrai. Geriausias rezultatas gautas antruoju bandymu.

5 lentelė. Bandymo rezultatai

Nr.	Kartų sk.	Mašinų sk.	Mutacijos koficientas	Kryžminis koficientas	Populiacija	Atstumas
1	10	7	0.2	0.5	50	3 187 834.50 km
2	25	7	0.2	0.5	100	2 870 007.70 km
3	50	7	0.2	0.5	200	3 217 361.50 km

Išvados ir rekomendacijos

- Šis CVRP tipo problemos sprendimo algoritmas yra ypač tinkamas įvairioms logistikos problemoms spręsti, susijusioms su skirtingomis transporto priemonėmis,pvz. lėktuvais ar automobiliais. Transporto priemonės gali turėti talpinamo krovinio ribojimus, į kuriuos atsižvelgiant skaičiuojami galimi maršrutai.

- Šis sukurtas įrankis yra pritaikytas transporto žemėlapui nepilnam grafui. Taip pat, algoritmas pirmiausiai skaičiuoja atstumą nuo taško A iki B. Ir tikrina ar galima iš transporto iškrauti krovinio talpą reikiamam taškui pagal jo talpos užklausą. Programai įtakai turi genetinio algoritmo parametrai, kaip kartų skaičius ar mutacijos koeficientas.

- Sprendžiamos TSP ir VRP problemos yra žinomos nuo senovės, tačiau jos yra NP klasės uždaviniai, kurių šiuo metu vis dar neįmanoma išspręsti momentaliai, turint ribotus išteklius. Sprendinio sutrumpinimo laikas gali būti pasiektas optimizuojant turimų ribotų išteklių panaudojimą.

- Genetinis algoritmas, skirtas spręsti CVRP (angl. Capacity vehicle route problem) - transporto užimtumo maršrutų problemą, veikia gerai su tam tikrais algoritmo parametrais. Atllikus bandymą pastebėta, kad pasirinkus tam tikrus parametrus algoritmas veikia efektyviau. Vienintelė genetinio algoritmo problema - prasta greitaveika. Nepaisant to, genetinio algoritmo sudaryti maršrutai yra kokybiški.

- Dviejų pasirinktųjų apsikeitimo algoritmo sprendimo kokybė ir greitaveika yra pakankamai gera, lyginant su likusiais algoritmais. Tačiau naujo hibridinio algoritmo kūrimo metu buvp atsižvelgta į dviejų pasirinktųjų apsikeitimo algoritmą tik optimizuojant maršrutą, nes dviejų pasirinktųjų algoritmas sprendžia tik maršruto optimizavimo, bet ne radimo problemą.

- Genetinio algoritmo sprendimo kokybė labai aukšta, tačiau greitaveika yra viena iš didžiausių algoritmo problemų. Bandymo metu buvo koreguotas tik mutacijos tikimybės parametras, todėl greitaveikos problemai reikalingas išsamesnis tyrimas.

Literatūros šaltiniai

[1] 2opt psuedo kodas.
https://github.com/rellermeyer/99tsp/tree/master/python/2opt.

[2] Cvrp - talpos transporto maršruto problemos genetinio algoritmo psuedo kodas.
https://github.com/tazo90/cvrp-python/blob/master/cvrp_random.py.

[3] Genetinio algoritmo psuedo kodas.
https://github.com/maoaiz/tsp-genetic-python.

[4] Modeliuojamo atkaitinimo psuedo kodas.
https://github.com/kz26/CMSC-27600/blob/master/project2/simAnneal.py.

[5] The nearest neighbor algorithm the nearest neighbor algorithm.
https://web.engr.oregonstate.edu/~tgd/classes/534/slides/part3.pdf.

[6] Wesam Ashour Ahmad Fouad El-Samak. Optimization of travelling salesman problem using a nity propagation clustering and genetic algorithm. JAISCR, 2015.
https://www.degruyter.com/downloadpdf/j/jaiscr.2015.5.issue-4/jaiscr-2015-0032/jaiscr-2015-0032.pdf.

[7] Jonas Blonskis Alfonsas Misevičius, Vytautas Bukšnaitis. Kombinatorinio optimizavimo ir genetinių algoritmų aspektai. Informacijos mokslai, 2005.
https://www.researchgate.net/publication/264893793_Kombinatorinio_optimizavimo_ir_genetiniu_algoritmu_aspektai.

[8] Zachariah A. Allen. Traveling salesman problem, 2015.

[9] Ahmad Sharieh Aryaf Al-Adwan, Basel A. Mahafzah. Solving traveling salesman problem using parallel repetitive nearest neighbor algorithm on otis-hypercube and otis-mesh optoelectronic architectures. The Journal of Supercomputing, 2018.
https://link.springer.com/article/10.1007/s11227-017-2102-y.

[10] Ronald Askin. A thesis presented in partial fulfillment of the requirements. Arizon State University, 2015.
https://pdfs.semanticscholar.org/acc2/785529ecd91c12387d8b8e08567c1ce9ab50.pdf.

[11] Michael J. Brusco. A comparison of simulated annealing algorithms for variable selection in principal component analysis and discriminant analysis. Computational Statistics and Data Analysis, 2014.
https://www.sciencedirect.com/science/article/pii/S0167947314000668.

[12] Yves Deville. Optimization for vehicle route problems, 2013.

[13] Jakub. Traveling salesman problem. JAMK University, 2013.
https://www.theseus.fi/bitstream/handle/10024/59942/Jakub_Stencek_TSP_Bachelor_Thesis.pdf?sequence=1.

[14] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. University of Montreal, Montreal, Quebec, Canada, 1991.

[15] Tarek Y. El Mekkawy Mohamed M. S. Abdulkader, Yuvraj Gajpal. Hybridized ant colony algorithm for the multi compartment vehicle routing problem. Applied Soft Computing, 2015.
https://www.sciencedirect.com/science/article/pii/S1568494615005219.

[16] Harilaos N. Psaraftis Paul M. Thompson. Cyclic transfer algorithms for multivehicle routing and scheduling problems. National Technical University of Athens, Athens, Greece, 1992.
http://web.a.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=
b35f0ed0-bbc5-4e06-96ca-b38da7f1f464%40sessionmgr4007.

[17] Tuomas Pellonpera. Taillard algorithm, 1996.
http://mistic.heig-vd.ch/taillard/articles.dir/Taillard1999.pdf.

[18] Tuomas Pellonpera. Ant colony optimization and the vehicle routing problem. Tampere University, 2014.
http://tampub.uta.fi/bitstream/handle/10024/95400/GRADU-1401262041.pdf;sequence=1.