Ajoneuvojen reititysongelma

Rasmus Harmanen

LuK-tutkielma
Helmikuu 2024

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Rasmus Harmanen: Ajoneuvojen reititysongelma
LuK-tutkielma, 19 s.
Sovellettu Matematiikka
Helmikuu 2024

Tässä tutkielmassa käsitellään yleistä ajoneuvojen reititysongelmaa, joka on yksi matemaattisen optimoinnin tutkituimmista ongelmista. Ajoneuvojen reititysongelmassa pitää palvella joukko asiakkaita jakeluautoilla, jotka lähtevät varastolta. Tämän lisäksi asiakkaat jaetaan jakeluautoille siten, että jokainen asiakas palvellaan tasan kerran. Tehtävässä minimoidaan jakeluautoille määrättyjen reittien yhteiskustannuksia. Tehtävänä ajoneuvojen reititysongelma on NP-vaikea eli sille tiedetään vain eksponentaalisen ajan vieviä algoritmeja. Lisäksi ongelmasta on olemassa monia erilaisia muunnoksia, kuten kapasiteetillinen, usean varaston sekä dynaaminen versio.

Tässä työssä esitetään kaksi erilaista tapaa mallintaa yleinen ajoneuvojen reititysongelma. Lisäksi esitellään kaksi eri menetelmää ajoneuvojen reititysongelman ratkaisemiseksi. Ensimmäinen menetelmistä on säästävä algoritmi, jossa kaikki mahdolliset kahden asiakkaan asiakasparit listataan paremmuusjärjestykseen. Paremmuusjärjestys saadaan sen perusteella, minkä verran muodostuu säästöä kahden asiakkaan yhdistämisestä samalle reitille. Säästävästä algoritmista on olemassa kaksi eri versiota: rinnakkain ja peräkkäin tehty reitin muodostus. Rinnakkaisella tavalla voidaan muodostaa usempi reitti samanaikaisesti, kun taas peräkkäisellä versiolla reitit muodostetaan yksi kerrallaan. Työssä esitellään molemmat tavat. Peräkkäisellä tavalla saadaan usein parempi ratkaisu, minkä lisäksi se on yleensä tehokkaampi tapa muodostaa ratkaisu useammalle asiakkaalle. Toinen työssä esiteltävä algoritmi on pyyhkäisy algoritmi. Siinä kaikki asiakkaat listataan ensin napakoordinaattien mukaan listaan, jota hyödyntäen määrätään asiakkaat jakeluautoille. Jokaisen jakeluauton reittiä parannetaan lopuksi vielä 2-optimaalisella haulla.

Asiasanat: ajoneuvojen reititysongelma, säästävä algoritmi, pyyhkäisy algoritmi.

Sisällys

1 Johdanto

Matemaattisessa optimoinnissa pyritään löytämään paras mahdollinen ratkaisu tarkastelun kohteena olevalle ongelmalle. Tätä varten ongelma täytyy mallintaa matemaattisesti. Näin ollen ongelmalle tarvitaan ensinnäkin päätösmuuttujat, jotka kuvastavat tehtäviä päätöksiä. Päätösmuuttujien avulla tehtävälle saadaan kohdefunktio, jota maksimoidaan tai minimoidaan. Usein tehtävissä minimoidaan esimerkiksi kustannuksia. Maksimoinnilla pyritään puolestaan esimerkiksi mahdollisimman suureen hyötyyn tai tuottoon. Lisäksi kohdefunktion f maksimointi on sama asia kuin funktion $-f$ minimointi.

Usein optimointiongelmassa tarvitaan myös rajoitteita, jotka muodostavat tehtävän sallitut ratkaisut, joiden joukosta parasta ratkaisua etsitään. Rajoitteet luokitellaan luonnollisiin ja varsinaisiin rajoitteisiin. Luonnollinen rajoite on jokin päätösmuuttujan luonnollinen ominaisuus, joka pitää myös esittää matemaattisesti. Esimerkiksi jos päätösmuuttuja esittää ämpärien lukumäärää, pitää sen arvon olla nolla tai sitä suurempi kokonaisluku, koska negatiivista määrä ämpäreitä ei voi kerätä. Varsinaiset rajoitteet taas pitävät sisällään kaikki muut ongelman mallinnukseen tarvittavat rajoitteet. Yksi esimerkki on marjaämpärien maksimimäärä, jota enempää ei pystytä keräämään, koska muuten ämpärit loppuvat kesken.

Rajoitteiden, kohdefunktion ja päätösmuuttujien ominaisuudet määrittävät yhdessä optimointiongelman tehtävätyypin. Optimointitehtävät voi karkeasti luokitella lineaarisiin sekä epälineaarisiin ongelmiin. Jotta jokin tehtävä on lineaarinen, täytyy viiden ehdon täyttyä [12]. Ensimmäinen ehto on verrannollisuus eli jokaisen päätösmuuttujan vaikutus kohdefunktioon ja rajoituksiin on suoraan verrannollinen sen arvoon. Toisena ehtona on, että jokaisen päätösmuuttujan vaikutus kohdefunktioon ja rajoituksiin on riippumaton muiden päätösmuuttujien arvoista. Kolmantena vaatimuksena on jaollisuus, joka edellyttää, että jokainen päätösmuuttuja on jatkuva. Neljäntenä ehtona jokaisen tehtävän parametrin täytyy olla vakio eikä satunnaismuuttuja. Viidentenä vaatimuksena on, että tehtävällä on ainoastaan yksi kohdefunktio. Kaksi ensimmäistä ehtoa yllä takaavat tehtävän lineaarisuuden ja jos ne eivät ole voimassa tehtävä on epälineaarinen. Jos puolestaan kolmas ehto ei täyty, tehtävä on diskreetti. Mikäli neljäs ehto ei ole voimassa on kyseessä stokastinen optimointitehtävä, jossa esiintyy yksi tai useampi satunnaismuuttuja. Mikäli viimeinen ehto puolestaan ei ole voimassa, tällöin kyseessä on monitavoiteoptimointitehtävä. Erilaisille optimointitehtävätyypeille on kehitetty omia ratkaisumenetelmiä. Tässä työssä keskitytään reititysongelmiin, jotka ovat diskreettejä optimointiongelmia.

Yksi eniten tutkittu reititysongelma on kauppamatkustajaongelma (*travelling salesman problem*). Ongelmassa on annettu joukko kaupunkeja ja kaikki etäisyydet

kaupunkien välillä. Tehtävänä on määrätä kauppamatkustajalle optimaalisin eli lyhyin mahdollinen reitti, joka käy joukon jokaisessa kaupungissa tasan kerran ja palaa lähtökaupunkiin. Kauppamatkustajaongelman tavoitteena on siis minimoida yllä kuvatun reitin pituutta tai reitin kustannuksia. Tehtävänä kauppamatkustajaongelma on helppo mallintaa, mutta vaikea ratkaista. Tämä johtuu siitä, että kauppamatkustajaongelma on NP-vaikea eli tällä hetkellä ongelman ratkaisemiseksi tunnetaan ainoastaan eksponentiaalisen ajan vieviä algoritmeja.

Kauppamatkustajaongelmaan on törmätty jo kauppamatkaajan käsikirjassa vuonna 1839, mutta 1900-luvulla matemaatikot Hamilton ja Kirkman esittivät sen matemaattisesti [1]. Tämän jälkeen kauppamatkustajaongelmaa on tutkittu paljon ja se on matemaattisen optimoinnin yksi tutkituimmista ongelmista. Vaikka kyseinen ongelma on vaikea ratkaista, sille on kehitetty useita erilaisia algoritmeja sekä heuristiikkoja, joilla tehtävälle saadaan likimääräisiä ratkaisuja. Algoritmien ja heuristiikoiden kehityksestä päätellen ongelman ratkaisemiselle on tarvetta [5]. Kauppamatkustajaongelman tärkeimpiä muunnoksia ovat kauppamatkustajaongelma aikaikkunalla [16], yleinen ajoneuvojen reititysongelma [4] sekä monitavoitteinen kauppamatkustajaongelma [9]. Kauppamatkustajaongelmaa käytetään ratkaisemaan esimerkiksi logistiikkassa ja mikrosirujen valmistuksessa ilmeneviä ongelmia. Joissakin logistiikan tehtävissä voi olla käytettävissä useampi ajoneuvo, jolloin kauppamatkustajaongelma muuttuu yleiseksi ajoneuvojen reititysongelmaksi.

