Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

Tuikku Anttila

Trukkitöiden reittioptimointi varastossa

Diplomityö
Espoo, 1. lokakuuta 2015

Valvoja: Professori Antti Ylä-Jääski
Ohjaaja: Timo Mulju

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Tuikku Anttila		
Title:	Optimizing forklift routes for putaway and replenishment		
Date:	Oct 1, 2015	**Pages:**	54
Major:	Data Communication Software	**Code:**	T-110
Supervisor:	Professor Antti Ylä-Jääski		
Advisor:	Timo Mulju M.Sc. (Tech.)		

In a warehouse, putaway and replenishment need to be done efficiently so that order picking is not delayed. Two different route optimization strategies for putaway and replenishment tasks are presented in this thesis. In the first one, routes are formed from all currently open tasks. The problem thus becomes a Vehicle Routing Problem with Pickups and Deliveries. In the second strategy, only one task is considered at a time. The selection is based on proximity.

The optimization is evaluated by simulating the warehouse. We write programs for both optimization strategies that read through a list of past tasks from the warehouse management system and distribute them to forklifts according to the chosen optimization strategy. Compared to the estimated driving distances that the forklifts covered when carrying out the tasks in reality, the optimized driving routes are 7-11 % shorter. Average time for putaway tasks decreased, but replenishment tasks took the same time as before or longer.

New tasks arrive continuously in the warehouse management system. Thus the optimization problem is a dynamic one. The simulation algorithms currently solve the problem as a series of static problems, considering only the tasks available at the moment. This strategy was enough to decrease the driving distance, but we also discuss how the algorithms could be developed further to update the solutions when new tasks arrive. We also describe how to implement this type of warehouse optimization in practice.

Keywords:	forklift, warehouse, optimization, routing, vehicle routing problem, pickup and delivery problem
Language:	Finnish

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Tuikku Anttila		
Työn nimi:	Trukkitöiden reittioptimointi varastossa		
Päiväys:	1. lokakuuta 2015	**Sivumäärä:**	54
Pääaine:	Tietoliikenneohjelmistot	**Koodi:**	T-110
Valvoja:	Professori Antti Ylä-Jääski		
Ohjaaja:	Timo Mulju		

Varastossa trukkitöillä tarkoitetaan hyllytystä, eli lavojen siirtoa vastaanottoalueelta hyllyihin, ja aktiivin täydennystä, eli tyhjien hyllypaikkojen täyttöä. Näiden tehtävien nopea suoritus on edellytyksenä tehokkaalle keräilylle. Tässä työssä tutkitaan kahta erilaista optimointimenetelmää trukkitöille. Ensimmäisessä menetelmässä muodostetaan vapaana olevista tehtävistä kokonaisia reittejä. Ratkaistava ongelma on tällöin ajoneuvon reititysongelma noudoilla ja toimituksilla. Toisessa menetelmässä huomioidaan vain lähin vapaa tehtävä.

Optimoinnin tehokkuutta tutkitaan simuloimalla. Laaditaan ohjelmat, jotka käyvät läpi varastonhallintajärjestelmästä poimittuja todellisia tehtäviä ja jakavat niitä trukeille ohjaussäännön mukaan. Vertaamalla ohjelman laskemia matkoja alkuperäisestä tilanteesta arvioituihin ajomatkoihin havaitaan, että optimoimalla voidaan lyhentää trukkien ajomatkaa 7-11 prosenttia. Hyllytystehtävät tulivat tehdyksi huomattavasti tehokkaammin uudella ohjausmenetelmällä. On kuitenkin huomattava, että todellisuudessa hyllytettävät lavat viedään usein ensin väliaikaiseen varastoon ja vasta sitten lopulliselle paikalleen, minkä takia varastonhallintajärjestelmässä näkyvä suoritusaika on pitkä. Aktiivin täydennystehtävät tehdään nykyiselläänkin nopeasti eikä niiden suoritusaikaan saatu tässä tapauksessa muutosta.

Uusia trukkitöitä tulee jatkuvasti, joten optimointiongelma on dynaaminen. Simulaation algoritmit ratkaisevat tällä hetkellä ongelman staattisena, huomioiden tietyllä ajanhetkellä vapaana olevat tehtävät. Tälläkin ratkaisulla ajomatka lyheni alkutilanteesta, mutta työssä käsitellään lyhyesti, miten algoritmi voisi paremmin huomioida optimointiongelman dynaamisuuden. Lisäksi käsitellään, miten optimoinnin voisi toteuttaa käytännössä.

Asiasanat:	trukkityöt, siirtotilaukset, varasto, optimointi, reititys, ajoneuvon reititysongelma
Kieli:	Suomi

Alkusanat

Kiitokset Timo Muljulle aiheen ideoinnista ja työn ohjauksesta ja Sami Vesteriselle suuresta avusta käytännön työssä. Kiitokset Antti Ylä-Jääskelle valvojana toimimisesta ja tärkeästä palautteesta työn muotoilua ja sisältöä koskien. Kiitokset myös perheelleni kannustuksesta koko opiskelujen ajan. Erityisesti kiitän avopuolisoani Joelia, joka on jaksanut tukea ja torjua ennustuksiani siitä, että työ ei joko tule koskaan valmistumaan tai saa erityismaininnan surkeimmasta diplomityöstä ikinä. Kuten tuloksesta näkee, niin ei käynyt.

Espoo, 1. lokakuuta 2015

Tuikku Anttila

Termit ja lyhenteet

Aktiivi	Varastopaikka, jolta kerääjät poimivat tavaroita toimituksiin
Hyllytys	Tavaran siirto vastaanottoalueelta reservipaikalle
PDP	Pickup and Delivery Problem, reititysongelma, jossa kuormia pitää noutaa alkupisteistä ja viedä loppupisteisiin
Reservi	Pidempiaikainen säilytyspaikka, josta tuotteita siirretään aktiivipaikoille
Siirtotilaus	katso Trukkityö
Trukkityö	Tavaran siirto varastossa paikasta toiseen, joko hyllytys tai täydennys
Täydennys, täyttö	Tavaran siirto reservistä aktiivipaikalle
Vastaanottoalue	Alue, johon lavat saapuvat ja josta ne viedään hyllypaikoille
VRP	Vehicle Routing Problem, ajoneuvon reititysongelma
VRPPD	Vehicle Routing Problem with Pickup and Delivery, ajoneuvon reititysongelman erityistapaus
WMS	Varastonhallintajärjestelmä, Warehouse Management System

Sisältö

Luku 1

Johdanto

Varasto on merkittävä osa mitä tahansa toimitusketjua, ja varastojen ohjausta ja optimointia on tutkittu ainakin 1970-luvulta lähtien laajalti. Yli 50 prosenttia varaston kokonaiskustannuksista syntyy kulkukustannuksista [1], joten reittien optimointi varastossa on yksi merkittävä tutkimuskohde. Varsinkin keräilyn reititystä on tutkittu paljon, sillä keräilyn osuus varastotyön kokonaiskustannuksista on erityisesti manuaalivarastossa suuri. Tämä työ käsittelee trukkitöitä eli siirtotilauksia, jotka sisältävät hyllytys- ja täydennystehtävät. Trukkityöt on tärkeää suorittaa nopeasti, jotta keräily sujuu viiveettä. Trukkitöiden optimointia varsinkaan reittien suhteen ei kuitenkaan ole tutkittu muihin varastoprosesseihin verrattuna kovin laajasti.

Tässä työssä keskitytään trukkitöiden optimointiin automatisoimattomassa varastossa, jossa lavoille on aina määritelty tietty paikka ja työpyyntö sisältää lavan lähtö- ja kohdepaikan. Kaikki tehtävät tulevat suoraan varastonhallintajärjestelmästä, joten optimointi kannattaa suunnata trukkien reitteihin ja tehtävien suoritusjärjestykseen. Tehtävien jonossaoloaika halutaan minimoida, koska avoinna olevat täydennystehtävät aiheuttavat odotusaikaa keräilijöille. Lisäksi tuottamaton tyhjänä ajomatka halutaan minimoida.

Trukkitöiden optimointia tutkitaan simuloimalla. Simulaatiomalli on tässä tapauksessa ohjelma, joka kuvaa varaston toimintaa yksinkertaistettuna. Ohjelma saa syötteenä listan trukkitöistä vuorokauden ajalta sekä kartan hyllypaikkojen sijainnista, ja jakaa tehtäviä trukeille valitun optimointistrategian mukaan. Ohjelman laskemia reittejä ja tehtävien suoritusaikaa verrataan siihen aikaan, missä tehtävät todellisuudessa saatiin tehtyä. Lisäksi arvioidaan trukkien ajomatkaa lähtötilanteessa vertailua varten.

Jotta simulaation tuloksia voi hyödyntää, täytyy lähtötietojen olla mahdollisimman realistisia. Trukkien reittitietoja ja tehtävien suoritusaikoja täytyy

kerätä tai arvioida jollain tavalla. Tutkimuksen varastossa on käytössä varastonhallintajärjestelmä, johon tallentuu tieto tehtävän aloitus- ja lopetusajasta. Tämän perusteella voidaan arvioida tehtävien suoritusaikaa. Trukkien reittejä arvioidaan tehtävien tietojen ja varaston pohjakartan perusteella.

Kartan ja menneiden trukkitöiden perusteella voidaan laatia simulaatio-ohjelma ja kokeilla, miten valittu menetelmä vaikuttaa tehtävien suoritusnopeuteen ja trukkien ajomatkoihin. Lisäksi simulaation lähtötiedoiksi kerättävät havainnot trukkitöistä, esimerkiksi tehtävien keskimääräiset suoritusajat, sekä simulaation tulokset voivat olla varaston pitäjälle arvokkaita, vaikka uusia menetelmiä ei otettaisikaan käyttöön.

1.1 Ongelman kuvaus

Tässä työssä tutkitaan siirtotilaustehtävien optimointia eräässä korkeavarastossa, jossa tehtäviä suorittavat nostotrukit. Varaston kaikki siirtotilauspyynnöt tulevat varastonhallintajärjestelmästä, ja pyynnössä on määritelty tavaran lähtö- ja kohdepaikka. Tällä hetkellä trukit tekevät aina joko täyttö- tai hyllytystehtäviä. Asiakas pohtii, olisiko tehtävien kokonaissuoritusaikaa mahdollista lyhentää muuttamalla trukkitöiden ohjausta. Uudessa mallissa kaikki avoimet tehtävät päätyisivät yhteen jonoon. Jonosta kaikille trukeille jaettaisiin tehtäviä niin, että trukkien ajoreiteistä muodostuu mahdollisimman lyhyitä. Tutkin simulaatio-ohjelmalla uuden ohjaustavan vaikutusta tehtävien suoritusaikaan ja trukkien ajomatkaan kahta eri reititysalgoritmia käyttäen.

Tällä hetkellä työntekijät voivat vapaasti poimia seuraavan tehtävän listalta. Tutkitaan myös, muuttuuko tehtävien suoritusaika, kun seuraava suoritettava tehtävä tulee ohjelman kautta valmiiksi määrättynä. Lisäksi arvioidaan, miten tämäntyyppinen optimointisovellus voitaisiin toteuttaa varsinaiseen käyttöön.

Varastonhallintaprosesseja tai varaston rakennetta ei tässä vaiheessa ole mahdollista muuttaa, joten täydennysten ajankohtaa tai hyllyjen sijainnin optimointia ei ole järkevää tutkia. Poimijat kulkevat ääniohjauksen perusteella jo jossain määrin optimoituja reittejä. Siirtotilausten optimoinnissa kannattaa siis kiinnittää huomio siihen, miten tehtävät saadaan tehtyä niin, että tyhjänä ajomatka minimoidaan. Tyhjänä ajomatka minimoidaan valitsemalla trukin seuraava tehtävä mahdollisimman läheltä edellisen tehtävän päätepistettä.

Pohjakartan ja varastohallintajärjestelmään tallentuneiden historiatietojen perusteella pystytään arvioimaan, mitä tehtäviä on suoritettu ja miten pitkä

aika niiden suorittamiseen on kulunut. Lisäksi voidaan arvioida trukkien kulkemia reittejä sillä oletuksella, että ne ovat kulkeneet aina edellisen tehtävän loppupisteestä seuraavan tehtävän alkupisteeseen. Tarkempia laskelmia varten olisi kuitenkin hyödyllistä tuntea kaikkien lavojen tarkat sijainnit ja trukkien käyttämät reitit. Tätä varten pitäisi olla käytössä jonkinlainen paikannusjärjestelmä.

Ajomatkan arviointi ilman, että tiedetään todellisia reittejä, on väistämättä epätarkkaa. Tämän vuoksi käsittelen hieman myös erilaisia sisätilapaikannusmenetelmiä varastoissa. Tutkin, miten paikannustulosten avulla voisi luoda tarkemman laskelman ja miten paikannuksen voisi ottaa käyttöön optimoinnin osana, jos optimointisovellusta halutaan todellisuudessa hyödyntää.

1.2 Työn rakenne

Työ koostuu 7 luvusta. Toisessa luvussa käsitellään varastoprosesseja lyhyesti. Kolmannessa luvussa käsitellään varasto-optimointia ja trukkitöiden reititystä. Optimointimenetelmistä keskitytään erityisesti sijainnin huomioiviin menetelmiin ja tapoihin, joita voidaan hyödyntää, kun varaston rakennetta tai toimintaprosesseja ei voida radikaalisti muuttaa. Neljännessä luvussa esitellään simulointiohjelman toteutus ja käydään lyhyesti läpi varaston simulointimenetelmiä. Tulokset käsitellään viidennessä luvussa. Kuudennessa luvussa käsitellään jatkokehitysehdotuksia. Viimeinen luku kokoaa yhteen johtopäätökset.

Luku 2

Varastoprosessit ja varaston hallinta

Varastot ovat tärkeä osa yrityksen logistiikkaketjua. Niiden tarkoituksena on muodostaa puskuri kysynnän vaihteluita varten ja kerätä tilauksia eri toimittajilta yhteen kuljetukseen asiakkaalle. Lisäksi niillä on erilaisia lisäarvoa tuottavia toimintoja [11]. Jos rakennuksen päätarkoitus on säilyttää tuotteita, kutsutaan sitä varastoksi. Jos tuotteiden jakelu on rakennuksen päätarkoitus, käytetään usein termiä jakelukeskus.

Varaston toiminta koostuu varastonimikkeiden vastaanottamisesta, säilyttämisestä, tilausten vastaanottamisesta ja toimittamisesta. Tilausten toimittamiseen liittyy myös tilaukseen kuuluvien nimekkeiden poimiminen aktiivipaikoilta ja pakkaaminen tilaukseen. Lisäksi toimintaan kuuluu aktiivipaikkojen täydentäminen reservipaikoilta. Tämä työ käsittelee erityisesti trukkitöitä, eli hyllytystä ja aktiivipaikkojen täydennystä, joiden sijoittuminen varastointiprosessiin on kuvattu kuvassa 2.1. Trukkitöiden eli siirtotilausten tarkoituksena on mahdollistaa sujuva tuotteiden vastaanotto ja keräily.

Varastossa pyritään siihen, että tavara virtaisi varaston läpi jatkuvasti [1]. Aina, kun tavara siirretään johonkin väliaikaiseen paikkaan, se tarvitsee siirtää sieltä myös pois, mikä aiheuttaa ylimääräisiä kustannuksia. Toinen tärkeä periaate on varastonimikkeen sijainnin jatkuva tunteminen. Nimikkeen tunniste luetaan aina, kun nimike siirtyy johonkin, ja uusi sijainti tallennetaan. Jos näihin tietoihin lisätään vielä aikaleimat, pystytään varaston toimintaa seuraamaan tehokkaasti.

