VARASTOPAIKKOJEN KÄYTÖN OPTIMOINTI JA KERÄILYN SIMULOINTI

Turun yliopisto
University of Turku

Pro Gradu - tutkielma
Turun yliopisto
Tulevaisuuden teknologioiden laitos
Tietojenkäsittelytiede
Joulukuu 2020
Matti Ahonen

UNIVERSITY OF TURKU

Faculty of Science and Engineering / Department of Future Technologies

AHONEN, MATTI: Optimization of the use of storage locations and picking process simulation, Master's Thesis, 90 p., Computer science

October 2020

As the number of delivered articles and general competition continues to grow in the logistics sector, the efficiency of warehouse capacity utilization and operations is emphasized more than ever. The subject of this thesis comes from a practical problem where storage location content optimization turned out to be challenging in respect of the distance traveled by order collectors. The optimization algorithm designed for item placement enables the design of an optimized collection route according to selected parameters.

The target company's warehouse and its order collection area are vast, covering hundreds of storage locations. This research project implements a sorting algorithm that arranges items into their optimized locations according to the given parameters. In order to evaluate the results of the optimization, it is necessary to compare the differences between the different item placements in practice. In the evaluation, three different item placements have been taken into account: random item placement, currently used placement, and placement achieved through optimization. The evaluation method used is simulation, for which a separate simulation algorithm has been implemented.

The project also delivers percentage estimates of how significant the achieved benefits with the new item placement compared to the current situation can be. The benefits of optimization have also been converted into potential cost savings. The results show that an optimized item placement can reduce the distance traveled by up to 15% compared to the current one and by up to 27% compared to a random placement. The savings of 15% in the total distance traveled converted into cost savings is approximately 4.5% of the total operating costs of the average warehouse.

The results of the study align with previous studies on the effectiveness of volume-based item placement. By switching over to the placement proposed by the algorithm, one can achieve even the same benefits as changing from a random order to the current placement. Utilizing previous research data, the thesis introduces a new approach to an already known problem.

Keywords: optimization, programming, algorithms, logistics

TURUN YLIOPISTO

Luonnontieteiden ja Tekniikan tiedekunta / Tulevaisuuden teknologioiden laitos

AHONEN MATTI: Varastopaikkojen käytön optimointi ja keräilyn simulointi

Pro gradu tutkielma, 90 s., Tietojenkäsittelytiede

Lokakuu 2020

Logistiikka-alalla toimitettavien artikkelien määrän ja kilpailun alati kasvaessa varastojen käyttökapasiteetin sekä toiminnan tehostamisen tarve on korostunut. Tutkielman aihe tulee käytännön ongelmasta, jossa varastopaikkojen sisällön optimointi asiakastilausten keräilijöiden kulkeman matkan suhteen on haastavaa. Tuotesijoitteluun suunniteltu optimointialgoritmi mahdollistaa valittujen parametrien mukaan optimoidun keräilyreitin suunnittelun.

Kohdeyrityksen varasto ja sen keruualue ovat laajat kattaen satoja lavapaikkoja. Tutkimusprojektin tarkoituksena on toteuttaa lajittelualgoritmi, joka järjestää saatavilla olevat tuotteet annettujen parametrien mukaisesti niiden optimoiduille paikoille. Optimoinnin tulosten arvioimiseksi on vertailtava sitä, millaiset erot erilaisten tuotesijoitteluiden välillä käytännössä on. Arvioinnissa on otettu huomioon kolme erilaista tuotesijoittelua: satunnainen tuotesijoittelu, varaston nykyinen sijoittelu sekä optimoinnilla saavutettu sijoittelu. Arviointimenetelmänä käytetään simulointia, jota varten on toteutettu erillinen simulointialgoritmi.

Projektissa toimitettiin onnistuneesti myös prosentuaaliset arviot siitä, kuinka merkittävä hyöty uudella sijoittelulla voidaan saavuttaa nykytilanteeseen verrattuna. Optimoinnilla saavutettu hyöty on myös onnistuttu muuttamaan potentiaalisiksi kustannussäästöiksi. Kuten tuloksista käy ilmi, optimoitu asettelu voi vähentää liikkumiseen käytettyä matkaa nykyiseen verrattuna jopa 15% ja täysin satunnaiseen asetteluun verrattuna jopa 27%. 15 prosenttiyksikön säästö kuljetussa matkassa on kustannussäästöiksi muunnettuna noin 4,5% keskimääräisen varaston kaikista operatiivisista kustannuksista.

Tutkimuksen tulokset tukevat aiempia tutkimuksia volyymiperusteisen tuotesijoittelun tehokkuudesta. Algoritmin ehdottamaan sijoitteluun siirryttäessä voidaan saada jopa vastaavaa hyötyä, kuin siirryttäessä satunnaisesta järjestyksestä varaston nykyiseen sijoitteluun. Aiempaa tutkimustietoa hyödyntäen, tutkielma esittelee uudenlaisen lähestymistavan aiemmin tunnettuun ongelmaan.

Asiasanat: optimointi, ohjelmointi, algoritmit, logistiikka

Sisällys

1 Johdanto

Logistiikka-alalla toimitettavien artikkelien määrän ja kilpailun alati kasvaessa on varastojen käyttökapasiteetin sekä toiminnan tehostaminen korostunut. Asiakastilaukset ovat muuttuneet yhä pirstaloituneemmiksi, jolloin yhden suuren tilauksen sijasta usein toimitetaan monta pientä tilausta tiheämmällä toimitusrytmillä (Le-Duc & De Koster, 2007). Osaltaan tilausten pirstaloitumiseen vaikuttaa myös tuotteiden elinkaaren lyheneminen ja valmistuserien pienentyminen. Syynä edellä mainittuihin ilmiöihin voivat olla esimerkiksi toimitusten ja tuotteiden kohdentaminen yksittäisiin tapahtumiin tai sesonkeihin.

Keräilyllä tarkoitetaan tutkielmassa asiakastilauksen mukaisten tuotteiden keräämistä niille osoitetuilta varastopaikoilta. Keräilyä pidetään kaikkein eniten aikaa vievänä ja vaivalloisimpana osana varastointia, mistä johtuen voimme olettaa sen tarjoavan mahdollisuuksia optimoinnille (Le-Duc & De Koster, 2007).

Tutkielman aihe tulee käytännön ongelmasta, jossa varastopaikkojen sisällön optimointi asiakastilausten keräilijöiden kulkeman matkan suhteen on haastavaa. Tutkielman tavoitteena on selvittää voidaanko kohdevaraston keräilypaikkojen järjestystä optimoida niin, että asiakastilausta keräilevän työntekijän trukilla ajama matka annetulla reitillä lyhenisi verrattuna aiempaan. Keräilijän varastossa kulkema reitti on osittain ennalta määrätty varaston pohjapiirroksen sekä keräystyön aloitus- ja lopetuspaikan mukaan, eikä se siten mahdollista reitiltä poistumista sen mielivaltaisessa kohdassa. Reitillä on kuitenkin kaksi poikittaista käytävää, jotka tulee huomioida mahdollisia oikoreittejä arvioitaessa. Huomioitavaa on myös se, etteivät kaikki aktiiviset tuotteet mahdu annetulle keräilyalueelle, joten osa tuotteista jää ilman sijoittamista.

Tutkimushypoteesi on, että parantamalla keräilyalueen tuotesijoittelua vähenee tuottamaton työaika, kun kuljettu matka ja näin ollen keräilyyn käytetty aika lyhenevät. Useissa tapauksissa tilausten keräily kattaa arviolta noin 55% kaikista varaston operatiivisista kustannuksista, saman osuuden (55%) ollessa pelkästään liikkumiseen käytetty aika itse keräilyprosessissa (Bartholdi & Hackman, 2014). Täten pelkästään keräilyn aikana käytettyyn liikkumiseen kuluu noin 30% operatiivisista kustannuksista. On siis täysin ymmärrettävää, että yritykset pyrkivät vähentämään tätä kallista, mutta tuottamatonta liikkumista.

Ratkaisua keräilyn kustannustehokkuuden parantamiseen on useiden lähteiden mukaan haettu optimoimalla keräilyalueen pohjapiirrosta mm. pysty- ja poikkikäytävien määrää tutkimalla (Bottani, Montanari & Rinaldi, 2017). Lisäksi useissa tutkimuksissa keskitytään kuljettavan reitin optimointiin muun muassa dynaamisella ohjelmoinnilla, jolla voidaan valita esimerkiksi se mille

poikkikäytävälle on mielekkäintä siirtyä seuraavaksi (Roodbergen & De Koster, 2001). Useita tutkimuksia yhdistää se, että niissä kaikissa pyritään optimoimaan kuljettavaa reittiä annetussa ympäristössä ns. ”kauppamatkustajan ongelman” (TSP Travelling Salesman Problem) muunnelmana.

Tässä tutkielmassa ratkaisua optimointiongelmaan on lähdetty etsimään käänteisesti. Optimaalisen reitin löytämisen sijasta ajatuksena on optimoida tuotteiden sijoittelua niin, että jo annettu globaali (kaikki paikat kattava) reitti olisi aiempaa parempi. Ydinajatuksena on tarkastella ongelmaa joukkona käänteisiä kauppamatkustajan ongelmia (Chung & Demange, 2008), joissa yksittäinen asiakastilaus edustaa yhtä osaongelmaa. Tehtäessä muutoksia eniten myytävien tuotteiden sijoitteluun on sillä väistämättä vaikutusta moniin osaongelmiin, minkä johdosta kokonaisuudessa keräilyyn käytetyn matkan voidaan odottaa vähenevän. Tärkeimpiä muutoksia sijoittelussa ovat tuotteiden kokoon ja volyymiin perustuva tuotteiden uudelleen järjestely.

Tutkielma osoittaa, että optimoimalla tuotteiden sijoittelua esitetyllä tavalla, voidaan saavuttaa merkittävä parannusta nykyiseen asetteluun verrattuna. Optimoitu asettelu voi vähentää liikkumiseen käytettyä matkaa nykyiseen verrattuna jopa 15% ja täysin satunnaiseen asetteluun verrattuna jopa 27%. 15 prosenttiyksikön säästö kuljetussa matkassa on kustannussäästöiksi muunnettuna noin 4,54 % varaston kaikista operatiivisista kustannuksista.

Optimoinnin käytännön toimivuutta ja skaalautuvuutta suuremmille tilausmäärille on arvioitu simuloimalla keräilyä todellisilla tuotantotilauksilla. Simulointia varten on toteutettu algoritmi heuristisen etäisyysarvion tekemiseksi liikuttaessa varastopaikkojen välillä. Algoritmi huomioi mahdolliset oikoreitit sekä osaa ottaa huomioon tilauksen kannalta mahdollisesti tarpeettomat käytävät. Algoritmin tuottamat arviot ovat pessimistisiä, joten todellinen matka ei voi koskaan olla pidempi kuin algoritmin arvioima etäisyys. Simulointitulokset osoittavat, että optimoitu järjestys on kuljettua matkaa kuvaavalta kustannukseltaan alkuperäistä merkittävästi parempi myös suurilla tilausmäärillä. Simuloinnissa käytetyn tilausmäärän kasvaessa alkaa satunnaisen asettelun kustannus kasvaa merkittävästi alkuperäiseen ja optimoituun verrattuna.

Yleisesti varaston optimoinnista on tehty runsaasti aiempia tutkimuksia, jotka keräilyn osalta keskittyvät pääosin varaston pohjapiirroksen, keräilyreitin ja prosessien optimointiin. Luvun kaksi kuvailevassa kirjallisuuskatsauksessa käsitellään tutkittavaa ilmiötä tarkemmin keskittyen erityisesti keräilyprosessiin ja sen optimoinnin taustavaikuttajiin. Muilta osin tutkielman rakenne koostuu projektiluontoisesta toteutuksesta, jonka sisällä käsitellään varastoa toimintaympäristönä, saatavilla olevaa aineistoa, projektin toteutusta ja käytettyjä metodeja. Tämän jälkeen käsitellään toteutuksen arviointi ja tulokset, josta siirrytään yhteenvedon kautta jatkokehitysmahdollisuuksiin.

2 Projektin esivalmistelu

Kuten monissa tieto-ohjautuvissa projekteissa, on myös tämän tutkielman tapauksessa esivalmisteluilla ja saatavilla olevan datan laadulla suuri merkitys lopputuloksen kannalta. Jos väärää tai muuten huonolaatuista tietoa annetaan syötteeksi ohjelmalle on lopputuloksen paikkansa pitävyys epätodennäköistä. Tätä ilmiötä kuvataan yleisesti George Fuechselin alun perin esittämällä termillä ”garbage in, garbage out”, josta on sittemmin tullut laajalti käytetty ilmaisu kuvaamaan informaation laadun tärkeyttä lopputuloksen kannalta (Rouse, 2019).

Projektin esivalmisteluvaihe alkaa ongelman kuvaamisella, saatavilla olevan tiedon kartoittamisella ja sen ymmärtämisellä optimointitehtävän kannalta. Seuraavaksi tarkastellaan datan esikäsittelyä ja siihen käytettyjä menetelmiä. Lisäksi kuvataan keräilyä prosessina sekä kuljettavaa reittiä varaston pohjapiirrokseen nähden. Lopuksi arvioidaan valitun algoritmin soveltuvuutta annettuun tehtävään ja ympäristöön.

2.1 Ongelman kuvaus

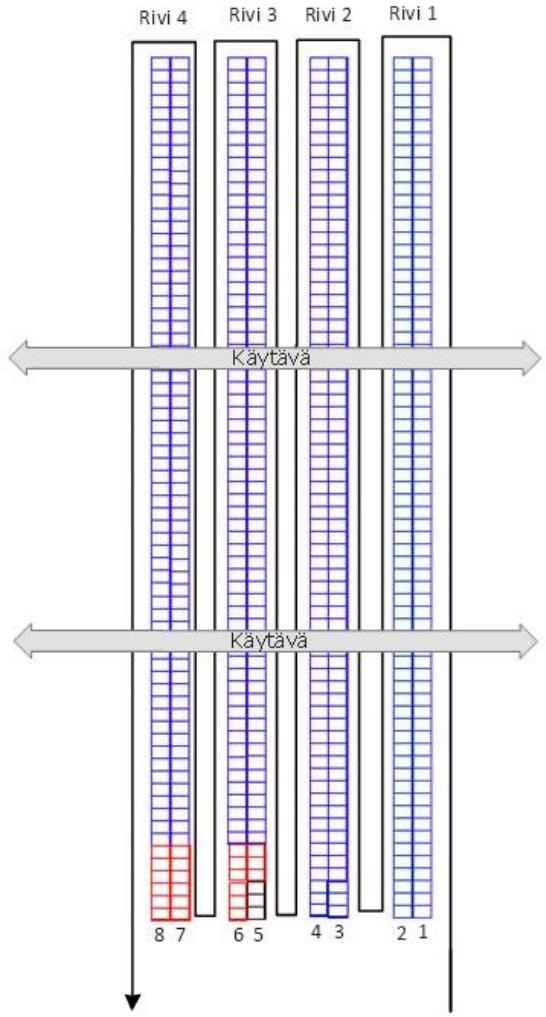
Projektin lähtökohtana toimii tarve tutkia, onko asiakastilausten keräilyprosessia mahdollista tehostaa optimoimalla tuotteiden järjestystä keräilyalueella. Oletamme, että on mahdollista lyhentää keräilyyn käytettävää aikaa parantamalla tuotteiden sijoittelua keräilyalueella, mikä toteutuessaan vähentäisi tuottamatonta työaikaa ja siten johtaisi kustannussäästöihin. Varaston nykyisessä tuotteiden sijoittelussa on todettu olevan parantamisen varaa, sillä usein asiakastilausta keräilevä työntekijä joutuu kulkemaan pitkiä matkoja annetulla reitillä ilman, että matkalla on yhtään tilaukseen poimittavaa tuotetta. Osaksi ongelmaa on myös annettu tieto siitä, että keräilyalueen ulkopuolelle jää toisinaan tuotteita, jotka olisivat menekkinsä (myyntivolyymin) puolesta parempi sijoittaa keräilyalueelle korvaten pienemmän menekin tuotteita.

Projektin tavoitteena on vastata seuraaviin kysymyksiin:

- Pystyykö keräilyprosessia tehostamaan optimoimalla tuotteiden sijoittelua keräilyalueella?
- Kuinka suuri hyöty optimoinnilla on saavutettavissa nykyiseen tilanteeseen verrattuna?

Keräilijän varastossa kulkema reitti on osittain ennalta määrätty varaston pohjapiirroksen sekä keräystyön aloitus- ja lopetuspaikan mukaan, eikä se siten mahdollista reitiltä poistumista sen

mielivaltaisessa kohdassa. Tämän reitin kattamaa aluetta kutsutaan keräilyalueeksi tai keräilyradaksi. Keräilyradan reitillä on kaksi poikittaista käytävää, jotka mahdollistavat liikkumisen myös poikittaissuunnassa. Kuvan 1 poikittaiskäytäviä voidaan hyödyntää siirryttäessä seuraavaan riviväliin, ellei sen hetkisellä rivillä ole enää muuta kerättävää. Tällöin siirryttäessä poikkikäytävää pitkin seuraavalle riville voidaan tällä rivillä tarvittaessa liikkua annetun kulkusuunnan vastaisesti aina sille lavapaikalle asti, jossa seuraava keräiltävä tuote sijaitsee. Näin voidaan menetellä, mikäli siitä ei ole haittaa muille radalla liikkujille. Tämän jälkeen noudatetaan normaalia kuvassa esitettyä S-muotoista kulkusuuntaa.



Kuva 1 Keräilyalue ja sen kulkusuunta

Tilauksen tai yksittäisen lavan valmistuessa tulee keräilijän palata samaan hyllypäätyyn mistä keräily on aloitettu. Tämä käytäntö lisää siis keräilyaikaan paluun takaisin lähtöpaikkaan. Paluukustannuksen suuruus riippuu siitä, kuinka pitkällä rivivälissä keräilijä on lopettamishetkellä. Keräilyaluetta ja mahdollisia kulkureittejä on kuvattu tarkemmin luvussa 2.5 sekä 5.3.

Huomioitavaa ongelman määrittelyssä on myös se, etteivät kaikki aktiiviset tuotteet mahdu annetulle keräilyalueelle, jolloin osa tuotteista jää väistämättä ilman sijoittamista. Tuotteilla voi myös olla muita niiden sijoittelua rajoittavia piirteitä kuten pakkauksen koko. Huomioiden edellä mainitut rajoitteet, on optimoinnissa tavoitteena täyttää keräilyalueen paikat tilausten keräilyn kannalta parhailla mahdollisilla tuotteilla. Uusien myyntiennusteiden päivittyessä esimerkiksi kerran kuukaudessa, voidaan sen olettaa vaikuttavan optimointituloksiin ja siten optimointiprosessi tulee suorittaa uudestaan säännöllisin väliajoin.

