Tampereen yliopisto

Eppu Hassinen

AJONEUVOREITITYSONGELMAN SOVELTAMINEN JOULUPUKKIEN REITTEIHIN

Kandidaatintutkielma
Informaatioteknologian ja viestinnän tiedekunta
Syyskuu 2023

TIIVISTELMÄ

Eppu Hassinen: Ajoneuvoreititysongelman soveltaminen joulupukkien reitteihin
Kandidaatintutkielma
Tampereen yliopisto
Tietotekniikan tutkinto-ohjelma
Syyskuu 2023

Tässä kandidaatintutkielmassa tutkitaan joulupukkipalvelun joulupukkien reittejä, mitä ajoneuvoreititysongelman muunnosta ne parhaiten kuvastavat ja millaisella algoritmilla niille voidaan löytää tehokkaat reitit. Yksi operaatiotutkimuksessa eniten tutkituista kombinatorisista optimointiongelmista on ajoneuvoreititysongelma. Ongelmassa pyritään löytämään mahdollisimman lyhyet reitit ajoneuvoille, jotka lähtevät varastolta, palvelevat asiakkaita ja palaavat takaisin lähtöpisteeseen. Ajoneuvoreititysongelma ja kaikki sen versiot ovat NP-vaikeita eli niille ei pystytä löytämään realistisessa ajassa täydellisiä ratkaisuja. Ajoneuvoreititysongelmien ratkaisemisessa hyödynnetään heuristisia menetelmiä, joilla pyritään löytämään lähes optimaalinen ratkaisu murto-osassa siitä ajasta, mitä optimaalisen ratkaisun löytämiseen kuluisi.

Tutkimus jakaantuu kahteen osioon. Teoriaosiossa käydään läpi kirjallisuudessa esiintyviä ajoneuvoreititysongelman muunnoksia ja niiden ratkaisuun hyödynnettyjä algoritmeja. Tutkittuja ajoneuvoreititysongelman muunnoksia ovat esimerkiksi avoimet reitit, joissa ei palata lähtöpisteeseen, asiakkaiden palvelemiseen varatut aikaikkunat sekä useat varastot. Nämä muunnokset yhdistämällä saatiin avoin useiden varastojen aikaikkunallinen ajoneuvoreititysongelma, jonka havaittiin kuvastavan työn joulupukkireititysongelmaa. Algoritmeja löydettiin useita, mutta niistä valittiin työhön tarkasteltavaksi kolme eniten tuoreessa kirjallisuudessa esiintynyttä algoritmia: geneettinen algoritmi, vaihtelevan naapuruston hakualgoritmi sekä muurahaispesäoptimointi. Käytännön osiossa luotiin ohjelma joulupukkireititysongelman ratkaisemiseksi.

Joulupukkireititysongelman tarkempaan määrittelyyn eivät riitä pelkästään useat varastot, reittien avoimuus ja aikaikkunat, mutta tässä tutkimuksessa keskityttiin kuitenkin vain niillä määritellyn joulupukkireititysongelman ratkaisemiseen. Ongelman ratkaiseminen toteutettiin tekemällä C++-ohjelmointikielellä muurahaispesäoptimointiin perustuva ohjelma. Ohjelmalla saa laskettua reitit tutkimuksen yksinkertaistetulle ongelmalle. Jatkotutkimusta tarvitaan ohjelman hyödyntämiseen käytännön tilanteessa. Ohjelma hyödyntää osoitetietojen sijaan pelkkiä koordinaatteja. Parannusehdotuksia ja jatkokehitysideoita esitellään kandidaatintutkielman lopussa.

Avainsanat: VRP, ajoneuvoreititysongelma, MDOVRPTW, aikaikkunat, muurahaispesäoptimointi, ACO

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

ACO	Ant Colony Optimization, Muurahaispesäoptimointi
MDOVRP	Multi-Depot Open Vehicle Routing Problem, Avoin ajoneuvoreititysongelma useilla varastoilla
MDOVRPTW	Multi-Depot Open Vehicle Routing Problem with Time Windows, Avoin ajoneuvoreititysongelma useilla varastoilla ja aikaikkunoilla
MDVRP	Multi-Depot Vehicle Routing Problem, Ajoneuvoreititysongelma useilla varastoilla
MDVRPTW	Multi-DepotVehicle Routing Problem with Time Windows, Ajoneuvoreititysongelma useilla varastoilla ja aikaikkunoilla
OVRP	Open Vehicle Routing Problem, Avoin ajoneuvoreititysonglma
TSP	Travelling Salesman Problem, Kauppamatkustajan ongelma
VNS	Variable Neighbourhood Search, Vaihtuvan naapuruston hakualgoritmi
VRP	Vehicle Routing Problem, Ajoneuvoreititysongelma
VRPTW	Vehicle Routing Problem with Time Windows, Ajoneuvoreititysongelma aikaikkunoilla

1. JOHDANTO

Ajoneuvoreititysongelmaa ja sen lukuisia muunnoksia on tutkittu paljon. Ongelmassa pyritään löytämään mahdollisimman lyhyt tai muulla tavalla kustannustehokas reitti useille ajoneuvoille, jotka vierailevat asiakkaiden luona. Asiakkailla ja ajoneuvoilla voi olla erilaisia rajoitteita ja määritteitä. Näistä ehdoista saadaan lukematon määrä erilaisia ongelmia, joista jokaisen ratkaisemisessa on omat haasteensa.

Optimaalinen reitti on teoriassa suoraviivaista laskea kaikille ongelmille, mutta laskentajoukon kasvaessa ei nykyisten tietokoneiden laskentateho riitä ratkaisuiden saamiseen kohtuullisessa ajassa. Aikaa on pyritty lyhentämään erilaisilla optimoinneilla ratkaisualgoritmeissa. Tehokkaimmat algoritmit ovat metaheuristisia tai heuristisia algoritmeja, jotka pyrkivät löytämään riittävän tehokkaan reitin optimaalisen reitin sijaan.