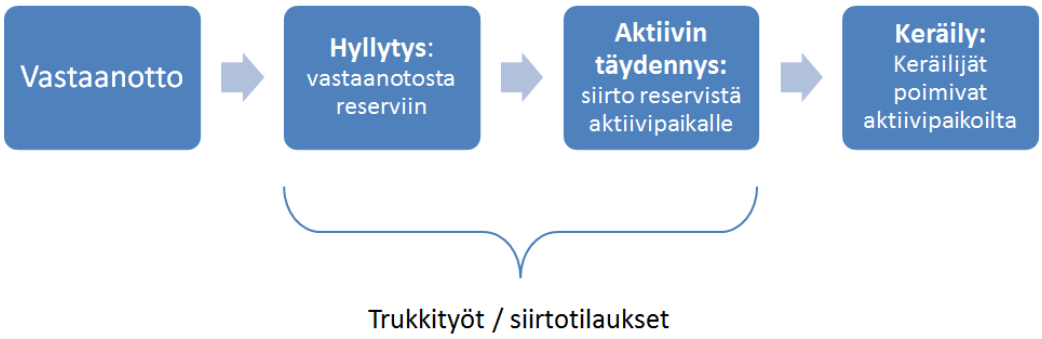
Trukkien tehtävät varastossa liittyvät erityisesti säilytykseen (sisältäen kuljetuksen vastaanottoalueelta varastoon), keräilyyn ja täydentämiseen. Nostotrukit, joita tässä tutkimuksessa käsitellään, kuljettavat vastaanotetut va-

rastonimikkeet saapumisalueelta reservipaikoille ja siirtävät tuotteita reservipaikoilta aktiivipaikoille.

Keräilyä tekevät ääniohjauslaitteilla varustetut keräilijät, jotka poimivat yksittäisiä tuotteita keräilyvaunuun. Siirtotilaustehtäviä suorittavat trukit voivat myös tarvittaessa siirtää kokonaisia lavoja lähettämöalueelle. Tätä kutsutaan täyslavakeräilyksi. Täyslavakeräilyä ei huomioida tässä tutkimuksessa tarkemmin.

Varastoprosessit ovat samantyyppisiä kaikissa varastoissa, mutta niiden toteuttaminen voi poiketa toisistaan paljonkin varastosta riippuen. Varaston toimintaan vaikuttaa esimerkiksi varastonhallintajärjestelmän toteutus tai sen olemassaolo ylipäätään, varaston rakenne ja automatisoinnin aste.

Varastojen tutkimuksella on joitakin erikoispiirteitä. Suuri osa varastojen tutkimuksesta on analyysikeskeistä, keskittyy tiettyyn osaan varaston toimintaa ja usein vielä yksittäiseen tapaukseen. Moni varastonohjaustutkimus, varsinkin simulaatiomalliin perustuva, ei siten ole helposti yleistettävissä.



Kuva 2.1: Trukkitöiden osuus varastointiprosessista

2.1 Vastaanotto

Vastaanotossa tuotteet kuitataan saapuneiksi ja puretaan vastaanottoalueelle. Varastosta riippuen vastaanottovaiheeseen voi liittyä esimerkiksi tavaran kunnon tarkastaminen. Vastaanoton osuus varaston toimintakustannuksista on yleensä melko pieni, noin 10 prosenttia [1]. Vastaanottokustannuk-

sia vähentää tehokas tuotetunnisteiden lukutapa, esimerkiksi RFID-tagien käyttö.

2.2 Hyllytys

Hyllytys on tavaroiden siirtoa vastaanottoalueelta säilytyspaikalle. Säilytyspaikkojen sijainti määrittää, miten pitkiä matkoja tavaroita joudutaan siirtämään tilauksia varten, joten paikkojen sopiva sijainti on tärkeää. Matkat vastaanottoalueelta säilytyspaikoille voivat olla pitkiä, joten hyllytyksestä aiheutuu yleensä jonkin verran kustannuksia. Hyllytyskustannukset ovat noin 15 prosenttia varaston toimintakustannuksista [1]. Lisäksi on tärkeää, että tuotteiden hyllypaikat tallennetaan, jotta keräily myöhemmin on tehokasta.

Tutkimuksen varasto käyttää aktiivi- ja reservialueita jaotteluun. Tavara siirretään vastaanottoalueelta reservipaikoille, jotka sijaitsevat hyllyjen yläosissa. Keräily tapahtuu aktiivipaikoilta, jotka ovat hyllyjen alaosassa ja joista keräilijöiden on helppo poimia tuotteet. Yksi trukkiryhmä suorittaa täyttötehtäviä, eli siirtää tavaroita reservistä aktiivipaikoille niiden tyhjentyessä.

2.3 Keräily

Keräilyssä tavaraa poimitaan varastopaikoilta asiakkaalle tilauksen toimitusta varten. Tilaus koostuu riveistä, joista jokaisella on yksi nimike. Usein monia tilauksia on lähdössä samaan aikaan ja yhdessä tilauksessa on monta riviä, joten keräilyreitin valinnalla on merkitystä. Keräily muodostaa 55 % varaston toimintakustannuksista, ja näistä kustannuksista 55 % on kulkukustannuksia [1]. Keräilyä onkin tutkittu paljon ja eri keruumenetelmiä on kehitetty useita.

Keräilytavat voidaan karkeasti jaotella keräilijä tavaran luo -malliin, jossa keräilijä kulkee hyllypaikalta toiselle ja poimii tavaroita, yleensä keräilyvaunuun tai trukkiin, ja tavara keräilijän luo -malliin, jossa tuote tuodaan keräilijän luo kuljetinjärjestelmällä tai keräilijä poimii sen karusellista.

Keräilyn nopeuttamiseksi varastoissa on usein käytössä esimerkiksi ääniohjaus, joka kertoo seuraavan tuotteen sijainnin, tai valo-ohjaus, joka näyttää oikean hyllypaikan. Ohjausjärjestelmä voi myös optimoida reittejä esimerkiksi laskemalla keräilijöille sellaiset reitit, että yhdellä pysähtymisellä voi poimia usean tuotteen.

Keräily tehdään usein eri tekniikalla kuin muut varaston työt. Tämänkin työn varastossa keräily tehdään keräilyvaunujen avulla, joihin voi poimia useita tuotteita. Siirtotilauksissa taas siirretään yksittäisiä lavoja trukeilla. Tämän takia keräilyä ei käsitellä tarkemmin tässä työssä.

2.4 Täydennys

Täydennys tai (aktiivin) täyttö tarkoittaa varastonimikkeiden täydentämistä keräilijöitä varten. Keräilijät poimivat yleensä yksittäisiä tuotteita ja täydentäjät siirtävät suurempia eriä keräilypaikoille. Ohjeena voi pitää, että viittä keräilijää kohden tulee olla yksi täydentäjä [1]. Täydennys on kalliimpaa kuin keräily, koska täydentäjän pitää myös avata pakkaus, jotta keräilijät pääsevät poimimaan tuotteita.

Monissa varastoissa säilytyspaikat ja keruupaikat on eroteltu toisistaan tehokkuuden vuoksi [7]. Tavallisin tapa jakaa varasto on pitää keräiltävät tuotteet hyllyn alaosassa maan tasalla ja kokonaiset lavat hyllyn yläosassa [1]. Myös tutkimuksen varasto on toteutettu näin.

Tutkimuksen varastossa täydennystehtävä syntyy, kun keräilijä poimii viimeisen tuotteen varastopaikalta, tai jos tarvittavaa tuotetta ei löydy paikalta. Täydennystehtävät on tärkeää tehdä nopeasti, koska kerääjä joutuu odottamaan, jos paikka on tyhjä.

2.5 Varastonhallintajärjestelmät

Varastonhallintajärjestelmä (warehouse management system, WMS) on järjestelmä, joka pitää kirjaa varaston tuotteista. Vähimmillään järjestelmän tulee rekisteröidä tuotteen saapuminen ja lähteminen varastosta. Varastonhallintajärjestelmä tuntee myös yleensä tuotteiden tarkat sijainnit ja tallentaa kaikki tuotteen siirrot. Koska järjestelmä tuntee tuotteiden sijainnit, se voi muodostaa tehokkaasti järjestettyjä keruulistoja keräilijöille tilausten toimitusta varten.

Erilaisia järjestelmiä on lukuisia. Myös jotkut toiminnanohjausjärjestelmät, kuten SAP [22], sisältävät varastonhallintatoiminnallisuutta.

Varastonhallintajärjestelmä ei kuitenkaan yleensä optimoi varaston toimintaa muuten [1]. Lisäksi varastonhallintajärjestelmät hyödyntävät harvoin tutkimustuloksia varaston ohjauksesta ja toiminnasta [12]. Järjestelmät ovat

kuitenkin usein räätälöitävissä ja niihin voi lisätä moduuleja toiminnallisuuden lisäämiseksi.

Tutkimuksessa toteutettava ohjelman tapainen ohjausjärjestelmä toimisi todellisuudessa varastonhallintajärjestelmän kanssa yhteistyössä. Ohjelma saa järjestelmältä tiedot avoinna olevista tehtävistä, laskee trukeille seuraavat tehtävät ja ilmoittaa järjestelmälle, mitkä tehtävät on valittu, jotta ne poistuvat avointen tehtävien listalta. Lisäksi järjestelmä pitää kirjaa siitä, milloin tehtävää alettiin suorittaa, milloin tehtävä valmistui, ja kauanko trukki odotti seuraavaa tehtävää. Näitä tietoja ei tällä hetkellä tallennu, joten siirtotilauksia voitaisiin ohjelman avulla seurata huomattavasti tarkemmin.

Luku 3

Trukkitöiden optimointi

Varastojen toiminnasta ja suunnittelusta on olemassa suuri määrä tutkimuksia. Kattavia yleiskatsauksia varastojen toiminnan ja ohjauksen tutkimukseen ovat laatineet esimerkiksi Gu et al. [11], [12]. Tutkimuksista ensimmäinen käsittelee varaston ohjausta ja toinen varaston suunnittelua. Keskityn tässä trukkitöiden optimointiin ja erityisesti metodeihin, joita voidaan hyödyntää, kun varaston rakennetta tai tuotteiden sijoittelua ei ole mahdollista muuttaa. Tällöin keskitytään lähinnä datan keruuseen ja havainnointiin, jonka perusteella voidaan analysoida toimintatapoja ja ohjata ja optimoida työtä. Myös automaatiota lisäämällä voidaan vaikuttaa töiden valmistumisnopeuteen ja kustannuksiin, mutta tässä työssä keskitytään olemassa olevia (= kyseisen varaston tällä hetkellä käyttämiä) työvälineitä hyödyntäviin järjestelmiin.

Tässä luvussa käsittelen erityyppisiä trukkitöiden optimointimenetelmiä yleisellä tasolla sekä tarkemmin trukkien reittien optimointia. Työssä toteutettu optimointimenetelmä on yksinkertaisempi kuin tässä esitellyt. Emme havainnoi trukkien ympäristöä tai paikanna niitä lähimmän tehtävän selvittämiseksi, vaan trukin sijainti uutta tehtävää valitessa arvioidaan edellisen tehtävän loppupisteen perusteella. Menetelmät on kuitenkin sisällytetty kappaleeseen, koska ne havainnollistavat, miten erilaisia mahdollisuuksia trukkitöiden optimointiin on. Lisäksi tutkimuksessa toteutettua työkalua olisi mahdollista laajentaa, jos varastossa esimerkiksi otetaan käyttöön paikannusjärjestelmä.

3.1 Optimointi

Trukkitöille on yleensä valmiiksi määritelty siirron alku-ja loppupiste, eikä optimointivaraa varsinaisessa siirrossa välttämättä ole. Trukkitöihin liittyy paljon tuottamatonta tyhjänä ajamista, mutta tehtäviä on harvoin tiedossa kerrallaan niin paljon, että monelle tehtävälle voisi määritellä optimaalisen reitin. Lisäksi trukkitöihin liittyy paljon manuaalisia työvaiheita, kuten lavan tunnisteen skannaus, tehtävän kuittaaminen järjestelmään jne., joissa voi tapahtua virheitä. Virheet vaikuttavat nopeasti koko varaston toimintaan. Esimerkiksi keräily hidastuu, jos lava on kuitattu väärään paikkaan. Trukkityöt olisikin tärkeää tehdä mahdollisimman tehokkaasti ja virheettä. Ohjauksessa voidaan keskittyä virheiden vähentämiseen ja tehtävien jakamiseen mahdollisimman järkevästi, jos varsinaisissa siirroissa ei ole paljon optimoitavaa.

Trukkitöiden ohjaukseen on kehitetty jonkun verran sovelluksia. Monet perustuvat jonkinlaisen paikannusjärjestelmän käyttöönottoon, jonka avulla trukkien sijaintia voidaan seurata ja jakaa trukeille tehtäviä, jotka ovat mahdollisimman lähellä niiden sijaintia. Varastosta voidaan havainnoida myös muuta liikennettä ja välttää törmäyksiä. Lisäksi on mahdollista automatisoida lavojen sijainnin tallennus varastonhallintajärjestelmään esimerkiksi lavojen RFID-tagien ja trukin liikehavaintojen perusteella. Uusimmissa sovelluksissa hyödynnetään lisättyä todellisuutta, esimerkiksi näytetään, mihin lava tulee viedä.

Myös toteutettavaan optimointisovellukseen olisi mielenkiintoista lisätä todelliset havainnot trukkien sijainnista ja niiden perusteella määrätä seuraava tehtävä, sekä havainnoida siirtoihin kuluvaa aikaa. Lisäksi todellista sijaintia seuraamalla saataisiin tietää trukkien kulkemat tarkat reitit. Reittitiedon pohjalta voitaisiin tehdä tarkempia päätelmiä siitä, miten trukkityöt tällä hetkellä suoritetaan ja sen perusteella kehittää uusia tapoja töiden ohjaukseen. Sovelluksen kehitysvaiheessa sijaintitietoja ei kerätty muuten kuin arvioimalla, mutta luvussa 6 käsittelen, miten sijaintihavaintoja voisi mahdollisimman pienin muutoksin hyödyntää varastossa.

3.1.1 Sensorihavaintoihin perustuva seuranta ja ohjaus

Trukkitöiden optimoinnissa tehtävien suoritusjärjestys ja suoritusnopeus on usein ainoa asia, johon voidaan vaikuttaa. Erilaisten sensorien avulla voidaan havainnoida ympäristöä ja hyödyntää saatua tietoa seuraavan tehtävän valinnassa. Lisäksi etähavaintojen perusteella voidaan vähentää käsin tehtävää

työtä ja siten nopeuttaa trukkitöitä. Jos havaitaan, että trukki on poiminut lavan, ja lavassa on etäluettava tagi tunnisteena, se voidaan lukea ja automaattisesti päivittää varastonhallintajärjestelmään, että siirtotilaus on aloitettu. Kun lava siirretään paikoilleen, tehtävä voidaan kuitata päättyneeksi. Tällainen seurantajärjestelmä on toteutettu Fraunhofer-instituutissa Saksassa [21].

Dataa trukkien sijainnista, akkujen tasosta ym. voidaan suoraan hyödyntää varaston toiminnan optimoinnissa. Estanjini et al. rakensivat järjestelmän, joka havainnoi trukkien sijaintia, käyttöaikaa, akun tasoa ja käytävien ruuhkaisuutta, ja sen perusteella jakaa tehtäviä trukeille [8]. Lisäksi järjestelmällä on tiedossa varaston inventaario. Tapauksessa varasto on hyvin samanlainen kuin tässäkin tutkimuksessa, ja käsiteltävät tehtävätyypit ovat hyllytys ja täydennys. Kun trukki saa edellisen tehtävän valmiiksi, järjestelmä määrittää sille uuden tehtävän, joka voi olla joko täydennys, hyllytys, trukin vienti lataukseen tai huoltopisteeseen tai odotus (sopivaa tehtävää ei löydy tai käytävillä on ruuhkaa). Tutkimuksessa tehtävät eivät kuitenkaan tule varastonhallintajärjestelmässä vaan mallissa on neljä varastonimikettä. Jos joku nimike odottaa vastaanottoalueella tai sen paikka on tyhjä, on trukin tehtävälistassa valittavana nimikkeen hyllytys tai täydennys.

