Tampereen yliopisto

Joel Rauma

MIEHITTÄMÄTTÖMIEN ILMA-ALUSTEN UUDET REITITYSALGORITMIT

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintutkielma
Toukokuu 2021

TIIVISTELMÄ

Joel Rauma: Miehittämättömien ilma-alusten uudet reititysalgoritmit
Kandidaatintutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2021

Miehittämättömien ilma-alusten kehityksessä on otettu viimeisen vuosikymmenen aikana suuria askeleita. Alun perin lähinnä sotilaskäytössä olleet miehittämättömät ilma-alukset ovat saavuttaneet myös siviilimarkkinat ja teknologia on nykyään lähes kenen tahansa saatavilla. Halvat ja tehokkaat pienoiskopterit ovat vallanneet markkina-alaa ja niillä on jo runsaasti käyttöä niin valokuvauksessa kuin tavarantoimituksissa. On odotettavissa, että alusmäärän kasvaessa syntyy lisää taloudellista painetta näiden alusten toiminnan automatisointiin. Tämän tutkielman tavoite on selvittää, minkälaista kehitystä lentoreittien algoritmisessa suunnittelussa on tapahtunut viimeisinä vuosina ja tarkastella uusia ratkaisuja, joita on tarjottu erilaisiin itsenäisestä lennosta nouseviin ongelmiin.

Tutkielmassa esitellään graafiteorian perinteisiä algoritmeja ja niiden käyttöä ilma-alusten reittien kontekstissa. Tutkielmassa perehdytään myös pintapuolisesti parviälyn optimointimenetelmiin. Tämän jälkeen tutkielmassa siirrytään käsittelemään viimeisen viiden vuoden aikana julkaistuja tutkimuksia, joissa on esitelty uusia algoritmeja ja ratkaisuja lentoreitin suunnitteluun. Ratkaisut hyödyntävät sekä graafialgoritmeja että parviälyn optimointimenetelmiä. Osa näistä ratkaisuista on reaaliajassa, aluksen omalla laskentateholla ajettavia algoritmeja, kuten hätälaskeutumiskäyttöön suunniteltu algoritmi. Osassa ratkaisuista on vaatimuksena ennen lentoa ulkoisella laskentateholla muodostettava näkyvyysgraafi tai sitä vastaava vapaan lentotilan määrittävä kartta, jonka pohjalta lentoreitin laskeminen voidaan suorittaa.

Tutkielman perusteella selvitettiin, että viime vuosina kehitetyt lentoreitin suunnittelualgoritmit tarjoavat aiempia nopeampia ja tehokkaampia ratkaisuja lentoreittien suunnitteluun. Tutkimustyön tuloksena reittejä on onnistuttu lyhentämään, erilaisia lentotehtäviä on saatu toteutettua tehokkaammin ja lentotehtävien energiankulutusta on onnistuttu vähentämään. On myös odotettavissa, että uusia algoritmeja tullaan näkemään tulevina vuosina ja että kehitys alalla jatkuu.

Avainsanat: miehittämätön ilma-alus, lentoreitti, reititysalgoritmi, unmanned aerial vehicle (UAV), näkyvyysgraafi

SISÄLLYSLUETTELO

1 Johdanto

Miehittämättömät ilma-alukset (*Unmanned aerial vehicle*, *UAV*) ovat nähneet valtaisaa kehitystä useilla eri aloilla, kuten tavaratoimituksissa, kartoituksissa, muutoin hankalasti tai vaarallisesti saavutettavissa kohteissa (Majeed ja Lee, 2019) mitä erilaisimmissa käyttötilanteissa, niin siviili- kuin sotilaskäytössä. Miehittämättömien ilma-alusten käyttö matalilla lentoreiteillä on myös halvempaa kuin tavallisen ilmaliikenteen hyödyntäminen (Ahmad et al., 2017). Poistamalla vaatimus ihmisohjauksesta saavutettaisiin lentoreitin suunnittelun automatisoinnilla vielä lisää tehokkuutta ja mahdollistettaisiin näiden alusten nykyistä laajempi käyttö. Miehittämättömien ilma-alusten automaattiseen, ihmisohjauksettomaan käyttöön, liittyy kuitenkin useita ratkaisemattomia haasteita (Majeed ja Lee, 2019). Näitä ovat muun muassa tuulen tuoma satunnaisuus, muut ilma-alukset, akunkesto, turvallisuus ihmisille ja omaisuudelle, sekä tämän tutkielman varsinainen aihe, eli lentoreitin suunnittelu näiden haasteiden kontekstissa. Muun muassa nämä haasteet toimivat motivaationa uusien järjestelmien ja ratkaisujen kehittämiseen. Motivaationa voi myös toimia konkreettinen sovellus, kuten vanhusten kaatumista ehkäisevän ja kaatumistilanteissa avustavan järjestelmän kehittäminen (Fakhrulddin ja Gharghan, 2020). Konkreettisten sovellusten toteuttamista varten teoreettisen pohjatiedon on oltava käyttövalmista ja siksi tehokkaiden ja turvallisten reititysalgoritmien kehittäminen on tärkeää. Muita haasteita ovat myös ilmailualan sertifikaatioiden vastaaminen (Paces, 2019). Pacesin (2019) mukaan on kuitenkin vielä sertifikaatioiden laatijoiden käsissä, ovatko itseohjautuvat ilma-alukset ja pienet miehittämättömät ilma-alukset ylipäänsä samojen sertifikaatioiden alaisia kuin varsinaiset lentokoneet ja ilma-alukset, vai tarvitaanko alalle omia sertifikaatteja.