Joulupukkireititysongelma on käytännön ajoneuvoreititysongelma, partiolippukuntani varainhankintakeino, jossa on viime vuosina ollut suuri työ luoda joulupukkien reitit manuaalisesti. Joulupukkireititysongelman ratkaisussa jokainen joulupukki ajaa omalla ajoneuvollaan mahdollisimman lyhyen reitin ja jokaisen asiakkaan luona käydään. Joulupukkireititysongelmassa on erityisiä piirteitä verrattuna yleisimpiin ajoneuvoreititysongelman muunnoksiin. Joulupukkivierailuiden kaikkia asiakkaita ei tiedetä kerralla, mutta asiakkaille tulisi saada ilmoitettua vierailuajankohta mahdollisimman pian tilauksen tekemisen jälkeen. Tässä tutkielmassa tutkin ratkaisualgoritmia vain tilanteelle, jossa lasketaan reitit kerralla. Luotuihin reitteihin ei lisätä yksittäisiä asiakkaita ohjelmallisesti.

Tässä kandidaatintyössä tutkin erilaisia ajoneuvoreititysongelmien muunnoksia sekä niiden ratkaisualgoritmeja ja pyrin löytämään joulupukkireititysongelmaan parhaan mahdollisen ratkaisualgoritmin. Osana kandidaatintyötä toteutin ohjelman ratkaisemaan yksinkertaistetun version joulupukkireititysongelmasta ja pohdin sille parannusehdotuksia. Toteutin ohjelman hyödyntäen tutkittuja muurahaispesäoptimointialgoritmeja ohjelmointikielellä C++. Ohjelma löytyy GitHubista [7].

Kirjallisuuskatsauksen tutkimusmenetelmä on ollut prosessi, jossa aiheesta vähän tietäneenä lähdin etsimään tietoa hakusanoilla Traveling Salesperson's Problem sekä TSP. Löydettyjen lähteiden avulla pystyin rajaamaan haun Ajoneuvoreititysongelmaan VRP. Toistin prosessia vielä muutamia kertoja ja päädyin käyttämään hakusanoja: "multi-depot open vehicle routing problem" ja algorithm. Kun hakutuloksista otettiin vain uusimmat

teokset, saatiin jo hyvän kokoinen joukko lähteitä. Tutkielman jatkuessa etsin teoksia erilaisista algoritmeista ja ajoneuvoreititysongelmien muunnoksista, joita työn edetessä tuli vastaan.

Luvussa 2 esitellään erilaisia ajoneuvoreititysongelman muunnoksia. Luvussa 3 käydään läpi erilaisia kirjallisuudessa VRP:n ratkaisemiseen käytettyjä algoritmeja. Luvussa 4 esitellään tässä tutkielmassa käsiteltävä uusi ajoneuvoreititysongelma ja luvussa 5 esitellään yksinkertaistetun joulupukkireititysongelman ratkaisemiseen luotu ohjelma. Lopuksi yhteenvedossa 6 pohditaan ohjelman toimivuutta sekä jatkokehitysideoita.

2. AJONEUVOREITITYSONGELMA

Tässä luvussa esitellään ajoneuvoreititysongelma ja sen erilaisia kirjallisuudessa esiintyviä muunnoksia. Muunnoksia on muitakin kuin tässä luvussa mainitut, mutta ne riittävät joulupukkireititysongelman määrittämiseen luvussa 4.

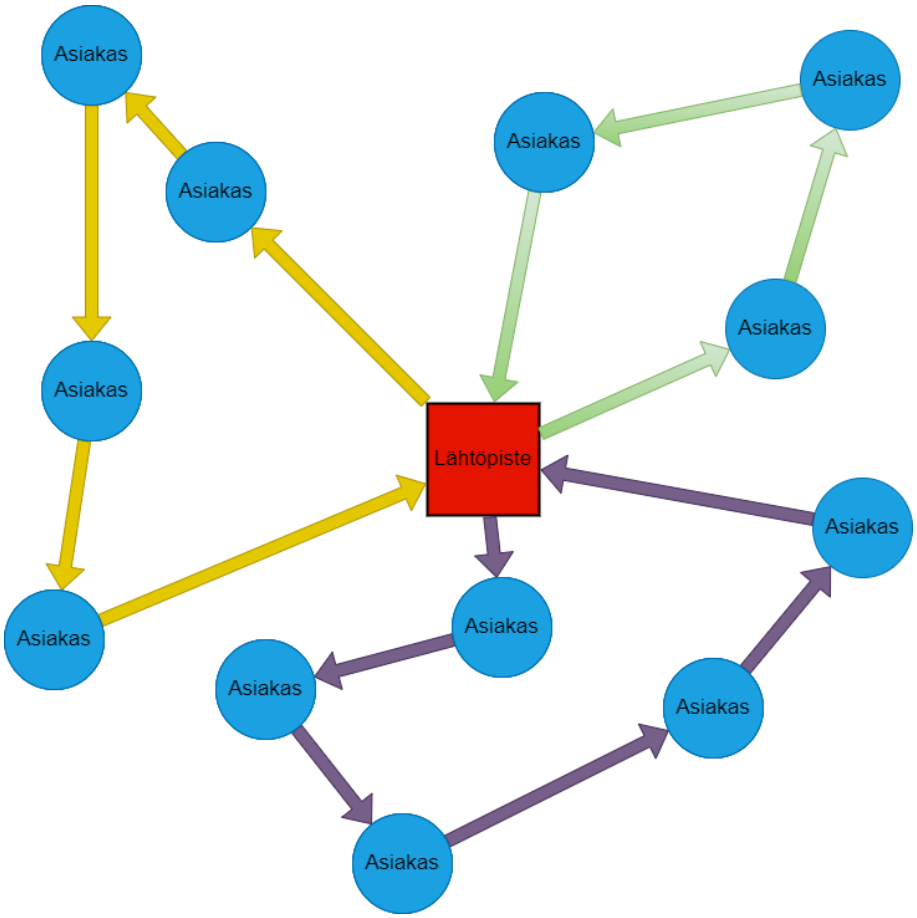
Ajoneuvojen reititysongelmat ovat paljon tutkittuja ongelmia. Monenlaiset yritykset ja muut toimijat toimittavat tuotteita sekä palveluita asiakkaille. Toimituksesta halutaan saada mahdollisimman kustannustehokasta, joten ajoneuvojen reiteistä halutaan saada mahdollisimman lyhyitä [16, s. 6-9].