2.2 Aineiston kerääminen, luokittelu ja analysointi

Annettua aineistoa on kerätty jo ennen projektia tallentamalla tuotteiden myyntiennusteet ja toteumat kuukausittain Excel-tiedostoihin. Tämän lisäksi materiaalina ovat simuloinnissa käytetyt kopiot tuotantotilauksista sekä ERP-järjestelmästä saatavat tuotetiedot.

Projektin aikana tuotettua materiaalia ovat mm.

- Tekstimuotoon muutettu keräilyalueen kartta (paikka, kustannus palata, käytävän puoli).
- Raportti keräilyalueen paikkojen varauksista lattiatasolla.
- Optimointi algoritmin tuottama lista optimaalisista tuotteista paikoittain.
- Kartoitus-tiedosto tilauksen ja nykyisten keräilypaikkojen yhdistämiseksi.
- Tiedosto vuoden 2019 syyskuussa keräillyistä tilausriveistä tuotetasolla (epärelevantit ja laskuttamattomat rivit on rajattu pois esim. pantit, rahtiveloitukset ja lavat).

Projektin kohdevarastossa kaikki tuotteet ovat alkoholijuomia lukuun ottamatta tiettyjen tuotteiden alkoholittomia versioita (viinit ja oluet), mikä tekee aineistosta hyvin homogeenisen. Tämä näkyy yhteneväisinä piirteinä kaikkien tuotteiden välillä, joita ovat mm. seuraavat:

- Kaikki tuotteet kuuluvat samaan pääkategoriaan ja niillä on pääasiallisesti yhteneväiset myyntikanavat.
- Tuotteiden tai tuotekategorioiden kausivaihtelut ovat melko hyvin ennustettavissa historiallisen myynnin perusteella.

- Tuotteiden kysyntä vaihtelee tasaisesti pääkategorian (alkoholit) sekä alakategorioiden mukaisesti (esim. punaviinit, kuohuviinit, alkoholittomat ja vähäalkoholiset juomat).
- Kaikkien tuotteiden menekkiä voidaan ennustaa ja käsitellä samassa yksikössä (pulloina/litroina).

Seuraavassa taulukossa (Taulukko 1) on esitetty kerättyä materiaalia eri tunnuslukujen muodossa. Tuotteiden ja paikkojen määrästä voidaan nähdä, ettei vapaita paikkoja ole saatavilla keräilyalueelta läheskään kaikille tuotteille. Tilausrivien määrästä voidaan päätellä, että jo yhden kuukauden aikana kerätään merkittävä määrä asiakastilauksia ja se tulee ottaa huomioon arvioitaessa optimoinnin toteutusta. Lisäksi on ilmoitettu tuotteiden ja paikkojen käytetyt painoarvot, joista jatkossa paikkojen painoarvot säilyvät muuttumattomina, mutta tuotteiden painoarvot vaihtelevat kysynnän mukaan. Kappaleessa 3.1.3. on kerrottu tarkemmin painoarvojen muodostamisesta ja niiden käyttötarkoituksesta.

Kategoria	**Kappaletta / yksikkö**
Aktiivisia tuotteita	2346 kpl, joista 1442:lla on ennustettua tai toteutunutta myyntiä.
Keräilypaikkoja keräilyradan alueella	597 kpl
Tilausrivejä myyntitilausaineistossa (1kk)	108158 riviä (toteutuksessa on valittu satunnaisesti tilauksia siten, että käytössä on 45191 riviä).
Simuloinnissa käytettyjä tilauksia	4683 kpl
Tuotteiden painoarvot välillä	1-19 (ka. 14,08)
Paikkojen painoarvot välillä	5-10 (0 = paikka ei käytössä)

Taulukko 1 Aineisto lukuina

2.3 Tietojen esikäsittely

Varastopaikat

Varastopaikkojen tiedot ovat tyypiltään diskreettejä eikä esimerkiksi niiden sijaintia, kapasiteettia tai numerointia voida muuttaa. Tämä mahdollistaa sen, että paikoille voidaan tehdä esivalmisteluja, jotka toimivat kaikilla suorituskerroilla syötteestä riippumatta.

Varastopaikoille on tehty seuraavat määrittelyt:

- Kullekin paikalle on määritelty pienin sallittu tuotteen koko.
- Paikoille on annettu painoarvo välillä 0-10 painottamalla keräilyradan alkuosat parhaiksi paikoiksi ja laskemalla luokitusta loppua kohden. Tämä perustuu oletukseen siitä, että mitä pidemmälle rataa joudutaan kulkemaan, sitä enemmän väliin on jäänyt tyhjiä paikkoja joissa optimitilanteessa olisi ollut tilaukseen tarvittavia tuotteita.
- Paikat joille ei voida asettaa tuotetta (paikalla esim. käytävä) on rajattu pois asettamalla minimi tuotekooksi 99 ja painoarvoksi 0.

Tuotteet

Tuotteiden osalta esikäsittelynä suodatettiin pois tuotteet, jotka eivät kuulu keräilyalueelle, ovat lopetettuja tai niiden ennustettu myynti on 0. Jäljelle jääville tuotteille lasketaan keskiarvo menneen 4 kuukauden toteutuneesta myynnistä sekä tulevan 4 kuukauden ennustetusta myynnistä. Keskiarvoa käytetään, jotta täysin uudet tuotteet saataisiin tasavertaisina mukaan aineistoon/dataan.

Tuotteille on tehty seuraavat määrittelyt.

- Tuotekohtaisesta volyymista (keskiarvo +/- 4kk) on otettu 2-kantainen logaritmi, joka on pyöristetty kokonaisluvuksi (alkuperäiset arvot välillä 100 – 300 000).
- Kullekin tuotteelle on määritelty laatikkokoko, jota voidaan verrata keräilypaikan pienimpään sallittuun kokoon.

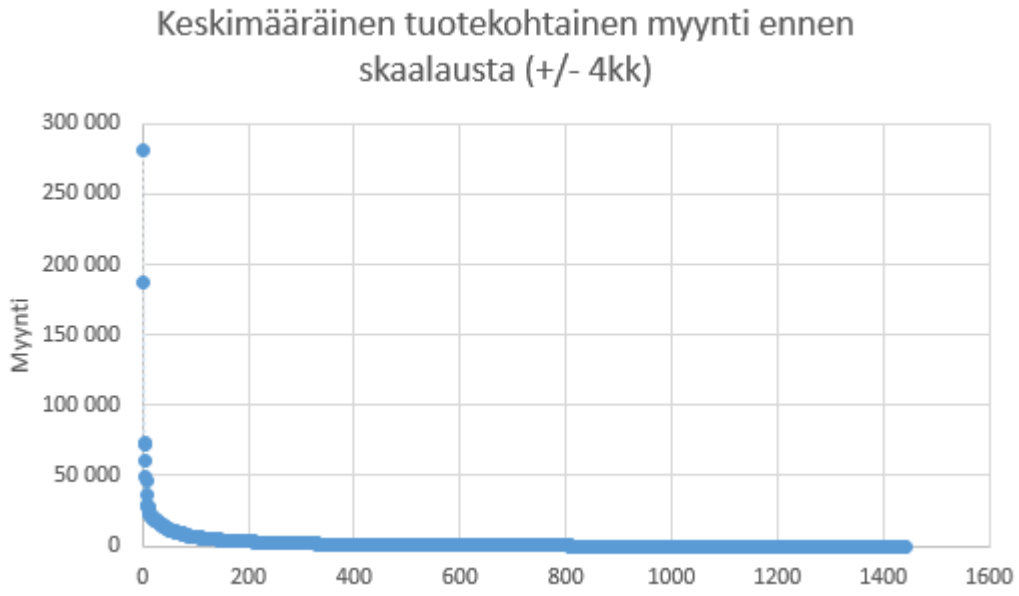
Tilaukset

Työssä käytettävät tilaukset ovat tuotannosta kopioituja tilauksia, jotka on hankittu kopioimalla yrityksen toiminnanohjausjärjestelmän tilausrivitaulusta kaikki rivit kuukauden ajalta. Tilauksista on rajattu pois laskuttamattomat tilaukset, epärelevantit tuotteet ym. selkeästi tarpeettomat rivit. Jäljelle jääneistä riveistä on luotu Pandas-kirjastoa apuna käyttäen omat yksittäiset tilaukset, jotka on tallennettu erillisiin tiedostoihin. Tilausten jakamisen toteutus nähtävissä liitteessä (Liite 3).

2.3.1 Ymmärtäminen ja normalisointi

Kun saatavilla oleva aineisto on määritetty ja tiedetään, millaisen tietokokonaisuuden kanssa työskennellään voidaan aloittaa tietojen esikäsittelyn seuraava vaihe eli datan havainnointi ja ymmärtäminen. Datan graafinen visualisointi auttaa kuvaamaan ja ymmärtämään tietokokonaisuutta paremmin, mikä auttaa erityisesti poikkeamien havainnoinnissa. Tässä vaiheessa pystytään osoittamaan tarpeelliset ja tarpeettomat tiedot sekä luomaan uusia muuttujia hyödyntäen alan tuntemusta. (Boschetti & Massaron, 2016, s. 65.)

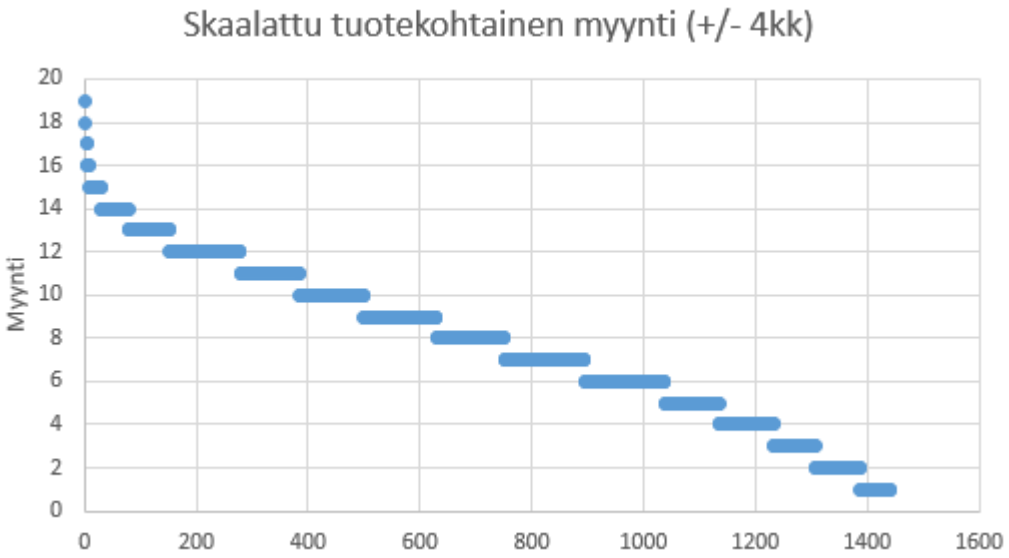
Visualisoitaessa tuotekohtainen myynti kuvan 2 mukaisesti, voidaan sen nähdä jakautuvan hyvin epätasaisesti laajalle asteikolle tuotekohtaisten arvojen sijoittuessa välille 1-280 000. Suurimmalla osalla tuotteita (1236 kpl) neljä kuukautta eteen- ja taaksepäin katsova keskimääräinen myynti painottuu välille 1-3101 kappaletta kuukautta kohden. Visualisoinnin perusteella voidaan todeta, ettei tuotteiden paremmuuden arvioinnissa ole mielekästä käyttää yhdelle parametrille näin laajaa asteikkoa, sillä erot tuotteiden sijoittelun kannalta tuskin ovat näin suuria. Optimointialgoritmi pyrkii sijoittamaan kullekin vapaana olevalle varastopaikalle parhaan vapaana olevan tuotteen sen keskimääräisen myynnin mukaan mitattuna. Sillä ei ole käytännön kannalta merkitystä onko myynti esimerkiksi 500 vai 800 pulloa kuukaudessa, joten algoritmin ei ole tarpeellista tehdä eroa tällaisten tuotteiden välille. Mikäli ero tuotteiden välillä on tuhansia pulloja tulee se huomioida laitettaessa tuotteita niiden paremmuusjärjestykseen.



Kuva 2 Tuotteiden jakauma keskimääräisen myynnin mukaan ennen skaalausta.

Jotta keskimääräinen myynti saadaan parametrina skaalautumaan käyttökelpoiselle asteikolle, on siitä otettu 2-kantainen logaritmi, joka tunnetaan myös binäärisenä logaritmina. Logaritmi on valittu skaalaamaan tuotteet siitä syystä, että se osoittautui käytännössä parhaiten toimivaksi ratkaisuksi painottaen kaikki tuotteet välille 1-19. Lopuksi tuotteiden painoarvot on pyöristetty ylöspäin kokonaisluvuiksi ja järjestetty niille lasketun painoarvon mukaiseen ei-vähenevään suuruusjärjestykseen jatkokäsittelyä varten. Näin tuotteiden myynneistä saadaan esiin käytännön kannalta merkitykselliset erot, joka mahdollistaa tuotteiden paremmuuden vertailun sopivan kapealla asteikolla. Tiedosto Item_Volume2.csv (Liite 4).

Jos tuotteiden painotuksessa halutaan ottaa keskimääräisen myynnin lisäksi huomioon muita ominaisuuksia, tulisi niiden keskiarvon (mean value) olla samaa suuruusluokkaa tämän painoarvon kanssa. Muiden mahdollisten ominaisuuksien skaalaaminen samalle asteikolle voi myös parantaa koneoppimista, mikäli aineistoa käytettäisiin opetusaineistona. (Boschetti & Massaron, 2016, s. 263.) Tämä aiempaa kapeampi asteikko helpottaa myös manuaalista säätämistä siinä tapauksessa, että tuotteiden prioriteetteja halutaan muuttaa. Logaritmin käyttö myös poistaa tehokkaasti vinouman kohti suuria yksittäisiä lukuja, joita esiintyy aineistossa vain harvoin. Kuva 3 esittää tuotekohtaisen myynnin jakautumista skaalauksen jälkeen.



Kuva 3 Tuotekohtaisen myynnin jakauma skaalauksen jälkeen.

Varastopaikkojen osalta normalisointia tai muun tyyppistä skaalausta ei tarvita, sillä niiden järjestys ja keskinäiset suhteet määräytyvät varaston pohjapiirroksen perusteella, emmekä siten

voi vaikuttaa siihen. Osana esivalmistelua varastopaikoilta on poistettu etu- ja jälkiliitteet, jotta niitä voidaan käsitellä numeerisessa muodossa. Esimerkiksi ensimmäinen keräilypaikka P001A on muutettu muotoon 001, sillä P:n kuvaama varastoalue tai A:n kuvaama hyllykerros eivät ole tutkielman kannalta oleellisia.

Varastopaikkojen järjestyksen ymmärtämiseksi niistä on toteutettu esimerkinomainen visuaalinen esitys, joka kuvaa keräilyalueen paikat riveittäin (Kuva 4). Tästä esityksestä voidaan nähdä kuinka monta paikkaa kullakin rivipuoliskolla voi enimmillään olla (69 kpl) ja mikä on kustannus palata takaisin alkuun mistä tahansa keräilyalueen paikasta. Kustannus on esitetty ainoastaan kuvan oikeassa laidassa, sillä se säilyy samana huolimatta siitä, missä rivivälissä keräilijä on lähtiessään palaamaan takaisin rivin alkupäähän. Poikittaisen liikkumisen aiheuttamaa kustannusta ei ole tarpeen huomioida, sillä jokaisen rivin alkupäässä on tilauksille jättöpaikka, jossa keräystehtävän viimeistely on mahdollista. Rivien numeroidut solut kuvaavat keräilypaikkojen numerointia ilman etu- tai jälkiliitettä. Numerot havainnollistavat myös keräilyradalla liikkumiseen tarkoitettua kulkusuuntaa. Harmaa alue kuvaa poikittaista käytävää jossa alimman A-kerroksen paikat eivät ole käytössä.

A	B		A	B		A	B		
Rivi 3			Rivi 2			Rivi 1			Kustannus palata
346	345		208	207		70	69		69
347	344		209	206		71	68		68
348	343		210	205		72	67		67
349	342		211	204		73	66		66
350	341		212	203		74	65		65
351	340		213	202		75	64		64
352	339		214	201		76	63		63
353	338		215	200		77	62		62
354	337		216	199		78	61		61
355	336		217	198		79	60		60
356	335		218	197		80	59		59
357	334		219	196		81	58		58
358	333		220	195		82	57		57
359	332		221	194		83	56		56
360	331		222	193		84	55		55
361	330		223	192		85	54		54
362	329		224	191		86	53		53
363	328		225	190		87	52		52
364	327		226	189		88	51		51
365	326		227	188		89	50		50
366	325		228	187		90	49		49
367	324		229	186		91	48		48
368	323		230	185		92	47		47
369	322		231	184		93	46		46
370	321		232	183		94	45		45
371	320		233	182		95	44		44
372	319		234	181		96	43		43
373	318		235	180		97	42		42
374	317		236	179		98	41		41
375	316		237	178		99	40		40
376	315		238	177		100	39		39
377	314		239	176		101	38		38
378	313		240	175		102	37		37
379	312		241	174		103	36		36
380	311		242	173		104	35		35
381	310		243	172		105	34		34
382	309		244	171		106	33		33
383	308		245	170		107	32		32
384	307		246	169		108	31		31
385	306		247	168		109	30		30
386	305		248	167		110	29		29
387	304		249	166		111	28		28
388	303		250	165		112	27		27
389	302		251	164		113	26		26
390	301		252	163		114	25		25
391	300		253	162		115	24		24
392	299		254	161		116	23		23
393	298		255	160		117	22		22

Kuva 4 Esimerkki varastopaikkojen visualisoinnista riveittäin sekä kustannuksesta palata takaisin rivin alkuun kultakin keräilypaikalta. Kuvassa ei näy koko keräilyalue.

Edellä kuvatun varastopaikkojen visualisoinnin perusteella on luotu simuloinnissa käytetty tekstimuotoinen esitys LocationSimulation.csv (Liite 4). Tiedostossa varastopaikat on listattu kokonaislukuina ei-vähenevään suuruusjärjestykseen, samalla kertoen jokaisen paikan kohdalla sen paluukustannuksen takaisin alkuun sekä millä hyllyrivin puolikkaalla paikka sijaitsee (1-8). Kohdevarastossa on hyllyrivejä yhteensä neljä ja koska jokaisessa rivissä on käytössä kaksi puolta, on siten käsiteltäviä rivipuoliskoja yhteensä kahdeksan.