Yleinen ajoneuvojen reititysongelma (*vehicle routing problem*) on eräs muunnos kauppamatkustajaongelmasta, jossa on mukana useampi kauppamatkustaja eli ajoneuvo. Tehtävässä kierrettävät kaupungit ovat asiakkaita ja lähtökaupunki on varasto eli ajoneuvojen reitit lähtevät varastolta. Varastolta jaetaan asiakkaille tilauksia, jotka vievät tietyn määrän ajoneuvojen kapasiteettia. Asiakkaan koko tilaus toimitetaan yhdellä ajoneuvolla. Jokaiselle ajoneuvolle rakennetaan jollain kriteerillä optimaalinen reitti, joka kiertää ajoneuvolle valikoitujen asiakkaiden kautta ja huomioi ajoneuvojen kapasiteetit sekä asiakkaiden sijainnit. Reitit muodostetaan siten, että jokainen asiakas palvellaan tasan kerran ja reittien yhteiskustannus on mahdollisimman pieni.

Ajoneuvojen reititysongelma on erityisen yleinen logistiikassa, sillä monet yritykset pyrkivät minimoimaan kuljetuksesta aiheutuvia kustannuksia. Kustannuksia syntyy aina, kun ajoneuvot lähtevät kuljettamaan tavaroita varastosta asiakkaille tai myymälöihin. Jokaisen ajoneuvon reitin täytyy olla tarkasti suunniteltu etukäteen, jotta ylimääräiset kulut saataisiin karsittua. Tämänlaisen ongelman voi mallintaa ajoneuvojen reititysongelmana, sillä yritys tietää esimerkiksi toimitettavien pakettien koot ja tilausten tehneiden asiakkaiden osoitteet, joiden luona täytyy käy-

dä toimittamassa paketti. Yleensä minimoitavana kohdefunktiona on reittien yhteispituudet, reiteiltä syntyvät kulut tai reitteihin kuluva aika. Yleistä ajoneuvojen reititysongelmaa voidaan hyödyntää myös matkapuhelinverkkojen läpi kulkevien datapakettien reitityksessä [11].

Yleinen ajoneuvojen reititysongelma täytyy ensin mallintaa sopivalla tavalla, joka huomioi esimerkiksi ajoneuvojen kapasiteetin sekä asiakkaiden sijainnin. Ongelman mallintamisen jälkeen sen ratkaisua pyritään hakemaan jollakin algoritmilla. Algoritmi on jono järjestyksessä suoritettavia sääntöjä, ohjeita tai tekoja, joita seuraamalla voidaan ratkaista tarkastelussa oleva optimointitehtävä. Deterministiset algoritmit ovat erittäin tutkittuja algoritmeja ja ne tuottavat jokaisella suorituskerralla saman ratkaisun, kun lähdetään liikkeelle samasta lähtöpisteestä. Tarkka algoritmi on deterministinen algoritmi, joka löytää aina globaalin optimin. Ratkaisuheuristiikka puolestaan ei välttämättä ratkaise tehtävää optimaalisesti, mutta tuottaa nopeasti ratkaisuja, jotka ovat usein melko hyviä. Heuristiikoissa on lisäksi lähes aina mukana satunnaisuutta eli heuristiikan eri suorituskerrat voivat tuottaa erilaisen tuloksen. Vaikka ajoneuvojen reititysongelman ratkaiseminen on laskennallisesti haastavaa, olemassa on useita ratkaisuheuristiikkoja, joilla pystytään tuottamaan järkevässä ajassa hyviäkin ratkaisuja.

Ensimmäisen heuristiikan yleiselle ajoneuvojen reititysongelmalle julkaisivat Ramser ja Dantzig vuonna 1959 artikkelissa [8]. Kyseinen algoritmi oli erityisesti kehitetty ajoneuvojen reititysongelman sovellukseen, jossa tarkasteltiin polttoaineen jakelun tehostamista. Vuonna 1964 Clarke ja Wright jatkokehittivät Dantzigin ja Ramserin algoritmia ja esittivät säästävän algoritmin (*savings algorithm*) ajoneuvojen reititysongelmalle, joka kuuluu niin sanottujen ahneiden algoritmien luokkaan. Ahneet algoritmit siirtyvät joka kierroksella aina sellaiseen uuteen ratkaisuun, joka pienentää kohdefunktion arvoa mahdollisimman paljon. Tämän seurauksena ahneet algoritmit päätyvät usein vain lokaaleihin optimeihin eivätkä huomioi tehtävän globaalia käyttäytymistä.

Tässä työssä tarkastellaan yleistä ajoneuvojen reititysongelmaa. Aluksi luvussa 2 esitellään kaksi eri tapaa mallintaa yleinen ajoneuvojen reititysongelma. Luvussa 3 esitellään puolestaan kaksi ratkaisualgoritmia yleiselle ajoneuvojen reititysongelmalle. Lopuksi luvussa 4 esitellään työn yhteenveto.

2 Yleinen ajoneuvojen reititysongelma

Yleisessä ajoneuvojen reititysongelmassa on joukko asiakkaita, jotka saavat ostamansa tavarat varastolta. Suurelta varastolta lähtee päivittäin monta jakeluautoa

mahdollisimman täynnä tavaroita kohti asiakkaita. Tästä syystä jokaiselle jakeluautolle täytyy määrätä palveltavat asiakkaat sekä ratkaista optimaalisin jakelureitti ennen lähtöä. Jakelureitin tulee lähteä varastolta, käydä jokaisella ajoneuvolle määrätyllä asiakkaalla vain kerran ja päätyä takaisin varastolle. Reitin luomisessa on tyypillisesti monta rajoittavaa tekijää, kuten esimerkiksi kuljettajien ajoajat ja jakeluautojen kapasiteetit. Yleensä reititysongelmien tavoitteena on saada kustannukset mahdollisimman pieniksi. Jotta tähän päästään, jokaisen jakeluauton tulisi ensinnäkin olla mahdollisimman täynnä, sillä tällöin yksittäisen reitin kustannukset jakautuisivat useammalle asiakkaalle. Myöskään usean jakeluauton ei kannata käydä viemässä samalle asiakkaalle yksittäisestä tilauksesta osia, vaan yhden jakeluauton kannattaa viedä samalla kerralla koko tilaus, jos se vain mahtuu kokonaisuudessaan kyytiin. Jakeluautojen reittien paremmuutta mitataan usein ajan tai matkan mukaan. Aikaa käytetään mittarina, koska jokaiselta minuutilta työntekijöille pitää maksaa palkkaa ja kuljettajien ajoaikarajoitteet saattavat rajoittaa reittejä. Matkaa halutaan puolestaan usein minimoida, koska jakeluautot kuluttavat polttoainetta ja tiettyjen kilometrimäärien jälkeen ajoneuvoja pitää aina huoltaa. Näin ollen, tehtävän mallintamisessa on otettava huomioon useita tekijöitä.