Kaikki ajoneuvoreititysongelmien versiot ovat NP-kovia, ja niille voidaan järkevässä ajassa laskea paras ratkaisu vain pienissä tapauksissa. Realistisen kokoisille tapauksille on viime vuosikymmenten aikana kehitetty useita tehokkaita heuristiikkoja [2, luku 1].

Vehicle Routing Problem eli VRP on yksi operaatiotutkimuksessa eniten tutkituista kombinatorisista optimointiongelmista [2, luku 1]. Dantzig ja Ramser [6] julkaisivat ongelmasta artikkelin jo vuonna 1959. VRP on ajoneuvoreititysongelmista yksinkertaisin [1, luku 1]. VRP:ssä ajoneuvot lähtevät kaikki samasta varastosta, ja käyvät asiakkaiden luona. Jokaisen asiakkaan luona käy yksi ajoneuvo, ja ajoneuvot palaavat viimeisen asiakkaan jälkeen takaisin varastolle. [16, s. 6-7] Kuvassa 2.1 on esitetty eräs yksinkertaisen VRP:n ratkaisu.

Mikäli ajoneuvoja on vain yksi, puhutaan kauppamatkustajan ongelmasta eli TSP:stä. Kauppamatkustajan ongelmassa kauppamatkustaja käy jokaisessa kaupungissa ja palaa takaisin lähtökaupunkiin mahdollisimman lyhyttä reittiä [16, s. 1]. Kaikissa ajoneuvoreititysongelmien muunnoksissa ajoneuvot voivat olla joko homogeenisia tai heterogeenisia. Homogeenisillä ajoneuvoilla kaikki ominaisuudet ovat samat. Heterogeenisillä ajoneuvoilla ajoneuvokohtaiset rajoitteet kuten kapasiteetti, poikkeavat toisistaan [1, luku 1]. Ajoneuvojen kapasiteetteja käsitellään luvussa 2.4.

Seuraavaksi esitetään erilaisia kirjallisuudessa tutkittuja lisävaatimuksia ajoneuvoreititysongelmille. Niitä voi yhdistellä keskenään käyttötilanteen mukaan.



Kuva 2.1. *Yksinkertaisen VRP:n ratkaisu.*

2.1 Avointen reittien ajoneuvoreititysongelma

Avoimessa ajoneuvoreititysongelmassa (Open Vehicle Routing Problem, OVRP) tuotteiden jakelu on ulkoistettu jollekin ulkopuoliselle toimijalle. Tällöin ei tarvitse laskea reittiä takaisin varastolle, sillä ajoneuvojen reittien jatkaminen viimeiseltä asiakkaalta eteenpäin jää jakeluyrityksen vastuulle [4][15][16]. OVRP:tä on tutkittu ensimmäisen kerran vuonna 1981. Siitä lähtien OVRP:tä ja sen muunnoksia on tutkittu laajasti nykypäivään saakka [5, luku 1].

Samat säännöt pätevät esimerkiksi myös bussilinjoihin, junalinjoihin sekä lentoliikenteeseen mikäli alku- ja päätepiste eroavat toisistaan. Nämä esimerkit kuvaavat paremmin OVRP:n todellista käyttöä [4, luku 1].

2.2 Aikaikkunallinen ajoneuvoreititysongelma

Aikaikkunallinen ajoneuvoreititysongelma (Vehicle Routing Problem with Time Windows, VRPTW) rajoittaa ajoneuvoreititysongelman reittejä siten, että jokaisella asiakkaalla on aikaikkuna. Ajoneuvon pitää siis käydä jokaisen asiakkaan luona ennalta määrätyn asiakaskohtaisen minimi- ja maksimiajan välissä. Aikaikkunoita on kahdenlaisia: pehmeitä ja

kovia [4, luku 2][16, s. 17].

Käytettäessä pehmeitä aikaikkunoita voi ajoneuvo saapua asiakkaan luo myöhässä, mutta siitä sakotetaan. Sakon suuruus on määriteltävissä käyttötilanteen mukaan. Kovia aikaikkunoita käytettäessä asiakkaan luo myöhässä saapuminen tekee ratkaisusta käyttökelvottoman. Pehmeistä aikaikkunoista saadaan helposti kovia nostamalla myöhästymissakon suuruus todella suureksi [16, s. 17-19].

2.3 Useiden varastojen ajoneuvoreititysongelma

Useiden varastojen ajoneuvoreititysongelmassa (Multi-depot Vehicle Routing Problem, MDVRP) vähennetään jakelun kustannuksia siten, että kaikki ajoneuvot eivät välttämättä lähde samalta varastolta. Varastojen paikat ovat määritelty etukäteen ja kaikki ajoneuvot lähtevät sellaisesta varastosta, josta lähtiessä reitistä tulee mahdollisimman lyhyt. Kaikista varastoista ei välttämättä lähde ajoneuvoja [16, s. 14-16][11, luku 2].

Ajoneuvot voidaan määrittää joko palaamaan takaisin lähtövarastoonsa tai johonkin muuhun varastoon, johon reitti on lyhyempi [11, luku 2]. Lähtökohtaisesti ongelmien ratkaisut eivät ota kantaa ajoneuvojen määrään, mutta esimerkiksi Bezerran ym. algoritmi [4, luku 5.7] painottaa ratkaisussaan ajoneuvojen määrän minimointia reittien pituuksien lisäksi. MDVRP:stä on avoimien reittien versio MDOVRP, joka yhdistää avoimet reitit useiden varastojen ongelmaan [5, luku 1].

2.4 Kapasiteettirajoitettu ajoneuvoreititysongelma

Kapasiteetillinen ajoneuvoreititysongelma (Capacitated Vehicle routing Problem, CVRP) ottaa huomioon ajoneuvojen kyvyn kuljettaa tavaroita. Ajoneuvo ei voi kuljettaa kerralla kapasiteettiaan enempää tavaraa. Asiakkaat voivat tarvita tavaraa eri määriä, joten kapasiteetti ei suoraan kerro montako asiakasta ajoneuvo voi palvella yhdellä reitillä [16, s. 9-11].