2.4 Keräilyprosessi

Keräily kuuluu prosessina osaksi asiakastoimitusten käsittelyä aloittaen asiakastoimituksen valmistamisen. Keräysmenetelmät jakaantuvat kahteen ryhmään riippuen siitä meneekö keräilijä tavaran luokse vai tuleeko tavara keräilijän luokse esim. automaattihissillä. (Karhunen, Pouri & Santala, 2004) Tämän tutkielman kohdevarastossa keräily tapahtuu aina siirtymällä keräiltävälle paikalle, eikä käytössä ole hissein varustettua korkeavarastoa tai muunlaista automaatiota tavaroiden tuomiseksi kerääjän kohdalle. Keräilyyn käytetty kokonaisaika voidaan karkeasti jakaa ajamiseen tai kävelyyn sijaintien välillä (matka-aika), tuotteiden poimimiseen ja aikaan joka käytetään muihin aktiviteetteihin kuten keräilytehtävän aloittamiseen ja lopettamiseen (uuden keräilyalustan hakeminen ja valmiin lavan pois vieminen) (Holste, 2009).

Kohdevarastossa on keräiltäviä tuotteita varten eriytetty alue, jossa liikutaan käyttäen erilaisia keräilyyn ja hyllyttämiseen tarkoitettuja trukkeja. Keräilytyö alkaa aina keräilyradan alusta ja yksittäinen keräystehtävä loppuu, kun keräiltävä alusta (esim. EUR-lava tai rullako) on täynnä tai asiakastilaus on kokonaan valmis. Tällöin tulee jatkokäsittelyä varten palata takaisin samaan hyllypäätyyn, josta keräilytyö on alkanut. Uusi keräysmääräys annetaan päätelaitteelle automaattisesti, eikä sen saamiseksi tarvitse tehdä mitään toimenpiteitä.

Karhusen ym. (2004, s.378) mukaan tehokas keräystyö edellyttää osoitejärjestelmää (Address system) ja sopivien keräysreittien muodostamista eikä järjestys siten voi olla sattumanvarainen. Osoitejärjestelmän eli varastopaikkajärjestelmän tarkoituksena on helpottaa ja nopeuttaa tuotteiden löytymistä varastosta osoittamalla tuotteiden sijainti yksiselitteisesti. Osoitejärjestelmä on yleensä varastokohtainen ja se voi koostua numeroista, kirjaimista tai näiden yhdistelmistä. Osoitteiston tulisi olla selkeä, loogisesti etenevä ja helposti omaksuttava myös uusille työntekijöille (Benson, 2020).

”*Keräysreitit muodostetaan yleensä siten, että nimikkeiden ottotiheyden mukaan usein kysytyt nimikkeet ovat keräysreitin alussa, jolloin useimmissa keräyskerroissa keräys voidaan lopettaa jo keräysreitin alkupäässä ja näin pitää kuljettavat matkat lyhyinä*”. (Karhunen ym., 2004)

Työssä käytetty neljä kuukautta taakse- ja eteenpäin katsova tuotekohtainen volyymi on toimiva piirre tuotteiden järjestämiseksi niiden ottotiheyden mukaiseen järjestykseen, mutta sen lisäksi tulee ottaa huomioon muitakin näkökohtia. Esimerkiksi painavat tavarat tulee sijoittaa keräilyradan alkuun ja helposti särkyvät loppuun, koska muuten on riskinä heikompien tavaroiden rikkoontuminen niiden päälle lastatusta painosta johtuen (Karhunen ym., 2004).

Tuotteiden painon sijaan työssä on käytetty tuotteen pakkauskokoa indikoimaan sen kantavuutta ja siten sijoittumista keräilyalueella. Oletuksena on se, että jos tuote on pakattu esim. 12 yksikön (yleensä pulloa) pakkaukseen on se hyvin todennäköisesti suurempi ja painavampi kuin 6 yksikön pakkaukset tai sitä pienemmät. Toisaalta osassa tuotteista saattaa pakkaukseen kuulua useita yksiköitä, mutta ne ovat kooltaan pieniä kuten esimerkiksi 12 kpl 5cl kokoisia pulloja.

Tällaisissa poikkeustapauksissa tulee puntaroida, onko yksikkömäärältään suuri pakkauskoko riittävä peruste siirtää tuote keräilyalueen alkupäähän vai ei. Ongelmallisissa tapauksissa tuotekohtaista sijoittumista voidaan tarvittaessa manuaalisesti ohjata säätämällä tuotteiden painokertoimia ennen järjestyksen optimoinnin suorittamista. On varsin ilmeistä, että tilavuudeltaan vain 0,6 litran (12 x 0,05cl) kokoinen pakkaus ei lukeudu parhaisiin tuotteisiin lastattavaksi kuorman alimmaiseksi. On kuitenkin erittäin epätodennäköistä, että kyseisen kaltainen tuote olisi myyntilitrojensa puolesta suurimpien tuotteiden joukossa. Niinpä voidaan olettaa, että keräilyalueen alussa sijaitsevat paikat täyttyvät muilla volyymiltaan suuremmilla tuotteilla, joilla on myöskin suuri pakkauskoko. Tällöin tämä poikkeustuote sijoittuu johonkin myöhempään kohtaan keräilyrataa, jossa minimilaatikkokoko saattaa olla myös pienempi kuin esimerkin 12 kpl. Tarvittaessa on mahdollista ottaa sijoittelussa käyttöön myös paino-parametri, mutta sen tarpeellisuus näin homogeenisen tuoteportfolion osalta ei ole mitenkään ilmeistä.

Muita keräilyn tehokkuuteen merkittävästi vaikuttavia asioita ovat mm. keräysmääräyksen osoittama keräilyjärjestys, jonka tulee järjestää asiakastilauksen tuotteet järjestykseen perustuen tehokkaisiin keräysreitteihin. Tässä tapauksessa tehokas keräysreitti olisi valmiiksi annettu reitti radan alusta sen loppuun, sisältäen optimoidut tuotteet kullekin paikalle. Toinen tehokkaalle keräykselle tärkeä periaate on se, että samalla keräyskerralla voidaan keräillä usean asiakkaan tuotteet, jotka ovat menossa samaan osoitteeseen. Tämä mahdollistaa sen, että hyllystössä liikuttua matkaa kohden kerätyn tavaran määrä on mahdollisimman suuri. (Karhunen ym., 2004.) Kohdevarastossa tilausten yhdistäminen osoitteiden perusteella yksittäiseksi keräilytehtäväksi on

toteutettu varastonhallintajärjestelmässä, kuten myös keräysmääräyksen järjestäminen annetun reitin mukaisesti.

Yleisesti keräilytoiminnot kattavat merkittävän osan varaston operatiivisista kustannuksista, mikä lähteestä riippuen, voi olla jopa 55% (Bartholdi & Hackman, 2014) tai peräti 65% (HUB Logistics, 2009) kaikista varastoinnin kustannuksista. Yksinkertaisillakin parannuksilla voidaan siten saavuttaa merkittäviä keräilyajan lyhennyksiä, jotka sekä alentavat kustannuksia että pienentävät tilausten käsittelyyn käytettyä aikaa. Kuten sanottu, on tuotteiden sijoittelu yksi tilausten käsittelyyn olennaisesti vaikuttava osio. Tutkimusten mukaan volyymiin perustuva tuotteiden sijoittelu (VBS, volume-based-storage) on kaikkein tehokkain menetelmä, mutta se vaatii tietoa tuotteiden liikkumisesta ja säännöllistä ylläpitoa. Muita tyypillisiä sijoittelumenetelmiä ovat mm. satunnainen sijoittelu (random storage) sekä tuoteperheisiin perustuva sijoittelu, joka on lähimpänä kohdevaraston tämän hetkistä sijoittelua (HUB Logistics, 2009). Holstenin (2009) mukaan satunnainen sijoittelu on kaikista helpoin käyttää, sillä siinä tuotteet voidaan sijoittaa mihin tahansa vapaaseen paikkaan.

Petersenin ym. (2004) tutkima luokkiin perustuva sijoittelu (CBS, class-based-storage) hyödyntää volyymiin perustuvaa jakoa eri keräilyalueille siten, että eniten liikkuvat tuotteet sijoitetaan helpoiten saavutettavissa oleville alueille. Tutkimuksen mukaan luokkiin perustuvat sijoittelut olivat keräilytehokkuudeltaan 12% - 26% tehokkaampia kuin satunnainen sijoittelu ja sillä saavutettiin 78-94% volyymiin perustuvan luokittelun tehokkuudesta. Tutkimuksessa huomattiin, että mitä pidemmät keräilylistat ovat, sitä vähemmän merkitystä tuotteiden sijoittelulla on tehokkuuteen. Tämä johtuu siitä, että luokkaperustaisessa mallissa pitkät keräyslistat todennäköisemmin myös sisältävät vähemmän liikkuvia tuotteita huonommista luokista. Mitä keskittyneempää yritysten tuotteiden kysyntä on, sitä enemmän voidaan saada hyötyä luokkiin perustuvasta sijoittelusta. Keskittyneisyydellä tarkoitetaan tuotteiden kysynnän jakautumista esimerkiksi siten, että 20% kaikista tuotteista muodostaa 80% niiden kokonaiskysynnästä. (HUB Logistics, 2009.)

Luokkiin perustuva sijoittelu voisi toimia kohdevarastossa tuotteiden keskittyneisyyden puolesta hyvin, sillä 20% tuotteista (288 kpl) muodostaa 89% kokonaisvolyymista. Loput 11% jakaantuu epätasaisesti 1153:n tuotteen kesken, joten niiden sijoittelu ei ole kokonaisuuden kannalta yhtä merkityksellistä. Huomioitavaa on myöskin se, että vähiten myyvä 20% kaikista tuotteista kattaa kokonaismyynnistä vain 0,05%.

2.5 Pohjapiirros ja kuljettava reitti

Optimaalisen reitin määrittäminen voi varastonpohjapiirroksesta riippuen olla haastavaa tai jopa mahdotonta käytettävissä olevilla työkaluilla. Täysin optimaalinen reitti edellyttäisi tarkan geometrisen mallin varaston asettelusta, jonka avulla tietokone pystyisi laskemaan jokaisen varastopaikan keskinäisen etäisyyden suhteessa ajoreittiin (Bartholdi & Hackman, 2014). Bartholdin & Hackmanin (2014) mukaan vielä vuonna 2014 ei yksikään silloin tutkituista varastonhallintajärjestelmistä hallinnut täysin tarkkaa varaston mallinnusta, jotta todellinen optimaalisuus voitaisiin todentaa. Käytännössä reitin optimointi voidaan usein ratkaista käyttämällä heuristiikkaa, jolla pystytään välttämään täysin optimaalisen reitin huonoja ominaisuuksia. Huonoja ominaisuuksia ovat mm. reitin epäloogisuus, vaikea hahmottaminen ja tiettyjen käytävien mahdollinen ruuhkaantuminen (De Koster ym., 2007, s.18). Tällaisia ongelmia esiintyy todennäköisimmin silloin, kun optimoidaan vain yhtä tekijää, jolloin muut asiat voivat kärsiä.

Heuristiikan sijasta vaihtoehtona optimaalisen reitin huonojen ominaisuuksien välttämiseksi voisi toimia monitavoiteoptimointi. Monitavoiteoptimointia tarvitaan silloin, kun päätöksen teko edellyttää usean ristiriitaisen tavoitteen optimoimista samanaikaisesti (Miettinen, 2009). Monitavoiteoptimointitehtävässä on useita optimoitavia kohdefunktioita eli ns. kriteereitä. Reitin määrittämisessä tällaisia kriteereitä voisivat olla mm. käytävien annettu kulkusuunta, ruuhkautumisen välttäminen, tietyn tuotekategorian lastaaminen kuorman alimmaiseksi tai päällimmäiseksi ja muut keräyksen laadun varmistamiseen liittyvät ehdot.

Monitavoiteoptimointi voi auttaa löytämään parhaan kompromissin useiden ristiriitaisten tavoitteiden väliltä, olipa sitten tavoitteena minimoida keräilyn aikana kuljettu matka tai järjestää tuotteet lavalle optimaaliseen järjestykseen. Tällaisessa tapauksessa yksiselitteisen optimin sijasta käsiteltävänä on kompromissiratkaisujen joukko eli niin sanottuja Pareto-optimeja, joista yhdenkin ratkaisun arvon parantaminen vaikuttaa negatiivisesti toisen ratkaisun tavoitteen arvoon. Monitavoiteoptimoinnissakin joudutaan yleensä valitsemaan yksi ratkaisu lopulliseksi toteutukseksi, mikä vaatii inhimillisen päätöksentekijän osallisuuden tapauksessa, jossa kompromissiratkaisut ovat keskenään yhtä hyviä. (Miettinen, 2009.)

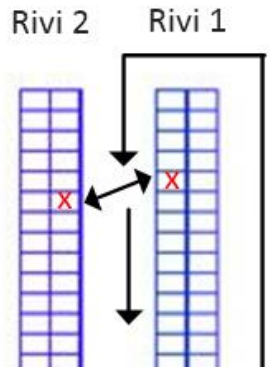
Vaikka monitavoiteoptimoinnilla voitaisiin saavuttaa optimaalinen reitti, joka välttäisi edellä mainittuja optimaalisuuden aiheuttamia ongelmakohtia, olisi sen käytännön toteuttaminen haastavaa. Keräilyreittiin vaikuttavien tavoitteiden järkevä painottaminen vaatii syvällistä tutustumista todellisiin riippuvuussuhteisiin eri tavoitteiden välillä.

Tilauksen keräämiseen käytetystä ajasta jopa yli 50% kuluu liikkumiseen keräilypaikkojen välillä. Erilaiset reititys- ja varastointi strategiat voivat molemmat osaltaan vähentää keräilijöiden kulkemaa matkaa, mutta yhdessä niiden vaikutus on kaikkein tehokkain. (Holste, 2009.)

Käytännössä keskimääräiseen liikkumiseen käytettyyn aikaan vaikuttaa monet tekijät. Esimerkiksi se miten varaston hyllyt on aseteltu ja minkä tyyppisiä ne ovat, varaston pinta-ala, laitekanta, keräystehtävien koko, aloitus- ja lopetuspisteiden sijainti ym. paikalliset tekijät. (Roodbergen & De Koster, 2000.) Tämän kaltaisista ympäristötekijöistä johtuen, on tavoitteena vähentää keräilyyn tarvittavaa aikaa muuttamatta varaston pohjapiirrosta tai materiaalien käsittelyyn käytettyä välineistöä. Tämä ei kuitenkaan ole helppo tehtävä, sillä se kattaa valtavan määrän eri vaihtoehtoja. (Holste, 2009.) Tässä tutkimuksessa keskitytään arvioimaan ainoastaan tuotteiden sijoittelun vaikutusta liikkumiseen käytettyyn aikaan. Erityisesti tavoitteena on selvittää, onko tuotteiden sijoittelulla valmiiksi annetulla reitillä merkittävää vaikutusta kuljettuun matkaan ja siten myös keräilytehtävään käytettyyn aikaan.

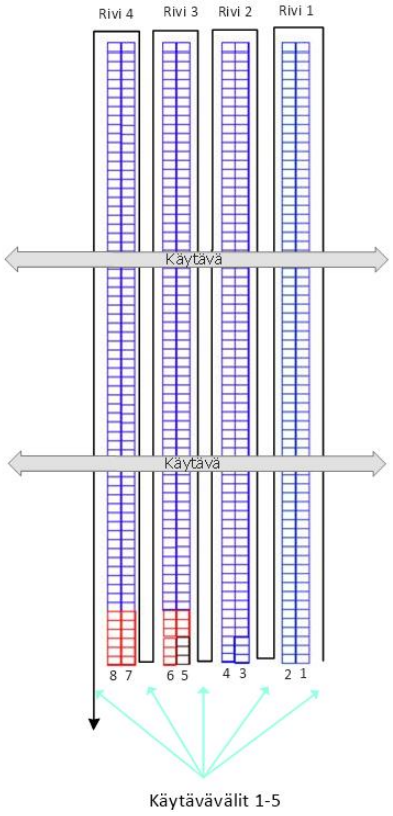
Varaston pohjapiirrosta tarkasteltaessa on tärkeää huomioida, että keräilyreitti rajoittuu ainoastaan vaaka- ja pystysuunnassa kuljettaviin käytäviin sekä kahteen poikittaiseen käytävään, joista voidaan siirtyä seuraavaan käytäväväliin (Kuva 1, Luku 2.1). Reitti on täten vain osittain ennalta määrätty eikä kaikkien rivivälien kautta tarvitse kulkea, jos halutut tavarat ovat jo mukana. Tämä on oleellista, sillä muussa tapauksessa kuljettua matkaa ei voisi optimoida. Varaston pohjapiirroksen malli on nimeltään ns. köysitikasasettelu, joka tarkoittaa sitä, että tuotteet on varastoitu hyllyihin ja hyllyt on järjestetty useille riveille muodostaen köysitikasmaiset käytävät sekä poikkikäytävät (Seward, 2015).

Keräilyalueella on voimassa ns. normaalit liikennesäännöt eikä käytäväpuoliskon ajaminen annetun kulkureitin vastaiseen suuntaan ole suositeltua, muutoin kuin poikittaiskäytäviä pitkin tapahtuvan käyvävälin vaihdon yhteydessä. Myöskin samassa käytävävälissä tapahtuva molemminpuolinen keräily on rajattu pois sen epäkäytännöllisyyden ja vaikean ennustettavuuden vuoksi. Käytännössä tilanne on mahdollinen, jos tilauksella on olemassa tuote, joka sijoittuu sopivasti vastakkaiselle puolelle käytävää, ollen vertikaalisesti samalla tai lähes samalla kohdalla karttaa jolta keräilijä on poiminut aiemman tuotteen. Teoriassa vastakkaisen käytäväpuolen tuotteen poimiminen samalla kerralla toisen tuotteen kanssa voisi vähentää kuljettua matkaa, mutta se myös lisäisi tarpeettomasti riskiä käytävän ruuhkaantumiseen (Kuva 5).



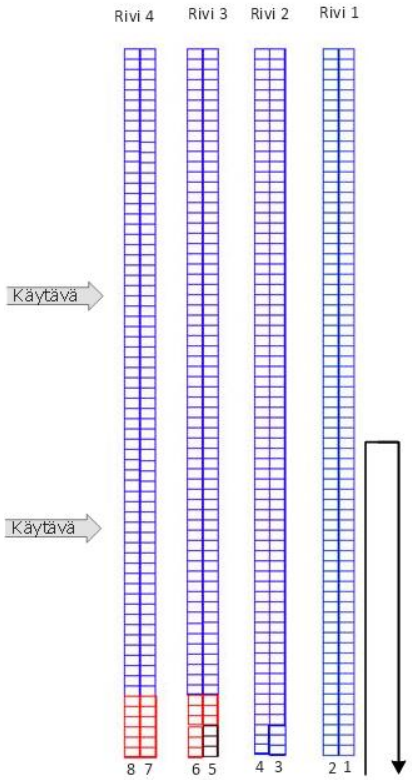
Kuva 5 Tuotteiden kerääminen eri puolilta käytävää

Keräilyn aloituspaikaksi varastonpohjapiirroksella on annettu ensimmäisen hyllyrivin ensimmäinen paikka, josta keräilyjärjestys etenee nuolen osoittamalla tavalla päättyen samaan päätyyn josta keräily aloitettiin (Kuva 6). Annetun aloituspaikan perusteella eri rivit voidaan karkeasti luokitella paremmuusjärjestykseen sen mukaan, kuinka pitkän matkan käyttäjä joutuu kulkemaan lähtöpisteestä päästäkseen poimimaan ko. rivillä sijaitsevan tuotteen. Riviltä yksi on pidempi matka esimerkiksi riville kolme, kuin mitä se olisi siirryttäessä riville kaksi.