Estanjini et al. [8] havaitsivat, että ruuhka käytävillä vähentyi selvästi ja trukit pystyivät siten suorittamaan tehtäviä aiempaa nopeammin. Sijainnin määrittäminen luotettavasti oli kuitenkin hankalaa, ja osalla alueista trukit paikannettiin oikein vain alle puolet ajasta.

3.1.2 Sisätilapaikannus

Sisätilapaikannusta hyödyntämällä voidaan seurata trukkien ja lavojen todellista sijaintia ja sen perusteella optimoida trukkien ja keräilijöiden reittejä. Paikannuksen voi toteuttaa esim. erilaisilla sensoreilla, WLANin tai bluetoothin avulla, GPS:llä tai optisesti.

Bluetooth-beaconit ovat muutaman viime vuoden aikana yleistyneet sisätilapaikannuksessa merkittävästi. Niillä on toteutettu esimerkiksi kauppakeskusten tai museoiden sisäistä paikannusta. Tämä voisikin olla edullinen ja helppo tapa seurata trukkien liikettä, vaikka senttimetritarkkaan paikannukseen (esimerkiksi lavojen tarkan sijainnin havaitsemiseen) ei päästäisi.

Paikannusta hyödyntämällä voidaan myös seurata tuotteiden liikkumista suhteellisen helposti, verrattuna esimerkiksi RFID-tagien kiinnittämiseen kaikkiin lavoihin ja hyllypaikkoihin. Yhden melko varhaisen WLAN-pohjaisen

paikannusjärjestelmän varastossa toteuttivati Ibach et al. vuonna 2005. He asensivat viivakoodinlukijoihin paikannusohjelman [14]. He olettivat, että skannauksen yhteydessä tuote on lähellä lukulaitetta, joten sijainnin seuraamiseen riittää tuntea lukijan sijainti sekä luettu koodi. Lisäksi sovellus vaatii vähän resursseja ja toimii trukin käsipäätteellä.

WLAN-paikannus voi kuitenkin olla sisätiloissa melko epätarkkaa muun muassa metallihyllyjen aiheuttamien heijastusten takia. Ibach et al. pääsivät 3,25 metrin tarkkuuteen, mikä ei vielä riitä hyllypaikan tarkkuudella paikantamiseen.

Varaston sisäistä paikannusta on toteutettu myös optisesti, kameraan perustuen. Katsaus tekniikkaan yleisesti on esimerkiksi [17]. Paikannuksessa voidaan päästä millimetrien tarkkuuteen sisätiloissa. Paikannuksen toteutustavasta riippuen voidaan tosin joutua tekemään ympäristöön muutoksia.

Optinen paikannus voidaan toteuttaa eri tavoin. Yksi mahdollisuus on käyttää tunnistettavia kuvioita ympäristöstä, joko olemassaolevia maamerkkejä tai koodattuja kuvioita, joiden sijainnit on tallennettu etukäteen. Muutama kaupallinen varastonhallintajärjestelmä, Zenotrack [27] ja TotalTrax [26], toteuttaa paikannuksen näin.

Sisätilapaikannuksessa voidaan myös yhdistää useita sensoreita, esimerkiksi hyödyntää wifi-signaalia ja optista paikannusta [18]. Älypuhelimessa on on lukuisia sensoreita, joita voi hyödyntää paikannuksessa. Useita erityyppisiä havaintoja käyttävän varastotietojärjestelmän toteuttivat esimerkiksi Chen et al. [4]. Puhelin paikannetaan wifin avulla, lisäksi hyödynnetään tietoa puhelimen suunnasta ja liikkeistä. Käsiteltävät kohteet ovat kirjoja, ja ne tunnistetaan vertaamalla otettua valokuvaa etukäteen muodostettuun kuvatietokantaan.

3.1.3 Lisätty todellisuus

Älylaseista on viime vuosina puhuttu paljon varasto-optimoinnin yhteydessä. DHL:n lisättyä todellisuutta logistiikassa käsittelevässä raportissa vuodelta 2014 on maininta, että ainakin kolme yritystä kehittää lisättyä todellisuutta hyödyntävää järjestelmää varastonhallintaan [10].

Älylaseille on helppo nähdä sovelluksia esimerkiksi keräilijöiden ohjauksessa. Sebastian Pickl tutki keräilyä älylasien avulla kandidaatintyössään. Verrattuna muihin keräilijöiden ohjaustekniikoihin kuten ääniohjaukseen tai paperiseen tilauslistaan, keräily lasien kanssa aiheutti vähiten virheitä. Keräily lasien kanssa oli kuitenkin hitain tapa, ja osa osallistujista valitti lasien ras-

kautta ja näytön hitautta. [19] Tutkimusta trukkitöiden ohjauksesta älylasien avulla en löytänyt, mutta lasien avulla voitaisiin esimerkiksi näyttää, mistä seuraava lava tulee hakea ja mihin se tulee viedä, kuten keräilijöiden ohjauksessa. Lasien näytöllä voitaisiin myös näyttää nopein reitti paikalle. Tämä voisi vähentää virheitä trukkitöissä. Lisättyä todellisuutta hyödyntävät järjestelmät tarvitsevat myös toimivan sisätilapaikannustavan.

3.2 Reittioptimointi

Reititys varastossa on paljon tutkittu ongelma. Vuonna 2006 Jinxiang Gu et al.:n kirjallisuuskatsauksessa 38 prosenttia tutkitusta aineistosta käsitteli reititysongelmia [11]. Suurin osa tästä on keräilijöiden reitityksen tutkimista, koska keräily muodostaa niin suuren osan varaston kustannuksista.

Täydennysten tai hyllytysten reititystä ei ole tutkittu paljon. Yksi syy voi olla näiden tehtävien pienempi osuus toimintakustannuksista ja erilainen toteutustapa. Hyllytys- ja täydennystehtäviä tekee huomattavasti pienempi määrä työntekijöitä kuin keräilyä; täydennystä tekee yleensä 1 henkilö viittä keräilijää kohti. Lisäksi siirtotilauksissa siirretään yleensä kokonainen lava kerrallaan ja siirron alku- ja loppupiste on ennalta määrätty, joten tehtävän reitille on vain yksi vaihtoehto. Keräilyssä taas haetaan yksittäisiä tuotteita, joten keräilijän kannattaa valita reitti, joka käy tarvittavat sijainnit läpi mahdollisimman kätevästi.

Varastonimikkeiden säilytyspaikkojen sijainnista (storage location problem) on olemassa paljon tutkimusta, koska sijainti vaikuttaa suoraan hyllytyksen suoritusnopeuteen ja keräilijöiden reitteihin. Tässä työssä ei kuitenkaan käsitellä vaihtoehtoisia sijainteja hyllypaikoille.

Reititys varastossa voidaan usein mallintaa ajoneuvon reititysongelmana, jossa joukko ajoneuvoja lähtee samasta pisteestä, poimii tuotteita eri sijainneista ja palaa lähtöpisteeseen. Siirtotilaukset ovat ajoneuvon reititysongelman erikoistapauksia, Vehicle Routing Problem with Pickup and Delivery. Tässä erikoistapauksessa tehtävät muodostuvat tuotteen poimimisesta ja viemisestä johonkin paikkaan, kun perinteisessä reititysongelmassa riittää vierailla eri pisteissä. Tehtävien suorittaminen yksi kerrallaan voidaan mallintaa asettamalla ajoneuvon kapasiteetti ongelmassa yhdeksi.

3.2.1 Ajoneuvon reititysongelma

Ajoneuvon reititysongelma (Vehicle Routing Problem, VRP) on kombinatorinen optimointiongelma, joka on merkittävä varsinkin logistiikan alalla. VRP:n esittivät ensimmäisen kerran Dantzig ja Ramser vuonna 1959 [6]. VRP:ssä ratkaistavana on optimaaliset reitit joukolle ajoneuvoja, joiden tulee käydä läpi joukko asiakkaita. VRP on yläkäsite suurelle joukolle reititysongelmia. Ratkaistavan ongelman tyyppi riippuu kuljetettavista tuotteista, ajoneuvoista ja palvelun tasosta.

Viime vuosikymmeninä on tutkittu paljon VRP:n monimutkaisempia vaihtoehtoja, niin kutsuttuja rikkaita VRP:tä, joissa on enemmän rajoitteita ja jotka siten kuvaavat todellisia tilanteita tarkemmin. Näitä variantteja ovat esimerkiksi VRP with Time Windows, jossa tehtävät tulee tehdä tietyssä aikaikkunassa, Capacitated VRP, jossa huomioidaan ajoneuvojen kapasiteetti, ja VRP with Pickups and Deliveries (tai Pickup and Delivery Problem), jossa tehtävillä on määrätyt alku- ja loppupisteet. Tutkimuksen kohteena olevat siirtotilaukset voidaan kuvata tällaisena ongelmana.

3.2.2 VRPPD

VRPPD, Vehicle Routing Problem with Pickup and Delivery on VRP:n variantti, jossa tehtävät koostuvat noudoista ja vienneistä. Jokaisella tehtävällä on siis määrätty alku- ja loppupiste. Ongelma on osa laajempaa PDP (Pickup and Delivery Problem)- ongelmien joukkoa. Tehtävänä on määritellä ajoneuvoille reitit, joilla on mahdollisimman pieni kustannus ja jotka käyvät läpi kaikki kohteet. Jos jokaisella tehtävällä on vain yksi mahdollinen lähtö- ja kohdepaikka, ongelmaa kutsutaan one-to-one -PDP:ksi. [5] PDP-ongelmia esiintyy usein tavaroiden ja ihmisten kuljetuksessa, esimerkiksi kutsutaksien reittien suunnittelu on PDP-ongelma.

Trukkitöiden optimointiongelma voidaan kuvata seuraavasti. Tehtävä tai pyyntö i on tietyn kuorman d_i kuljetus alkupisteestä i^+ loppupisteeseen i^-. R on mahdollisten pyyntöjen joukko. Ongelma on määritelty suunnatulla graafilla $G = (V, A)$, jossa V on alku- ja loppupisteiden (graafin solmujen) joukko ja A on kaarien joukko. V on määritelty $V = \{i^+ | i \in R\} \cup \{i^- | i \in R\}$. A on määritelty $A = \{(i, j) : i, j \in V, i \neq j\}$. Jokaisella kaarella on kustannus c_{ij} ja matka-aika t_{ij}, jotka ovat suurempia kuin 0.

Reitti on polku graafissa, joka käy joidenkin solmujen kautta. Tehtävän alku- ja loppupisteet eivät voi olla eri reiteillä, ja alkupisteen on oltava ennen loppupistettä. PDP-ongelmissa määritellään yleensä, että reitin tulee alkaa ja

päättyä varikkopisteeltä. Todellisuudessa reittien päät ovat kuitenkin usein avoimia, eli reitti päättyy viimeisen tehtävän loppupisteeseen. Myös trukkitöiden optimoinnissa on näin. Yksi reitti päättyy viimeisen tehtävän loppupisteeseen ja seuraavan laskentakierroksen reitti alkaa samasta pisteestä.

3.2.3 Dynaamiset ongelmat

Reititysongelma on staattinen, jos kaikki tehtävät ovat tiedossa etukäteen. Jos vain osa tehtävistä on tiedossa etukäteen ja uusia tehtäviä tulee reaaliajassa, kun reittien suoritus on meneillään, kyseessä on dynaaminen ongelma. [23] PDP-ongelmat ovat useimmiten dynaamisia, toisin kuin useimmat VRP-ongelmat. Myös käsiteltävä trukkitöiden optimointiongelma on dynaaminen ongelma - uusia siirtotilauspyyntöjä tulee jatkuvasti, kun uusia lavoja saapuu vastaanottoalueelle ja aktiivipaikkoja tyhjenee.

Dynaamisten ongelmien ratkaisuun on ainakin kaksi tapaa. Yksi vaihtoehto on ratkaista uusi staattinen ongelma aina, kun uutta tietoa ilmenee. Tämä voi kuitenkin olla liian hidasta reaaliajassa toteutettavaksi. Toinen, yleisempi vaihtoehto on muodostaa aluksi yksi ratkaisu olemassa olevan tiedon perusteella. Tätä ratkaisua päivitetään aina, kun uusia pyyntöjä ilmestyy. Päivityksessä käytetään erilaisia heuristiikkoja ja paikallista hakua. [2].

Trukkitöiden optimoinnissa yksi trukki voi kuljettaa vain yhtä lavaa kerrallaan, kun taas PDP-ongelmissa perinteisesti ajoneuvojen kapasiteetti on suurempi kuin yksi. Dynaamiset VRPPD-ongelmat, joissa on suoritettava yksi tehtävä kerrallaan, muodostavatkin oman ryhmänsä. Kirjallisuudessa tällaisia ongelmia kutsutaan nimellä Dynamic Stacker Crane Problem tai Dynamic Full Truckload Pickup and Delivery Problem. [2]. Näiden ratkaisuun on kehitetty algoritmeja, jotka huomioivat uusia tehtäviä jatkuvasti ja toimivat reaaliajassa.

Esimerkiksi Gutenschwager et al. toteuttivat automaattisen varaston kuormankuljettimia ohjaavan järjestelmän, jolla pystyttiin vähentämään tarvittavien kuljettimien määrää 24:stä 22:een [13]. Ongelma piti pystyä ratkaisemaan tiukasti reaaliajassa, kyselyyn seuraavasta tehtävästä piti saada vastaus 0,1 sekunnissa. Tämän takia ratkaisuun käytettiin heuristista optimointimenetelmää, tabuhakua.

Trukkitöiden optimointi tosin eroaa tyypillisestä stacker crane problem -ongelmasta lyhyemmän suunnitteluhorisonttinsa takia. Stacker crane -ongelmissa osa tehtävistä on yleensä tiedossa päiviä etukäteen ja reittien laskenta voidaan suorittaa jo päiviä ennen niiden ajoa.

3.2.4 Ratkaisualgoritmit

Dynaamisten ongelmien ratkaisualgoritmit eroavat toisistaan sen perusteella, miten tulevaisuus otetaan niissä huomioon. Ongelma voidaan ratkaista huomioimalla vain nykyhetkellä tiedossa olevat tehtävät tai muodostamalla lisäksi ennusteita kysynnästä tulevaisuudessa eri tekniikoin [20]. Käsittelen tässä lyhytnäköistä mallia, joka ottaa huomioon vain nykyhetkellä tiedossa olevat tehtävät. Tällöin tehtävänä on minimoida trukkien kulkema matka $\sum_{i,j \in A} c_{ij} i$, missä i on tehtävä ja c_{ij} on tehtävän matka (kustannus).

Tarkalleen ottaen halutaan minimoida tyhjänä kuljettu matka. Siirtotilausten sisäiset matkat ovat vakioita, eli voidaan vaikuttaa vain tehtävien välillä kuljettavaan, tyhjään, matkaan. Siten pienin kokonaismatka on myös pienin tyhjänä kuljettu matka. Matkan minimointi on VRP-ongelma.

VRP:lle on kehitetty lukuisia ratkaisualgoritmeja, sekä eksakteja että heuristiikkoihin perustuvia. Hyödynnettävä ratkaisu riippuu ongelman tyypistä. Käsittelen tässä VRPPD:n ratkaisualgoritmeja. VRPPD on yleistys kauppamatkustajan ongelmasta ja siten NP-vaikea.