Mikäli ajoneuvojen yhteistilavuus on liian pieni, kaikkia asiakkaita ei välttämättä pystytä palvelemaan. Tällaisiin tilanteisiin on kehitetty VRP:n muunnoksia [16, s. 11]:

- VRPMT (VRP with Multiple Trips): Usean reitin sallivassa VRP:ssä ajoneuvot voivat palata varastoon ja suorittaa useita reittejä.
- VRPP (Vehicle Routing Problem with Profits): Tuottoihin perustuvassa VRP:ssä jätetään vähiten tuottavat asiakkaat palvelematta kapasiteetin ylittyessä.

Näitä muunnoksia ei tässä tutkielmassa käsitellä enempää.

3. TUTKITUT VRP:N RATKAISUALGORITMIT

Ajoneuvoreititysongelmien ratkaisemiseen on luotu useita heuristisia sekä metaheuristisia algoritmeja. Viime vuosina eniten kirjallisuudessa esiintyvistä algoritmeista avataan seuraavaksi kolmea eniten esiintynyttä:

- Muurahaispesäoptimointi (Ant colony optimization) [10, luku 1]
- Geneettinen algoritmi (Genetic algorithm) [16, s. 20-33]
- Vaihtelevan naapuruston hakualgoritmi (Variable neighbourhood search) [4, luku 5][15]

Heuristiset ratkaisut pyrkivät tasapainottelemaan ratkaisun oikeellisuuden ja sen laskemisen tehokkuuden välillä. Useiden ratkaisujen tehokkuuden on huomattu riippuvan hyvin paljon siitä, miten hyvin algoritmi pystyy tunnistamaan, kuinka hyvä saatu ratkaisu on, tietämättä optimaalista ratkaisua. Arnold ja Sörensen tutkivat artikkelissaan [2] hyvän ratkaisun piirteitä. He saivat selville, että lähes optimaalisen ratkaisun voi tunnistaa suuressa osassa tapauksia tietämättä optimaalista ratkaisua. Lähes optimaalinen VRP:n ratkaisu erottuu huonoista ratkaisuista esimerkiksi siten, että reitti on kompakti ja reittien risteyksiä on vähän. Vaikka artikkelissa on tutkittu vain VRP:n ratkaisuja, siinä mainitaan löydön olevan laajennettavissa muihinkin ongelman variaatioihin.

3.1 Muurahaispesäoptimointi

Muurahaispesäoptimointi (Ant Colony Optimization, ACO) on osa parviälyn (swarm intelligence) tutkimusta. Siinä sovelletaan muurahaispesän toimintaperiaatteita ratkaistaessa monimutkaisia kombinatorisia ongelmia. Muurahaispesäoptimointi simuloi muurahaisten luontaista kykyä löytää tehokkaimmat reitit pesiltä ravintolähteiden luokse. Muurahaisten päätöksentekoa simuloidaan virtuaalisten muurahaisten avulla, mitä on visualisoitu Sebastian Laguen videossa Coding Adventure: Ant and Slime Simulations. [12] Muurahaiset jättävät peräänsä kemiallista ainetta, jota kutsutaan feromoniksi. Muurahaiset hyödyntävät muiden muurahaisten feromonireittejä omassa reitinvalinnassaan. Reitin valinta on pseudo-satunnaista, sillä muurahaiset kulkevat satunnaista reittiä kuitenkin painottaen valinnoissaan edellisten muurahaisten feromonijälkiä. Löytäessään ravintolähteen muurahainen palaa pesään. Tällöin ravintolähteelle kulkee kaksinkertainen feromonijälki, joka

edesauttaa yhä useamman muurahaisen reitinvalinnan päätyvän kyseiselle ravintolähteelle. Reitin löytäminen ei vielä takaa reitin olevan lyhyin. Lyhyimpien reittien feromonijäljet vahvistuvat tiheimmin, joten lopulta lähes kaikki muurahaiset valitsevat lyhyimmän reitin. Muurahaispesäoptimoinnin vahvuus on sen kyky löytää uusia parempia reittejä, sillä jokainen muurahainen kulkee omaa satunnaista reittiään. [3, luku 3]

Kokeelliset tulokset osoittavat muurahaispesäoptimoinnin olevan kilpailukykyinen muiden tutkittujen VRP:n ratkaisualgoritmien joukossa [3, luku 5.2]. Yu ja Li ovat artikkelissaan [17] soveltaneet ACO:ta aikaikkunallisen VRP:n ratkaisuiden löytämiseksi. Useiden varastojen aikaikkunalliselle ajoneuvoreititysongelmalle on myös etsitty ratkaisuja ACO:n avulla [13]. Muurahaispesäoptimoinnissa haasteena on parametrien säätäminen, jotta algoritmi toimii halutussa tilanteessa parhaalla mahdollisella tavalla [8, luku 1].

3.2 Geneettinen algoritmi

Geneettinen algoritmi perustuu luonnon evoluution tavoin siihen, että se valitsee populaatiosta parhaat ratkaisut ja luo uuden populaation niiden perusteella. Jokaisen populaation kaksi parasta ratkaisua valitaan vanhemmiksi seuraavalle populaatiolle. [16, s. 20-33]

Populaatio alustetaan yksilöillä. Tämän jälkeen yksilöille arvioidaan kuntoarvot (fitness-arvot), joka määritellään ongelmalle yksilöllisellä fitness-funktiolla. Populaation kaksi parhaan kuntoarvon saanutta ratkaisua valitaan vanhemmiksi, joiden perusteella luodaan uusi populaatio. Uuden populaation jäsenet ovat risteytyksiä vanhemmistaan tai toisen vanhemman tarkkoja kopioita. Yksilöillä on lisäksi jokin ennalta määritelty todennäköisyys mutatoitua. Geneettinen algoritmi luo uusia populaatioita iteratiivisesti, kunnes ennalta määritellyt ehdot algoritmin suorittamisesta täyttyvät. [16, s. 20-33]

3.3 Vaihtelevan naapuruston hakualgoritmi

Vaihtelevan naapuruston hakualgoritmissa hyödynnetään lokaalien ratkaisujen etsimistä vaihtuvassa naapurustossa. Mladenović ja Hansen [14] käsittelivät algoritmia ensimmäistä kertaa artikkelissaan vuonna 1997. Algoritmissa jaetaan asiakkaat naapurustoihin jollakin ennalta määritellyllä perusteella. Naapurusto tarkoittaa VRP:n tapauksessa esimerkiksi yhden ajoneuvon asiakkaita. Jaon jälkeen hyödynnetään yksittäisten reittien tehokkaimman järjestyksen laskemisen tehokkaita lokaaleja ratkaisuita sekä satunnaista reittien sekoittamista. Sekoittamisella pyritään pakenemaan lokaaleista minimeistä.