Tehtävän mallintaminen alkaa asiakkaiden määrittämisestä eri jakeluautoille. Tämä tarkoittaa sitä, että yleinen ajoneuvojen reititysongelma koostuu useasta kauppamatkustajaongelmasta, sillä periaatteessa jokaiselle ajoneuvolle ratkaistaan oma kauppamatkustajaongelma sen jälkeen, kun ajoneuvon asiakkaat ovat tiedossa. Näin ollen yleisessä ajoneuvojen reititysongelmassa jokaiselle ajoneuvolle määrätty reitti on vastaavanlainen kuin kauppamatkustajaongelmassa eli jokaisen reitin asiakkaan luona käydään vain kerran ja reitin pitää olla jollain mittarilla optimaalisin. Näin ollen jokaisen jakeluauton reitti on Hamiltonin sykli, joka käy jokaisen reitin asiakkaan luona tasan kerran ja lopuksi palaa varastolle. Mikäli kaikki toimitettavat tavarat mahtuisivat yhteen jakeluautoon on kyseessä tavallinen kauppamatkustajaongelma. Vastaavasti ajateltuna kauppamatkustajaongelma muuttuu yleiseksi ajoneuvojen reititysongelmaksi, mikäli yksi jakeluauto joutuu ajamaan vähintään kaksi eri reittiä kaikkien tavaroiden toimittamiseksi.

Yleisestä ajoneuvojen reititysongelmasta on olemassa erilaisia muunnoksia. Eräässä niistä on yhden varaston sijaan useampi varasto. Usean varaston ongelmassa voi olla haasteena, että jollain varastolla ei ole riittävästi tiettyä tavaraa tai sieltä puuttuu kokonaan tietty tavara, jolloin jakeluauton täytyy käydä asiakkaiden lisäksi toisellakin varastolla. Useamman varaston ongelmaa on esimerkiksi tutkittu artikkelissa [3]. Tässä työssä keskitytään kuitenkin vain sellaiseen ajoneuvojen reititysongelmaan, jossa on yksi varasto.

2.1 Ensimmäinen malli

Yleisen ajoneuvojen reititysongelman mallinnus alkaa yleensä listasta, jossa on asiakkaiden osoitteet ja tieto siitä, mitä tavaroita kyseiselle asiakkaalle toimitetaan. Kyseisellä listalla on myös kuljetettavien tavaroiden kapasiteettivaatimukset. Lisäksi tiedetään etäisyys varastolta jokaisen asiakkaan luo sekä etäisyys jokaisen kahden asiakkaan välillä. Käytössä olevien jakeluautojen kapasiteettirajoitukset ovat tiedossa ja ne täytyy huomioida reittejä laatiessa. Jakeluautot lähtevät aina varastolta palvelemaan asiakkaita ja palaavat lopuksi takaisin varastolle. Tavoitteena on määrätä jokaiselle jakeluautolle optimaalisin reitti etukäteen sekä palvella kaikki asiakkaat.

Tehtävän mallintamiseksi pitää ensin jollain tavalla esittää matemaattisesti asiakkaiden osoitteet, reitit varastolta asiakkaille sekä asiakkaiden välillä. Reiteille tarvitaan myös painot. Painot kuvastavat reittien kustannuksia, jotka voivat esittää esimerkiksi etäisyyksiä tietyssä pituusyksikössä, kuten kilometreinä, tai vaihtoehtoisesti rahamääränä, jonka reitit kustantavat.

Seuraavaksi esitellään yleiselle ajoneuvojen reititysongelmalle Dantzigin ja Ramserin malli vuodelta 1959. Esitystapa perustuu lähteeseen [4]. Tehtävän data esitetään graafilla

$$G = (V, E),$$

jossa joukko V koostuu graafin solmuista eli asiakkaista ja varastosta. Joukko E puolestaan pitää sisällään solmuja yhdistävät kaaret eli reitit asiakkaiden sekä varaston välillä. Jatkossa graafin solmuja esitetään kirjoittamalla

$$V = \{v_0, v_1, v_2, ..., v_n\} = \{0, 1, 2, ..., n\}.$$

Normaalisti varastosta käytetään merkintää v_0 ja joukko $\{v_1, v_2, ..., v_n\}$ puolestaan pitää sisällään kaikki asiakkaat. Asiakkaiden kokonaislukumäärä on n. Yleisessä ajoneuvojen reititysongelmassa kaarijoukoksi valitaan

$$E = \{(v_i, v_j) \,|\, v_i \,, v_j \in V,\, i < j\}. \tag{1}$$

Näin ollen yksittäinen kaari (v_i, v_j) on reitti solmujen v_i ja v_j välillä. Kaaret ovat tyypillisesti suuntaamattomia, jolloin kuljettaessa solmusta v_i solmuun v_j, kaaren paino on sama riippumatta kulkusuunnasta. Tällaisessa tilanteessa tehtävä on symmetrinen. Kaarijoukossa (1) esiintyvä ehto $i < j$ takaa, että jokaisen solmuparin välille löytyy vain yksi kaari. Jos tehtävä on epäsymmetrinen, kaarella on eri paino kulkusuunnasta riippuen. Epäsymmetrisissä tehtävissä jokainen kaari jaetaan kahdeksi nuoleksi, joita voi kulkea ainoastaan yhteen suuntaan. Tässä työssä käsitellään ainoastaan symmetrisiä tilanteita.

Kaarten painot esitetään kustannusmatriisilla

$$C = (c_{ij})_{(v_i,v_j)\in E}.$$

Koska tarkasteltava tehtävä on symmetrinen, matriisissa C toteutuu ehto

$$c_{ij} = c_{ji}.$$

Lisäksi tehtävässä yksikään kaari ei kulje samalta asiakkaalta takaisin samalle asiakkaalle, jolloin kaarta (v_i, v_i) ei ole joukossa E ja kustannusmatriisin diagonaalin alkiot ovat nollia.

Kuten aikaisemmin mainittiin paino c_{ij} kuvastaa reitin (v_i, v_j) kustannuksia. Painot voivat ilmaista esimerkiksi ajallista kustannusta, kuten mitä maksaa kuljettajan ja jakeluauton käyttö reitillä. Painot valitaan aina ratkaistavan ongelman mukaan. Tyypillisesti ajoneuvojen reititysongelmassa kulujen minimointi on sama asia kuin matkan minimointi. Tämä johtuu siitä, että jokainen jakeluauto kuluttaa polttoainetta sekä aikaa. Tehtävässä voidaan myös haluta rajoittaa yksittäisen reitin pituutta ja jatkossa reitin mahdollista maksimipituutta merkitään parametrilla $L > 0$.

Tehtävän päätösmuuttujat valitaan siten, että niiden avulla saadaan selville jakeluautojen reitit asiakkaiden ja varaston välillä. Jakeluautoja on varastolla M kappaletta, joista jokaisen maksimikapasiteetti on $K > 0$. Jakeluautoja esitetään indekseillä $k = 1, ..., M$. Päätösmuuttujina ovat binäärimuuttujat

$$x_{ij}^k = \begin{cases} 1, & \text{jos jakeluauton } k \text{ optimireitti kulkee kaarta } (i,j) \text{ pitkin} \\ 0, & \text{muuten.} \end{cases}$$

Yksittäisen asiakkaan i tilauksen kapasiteettia kuvataan parametrilla $q_i > 0$. Jakeluautojen määrällä M ei ole varsinaista ylärajaa esitettävässä mallissa, mutta alarajan saa selville laskemalla yhteen kaikkien tilausten kapasiteetit ja jakamalla se yhden jakeluauton kapasiteetilla, minkä jälkeen saatu luku pyöristetään ylös päin lähinpään kokonaislukuun. Lisäksi vaaditaan, että ainoastaan yksi jakeluauto voi käydä tietyn asiakkaan luona. Jokainen jakeluauto palaa myöskin varastolle tyhjänä, jolloin $q_0 = 0$. Mikäli kaarta (v_i, v_j) on käytetty ajoneuvon k reitissä, saa binäärimuuttuja x_{ij}^k arvoksi yksi. Kun binäärimuuttujat yhdistetään kustannusmatriisiin, saadaan tehtävän kohdefunktio, joka on muotoa

$$\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{M} c_{ij}x_{ij}^k.$$

Koska ajetuista reiteistä muodostuvat kulut halutaan pitää mahdollisimman pieninä, kohdefunktiota minimoidaan. Mikäli jotakin kaarta käytetään ajoneuvon k reitissä on $x_{ij}^k = 1$. Kyseisten solmujen välinen kustannus lasketaan mukaan kohdefunktioon, sillä