Kirjallisuudessa näihin järjestelmiin viitataan myös termeillä *small unmanned aerial vehicle* (*SUAV*), jolla korostetaan, että kyseessä on pieni ilma-alus (Ahmad et al., 2017), sekä *unmanned aerial system* (*UAS*), jolla viitataan järjestelmän moninaisuuteen, sisältäen siis esimerkiksi maa-asemalla olevan keskusohjausjärjestelmän (Ten Harmsel et al., 2016).

Oleellisena osana näitä ratkaisuja on lentoreitin suunnittelualgoritmi. Alalla on tapahtunut viime vuosien aikana paljon kehitystä ja konkreettisia reaalimaailman ratkaisuja on tarjottu tämän osan ratkomiseen. Tarjotut ratkaisut rakentuvat pitkälti graafiteorian tunnettujen algoritmien päälle. Klassisia graafialgoritmeja ovat muun muassa leveys ensin -haku *(Breadth-first search, BFS)*, syvyys ensin -haku *(Depth-first search, DFS)*, leveys ensin -

hakua tehostava Dijkstran algoritmi, joka tunnetaan myös nimellä *Uniform Cost Search (UCS)* sekä Dijkstran algoritmin tehostettu versio A* (lausutaan ”*A-tähti”*). Lisäksi tunnetaan myös A*:n muunnokset Theeta* ja iteratiivisesti syvenevä A* *(Iterative Deepening A*, IDA*)*, sekä *Rapidly Exploring Random Tree (RRT*)*. Useimmat ratkaisut hyödyntävät A*:ä ja sen muunnoksia. (Paces, 2019)

Osa ratkaisuista perustuu parviälyn sovellutuksiin. Tunnettu parviälyn sovellus on muurahaiskolonnaoptimointi (*Ant Colony Optimization, ACO)* (Colorni et al. 1992), jossa ongelmanratkaisua optimoidaan imitoimalla muurahaisyhteisön toimintaa moniagenttijärjestelmänä, jossa yksinkertaiset ohjelmayksiköt kommunikoivat keskenään ja muodostavat tehokkaan reitin löytävän kokonaisuuden. Lisäksi tunnetaan myös hiukkasparvioptimointi *(Particle Swarm Optimization, PSO)* (Kennedy ja Eberhart, 1995), joka matkii luonnollisten prosessien, kuten lintu- tai kalaparven liikehdintää.

Ainakin A*:ä, IDA*:ä, Theeta*:ä, muurahaiskolonnaoptimointia ja näiden muunnelmia on käytetty, tai yritetty käyttää, itseohjautuvien ilma-alusten lentoreittien suunnittelussa. Kolmiulotteisessa tila-avaruudessa kiinteillä esteillä kahden pisteen välisen lyhimmän reitin löytäminen on kuitenkin NP-kova ongelma, joten ratkaisut eivät voi koskaan olla täydellisen optimaalisia, ainakaan klassisilla tietokoneilla. Ongelmaa voidaan myös verrata kauppamatkaajan ongelmaan (*traveling salesman problem, TSP)*; tietojenkäsittelyssä tunnettuun esimerkkiin, jossa tulee löytää graafista kaikki pisteet läpikäyvä lyhin reitti. Tietynlaisissa lentotehtävissä, kuten tietyn alueen kokonaan peittävällä lentotehtävällä *(coverage path planning, CPP)* (Majeed ja Lee, 2019), voidaan koko ongelma luonnehtia kauppamatkaajan ongelmana.

Oleellisena käsitteenä lentoreittien suunnittelualgoritmeille on näkyvyysgraafi (*visibility graph*). Näkyvyysgraafi kuvaa tilannetta, jossa kaari osoittaa esteetöntä kulkuyhteyttä kahden solmun välillä (Räisänen, 2008). Näkyvyysgraafi voitaisiin siis periaatteessa rakentaa koko reitille, jonka jälkeen tältä graafilta valittaisiin lyhin reitti. Näkyvyysgraafin laskeminen ei kuitenkaan ole mukaan triviaali tehtävä, vaan aikakompleksisuudeltaan $O(n^3)$ (Räisänen, 2008). Täten koko reitin täydellisen näkyvyysgraafin laskeminen ei ole järkevää. Uusissa ratkaisuissa tämä ongelma on huomioitu ja sen sijaan on usein laskettu vain osittainen näkyvyysgraafi, tai jokin sen muunnelma.

Tässä tutkielmassa pyritään tekemään selkoa viimeaikaisista kehityksistä miehittämättömien ilma-alusten lentoreittien suunnittelualgoritmeissa kirjallisuuskatsauksen keinoin. Hakuja tehtiin Andorin, ACM Digital Libraryn ja Google Scholarin kautta, muun muassa hakusanoilla ”unmanned aerial vehicle” AND autonomous sekä ”flight path planning algorithm” AND UAV, rajaten tulokset viime vuosiin. Tässä tutkielmassa käsitellyissä lähteissä motivaatioina ovat olleet hätälaskeutuminen (Ten Harmsel et al., 2016), energiansäästö (Ahmad et al., 2017), sekä erityiset lentoskenaariot, kuten tietyn alueen kokonaan peittävä lentotehtävä, joka on käyttökelpoinen esimerkiksi alueita kartoittaessa tai kuvatessa (Majeed ja Lee, 2019). Lisäksi Majeed ja Lee (2018) ovat kehittäneet ratkaisua reitinetsintäalgoritmien nopeuden parantamiseen.