Algoritmin tehokkuus perustuu satunnaisuuden ja sääntöjen yhteisvaikutukseen. Siinä valitaan satunnaisesti jokin operaatio, joka vaihtaa asiakkaan järjestystä reitillä, asiakkaalla vierailevaa ajoneuvoa tai muuta osaa reitistä. Tämän jälkeen katsotaan paransiko muutos alkuperäistä reittiä. Muutos jää voimaan vain tilanteissa, joissa reitti muuttuu tehokkaammaksi muutoksen ansiosta. [14, luku 2]

4. JOULUPUKKIREITITYSONGELMA

Joulupukkireititysongelma kuvastaa tutkituista ongelmista parhaiten useiden varastojen avointa aikaikkunallista ajoneuvoreititysongelmaa eli MDOVRPTW:tä, sillä joulupukkireititysongelman rajoitteita ovat:

1. Multi-depot: Joulupukit lähtevät omista kodeistaan.
2. Open: Joulupukin ei tarvitse palata takaisin lähtöpisteeseen.
3. VRP: Reittien tulee olla mahdollisimman lyhyet.
4. Time windows: Joulupukeilla on tietty aika, jolloin asiakkaan luona pitää olla.

Joulupukit lähtevät jouluaattona omista kodeistaan, joten jokaisesta lähtöpaikasta saisi lähteä vain yksi joulupukki. Tämän toteutus on jätetty tutkielmasta pois, eli joulupukkien lähtöjen määrää lähtöpisteistä ei ole rajoitettu, eikä jokaisesta lähtöpisteestä tarvitse lähteä joulupukkia.

Joulupukit palaavat tehdyn reitin jälkeen todennäköisesti takaisin kotiin, joten kokonaisajomatkasta saataisiin lyhyempi jättämällä avoimuus pois vaatimuksista [katso 4, table 7-8]. Tässä tutkielmassa on kuitenkin päätetty painottaa itse reittien tehokkuutta, jotta päivän aikana ehditään palvella mahdollisimman monta asiakasta. Siksi reitti viimeiseltä asiakkaalta takaisin lähtöpisteeseen jätetään huomiotta joulupukkireititysongelmassa.

Joulupukkivierailujen määrää on tähän asti rajoittanut joulupukkien määrä. Palvelun kokonaiskapasiteetti on ollut viime vuodet noin 120 vierailua yhteensä kymmenellä joulupukilla. Joulupukkivierailutilauksia tulee palvelun avautuessa marraskuussa noin puolet. Loput tilaukset tulevat tasaiseen tahtiin jouluaattoon asti. Marraskuussa tilanneille asiakkaille pyritään lukitsemaan oma vierailuajankohta joulukuun alkuun mennessä ja lopuille tilausta seuraavan arkipäivän aikana. Suurin työ manuaalisesti reittien laskemisessa on marraskuun tilauksille reittien luominen. Loput tilaukset voi yksi kerrallaan sovittaa jo tehdyille reiteille manuaalisesti.

Joulupukkiprojekti koostuu joka vuosi seuraavista osista:

1. Kesäkuu-lokakuu: Projektin suunnittelu ja joulupukkien rekrytointi.
2. Marraskuu: Tilauksia otetaan vastaan.
3. Marraskuun loppu: Lasketaan marraskuun tilauksille reitit.
4. Joulukuu: Sovitetaan uusille tilauksille vierailuajat jo laskettuihin reitteihin.
5. Jouluaatto: Pukit vierailevat asiakkailla suunnitellusti.
6. Tammikuu: Palautteen analysointi.

Tässä tutkielmassa keskitytään vain marraskuun tilausten reittien luomiseen.

Ainoa löydetty kirjallisuuslähde, joka käsittelee suoraan MDOVRPTW:tä, on Sinaide Nunes Bezerra ym. tekemä artikkeli [4]. Kaikkia MDOVRPTW:n rajoitteita on erikseen käsitelty kirjallisuudessa useasti [15][13][17]. Lähes kaikki paljon tutkitut algoritmit ovat sovellettavissa joulupukkireititysongelman ratkaisuun.

Vertailtaessa Xun ja muiden [13, luku 3] sekä Bezerran ja muiden [4, luku 5] muurahaispesäoptimointiin ja vaihtelevan naapuruston hakualgoritmiin perustuvia VRP:n ratkaisualgoritmeja huomataan, että muurahaispesäoptimointi on ohjelmallisesti yksinkertaisempi joulupukkireititysongelman tilanteessa. Muurahaispesäoptimointi vaatii parametrien säätämisen toimiakseen halutulla tavalla, mutta se vaikuttaa silti parhaalta algoritmilta lähteä ratkaisemaan joulupukkireititysongelmaa. Ongelman ratkaisua käsitellään luvussa 5.

5. TOTEUTETTU OHJELMA

Tässä tutkielmassa tehtiin ohjelma [7], joka laskee ratkaisun MDOVRPTW:lle, eli yksinkertaistetulle joulupukkireititysongelmalle, joka on kuvattu luvussa 4. Ohjelman algoritmi toteutettiin Ma ym. algoritmin [13, luku 3] pohjalta. Ohjelmointikielenä oli C++ ja reittien visualisointi tehtiin The Mathworks Inc:in ohjelmalla Matlab [9].

Ohjelma etsii heuristisen ratkaisun, jossa kaikkien asiakkaiden luona käydään. Joulupukkien ja lähtöpaikkojen määriä ei ole rajoitettu. Algoritmi lisää joulupukkien määrää kunnes kaikkien asiakkaiden luona on vierailtu. Joulupukkien nopeus on asetettu olemaan 20 yksikköä / minuutissa. Mikäli jokin asiakaspiste on niin kaukana, että sinne ei ehdi aikaikkunan aikana vaikka lähtisi lähtöpisteestä asiakkaalle suoraan, antaa ohjelma virheilmoituksen ja lopettaa. Ohjelma tallentaa löydetyt reitit csv-tiedostoon. Tiedoston jokainen rivi on yhden joulupukin reitti. Ensimmäinen sarake on lähtöpisteen järjestysluku ja loput ovat asiakkaiden järjestyslukuja vierailujärjestyksessä. Algoritmin parametrit testattiin systemaattisesti ja säädettin parhaiden tulosten saamiseksi kokeilemalla erilaisia arvoja. Säädetyt parametrien arvot ovat liitteessä A.3.