VRPPD:n suosituimpia eksakteja ratkaisutapoja ovat sarakegenerointiin perustuvat algoritmit sekä branch-and-cut -algoritmit [5]. Käytettyjä heuristisia optimointimenetelmiä ovat tabuhaku, geneettiset algoritmit ja simuloitu jäähdytys. Heuristinen optimointi pyrkii 'tarpeeksi hyvän' ratkaisun etsimiseen. Heuristiset menetelmät ovat usein nopeampia ja pystyvät käsittelemään suurempaa joukkoa. Ratkaisu ei kuitenkaan ole välttämättä optimaalisin. Tässä työssä käytän heuristista optimointimenetelmää, koska laskennan on oltava mahdollisimman nopeaa. Lisäksi reitin ei ole pakko olla optimaalisin, kunhan tehtävät suoritetaan järkevässä ajassa.

Esittelen tarkemmin yhden heuristisen menetelmän, ruin-and-recreate -algoritmin. Simulaatiossa käytän tätä algoritmia reittien laskemiseen. Algoritmi valittiin, koska se ratkaisee tehokkaasti VRP-ongelmia, myös, kun mukaan lisätään noudot ja viennit. Algoritmin tuottamien ratkaisujen kustannus eroaa 0-5 prosenttia VRPPD-vertailuongelmien parhaiksi todetuista ratkaisuista [25].

Ruin-and-recreate- algoritmityypin esittivät Gerard Schrimpf et al. vuonna 2000 [24] ja sitä on sen jälkeen hyödynnetty erityisesti VRP:den ratkaisussa. Algoritmi kehitettiin ratkaisemaan erityisesti monimutkaisia ongelmia, joilla on paljon rajoitteita ja joita voidaan pitää 'epäjatkuvina', eli ratkaisun naapurustosta löytyvät ratkaisut voivat olla laadultaan suuresti poikkeavia. Algoritmityypissä etsitään ensin yksi ratkaisu, jonka jälkeen ratkaisu pilataan (ruin-vaihe) esimerkiksi poistamalla reiteiltä osa pisteistä, jolloin ratkaisu ei enää ole täydellinen. Tämän jälkeen ratkaisu palautetaan validiksi (recreate-

vaihe) lisäämällä pisteet takaisin joihinkin kohtiin reittejä. Sekä ruin-että recreate-vaiheeseen on useita erilaisia toteutustapoja.

Algoritmin perusperiaate on seuraava:

- Aloitetaan yhdestä mahdollisesta ratkaisusta.
- Päätetään ruin-tekniikka.
- Päätetään, montako solmua poistetaan ruin-vaiheessa.
- Pilataan ratkaisu.
- Korjataan ratkaisu.
- Päätetään, hyväksytäänkö ratkaisu ja jatketaanko siitä, vai aloitetaanko uusi laskentakierros edellisellä ratkaisulla.

Dynaamisessa PDP-ongelmassa algoritmin on tehtävä kahdentyyppisiä ratkaisuja: odottaa vai liikkua, tai hyväksyä vai hylätä. Odottaa vai liikkua -ratkaisu tulee tehtäväksi, kun yksi tehtävä on suoritettu. Ajoneuvo voi joko odottaa uutta tehtävää, tai liikkua johonkin pisteeseen, jossa on avoin tehtävä ajanhetkellä t. Hyväksyä vai hylätä -ratkaisu tulee eteen, kun uusi tehtäväpyyntö vastaanotetaan. Trukkitöiden ohjauksessa kaikki pyynnöt on otettava vastaan.

Luku 4

Toteutus

Jotta trukkitöiden uutta ohjaustapaa voitaisiin arvioida, toteutettiin sovellukset, jotka simuloivat trukkien toimintaa uusilla ohjaustavoilla. Meillä oli saatavilla varastonhallintajärjestelmästä trukkityöt kuukauden ajalta. Simulaatiota varten poimittiin tehdyt työt yhden vuorokauden ajalta.

Vuorokauden otoksesta laskettiin tilastoja tehtävien keskimääräisistä suoritusajoista yhteensä ja tehtävätyypeittäin. Lisäksi arvioitiin trukkien ajamaa matkaa olettaen, että trukki liikkuu edellisen tehtävän loppupisteestä suoraan seuraavan tehtävän alkupisteeseen. Simulaation aikana tallennetaan samoja havaintoja optimoinnin tehokkuuden arviointia varten.

Kuvassa 4.1 on kuvattu, miten optimointiohjelma voisi käytännössä toimia. Ohjelma lukee avoimet tilaukset SAP-järjestelmästä, ja kertoo lähimmän avoimen tehtävän trukin pyytäessä sitä. Tiedot tehtävän varaamisesta, lavan poimimisesta ja kuittauksesta tallennettaisiin esimerkiksi SAP:iin. Näin pystytään seuraamaan, miten kauan tehtävä oli jonossa, ja miten kauan tyhjänä ajomatka ja varsinainen siirtomatka kestivät.

Simulaatiovaiheessa tehtäviä ei kysellä SAP:sta, vaan ne luetaan suoraan tiedostosta.

4.1 Varaston simulointi

Varaston optimoinnin tutkimuksessa käytetään useimmiten työvälineenä simulointia [12]. Simuloinnilla voidaan tutkia toimintatapoja helposti tekemättä muutoksia varastoon ja yksinkertaistaa tilannetta helpommin käsiteltäväksi. Lisäksi simulaatiomallilla on helppo huomioida paljon yksityiskohtia. Simu-

lointia hyödyntävien tutkimusten heikkous on tulosten heikko yleistettävyys. Simulaatiomalli rakennetaan yleensä tietyn kohteen mukaiseksi. Tällöin tulokset ovat usein sidottuja kyseisen kohteen kaltaisiin varastoihin sekä itse simulaation toteutustapaan [12].

Tässä tutkimuksessa varastoa tutkitaan simuloimalla, koska sovelluksen kokeileminen käytännössä ei olisi ollut mahdollista. Simulaatiomalli valittiin yleisemmän, analyyttisen mallin sijaan myös siksi, että mallin tulee huomioida varaston yksityiskohdat ja optimointi on rajattu vain osaan varaston tehtävistä. Lisäksi simulaatiosovellusta kehitettiin osittain siitä näkökulmasta, että sen voisi muokkauksin ottaa käyttöön varastossa.

Myös moni pelkästään trukkien ohjausta käsittelevä tutkimus perustuu simulaatiomalliin. Esimerkiksi Estanjini et al. [8] tutkivat trukkien ohjausta sensoridatan avulla. He testasivat ohjausmenetelmän toimivuutta simuloimalla varastoa käyttäen oppivaa actor-critic -algoritmia. Gutenschwager et al. :lla [13] oli ratkaistavana hyvin samankaltainen dynaaminen reititysongelma kuin tässä työssä. He hyödynsivät oliopohjaista eM-Plant -simulointityökalua, joka perustuu tapahtumapohjaiseen simulaatioon.

Toteutin tutkimuksen simulaation tapahtumapohjaisena simulaationa. Tapahtumapohjaisessa simulaatiossa mallinnetaan tilanne sarjana tapahtumia ja keskitytään ajanhetkiin, jolloin tapahtuu jotain mielenkiintoista sen sijaan, että simulaatio etenisi aina tietyn aikamäärän verran simulaatioaskeleessa. Päädyin tähän toteutustapaan lähtöaineiston rakenteen takia. Tehtävät ilmestyvät jonoon tiettynä määriteltynä aikana ja tehtävien varastopaikkojen sijainti on tiedossa, joten on helppo laskea, miten kauan trukilta kestää suorittaa tehtävä ja milloin tehtävä tulee olemaan valmis.

Simulaatiossa tapahtumana käsitellään trukkityön valmistumista ja tehtävän saapumista jonoon. Trukit on mallinnettu hyvin yksinkertaisesti: niistä on tiedossa vain tämänhetkisen tehtävän valmistumisaika ja sijainti. Seuraava tehtävä määräytyy suoraan optimointistrategian perusteella. Jos trukeille lisättäisiin ominaisuuksia ja itsenäistä päätöksentekoa seuraavan tehtävän suhteen (esimerkiksi siirtyminen latauspisteeseen, jos trukin akku on tyhjä) agenttipohjainen simulaatio voisi olla parempi valinta. Tähän työhön tapahtumapohjainen simulaatio kuitenkin riittää hyvin.

Uusi simulaatiokierros käynnistyy, kun jonkun trukin tehtävä valmistuu. Simulaation kelloa siirretään kyseiseen ajanhetkeen ja katsotaan listalta, mitkä tehtävät ovat vapaana tuona ajanhetkenä. Trukille joko annetaan uusi tehtävä tai, mikäli tehtäväjono on tyhjä, siirretään simulaation kelloa eteenpäin ajanhetkeen, jolloin seuraava tapahtuma ilmenee. Tapahtuma voi olla joko tehtävän valmistuminen tai tehtävän saapuminen jonoon. Tallennetaan lisäksi seuraa-

via tilastoja simulaation aikana:

- trukkien ajoaika yhteensä
- trukkien odotusaika yhteensä
- trukkien tyhjänä ajama matka
- tehtävien jonossaoloaika
- paljonko tehtäviä simulaation aikana suoritettiin
- tehtävän keskimääräinen suoritusaika

Ajomatkoja laskettaessa käytetään hyväksi sitä, että kaikkien hyllypaikkojen sijainti tunnetaan. Varaston kartan avulla voidaan ennen simulaatiota laskea etäisyydet kaikista hyllypaikoista kaikkiin. Näin pystytään käyttämään todellisia etäisyyksiä laskennassa.

4.2 Lähtötilanne ja aineisto

Aineistona oli tukkuvarasto, jossa on n. 3000 hyllypaikkaa. Trukkien tehtäviä varastossa ovat keräily, hyllytys ja aktiivin täyttö. Keräily poikkeaa huomattavasti aktiivin täytöstä ja hyllytyksestä, joten sitä ei huomioida simulaatiossa. Trukit tekevät pääasiallisesti vain yhtä tehtävätyyppiä, mutta voivat tarvittaessa avustaa muuallakin. Lähtöarvoja laskettaessa on kuitenkin oletettu, että yksi trukki tekee yhdentyyppisiä tehtäviä.

Trukkitöiden ohjauksessa pyritään minimoimaan kahteen tehtävään, hyllytykseen ja aktiivin täyttöön, kuluvaa aikaa. Tavara ei voi olla liian kauan vastaanottoalueella ja hyllytystehtävän suoritusaika tulisi olla alle 15 minuuttia. Aktiivin täyttötehtävät tulisi taas suorittaa mahdollisimman nopeasti, jotta keräily ei viivästy. Aktiivin täyttötehtävä syntyy, kun keräilijä ei pysty poimimaan tarvitsemaansa tuotetta, joten odotusaikaa keräilijöille on muodostunut jo tehtävän syntyessä.

Tehtävänä oli selvittää, voisiko siirtotilauksen suoritukseen kuluvaa aikaa lyhentää jakamalla tehtävät niin, että kaikki trukit voivat tehdä kaikkia tehtäviä vain tietyn tehtävätyypin sijaan. Lisäksi tavoitteena oli lyhentää tyhjänä ajomatkaa. Lähtöaineistona minulla oli varaston kaikki siirtotilaustehtävät kuukauden ajalta Excel-tiedostona. Tiedostosta selviää tehtävän tyyppi, lähtö- ja kohdevarastopaikka, tehtävän luonti- ja kuittausaika sekä

suorittaja. Luontiaika on ajanhetki, jolloin tehtävä on ilmestynyt jonoon, ja kuittausaika se aika, jolloin trukkikuski on merkinnyt tehtävän valmiiksi. Aineistosta ei siis suoraan selviä, milloin tehtävää on alettu suorittaa, tai miten kauan se oli jonossa.

Tällä hetkellä trukit valitsevat uusia tehtäviä tehtäväjonosta vapaasti. Simuloin tilannetta kahdella uudella tavalla - ensin niin, että kaikki tällä hetkellä avoimet tehtävät jaetaan kaikille käytössä oleville trukeille muodostamalla tehtävistä reittejä. Reitillä voi olla yksi tai useampi tehtävä riippuen jonossa olevien tehtävien lukumäärästä. Ratkaistava tehtävä on tässä tapauksessa Vehicle Routing Problem with Pickup and Delivery. Kun tehtävät valmistuvat, lasketaan uudet reitit.

Tehtäviä tulee kuitenkin jatkuvasti lisää, ja uudet tehtävät olisi mahdollisesti hyvä huomioida jo ennen edellisten reittien valmistumista. Simuloin tilannetta myös antamalla trukille vain yhden tehtävän kerrallaan. Tässä vaihtoehdossa katsottiin vain, mikä tehtävä vapaana olevista on lähin. Tähän vaihtoehtoon lisäsin myös mahdollisuuden asettaa rajoitteita, esimerkiksi maksimiajan, jonka tehtävä saa olla jonossa.

Simulaatiota varten poimin aineistosta yhden vuorokauden, 19.5.2014, aikana suoritetut tehtävät. Laskin tehtävien keskimääräiset suoritusajat ja tehtävätyyppejä ja suorittajia tutkimalla selvitin, miten monta trukkia teki mitäkin tehtävätyyppiä kussakin vuorossa. Lisäksi arvioin vuorokauden aikana ajettua matkaa, tosin tätä on hankala selvittää luotettavasti, sillä esim. hyllytystehtävien kohdalla tarkkaa lähtöpaikkaa ei tiedetä. Lähtöpaikka voi olla joko missä tahansa vastaanottoalueella, tai siirretty varaston puolelle väliaikaisesti. Simulaatiossa olen olettanut, että hyllytystehtävässä lava on alkutilanteessa aina keskellä vastaanottoaluetta.

4.3 Sovelluksen toteutus

Toteutin simulaatiota varten kaksi ohjelmaa, toisen yksinkertaisella algoritmilla, joka etsii vain lähimmän vapaana olevan tehtävän, ja toisen, joka jakaa kaikki tällä hetkellä vapaana olevat tehtävät trukeille minimoiden reittien pituuden. Reittilaskennan tein Java-ohjelmana, sillä halusin hyödyntää Jsprit-Java-kirjastoa VRP:n ratkaisussa. Yksinkertaisemman algoritmin tein Pythonilla. Lisäksi käytin ArcMap-ja QGIS-paikkatieto-ohjelmia varaston pohjakartan muodostamiseen.

Simulaatiot on laskettu koneella, jolla on 8 GB keskusmuistia, Intel i5-2520M 2.50 GHz prosessori ja käyttöjärjestelmänä Windows 7.

Kummankin laskennan tuloksena sain trukkien kulkeman matkan, tehtäviin kuluneen ajan ja muita tietoja, jotka on tarkemmin eritelty luvussa 5. Vertailin tuloksia varastonhallintajärjestelmään kyseisen päivän ajalta tallentuneisiin tietoihin. Ajan suhteen tuloksia pystyy vertaamaan melko suoraan, matkaa voi vain arvioida.

4.3.1 Aineiston esikäsittely

Sain käyttööni mittakaavalla varustetun kuvan varastosta sekä Excel-tulosteen kuukauden aikana suoritetuista tehtävistä. Jotta simulaatiossa voidaan huomioida todelliset etäisyydet, tarvitsee aluksi laskea etäisyydet kaikista varastopaikoista kaikkiin. Tätä varten varastosta oli muodostettava graafi.

Varastopaikkojen koko ja käytävien leveys on aina sama, samoin hyllyn korkeus. Lisäksi varastopaikat on nimetty säännöllisesti niin, että ensimmäinen osa nimestä kertoo käytävän, toinen hyllyn ja kolmas tarkan paikan hyllyssä. Näiden tietojen perusteella pystyin muodostamaan graafin (eli digitoimaan kuvan) Python-ohjelmalla sen sijaan, että kartta olisi pitänyt piirtää yksi viiva kerrallaan paikkatieto-ohjelmassa.