2 Graafialgoritmeja

Tässä luvussa käsitellään graafiteorian yleisimpiä reitinetsimisalgoritmeja, algoritmeja, jotka etsivät pienimmän aligraafin graafista, joka kuvaa reittiä aloitussolmusta maalisolmuun. Graafiteoria on matematiikan osa-alue, joka tutkii verkkorakenteita eli graafeja. Näiden graafien läpikäymiseen on kehitetty monenlaisia algoritmeja, joista osat soveltuvat tiettyihin tehtäviin toisia paremmin. Graafiteorian historia ulottuu aina 1700-luvulle Leonhard Eulerin työhön, ja algoritmeja graafien läpikäyntiin alettiin kehittämään 1900-luvun aikana (Gross et al., 2013). Osaa näistä algoritmeista on käytetty myös tämän tutkielman käsittelemien algoritmien pohjina. Hart ja muut (1968) jakavat nämä algoritmit kahteen lähestymistyyliin: matemaattiseen ja heuristiseen. Matemaattisessa lähestymisessä käsitellään yleensä abstrakteja graafeja, eikä niissä niinkään olla kiinnostuneita laskentatehokkuudesta, vaan lopullisesta ratkaisun saavuttamisesta. Heuristisessa lähestymistyylissä ongelman ratkaisuun sovelletaan erityistietämystä ongelman aihealueesta ongelman ratkaisun nopeuttamiseksi. Tässä apuna käytetään usein heuristista funktiota, joka on rakennettu tiettyä ongelmaa varten.

Hartin ja muiden (1968) tekstissä esitellään algoritmi, jossa yhdistetään näiden lähestymistyylien tehokkuudet toisiinsa. A* on Hartin ja muiden (1968) Shakey-projektia, ensimmäistä itsenäisesti liikkuvaa robottia, varten kehittämä tunnettu graafialgoritmi ja monissa tapauksissa myös optimaalinen ratkaisu reitin löytämiseksi. Lisäksi Hart ja muut (1968) ovat määritelleet kaksi avainkäsitettä heuristisen funktion arviointiin: hyväksyttävyys (*admissibility*) ja johdonmukaisuus (*consistency*). Mikäli algoritmi löytää reitin

aloitussolmusta haluttuun maalisolmuun mille tahansa graafille, jonka jokaisen kaaren paino on nolla tai enemmän, on algoritmi hyväksyttävä. Algoritmi on myös johdonmukainen, mikäli sen arvio tietystä solmusta maalisolmuun on. Paces (2019) on tehnyt tunnettujen, klassisten algoritmien vertailua tutkimuksessaan.

2.1. A*

A* on Hartin ja muiden (1968) jo 1960-luvulla kehittämä tunnettu hakualgoritmi. A*:n on osoitettu olevan optimaalinen ratkaisu lyhimmän reitin löytämiseen graafista, ja sillä on käyttöä lähes kaikissa tätä tehtävää vaativissa sovelluksissa, joko alkuperäisessä muodossaan, tai paranneltuina versioina, joita käsitellään seuraavassa luvussa. Pacesin (2019) mukaan A* parantaa Dijkstran algoritmin toimintaa arvioimalla nykyisestä solmusta tavoitesolmun saavuttamisen hintaa. Hinta tarkoittaa tässä kontekstissa etäisyyttä kohteeseen, tai kohteen saavuttamiseen tarvittavaa energiamäärää. Tämän hinnan arviointifunktiota kutsutaan heuristiseksi funktioksi $h(n)$. Heuristinen funktio rakennetaan aina tiettyä ratkaistavaa ongelmaa varten, kuten tässä tapauksessa lentoreitin löytämistä varten kolmiulotteisessa tila-avaruudessa. Se voi olla esimerkiksi yksinkertainen euklidinen etäisyys maalisolmuun. A* lajittelee solmut käymällä nykyisen solmun kaikki naapurisolmut läpi alla olevan funktion mukaisesti

$$f(n) = g(n) + h(n)$$

jossa n on seuraava solmu reitillä, $g(n)$ on hinta lähtösolmusta solmuun n, ja $h(n)$ on heuristisen funktion arvio halvimmasta reitistä solmusta n maalisolmuun. Tämän jälkeen algoritmi siirtyy käsittelemään halvinta löytynyttä naapurisolmua ja sen naapurisolmuja, eli algoritmi alkaa laajentamaan kyseistä solmua. Hartin ja muiden (1968) mukaan tiettyjen oletusten vallitessa A* löytää aina optimaalisen ratkaisun reitinetsimisongelmaan, laajentaen mahdollisimman vähän solmuja. A*:n implementointi käyttää useimmiten hyväkseen jonoa, johon täytetään aina käsiteltävän solmun naapurit. Kun solmun naapurit saadaan läpikäytyä, poistetaan käsiteltävä solmu jonosta ja aletaan rakentaa jonoa taas halvimman löytyneen solmun pohjalta, ja niin edelleen. Kun jonosta poistettu solmu on itse maalisolmu, muodostaa poistettujen solmujen sarja halvimman reitin lähtöpisteestä maalipisteeseen. Oheinen pseudokoodi kuvaa A*:n tyypillistä toimintaa koodina.

```
Input: Graafi G(V,E), lähtösolmulla alku ja maalisolmulla loppu
Output: Lyhin reitti pisteiden alku ja loppu välillä

Init
    avoin := { alku }  //jono läpikäytävistä solmuista
    suljettu := { }    //jono läpikäydyistä solmuista
    g(alku) := 0       //hinta alku:sta solmuun n
    h(alku) := heuristinen_funktio(alku, loppu)
          //arvio hinnasta solmusta n loppu:un
    f(alku) := g(alku) + h(alku)  //hinta alku:sta loppu:un

while avoin != Ø
    m := avoin-jonon päällimmäinen solmu, pienimällä f:llä
    if m == loppu
        return
    poista m jonosta avoin
    lisää m jonoon suljettu
    for each n in lapsi(m)
        if n in suljettu
            break
        hinta = g(m) + etaisyys(m,n)
        if n in avoin and hinta < g(n)
            poista n jonosta avoin, uusi reitti on halvempi
        if n in suljettu and hinta < g(n)
            poista n jonosta suljettu
        if n not in avoin and n not in suljettu
            lisää n jonoon avoin
            g(n) := hinta
            h(n) := heuristinen_funktio(n,loppu)
            f(n) := g(n) + h(n)
return epaonnistui
```