Algoritmissa luodaan feromonikartta, johon tallennetaan feromonijäljet jokaisesta solmusta kaikkiin muihin solmuihin. Tätä karttaa hyödynnetään seuraavan solmun valinnassa. Algoritmia suoritetaan silmukassa ja vahvistetaan feromonijälkiä lyhimpien reittien mukaisesti. Feromonijälkiä päivitetään sekä iteraatioiden sisällä että iteraatioiden välillä. Iteraatioiden sisäiset päivittämiset auttavat algoritmia löytämään sopivia reittejä tehokkaammin. Iteraatioiden väliset päivittämiset taas painottuvat lyhimmän reitin vahvistamiseen. Feromonikarttaa hyödynnetään reitin seuraavan solmun valinnassa. Algoritmin sisällä seuraava solmu valitaan joko satunnaisesti tai heuristisen arvon perusteella. Heuristinen arvo lasketaan kullekin solmulle hyödyntäen matkaa nykyisestä solmusta, vierailun aikaikkunaa sekä feromonijälkeä kyseisellä matkalla määrtiteltyjen parametrien mukaisesti. Satunnaisuus on painotettu samoilla heuristisilla arvoilla. Algoritmin yleinen rakenne on esitetty algoritmissa 1.

```
for Iteraatioiden määrä do
    for Muurahaisten määrä do
        reitti = [];
        retti.ensimmäinen = satunnainen lähtöpaikka;
        while Solmuja jäljellä do
            Random: satunnainen tai heuristinen;
            if satunnainen then
                reitti.seuraava = painotetusti satunnainen solmu;
            end
            if laskennallinen then
                reitti.seuraava = heuristisesti paras seuraava solmu;
            end
            if Ei mahdollisia seuraavia solmuja then
                reitti.seuraava = satunnainen lähtöpaikka;
            end
        end
        päivitä feromonikartta;
        if reitti < lyhin reitti then
            lyhin reitti = reitti;
        end
    end
    päivitä feromonikartta;
end
return Lyhin reitti;
```

Algoritmi 1: Algoritmin yleisen rakenteen pseudokoodi.

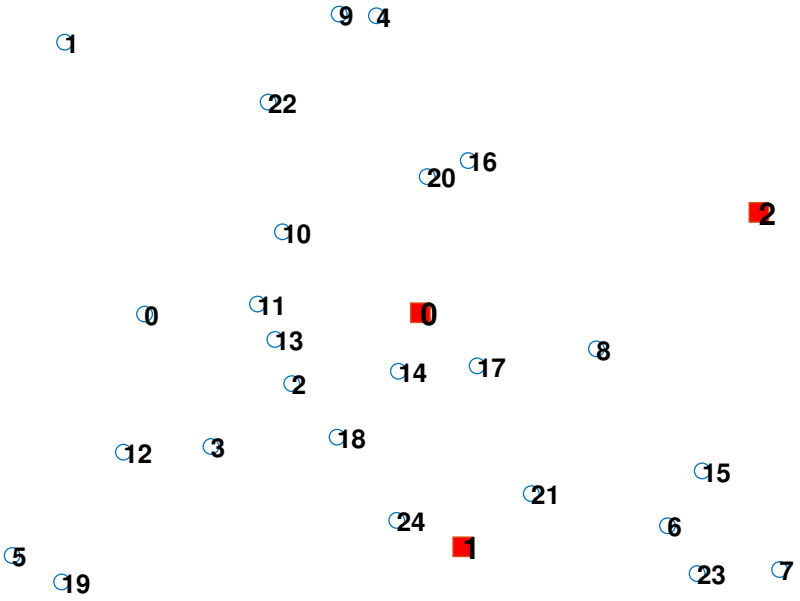
5.1 Esimerkkiongelma

Ohjelman toimintaa esitetään esimerkkitapauksella, jonka arvot ovat liitteessä A. Ohjelma lukee csv-tiedostoista tiedot asiakkaista ja lähtöpisteistä. Asiakkaista tarvittavia tietoja ovat x- ja y-koordinaatit sekä palvelun aikaikkunan alku- sekä loppuajat. Ajat ovat lukuja, jotka kertovat ajan minuutteina lähtöhetkestä eteenpäin. Lähtöpisteistä tarvittavia tietoja ovat koordinaatit.

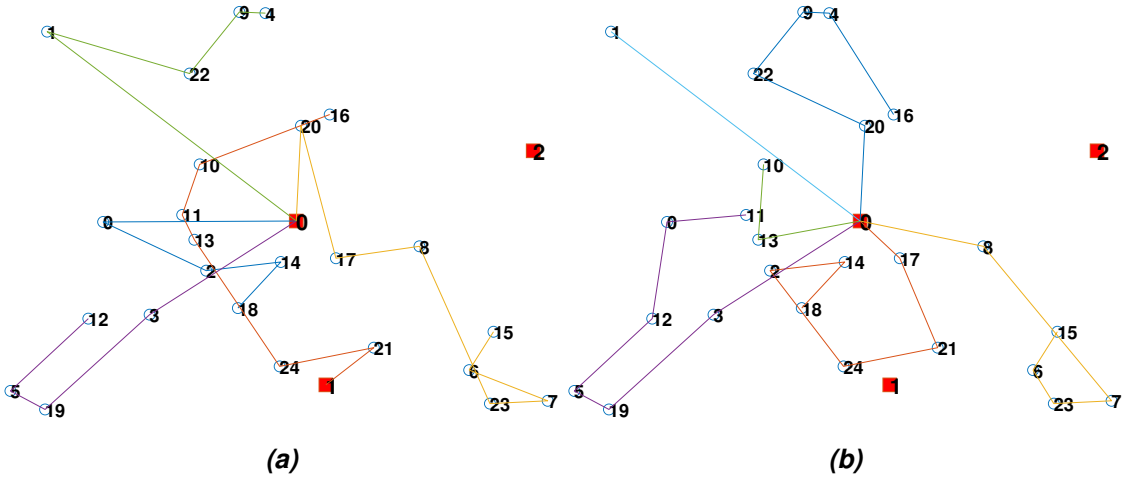
Kuvassa 5.1 on esitetty visuaalisesti liitteen A tiedot eli asiakkaiden ja lähtöpisteiden sijainnit. Satunnaisesti luotuja asiakkaita on esimerkkitapauksessa 25 kappaletta. Asiakkaiden aikaikkunat ovat kaikki 100 minuuttia pitkiä ja niiden alkuajat ovat satunnaisesti luotuja. Algoritmissa on mahdollisuus määritellä jokaiselle asiakkaalle oma palveluaika, mutta esimerkin yksinkertaistamiseksi palveluaika on jokaisella asiakkaalla 20 minuuttia.