$$x_{ij}^k c_{ij} = 1 c_{ij} = c_{ij}.$$

Jos kaarta ei puolestaan käytetä ajoneuvon k reitissä on $x_{ij}^k = 0$ ja kaaren kustannus ei realisoidu, sillä kyseinen termi kohdefunktiossa on

$$x_{ij}^k c_{ij} = 0 c_{ij} = 0.$$

Yleinen ajoneuvojen reititysongelma saadaan mallinnettua seuraavasti

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{M} c_{ij} x_{ij}^k \tag{2}$$

$$\text{s. t.} \quad \sum_{i=0}^{n} \sum_{k=1}^{M} x_{ij}^k = 1 \quad \forall\, j \in \{1, ..., n\} \tag{3}$$

$$\sum_{j=0}^{n} \sum_{k=1}^{M} x_{ij}^k = 1 \quad \forall\, i \in \{1, ..., n\} \tag{4}$$

$$\sum_{i=0}^{n} x_{ip}^k - \sum_{j=0}^{n} x_{pj}^k = 0 \quad \forall\, p \in \{1, ..., n\},\ k \in \{1, ..., M\} \tag{5}$$

$$\sum_{i=0}^{n} q_i (\sum_{j=0}^{n} x_{ij}^k) \leq K \quad \forall\, k \in \{1, ..., M\} \tag{6}$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij}^k \leq L \quad \forall\, k \in \{1, ..., M\} \tag{7}$$

$$\sum_{j=1}^{n} x_{0j}^k \leq 1 \quad \forall\, k \in \{1, ..., M\} \tag{8}$$

$$\sum_{i=1}^{n} x_{i0}^k \leq 1 \quad \forall\, k \in \{1, ..., M\} \tag{9}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall\, i, j \in \{0, ..., n\}, \quad k \in \{1, ..., M\}. \tag{10}$$

Tehtävän rajoite (3) takaa, että jokaisen asiakkaan luokse saavutaan tarkalleen yhdellä jakeluautolla ja rajoite (4) varmistaa, että jokaisen asiakkaan luota lähdetään yhdellä jakeluautolla. Näin ollen jokainen asiakas on vain yhdellä reitillä. Rajoite (5) takaa reittien jatkuvuuden eli sen että, kun asiakkaan luokse on saavuttu jakeluautolla, hänen luotaan myös jatketaan reittiä eteenpäin samalla jakeluautolla. Seuraava rajoite (6) estää jakeluautojen kapasiteetin ylittämisen ja rajoite (7) puolestaan pitää huolta, ettei ajoneuvojen reittien maksimipituudet ylity. Rajoitteet (8) ja (9) takaavat, että jokainen jakeluauto lähtee varastolta korkeintaan kerran ja palaa takaisin varastolle korkeintaan kerran. Viimeinen rajoite (10) varmistaa, että muuttujat x_{ij}^k ovat binäärisiä.

Mallissa on haasteena, että yksittäisen ajoneuvon reitti ei välttämättä ole yhtenäinen. Ajoneuvon reitti voi siis koostua erillisistä alisykleistä. Tästä ongelmasta

päästään kuitenkin eroon lisäämällä malliin lisärajoitteet, joilla suljetaan alisyklit pois.

2.2 Toinen malli

Edellisen mallinnuksen merkintöjä hyödyntäen voidaan myös esittää toinen malli yleiselle ajoneuvojen reititysongelmalle. Seuraavaksi esitettävä malli perustuu lähteisiin [8, 14, 15].

Mallissa reittejä esitetään päätösmuuttujilla

$$x_{ij} = \begin{cases} 1, & \text{jos optimireitti kulkee kaarta } (i,j) \text{ pitkin} \\ 0, & \text{muulloin.} \end{cases}$$

Jakeluauton k käymistä asiakkaan i luona kuvataan puolestaan muuttujalla

$$y_{ik} = \begin{cases} 1, & \text{jos jakeluauto } k \text{ käy asiakkaalla } i \\ 0, & \text{muuten.} \end{cases}$$

Näin ollen yleinen ajoneuvojen reititysongelma saadaan mallinnettua seuraavasti

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{11}$$

$$\text{s. t.} \quad \sum_{i=0}^{n} q_i y_{ik} \leq K, \quad \forall\, k = 1, 2, ..., M \tag{12}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad \forall\, S \subseteq V \backslash \{v_0\} \tag{13}$$

$$\sum_{k=1}^{M} y_{ik} = \begin{cases} M, & i = 0 \\ 1, & \forall\, i = 1, 2, ..., n \end{cases} \tag{14}$$

$$\sum_{i=0}^{n} x_{ij} = \sum_{i=0}^{n} x_{ji} = \begin{cases} M, & j = 0 \\ 1, & \forall\, j = 1, 2, ..., n \end{cases} \tag{15}$$

$$x_{ij} \in \{0, 1\}, \quad \forall\, i, j \in \{0, 1, ...n\} \tag{16}$$

$$y_{ik} \in \{0, 1\}, \quad \forall\, i \in \{0, 1, ...n\}, \quad k \in \{1, ..., M\}. \tag{17}$$

Kohdefunktio (11) minimoi muodostettujen reittien pituutta. Rajoite (12) vastaa siitä, ettei tilauksien kapasiteetin tarve ylitä ajoneuvojen kapasiteettia. Seuraava rajoite (13) varmistaa, että asiakkaille on tehty oikea määrä kohdistuksia. Rajoitteella (14) rajataan, että jokaisella asiakkaalla käydään tasan kerran ja että kaikki

ajoneuvot käyvät varastolla. Rajoite (15) määrää ajoneuvot aloittamaan ja lopettamaan reittinsä varastolle. Toiseksi viimeisin rajoite (16) ja viimeinen rajoite (17) takaavat, että päätösmuuttujat ovat binäärisiä.

Mikäli ajoneuvojen lukumäärä $M = 1$, tehtävä on kauppamatkustajaongelma. Lisäksi yllä olevan mallin ongelmana on, että päätösmuuttujia x_{ij} ja y_{im} ei ole linkitetty toisiinsa, jolloin muuttujien x_{ij} määräämiä reittejä ei välttämättä ajeta ajoneuvoilla, koska ajoneuvoille annetut asiakkaat voivat erota reittien asiakkaista.

Usein reititysongelman rajoitteissa voidaan huomioida reitin maksimiaika D_{max}, joka kertoo kauanko yksi ajoneuvo saa olla reitillään. Mikäli maksimiaikaa käytetään, joudutaan arvioimaan maksimipalveluaika, joka kuluu asiakkaan luonna. Maksimipalveluajasta ei voida joustaa, koska silloin reittiin kuluva kokonaisaika saattaisi kasvaa liian suureksi. Tästä käytetään termiä kova aikaikkuna, joka on etukäteen asetettu aika, joka joudutaan kuluttamaan jokaisen asiakkaan luona loppuun asti vaikka aikaa menisikin vähemmän. Reitin ajamiseen kuluu myös aikaa. Vaikka ajoneuvot harvoin etenevät koko ajan reitillä yhden kilometrin esimerkiksi yhtä minuuttia tai tuntia kohden, mallissa täytyy tehdä kyseisenlainen yksinkertaistus. Tästä syystä on asetettava jokin sopiva suhdearvo ajan ja etäisyyden välille. Yleensä suhde on 1.00, jolloin yksi matkan yksikkö vastaa yhtä ajan yksikköä.

3 Ratkaisuheurestiikkoja

Yleistä ajoneuvojen reititysongelmaa ei yleisesti pystytä ratkaisemaan tarkoilla ratkaisualgoritmeilla, koska tehtävä on NP-vaikea. Asian todistivat Bodin ja Golden vuonna 1981 [2]. NP-vaikeus tarkoittaa sitä, että tehtävä ratkaisemiseksi tiedetään ainoastaan eksponentiaalisen ajan vieviä algoritmeja. Koska yleinen ajoneuvojen reititysongelma on tärkeä sovellus ja alunperin esitetty jo yli 60 vuotta sitten, sen ratkaisemiseksi on kehitetty monia erilaisia ratkaisuheuristiikkoja.