A:n pseudokoodi*

A* on sovellettavissa itseohjautuviin ilma-aluksiin muodostamalla tila-avaruudesta esimerkiksi kartta- ja muototietojen, tai sensorien havaintodatan perusteella graafi, jota seuraamalla A* löytää optimaalisen ratkaisun. Tila-avaruus voi olla diskretisoitu esimerkiksi kuutiomaisiksi tai muutoin monikulmiomaisiksi kappaleiksi, jotta graafin muodostaminen onnistuu.

2.2. Muita graafialgoritmeja

A* on yleisesti optimaalinen ratkaisu lyhimmän reitin löytämiseen graafista, mutta tiettyjen rajoitteiden puitteissa muut, A*:n kaltaiset ratkaisut, voivat tarjota etuja. Ehdotetut algoritmit tarjoavat ratkaisuja muun muassa muistivaatimuksen kasvaessa kohtuuttomaksi, jatkuvan

maailman neliömäiseksi diskretisoinnin aiheuttamiin tehokkuusongelmiin, tai tarjoavat täysin erilaista lähtökohtaa reitin löytämiseen (Paces, 2019)

Eräs A*:n muunnelma on iteratiivisesti syvenevä A*. Tämä algoritmi yhdistää A*:n ja syvyys ensin -haun käyttäytymistä. Oikealla heuristiikalla algoritmi löytää optimaalisen ratkaisun ja käyttäytyy kuin A*. Algoritmi vaatii vähemmän muistia ja enemmän aikaa kuin A*. Mikäli muistivaatimus muodostuu rajoittavaksi tekijäksi, tarjoaa iteratiivisesti syvenevä A* vaihtoehdon (Paces, 2019).

Theeta* on A*:n parannus oikean, jatkuvan maailman tarpeisiin. Siinä kun A* vaatii tilan diskretisointia neliölliseksi avaruudeksi, lisää Theeta* jälkiprosessointia tarkistamalla, mikäli solmu on suorassa näköyhteydessä aiemmalta solmulta, ja tekee reittiin reittiä lyhentäviä parannuksia tämän jälkiprosessoinnin avulla. Theeta* löytää lähes optimaalisia ratkaisuja A*:een verrattavassa ajassa (Paces, 2019).

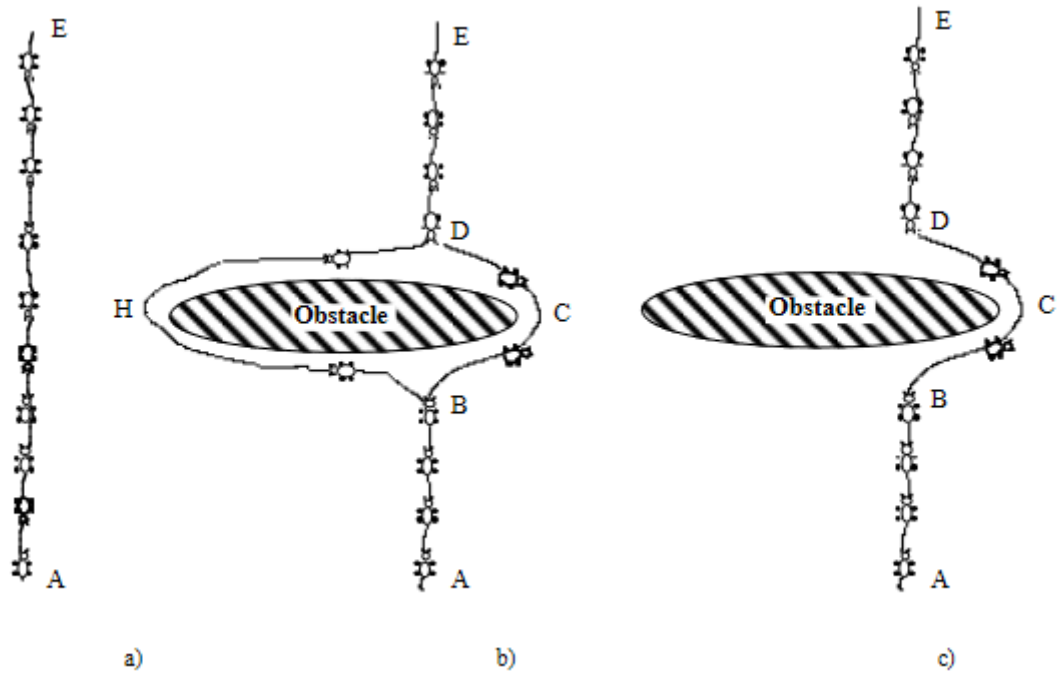
Rapidly exploring random tree (RRT*) on LaVallen (1998) kehittämä algoritmi. Tässä algoritmissa luodaan satunnaisia, puun kaltaisia rakenteita saatavilla olevaan tila-avaruuteen. Puiden haarojen muodostamista voidaan ohjata tiettyyn suuntaan, tai esimerkiksi suurimpaan vapaaseen tilaan. RRT* voi näin löytää optimaalisen reitin, tai saattaa myös tarjota vaihtoehtoisia reittejä. RRT* voisi siis olla käyttökelpoinen reittivaihtoehtoja etsittäessä, esimerkiksi yllättävän esteen ilmaantuessa.