ACO perustuu satunnaisuuteen, joten jokaisella suorituskerralla ohjelma tuottaa erilaisen reitin. Taulukossa 5.1 on esitetty yhdeksän peräkkäisen suorituskerran reittien yhteispituus sekä joulupukkien lukumäärä. Kuvassa 5.2 on kaksi erilaista ratkaisua esimerkkitilanteen joulupukkireititysongelmalle.

Kuvasta 5.2 huomataan, että kaikkia lähtöpisteitä ei välttämättä tarvita. Kuvan 5.2a reittien



Kuva 5.1. *Esimerkkitapauksen asiakkaiden sekä joulupukkien lähtöpisteiden sijainnit. Punaiset neliöt ovat lähtöpisteitä ja ympyrät asiakkaita.*



Kuva 5.2. *Esimerkkitapauksen joulupukkireittien eräät ratkaisut.*

yhteispituus on 10267 ja lukumäärä 5. Kuvan 5.2b reittien yhteispituus on 9338, mutta niitä on 6 kappaletta. Ohjelman suoritusaika esimerkkidatalla eli 25:llä asiakkaalla oli niin lyhyt, ettei sitä tarkemmin mitattu. Asiakkaiden määrän ollessa 100 suoritusaika oli noin 10 sekuntia. 200:lla asiakkaalla suoritusaika oli noin 50 sekuntia.

5.2 Parannusehdotukset

Ohjelmaa tulee vielä parantaa, jotta sitä voidaan käyttää todelliseen tarkoitukseensa. Nykyinen ohjelma keskittyy itse reitin luomiseen. Siltä onnistuu reittien luominen, mutta reittien määrittämiseen tarvitaan lisää kontrollia.

Suorituskerta	Reitin yhteispituus	Joulupukkien määrä
1	10332	6
2	10070	6
3	10573	6
4	10107	6
5	9627	6
6	10875	6
7	9882	6
8	9757	5
9	10322	6
10	10509	6
Keskiarvo	**10205**	**5,9**

Taulukko 5.1. *Joulupukkien reittien pituudet esimerkkitilanteessa.*

Joulupukkien määrä pitäisi pystyä päättämään etukäteen. Muurahaispesäoptimointi ei suoraan tarjoa helppoa tapaa päättää joulupukkien lukumäärää, sillä algoritmi lisää reittejä tarvittaessa rajattomasti, kunnes kaikkien asiakkaiden luona on vierailtu. Tämä voitaisiin mahdollisesti ratkaista palkitsemalla algoritmin niitä ratkaisuja, joissa reittien määrä on haluttu.

Kriittinen vaatimus ohjelman hyödyntämiselle tositilanteessa on oikeiden osoitteiden käyttäminen. Matka-aika osoitteesta toiseen tulisi pystyä tehokkaasti selvittämään, jotta algoritmia pystyisi hyödyntämään todellisiin osoitteisiin. Algoritmi voidaan muokata käyttämään uutta etäisyysfunktiota jostakin ulkopuolisesta ohjelmointirajapinnasta (API). Haasteena on sopivan ohjelmointirajapinnan löytäminen ja sen hyödyntäminen tehokkaasti.

Lähtöpisteet ovat toteutetussa ohjelmassa vapaavalintaisia. Kuten luvussa 4 todettiin, jokaisen joulupukin tulisi kuitenkin lähteä omasta kodistaan. Ohjelmaan tulisi määrittää jokaisesta lähtöpisteestä lähteväksi yksi reitti.

Algoritmin tehokkuuteen ei tässä työssä otettu kantaa, vaan painotettiin niiden toimivuutta ja ymmärrettävyyttä. Ohjelmassa on varmasti monta kohtaa, jotka voitaisiin suorittaa tehokkaammin.

6. YHTEENVETO

Työssä käytiin läpi useita erilaisia ajoneuvoreititysongelman muunnoksia sekä tutkittuja ratkaisualgoritmeja. Työssä tarkasteltiin kolmea erilaista ratkaisualgoritmia: muurahaispesäoptimointia, vaihtuvan naapuruston hakualgoritmia sekä geneettistä algoritmia. Niiden avulla pohdittiin joulupukkireititysongelmaa ja luotiin sen yksinkertaistetulle versiolle ratkaisualgoritmi.

Luodulle ohjelmalle löydettiin parannusehdotuksia, jotta sitä voisi hyödyntää oikeasti joulupukkien reittien luomiseen. Helpoiten lähestyttävä MDOVRPTW:n ratkaisuun sovellettavissa oleva algoritmi oli muurahaispesäoptimointiin perustuva algoritmi, mutta siinä on haasteensa. Esimerkiksi joulupukkien määrän rajoittaminen ja reittien lähtöpisteiden määrittäminen ovat haasteellisia. Muilla ajoneuvoreititysongelmien tutkituilla ratkaisualgoritmeilla voi myös ratkaista joulupukkireititysongelman, mutta tässä työssä valittiin käyttöön vain muurahaispesäoptimointi.

Ajoneuvoreititysongelman muunnoksia, ratkaisualgoritmeja ja näkökulmia keksitään jatkuvasti lisää ja jo muutaman vuoden vanha tutkimus saattaa olla vanhentunutta tietoa vaikka ajoneuvoreititysongelma itse on jo yli 60 vuotta vanha.

LÄHTEET

[1] Anita Agardi, Laszlo Kovacs ja Tamas Banyai. ”Mathematical Model for the Generalized VRP Model”. eng. *Sustainability (Basel, Switzerland)* 14.18 (2022), s. 11639–. ISSN: 2071-1050.