Laskin OD (origin-destination) -matriisin varastopaikoille Esrin ArcGIS-ohjelmalla. Tässä tapauksessa yksi matriisin alkio sisältää kahden pisteen välisen lyhimmän etäisyyden. Voidaan olettaa, että trukinkuljettajat tuntevat varaston niin hyvin, että he käyttävät aina lyhyintä reittiä. Matriisi tallennetaan csv-muotoon reittienlaskentasimulaatiota varten. Yksi tehtävä kerrallaan -simulaatiota varten tallensin matriisin myös SQLite-tietokantaan. Tällöin yhdelle riville tulee lähdepaikka, kohdepaikka ja etäisyys.

Myös siirtotilausaineisto täytyy muokata käsittelyä varten. Aineistosta poistettiin ylimääräisiä kenttiä. Lisäksi, koska etäisyysmatriisi oli laskettu, pystyin lähde- ja kohdevarastopaikkojen koodien perusteella laskemaan siirroille arvioidut matkat ja ajat.

Matriisin koko aiheutti joitakin ongelmia laskennassa. Matriisi sisältää yli 12 miljoonaa riviä, joten se ei mahtunut Esrin shape-tiedostoon kokonaan (tiedoston yläraja on 2 GB). Tästä ei kuitenkaan tullut varoitusta ja havaitsin asian vasta myöhemmin, kun reittilaskennassa kaikkia varastopaikkoja ei löytynyt. Rajoituksen voi kiertää joko käyttämällä geodatabase- tai csv-formaattia.

4.3.2 Kokonaisten reittien laskenta

Aluksi simuloin tilannetta niin, että kaikki tällä hetkellä jonossa olevat tehtävät jaettiin trukeille. Jos tehtäviä on enemmän kuin trukkeja, yhdelle tai useammalle trukille muodostuu useamman tehtävän kattavia reittejä. Simulaatio tehtiin Java-ohjelmalla, joka käytti Jsprit-kirjastoa VRP:n ratkaisuun.

Ohjelma lukee aluksi tehtävät, trukkien alkusijainnit ja etäisyysmatriisin. Tämän jälkeen jaetaan tehtävät eri trukeille niin, että saadaan yksi ratkaisu, joka toteuttaa ongelman. Tämän jälkeen ruin-and-recreate -algoritmilla muodostetaan uusia ratkaisuja. Algoritmi rikkoo ensin ratkaisun poistamalla joltakin reitiltä tehtävän, jolloin ratkaisu ei enää toteuta ongelmaa. Sen jälkeen tehtävä sijoitetaan jollekin toiselle reitille ja tarkastellaan, onko uusi ratkaisu parempi. Tätä jatketaan määrätty lukumäärä iteraatiokierroksia, jonka jälkeen ratkaisu palautetaan.

Simulaatiossa algoritmi käyttää kahta ruin-tekniikkaa: random ruin, jossa poistetaan satunnainen määrä satunnaisia solmuja ratkaisusta, ja radial ruin, jossa poistetaan tietty solmu ja sen N-1 lähintä naapuria. Kumpaakin tekniikkaa käytetään 50 prosentin todennäköisyydellä.

$$\text{ratkaisun arvo} = T_{\text{pisin reitti}} + a \sum T_{\text{reitti}} + b \times \text{jakamattomat tehtävät} \quad (4.1)$$

Ohjelmaa ohjataan parempien ratkaisujen suuntaan objektiivifunktion (kaava 4.1) ja jokaisen insertion kohdalla tapahtuvan tarkastelun avulla. Kun uusi tehtävä sijoitetaan reitille, katsotaan, pitenikö kokonaismatka-aika (kaikkien trukkien yhteensä matkoihin käyttämä aika). Jos aika pitenee, huononnetaan ratkaisua. Lisäksi tulee minimoida objektiivifunktio. Objektiivifunktiossa a on pieni kerroin, b suuri kerroin. Objektiivifunktion arvo on sitä pienempi, mitä pienempi on pisimmän reitin matka-aika. Lisäksi vaikuttaa kokonaismatka-aika, tosin pienemmällä kertoimella. Funktion arvo nousee myös suuresti, jos ratkaisussa eivät ole mukana kaikki tehtävät eli jos ratkaisu ei ole validi.

Simulaatiokierros käynnistyy, kun yhden trukin reitti valmistuu. Käydään läpi tehtävälista ja muodostetaan niistä uudet reitit kaikille trukeille, jotka ovat tällä hetkellä vapaana. Jos tehtäviä ei ole, odotetaan, kunnes jonoon saapuu tehtävä. Algoritmi ei siis huomioi mitenkään trukkeja, jotka tällä hetkellä suorittavat tehtävää. Ne huomioimalla pystyttäisiin luultavasti parantamaan ratkaisua. Algoritmi voisi yksinkertaisimmillaan tarkastella, missä tällä hetkellä työskentelevät trukit lopettavat reittinsä, ja huomioida nekin uusia reittejä laskettaessa.

4.3.3 Lähimmän tehtävän huomiointi

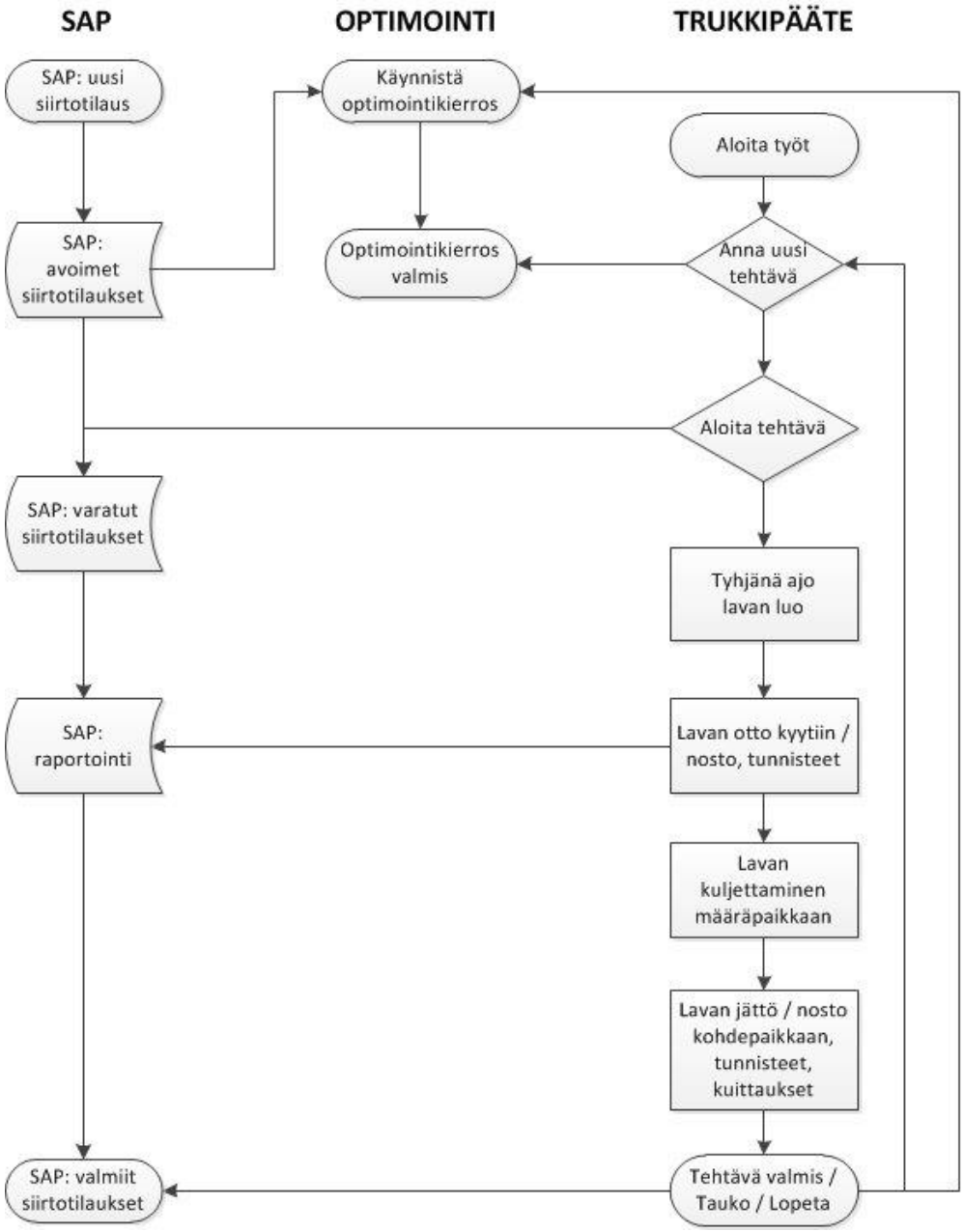
Koska uusia siirtotilauksia tulee jonoon jatkuvasti ja toisaalta jonossa on harvoin monia tehtäviä kerrallaan, simuloin tilannetta myös niin, että trukeille annetaan vain yksi tehtävä kerrallaan. Tämäntyyppinen ratkaisu voisi mahdollisesti toimia todellisuudessa paremmin, koska optimaalisin reitti voi mahdollisesti muuttua aina, kun jonoon tulee uusi tehtävä. Jos jonossa on useampi tehtävä, valitaan se, joka on lähinnä trukin tämänhetkistä sijaintia.

Tämäkään tekniikka ei varsinaisesti huomioi ongelman dynaamisuutta, sillä ratkaisua ei päivitetä uuden tehtävän saapuessa jonoon. Laskenta huomioi aina vain laskentahetkellä vapaana olevat tehtävät ja trukille lasketaan uusi tehtävä vasta, kun edellinen valmistuu. Yksinkertaista ratkaisutapaa voidaan perustella sillä, että jonoon saapuu uusia tehtäviä verrattain harvoin, hieman alle 2 tehtävää minuutissa.

Simulaatiokierros käynnistyy, kun jokin tehtävä valmistuu. Käydään läpi tehtävälista ja annetaan trukille uusi tehtävä mahdollisimman läheltä. Trukin oletetaan olevan edellisen tehtävän kohdepaikassa. Jos yhtään tehtävää ei ole vapaana, odotetaan, kunnes jonoon ilmestyy tehtävä. Simulaatiossa tämä tarkoittaa, että kelloa siirretään siihen ajanhetkeeen, jolloin seuraava tehtävä ilmestyi jonoon. Tehtäviä jaetaan FIFO-periaatteella, eli kauimmin odottanut trukki saa tehtävän ensiksi.

Tieto varastopaikkojen etäisyyksistä toisiinsa nähden tallennetaan SQLite-tietokantaan. Taulu sisältää tiedon lähtöpisteestä, kohdepisteestä ja niiden välisestä etäisyydestä. Lähtö- ja kohdepistekentät on indeksoitu, joten haut ovat nopeita. Lisäksi ratkaisu soveltuu tähän kohteeseen, koska varastossa on tehtäviä yleensä jonossa vain muutama. Haut eivät siis hidasta laskentaa liiaksi.

Ratkaisua kehitettäessä pidettiin mielessä myös, miten ratkaisun voisi mahdollisesti ottaa käyttöön varastossa. Vaadittavista muutoksista on kerrottu tarkemmin luvussa 6.



Kuva 4.1: Optimointiohjelman sijoittuminen varaston toimintaan. Simulaatiovaiheessa avoimia tilauksia ei haeta SAP-järjestelmästä, vaan ne luetaan tiedostosta.

Luku 5

Tulokset

Simulaation perusteella arvioitiin, miten reittien optimointi ja töiden ohjaus uudella tavalla vaikutti tehtävien suoritusaikaan. Arviointi tehtiin vertailemalla varastonhallintajärjestelmästä saatuja tietoja tehtävien todellisista suoritusajoista simulaatio-ohjelman laskemiin aikohin. Lisäksi verrattiin trukkien arvioitua alkuperäistä ajomatkaa simulaation laskemaan matkaan. Tässä luvussa kerron, millaisia tuloksia trukkitöiden reittioptimointi tuotti. Luvussa 6 on kerrottu tarkemmin jatkokehitysajatuksista, joita tulokset herättivät.

5.1 Tulosten arviointi

Optimoinnin tehokkuutta voi arvioida tutkimalla ajomatkojen ja tehtävien suoritusaikojen muutosta. Lisäksi on tarkasteltava simulaation suoritusnopeutta, erityisesti lähimmän tehtävän laskentaan kuluvaa aikaa, jotta voidaan arvioida, miten toteuttamiskelpoinen tämäntyyppinen ohjaus olisi varastossa.

Tehtävän suoritusajalla tarkoitetaan tehtävän järjestelmään luonti-ja kuittausajan välistä erotusta. Sekä simulaatiossa että alkutilanteessa suoritusaika sisältää myös ajan, jonka tehtävä odotti suoritukseen pääsyä. Vertailtaessa alkuperäisiä ja simulaation tuottamia suoritusaikoja hyllytystehtäville on huomioitava, että alkutilanteessa trukit ovat mahdollisesti siirtäneet lavat ensin väliaikaiseen varastoon ja sieltä jonkun ajan kuluttua varsinaiselle hyllypaikalle. Alkuperäisessä suoritusajassa ei kuitenkaan näy väliaikaiseen varastointiin kulunutta aikaa, vaan tehtävä on kuitattu tehdyksi vasta, kun lava on siirretty lopulliselle paikalleen. Tämän takia osa alkuperäisistä suori-

tusajoista on hyvin pitkiä. Simulaatiossa ei ole mahdollisuutta väliaikaiseen varastointiin, vaan kaikki siirrot on tehty suoraan tehtävän määrättyyn kohdepaikkaan. Tämän takia suoritusaikoja ei voi täysin verrata keskenään.

Ajomatkojen lyhentymistä voidaan tutkia karkealla tarkkuudella lähtöaineiston perusteella, jos oletetaan, että kuljettaja on vertailutilanteessa ajanut aina suorinta reittiä ja siirtynyt edellisen tehtävän loppupisteestä seuraavan tehtävän alkupisteeseen. Näin pystytään arvioimaan alkuperäinen ajomatka. Tehtävien suoritusnopeus lähtötilanteessa saadaan aineistosta löytyvistä tehtävän luonti- ja kuittausajoista. Tähän tosin sisältyy myös tehtävän jonossaoloaika, sillä varastonhallintajärjestelmään ei tallennu, milloin tehtävä otettiin suoritukseen.

Lähtöaineistosta arvioidun ajomatkan tarkkuutta on vaikea määrittää. Toisaalta ajomatka voi olla pidempi kuin lähtöaineistosta näkyy, sillä hyllytystehtävissä trukit siirtävät lavoja usein vastaanottoalueelta ensin väliaikaiselle varastoalueelle, josta ne siirretään myöhemmin eteenpäin varsinaisille hyllypaikoille. Järjestelmään tehtävä kuitataan valmiiksi, kun lava siirretään varsinaiselle paikalleen, mutta mihinkään ei tallennu, siirrettiinkö se väliaikaiselta paikalta vai vastaanottoalueelta. Toisaalta jos esimerkiksi käsiteltävän ajanjakson aluksi hyllytettävät lavat on todellisuudessa siirretty väliaikaiselta alueelta hyllypaikoille eikä vastaanottoalueelta saakka, kyseisen ajanjakson ajomatka voi olla lyhyempi kuin arvioitu. Väliaikaisen varastoinnin aiheuttamaa eroa on vaikea arvioida, sillä ei ole tiedossa, miten suuri osa lavoista viedään ensin väliaikaiseen varastoon.