3 Parviälyn optimointimenetelmiä

Parviälyssä yksinkertaisten, keskittämättömästi ja itseohjautuvien agenttien kokonaisuus luo monimutkaista yhteistoimintaa. Yksittäisellä agentilla on muutama yksinkertainen sääntö, ja näitä sääntöjä seuraamalla agenttien kokonaisuus voi muodostaa hyvinkin erilaistunutta toimintaa, kuten luonnossa muurahais- ja mehiläispesiä, lintuparvia, kuin myös mikrobien yhteistoimintaa. Parviäly on myös sovellettavissa keinotekoisina sovelluksina, kuten ensimmäisenä monista Reynoldsin (1987) Boids, joka jäljittelee lintuparven parveilemistapaa. Parviälyn keinotekoiset ratkaisut tarjoavat mahdollisuuksia hajautettuun, äärimmilleen rinnakkaistettuun ongelmanratkaisuun ja optimointiin (Colorni et al., 1992). Seuraavassa luvussa käsitellään parviälyn optimointimenetelmistä muurahaiskolonnaoptimointia.

3.1. Muurahaiskolonnaoptimointi

Muurahaiskolonnaoptimoinnin kehitti Colorni ja muut vuonna 1992 julkaistussa tutkimuksessa. Colorni ja muut (1992) tarjoavat paperissaan muurahaiskolonnaoptimointia ratkaisuksi kauppamatkaajan ongelmaan. Muurahaiskolonnaoptimointi mallintaa muurahaisyhteiskunnan toimintaa, mallintamalla muurahaisten tapaa jättää feromoneja käyttämälleen reitille, jolloin useammat muurahaiset päätyvät käyttämään samaa reittiä. Muurahaisten välinen kommunikaatio voi siis olla hyvin yksinkertaista, mutta ajan kuluessa niiden yhteistoiminta johtaa tehokkaisiin ratkaisuihin. Paperissa demonstroidaan, kuinka esteen asettaminen reitille johtaa siihen, että muurahaiset ottavat ennen pitkää lyhyemmän reitin käyttöönsä. (Colorni et al., 1992) Alla oleva kuva havainnollistaa algoritmin toimintaa.



Kuva 1 – Muurahaiskolonnaoptimoinnin toiminta. a) Muurahaiset kävelevät pisteiden A ja E välillä. b) Reitille asetetaan este ja muurahaisten tulee löytää reitti esteen ohitse. c) Koska useammat muurahaiset ehtivät kulkemaan pisteiden D ja B välin käyttämällä pisteen C kautta kulkevaa reittiä, vahvistuu sillä reitillä feromonijälki nopeammin ja muurahaiset valitsevat ennemmin tai myöhemmin lyhimmän reitin. Kuvan lähde Colorni et al. (1992).

Algoritmi mallintaa muurahaisten käyttäytymistä neljän parametrin kautta: herkkyys feromonijäljille, herkkyys etäisyydellä jäljestä, jälkien haihtumisnopeus ja feromonien jättämismäärä. Näitä parametreja säätämällä saadaan tehokkaasti muurahaiskolonnan toimintaa jäljittelevä algoritmi, joka löytää lyhimmän reitin määritellyssä tilassa. Mallintamalla

tilaa, jossa lento suoritetaan, sekä simuloimalla tällaisten agenttien kulkua kyseisessä tilassa, on muurahaiskolonnaoptimointi käyttökelpoinen lentoreittejä suunnitellessa.

4 Uusia algoritmeja lentoreitin suunnitteluun

Tässä luvussa käsitellään viime vuosina kehitettyjä uusia ratkaisuja lentoreittien suunnitteluun. Uusien algoritmien kehittämiseen on ollut useita erilaisia motiiveja; niin energiansäästöä, hätätilannetoimintaa, kuin ylipäänsä tehokkaampien vaihtoehtojen kehittäminen. Uudenlaiset algoritmit rakentuvat useimmiten A*:n tai muurahaiskolonnaoptimoinnin päälle.

4.1. Energiatehokkuus lentoreittejä suunnitellessa

Ahmadin ja muiden (2017) tutkimuksessa haetaan ratkaisua lentoreitin suunnitteluun pienille miehittämättömille ilma-aluksille *(SUAV)* kolmiulotteisessa, kiinteäesteisessä ympäristössä. Ratkaisun tarkoituksena on löytää törmäysvapaa ja energiatehokas reitti tällaiselle ilma-alukselle. Motivaationa tutkielman tekijöillä algoritmin kehittämiseen on energian säästö. Ratkaisuksi ehdotetaan näkyvyysgraafin laajennosta, jota kutsutaan kolmiulotteiseksi näkyvyysetenemissuunnitelmaksi (*visibility roadmap*).