[2] Florian Arnold ja Kenneth Sörensen. ”What makes a VRP solution good? The generation of problem-specific knowledge for heuristics”. *Computers & Operations Research* 106 (2019), s. 280–288.

[3] John E. Bell ja Patrick R. McMullen. ”Ant colony optimization techniques for the vehicle routing problem”. eng. *Advanced engineering informatics* 18.1 (2004), s. 41–48. ISSN: 1474-0346.

[4] Sinaide Nunes Bezerra, Sérgio Ricardo de Souza ja Marcone Jamilson Freitas Souza. ”A general VNS for the multi-depot open vehicle routing problem with time windows”. *Optimization Letters* (2023), s. 1–31. ISSN: 1862-4472.

[5] José Brandão. ”A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem”. *European Journal of Operational Research* 284.2 (2020), s. 559–571.

[6] George B Dantzig ja John H Ramser. ”The truck dispatching problem”. *Management science* 6.1 (1959), s. 80–91.

[7] Eppu Hassinen. *joulupukki-ovrptw*. https://github.com/eppuhassinen/joulupukki-ovrptw. 2023.

[8] Marwan A. Hefnawy, Saad M. Darwish ja Amr A. Elmasry. ”Tuning the Evaporation Parameter in ACO MANET Routing Using a Satisfaction-Form Game-Theoretic Approach”. eng. *IEEE access* 10 (2022), s. 98004–98012. ISSN: 2169-3536.

[9] The MathWorks Inc. *MATLAB version: (R2022a)*. Natick, Massachusetts, United States, 2023. URL: https://www.mathworks.com.

[10] Ya-Hui Jia, Yi Mei ja Mengjie Zhang. ”A Bilevel Ant Colony Optimization Algorithm for Capacitated Electric Vehicle Routing Problem”. eng. *IEEE transactions on cybernetics* PP.10 (2022), s. 1–14. ISSN: 2168-2267.

[11] Sašo Karakatič ja Vili Podgorelec. ”A survey of genetic algorithms for solving multi depot vehicle routing problem”. *Applied Soft Computing* 27 (2015), s. 519–532.

[12] Sebastian Lague. *Coding Adventure: Ant and Slime Simulations*. Maaliskuu 2021. URL: https://www.youtube.com/watch?v=X-iSQQgOd1A.

[13] Yanfang Ma, Jie Han, Kai Kang ja Fang Yan. ”An Improved ACO for the Multi-depot Vehicle Routing Problem with Time Windows”. Teoksessa: *Proceedings of the Tenth International Conference on Management Science and Engineering Management*.

Toim. Jiuping Xu, Asaf Hajiyev, Stefan Nickel ja Mitsuo Gen. Singapore: Springer Singapore, 2017, s. 1181–1189. ISBN: 978-981-10-1837-4.

[14] Nenad Mladenović ja Pierre Hansen. ”Variable neighborhood search”. eng. *Computers & operations research* 24.11 (1997), s. 1097–1100. ISSN: 0305-0548.

[15] Erdener Ozcetin ja Gurkan Ozturk. ”A variable neighborhood search for Open Vehicle Routing Problem”. *Concurrency and Computation: Practice and Experience* 35.7 (2023), e7598.

[16] Teemu Perasto. ”Ajoneuvoreititysongelman analysointi ja ratkaiseminen geneettisellä algoritmilla”. Diplomityö. Tampereen yliopisto, 2021.

[17] Su Ping Yu ja Ya Ping Li. ”An Improved Ant Colony Optimization for VRP with Time Windows”. eng. *Applied Mechanics and Materials* 263-266.1 (2013), s. 1609–1613. ISSN: 1660-9336.

LIITE A: TYÖSSÄ KÄYTETYT ASIAKKAAT JA VARASTOT

Esimerkissä käytetty data on esitetty taulukoissa A.2 ja A.1. Asiakkaiden koordinaatit on luotu satunnaisesti välille [-1000,1000]. Aikaikkunoiden alkuajat on luotu satunnaisesti välille 120:stä 300:aan minuuttiin. Aikaikkunoiden loppuaika on alkuaika + 100 minuuttia.

	Koordinaatti x	Koordinaatti y	Aikaikkunan alku	Aikaikkunan loppu
0	-652	-4	245	345
1	-841	809	182	282
2	-304	-212	214	314
3	-495	-400	146	246
4	-104	888	252	352
5	-965	-726	257	357
6	584	-637	274	374
7	849	-768	212	312
8	416	-108	200	300
9	-192	893	216	316
10	-326	242	243	343
11	-385	26	272	372
12	-702	-417	280	380
13	-344	-80	171	271
14	-52	-175	218	318
15	666	-473	298	398
16	113	455	299	399
17	134	-159	154	254
18	-197	-372	263	363
19	-848	-805	167	267
20	16	407	134	234
21	262	-541	176	276
22	-360	630	191	291
23	654	-780	220	320
24	-56	-621	197	297

Taulukko A.1. *Esimerkkitapauksen asiakkaat ja aikaikkunat.*

	Koordinaatti x	Koordinaatti y
0	0	0
1	100	-700
2	800	300

Taulukko A.2. *Esimerkkitapauksen varastot eli joulupukkien lähtöpaikat.*

Parametri	Arvo	Selitys
int r_0	85	0-100% Määrittää satunnaisten solmujen todennäköisyyden.
Pheromone_init	5	Feromonikartan alustusarvo.
Number_of_ants	15	Muurahaisten määrä.
Pheromone_param	1	Feromonijäljen painotus heurisen arvon laskemisessa.
Heuristic_param	10	Heuristiikan painotus heurisen arvon laskemisessa.
Evaporation_factor	0.07	0-1, Feromonijäljen heikkeneminen ja vahvistuminen iteraation aikana.
G_evaporation_factor	0.7	0-1, Feromonijäljen heikkeneminen ja vahvistuminen iteraatioiden välillä.
Iteration	700	Iteraatioiden määrä.
W1	0.3	0-1, Solmujen etäisyyksien ja aikaikkunoiden suhde heuristista arvoa laskettaessa.

Taulukko A.3. *Säädetyt parametrit.*