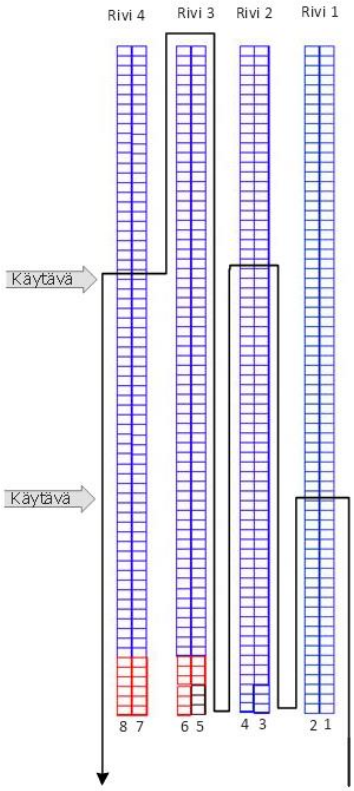


Kuva 6 Keräilyalueen pohjapiirros ja kuljettava reitti

Lähtöpisteestä eteenpäin kuljettu matka määrittää paikkojen paremmuuden huolimatta siitä, että osa rivejä voidaan jättää kulkematta ja jo kuljettuja rivejä voidaan palata takaisin esim. tilanteessa jossa tavarat löytyvät heti rivin alusta (Kuva 7). Keräilyradalta voidaan poistua kunkin hyllyvälin päädystä eikä liikkumiselle pysty- ja vaakasuunnassa ole muita kuin käytävien ja kulkusuunnan osoittamat rajoitteet (Kuva 8).



Kuva 7 Kaikki tuotteet löytyvät jo rivin alusta.



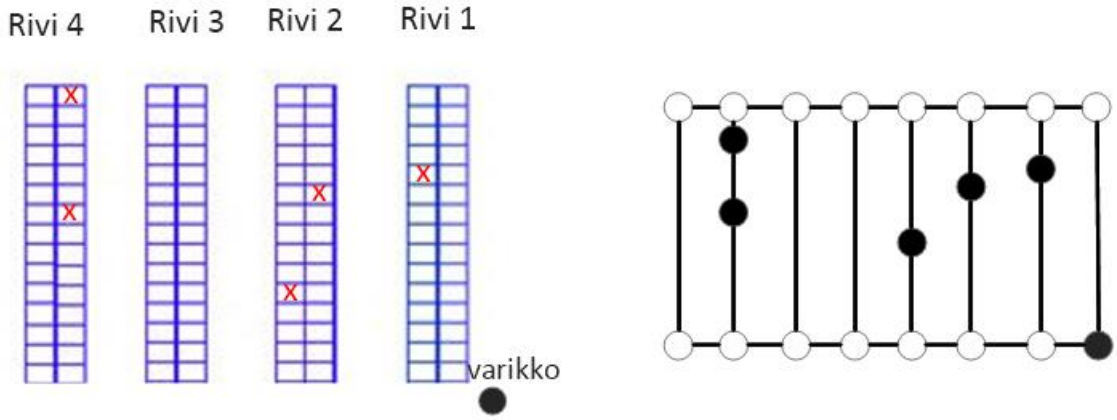
Kuva 8 Rivejä ei ole välttämätöntä kulkea loppuun poikittaiskäytävien ansiosta.

2.6 Algoritmin valinta

Optimoinnin toteutustavan valinta tapahtui heurististen optimointialgoritmien, neuroverkkojen, paritusalgoritmien sekä perinteisen listojen käsittelyn väliltä. Perinteisen listoja käsittelevän algoritmin valintaan vaikutti oleellisesti se, että työssä oli mahdollista tehdä paljon esivalmisteluja jo tunnetulle datalle. Osoittautui myös ettei neuroverkkoja hyödyntävän ratkaisun sinänsä raskasta rakennetta tarvittu, vaikka sillä oletetusti voitaisiin parantaa algoritmin suorituskykyä. Paritusalgoritmiin perustuva toteutustapa puolestaan tukee ajatusta esivalmisteluina tehtävien preferenssilistojen hyödyntämisestä osana ratkaisua.

2.6.1 Kauppamatkustajan ongelma

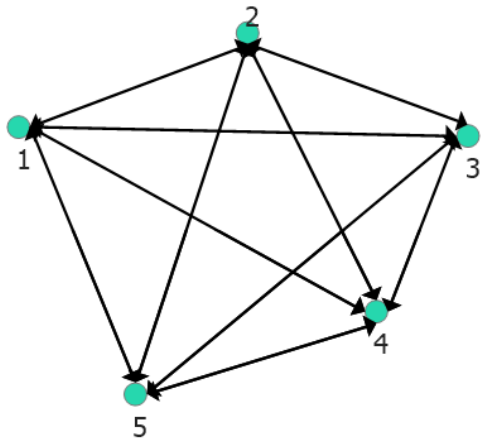
Tarkasteltaessa tilauksen keräämiseen käytetyn ajan optimointia parhaan mahdollisen reitin löytämisen näkökulmasta, voidaan huomata sen olevan eräänlainen erityistapaus tunnetusta kauppamatkustajan ongelmasta. Kauppamatkustajan ongelma saa nimensä seuraavasta tilanteesta. Myyntimies lähtee matkaan hänen omasta kotikaupungistaan tavoitteenaan vierailla tietyssä määrässä kaupunkeja ja palata takaisin kotiin siten, ettei hän vieraile samassa kaupungissa toistamiseen. Hän tietää kaupunkien väliset etäisyydet ja haluaa määrittää niiden vierailujärjestyksen niin, että kokonaisuudessa kuljettu matka on mahdollisimman lyhyt. Kuvauksesta voidaan huomata, että ongelmassa on paljon samankaltaisuuksia verrattuna varastossa tapahtuvaan keräilyyn, jossa keräilypaikkojen voidaan nähdä edustavan vierailtavia kaupunkeja. Täten kerääjän tulee vierailla jokaisessa keräilypaikassa (kaupungissa) josta hänen tilaukseensa tulee tuotteita, löytäen samalla mahdollisimman lyhyen reitin keräykselleen. Keräily alkaa keräilyalueen aloituspaikasta ns. varikolta (kotikaupunki) jonne on myös palattava valmiin keräyksen jälkeen. Kuvassa 9 on nähtävissä esimerkki keräilyalueesta keräysmääräyksen kanssa sekä sen graafi-muotoinen esitystapa. (De Koster, Le-Duc & Roodbergen, 2007.)



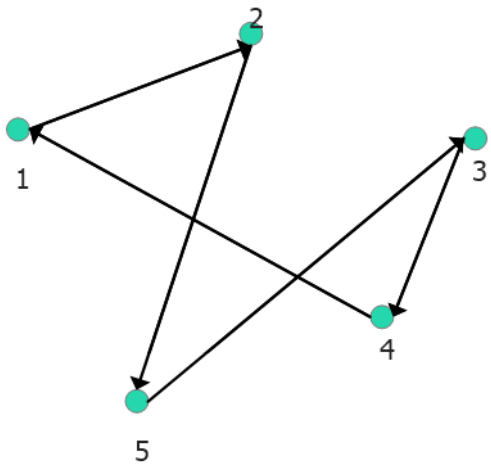
Kuva 9 Keräilytehtävän esitys keräilyalueella (vasen) sekä sen graafi-muotoinen esitys (oikea) (mukaillen De Koster ym., 2007, s.17)

Verrattuna alkuperäiseen kauppamatkustajan ongelmaan on varastossa tapahtuvassa keräilyssä kuitenkin tiettyjä eroavaisuuksia. Esimerkiksi edellä kuvatussa graafissa (Kuva 9) monet esitetyistä solmuista eivät ole pakollisia vierailtavia (valkoiset solmut). Huomioitavaa on, että kuvatut solmut ovat käytännössä pysty- ja vaakakäytävien risteyskohtia. Tilauksen keräilijän on sallittua vierailla niissä, mutta ei välttämätöntä. Mustat solmut edustavat keräilypaikkoja sekä varikkoa josta keräilytyö alkaa ja johon se päättyy; näissä solmuissa on välttämätöntä vierailla. On myös sallittua vierailla keräilypaikoissa sekä varikolla useammin kuin kerran. (De Koster ym., 2007.)

Kauppamatkustajan ongelman näkökulmasta tämä keräilyreitin optimointiongelma olisi käytännössä hyvin vaikea ratkaista ja vaatisi tietokoneelta paljon laskentatehoa. Kyseessä olevassa kombinatorisen optimointiongelman ratkaisussa halutaan löytää mahdollisimman hyvä yhdistelmä keräilypaikkoja joilla kohdefunktion arvo olisi optimaalinen eli pienin mahdollinen kuljettu matka. Koska tällaisissa ongelmissa tulisi käydä läpi kaikki mahdolliset yhdistelmät optimaalisen reitin löytämiseksi, niin usein nämä tehtävät ovat NP-täydellisiä. Koska kauppamatkustajan ongelma on NP-täydellinen, voidaan se ratkaista optimaalisesti pienille tietojoukoille, mutta yleisesti on järkevämpää käyttää erilaisia heuristisia menetelmiä ratkaisun approksimoimiseen (MHI, 2019). Kauppamatkustajan ongelman abstraktio on nähtävissä graafi-esityksenä kuvissa 10 ja 11.



Kuva 10 Täydellinen yhdistetty graafi pienestä kauppamatkustajan ongelmasta (mukaillen MHI,2019, s.3)



Kuva 11 Optimaalinen reitti on vierailla kaikki kaupungit järjestyksessä 1,2,5,3,4 (mukaillen MHI,2019, s.3)

Kauppamatkustajan ongelman ratkaisu ei sellaisenaan toimi ratkaisuna tuotteiden optimaaliseen sijoittelemiseen, vaan pikemminkin auttaa ymmärtämään kuinka arvioida tuotteiden sijoittelun merkitystä lopullisen ratkaisun kannalta. Siten kauppamatkustajan ongelma toimii lähtökohtana tuotteiden sijoittelun paremmuuden arvioimisessa. Ajatuksena on kuvan 9 kaltaisessa tilanteessa löytää paras mahdollinen järjestys johon kuvan punaiset rastit kannattaisi sijoittaa, kun tilauksia on useita erilaisia. Kunkin järjestyksen toimivuutta voidaan mitata ratkaisemalla kauppamatkustajan ongelma ja vertaamalla sitä muihin ratkaisuihin.

Tilauksen keräily voidaan ongelmana asettaa luokkaan Steiner kauppamatkustajan ongelma (Steiner Travelling Salesman Problem) johtuen kahdesta syystä; osassa solmuja ei ole pakko vierailla ja osassa solmuja voidaan vierailla useammin kuin kerran. Tässä tapauksessa se ei ole yleisesti ratkaistavissa polynomisessa ajassa. On kuitenkin osoitettu, että on olemassa algoritmi,

jolla ongelman voi ratkaista lineaarisessa ajassa suhteessa käytävien ja keräilypaikkojen määrään kuvan 9 kaltaisessa varastossa (Ratliff and Rosenthal, 1983). Sekä algoritmin laajennettuna versiona kaikissa varastoissa, jotka voidaan mallintaa sarjana rinnakkaisia graafeja (De Koster & Van der Poort, 1998). (De Koster ym., 2007.)

Ottaen huomioon, että ratkaistavana on pikemminkin joukko kauppamatkustajan ongelmia (joukko asiakastilauksia), on ratkaisun toimivuuden arvioiminen tällä tavoin suoritusajaltaan varsin raskas menetelmä ja siten sen toteuttamiseen tarvitaan suoritusajaltaan kevyempi arviointimenetelmä. Oletus siitä, että ongelma on mallinnettavissa muunnelmaksi kauppamatkustajan ongelmasta tukee osaltaan ajatusta ratkaisusta, joka olisi toteutettavissa perinteisillä algoritmeilla. Algoritmin tulisi ottaa huomioon kohdevaraston pohjapiirroksen asettamat vaatimukset ja muut reunaehdot, jotka vaikuttavat sijoitteluun. Algoritmi voisi järjestää tuotteet paikoille juuri kohdevaraston asettamien erityispiirteiden ja vaatimusten mukaisesti ja samalla sen toteutuksessa voitaisiin hyödyntää jo olemassa olevaa tietoa ympäristöstä ja tuotteista (domain knowledge). Olemassa oleva tieto mahdollistaa yksinkertaisemman ratkaisun toteuttamisen ja tekee toteutuksesta todennäköisesti uniikin.

2.6.2 Heuristiset menetelmät

Kuten jo aiemmin kappaleessa 2.5 todettiin, ei pelkkä reitin optimaalisuus riitä ratkaisemaan koko ongelmaa vaan käytännössä tulee huomioida muitakin keräilyyn vaikuttavia asioita. Esimerkiksi varaston pohjapiirros ja reittien käytännöllisyys, johon vaikuttaa mm. riski ruuhkaantumisesta. Tästä johtuen edellä kuvattu muunnos kauppamatkustajan ongelmasta keräilytehtävänä on usein ratkaistu käyttäen heuristiikkoja. (De Koster ym., 2007.)

Ongelman ratkaisemiseksi on olemassa käytännössä kahdenlaisia heuristiikkoja ja arvioita.

- **Rakentamisen heuristiikat** (Construction Heuristics): Tavoitteena luoda alkuratkaisu, joka toteuttaa kaikki ehdot eli rajoitteet (Puranen, 2012). Perusideana reitin rakentamisessa on valita piste, josta reittiä aletaan rakentaa osa kerrallaan niin, että siitä lopulta muodostuu kaikki pisteet kattava Hamiltonin polku.

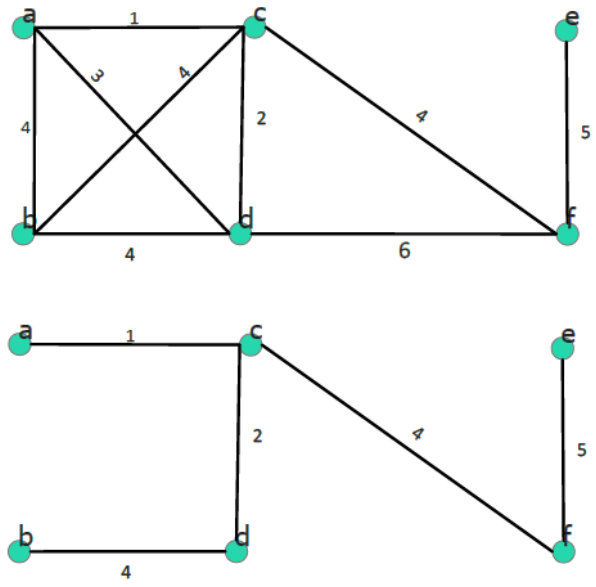
- **Parantamisheuristiikat** (Improvement Heuristics): Aloitetaan ”jostakin” globaalista reitistä ja kokeillaan vaihtaa sitä parempaan niin kauan kuin se on mahdollista. (Universiteit Utrecht, 2019.) Parannusheuristiikat etsivät pienillä muutoksilla saatavia ratkaisuja, jotka olisivat parempia kuin tämänhetkinen (Lokaalihaku). Perusideana on ottaa piste reitiltä ja siirtää se toiselle reitille, mikäli se parantaa ratkaisua. (Puranen, 2012.)

Rakentamisen heuristiikat

Rakentamisen heuristiikkojen perusperiaatteena on koota ratkaisua kasvattamalla reittiä osa kerrallaan, kunnes se kattaa kaikki graafin pisteet. Käytetystä menetelmästä riippuen reitin rakentaminen tapahtuu liittämällä lähtöpisteeseen yksittäisiä pisteitä (esim. Nearest neighborhood heuristics), laajentamalla olemassa olevaa reittiä lisäämällä siihen muita reittejä (esim. Nearest Addition) tai laajentaen jo alustettua puurakennetta (esim. Minimum span tree heuristic). (Chiarandini, 2008.)

Rakentamisen heuristiikat, kuten lähimmän lisäys (Closest Insertion heuristic), alkavat rakentaa reittiä jostakin solmusta, mistä sitten laajenee sen lähimpiin naapureihin solmu kerrallaan. Algoritmista on olemassa monia variantteja kuten kauimman lisäys (Farthest insertion), satunnaisen lisäys (Random insertion) sekä halvimman lisäys (Cheapest insertion). Näistä kaksi ensimmäistä kuuluvat aikavaativuudeltaan luokkaan $O(n^2)$, mutta kolmas vaihtoehto, halvimman lisäys, on laskennallisesti kalliimpi. (Universiteit Utrecht, 2019) Jüngerin ym. (1995) mukaan halvimman lisäys voidaan suorittaa aikaluokassa $O(n^2 \log n)$ tallentamalla jokaiselle ulkoiselle solmulle keko mahdollisien lisäyspisteiden lisäyskustannuksista (Jünger, Reinelt, & Rinaldi, 1995). $O(n^2)$ tilavaativuudesta johtuen tätä ei voida käyttää laajoille joukoille.

Aiemmin esitetyt algoritmit ovat kaikki toimineet hyödyntämättä mitään esitietoja reitistä, mutta reitti voidaan hyvin myös alustaa esimerkiksi puurakenteiden avulla. Voimme esimerkiksi käyttää Primin (1957) tai Kruskalin (1956) algoritmia pienimmän virittävän puun (MST, Minimum Spanning Tree) muodostamiseen. Yleisperiaatteena molemmissa edellä mainituissa algoritmeissa on lisätä puuhun kaaria yksi kerrallaan niin, ettei synny sykliä ja valittuna on aina painoltaan pienin kaari. Algoritmin suoritusta varten graafin ei tarvitse olla *täydellinen*, mutta sen tulee olla *yhtenäinen.* Graafi on yhtenäinen, jos sen mitkä tahansa kaksi eri pistettä ovat yhdistetyt ja täydellinen, mikäli siinä on kaikki mahdolliset kaaret (Ruohonen, 2013). Näin lopulta löytyy pienin virittävä puu, jota voidaan käyttää perustana seuraavalle algoritmille (Kuva 12). Seuraava algoritmi voi siten käyttää puuta hyödyksi Hamiltonin kierroksen muodostamisessa. Käytettäessä Primin algoritmia, voidaan Hamiltonin kierros muodostaa samanaikaisesti virittävän puun laskennan yhteydessä (Jünger ym., 1995, s.17-18).