Matkan arvioinnin tarkkuuteen vaikuttaa lisäksi laaja vastaanottoalue. Tehtävän tiedoista ei näy, missä kohtaa vastaanottoaluetta lava sijaitsee. Simuloitaessa tehtäviä on oletettu, että lavat sijaitsevat keskellä vastaanottoaluetta. Lava voi siis enimmillään sijaita noin 70 metriä y-suunnassa sen oletetusta sijainnista. Tämä ei vaikuta optimoinnin tehokkuuden arviointiin, sillä lavan sijainti on sama sekä lähtöaineistosta laaditussa arviossa että simulaatiossa. Alkuperäisen ajomatkan arvio voi kuitenkin olla liian suuri tai pieni riippuen siitä, missä lavat todellisuudessa ovat sijainneet. Vaikutusta on vaikea arvioida tarkkaan, sillä lavan sijainti vaikuttaa siihen, mistä pisteestä sinne on liikuttu.

Todellisten reittitietojen puute vaikeuttaa optimoinnin arviointia. Jos todelliset ajomatkat ovat lyhyempiä kuin arvioidut, optimointialgoritmien tehokkuus arvioidaan liian suureksi. Toisaalta jos arvio on liian lyhyt, reittioptimoinnin hyöty ei näy kokonaisuudessaan. Voidaan kuitenkin olettaa, että arvio ajomatkoista alkutilanteessa ei ole liian suuri. Arvio sisältää vain tehtävien sisäiset ja niiden väliset siirtymät, ja on todennäköistä, että trukeil-

la on myös muita siirtymiä, esimerkiksi akkujen latauspisteelle, jotka jäävät arviosta pois. Näin ollen vertaamalla arviota ja simulaation laskemaa ajomatkaa voidaan arvioida, onko tutkimuksessa toteutetusta trukkitöiden ohjaustavasta hyötyä.

5.2 Yleisiä havaintoja

Tiukkojen aikarajojen asettaminen tehtäville yksi tehtävä kerrallaan optimoitaessa ei vaikuttanut kannattavan. Jos tehtäviä saapui tietyssä ajassa niin paljon, että niitä ehti kerääntyä jonoon, yläraja tehtävien jonotusajalle tuli nopeasti vastaan, ja ne siirtyivät jonon kärkeen vanhimmasta alkaen. Jono purkautui nopeammin, jos etäisyys oli aina ensiksi huomioitava asia.

Lisäksi toisen tehtävätyypin priorisointi johtaa siihen, että toisen tehtävätyypin keskimääräinen suoritusaika pitenee. Jos hyllytystehtävien aikaraja asetetaan hyvin pieneksi, kaikki trukit siirtyvät tekemään hyllytystä, jos tehtäviä on useampi jonossa. Aktiivin suoritusaika vastaavasti pitenee. Nykyisessä mallissa paras tulos saadaankin, jos rajoituksia tehtävien suoritusajalle ei aseteta. Silloin tehtävät valikoidaan etäisyyden mukaan, jos jonossa on monta, ja muuten poimitaan seuraava. Matka-ajan lyhentyessä myös tehtävien suoritusaika vähenee.

Osa näistä havainnoista voi selittyä mallin yksinkertaisuudella. Laskennassa huomioidaan nyt vain trukit, jotka odottavat tehtävää. Tulos voisi olla hyvinkin erilainen, jos malli esimerkiksi tutkisi, tuleeko joku trukki pian valmistumaan avoimen tehtävän lähellä. Avoin tehtävä voitaisiin säästää sille sen sijaan, että tehtävä määrätään odottavalle trukille. Tulevaisuuden huomioimista kannattaisi ehdottomasti pohtia mallin jatkokehityksessä. Lisäksi näin voitaisiin välttää myös ruuhkaa käytävillä, kun odottavaa trukkia ei lähetettäisi alueelle, missä on jo toinen trukki. Tehtävien jakamisessa heti on kuitenkin etunsa - silloin minimoidaan trukkien odotusaika. Trukinkuljettajalle ei välttämättä olisi mielekästä odottaa useita minuutteja kokonaisuuden kannalta optimaalisinta tehtävää.

Simulaatio-ohjelman rakennusvaiheessa ja tuloksia havainnoidessa kävi myös selväksi, että nykyistä toimintatapaa on melko vaikea optimoida enempää. Trukkitöillä on aina ennalta määritelty lähde- ja kohdepaikka, joten tilausten suoritusaikaa ja tyhjänä ajomatkaa voi yrittää lyhentää vain tilausten suoritusjärjestyksen kautta. Suoritusjärjestystäkään ei aina ole mahdollista valita, sillä tehtäviä tulee yleensä tasaiseen tahtiin ja niitä ei ehdi kertyä paljon jonoon. Trukki saa yleensä jonosta ainoan vapaana olevan tehtävän,

joka voi olla missä päin tahansa. Ajomatkaan on tällöin vaikea vaikuttaa, ellei tehtävää odottavia trukkeja esimerkiksi siirrettäisi etukäteen paikkoihin, johon seuraavia tehtäviä ennustetaan tulevan.

Koska optimointivaraa on usein vähän, voisi olla jatkossa mielenkiintoista tutkia reittioptimoinnin lisäksi, vähentääkö sovellus virheitä trukkitöissä. Toteutetun kaltaisella sovelluksella olisi yksinkertainen käyttöliittymä, jossa näkyisi vain suoritettava tehtävä, mahdollisuus kuitata tehtävä päättyneeksi ja pyytää seuraava tehtävä tai siirtyä tauolle. Varastonhallintajärjestelmään ei tarvitsisi (tai voisi) syöttää käsin tietoja, jolloin ainakaan näppäilyvirheille ei olisi mahdollisuutta.

Lavan sijoittamista väärään paikkaan ei kuitenkaan voisi tällä keinolla estää: kun tehtävä kuitataan tehdyksi, oletetaan aina, että lava on tehtävän kohdepaikassa. Paikannuksella lavan todellisen sijainnin voisi varmistaa, ja hälyttää, jos tapahtuu virhe. Hyödyntämällä lisättyä todellisuutta voitaisiin oikea sijainti näyttää selkeästi. SAP on kehittänyt tällaisia sovelluksia keräilyyn [9].

5.3 Hyllytys

Kun tehtäviä ei erikseen priorisoitu tai rajoitettu, yksi tehtävä kerrallaan optimoitaessa hyllytystehtävissä suoritusaika parani eniten. Keskimääräinen luonti- ja kuittausajan erotus oli 15 minuuttia, kun se ennen oli 1 h 37 min (taulukko 5.2). Asteriski trukkien lukumäärän perässä kuvaa sitä, että simulaatiossa kaikki trukit tekivät kaikkia tehtäviä.

Suurta suoritusajan muutosta selittää se, että mallissa hyllytystehtävät pakotettiin tehtäväksi niin pian kuin mahdollista. Todellisuudessa hyllytystehtävillä on vähemmän tekijöitä kuin aktiivin täyttötehtävillä, joten tehtäviä kertyy jonoon ja lavat saattavat odottaa paikalleen vientiä kauankin. Lisäksi simulaatiossa tehtävät suoritettiin samantien kokonaisuudessaan, kun taas todellisuudessa lavat siirretään usein ensin väliaikaiseen varastoon. Tehtävä kuitataan kuitenkin järjestelmään tehdyksi vasta, kun lava siirretään lopulliselle paikalleen. Tästä syystä alkuperäiset suoritusajat ovat usein hyvin suuria, eivätkä täysin verrattavassa simulaation tuottamiin aikoihin.

Tarkasteltavassa aineistossa hyllytystehtäviä oli 1207 kappaletta.

Vertailtava kohde	**Alkuperäinen**	**Optimoitu**	**Yksikkö**
Tehtäviä yhteensä	2637	2637	kpl
Trukkeja ajossa klo 22-06	5	5	kpl
Trukkeja ajossa klo 06-14	9	9	kpl
Trukkeja ajossa klo 14-22	9	9	kpl
Tehtävän suoritusaika keskimäärin	0:47:18	0:12:48	hh:min:ss
Lyhimmän tehtävän kesto	0:00:02	0:00:38	hh:min:ss
Pisimmän tehtävän kesto	10:14:00	1:07:45	hh:min:ss
Tehtävä keskimäärin jonossa	(ei tiedossa)	0:09:48	hh:min:ss
Tehtävien kokonaismatka	719	667	km
Tehtävien jonoaika yhteensä	(ei tiedossa)	432	h
Tehtävien suoritusaika yhteensä	2078:47:28	562:26:50	hh:min:ss

Taulukko 5.1: Kaikki tehtävät. Tehtävän kestolla tarkoitetaan tehtävän luonti- ja suoritusajankohdan välistä erotusta.

Vertailtava kohde	**Alkuperäinen**	**Optimoitu**	**Yksikkö**
Tehtäviä yhteensä	1207	1207	kpl
Trukkeja ajossa klo 22-06	2	5*	kpl
Trukkeja ajossa klo 06-14	3	9*	kpl
Trukkeja ajossa klo 14-22	3	9*	kpl
Tehtävän suoritusaika keskimäärin	1:37:19	0:15:18	hh:min:ss
Lyhimmän tehtävän kesto	0:03:03	0:01:33	hh:min:ss
Pisimmän tehtävän kesto	10:14:00	0:48:03	hh:min:ss
Suoritettu 10 min kuluessa	2	38	prosenttia
Suoritettu 15 min kuluessa	3	56	prosenttia
Suoritettu 20 min kuluessa	5	72	prosenttia
Suoritettu 25 min kuluessa	7	81	prosenttia
Suoritettu 30 min kuluessa	10	89	prosenttia
Suoritettu 60 min kuluessa	42	100	prosenttia
Tehtävien suoritusaika yhteensä	1957:31:42	307:44:20	hh:min:ss
Tehtävien työaika (ilman jonoaikaa) yhteensä	(ei tiedossa)	62:19:49	hh:min:ss

Taulukko 5.2: Hyllytystehtävät. Tehtävän kestolla tarkoitetaan tehtävän luonti- ja suoritusajankohdan välistä erotusta.

5.4 Aktiivin täyttö

Aktiivin täyttötehtävät tehtiin simulaatiossa hitaammin kuin todellisuudessa on tehty. Tehtävien keskimääräinen luonti- ja kuittausajan erotus oli simulaatiossa 10 minuuttia, kun se vertailutilanteessa oli 5 minuuttia (taulukko 5.3). Asteriski trukkien lukumäärän perässä kuvaa sitä, että simulaatiossa kaikki trukit tekivät kaikkia tehtäviä.

Yksi selitys suoritusajan kasvulle on tehtävien jakautuminen tasan kaikille tekijöille. Todellisuudessa aktiivin täyttötehtäville on enemmän tekijöitä kuin hyllytystehtäville. Päivävuorossa aktiivin täytöllä on 6 työntekijää ja hyllytyksillä 3.

Jos molempien tehtävien aikaa halutaan lyhentää, tulisi trukkien määrän yhtä aikaa tällä optimointitavalla olla noin 10-11. Tällöin hyllytystehtävän keskiarvoksi saadaan 10 minuuttia ja aktiivin keskiarvoksi 6 minuuttia. Jos halutaan priorisoida jonkun tehtävätyypin suoritusaikaa tai tietyn aikarajan alapuolella pysyminen on tärkeää, voitaisiin optimoinnissa mahdollisesti säilyttää trukin ensisijainen tehtävätyyppi. Toisen tyypin tehtäviä voitaisiin antaa, jos ne ovat lähellä nykyistä sijaintia tai jos omia tehtäviä ei ole tarjolla.

Tarkasteltavassa aineistossa aktiivin täyttötehtäviä oli 1430 kappaletta.

5.5 Yksittäiset tehtävät verrattuna reittien tarkasteluun

Simulaatiota suoritettiin kahdella tapaa, muodostamalla reitit kaikista ajanhetkellä vapaana olevista tehtävistä, ja antamalla trukille vain yksi tehtävä kerrallaan. Tehtäviä tulee kuitenkin sen verran tasaisesti, että jono on suurimman osan ajasta tyhjä. Tällöin ratkaistavana oleva ongelma on käytännössä sama molemmissa tapauksissa. Reittien laskenta on kuitenkin hieman hitaampaa, koska (ilman muutoksia Jsprit-kirjastoon) suorituksen aluksi on muodostettama VRP:n ratkaisufunktion tarvitsema matriisi etäisyyksistä varastossa. Tehtävien antamista yksi kerrallaan voidaan siis pitää tehokkaampana tapana.

Lisäksi uusia tehtäviä voi tulla jonoon milloin tahansa, jolloin reitit pitäisi muokata ennen niiden valmistumista, tai ne eivät välttämättä ole tehokkaimmat mahdolliset. Simulaatiossa reittejä ei muokattu niiden suorituksen aikana, vaan ne laskettiin uudestaan vasta, kun tehtävät valmistuivat. Laskemalla ne aina, kun jonoon saapuu tehtävä, voitaisiin matkaa ehkä entisestään

Vertailtava kohde	**Alkuperäinen**	**Optimoitu**	**Yksikkö**
Tehtäviä yhteensä	1430	1430	kpl
Trukkeja ajossa klo 22-06	3	5*	kpl
Trukkeja ajossa klo 06-14	6	9*	kpl
Trukkeja ajossa klo 14-22	6	9*	kpl
Tehtävän suoritusaika keskimäärin	0:05:05	0:10:42	hh:min:ss
Lyhimmän tehtävän kesto	0:00:02	0:00:39	hh:min:ss
Pisimmän tehtävän kesto	0:15:47	1:07:45	hh:min:ss
Suoritettu 2 min kuluessa	15	4	prosenttia
Suoritettu 3 min kuluessa	30	17	prosenttia
Suoritettu 4 min kuluessa	47	37	prosenttia
Suoritettu 5 min kuluessa	60	47	prosenttia
Suoritettu 6 min kuluessa	66	51	prosenttia
Suoritettu 7 min kuluessa	76	55	prosenttia
Tehtävien suoritusaika yhteensä	121:15:46	254:42:27	hh:min:ss
Tehtävien työaika (ilman jonoaikaa) yhteensä	(ei tiedossa)	67:26:41	hh:min:ss

Taulukko 5.3: Aktiivin täydennystehtävät. Tehtävän kestolla tarkoitetaan tehtävän luonti- ja suoritusajankohdan välistä erotusta.

lyhentää. Tämä kuitenkin vaatisi ohjelmaan muutoksia, koska nyt tehokkuuden takia graafista poimittiin aina vain tarvittava osa laskentaa varten. Jos kohdepisteiden joukko muuttuu laskennan aikana, graafista joudutaan uudestaan etsimään tarvittava osa.

Tilanne	**Ajettu matka**
Alkuperäinen (arvioitu)	718,8 km
Tehtävistä laskettu reittejä	636,766 km
Tehtäviä annettu yksi kerrallaan	667 km

Taulukko 5.4: Kaikkien trukkien ajama matka yhteensä vuorokauden aikana

Tarkastelemalla kokonaisia reittejä saatiin kuitenkin ajomatkaa lyhennettyä hieman enemmän kuin antamalla yksi tehtävä kerrallaan. Ajomatkan arvioitiin aluksi olevan 718,8 kilometriä yhden vuorokauden aikana. Optimoiduilla reiteillä matkaksi saatiin keskimäärin 636,766 km (taulukko 5.4). Matka lyheni siis 11 prosenttia. Kun tehtäviä annettiin yksi kerrallaan, ajomatka lyheni 7 prosenttia.

5.6 Ohjelman suoritusaika

Yksi simulaation tavoitteista oli selvittää, voisiko vastaavanlaista ohjelmaa käyttää trukkityön ohjauksessa käytännössä. Kun trukeille laskettiin kokonaisia reittejä, yksi laskentakierros kesti muutamista sekunnista kymmeniin sekunteihin iteraatiokierrosten lukumäärästä riippuen. Tämä voisi käytännössä olla turhauttavan pitkä aika käyttäjälle odottaa seuraavaa tehtävää, lisäksi mukaan tulisi vielä viive tiedonsiirrosta asiakasohjelman ja varastonhallintajärjestelmän välillä. Ohjelmaa pitäisikin muokata esimerkiksi laskemaan seuraava reitti, kun edellisen suoritus on vielä meneillään. Yksi tehtävä kerrallaan laskettaessa seuraava tehtävä saadaan alle sekunnissa.