Heuristiikat jaetaan yleisesti kahteen eri luokkaan. Tavallisiin heuristiikkoihin ja metaheuristiikkoihin. Tavallisia heuristiikkoja on kehitetty jo 1960-luvulta lähtien, kun taas metaheuristiikat ovat yleistyneet vasta viime vuosikymmeninä, kun tietokoneiden laskentateho on parantunut huomattavasti. Metaheuristiikat vaativat paljon enemmän muistia, aikaa ja laskentatehoa kuin tavalliset heuristiikat. Tavalliset heuristiikat löytävätkin tästä syystä ratkaisun nopeammin kuin metaheuristiikat. Eri heuristiikat tuottavat kuitenkin vain likimääräisiä ratkaisuja ja näin ollen ei ole takeita edes ratkaisun lokaalista optimaalisuudesta. Heuristiikkojen ratkaisut ovat kuitenkin usein melko hyviä, vaikka ne eivät välttämättä ole optimaalisia missään mielessä.

Tavalliset heuristiikat yleiselle ajoneuvojen reititysongelmalle ovat suurimmilta osin konstruktiivisia heuristiikkoja. Konstruktiiviset heuristiikat rakentavat ratkaisua pala palalta siten, että alussa jokaiselle ajoneuvolle asetetaan tyhjä reitti. Tyhjiin reitteihin lisätään kerta toisensa jälkeen aina jokin uusi asiakas, kunnes konstruktiivisen heuristiikan muodostamassa ratkaisussa on mukana kaikki asiakkaat. Näin ollen konstruktiivinen heuristiikka lopetetaan, kun yksittäinen sallittu ratkaisu on saatu valmiiksi. Toinen mahdollinen heuristiikka on parantava haku. Parantavassa haussa hyödynnetään lokaalia hakua ja lähdetään liikkeelle jostain sallitusta ratkaisusta. Lokaalilla haulla pyritään jollain ennalta määrätyllä tavalla parantamaan nykyistä ratkaisua. Mikäli parempi ratkaisu löytyy, niin siihen siirrytään ja tätä uutta ratkaisua parannetaan seuraavaksi lokaalilla haulla. Tätä prosessia toistetaan niin kauan kunnes lokaali haku ei enää tuota parempaa ratkaisua. Näin ollen konstruktiivinen heuristiikka ja parantava heuristiikka ovat keskenään hyvin erilaisia menetelmiä. Toinen vain konstruoi sallitun ratkaisun, kun taas toisen suoritus lähtee liikkeelle sallitusta ratkaisusta. Joissakin tapauksissa konstruktiivisella heuristiikalla pystytään tuottamaan parantavan haun lähtöratkaisu.

Yleiselle ajoneuvojen reititysongelmalle vanhin ratkaisuheuristiikka on säästävä algoritmi (*savings algorithm*), jonka kehittivät Clarke ja Wright vuonna 1964 [6]. Kyseinen heuristiikka on erittäin tunnettu yleisen ajoneuvojen reititysongelman ratkaisumenetelmä ja se esitellään seuraavaksi tarkemmin. Tämän jälkeen tutustutaan pyyhkäisy algoritmiin (*sweeping algorithm*) [10], joka on myös erittäin käytetty ratkaisumenetelmä. Molemmat algoritmit ovat konstruktiivisia heuristiikkoja.

3.1 Säästävä algoritmi

Ensimmäinen ja yksi tunnetuimmista heuristiikoista yleisen ajoneuvojen reititysongelman ratkaisemiseksi on tässä luvussa esiteltävä säästävä algoritmi. Clarken ja Wrightin kehittämä algoritmi toimii suunnatuissa sekä suuntaamattomissa ajoneuvojen reititysongelmissa [6]. Algoritmin tavoitteena on minimoida jakelukustannukset varastolta asiakkaille sekä käydä jokaisella asiakkaalla tasan kerran. Säästävä algoritmi ottaa huomioon ajoneuvojen kapasiteetit sekä löytää suhteellisen nopeasti ratkaisun. Ratkaisun optimaalisuutta ei kuitenkaan pystytä takaamaan.

Säästävästä algoritmista on kaksi eri versiota, jotka ovat rinnakkainen ja peräkkäinen reitinrakennus. Rinnakkaisella tavalla voidaan muodostaa yhden tai useamman ajoneuvon reitti samanaikaisesti, kun taas peräkkäisellä tavalla muodostetaan yhden ajoneuvon reitti ensin kokonaan valmiiksi, minkä jälkeen vasta siirrytään seuraavan ajoneuvon reitin muodostukseen. Molemmissa algoritmin versioissa ensimmäinen askel on sama ja siinä lasketaan säästön määrä jokaisen kahden asiakkaan i

ja j välille, kun asiakkaat yhdistetään toisiinsa käymättä varastolla välissä. Säästön määrän saamiseksi ensin summataan yhteen kustannukset kulkea varastolta asiakkaalle i sekä asiakkaalta j takaisin varastolle. Tästä summasta vähennetään vielä kustannus, joka tulee kulkiessa suoraan asiakkaalta i asiakkaalle j. Säästöjen laskemisen jälkeen muodostetaan säästölista, johon säästöt listataan laskevaan järjestykseen. Tämän jälkeen säästölista käydään järjestyksessä läpi, koska mitä suurempi säästö sitä suurempi hyöty saadaan yhdistettäessä kyseiset asiakkaat samaan reittiin. Jokaisen kaaren kohdalla selvitetään lisäksi erikseen voidaanko kyseiset asiakkaat yhdistää samaan reittiin ilman ajoneuvon kapasiteetin ylittymistä. Myöskään aikaisemmin muodostettujen ajoneuvojen reitit eivät saa muuttua vääränlaisiksi. Vääränlaisella reitillä tarkoitetaan, että saman asiakkaan luona käytäisiin kaksi kertaa.

Seuraavaksi esitellään tarkemmin säästävä algoritmi.

Algoritmi 1. Säästävä algoritmi

Askel 0. *(Alustus)* Laske säästöt jokaisen kahden asiakkaan välille kaavalla

$$s_{ij} = c_{i0} + c_{0j} - c_{ij}, \quad \forall\, i, j \in V \text{ ja } i \neq j.$$

Askel 1. Järjestä säästöt paremmuusjärjestykseen eli suurimmasta pienimpään. Tämä muodostaa säästölistan.

Askel 2. Valitse suoritetaanko algoritmista peräkkäinen vai rinnakkainen versio. Jos valinta on peräkkäinen mene askeleeseen 4.

Askel 3. Rinnakkainen reitinmuodostus

- **Askel 3a.** Valitse tarkasteluun säästölistan ensimmäistä alkiota vastaava asiakaspari.
- **Askel 3b.** Ota ratkaisuun mukaan uusi tyhjä jakeluauton reitti.
- **Askel 3c.** Mikäli tarkastelussa olevan asiakasparin molemmat asiakkaat löytyvät jo tehdyistä reiteistä, mene askeleeseen 3f.
- **Askel 3d.** Jos toinen asiakkaista on jollain tehdyistä reiteistä, niin lisää asiakaspari kyseiselle reitille, ellei reitin kapasiteetti ylity, ja mene askeleeseen 3f.
- **Askel 3e.** Lisää asiakaspari jollekin olemassa olevista reiteistä, jonka kapasiteetti ei ylity. Jos kaikkien reittien kapasiteetti ylittyy, palaa askeleeseen 3b.

Askel 3f. Jos säästölistalla on seuraava alkio, valitse sitä vastaava asiakaspari tarkasteluun ja palaa askeleeseen 3c. Muutoin päätä algoritmin suorittaminen, sillä reitit ovat valmiit.