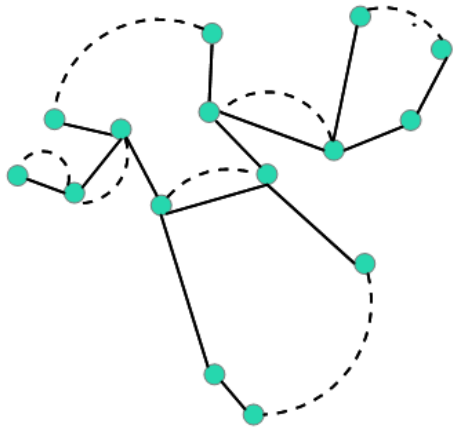


Kuva 12 Painollinen graafi ja sen eräs pienin virittävä puu (mukaillen Kivinen 2008, s.3)

Hamiltonin kierros tarkoittaa reittiä, joka käy suuntaamattoman graafin kaikkien solmujen kautta ja palaa lopulta lähtöpisteeseensä. Tässä ongelmassa on tavoitteena Hamiltonin kierros jonka c-pituus (kaarien pituuksien summa) on pienin mahdollinen, kuvastaen lyhintä mahdollista reittiä. (Jünger ym., 1995.) Monissa reitin muodostavissa algoritmeissa puurakenteen ideana on rajoittaa tutkittavien reittien määrää siten, että kaikki reitit perustuvat pienimmästä virittävästä puusta luotaviin reitteihin.

On olemassa myös hienostuneempia algoritmeja kuten Christofides-algoritmi, joka hyödyntää myös virittävän puun ulkopuolella olevia reittejä laskien vähimmäispainon parhaan vastaavuuden (minimum weight perfect matching) puun parittoman-asteen solmuille ja lisää sen uudeksi reitiksi puuhun. Aikavaativuudeltaan tämä on kuitenkin merkittävästi edellisiä ratkaisuja kalliimpi, ollen kustannukseltaan kuutiollinen $O(n^3)$. Christofiden pahimmillaan kuutiollinen suoritusaika johtuu siitä, että viritetyllä puulla voi olla O(n) lehteä. (Jünger ym., 1995.)

Kuva 13 havainnollistaa heuristiikan periaatetta esittäen kiinteillä viivoilla viritetyn puun kaaret sekä katkoviivoilla parhaat vastaavuudet puun parittoman asteen solmuissa. Kahden kaarijoukon liitto muodostaa reitin (Jünger ym., 1995).



Kuva 13 Christofiden heuristiikka (mukaillen Jünger ym. 1995, s.18)

Muita rakentamisen heuristiikkoihin kuuluvia algoritmeja ovat mm. konveksin peitteen heuristiikka (Convex Hull Heuristic) sekä talletus algoritmi (Saving Algorithm), jotka molemmat kuuluvat aikavaativuudeltaan luokkaan $O(n^3)$. Konveksin peitteen heuristiikka on aikavaativuudeltaan kuutiollinen siinä tapauksessa, että reitti on alustettu halvimman lisäys heuristiikalla, mutta neliöllinen ($O(n^2)$) mikäli käytetään lähimmän naapurin, kauimman naapurin tai lisäys heuristiikkaa. Talletusheuristiikka on alun perin kehitetty ajoneuvojen reititysongelmiin, mutta sitä voidaan hyödyntää myös kauppamatkustajan ongelman kaltaisiin tehtäviin. Tässä tapauksessa ajoneuvolla voidaan ajatella olevan rajaton käyttökapasiteetti. Heuristiikan perusajatuksena on yhdistellä alireittejä peräkkäin siten, että ne lopulta toteuttavat Hamiltonin kierroksen. (Jünger ym., 1995.)

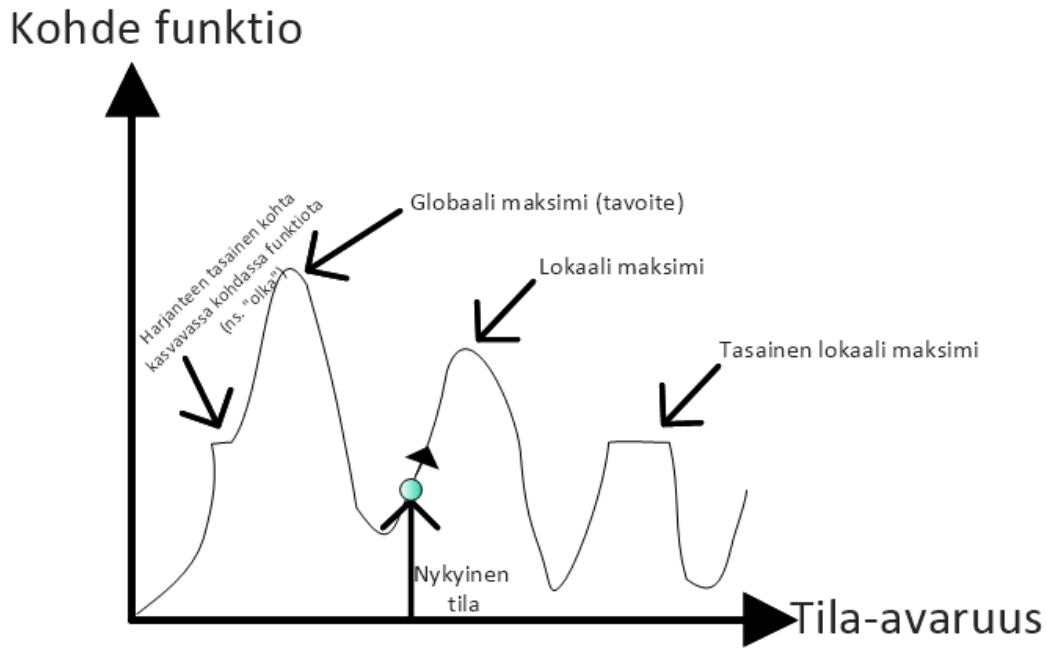
Utrechtin yliopiston materiaali tiivistää Jüngerin ym. (1995) toteuttamien testien suorituskykyvertailut eri rakentavien heuristiikkojen välillä, kun tarkasteltavana on 105-2392 solmun kauppamatkustajan ongelma. Yleiskatsauksen mukaan monet algoritmeista toimivat hyvin päästen keskimääräisillä suorituskerroilla lähelle optimaalista tulosta. Huonoimpiin suoriutujiin lukeutuva lähimmän naapurin menetelmä suoriutui ongelmasta jääden 24% optimista. Vastaavasti parhaimpiin lukeutuva talletus menetelmä saatiin testeissä toimimaan nopeasti ja tehokkaasti, jääden vain 10% optimista. Yhtä hyvään suorituskykyyn pääsivät ainoastaan Christofides sekä kauimman lisäys algoritmit. (Universiteit Utrecht, 2019.) Jüngerin ym. (1995) mukaan n. 10% jääminen optimista on paras tunnettu huonoimman tapauksen raja kauppamatkustajan ongelman ratkaisemiseen heuristiikalla, sillä kokeissa tuotetut ratkaisut harvoin johtavat tätä parempaan laatuun.

Parantamisheuristiikat

Edellä esitetyillä heuristiikoilla saavutetaan Jüngerin ym. (1995) mukaan vain kohtuullisen laadukkaita ratkaisuja eivätkä ne ole yleisesti tyydyttäviä, vaikka niitä voidaan hyödyntää joissain sovelluskohteissa. Seuraavassa kappaleessa tullaan käsittelemään eräitä tekniikoita, joilla tuloksia voidaan yrittää parantaa.

Parantamisheuristiikkoja käytetään, koska aiemmin toteutetussa alkuratkaisussa on usein parantamisen varaa. Parantamisheuristiikoilla yritetään saada aikaan tämänhetkistä parempi ratkaisu tekemällä alkuperäiseen pieniä muutoksia. Yksinkertainen parantamisheuristiikka toimii esimerkiksi ottaen pisteen alkuratkaisussa luodulta reitiltä ja siirtäen sen toiselle reitille, jos ratkaisu näin tekemällä paranee. (Puranen, 2012.) Tätä prosessia voidaan toistaa esimerkiksi ns. kukkulalle kapuamisen menetelmällä (Hill climbing), jota on kutsuttu myös ahnaaksi lokaaliksi hauksi (greedy local search) sen kuuluessa lokaaleihin hakualgoritmeihin ja toimien näkemättä mitään muita kuin sen välittömiä naapurisolmuja (JavaTPoint, 2020).

Kukkulalle kapuamista on käytetty yleensä silloin, jos saatavilla on hyvin toimiva heuristiikka. Se on yksinkertainen toteuttaa, sillä tässä algoritmissa ei tarvitse ylläpitää ja käsitellä hakupuita tai graafeja, vaan tarkasteltavana on ainoastaan suorituksen nykyinen tila (ei muista aiempia jo käytyjä tiloja). Ongelmana lokaaleissa hakumenetelmissä kuten kukkulalle kapuamisessa voivat olla mm. juuttuminen lokaaleihin optimeihin, harjanteisiin tai tasanteisiin (Kuva 14). Ratkaisuna lokaaleista optimeista pois pääsemiseen voidaan käyttää mm. simuloitua jäähdyttämistä (Simulated Annealing), askelten pituuden säätämistä tai yrittää muulla tavalla satunnaistaa nykytilan sijaintia suorituskertojen välillä. (JavaTPoint, 2020; Henderson, Jacobson, & Johnson, 2006.)



Kuva 4 Esimerkki kukkulalle kapuamisen tila-aika kaaviosta (mukaillen JavaTPoint, 2020)

Optimoitaessa reittiä lokaalin haun siirrot reitiltä toiselle päätyvät hyvin usein lokaaliin optimiin, jolloin näyttää siltä, ettei parannettavaa enää olisi. Tämä johtuu siitä, että sokeasti löydetty ”vuorenhuippu” näyttää korkeimmalta mahdolliselta, sen lähimpien naapuriarvojen ollessa huonompia. Tässä tilanteessa ei kuitenkaan ole takeita siitä, että ollaan löydetty paras tai edes hyvä ratkaisu. Jotta päästäisiin pois tai pystyttäisiin välttämään lokaaleja optimeja sekä harjanteita, on heuristiikkojen parantamiseen kehitetty ns. meta-heuristiikkoja. Metaheuristiikka pyrkii ohjaamaan lokaalin haun heuristiikkojen toimintaa eri keinoin kuten: valitsemalla millaisia siirtoja tehdään ja arvioimalla siirtojen hyvyyttä, muuttamalla tavoitefunktiota sopivammaksi tai muuttamalla ratkaisua (Puranen, 2012).

Esimerkki metaheuristisesta hakumenetelmästä on Pierre Hansenin ja Nenad Mladenović:n (Hansen, Mladenović & Perez, 2010) kehittämä usean naapuruston haku (Variable neighborhood search, VNS), joka kehitettiin ratkaisemaan kombinatorisia ja globaaleja optimointiongelmia. Menetelmä tutkii nykyisen ratkaisun kaukaisia naapurustoja ja siirtyy sieltä uuteen vain sen parantaessa tulosta. Lokaalia hakumetodia (local search) käytetään toistuvasti, jotta lopulta päästäisiin naapurustosta lokaaliin optimiin. Usean naapuruston haussa naapurustolla tarkoitetaan hakuoperaattorin läpi käymien vaihtoehtojen joukkoa. Purasen (2012) mukaan tällaisissa lokaaleissa hauissa voidaan käyttää useita operaattoreita, jotka tekevät eri siirtoja (esim. ”relocate”, ”exchange”, ”2-opt” ja ”3-opt” operaattorit). Ajatuksena on käyttää jotain toista operaattoria aina, kun nykyinen jää jumiin lokaaliin optimiin, sillä näistä kukin operaattori näkee

eri ”vuoriston”. Uusien operaattoreiden mukaan tuominen auttaa pääsemään pois lokaaleista optimeista, mutta samalla lisää suoritukseen tarvittavaa laskenta-aikaa. Muita tunnettuja metaheuristiikkoja ovat esim. ohjattu lokaali haku, simuloitu jäähdytys ja iteroitu lokaali haku. (Puranen, 2012.)

Heuristisen algoritmien kehittämisestä on olemassa laajasti kirjallisuutta ja useita eri lähestymistapoja saman tyyppisten kombinatoristen optimointiongelmien ratkaisuun. Kaiken kaikkiaan menetelmät tarjoavat laajan valikoiman eri ratkaisuja, joilla kauppamatkustajan ongelma voidaan ratkaista polynomisessa ajassa hyödyntäen perinteistä algoritmiikkaa. Täytyy kuitenkin huomioida, että heuristiikat ovat luonteeltaan epätarkkoja eivätkä ne siten aina takaa optimiratkaisua, vaikka voivat päästä hyvin lähelle sitä ja samalla suoriutua paljon pienemmällä laskentakapasiteetilla.

2.6.3 Neuroverkot

Viimeaikaisen syvien neuroverkkojen kehittymisen myötä on pystytty ratkaisemaan monia vaikeita ongelmia, jotka ovat olleet aiemmin tavoittamattomissa tai vaatineet koneilta hyvin suurta laskentatehoa. Tästä syystä on varsin perusteltua tutkia millaisia ratkaisuja kehittyneet neuroverkot voivat tarjota keräilyreitin tai abstraktimmin kauppamatkustajan ongelman ratkaisemiseksi.

Käytännössä neuroverkolla tarkoitetaan joukkoa neuroneita (tiedonkäsittely-yksiköitä), jotka ovat kytkettynä toisiinsa eräänlaisella kytkentäverkolla. Pohjimmiltaan tämä lähestymistapa yrittää jäljitellä ihmisen aivojen toimintatapaa. Neuronin saamasta syötteestä riippuen laskee se tietynlaisen tuloksen, joka etenee muihin neuroneihin. Erilaisissa malleissa tarkastellaan neuronin aktivoivia tiloja, neuronien laskemien tulosten määrityksiä sekä signaalien etenemistä verkossa. Mallien tavoitteena on jonkinlaisen oppimismekanismin toteuttaminen. Neuroverkon laskema tulos näkyy joko nimenomaisesti tuloksena tai se annetaan neuronien tilan avulla. (Jünger ym., 1995, s. 44.) Neuroverkon tulee sisältää vähintään yksi syöteneuroni ja yksi tulosneuroni. Tulosneuronit muodostavat tuloskerroksen (output layer) ja syöteneuronit syötekerroksen (input layer). Neuroverkko tuottaa tulokseksi tuloskerroksen tai ennustetun arvon (tai useita arvoja jos kyseessä on useita tulosluokkia). (Ognjanovski, 2019.) Tarvittavien syöte- ja tulosneuronien määrä valitaan ratkaistavan ongelman mukaan esimerkiksi ennustettavien luokkien perusteella.

Esimerkkitapauksena reititysongelman ratkaisusta neuroverkkojen avulla toimii Sebastian Heinzin, Roland Vollgrafin ja Calvin Sewardin käyttämä syvä neuroverkko ohjausoperaatioihin

Zalandon muotivaatevarastoissa. He ovat kouluttaneet neuroverkon arvioimaan lyhyimmän mahdollisen reitin pituuden, joka vierailee kaikissa annetuissa varastopaikoissa. Vuonna 2015 julkaistussa artikkelissa Seward (2015) esittää, kuinka heidän rakentamaansa neuroverkkoa on käytetty nopeuttamaan prosessoinnin pullonkauloja, mikä puolestaan antaa mahdollisuuden jakaa työt tehokkaammin työntekijöiden kesken.

Kuten kohdevarastossamme, myös tässä esimerkissä keräily alkaa työntekijän lähtiessä liikkeelle varikolta keräilyrataa pitkin edeten. Keräilijä kulkee läpi varaston asettaen tuotteita tyhjään kärryyn ja palaa takaisin kärryn ollessa täysi. Joten pohjimmiltaan tämäkin on erikoistapaus aiemmin esitetystä kauppamatkustajan ongelmasta (TSP). Vaikkakin TSP on yleisesti NP-vaikea ongelma, voidaan se ratkaista lineaarisessa ajassa suhteessa käytävien määrään, hyödyntämällä köysitikasmaista varaston pohjapiirrosta (Kuva 1). (Seward, 2015.)

Ongelman ratkaisuksi tutkijat kehittivät "Optimal Cart Pick" algoritmin (OCaPi), joka ei pelkästään löydä keräilijälle optimaalista keräilyreittiä vaan lisäksi myös keräilijän kärryn liikkeille. Kuten ostosten tekijä supermarketissa, voi joskus olla järkevää jättää kärry odottamaan poikkikäytävälle ja keräillä tuotteita sen verran mitä pystyy kantamaan ilman kärryä. Tämä algoritmi mahdollistaa S-muotoisesta reititysheuristiikasta luopumisen ja keräilijän sekä kärryn reitin optimoinnin. OCaPi on tässä tapauksessa ainoastaan osa ratkaisua, sillä ongelman ratkaisun nopeuttamiseksi, on algoritmi muunnettu ns. mustan laatikon ongelmaksi (black box problem) neuroverkon käyttämistä varten. Neuroverkkoa hyödyntäen ongelmanratkaisua voidaan nopeuttaa, jonka lisäksi verkon skaalautuvuus mahdollistaa laskentaan käytettävän joko prosessori- tai grafiikkaprosessori (CPU/GPU) arkkitehtuuria. Lopullisen koulutetun neuroverkon todettiin suoriutuvan tehtävästä erittäin nopeasti. (Seward, 2015.)

Huomioitavaa on, että tässäkin tapauksessa ongelma oli ratkaistu käyttäen perustana perinteistä algoritmiikkaa (OCaPi), jota sittemmin nopeutettiin neuroverkkoja hyödyntämällä. Täten ei ole selvää, että neuroverkoista olisi hyötyä tämän ongelman ratkaisemiseen muilta kuin suorituskyvyn osalta, mikäli se osoittautuisi valitun ratkaisun kohdalla ongelmaksi. Perinteisessä algoritmiikassa laskentateho voi muodostua ongelmaksi isolla aineistolla. Toisaalta neuroverkkojen käytössä voisi lähestymistavaksi ottaa ajatuksen siitä, että johonkin saatavilla olevaan tietojoukkoon (esim. toimituksissa esiintyneet tuotteet) perustuen pyrittäisiin ennustamaan jokaiselle paikalle sille optimaalinen tuote. Tästäkin näkökulmasta neuroverkko voidaan nähdä liian raskaana ratkaisuna toteuttaa suhteessa ongelman kompleksisuuteen.