Tutkimus on jaettu kahteen osaan: esiprosessointiin ja hakuvaiheeseen. Esiprosessointivaiheessa luodaan täysi näkyvyysgraafi, joka sisältää näkyvyysgraafin määritelmän mukaisesti jokaiselta solmulta näkyvillä ja suoraan saavutettavissa olevat muut solmut. Ahmad ja muut (2017) kutsuvat tätä näkyvyysetenemissuunnitelmaksi ja sen luomista on nopeutettu sisällyttämällä luomisvaiheeseen korkeusintervalli, jotta pisteiden välillä olisi jokin järkevä korkeusero, eikä pisteitä syntyisi ääretöntä määrää korkeussuunnassa. Tämä data luodaan vain kerran, erillään lentotehtävästä. Näkyvyysgraafin luonti täytyy siis tehdä erikseen, sillä lennokin laskentatehon ei oleteta olevan siihen riittävä. Hakuvaiheessa ajetaan A*-algoritmi aloitus- ja lopetuspisteille tässä näkyvyysetenemissuunnitelmassa. Tarjottu algoritmi on reaaliajassa, lennokin omalla prosessointiteholla ajettava algoritmi. (Ahmad et al., 2017)

Algoritmia verrattiin Nachmanin (2007) uniform grid -algoritmiin, sikäli kun se on ollut tuolla hetkellä päivitetyin ja samankaltaisin työ. Kummallekin menetelmälle on käytetty reitin etsimiseen A*-algoritmia ja tutkimuksen tulokset osoittavat, että Ahmadin ja muiden 2017

ehdotuksessa on saavutettu keskimäärin 12 %:n energiansäästö viidelläkymmenellä koeajolla verrattuna Nachmanin (2007) ruudukkomenetelmään. Ahmadin ja muiden (2017) näkyvyysetenemissuunnitelmaratkaisu on myös laskenta-ajaltaan tehokkaampi kuin Nachmanin (2007) menetelmä. Nämä testit ajettiin simuloidussa, satunnaistetussa ympäristössä. Testejä ajettiin myös oikean maailman datalla. Näissä testeissä nähtiin 10 % lyhennys matkan pituudessa, ja 49,8 % parannus energiankäytössä, verrattuna Nachmanin (2007) ruudukkomenetelmään. (Ahmad et al., 2017)

4.2. Lentoreitin suunnitteleminen hätätilanteissa

Ten Harmselin ja muiden (2016) tutkimuksen motivaationa ovat hätätilanteet ja riskien minimointi ihmisille ja omaisuudelle. Tutkimuksessa etsitään algoritmia hätälaskeutumiseen odottamattomassa akun loppumisen tilanteessa. Ratkaisussa keskitytään julkisten tietokantojen hyödyntämiseen väestöjakaumasta, rakenteista ja maastosta hintakartan *(cost map)*, eli lentoreitin energiavaatimusta kuvaavan graafin, laskemiseen. Algoritmin vaatimuksena on hintakartan laskeminen maa-asemalla valmiiksi, sillä tällaisten eri tietueiden yhdistäminen ylitti ainakin senhetkisten (2016) miehittämättömien ilma-alusten laskentatehon. Algoritmi hyödyntää A*-algoritmia, joka on todistetusti käyttökelpoinen reaaliajassa. Lisäksi euklidiseen suoraan viivaan perustuvan heuristisen funktion käyttö rajoittaa haettavaa hakuavaruutta, joka on oleellista lennokissa, koska hätätilanteessa vaatimuksena on reitin laskeminen ilma-aluksen omalla laskentateholla reaaliajassa. Tutkimuksessa esitellään tuloksia simuloiduista tilanteista New Yorkissa. (Ten Harmsel et al., 2016)

Hätälaskeutumisen suunnitteluohjelma on jaettu tutkimuksessa kahteen osaan, jotka yhdistetään parasta tulosta varten; sensoripohjainen suunnittelija ja karttapohjainen suunnittelija. Karttapohjainen suunnittelija vastaa reitin suunnittelusta laskeutumispisteen välittömään läheisyyteen, kun taas sensoripohjainen suunnittelija vastaa lopullisesta lähestymisestä. Tekstissä keskitytään kuitenkin karttapohjaisen suunnitteluohjelman toimintaan. Algoritmi hyödyntää tietoa väestötiheydestä, rakennusten ja muiden esteiden korkeustiedoista, sekä maastonmuodoista. Ratkaisussa on priorisoituna ensimmäisenä turvallisuus ihmisille, jonka jälkeen omaisuudelle, ja lopulta itse ilma-alukselle. (Ten Harmsel et al., 2016)

Algoritmia testattiin simulaatiossa New Yorkin Manhattanilla 3 km * 3 km laajuisella alueella. Simulaatioissa hätälaskeutumisalgoritmin ajamisessa kului reilusta sekunnista noin kymmenen sekuntiin, mediaanin ollessa 4,1 sekuntia. Ten Harmselin ja muiden (2016) mukaan tämä osoittaa, että algoritmin ajaminen on käytännöllistä aluksen omalla laskentateholla. Lisäksi on otettava huomioon, että tutkimus on julkaistu 2016 ja laskentateho näissä aluksissa on kasvanut. Testeissä huomattiin myös, ettei reitinsuunnitteluajan ja reitin lentoajan välillä ollut korrelaatiota. Tuloksissa myös huomattiin, että suoran etäisyyden heuristiikan käyttö johtaa epätehokkaaseen laskeutumispaikan löytymiseen; algoritmi siis käy turhaan läpi ylimääräisiä kohteita. (Ten Harmsel et al., 2016)