Taulukossa 5.5 on kuvattu iteraatiokierrosten määrän vaikutusta ohjelman suoritusaikaan ja saadun ratkaisun hyvyyteen, kun lasketaan kokonaisia reittejä. Taulukon otoksessa laskettiin reitit 45 tehtävälle. Laskenta aloitettiin ajanhetkellä 00:00 ja mukaan otettiin tehtävät, jotka olivat saapuneet sitä ennen varastonhallintajärjestelmään. Näistä laskettiin reitit kuudelle trukille. Kaikki trukit lähtivät liikkeelle keskeltä vastaanottoaluetta. Pisin reitti -sarakkeessa on pisimmän reitin suoritusaika, eli aika, jossa kaikki tehtävät saatiin tehtyä.

Iteraatiokierroksia	Pisin reitti	Suoritusaika
5000	27,804 min	30 sekuntia
2000	26,097 min	16 sekuntia
1500	27,708 min	10 sekuntia
1000	28,161 min	15 sekuntia
500	27,394 min	3 sekuntia
200	26,593 min	4 sekuntia
100	29,399 min	2 sekuntia
10	28,138 min	4 sekuntia
1	29,071 min	3 sekuntia

Taulukko 5.5: Iteraatiokierroksien määrän vaikutus kokonaissuoritusaikaan ja tulokseen 23.30 - 00.00

Iteraatiokierrosten määrä ei suoraan näytä vaikuttavan lopputulokseen, mutta luonnollisesti suoritusaikaan. Selitys tuloksen pysymiselle samanlaisena on todennäköisesti se, että suuri osa tehtävistä oli hyllytystehtäviä, joten trukkien pitää aina liikkua vastaanottoalueelle takaisin. Reitin pituus on siis aina lähes sama. Vain muutamassa kohdassa oli mahdollista suorittaa samalla lähellä sijaitseva aktiivin täyttötehtävä. Iteraatiokierrosten määrän vaikutusten tutkimiseksi tarkemmin pitäisi tehdä enemmän mittauksia erilaisilla tehtäväjoukoilla. Suoritusaika ei aina suoraan lyhene kierroksia vähennettäessä. Tähän voi olla syynä esimerkiksi, että joillakin kerroilla ohjelma sattumalta kokeili aiemmin valideja ratkaisuja, ja toisilla kerroilla joutui kokeilemaan useampia insertioita löytääkseen sopivan ratkaisun. Näidenkin erojen poistamiseksi pitäisi tehdä enemmän mittauksia erilaisilla tehtäväjoukoilla.

Laskenta-ajat ovat nykyisellään pitkiä reitinlaskentaohjelmassa yhdellekin kierrokselle. Ohjelmassa olisikin paljon optimoitavaa. Suurin osa suoritusajasta kului matriisin lukemiseen. VRP:n ratkaisufunktio vaatii, että matriisi etäisyyksistä muodostetaan kokonaisuudessaan ennen laskennan alkua, ja sitä varten ohjelman pitää lukea sisään aiemmin laskettu matriisi. Voidaan myös poimia oleelliset rivit matriisista ennen laskennan alkua, jotta sovelluksen riittää lukea pienempi tiedosto. Esikäsittelyssä on silti luettava koko tiedosto. Ohjelman rakennetta olisi hyvä muuttaa niin, että matriisia ei tarvitse lukea kokonaan, vaan tarvittavat etäisyydet voidaan kysellä esimerkiksi tietokannasta tai hajautustaulusta.

Kun tehtäviä annettiin yksi kerrallaan, riittää yhden kierroksen aikana selvittää, mikä avoinna olevista tehtävistä on lähinnä trukin tämänhetkistä sijaintia. Koska etäisyydet pisteiden välillä on tallennettu tietokantaan ja indeksoitu lähtö-ja kohdepaikan mukaan, haku on nopeaa. Yksi kierros (eli seu-

raavan tehtävän määrittäminen trukille) kestää korkeintaan sekunteja. Tämä aika ei tosin ole suoraan verrattavissa kokonaisten reittien laskentaan, koska yksi kerrallaan tehtävät jaettaessa ratkaisu palautetaan suoraan, ilman iterointikierroksia. Reitinlaskenta kannattaisikin toteuttaa käytännössä niin, että tehtävää kysyttäessä palautetaan tämänhetkinen tulos ja sen jälkeen jatketaan iterointia.

45 tehtävän reittien laskentaan meni yksi tehtävä kerrallaan annettaessa 0,69 sekuntia ja tehtävät saatiin suoritettua 32,446 minuutissa. Tässä tapauksessa optimointistrategia, joka muodosti kokonaisia reittejä, pääsi siis selvästi parempaan ratkaisuun. Huomattavasti lyhyemmän suoritusajan takia yksinkertaisempi, yhden tehtävän kerrallaan huomioiva algoritmi olisi kuitenkin todennäköisempi vaihtoehto käytännössä hyödynnettäväksi.

Luku 6

Jatkokehitys

Työtä varten toteutettiin kaksi sovellusta: Python-sovellus, joka palauttaa aina lähimmän tehtävän trukille ja samalla kerää tilastoja muun muassa tehtävien suoritusajoista ja ajomatkoista ajomatkoista, sekä Java-sovellus, joka laskee kokonaisia reittejä ja kerää samoja suoritustilastoja. Keskityn tässä luvussa yksi tehtävä kerrallaan optimoivan sovelluksen jatkokehitysehdotuksiin, sillä se toimi huomattavasti nopeammin ja olisi luultavasti ensisijainen valinta toteutettavaksi varastossa. Kokonaisia reittejä laskemalla pystytään kuitenkin lyhentämään ajomatkaa enemmän. Kokonaisia reittejä laskevan sovelluksen jatkokehitys esimerkiksi päivittämään reittejä aina, kun uusi tehtävä tulee, voisi olla kaikkein tehokkain tapa edetä.

Yksi tehtävä kerrallaan optimoiva sovellus on nykyisellään simulaatio, joka poimii tehtäviä valmiilta listalta ja lähimmän tehtävän haku käynnistyy, kun simulaation sisäinen kello ylittää ajan, joka on jonkun trukin arvioitu valmistumisaika. Jotta sovellusta voisi hyödyntää varastossa, täytyisi sen pystyä reagoimaan todellisiin tapahtumiin, eli esimerkiksi uusien tehtävien saapumiseen jonoon ja trukilta tuleviin ilmoituksiin tehtävän valmistumisesta. Tässä luvussa käsittelen, mitä ehdotetun sovelluksen käyttöönotto todellisuudessa vaatisi; mitä muutoksia tarvittaisiin olemassaoleviin järjestelmiin ja optimointiohjelmaan.

Lisäksi käsittelen, miten sovellusta voisi kehittää edelleen käyttämällä paikannushavaintoja, ja miten optimointialgoritmia voisi kehittää edelleen.

6.1 Odotusstrategiat, sijainnin ja dynaamisuuden huomioiminen

Nykyisellään sovellus antaa yksi tehtävä kerrallaan optimoitaessa trukille uuden tehtävän saman tien, kun edellinen loppuu. Jos tehtäviä ei ole tällä hetkellä vapaana, trukki saa ensimmäisen uuden tehtävän. Jos jonossa on useampi trukki, kauinten odottanut trukki saa ensin tehtävän. Tällöin ei huomioida trukkien sijaintia. Tehtävien suoritusaika voisi lyhentyä, jos useamman trukin ollessa vapaana tarkistettaisiin, mikä on lähimpänä vapaata tehtävää.

Sovellus voisi myös odottaa hetken ennen uuden tehtävän jakamista, jos kaikki vapaat tehtävät ovat kaukana trukin tämänhetkisestä sijainnista. Dynaamisen ongelman tapauksessa on osoitettu, että ajoneuvojen voi olla joskus hyödyllistä odottaa tietyssä pisteessä hetken aikaa kokonaisajomatkan kannalta mahdollisimman optimaalista tehtävää [2]. Tilauksia voidaan myös puskuroida ennen niiden jakamista reiteille. Tätäkin strategiaa olisi mielenkiintoista kokeilla myös optimointiohjelmassa. Odottamis- tai puskurointistrategioiden hyöty riippuu tosin siitä, mitä suuretta ratkaisulla pyritään optimoimaan. Jos ensisijaisesti tavoitellaan tehtävien suoritusajan minimointia, ylimääräistä odotusaikaa ei ehkä hyväksytä ratkaisuun.

Trukkien odotusaikaa voisi myös yrittää hyödyntää jotenkin. Aiempien tehtävien sijaintien perusteella voitaisiin laskea, mihin seuraavia tehtäviä todennäköisesti tulee, ja liikkua paikkaan, josta voidaan helposti tavoittaa mahdollisimman moni uuden tehtävän lähtöpiste, esimerkiksi vastaanottoalueelle. Jos usealla trukilla ei ole tehtäviä, ne voisivat sijoittua niin, että mahdollisimman monen todennäköisen alkupisteen lähellä on trukki. On havaittu, että reititysalgoritmit, jotka varautuvat tulevaisuuteen näin, saavat hieman parempia tuloksia aikaan kuin lyhytnäköiset menetelmät [20].

Sovellus ei nykyisessä muodossaan myöskään huomioi kovin hyvin ongelman dynaamisuutta. Uusia trukkitöitä tulee järjestelmään koko ajan, mutta reititysongelma lasketaan aina sarjana staattisia ongelmia, ja laskenta käynnistyy aina vasta, kun yksi tehtävä tai reitti valmistuu. Tämä tarkoittaa sitä, että kun trukki pyytää optimointiohjelmalta uutta tehtävää, se palauttaa yhden tehtävän, joka voi olla toisella puolella varastoa, jos muuta ei ole vapaana. Jos ennen tehtävän suorituksen aloittamista järjestelmään saapuu uusi lähempänä oleva tehtävä, ratkaisua ei muuteta. Ohjelman rakennetta voisi olla hyvä muuttaa niin, että ohjelmalla on aina valmiina ratkaisu kaikkien trukkien seuraavaksi tehtäväksi niiden nykyisen tehtävän kohteen sijainnin perusteella. Kun uusi tehtävä tulee järjestelmään tai joku trukeista saa ny-

kyisen tehtävän valmiiksi ja aloittaa uuden, ohjelma päivittäisi ratkaisua. Jos järjestelmässä ei ole vapaita tehtäviä, ohjelman ehdottama sijainti voisi olla vastaanottoalueella. Vaihtoehtoisesti ehdotettu uusi sijainti voisi olla jossain pisteessä, jonne tilaushistorian analyysin perusteella on todennäköisesti tulossa tehtävä.

6.2 Sovelluksen muokkaus todelliseen ympäristöön

Sovelluksen arkkitehtuuria on muokattava, jos sitä halutaan hyödyntää trukkitöiden ohjauksessa varastossa. Ohjelman pitäisi kuunnella trukeilta tulevia pyyntöjä ja vastata mahdollisimman nopeasti.

Arkkitehtuuri voisi olla esimerkiksi kuvan 4.1 mukainen. Optimointiohjelma koostuu varastonhallintajärjestelmästä (kuvassa SAP), optimointimoduulista (palvelinsovelluksesta) ja asiakassovelluksesta. Varastonhallintajärjestelmästä luetaan avoimet tehtävät ja sinne päivitetään tehtävien tila. Optimointimoduuli lukee varastonhallintajärjestelmää, laskee trukeille seuraavat tehtävät avointen tehtävien perusteella ja lähettää trukeille seuraavan tehtävän, kun se saa trukilta tiedon tehtävän valmistumisesta. Asiakasohjelmassa kuitataan tehtävä aloitetuksi ja raportoidaan tieto varastonhallintajärjestelmään. Lisäksi raportoidaan lavan tiedot ja ajanhetki, milloin se nostetaan kyytiin, vienti kohdepaikkaan ja tehtävän lopetus.

Optimointi- ja asiakassovellus voidaan toteuttaa lähes millä tahansa tekniikalla. Helpointa on valita joku teknologia, jonka käyttö valitun varastonhallintajärjestelmän kanssa on yksinkertaista.

Asiakassovellus toimii trukkipäätteessä. Trukkipäätteeseen on usein kiinnitetty viivakoodinlukija, joka helpottaa lavan tietojen lukemista. Periaatteessa trukkisovellus voisi toimia hyvin myös älypuhelimessa tai tabletissa. Tällöin voitaisiin käyttää usb- tai bluetooth-viivakoodinlukijaa.

Mahdolliset virheet trukkitöissä tulee myös pystyä käsittelemään. Jos käytössä olisi jonkunlainen paikannus, voitaisiin lavaa noudettaessa ja viedessä tarkistaa, ollaanko suurin piirtein oikealla alueella.

WLAN-yhteyden tulisi toimia luotettavasti. Muuten on hankalaa huomioida uudet tehtävät dynaamisesti. Yhteyden katkeamisen varalta voitaisiin esimerkiksi ladata trukin käsipäätesovellukseen seuraava tehtävä muistiin, mutta silloin yhteyden on toimittava jos ratkaisua mahdollisesti halutaan päivittää, tai sama tehtävä voi mennä jollekin toiselle trukille.

Laskennan tulee olla nopeaa, ja vastaus seuraavan tehtävän sijainnista oli-

si hyvä saada sekunneissa tai lyhyemmässä ajassa. Nyt varsinainen laskenta sujuu noin 0,7 sekunnissa, mutta todellisuudessa tähän olisi lisättävä vielä tiedonsiirtoon kuluva aika. Edellisessä kappaleessa esitelty ehdotus voisi nopeuttaa ohjelman toimintaa. Ohjelmalla olisi siis aina tiedossa yksi validi ratkaisu, joka palautettaisiin, kun trukki kysyy tehtävää. Kun trukki vastaanottaa uuden tehtävän tai uusi tehtävä saapuu järjestelmään, lasketaan uusi ratkaisu.

6.3 Paikannusdatan hyödyntäminen

Tällä hetkellä sovelluksessa hyödynnetään varaston pohjakarttaa ja pyyntöihin liittyviä tietoja lavan lähtö-ja kohdepaikasta. Sovellusta varten on tietokantaan viety etukäteen kaikkien paikkojen sijainnit ja etäisyydet paikoista toisiin. Indeksoidusta kannasta hakeminen on nopeaa, lisäksi hakuja ei tule kovin paljon, sillä riittää joka tilanteessa hakea avoinna olevista tehtävistä lähin ja avoimia tehtäviä on yleensä vain muutama.

Tässä ”paikannustavassa”on kuitenkin muutamia selkeitä puutteita. Järjestelmä ei nyt pysty havaitsemaan trukin todellista reittiä, vaikka oletus nopeimmasta reitistä onkin perusteltu. Sen takia arviot ajomatkasta ovat epätarkkoja. Tarkat tiedot olisivat kuitenkin hyvin hyödyllisiä.

Lisäksi vastaanottoalueella olevien lavojen todellista sijaintia ei tiedetä tarkkasti. Lavoja voidaan myös siirtää vastaanottoalueelta väliaikaisiin sijainteihin, mutta ohjelma voi vain olettaa niiden olevan edelleen vastaanottoalueella.

Jotta ohjelma voisi hyödyntää tarkkaa paikkatietoa ja ajomatkoista kerättyä tietoa voisi luotettavasti käyttää, pitäisi toteuttaa myös jonkinlainen paikannusjärjestelmä. Paikannus voitaisiin toteuttaa esimerkiksi Bluetoothin avulla.

Bluetooth-paikannusta varten hyllyihin kiinnitetään majakoita (beacon), jotka lähettävät signaalia. Paikannus voidaan toteuttaa esimerkiksi laskemalla sijainti kolmen eri etäisyyshavainnon perusteella, tai voidaan vain määrittää, mikä on nykyistä sijaintia lähinnä oleva majakka.