2.6.4 Rivijärjestyksen optimointi

Eräs heuristinen toteutustapa keräilyreitin optimointiin on rivi järjestyksen optimointi (Line Sequence Optimisation, LSO). LSO laskee optimoidun rivijärjestyksen annetulle keräykselle, jotta kuljettu matka olisi mahdollisimman lyhyt. Moellerin (2011) tutkimuksen mukaan eräässä asiakastapauksessa on kvantitatiivisesti arvioiden saavutettu kokonaisparannusta kaikille päivän keräyksille 7,4 %. Suurin osa parannuksesta voitiin osoittaa yhdelle erityiselle varastoalueelle, jolla LSO osoitti useissa tapauksissa vahvoja potentiaalisia parannuksia niiden ollessa yli 27%.

Varastonhallintajärjestelmän (Warehouse Management Systems, WMS) avulla hallitaan ja ohjataan varaston tärkeimpiä toimintoja, kuten materiaalien ja tuotteiden siirtelyä, vastaanottoa, hyllytystä, keräilyä, pakkausta ja toimitusta. Varastonhallintajärjestelmä liittyy yleensä koko yrityksen toiminnanohjausjärjestelmään, joko osana sitä tai integroituna eri rajapintojen kautta. Järjestelmän keskeisimpiin ominaisuuksiin kuuluu myös tuotteiden tarkan varastopaikan ja sijainnin määritys. (Logistiikan maailma, 2020b.)

Ajatuksena on, että yleisesti varastonhallintajärjestelmä voi järjestää keräyksen rivit järjestelmässä olevien paikkojen nimen mukaiseen järjestykseen, joka siten voi näyttää loogiselta. Tästä käytetään nimeä järjestelmän järjestys (”System sequence”, SYS). Oletetaan, että keräilijä näkee annetun järjestelmä järjestyksen päätelaitteellaan ja hän voi muuttaa sitä haluamaansa järjestykseen reitin optimoimiseksi. Tätä muutettua reittiä kutsutaan nimellä keräilijän järjestys (”Picker sequence”, PIC). Kun otetaan huomioon etäisyysmatriisi ja optimointirutiinin käyttö, ollaan lähtökohtaisesti ratkaisemassa kauppamiehen ongelmaa siten, että rivit järjestetään järjestykseen jossa kokonaismatka-aika on mahdollisimman lyhyt. Tätä järjestystä kutsutaan nimellä rivijärjestyksen optimointi järjestys (”Line Sequence Optimization sequence”, LSO). (Moeller, 2011.)

SYS-rivisekvenssin järjestelmällinen määrittäminen edellyttää, että keräilyalueen lohkojen määrittely ja keräilypaikkojen osoittaminen on tehty jokaiselle hyllylle etukäteen ja sen on noudatettava tilaus-strategiaa (Moeller, 2011). Kohdevaraston tapauksessa tämä on tehty osittain valmiiksi suoraan varastonhallintajärjestelmässä. Lisäksi tarvittavaa parametrisointia voidaan tehdä jo esivalmisteluvaiheessa ennen optimointia, muun muassa paikkojen eri ominaisuuksien huomioimiseksi. Moellerin (2011) tutkimuksen mukaan LSO on lähtökohtaisesti tarkoitettu integroitavaksi lisätoimintoina toimittajan olemassa olevaan varastonhallintajärjestelmään, jolloin keräystehtävien järjestys olisi aina optimoitu erikseen kunkin tehtävän kohdalla. Mutta toisin kuin systemaattisesti toimiva reititysheuristiikka (esim. S-muoto), LSO:n luomat sekvenssit

saattavat näyttää keräilijästä epäloogiselta ja niiden toimivuutta pitää arvioida laadullisesti riippuen tapauksesta.

2.6.5 Paritusalgoritmit

Keräilyreitin optimoinnin ratkaisuksi voisi myös esittää ns. paritusalgoritmeja (matching algorithms), joilla tuotteet ja varastopaikat voidaan parittaa keskenään optimaalisesti. Eräs optimointiongelman ratkaisuun mahdollisesti sopiva algoritmi on Gale-Shapley-algoritmi, joka tunnetaan myös nimellä ”deferred acceptance algorithm”. Kyseinen algoritmi tuottaa tulokseksi niin sanotun stabiilin parituksen (stable matching) (Gale & Shapley, 1962). Esivalmisteluina algoritmin käyttäminen vaatii sen, että tuotteet on laitettu preferenssijärjestykseen niille sopivimman paikan mukaisesti ja varastopaikat on laitettu vastaavasti preferenssijärjestykseen niille sopivimman tuotteen mukaisesti. Tämän jälkeen paritusalgoritmilla tuotteet ja varastopaikat voidaan saattaa yhteen keskitetysti. Preferenssijärjestyksen laatimiseksi voisimme käyttää esimerkiksi tuotteille ja paikoille laskettuja painoarvoja siten, että painoarvoltaan suurin vapaana oleva paikka valitsee painoarvoltaan suurimman vapaana olevan tuotteen.

Tässä tapauksessa kyseessä ei ole yksittäisen tilauksen keräilyreitinoptimointi vaan optimaalisten paikkojen löytäminen aktiivisille tuotteille niiden preferenssijärjestyksen mukaisesti. Tavoitteena on, että tuotteiden tehokkaampi järjestäminen niiden kysynnän mukaiseen preferenssijärjestykseen lyhentäisi keräystyössä kuljettua matkaa.

Alkuperäisessä muodossaan Gale-Shapley-algoritmi takaa, että jokainen paikka saa tuotteen. Algoritmin suorituksen lopuksi, ei voi olla tuotetta ja paikkaa joista molemmat olisivat sijoittamatta, sillä paikan tulee olla täytetty jollakin nimikkeellä viimeistään silloin, kun jokaista nimikettä on tarjottu paikalle (mikäli välttämätöntä). Jos tuotteita on saman verran kuin paikkoja, on tällöin myös jokainen tuote sijoitettu johonkin paikkaan. Käytännön toteutuksessa tuotteita voi kuitenkin olla huomattavasti enemmän kuin paikkoja, joten tuotteista tulee parittaa vain parhaiten soveltuvat. Tämä voitaisiin toteuttaa esivalmisteluna, karsimalla tuotteiden määrää niiden painoarvon mukaisesti vastaamaan olemassa olevien paikkojen lukumäärää.

Tuotteiden ja paikkojen parittaminen on stabiili. Sanotaan, että tuote A ja paikka B ovat molemmat paritettuja, mutta eivät toistensa kanssa. Algoritmin suorituksen jälkeen ei ole mahdollista, että tuote A suosisi paikkaa B ja paikka B tuotetta A mieluummin, kuin niiden nykyisiä pareja. Jos paikka B suosisi tuotetta A ennemmin, kuin sen nykyistä tuotetta, olisi paikalle B pitänyt ensin kokeilla sijoittaa tuote A nykyisen tuotteen sijasta. Jos tuote A sopi

paikalle B, mutta ei ole lopussa sijoitettuna siihen, tarkoittaa se tuotteen löytäneen paremman paikan, eikä sen siten kuuluisi olla paikalla B. Jos tuote A ei sopinut paikalle B oli se jo valmiiksi paremmalla paikalla kuin B. (Gale & Shapley, 1962.)

Parituksessa aina toinen osapuoli on ns. puolueellinen saaden valita ensin. Käytännön toteutuksessa algoritmin suoritus tulisi olla puolueellinen varastopaikkojen suhteen tuottaen paikkojen kannalta parhaan stabiilin parituksen. Mikäli tuotteen annettaisiin valita ensin, olisi lopputuloksena tuotteiden kannalta paras stabiili paritus. Algoritmin suoritusta toistetaan niin kauan, kuin saatavilla on yhtään vapaata varastopaikkaa. Algoritmin suorituksen aikavaativuus sen perusmuodossaan on $O(n^2)$, missä n on tuotteiden tai varastopaikkojen lukumäärä riippuen siitä kumpi saa valita ensin (Kazuo & Shuichi, 2008).

2.6.6 Toteutuksen lähtökohta

Kuten aiemmissa kappaleissa todetaan, on kyseinen ongelma lähtökohtaisesti hyvin lähellä kauppamatkustajan ongelmaa, joka on optimoinnin kannalta varsin vaikeasti ratkaistava. Ongelman abstraktion ratkaisemiseksi optimaalisesti löytyy monia menetelmiä, mutta ne kaikki edellyttävät käytännön mallintamista matemaattisesti formaaliin muotoon, mikä harvoin on mahdollista, saati tarpeellista ratkaisun korkean laskentakustannuksen takia. Heuristiset menetelmät tarjoavat ongelmaan ratkaisun, jolla voidaan löytää vähintäänkin tyydyttävä ratkaisu polynomisessa ajassa, hyödyntäen perinteistä algoritmiikkaa. Siitä huolimatta on olemassa ympäristöjä, joissa reititysheuristiikkoja ei voida käyttää. Esimerkkinä tällaisista olosuhteista Moeller (2011) mainitsee tutkimuksessaan mm. ei-suorakaiteenmuotoisen geometrian, epähomogeeniset telinepalkkirakenteet ja poikkikäytävät eri paikoissa. Kohdevarastossa poikkikäytävät sijaitsevat epätasaisella välillä toisistaan mikä vaikeuttaa optimaalisen reitin laskemista.

Haastavista olosuhteista johtuen täysin optimaalisen reitin tai keräilypaikkojen järjestyksen etsiminen kohdevaraston tarpeisiin ei todennäköisesti ole järkevää. Optimaalinen reitti on myös mitä todennäköisimmin keräilijän kannalta epälooginen ja käytännössä vaikeasti hahmotettava verrattuna nykyiseen järjestelmällisesti etenevään reittiin. Lisäksi voidaan todeta, että varaston pohjapiirroksella ja tuotteiden asettelulla on suuri merkitys siihen, miten reitti määräytyy sitä optimoitaessa. Kauppamatkustajan ongelman ja sen mahdolliset ratkaisut luovat kuitenkin perustan ratkaisulle.

Algoritmin valinnassa kauppamatkustajan ongelma on muutettu ajatukseltaan päinvastaiseksi, eli mitä jos kaupunkien sijaintia voidaan muuttaa kauppamiehen matkustamisen sijasta? Alkuperäisessä ongelmanasettelussa ajatus on absurdi, mutta muunnettuna kauppamatka keräilyreitiksi varastossa, se tarkoittaa käytännössä varastointipaikkojen siirtelemistä kaupunkien sijasta. Tästä lähtökohdasta ongelman ratkaisu käänteisesti ei enää vaikuta mahdottomalta ja paikkojen muuttamisen volyymiperusteiseksi voidaan tutkitusti perustella olevan tehokas tapa lisätä työn tuottavuutta. On kuitenkin huomioitava, että jokainen keräiltävä tilaus on erilainen ja siten ei ole selvää, että se järjestys, joka toimii ensimmäiselle tilaukselle toimisi hyvin myös seuraavalle tilaukselle. Tästä huomiosta johtuen on tarkasteltavana pikemminkin joukko käänteisiä kauppamatkustajan ongelmia, joiden ratkaisun suoritusta yritetään parantaa muuttamalla ympäristöä eli tuotteiden sijoittelua keräilyalueella.

Tuotteiden uudelleensijoittelussa ajatuksena on järjestää tuotteet niiden kysynnän (volyymin) perusteella, niin että useimmin tilattu tuote löytyy todennäköisemmin läheltä ja siten ajomatkat tuotteiden välillä väistämättä lyhenevät. Tällä voidaan olettaa saavan vastaavaa hyötyä mitä LSO tarjoaa järjestäessään keräyksen rivit optimaaliseen järjestykseen, jotta kuljettava reitti olisi mahdollisimman lyhyt. Mitä useammin tuote esiintyy tilauksilla, sitä useamman osaongelman ratkaisua voidaan parantaa siirtämällä tuote paremmalle paikalle varastossa.

VALITTU RATKAISUTAPA

Lopulliseksi toteutustavaksi on valittu volyymiperusteisen varastopaikkojen asettelun toteuttaminen sekä sen automaattinen laskeminen perinteistä ohjelmointia hyödyntäen. Valinta perustuu siihen, että volyymipohjaisen sijoittelun on todettu olevan kaikkein tehokkain varastoasettelu (HUB Logistics, 2009). Saatavilla olevaan dataan perustuen, voidaan asettelun toteuttamisen olettaa olevan mahdollista perinteisellä ohjelmoinnilla. Toteutuksen toimivuuden arviointiin voidaan soveltaa heuristisia menetelmiä, joilla simuloidaan liikkumiseen käytettyä matkaa nykyisessä ympäristössä ja verrataan sitä optimoituun versioon. Tällä tavalla voidaan arvioida algoritmin toimintaa juuri kyseisessä varastoympäristössä.

Koska optimointi tarkoittaa systemaattisia tapoja taata parhaan mahdollisen ratkaisun tai päätöksen löytyminen (Miettinen, 2009) ei tämän tutkimuksen esittämää ratkaisua voida pitää optimina. Kyse on kuitenkin optimointitehtävästä, jossa minimoidaan tarkasteltavaa suuretta eli tavoitetta. Tässä tapauksessa tarkasteltava suure on liikuttu matka keräysmääräyksen aikana. Lopullista ratkaisua arvioidaan mittaamalla, kuinka paljon keräystyöhön käytetty matka on keskimäärin lyhentynyt simuloitaessa keräilyä useilla asiakastilauksilla.

Algoritmi pseudokoodina:

```
Varastopaikat = lista varastopaikoista P001 – P999
Nimikkeet = lista nimikkeistä 110000 - 999999
#Järjestä varastopaikat pikalajittelulla niiden painotuksen mukaiseen ei kasvavaan suuruusjärjestykseen.
jarjestetyt _varastopaikat = quicksort(varastopaikat, painoarvo, DESC)
#Järjestä nimikkeet pikalajittelulla niiden painotuksen mukaiseen ei kasvavaan suuruusjärjestykseen.
jarjestetyt_nimikkeet = quicksort(nimikkeet, painoarvo, DESC)
optimaaliset = [] #luo tyhjä lista optimaalisille nimikkeille ja niiden paikoille.
For i in (jarjestetyt_varastopaikat): #käydään läpi jokainen varastopaikka.
    j = 0
    While True: #käydään nimikkeitä läpi, kunnes löydetään paikalle sopiva nimike.
        koko = jarjestetyt_nimikkeet[j,'koko'] #j:nnen alkion attribuutti 'koko'
        if (koko == jarjestetyt _varastopaikat[i,'koko']):
            #lisää optimaalisten paikkojen listaan nimike j paikalle i
            optimaaliset.add(jarjestetyt_varastopaikat[i], jarjestetyt_nimikkeet[j])
            #poista nimike käsiteltävistä nimikkeistä.
            jarjestetyt_nimikkeet.remove(jarjestetyt_nimikkeet[j])
            break
        elif j == len(jarjestetyt _nimikkeet): #lopetetaan jos kaikki nimikkeet on testattu ilman vastaavuutta.
            #merkataan paikka tyhjäksi, ei sopivaa nimikettä
            optimaaliset.add(jarjestetyt_varastopaikat[i], "TYHJA")
            break
        else:  #jatketaan suoritusta tutkimalla seuraava nimike.
            j += 1
#lopuksi palautetaan lista optimaalisista nimikkeistä varastopaikoittain
Return optimaaliset
```

3 Varastopaikkojen käytön optimointiprojekti

Projektin tarkoituksena on toteuttaa lajittelualgoritmi, joka järjestää saatavilla olevat tuotteet annettujen parametrien mukaisesti mahdollisimman hyville varastopaikoille. Kaikki käsiteltävät tuotteet ovat alkoholijuomia, joten niiden yleiset piirteet ovat varsin yhteneväisiä keskenään. Tuoteportfolio on kokonaisuudessaan suhteellisen kapea ja helposti kategorisoitavissa. Tuotteiden joukosta voidaan tunnistaa omiksi kategorioiksi esimerkiksi puna- ja valkoviinit, hanapakkaukset, matala-alkoholiset juomat, ym. tuotteiden sijoitteluun mahdollisesti vaikuttavat tekijät. Projekti etenee nykytilan kartoittamisesta projektin laajuuden määrittelyyn, jonka jälkeen käydään läpi projektin toimitukset ja sen tavoitetila.

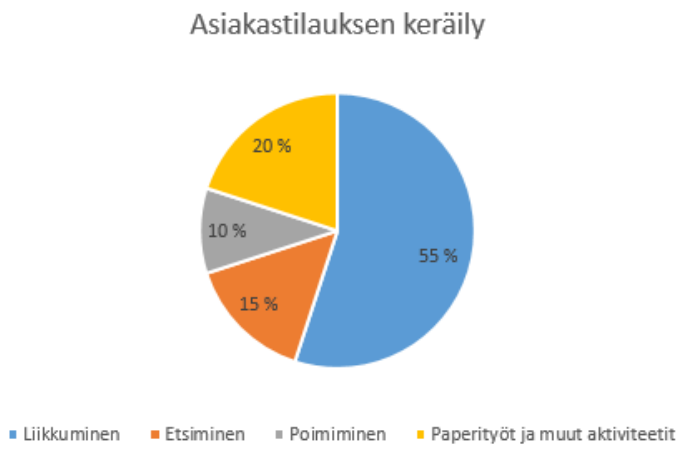
3.1 Nykytilan kartoitus

Projektin valmistelu alkoi nykytilan kartoittamisella, jossa selvitettiin kuinka nykyinen keräilyprosessi ja tuotteiden sijoittelu on toteutettu sekä mitkä ovat sen mahdolliset ongelmakohdat. Aluksi varaston toimintaa on tutkittu kokonaisuutena, jotta voidaan ymmärtää, mikä tehostamisen kohteena olevan keräilytoiminnan rooli on kokonaisuuden kannalta. Tämä on tärkeää arvioitaessa millaisia projektin mahdolliset vaikutukset voivat olla kokonaisuuden kannalta.

Tavoitteena on kehittää kohdeyrityksen keräilytoiminnan tehokkuutta, jolloin pystytään edelleen kehittämään varaston kokonaistehokkuutta. Keräilyn ollessa yksi varaston eniten aikaa vievistä prosesseista, voidaan sen tehokkuutta nostamalla parantaa koko tavaranlähetysprosessin tehokkuutta huomattavasti (Luotonen, 2017). Kuten useimmissa varastoissa on tässäkin tapauksessa tilausten kerääminen kaikkein tärkein prosessi, koska se vaatii eniten työvoimaa ja se määrittää toimitusketjun loppuasiakkaiden kokeman palvelutason (Bartholdi & Hackman, 2014). Keräilystä koituvat kustannukset vaihtelevat riippuen varastosta sekä siitä kuinka nykyaikaiset laitteet ja hyllyjärjestelmät ovat käytössä. Tutkimusten mukaan tilausten kerääminen kattaa arviosta riippuen jopa 50-70% (De Koster ym., 2007) kaikista varaston operatiivisista kustannuksista.