Askel 4. Peräkkäinen reitinmuodostus

Askel 4a. Ota ratkaisuun mukaan uusi tyhjä jakeluauton reitti ja valitse tarkasteluun säästölistan ensimmäinen asiakaspari, jota ei olla vielä lisätty millekkään reitille.

Askel 4b. Jos toinen tai molemmat asiakasparin asiakkaista löytyvät jo aikaisemmilla kierroksilla valmiiksi saaduilta reiteiltä, mene askeleeseen 4e.

Askel 4c. Jos toinen asiakasparin asiakkaista on jo lisättynä nykyiselle reitille, niin lisää puuttuva asiakas asiakasparista reittiin, ellei reitin kapasiteetti ylity, ja mene askeleeseen 4e.

Askel 4d. Jos kumpikaan asiakasparin asiakkaista ei ole nykyisellä reitillä ja reitin kapasiteetti ei ylity, niin lisää kyseisen asiakasparin molemmat asiakkaat reitille.

Askel 4e. Jos säästölistalla on seuraava alkio, valitse sitä vastaava asiakaspari tarkasteluun ja palaa askeleeseen 4b.

Askel 4f. Mikäli kaikki asiakkaat on jo lisätty jollekin reitille päätä algoritmin suorittaminen, sillä reitit ovat valmiit. Muutoin siirry askeleeseen 4a.

Seuraavaksi tutustutaan algoritmin toimintaan esimerkin avulla.

Esimerkki 1. Tarkastellaan ajoneuvojen reititysongelmaa, jossa on viisi asiakasta ja ratkaistaan tehtävä säästävällä algoritmilla. Tässä esimerkissä verkko on suuntaamaton eli kaikki kustannukset ovat symmetrisiä. Jakeluautojen määrä ei ole kiinnitetty etukäteen.

Aloitetaan tehtävä esittämällä kustannusmatriisi

$$C_{ij} = \begin{pmatrix} 0 & 17 & 20 & 9 & 14 & 23 \\ 17 & 0 & 10 & 18 & 15 & 9 \\ 20 & 10 & 0 & 27 & 9 & 21 \\ 9 & 18 & 27 & 0 & 19 & 16 \\ 14 & 15 & 9 & 19 & 0 & 14 \\ 23 & 9 & 21 & 16 & 14 & 0 \end{pmatrix},$$

jossa on kaikkien asiakkaiden sekä varaston ja asiakkaiden väliset kulut. Seuraavaksi esitetään vektori

$$\mathbf{q} = \begin{pmatrix} 0 & 3.7 & 3.5 & 3.0 & 2.5 & 3.2 \end{pmatrix}^\top,$$

jossa on jokaiselle asiakkaalle vietävän tilauksen kapasiteetin tarve. Jokaisen jakeluauton kapasiteetti K on puolestaan 10.

Tehtävän ratkaisemiseksi on esitetty yllä kaikki tarvittavat tiedot. Aloitetaan tarkastelemaan algoritmin 1 suoritusta. Askeleessa 0 lasketaan säästöt, joista saadaan muodostettua säästömatriisi S. Askeleessa 1 järjestämällä säästömatriisin alkiot suurimmasta pienimpään saadaan säästölista. Säästöt lasketaan kaavalla $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. Yksi esimerkki säästöistä on $s_{1,2} = 17 + 20 - 10 = 27$. Kun kaikki säästöt on laskettu, saadaan matriisiksi

$$S = \begin{pmatrix} 0 & 27 & 8 & 16 & 31 \\ 27 & 0 & 2 & 25 & 22 \\ 8 & 2 & 0 & 4 & 16 \\ 16 & 25 & 4 & 0 & 23 \\ 31 & 22 & 16 & 23 & 0 \end{pmatrix}.$$

Kun säästöt listataan laskevaan järjestykseen, lopputuloksena on lista

$$s_{1,5},\ s_{1,2},\ s_{2,4},\ s_{4,5},\ s_{2,5},\ s_{1,4},\ s_{3,5},\ s_{1,3},\ s_{3,4} \text{ ja } s_{2,3}.$$

Algoritmi määrää montako jakeluautoa tarvitaan, jotta kaikki asiakkaat tulee palveltua. Askeleessa 2 valitaan versio, jolla tehtävä ratkaistaan. Ensin tarkastellaan miten ratkaisun muodostaminen tapahtuu rinnakkaisella tavalla. Askeleessa 3a otetaan tarkasteluun säästölistan ensimmäinen alkio $s_{1,5}$. Seuraavaksi askeleessa 3b lisätään ratkaisuun ensimmäinen tyhjä jakeluauton reitti. Kumpaakaan tarkastelussa olevan asiakasparin asiakasta ei ole lisätty vielä millekkään reitille, joten askeleita 3c ja 3d ei suoriteta. Askeleessa 3e asiakaspari $s_{1,5}$ lisätään tällä hetkellä ratkaisussa olevaan ainoaan jakeluauton reittiin. Näin ollen jakeluauton reitti on $0 - 1 - 5 - 0$ ja kapasiteetista on käytetty $3.7 + 3.2 = 6.9$ yksikköä. Tämän jälkeen askeleessa 3f siirrytään tarkastelemaan säästölistalla seuraavana olevaa alkiota $s_{2,4}$ ja palataan askeleeseen 3c, jossa huomataan, että kumpaakaan asiakasparin asiakasta ei ole lisätty millekkään olemassa olevalle reitille, joten askeleet 3c ja 3d ohitetaan. Askeleessa 3e huomataan, että nykyisen ja ainoan jakeluauton kapasiteetti kuitenkin ylittyy, mikäli asiakas 2 lisätään jakeluauton reitille. Näin ollen siirrytään takaisin askeleeseen 3b, jossa otetaan uusi tyhjä jakeluauton reitti ratkaisuun mukaan. Tämän jälkeen tehdään samat huomiot askeleissa 3c ja 3d kuin äsken. Näin ollen askeleessa

3e asiakaspari voidaan lisätä uudelle jakeluautolle. Tässä kohtaa jakeluautojen reitit ovat $0-1-5-0$ ja $0-2-4-0$. Kyseiset reitit vievät kapasiteettia 6,9 ja 6 yksikköä. Tämän jälkeen askeleessa 3f siirrytään säästölistalla seuraavana olevaan alkioon $s_{4,5}$ ja palataan askeleeseen 3c, jossa huomataan, että molemmat asiakkaat ovat jo jollakin reitillä. Tämän takia palataan heti takaisin askeleeseen 3f, jossa otetaan seuraava asiakaspari säästölistalta tarkasteluun. Seuraavaksi tarkasteltavana ovat järjestyksessä asiakasparit $s_{2,5}$ ja $s_{1,4}$. Askeleessa 3c kuitenkin taas huomataan, että molempien säästöjen kaikki neljä asiakasta ovat jo lisättyinä jollekkin jakeluauton reitille, joten molemmissa tapauksissa palataan taas heti takaisin askeleeseen 3f valitsemaan uusi säästölistan alkio tarkasteluun. Asiakasparin $s_{1,4}$ jälkeen seuraavaksi valitaan asiakaspari $s_{3,5}$. Askelessa 3c huomataan, että asiakasta 3 ei ole vielä lisätty millekkään jakeluauton reitille, joten siirrytään askeleeseen 3d, jonka mukaan asiakas 3 voidaan lisätä reitille $0-1-5-3-0$, koska jakeluauton kapasiteetti ei ylity. Näin ollen kyseisen jakeluauton kapasiteettia on käytetty 9.9 yksikköä. Tämän jälkeen askeleessa 3f huomataan, että kaikki asiakkaat palvellaan jo jollakin jakeluauton reitillä, joten algoritmin suoritus päättyy. Saadut reitit ovat $0-1-5-3-0$ ja $0-2-4-0$. Ensimmäisen reitin kapasiteettivaatimus on 9.9 yksikköä ja toisen reitin on 6 yksikköä. Kustannukset ensimmäiseltä reitiltä ovat $17+9+21+20=67$ ja toiselta reitiltä $20+27+14=61$. Reittien yhteenlasketut kustannukset ovat 128.