4.3. Nopea algoritmi lentoreitille

Majeedin ja Leen (2018) tutkimuksessa esitetään uusi lentoreitinsuunnittelualgoritmi kolmiulotteiseen kiinteäesteiseen ympäristöön. Tämän tutkimuksen motivaationa on yksinkertaisesti nopean algoritmin löytäminen. Algoritmi löytää optimaalisia tai lähes-optimaalisia ratkaisuja nopealla laskenta-ajalla, kuitenkaan reittejä pidentämättä. Algoritmi muodostaa ympäröimällä puolisylinterin mallisen hakuavaruuden käyttäen tietoa esteiden geometriasta, josta lentoreitti voidaan laskea uhraamatta lähes-optimaalisuutta, saati nopeutta. Harva näkyvyysgraafi (*sparse visibility graph*) luodaan ympäröidystä tilasta, josta ensimmäinen reitti etsitään A*-algoritmilla, ja tätä reittiä optimoidaan vielä myöhemmin ajon aikana. Harvan näkyvyygraafin ajatuksena on jättää mahdolliset ylimääräiset solmut laskematta ja täten nopeuttaa näkyvyysgraafin muodostusta. Algoritmi ratkaisee tehokkaasti tasapainon tehokkuuden ja optimaalisuuden välillä.

Majeedin ja Leen (2018) mukaan erilaisista tilanteista luotujen simulaatioiden mukaan algoritmi on tehokkaampi kuin edeltäjänsä. Algoritmia verrattiin ApVL-algoritmiin ja rajoitetun tilan algoritmiin *(bounded space algorithm)*, jotka ovat tuolla hetkellä olleet samankaltaisimmat ja kehittyneimmät saatavilla olleet menetelmät. Kummatkin näistä käyttävät tiheää näkyvyysgraafia, verrattuna ehdotettuun harvaan näkyvyysgraafiin. Algoritmin käyttö vähensi ympäristön mallintamiseen kulunutta laskenta-aikaa 18,65 %. Reitin laskenta-aika väheni 11,9 % verrattuna ApVL:ään ja 26,9 % rajoitetun tilan algoritmiin. Algoritmia verrattiin myös RRT*-algoritmiin ja sen paranneltuun versioon RRT*-AB-algoritmiin. Näihin verrattuna tarjottu algoritmi nopeutti laskenta-aikaa keskimäärin 38,45 %. Kummassakin simulaatiossa tarjotun algoritmin ehdottamien reittien pituus oli myös

keskimäärin verrokkeja lyhyempi. Näiden simulaatioiden perusteella tutkimuksessa todetaan tarjotun algoritmin suoriutuvan paremmin tehtävistä kuin sen hetkisten tehokkaimpien vastaavien algoritmien.

4.4. Alueen peittävä lentoreitti

Majeedin ja Leen (2019) toisessa tutkimuksessa etsitään kolmiulotteisen, kiinteäesteisen urbaanin ympäristön tietyn alueen kokonaan peittävälle lentotehtävälle, kuten kuvauslennolle, soveltuvaa lentoreitinsuunnittelualgoritmia. Ratkaisu löytää kaikki halutut reittipisteet käsittävän reitin, samalla kun vähentää laskenta-aikaa, käännöksiä ja saman reitin uudelleenylityksiä. Ratkaisu hyödyntää menetelmää, jota Majeed ja Lee (2019) kutsuvat harvaksi reittipistegraafiksi (*sparse waypoint graph)*; tässä kontekstissa harva tarkoittaa ylimääräisiä ja hyödyttömiä solmuja sisältämätöntä. Ratkaisu hyödyntää myös sensorin pyyhkäisyalaa *(footprints sweep)*, joka tarkoittaa tietystä pisteestä sensorin havaitsemaa aluetta. Sensorin pyyhkäisyalan reittipistegraafiin sovittamisen jälkeen algoritmi määrittää pisteiden läpikäyntijärjestyksen muotoilemalla ongelman kauppamatkaajan ongelmana, ja tämän jälkeen muodostettu ongelma ratkaistaan muurahaiskolonnaoptimointialgoritmilla. Sensorin pyyhkäisyalat yhdistetään harvaksi reittipistegraafiksi yhdistämällä sensorin pyyhkäisyalojen päätöspisteet toisiinsa, jotta saataisiin muodostettua muurahaiskolonnaoptimoinnilla täydellinen peittolentoreitti. (Majeed ja Lee, 2019)

Majeed ja Lee (2019) testasivat algoritmiaan simuloidussa ympäristössä, verraten sitä muihin vastaavanlaisiin algoritmeihin, kuten BCDH-CPP:hen ja CA-CPP:hen. Tulosten mukaan uusi algoritmi suoriutui viidessä erilaisessa testiympäristössä laskenta-ajan puolesta 11,2 % ja 19,7 % nopeammin kuin vastaavat algoritmit. Uusi algoritmi myös tuotti 9,1 % lyhyempiä reittejä kuin BCDH-CPP ja 4 % lyhyempiä reittejä verrattuna CA-CPP:hen. CA-CPP tuottaa kuitenkin lyhyempiä reittejä pienillä ja vähäesteisillä alueilla, mutta ympäristön haastavuuden lisääntyessä algoritmi on tehokkaampi niin laskenta-aikaan kuin reittien pituuksiin nähden. Keskimäärin tehokkuuden lisäys on ollut 9,98 % verrattuna aikaisempiin algoritmeihin. Lisäksi otettiin huomioon simuloitavan lennettävän alueen muodon ja vertasivat sen perusteella suoriutumista muihin algoritmeihin. Epäsäännöllisen alueen lento on aina hitaampi kuin säännöllisen alueen, ja keskimäärin

vaihtelevilla muodoilla tarjottu algoritmi suoriutuu 21,34 % nopeammin kuin vertailtavat algoritmit ja tuottaa keskimäärin 8,98 % lyhyempi reittejä. Lopuksi mainitaan algoritmin tuottavan reittejä, joilla on vähemmän itseään ylittäviä kohtia kuin verrokeilla. (Majeed ja Lee, 2019)