Bartholdin ja Hackmanin (2014) mukaan tilausten keräily voidaan jakaa neljään eri osa-alueeseen niihin kohdistuvan ajankäytön mukaan (Kuva 8). Nämä osa-alueet ovat liikkuminen, tuotteiden etsiminen, poimiminen sekä muut vaadittavat toimenpiteet tilauksen käsittelemiseksi. Sama jakauma nähdään pätevän myös kohdevaraston tapauksessa, kuitenkin sillä erotuksella, että

liikkumisen osuus saattaa olla jopa suurempi kuin 55 %. Tämä johtuu siitä, että paperitöitä ja muita aktiviteetteja (20 %) on ajansaatossa automatisoitu, minkä voidaan olettaa pienentävän niiden suhteellista osuutta tässä kaaviossa. Ottaen huomioon pelkkään liikkumiseen käytetyn ajan osuuden koko prosessista sekä sen tuottamattomuuden, on hyvin perusteltua valita kehityskohteeksi juuri tämä osuus prosessista.



Kuva 8 Tilauksen keräilyyn käytetty aika (mukaillen Bartholdi & Hackman, 2014, s.25)

3.1.1 Keräilyprosessin nykytila

Kohdevarastossa keräily tapahtuu kappaleen 2.4 kuvaamalla tavalla, edeten suoraviivaisesti keräyksessä olevien tuotteiden järjestyksen mukaisesti. Käytössä oleva keräilyrata on suunniteltu tukemaan yhdestä tasosta tapahtuvaa manuaalista keräilyä. Useasta tasosta tapahtuvaa keräilyä tai automaatiota ei keräilyn osalta ole toteutettu eikä sen toteuttaminen ole todennäköistä lähitulevaisuudessa. Automaattisten keräilyjärjestelmien on todettu olevan kalliita investointeja, ja niiden sovittaminen alati muuttuvaan markkinaympäristöön voi olla hankalaa. Keräilyradan pohjapiirroksen suunnitteluun ei projektissa oteta tarkemmin kantaa, mutta radan suunnittelun voidaan todeta vaikuttavan optimaaliseen reittiin merkittävästi, sillä keräilijän käyttämästä ajasta noin puolet kuluu siirtymiin paikkojen välillä (Moeller, 2011, s. 2). Yleinen tavoite varastonpohjapiirroksen ja hyllysijoittelun suunnittelussa on minimoida käsittelykustannukset, jotka ovat De Kosterin mukaan monissa tapauksissa esitetty kuljetun matkan lineaarisena funktiona (De Koster ym., 2007, s.9).

Prosessia kehitettäessä tulee keräilyn itsessään säilyä työntekijälle mahdollisimman yksinkertaisena, jotta keräilyvirheet voidaan minimoida. Keräilyvirheiden käsittely on kallis prosessi, sillä se voi sisältää mm. virheellisesti toimitetun tuotteen palautuskustannukset, uuden

toimituksen sekä muita aikaa vieviä korjauksia esim. varastotilanteeseen. Tämä asettaa kehitystyölle rajoitteen olla tekemättä ratkaisusta liian monimutkaista keräilijän näkökulmasta. Johtuen vaatimuksesta pitää prosessi yksinkertaisena ovat vaihtoehdot tehokkuuden nostamiseksi parempi tuotesijoittelu ja tehokkaampien keräilyteknologioiden käyttö (Luotonen, 2017). Kohdevaraston tapauksessa olemassa olevaan teknologiaan ei ole tarvetta tehdä muutoksia. Käytössä olevat varastokuutiot on nykyisellään hyvin hyödynnetty ja tilan käytöstä on onnistuttu tekemään tehokasta. Pääosin keräilyyn käytetty varastoalue koostuu kuormalavahyllystöstä, jota on tarpeen mukaan täydennetty pientavarahyllyillä.

Yksinkertainen keräilyalueen pohjapiirros (Kuva 1) ja yhteneväiset hyllyratkaisut auttavat myös tuotteiden sijoittelun suunnittelussa ja automatisoinnissa. Varaston hyllyratkaisuihin sekä laite- ja trukkikannan uudistamiseksi on viime vuosina tehty useita investointeja. Investointien avulla on osin pystytty vähentämään turhaa liikkumista varaston eri osissa ja samalla kasvatettu varastointikapasiteettia. Mikäli tehokkuutta lähdetään hakemaan teknologiamuutoksista edellyttäisi se huomattavia lisäinvestointeja. Luotosen (2017) mukaan investointien ongelmana on yleisesti se, että takaisinmaksuaikaa ei pystytä täysin aukottomasti laskemaan, jolloin perustelu muuttuu haastavammaksi.

Hyvin pienten tilausten tai tuotteiden kohdalla paternosterit, eli vertikaalikarusellit tai hissityyppiset varastoautomaatit (Logistiikan maailma, 2020a) voisivat tuoda lisää tehokkuutta keräilyyn, vähentäen tarvetta tuottamattomalle liikkumiselle varastossa. Tämä uusi teknologia lisäisi keräilyprosessin tehokkuutta ainoastaan pienellä osalla tilauksia, sillä suuren volyymin tuotteita ei pystyttäisi käsittelemään tehokkaasti varastoautomaattien kautta. Tämä johtuu suurimmalta osin isojen tuotteiden varastointitilan tarpeesta, jota ei kyseisessä varastossa voida ratkaista pystysuuntaan varastoivilla automaateilla. Sen sijaan parempi tuotesijoittelu yhdistettynä olemassa oleviin keräilyteknologioihin pystyy kasvattamaan tehokkuutta suuremmalla osalla tilauksia.

Nykyisen keräilyprosessin tuotesijoittelusta tai sen puutteesta johtuviksi ongelmakohdiksi on tunnistettu seuraavat kohdat:

- Tuotesijoittelusta johtuvat keräilyvirheet (esim. samanlaiset pakkaukset viereisillä paikoilla).
- Pitkät välimatkat tilauksen tuotteiden välillä lisäävät tuottamatonta liikkumista.
- Uusia ja kasvavia tuotteita jää toistuvasti ilman vakituista paikkaa, mikä aiheuttaa tarpeetonta etsintää.
- Tuotteiden sijoittelua joudutaan arvioimaan manuaalisesti kuukausittain.

Projektin aihe täsmentyi keräilyprosessin tehostamiseen tuotteiden sijoittelua optimoimalla juuri edellä mainituista syistä.

3.1.2 Keräilyreitin optimoinnin problematiikka

Kuten kappaleessa 2.6.1 esitetään, on keräilyreitin optimointi eräänlainen erityistapaus kauppamatkustajan ongelmasta, jossa liikkumista on rajoitettu käytävärakenteen mukaisesti. Tämä rajoitus tekee ratkaisun laskemisesta tietokoneella huomattavasti nopeampaa. On kuitenkin monia käytännön syitä miksi useimmat varastonhallintajärjestelmät eivät tue tällaista optimointia. Tärkeimpänä syynä on se, että jokaisen optimointialgoritmin tulisi tuntea varaston geometrinen pohjapiirros ja välimatkat jokaisen varastopaikkaparin välillä. Monet varastonhallintajärjestelmät eivät ylläpidä tämän tason tietoa sen vaikean keräämisen ja ylläpidon takia. (Bartholdi & Hackman, 2014.)

Lisäksi ongelmaksi voi muodostua reitin noudattaminen, mikäli se on keräilijän näkökulmasta monimutkainen. Kohdevaraston WMS-järjestelmä toimii vastaaville järjestelmille tyypilliseen tapaan antaen käyttäjälle keräiltävien tuotteiden osoitejärjestelmän mukaiset varastopaikat siinä järjestyksessä, missä ne tulisi kerätä. Järjestelmä ei anna käyttäjälle varsinaista reittiä, jota keräilijän tulisi seurata. Keräilijän tulee siten itse löytää reitti paikasta toiseen, mikä voi olla vaikeaa kovassa kiireessä ja ainoastaan lokaalin tiedon varassa. Lokaalilla tiedolla tarkoitetaan sitä, että keräilijä näkee vain kyseisen käytävän, jolla hän milloinkin on, sekä listan paikoista jonne tulisi seuraavaksi liikkua. Voi olla hankalaa hahmottaa mikä on kokonaisuuden kannalta tehokkain reitti huomioiden kaikki siirtymät ja käytettävissä olevat käytävät tilauksen keräämisen aikana.

Kohdevaraston tapauksessa keräilijällä on nähtävissä kartta varaston pohjapiirroksesta sekä näyttöpäätteellä lista kaikista paikoista, joissa tilauksen aikana tulee vierailla. Hyödyntäen tietoa varaston pohjapiirroksesta keräilijän on mahdollista muodostaa yksinkertaisia reittejä tehokkaasti olemassa olevan lokaalin tiedon perusteella. Kuinka voimme luoda lyhyitä keräilyreittejä, jotka ovat ymmärrettäviä myös keräilijöille, joilla ei ole aiempaa kokemusta varastosta tai yksityiskohtaista karttaa sen pohjapiirroksesta? Ratkaisu tähän ongelmaan voi löytyä kappaleessa 2.6.2 esitettyjen heuristisiten menetelmien hyödyntämisestä.

Kuvitellaan, että keräilijän tulisi vierailla jokaisessa keräilyalueen varastopaikassa ja meidän tulisi löytää sopiva reitti tämän toteuttamiseksi. Voimme kertaalleen laskea tehokkaan sekä kaikki paikat kattavan reitin ja käyttää sitä yhä uudestaan ja uudestaan. Kun keräilijä liikkuu hakemaan tilauksella olevaa tavaraa, voimme vaatia hänen vierailevan varastopaikoissa samassa

järjestyksessä, kuin aiemmin laskettu tehokas ja kaikki paikat kattava ”globaali” reitti osoittaa. Näin ollen globaali reitti osoittaa järjestyksen, jota tulisi noudattaa kaikissa keräilyissä. Kun WMS-järjestelmä saa uuden asiakastilauksen, sen tulisi järjestää tuotteet varastopaikkojen mukaiseen järjestykseen siten, että järjestys vastaa tämän globaalin reitin osoittamaa järjestystä. Ajatuksena on, että mikäli globaali reitti on liikkumiskustannukseltaan tehokas, ovat todennäköisesti myös sen alireittien eli yksittäisten asiakastilausten käyttämät reitit tehokkaita. Ongelma löytää tehokas globaali reitti läpi kaikkien varastopaikkojen tunnetaan nimellä todennäköisyyspohjainen kauppamatkustajan ongelma (Probabilistic Travelling Salesman Problem, PTSP). (Bartholdi & Hackman, 2014, s.154.)

Varaston pohjapiirroksen ollessa köysitikasmainen on yksinkertaisten globaalien reittien muodostaminen mahdollista toteuttaa esimerkiksi yksinkertaista S-muotoista reititysheuristiikkaa käyttämällä. On kuitenkin epätodennäköistä, että yksistään tehokkaan globaalin reitin määrittäminen toisi merkittävää parannusta nykyiseen tilanteeseen, sillä jo olemassa oleva keräilyalueen suunnittelu pohjautuu vastaavaan menetelmään. Kappaleessa 2.5 esitetty globaali keräilyreitti on keräilijän kannalta helposti omaksuttavissa ja se tukee hyvin varaston pohjapiirrosta. Siten pohdittavaksi tuleekin kuinka saada aikaan parannusta nykyisellä globaalilla reitillä, jotta se olisi keräilyn kannalta tehokkaampi?

Kuten aikaisemmin todettu, myös tuotteiden sijoittelulla on merkittävä rooli tilauskohtaisten keräilyreittien muodostumisessa. Mitä vähemmän ja lyhempiä kiertoteitä keräilijä joutuu liikkumaan, sitä yksinkertaisempi keräilyreitistä tulee. Tämä tarkoittaa myös matka-ajan säästön todennäköisempää realisoitumista, koska keräilijän on helppo noudattaa yksinkertaista reittiä eikä hänen tarvitse hahmottaa pitkiä tai monimutkaisia kiertoteitä. Ratkaisuksi globaalin reitin tehostamiseen esitämme tuotteiden optimaalisempaa sijoittamista siten, että reittiä tarvitsee seurata mahdollisimman lyhyen matkaa ennen keräystehtävän valmistumista. Tässä avainasemassa toimii tuotteiden tehokas volyymiperusteinen sijoittelu, joka mahdollistaa sen, että suurin osa tilauksista voidaan päättää jo reitin alkupäässä. Koska huonoimmassa tapauksessa tilaus saattaa silti sisältää tuotteita jotka ovat sijoitettuna aivan reitin alku- ja loppupäähän, emme voi taata parannusta jokaiselle tilaukselle. Kappaleessa 4 tullaan osoittamaan, että uudella sijoittelulla voidaan kuitenkin parantaa keräilyn kokonaistehokkuutta.

3.1.3 Varastoitavien tuotteiden ominaispiirteet

Tuotteiden sopivuutta eri keräilypaikoille voidaan arvioida niiden menekin lisäksi myös muiden määritettyjen apuparametrien mukaisesti. Esimerkkeinä tuotteen varastopaikalle sopimiseen

vaikuttavista ominaispiirteistä toimivat tuotteen fyysiset ominaisuudet kuten paino, lava- tai laatikkokoko. Tässä tutkielmassa menekki on tärkein yksittäinen tuotteiden sijoitteluun vaikuttava parametri. Muut tuotteiden ominaisuuksista koostetut parametrit tulevat käytännön rajoitteista, jotka vaikuttavat tuotteen sijoitteluun ja ne on siten huomioitava jo esivalmisteluvaiheessa (Kappale 2.3). Menekkiin merkittävästi vaikuttavia muutoksia keräiltävien tuotteiden osalta ovat esimerkiksi kausivaihtelut eri tuotekategorioiden välillä. Kausittaiselle vaihtelulle tyypillinen esimerkki on punaviinin myynnin kasvu jouluna ja vastaavasti kuohuviinien kasvava kysyntä ennen uudenvuodenaattoa. Eniten kysynnän ennustettavuutta vaikeuttaa jatkuva uusien tuotteiden lanseeraus ja vanhojen hiipuminen, mikä vaatii järjestelmältä joustavuutta. Myös kulutustrendit kuten alkoholittomien tai matala-alkoholisten tuotteiden kysynnän kasvu voivat vaikuttaa tuotteiden kysyntään merkittävästi, mutta kasvu on yleensä helposti havaittavissa ja jopa osin ennustettavissa. Tällä hetkellä näiden muutosten huomioiminen tuotteiden sijoittelussa on täysin manuaalista.

Tutkimuksessa käytetty tuotteiden menekkiä arvioiva painokerroin ”weight” sisältää tärkeimpänä komponenttina keskiarvon neljän kuukauden toteutuneesta myynnistä sekä tulevan neljän kuukauden myyntiennusteesta. Keskiarvon käyttäminen on perusteltua, jotta myös uusille tuotteille saadaan ennuste ilman toteutunutta myyntiä. Myyntiennusteen keskiarvo vaihtelee rajusti, jopa 0 ja 1 000 000 välillä, mutta kuitenkin niin, ettei monikaan tuote ole lähellä maksimi tai minimi arvoja. Keskiarvo on painokertoimena päätetty muuttaa luvun 2-kantaiseksi logaritmiksi ja poissulkea tuotteet joiden ennusteen keskiarvo on 0. Tällä tavoin painokerroin saadaan skaalautumaan välille 1 – 20.

Tähän kertoimeen on mahdollista ottaa mukaan myös muita komponentteja, joilla halutaan vaikuttaa tuotteen painoarvoon sen sijoittelussa. Esimerkiksi vähentämällä kuohuviinien painoarvosta 1 kesäsesongin jälkeen, voidaan tällä tavoin säätää volyymiltaan tasavahvojen tuotteiden painoarvoja suhteessa toisiinsa ja ohjata niiden sijoittumista keräilyalueella. Tällainen toimenpide voi olla tarpeen, mikäli tuotteen tai tuoteryhmän kysyntään on tulossa jokin merkittävä muutos. Kyseessä todennäköisesti olisi varsin radikaali muutos johon halutaan reagoida voimakkaasti ennen kuin se näkyy myynnin ja ennusteen keskiarvossa. Jotta tämä säätömahdollisuus voidaan säilyttää, tullaan projektissa laskemaan lopullinen painokerroin piirteeseen ”weight2”, joka on myös muutettu kokonaisluvuksi. Käytännössä kaikki tuotteille käytetyt painokertoimet jakaantuvat tällöin välille 3-20.

Varastopaikkojen osalta painoarvot määräytyvät paikan etäisyyden mukaan välille 5-10 siten, että keräilyradan alkuosa on rankattu parhaiksi paikoiksi ja paikkojen luokitusta lasketaan radan loppua kohden. Jokaiselle paikalle on myös annettu pienin mahdollinen laatikkokoko sen mukaan,

millainen tuote paikalle voidaan sijoittaa. Tällä tavoin voidaan raskaimmat laatikot ohjata radan alkuosaan. Muita tuotteen sopivuutta rajoittavia ominaisuuksia voisivat olla mm. varastopaikan kapasiteetti ja tyyppi (hylly, vierintähylly, lattia, jne.), mutta ne ovat rajattu pois tutkielman laajuudesta. Käytännössä parametreja voidaan lisätä jopa useita kymmeniä esim. paikalle haluttu tuoteryhmä, lavakoko tai myyntikanava, mutta se ei ole kuljetun matkan optimoinnin kannalta oleellista. Muiden parametrien lisääminen on tarpeen tehtäessä monitavoiteoptimointia parhaan mahdollisen käytännön ratkaisun löytämiseksi juuri tietyn varaston tarpeisiin.

3.2 Laajuuden määrittely

Projektin tavoitteena on tuottaa malli, jonka avulla aktiivisille tuotteille voidaan osoittaa keräilyn tehokkuuden kannalta mahdollisimman hyvä sijoittelu varastossa. Jotta keräilyreitti pysyy yksinkertaisena, on toteutuksessa päätetty kuljettavan reitin osalta tukeutua olemassa olevaan tapaan edetä keräilyradalla. Tämä toteutuu käytännössä olemassa olevalla WMS-järjestelmällä, sillä keräilyradan kulkusuunta yhdistettynä paikkojen järjestykseen ohjaa keräilijän toteuttamaan tätä reittiä. Projektissa pyritään asettamaan tuotteet tälle globaalille reitille niin, että nykyistä reittiä käyttämällä keräilyyn kuluva keskimääräinen aika vähenee.

Volyymiin perustuva sijoittelumenetelmä on todettu olevan nykyisellään kaikkein tehokkain menetelmä, mutta usein sen ylläpitokustannukset voivat osoittautua liian kalliiksi. Projektissa toteutettava malli pyrkii toteuttamaan volyymiperusteisen järjestyksen automaattisesti, hyödyntämällä esitietoja olemassa olevista tuotteista ja varastopaikoista niiden reunaehdot huomioiden. Näin käyttäjä voi verrata nykyistä sijoittelua uuteen ja toteuttaa tarvittavat tuotteiden siirrot paikasta toiseen, mikäli siirroille on tarvetta.