Paikannuksen perusteella voitaisiin määrittää trukin sijainti ja reitti ja tarkistaa, onko lava menossa oikeaan suuntaan. Bluetooth-paikannus ei kuitenkaan vielä riittäisi esimerkiksi siihen, että lavan oikea sijoituspaikka voitaisiin luotettavasti tarkistaa. Tähän tarvittaisiin selvästi alle metrin tarkkuus. Esimerkiksi Microsoftin vuonna 2014 järjestämässä sisätilapaikannuskilpailussa

paras ryhmä pääsi 0,72 metrin keskivirheeseen, kaikilla muilla virhe oli yli 1,5 metriä [16].

Paikannuksen käyttöönotto vaatisi jonkun verran työtä varastossa. Majakat täytyy asentaa ja niiden sijainnit mitata, ellei varastosta ole jo valmiina sijaintitietoa. Tässä työssä muodostettiin digitaalinen kartta varastosta, joten fyysisten mittausten sijaan voitaisiin ennen asennusta merkitä kartan päälle majakoiden sijainnit ja tallentaa ne esimerkiksi JSON-formaatissa. Tällöin on luonnollisesti tärkeää pitää tarkkaa kirjaa siitä, mihin kohtaan mikäkin majakka asennetaan.

Paikannuksen tarkkuuden määrittäminen ja testaus voi myös viedä aikaa. Koska varastossa on täynnä metallisia hyllyjä, heijastukset voisivat mahdollisesti aiheuttaa ongelmia. Esimerkiksi museo Brooklynissa törmäsi tällaisiin ongelmiin, kun majakoita asennettiin eri taideteosten kohdalle [3]. Voimakkain signaali saattoi vaikuttaa tulevan jostain muualta kuin lähimmästä majakasta.

Muita vaihtoehtoja voisi olla esimerkiksi kuvapohjainen paikannus. Tällöin signaalin heijastumiset eivät häiritsisi. Liang et al. [15] rakensivat järjestelmän, joka vertaa puhelimella otettua valokuvaa etukäteen muodostettuun kuvatietokantaan. Menetelmällä päästiin alle metrin tarkkuuteen, lisäksi saadaan selville puhelimen suuntaus. Tämä mahdollistaisi lisätyn todellisuuden hyödyntämisen. Varastossa ongelmana on kuitenkin se, että varaston ulkonäkö muuttuu jatkuvasti, kun lavojen määrä vaihtelee. Lisäksi hyllyjen erottamiseksi toisistaan niihin luultavasti pitäisi kiinnittää jonkinlaisia merkkejä, muuten kaikki hyllyvälit ovat melko identtisiä. Lisäksi Liang et al. :n menetelmässä yhden kuvan paikannus vie 10-12 sekuntia kannettavalla tietokoneella, mikä olisi käyttäjälle pitkä aika odotella.

6.4 Virheiden vähentäminen optimoinnin sijaan

Kuten tuloksista havaittiin, siirtotilauksiin kuluvaa aikaa on vaikea enää vähentää ilman, että työntekijöitä lisättäisiin. Tehtäviä ei ole kerrallaan niin paljon, että niiden suoritusjärjestyksellä olisi kovin paljon vaihtoehtoja. Mielenkiintoista olisikin myös tutkia, mitä vaikutusta trukkitöiden virheiden vähentämisellä olisi töiden kokonaissuoritusaikaan.

Paikannuksella voitaisiin näyttää reittiohjeita tehtävän kohdepaikkaan ja tarkastaa lavan sijoittaminen oikeaan paikkaan. Vaikka trukkikuski tietäisikin

kaikki sijainnit ilman opastusta, voitaisiin ohjeilla välttää lavan sijoittaminen vahingossa väärään paikkaan. Lisäksi siinä vaiheessa, kun hidastetaan lavan hyllytystä varten, voitaisiin tarkastaa sijainti ja varmistaa, että lava ollaan sijoittamassa suurin piirtein oikein. Tässä vaiheessa voitaisiin myös aloittaa seuraavan tehtävän laskenta.

6.5 Simulointiohjelman hyödyntäminen jatkossa

Yksi työn tuloksista on työkalu varaston simulointiin kartan ja tilaushistoriadatan perusteella. Pienin muokkauksin näitä työkaluja voisi hyödyntää muissakin varastoissa trukkitöiden simulointiin ja uusien toimintatapojen testaamiseen.

Datankäsittely- ja reitinlaskentafunktiot hyödyntävät suoraan SAP-järjestelmästä saatavia tulosteita tilauksista. Pienin muutoksin toteutetuilla ohjelmilla voitaisiinkin simuloida minkä tahansa varaston siirtotilauksia, mikäli hyllyt ovat vakiokokoisia. Tarvittavia muutoksia olisivat hyllypaikkojen nimeämis-järjestelmä, trukkien keskimääräiset ajonopeudet ja eri hyllyjen korkeudet.

Luku 7

Yhteenveto

Tässä työssä tutkittiin siirtotilausten reittioptimointia varastossa. Tällä hetkellä työntekijät tekevät tiettyä tehtävätyyppiä. Varastoa simuloimalla tutkittiin optimointistrategiaa, jossa kaikki avoimet tehtävät sijoitettiin yhteen jonoon ja trukeille poimittiin tehtäviä niin, että tyhjänä kuljettu ajomatka olisi mahdollisimman lyhyt. Tehtävien jakoa kokeiltiin kahdella tapaa: muodostamalla reittejä vapaana olevista tehtävistä, ja jakamalla trukeille yksi tehtävä kerrallaan.

Havaittiin, että siirtotilaukset tehdään nyt jo melko tehokkaasti. Aktiivin täydennykset tehdään keskimäärin muutamassa minuutissa, eikä tähän aikaan saatu optimoinnilla parannusta. Hyllytystehtävien keskimääräinen suoritusaika taas lyheni selvästi, tosin nykyinen algoritmi suoritti hyllytystehtävät kerralla kokonaan. Todellisuudessa lavat siirretään hyllytyksessä usein väliaikaiseen varastoon, mutta tehtävä kuitataan suoritetuksi vasta, kun lava siirretään lopulliselle paikalleen. Tästä syystä hyllytystehtävien suoritusajat ovat pitkiä alkutilanteessa. Onkin vaikeaa sanoa, miten paljon suoritusajan lyhenemiseen vaikutti reittien optimointi ja miten paljon väliaikaisvarastoinnin poistuminen.

Kokonaisajomatka lyheni trukkitöitä optimoimalla. Ajomatka lyheni 11 prosenttia, jos vapaana olevista tehtävistä muodostettiin kokonaisia reittejä, ja 7 prosenttia, jos tehtäviä jaettiin yksi kerrallaan. Alkuperäinen ajomatka on arvio, mutta tutkimuksen perusteella vaikuttaa, että trukkitöiden ohjaus olisi hyödyllistä toteuttaa tutkitulla tavalla. Lisäksi ohjausjärjestelmä voisi kerätä seurantadataa tehtävien todellisesta suoritusajasta, tehtävien jonossaoloajasta ja trukkien odotusajasta. Näitä tietoja ei tällä hetkellä ole saatavilla, ja niiden perusteella ohjausta voitaisiin parantaa entisestään.

Trukkitöiden optimointi kokonaisia reittejä laskemalla lyhensi ajomatkaa eni-

ten. Optimointi yksi tehtävä kerrallaan laskemalla oli kuitenkin suoritusajaltaan nopeampaa ja lyhensi myös trukkien ajomatkaa verrattuna alkutilanteeseen. Lisäksi trukkitöitä tulee yleensä melko tasaisesti, ja jonoon ei yleensä kerry paljon tehtäviä. Tehtävien suoritusjärjestykselle ei siis ole montaa vaihtoehtoa. Sopivin optimointitapa trukkitöille voisikin olla tehtävien jako yksi kerrallaan.

Trukkitöiden ohjauksen jatkokehityksessä pitää huomioida paremmin ongelman dynaamisuus. Nyt seuraava tehtävä lasketaan vasta edellisen valmistuttua, eikä ratkaisua muuteta, kun uusia tehtäviä tulee jonoon. Ohjelmaa voitaisiin jatkokehittää esimerkiksi niin, että laskettuna on aina seuraava tehtävä jokaiselle trukille niiden nykyisen sijainnin tai tämänhetkisen tehtävän päättymispisteen perusteella. Tätä ratkaisua päivitettäisiin sitä mukaa, kun uusia tehtäviä tulee jonoon ja trukkien nykyiset tehtävät valmistuvat.

Työssä ei ollut käytettävissä trukkien todellisia reittiaineistoja, vaan arviot ajomatkasta tehtiin varastohallintajärjestelmään tallentuneiden tilausten perusteella. Vaikka tilanne onkin tyypillinen varastojen tutkimuksessa, olisi todellisia reittihavaintoja mielenkiintoista nähdä ja niiden perusteella voitaisiin tehdä tarkempia päätelmiä optimoinnin hyödyllisyydestä.

Todellisia reittihavaintoja voitaisiin saada paikantamalla trukkeja varastossa. Sisätilapaikannus voitaisiin toteuttaa esimerkiksi kiinnittämällä bluetooth-majakoita varastoon. Majakoiden sijainnit pystyttäisiin tallentamaan etukäteen paikan päällä tehtävien mittausten sijaan, koska optimointia varten muodostettiin kartta varastosta. Paikannuksen toimivuutta on kuitenkin vaikea ennustaa, sillä varasto on täynnä hyllyjä, jotka voivat aiheuttaa heijastuksia.

Trukkitöiden ohjausta ja optimointia on tutkittu vähemmän kuin keräilyä tai muita varastoprosesseja. Niihin kuitenkin kannattaa kiinnittää huomiota. Trukkitöiden tehokas suoritus edistää keräilyä, ja työn perusteella trukkitöihin kuluvaa ajomatkaa pystytään optimoimalla lyhentämään. Trukkitöissä itsessään on kuitenkin vain vähän muokattavaa, lähinnä voidaan valita tehtävien suoritusjärjestys. Trukkitöiden optimointia kannattaakin tarkastella laaja-alaisesti. Trukit joutuvat esimerkiksi silloin tällöin odottamaan seuraavaa tehtävää. Menneitä tehtäviä tarkastelemalla voitaisiin laskea, minne todennäköisesti voisi tulla seuraava pyyntö, ja pyytää trukkia siirtymään sinne.

Trukkityöt ovat osa varastointiprosessia ja niiden suoritusnopeuteen vaikuttavat monet muutkin asiat kuin ajoreitit. Reititysalgoritmeja on kuitenkin mahdollista testata simuloimalla koskematta varastoon, ja niistä on hyvä aloittaa.

Kirjallisuutta

[1] Bartholdi, J. J., and Hackman, S. T. *Warehouse and Distribution Science*. 2014. Release 0.96.

[2] Berbeglia, G., Cordeau, J.-F., and Laporte, G. Dynamic pickup and delivery problems. *European journal of operational research 202*, 1 (2010), 8–15.

[3] Bernstein, S. The realities of installing iBeacon to scale. Verkkosivu, 4. helmikuuta 2015. `https://www.brooklynmuseum.org/community/blogosphere/2015/02/04/the-realities-of-installing-ibeacon-to-scale/`. Haettu 17.7.2015.

[4] Chen, D., Tsai, S., Kim, K.-H., Hsu, C.-H., Singh, J. P., and Girod, B. Low-cost asset tracking using location-aware camera phones. In *SPIE Optical Engineering+ Applications* (2010), International Society for Optics and Photonics, pp. 77980R–77980R.

[5] Cordeau, J.-F., Laporte, G., and Ropke, S. *Recent models and algorithms for one-to-one pickup and delivery problems*. Springer, 2008.

[6] Dantzig, G. B., and Ramser, J. H. The truck dispatching problem. *Management science 6*, 1 (1959), 80–91.

[7] De Koster, R., Le-Duc, T., and Roodbergen, K. J. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research 182*, 2 (2007), 481–501.

[8] Estanjini, R. M., Lin, Y., Li, K., Guo, D., and Paschalidis, I. C. Optimizing warehouse forklift dispatching using a sensor network and stochastic learning. *Industrial Informatics, IEEE Transactions on 7*, 3 (2011), 476–486.

[9] Gagliordi, N. SAP launches enterprise apps for smart glasses, 2014.

[10] Glockner, H., Jannek, K., Mahn, J., and Theis, B. Augmented reality in logistics. Verkkosivu, 2015. `http://www.dhl.com/content/dam/downloads/g0/about_us/logistics_insights/csi_augmented_reality_report_290414.pdf`. Haettu 27.7.2015.

[11] Gu, J., Goetschalckx, M., and McGinnis, L. F. Research on warehouse operation: A comprehensive review. *European journal of operational research 177*, 1 (2007), 1–21.

[12] Gu, J., Goetschalckx, M., and McGinnis, L. F. Research on warehouse design and performance evaluation: A comprehensive review . *European Journal of Operational Research 203*, 3 (2010), 539 – 549.

[13] Gutenschwager, K., Niklaus, C., and Voss, S. Dispatching of an electric monorail system: Applying metaheuristics to an online pickup and delivery problem. *Transportation science 38*, 4 (2004), 434–446.

[14] Ibach, P., Stantchev, V., Lederer, F., Weiss, A., Herbst, T., and Kunze, T. WLAN -based asset tracking for warehouse management. In *IADIS International Conference e-Commerce, Porto, Portugal* (2005).

[15] Liang, J. Z., Corso, N., Turner, E., and Zakhor, A. Image-based positioning of mobile devices in indoor environments. In *Multimodal Location Estimation of Videos and Images*. Springer, 2015, pp. 85–99.

[16] Lymberopoulos, D., Liu, J., Yang, X., Choudhury, R. R., Handziski, V., and Sen, S. A realistic evaluation and comparison of indoor location technologies: experiences and lessons learned. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks* (2015), ACM, pp. 178–189.

[17] Mautz, R., and Tilch, S. Survey of optical indoor positioning systems. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on* (2011), IEEE, pp. 1–7.

[18] Papaioannou, S., Wen, H., Markham, A., and Trigoni, N. Fusion of radio and camera sensor data for accurate indoor positioning. In *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on* (2014), IEEE, pp. 109–117.

[19] Pickl, S. Augmented reality for order picking using wearable computers with head-mounted displays, 2014. Kandidaatintyö. University of Stuttgart, Institute for Visualization and Interactive Systems.

[20] Powell, W. B., Bouzaiene-Ayari, B., and Simao, H. P. Dynamic models for freight transportation. *Handbooks in operations research and management science 14* (2007), 285–365.

[21] Rohrig, C., and Spieker, S. Tracking of transport vehicles for warehouse management using a wireless sensor network. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on* (2008), IEEE, pp. 3260–3265.

[22] SAP. Warehouse management system (WMS). Verkkosivu, 17. heinäkuuta 2015. `http://help.sap.com/saphelp_erp60_sp/helpdata/en/c6/f85c504afa11d182b90000e829fbfe/frameset.htm`. Haettu 17.7.2015.

[23] Savelsbergh, M. W., and Sol, M. The general pickup and delivery problem. *Transportation science 29*, 1 (1995), 17–29.

[24] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics 159*, 2 (2000), 139–171.

[25] Stefan Schröder. Benchmark VRPPD. Verkkosivu, 23. toukokuuta 2014. `https://github.com/jsprit/jsprit/wiki/Benchmark-VRPPD` Haettu 14.7.2015.

[26] TotalTrax. The Sky-Trax System. Verkkosivu, 2015. `http://www.totaltraxinc.com/smart-forklift-solutions/the-sky-trax-system/`. Haettu 12.9.2015.

[27] ZENO TRACK GmbH. Vehicle tracking. Verkkosivu, 2015. `http://www.zenotrack.com/index.php?id=48&L=1`. Haettu 12.9.2015.