Seuraavaksi muodostetaan ratkaisu säästävän algoritmin peräkkäisellä tavalla. Askeleesta 4a valitaan ratkaisuun ensimmäinen tyhjä jakeluauton reitti. Samassa askeleessa otetaan myöskin tarkasteluun säästölistan ensimmäinen asiakaspari $s_{1,5}$. Askeleet 4b ja 4c jätetään välistä, koska kumpikaan asiakkaista ei ole jo aikaisemmin valmiiksi saadulla tai nykyiselle reitillä. Askeleessa 4d saadaan lisättyä asiakkaat 1 ja 5 nykyiseen reittiin, joka on tämän jälkeen $0-1-5-0$ ja jakeluauton kapasiteetista on käytetty $3.7+3.2=6.9$ yksikköä. Tämän jälkeen askeleessa 4e otetaan tarkasteluun säästölistalta seuraavana oleva asiakaspari $s_{1,2}$ ja palataan askeleeseen 4b, jossa todetaan, että kumpikaan asiakkaista ei voi olla aikaisemmin valmiiksi saaduilla reiteillä, koska tällaisia reittejä ei ole. Askeleessa 4c huomataan, että mikäli asiakas 2 yhdistetään nykyisen jakeluauton reittiin, sen kapasiteetti ylittyy, sillä asiakkaiden 1, 2 ja 5 tilaukset tarvitsevat $3.7+3.2+3.5=10.4$ yksikköä jakeluauton kapasiteetista. Näin ollen siirrytään askeleeseen 4e ja otetaan säästölistalla seuraavana oleva alkio $s_{2,4}$ tarkasteluun. Askeleet 4b ja 4c voidaan ohittaa, koska kumpaakaan näistä asikkaista ei ole lisätty vielä millekkään olemassa olevalle reitille. Askeleessa 4d huomataan ettei asiakkaan 2 kapasiteetti edellenkään mahdu mukaan nykyiselle reitille. Näin ollen mikäli asiakkaat 2 ja 4 haluttaisiin yhdistää, pitää ottaa käyttöön uusi reitti. Tämä ei kuitenkaan onnistu tällä algoritmin versiolla, sillä peräkkäisellä ta-

valla ratkaisu muodostetaan yksi reitti kerrallaan. Näin ollen algoritmin askeleessa 4e jatketaan nykyisen reitin $0-1-5-0$ tarkastelua ja siirrytään säästölistalla seuraavana olevan alkion $s_{4,5}$ tarkasteluun. Askel 4b voidaan ohittaa, koska vain toinen asiakkaista on määrätty nykyiselle reitillä. Askeleessa 4c huomataan, että asiakkaan 4 lisääminen onnistuu nykyiseen reittiin, jolloin reitiksi saadaan $0-1-5-4-0$ ja jakeluauton kapasiteettia on käytetty 9.4 yksikköä. Reitiltä puuttuvien asiakkaiden kapasitettitarvetta katsottaessa huomataan, että tälle reitille ei voida lisätä yhtään enempää asiakkaita, koska muuten jakeluauton maksimikapasiteetti ylittyy. Kun viimeinen säästölistan alkio on tarkasteltu, askeleesta 4e siirrytään askeleeseen 4f ja ensimmäinen reitti on valmis. Koska askeleessa 4f huomataan, että asiakkaita on vielä palvelematta, niin siirrytään takaisin askeleeseen 4a, jossa valitaan tarkasteluun säästölistan ensimmäinen palvelematon alkio $s_{2,3}$ ja lähdetään muodostamaan uuden ja toistaiseksi tyhjän jakeluauton reittiä. Askeleet 4b ja 4c voidaan ohittaa, koska kumpaakaan asiakasparin asiakasta ei ole vielä millään reitillä. Siirrytään askeleeseen 4d, jossa kyseinen asiakaspari mahtuu nykyisen jakeluauton reitille. Tällöin reitti on $0-2-3-0$ ja kapasiteettia on käytetty 6.5 yksikköä. Askeleessa 4e ei ole valittavana enää seuraavaa alkiota, joten siirrytään askeleeseen 4f, jossa huomataan, että kaikki asiakkaat ovat jo jollakin reiteistä $0-1-5-4-0$ tai $0-2-3-0$ ja reittien yhteiskustannuksiksi saadaan $17+9+14+14=54$ yksikköä ja $20+27+9=56$ yksikköä. Näin ollen ratkaisun yhteiskustannus on 110.

Yllä olevasta esimerkistä nähdään, että saman algoritmin kahdesta versiosta saadaan hyvin erilaiset ratkaisut. Peräkkäisellä tavalla saadaan lisäksi usein parempi ratkaisu kuin rinnakkaisella tavalla. Kuitenkin rinnakkainen tapa on lähes aina nopeampi suorittaa kuin peräkkäinen, mutta esimerkiksi edellä se antaa huonomman ratkaisun. Tästä syystä, mitä enemmän asiakkaita tehtävässä on sitä tehokkaampaa on ratkaista ongelmaa rinnakkaisella tavalla.

3.2 Pyyhkäisy algoritmi

Pyyhkäisy algoritmin ovat kehittäneet Gillett ja Miller vuonna 1974 julkaisussa [10] ja se on myös erittäin käytetty algoritmi yleisessä ajoneuvojen reititysongelmassa. Tämä algoritmi hyödyntää varastolta laskettavia napakoordinaatteja asiakkaille, joiden avulla se muodostaa asiakkaista klustereita eli ryppäitä. Alkuperäinen Gillettin ja Millerin algoritmi sisältää monimutkaisia reittien parannuskeinoja, mutta tässä työssä keskitytään algoritmin yksinkertaiseen versioon. Näin algoritmin perusidea saadaan havainnollistettua helpommin ja monet tutkimukset käsittelevät ainoastaan tätä perusversiota.

Napakoordinaatit määrittävät kaksiulotteisessa koordinaatistossa origosta jollekin pisteelle kiertokulman $\delta \geq 0$ ja säteen eli etäisyyden $r \geq 0$. Yleisessä ajoneuvojen reititysongelmassa origona käytetään varastoa ja pisteinä asiakkaita, joille tavaroita toimitetaan. Jokainen piste voidaan esittää käyttäen napakoordinaatteja, sillä asiakkaan etäisyys r tiedetään tilauksien tiedoista. Esimerkiksi x-akselilla oleva piste y-akselin oikealla puolella on (r, $0°$). Lisäksi asiakas, jonka napakoordinaatit ovat (4, 90°), on etäisyydeltään neljän kilometrin päässä ja suoraan varastolta pohjoiseen.

Pyyhkäisy algoritmissa hyödynnetään 2-optimaalista hakua, jota on käytetty myös usein kauppamatkustajaongelman ratkaisemiseen. 2-optimaalinen haku on parantava hakuheuristiikka, jonka esitteli Croes vuonna 1958 [7]. Tämän algoritmin perusideana on uudelleen järjestellä reitti, jos se risteää itsensä kanssa. Yksittäisen reitin ei kuulu kulkea ristiin itsensä kanssa, sillä kyseinen reitti ei voi tällöin olla lyhin mahdollinen. Reitin uudelleen järjestely tehdään algoritmissa käymällä läpi jokaisella kierroksella kaikki tietynlaiset ennalta listatut reitin muutokset ja valitsemalla muutoksista se, joka antaa lopputuloksena kaikista lyhimmän reitin, joka on myös nykyistä reittiä parempi. Tätä parennettua reittiä yritetään seuraavalla kierroksella parantaa vastaavasti ja menetelmä päättyy vasta, kun nykyistä ratkaisua ei saada parannettua millään muutoksista..