5 Yhteenveto ja johtopäätökset

Tässä tutkielmassa esiteltiin neljä uutta lentoreitin suunnittelualgoritmia miehittämättömille ilma-aluksille. Tutkielman alussa selostettiin A*-algoritmin toimintaperiaate, ja selitettiin graafialgoritmien perustavanlaatuista toimintaa. Tämän jälkeen avattiin uusien sovelluskohtaisten algoritmien motivaatioita, toimintaa ja tuloksia. Motivaatioita uusien algoritmien suunniteluun on monenlaisia ja tekniikan kehittyessä on odotettavissa, että tulevina vuosina algoritmiehdotelmia tullaan näkemään lisääntyvissä määrin. Kun tekniikka sallii konkreettisten ratkaisujen syntymisen, on teoreettisen pohjan ohjelmistoa varten oltava olemassa.

Tutkielmassa huomattiin, että uusia ratkaisuja on esitetty viimeisenä viitenä vuonna enemmän kuin sitä edeltävänä viitenä vuonna. Tutkielmassa saatiin myös selville, että algoritmien tehokkuus on kasvanut niin reitin laskenta-ajan, ympäristön laskenta-ajan kuin reitin lyhyyden osalta. Algoritmeja on myös kehitetty enemmän kuin näitä viittä vuotta edeltävänä viiden vuoden jaksona. Voidaan siis hyvinkin odottaa, että algoritmien kehittäminen jatkuu tulevina vuosina vähintään yhtä aktiivisena, ellei aktiivisempana.

Kevyet miehittämättömät ilma-alukset tarjoavat huomattavia taloudellisia hyötyjä, jollaisia ei ennen ole ollut saatavilla. Kevyet miehittämättömät ilma-alukset ovat jo ainakin ihmisohjauksellisessa käytössä päässeet markkinoille, ja niitä käytetään jo esimerkiksi tavarantoimituksiin Yhdysvalloissa. On siis vain ajan ja lainsäädännön kysymys, milloin vaikkapa tässä tutkielmassa esiteltyjä algoritmeja päästään käyttämään oikeassa maailmassa laajalti.

6 Lähdeluettelo

Ahmad, Z., Ullah, F., Tran, C., & Lee, S. (2017). Efficient Energy Flight Path Planning Algorithm Using 3-D Visibility Roadmap for Small Unmanned Aerial Vehicle. *International Journal of Aerospace Engineering*, *2017*, 1–13. https://doi.org/10.1155/2017/2849745

Colorni, A. (1991). *Distributed optimization by ant colonies*. ECAL91 - European Conference on Artificial Life, Paris, France.

Fakhrulddin, S. S., & Gharghan, S. K. (2020). An Elderly First Aid System Based-Fall Detection and Unmanned Aerial Vehicle. *IOP Conference Series: Materials Science and Engineering*, *745*, 012096. https://doi.org/10.1088/1757-899x/745/1/012096

Gross, J. L., Yellen, J., & Zhang, P. (2013). History of Graph Theory. In *Handbook of Graph Theory, Second Edition* (pp. 31–51). Taylor & Francis.

Hart, P., Nilsson, N., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, *4*(2), 100–107. https://doi.org/10.1109/tssc.1968.300136

Paces, P. (2019). Comparison of Flight-Planning Algorithms in View of Certification Requirements. Institute of Electrical and Electronics Engineers, American Institute of Aeronautics and Astronautics, American Institute of Aeronautics and Astronautics Digital Avionics Technical Committee, & IEEE Aerospace and Electronic Systems Society. (2019). *38th DASC*. IEEE. https://doi.org/10.1109/DASC43569.2019.9081624

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*. https://doi.org/10.1109/icnn.1995.488968

LaValle, S. M. (1998, October). *Rapidly-exploring random trees: A new tool for path planning*. http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.pdf

Majeed, A., & Lee, S. (2018). A Fast Global Flight Path Planning Algorithm Based on Space Circumscription and Sparse Visibility Graph for Unmanned Aerial Vehicle. *Electronics*, *7*(12), 375. https://doi.org/10.3390/electronics7120375

Majeed, A., & Lee, S. (2019). A New Coverage Flight Path Planning Algorithm Based on Footprint Sweep Fitting for Unmanned Aerial Vehicle Navigation in Urban Environments. *Applied Sciences*, *9*(7), 1470. https://doi.org/10.3390/app9071470

Nachmani, G. (2007). *Minimum-energy flight paths for UAVs using mesoscale wind forecasts and approximate dynamic programming*. https://apps.dtic.mil/sti/pdfs/ADA475882.pdf

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406

Räisänen, V. (2008). *Peltorobotin reititys ”siksak”-täytöllä*. http://vsr.name/siksak.pdf

Ten Harmsel, A. J., Olson, I. J., & Atkins, E. M. (2016). Emergency Flight Planning for an Energy-Constrained Multicopter. *Journal of Intelligent & Robotic Systems*, *85*(1), 145–165. https://doi.org/10.1007/s10846-016-0370-z