Pyyhkäisy algoritmissa yhtenä päätösmuuttujana on ajoneuvojen määrä. Algoritmi aloitetaan laskemalla jokaiselle asiakkaalle napakoordinaatit. Tämän jälkeen asiakkaat järjestetään listaksi napakoordinaatin kiertokulman mukaan nousevassa järjestyksessä. Seuraavaksi tarkastellaan ensimmäisen ajoneuvon reittiä ja määrätään sille mahdollisimman monta asiakasta listalta. Asiakkaita määrätään ajoneuvolle listan järjestyksessä, kunnes ajoneuvon kapasiteetti on täynnä. Tämän jälkeen, otetaan tarkasteluun uuden ajoneuvon reitti. Tällä menettelyllä varmistetaan se, että jokainen ajoneuvo ajaa toivottavasti mahdollisimman lyhyen matkan. Uudelle ajoneuvolle jatketaan asiakkaiden määräämistä listasta järjestyksessä siitä kohtaa mihin edellisen ajoneuvon kohdalla jäätiin. Tätä samaa prosessia jatketaan, kunnes jokainen asiakas on määrätty jollekin jakeluautolle. Kun kaikki reitit ovat valmiita, parannetaan jokaista reittiä erikseen 2-optimaalisella haulla.

Alla on esitetty pyyhkäisy algoritmi tarkemmin.

Algoritmi 2. Pyyhkäisy algoritmi

Askel 0. *(Alustus)* Laske jokaisen asiakkaan napakoordinaatti (r, δ), kun varasto on asetettu origoksi. Listaa asiakkaat järjestyksessä kiertokulman mukaan pienimmästä suurimpaan.

Askel 1. Valitse uusi jakeluauto.

Askel 2. Lisää jakeluautolle asiakkaita, joita ei ole vielä määrätty millekkään jakeluautolle, kiertokulmalistan järjestyksessä niin kauan, kunnes jakeluauton kapasiteetti on täynnä. Kun yhtään jäljellä olevista asiakkaista ei enää pystytä lisäämään ajoneuvolle, palaa askeleeseen 1, ellei jokainen asiakas ole jo määrätty jollekin jakeluautolle.

Askel 3. Paranna jokaista jakeluauton reittiä 2-optimaalisella haulla. Laske kustannukset reiteille.

4 Yhteenveto

Tässä työssä on tarkasteltu yleistä ajoneuvojen reititysongelmaa ja esitelty sen ratkaisemiseksi kaksi ratkaisuheuristiikkaa. Ajoneuvojen reititysongelmassa on lista asiakkaista, joista jokaisen luokse täytyy kuljettaa tilaus kokonaisuudessaan yhdellä ajoneuvolla varastolta. Ajoneuvojen kapasiteetti ja enimmäismäärä ovat yleensä tiedossa, mutta enimmäismäärä voidaan myös määrätään tehtävää ratkaistaessa. Samaan aikaa kun jakeluautolle määrätään palveltavat asiakkaat, sille pyritään ratkaisemaan mahdollisimman lyhyt reitti varastolta määrätyille asiakkaille ja asiakkaiden välille, jotta tehtävän kokonaiskustannukset minimoituvat. Ajoneuvojen reititysongelma on yksi matemaattisen optimoinnin eniten tutkittuja ongelmia. Tehtävänä reititysongelmat ovat NP-vaikeita, eli niiden ratkaisemiseksi ei tunneta kuin eksponentiaalisen ajan vieviä algoritmeja. Yleinen ajoneuvojen reititysongelma on lisäksi staattinen ongelma, koska siinä oletetaan, että maailma eli parametrit eivät muutu.

Ajoneuvojen reititysongelmalle on kehitetty paljon erilaisia ratkaisuheuristiikkoja, joilla saadaan tehtävälle likimääräisiä ratkaisuja. Näin ollen ratkaisut eivät välttämättä ole globaaleja eivätkä edes lokaaleja optimeita. Työssä esitellään kaksi erilaista heuristiikkaa. Ensimmäinen on säästävä algoritmi, jonka kehittivät Clarke ja Wright vuonna 1964. Ideana algoritmissa on laskea eri asiakkaiden yhdistämisestä saatavat säästöt, muodostaa säästöistä lista suurimmasta pienimpään ja määrätä asiakkaat listan järjestyksen mukaan ajoneuvoille. Toisena ratkaisumenetelmänä työssä on esitelty pyyhkäisy algoritmi, joka muodostaa ensin varastolta jokaiselle asiakkaalle napakoordinaatit. Napakoordinaatit muodostuvat kiertokulmasta ja etäisyydestä varastoon. Asiakkaat järjestetään kiertokulman mukaan listaan pienimmästä suurimpaan. Tämän jälkeen asiakkaita määrätään jakeluautolle listan järjestyksessä ja tarkasteltava asiakas otetaan aina mukaan, jos jakeluauton kapasiteetti ei ylity ja asiakas ei vielä ole millään muodostetuista jakeluauton reiteistä. Kun koko lista on käyty läpi ja osa asiakkaista on vielä palvelematta, niin otetaan ratkaisuun mukaan uusi tyhjä jakeluauto ja sen asiakkaat saadaan määrättyä vastaavasti käymällä taas

läpi koko säästölista alusta alkaen. Menetelmän idea on siis, että yhden jakeluauton tarvitsee käydä ainoastaan yhdellä sektorilla olevien asiakkaiden luona.

Viitteet

[1] L. Biggs, E. Lloyd, R. Wilson: *Graph Theory*. Clarendon Press, Oxford, 1986.

[2] L. Bodin, B. Golden: Classification in vehicle routing and scheduling. *Networks*, vol. 11, no. 2, s. 97-108, 1981.

[3] M. E. Bruni, S. Khodaparasti, I. Martínez-Salazar, S. Nucamendi-Guillén: The multi-depot k-traveling repairman problem. *Optimization Letters*, vol. 16, no. 9, s. 2681-2709, 2022.

[4] M. A. Burhanuddina, M. Mutara, A. Hameeda, N. Yusofa, H. Mutasharb: An efficient improvement of ant colony system algorithm for handling capacity vehicle routing problem. *International Journal of Industrial Engineering Computations*, vol. 11, no. 4, s. 549-564, 2020.

[5] O. Cosma, C. Sabo, C. P. Sitar, P. Pop: A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research*. Vol. 314, No. 3, s. 819-835, 2023.

[6] G. Clarke, J. W. Wright: Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, vol. 12, no. 4, s. 568-581, 1964.

[7] G. Croes: A method for solving traveling-salesman problems. *Operations research*, vol. 6, no. 6, s. 791-812, 1958.

[8] G. B Dantzig, T. H. Ramser: The truck dispatching problem. *Management Science*, vol. 6, no. 1, s. 80-91, 1959.

[9] E. Delimpasi, Y. Marinakis, I.-D. Psychas: Hybrid evolutionary algorithms for the multiobjective traveling salesman problem. *Expert Systems with Applications*, vol. 42, no. 22, s. 8956-8970, 2015.

[10] B. Gillet, L. Leland, J. Johanson: *Vehicle dispatching — Sweep algorithm and extensions*. Springer, Dordrecht, 1979.

[11] M. Hassan, S. Ullah, I. Khan, S. Hussain Shah, A. Salam, A. Ullah Khan: Unmanned aerial vehicles routing formation using fisheye state routing for flying ad-hoc networks. Konferenssiesite, Association for Computing Machinery, 2021.

[12] M. M. Mäkelä: Matemaattinen Optimointi I. Luentomoniste, Turun Yliopisto, 2020.

[13] S. Noor, S. Ghannadpour, K. Ghoseiri, R. Tacakkoli-Moghaddam: A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application. *Applied Soft Computing*, vol. 14, no. 1568-4946, s. 504-527, 2014.

[14] T. Perasto: Ajoneuvoreititysongelman analysointi ja ratkaiseminen geneettisellä algoritmilla. Diplomityö, Tampereen yliopisto, 2021.

[15] V. Pillac, M. Gendreau, C. Guéret, A. L. Medaglia: A review of dynamic vehicle routing problems. *European Journal of Operational Research*, vol. 225, no. 1, s. 1-11, 2013.

[16] R. Silva, S. Urrutia: A General VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*. vol. 7, no. 4, s. 203-211, 2010.