Projektin laajuuteen ei kuulu automaattisten siirtoehdotusten laatiminen. Ehdotukset ovat kuitenkin toteutettavissa tallentamalla tiedot sen hetkisestä sijoittelusta ja vertaamalla tätä automaattisesti viimeisimpään algoritmin suosittamaan sijoitteluun. Listattaessa eroja nykyisen ja ehdotetun sijoittelun välillä tulisi tuotteille asettaa raja-arvot siitä, miten merkittävä muutos volyymissa kahden eri paikan välillä laukaisee tuotteen siirto kehotuksen. Mikäli raja-arvo ei ylity, säilytetään vertailtavien tuotteiden olemassa olevat paikat.

Toteutettavan mallin toimintaa arvioidaan simuloimalla sen tuottamaa sijoittelua tuotantoympäristöä vastaavilla asiakastilauksilla. Simuloinnin toteuttaminen edellyttää keräilijän liikkeitä arvioivan reititysheuristiikan toteuttamista (kappale 5.3). Simuloinnin tulee mitata matkaa, joka keräyksen aikana joudutaan pahimmassa tapauksessa liikkumaan.

Tuotteiden sopivuutta kullekin varastopaikalle arvioidaan valittujen parametrien perusteella. Toteutukseen valitut keskeisimmät parametrit ovat paikalle soveltuva laatikkokoko ja tuotteen keskimääräinen myyntivolyymi. Parametrien määrän optimaalisuutta tai niiden käytännön riittävyyttä ei voida taata, sillä lyhyen elinkaaren tuotteiden vaikutusta lopulliseen järjestetykseen on hankala arvioida saatavilla olevan tiedon perusteella. Lisäksi kausittaiset vaihtelut voivat vaikuttaa mallin suorituskykyyn pitkällä aikavälillä, mikäli tuotekategorioita ei huomioida erillisenä parametrina tuotteiden sijoittelussa, vaan se tehdään edelleen manuaalisesti.

Projekti tulee vastaamaan kysymykseen siitä, onko volyymiperusteisella tuotesijoittelulla mahdollista lyhentää liikuttua matkaa tilauksen keräämisen aikana. Lisäksi projekti arvioi sitä, miten hyvin varaston nykyinen sijoittelu on toteutettu suhteessa satunnaiseen sijoitteluun ja millaista hyötyä sen muuttamisella volyymiperusteiseksi voidaan saavuttaa. Projektissa tuotettu malli ei ole tarkoitettu päivittäiseksi työkaluksi varastopaikkojen optimointiin sen muutosherkkyyden vuoksi. Malli soveltuu parhaiten kuukausittain tai harvemmin tapahtuvaan käyttöön, esimerkiksi ennustepäivityksen yhteydessä. Projektin laajuuteen ei kuulu graafista käyttöliittymää tai lopullista työkalua, joka osoittaa minkä tuotteen sijoittelua tulisi muuttaa kullakin suorituskerralla.

3.3 Toteutuksen tavoitteet

Projektin pääasiallisena toimituksena toteutetaan algoritmi tuotteiden yhdistämiseksi niille parhaiten sopiville varastopaikoille sekä lista ehdotetuista varastopaikoista tutkimushetken varastotilanteessa. Muita projektin toimituksia ovat keräilyn simulointiin käytetty reititysalgoritmi, tekstikäyttöliittymä ja arvio nykyisen sijoittelun toimivuudesta verrattuna optimoituun sijoitteluun.

Projektin jälkeen kohdeyrityksellä on käsitys siitä, miten suorituskykyinen nykyinen varastonasettelu on verrattuna volyymiperusteisesti optimoituun tai satunnaiseen vaihtoehtoon. Jatkossa kohdevarastossa on mahdollista hyödyntää algoritmia yhtenä työkaluna keräilyn kannalta tehokkaan varastoympäristön ylläpidossa ja kehityksessä. Projekti myös luo perustan keräilyalueen jatkokehitys mahdollisuuksille toimien vertailukohtana muita ratkaisuja arvioitaessa.

4 Toteutus

Olemme tutustuneet optimointiongelmaan, saatavilla olevaan taustatietoon ja muihin algoritmin valintaan vaikuttaviin tekijöihin kappaleessa 2 sekä tuotteisiin ja toimintaympäristön nykytilaan kappaleessa 3. Hankittua tietoa hyödyntäen, käsittelemme seuraavaksi varastopaikkojen optimointialgoritmin toiminnan ja testaamme sen toimivuutta käytännössä. Kappaleessa 4.1 esittelemme lyhyesti käytetyn ohjelmointiympäristön. Varsinainen toteutus toimintaperiaatteineen esitellään kappaleessa 4.2, jonka jälkeen simuloinnilla vahvistetut tulokset esitellään kappaleessa 4.3.

4.1 Ohjelmointiympäristö

Ohjelmointiympäristönä on käytetty avoimeen lähdekoodiin perustuvan Anaconda-jakeluympäristön (Anaconda, 2019) avulla luotua virtuaaliympäristöä sekä siihen kuuluvista editoreista Spyder (v.3.3.6) editoria. Spyder on hyvin tieteelliseen käyttöön soveltuva editori, joka on kirjoitettu Python ohjelmointikielellä ja se on tarkoitettu Python ohjelmointiin. Editorissa on mm. edistyneet muokkaus-, analysointi-, virheenkorjaus- ja visualisointiominaisuudet (Spyder, 2019). Erityisesti visualisointiominaisuudet ovat tärkeitä saatavilla olevan datan ymmärtämiseksi ja mahdollisten poikkeavuuksien havaitsemiseksi. Työssä on käytetty Python-kielen versiota Python 3.7., jonka standardikirjastojen (Python, 2019) lisäksi merkittävimpiä työssä käytettyjä kirjastoja ovat Pandas, Matplotlib, csv sekä NumPy.

Pandas on yleisimmin työssä käytetty kirjasto sen tarjoamien erityisten tietorakenteiden ansiosta (DataFrame), jotka mahdollistavat monimutkaisten ja erityyppisten tietotaulukoiden käsittelyn tavalla, jota esimerkiksi NumPy ja SciPy kirjastot eivät tarjoa. Erityisesti .csv tiedostojen sulava lataaminen, käsittely ja visualisointi ovat merkittäviä syitä tämän kirjaston käytölle. (Boschetti & Massaron, 2016, s. 25.)

NumPy on kirjasto, joka tarjoaa toimintoja n-ulotteisten taulukoiden ja matriisien käsittelyyn. Matplotlib on piirtokirjasto, joka hyödyntää NumPy:ä monenlaisten kaavioiden ja kuvaajien piirtämiseen, mukaan lukien piste- ja viivakaaviot, histogrammit sekä 3D-kuvaajat. (Baka, 2017) Lisäksi työssä on käytetty csv-kirjastoa, joka on tarkoitettu CSV-tiedostojen (comma separate

value) lukemiseen ja niihin kirjoittamiseen. CSV-tiedostojen käsitteleminen on keskeisessä osassa toteutusta, sillä ohjelma luo listasta tuotantotilauksia erilliset CSV-tiedostot jokaista tilausta kohden. Näitä tilauksia käytetään myöhemmin keräilyn simulointiin, jossa tiedostot käsitellään yksitellen.

4.2 Optimointityökalun toteuttaminen ohjelmana

Python-ohjelmointikielellä tehdyssä toteutuksessa kaikki keräilyalueeksi valitut paikat alustetaan ensin tyhjäksi ja sen jälkeen algoritmin suoritus etenee täyttämällä niitä kullekin paikalle optimaalisilla tuotteilla. Tätä varten keräilyalue on parametrisoitu niille soveltuvien tuotteiden mukaisesti, jonka lisäksi paikat on painotettu niiden paremmuuden mukaan. Tuotteiden ja mahdollisten keräilypaikkojen määrän ollessa suhteellisen matala on tuote ja keräilyalue kohtaiset painotukset voitu laskea Excel-tiedostossa tietojen esikäsittelyvaiheessa.

Toteutettu ohjelma järjestää varastopaikat (locations3.csv) niiden painotuksen ja varastopaikan numeron mukaiseen järjestykseen pikalajittelulla. Tuotteet puolestaan järjestetään niiden painotuksen ja laatikkokoon mukaiseen järjestykseen. Algoritmin suorituksen aikana jokainen varastopaikka käydään läpi ja kullekin paikalle tarjotaan painoarvoltaan suurinta vapaana olevaa tuotetta. Painoarvoltaan tasavahvat tuotteet noudattavat varastopaikan mukaista suuruusjärjestystä. Mikäli paikalle on testattu kaikkia vapaana olevia nimikkeitä eikä mikään niistä ole siihen soveltuva, merkataan paikka tyhjäksi ja suorituksessa siirrytään tutkimaan seuraavaa paikkaa. Lopuksi algoritmi tallentaa listan varastopaikoista sekä niiden optimaalisista nimikkeistä painoarvoineen. Samassa yhteydessä ilmoitetaan käsiteltyjen paikkojen lukumäärä sekä niille löytyneiden optimoitujen nimikkeiden määrän.

Ohjelman tallentamaa listaa ehdotetuista varastopaikoista kullekin tuotteelle voidaan käyttää apuna varaston järjestämiseen. Kun varastopaikat on järjestetty ensimmäisen suorituskerran osoittamaan järjestykseen, voidaan algoritmin käyttöä jatkaa säännöllisesti muutostarpeiden tunnistamiseksi. Suorittamalla sama ohjelma uudelleen esimerkiksi kuukauden kuluttua voidaan tulosta verrata edelliseen eli nykyiseen järjestykseen ja tarvittaessa tehdä siihen muutoksia.

Suorituskyvyn testaamiseksi toteutusta testattiin ensin koko nimikkeistölle siten, että käytössä oli myös keräilyalueen ulkopuolisia varastopaikkoja. Testiaineistossa varastopaikkoja oli yhteensä 1051 kpl, joista 292:lla oli vähimmäisvaatimuksena laatikkokoko 12 tai suurempi. Kaikkiaan 2346:sta tuotteesta, optimoitu tuote löydettiin ensimmäisen testiaineiston perusteella 1013

varastopaikalle. Algoritmin suorituskyvyn todettiin olevan riittävän hyvä myös nykyistä suuremmalle nimikemäärälle.

Käytännön toteutuksessa käytettiin testiaineistoa, jossa varastopaikkoja oli vähemmän, yhteensä 597 kpl. Näistä paikoista osa merkattiin suorituksen aikana tyhjäksi. Tyhjäksi merkityt paikat olivat käytäväpaikkoja tai varattuna muuhun käyttöön (Kuva 9).

```
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '143A')
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '142A')
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '141A')
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '140A')
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '139A')
Paikalle ei ole olemassa sopivaa nimikettä   1440 ('tyhja ', '138A')
...paikkojen optimointi on valmis, käsiteltyjä paikkoja  597  kpl
optimaalisia nimikkeitä löytynyt 552  kpl
```

Kuva 9 Keräilypaikkojen tyhjäksi merkitseminen

4.3 Tulosten simulointi tuotantoympäristöön

Optimoinnin tulosten arvioimiseksi on vertailtava sitä, millaiset erot erilaisten tuotesijoitteluiden välillä käytännössä on. Arvioinnissa on otettu huomioon kolme erilaista tuotesijoittelua: satunnainen eli stokastinen tuotesijoittelu, varaston nykyinen sijoittelu sekä optimoinnilla saavutettu sijoittelu. Arviointimenetelmänä käytetään simulointia, jota varten on toteutettu erillinen simulointialgoritmi. Simulointialgoritmin toimintaa käytännössä sekä simulointia tutkimusmenetelmänä tarkastellaan tarkemmin luvussa 5.

Testattaessa simulointia neljällä tuotannosta satunnaisesti valitulla tilauksella näytti optimoitu versio suoriutuvan näistä parhaiten. Kuljettua matkaa mittaavan simulointialgoritmin arvio pystyttiin myös varmistamaan laskemalla etäisyydet esimerkkitilauksilta manuaalisesti. Koska algoritmin arvio vastasi todellisuudessa kuljettavaa matkaa, voitiin arviointiheuristiikan olettaa toimivan. Satunnaisesti valitulla sijoittelulla neljän tilauksen keräämisen kustannus liikuttuina yksiköinä oli 3237:n varastopaikan verran, entisellä sijoittelulla 3034:n ja optimoidulla 2481:n yksikön verran. Liikkumiskustannuksella eli liikutuilla yksiköillä tarkoitetaan kustannusta, joka syntyy liikuttaessa varastopaikkojen välillä. Pienimpänä mittayksikkönä on käytetty yhden paikkavälin liikkumista esimerkiksi varastopaikalta 1 varastopaikalle 2. Tämä kustannus (myöhemmin c) on suure, jota toteutuksessa yritetään minimoida. C tarkoittaa liikkumiskustannusten summaa koko tilauksen osalta.

$$C = \sum_{\substack{i=1 \\ c=calculated\ cost\ to \\ move\ to\ next\ line}}^{\substack{n= \\ number\ of \\ order\ lines}} c(\{order\_line(i), order\_line(i+1)\})$$

Lopullisten tulosten arvioinnissa on huomioitava myös mahdolliset erityistapaukset ja simuloinnin rajoitteet. Simuloitava aineisto perustuu niihin tuotteisiin, jotka löytyvät tuotantoympäristöstä kopioiduilta tilauksilta ja joille löytyy alkuperäisessä sijoittelussa varastopaikka keräilyalueelta. Tuotteet joille ei alun perin ollut paikkaa varastossa ovat siten rajattu pois simulointimateriaalista. Siten normaalitapausta eli varaston nykyistä sijoittelua simuloitaessa tiedämme, että kaikille tuotteille on olemassa keräilypaikka jossakin kohtaa keräilyaluetta.

Suoritettaessa simulointi uudella optimoidulla tuotesijoittelulla voi käydä niin, ettei tilauksella olevaa tuotetta ole optimoitujen paikkojen joukossa lainkaan. Tämä johtuu siitä, että optimointialgoritmi on löytänyt keräilyradalle kyseistä tuotetta suuremman menekin omaavia tuotteita ja siitä johtuen pienemmän menekin tuote on tippunut pois keräilyradalle mahtuvien tuotteiden joukosta. Koska normaalitapauksessa tuote olisi löytynyt keräilyalueelta, ei sitä voida sulkea pois laajuudesta.

Tässä tapauksessa simulointi-algoritmi sijoittaa tuotteen paikalle 322, joka on keskellä keräilyaluetta, mutta siitä ei ole paluu kustannusta (myöhemmin e) takaisin alkuun, mikäli kyseessä sattuu olemaan keräystehtävän viimeinen paikka. Tämä perustuu käytäntöön sijoittaa pienet tuotteet hyllyjen päätyyn matalammille hyllyille. Vaikka tätä ei tapahdu usein, voi tapaus aiheuttaa puolueellisuutta kumpaan tahansa suuntaan. Tästä syystä olen laskenut muuttujaan "bias" (myöhemmin b) mahdollisen puolueellisuuden vaikutuksen jokaiselle tilaukselle ja lisännyt sen optimoidun version lopulliseen kustannukseen yhdessä paluukustannuksen kanssa (myöhemmin T).

Tarkastellaan esimerkiksi tilannetta, jossa alkuperäisessä sijoittelussa tuote olisi ollut paikalla 053, joka on huomattavasti parempi, kuin optimoinnissa käytetty paikka 322. Lasketaan tästä koituva ylimääräinen kustannus 322-53 = 269 ja lisätään se optimoidun version kustannuksiin kompensoimaan alkuperäisen asettelun paremmuutta. Vastaavasti jos alkuperäinen asettelu on ollut huonompi, voidaan kustannus vähentää optimoidusta versiosta. Esimerkiksi tapauksessa, jossa radalle mahtumaton tuote on optimoidussa versiossa paikalla 322 ja alkuperäisessä paikalla 450, tällöin kustannus 322-450 = -128 voidaan vähentää optimoidun version

kokonaiskustannuksesta. Tuotteen alkuperäinen sijoittaminen paikalle 450 on ollut virheellinen, koska paikalle on löytynyt optimoinnilla löydetty paremmin sopiva tuote. Tästä syystä on perusteltua vähentää liikkumiskustannus optimoidun version kokonaiskustannuksesta T.

$$T(total\ cost) = C(cost) + b(bias) + e(end\ cost)$$

Simuloitaessa keräilyä kahdeksalle asiakastilaukselle löytyi ainoastaan viisi tuotetta, joille ei optimoidussa versiossa ollut varastopaikkaa. Näiden tuotteiden aiheuttaman puolueellisuuden b lisääminen lopulliseen kustannukseen T oli vaikutukseltaan negatiivinen. Koska paikattomia tuotteita sisältäviä tilauksia oli vähän, on puolueellisuuden lisäämisen vaikutus lähes merkityksetön.

```
8  tilausta käsitelty:
alkuperäisillä keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  3034
optimoiduilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  2481
satunnaisilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  3161
optimoitu versio on alkuperäistä parempi  22.289399435711406  prosenttia ja  27.408303103587254
parempi satunnaista asettelua
```

Kuva 10 Keräilyn simulointi kahdeksalla asiakastilauksella.

Simuloitaessa 42 tilauksella, esiintyi lisäkustannus b vain 23 tilausrivillä. Tulokset olivat optimoidun version kannalta edelleen hyviä, optimoidun asettelun ollessa n. 24 % alkuperäistä parempi (Kuva 11).

```
42  tilausta käsitelty:
alkuperäisillä keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  13569
optimoiduilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  10926
satunnaisilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  13070
optimoitu versio on alkuperäistä parempi  24.190005491488183  prosenttia ja  19.622917810726705
parempi satunnaista asettelua
```

Kuva 11 Keräilyn simulointi 42:lla asiakastilauksella.

145 tilauksen simuloinnin jälkeen optimoidun asettelun ero alkuperäiseen sekä satunnaiseen asetteluun kaventui selkeästi. Huomioitavaa on, että ero satunnaisen ja alkuperäisen asettelun välillä on tässä tapauksessa ainoastaan 3 prosenttiyksikköä (Kuva 12).

```
145  tilausta käsitelty:
alkuperäisillä keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  46883
optimoiduilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  40378
satunnaisilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  45848
optimoitu versio on alkuperäistä parempi  16.110258061320508  prosenttia ja  13.546981029273358
parempi satunnaista asettelua
```

Kuva 12 Keräilyn simulointi 145:llä asiakastilauksella.

Lopullinen simulointi toteutettiin käyttäen yhteensä 4683:a tuotantoympäristöstä kopioitua asiakastilausta. Optimoidun asettelun paremmuus verrattuna aiempaan asetteluun oli edelleen selkeä, yli 15 %. Verrattaessa tulosta edelliseen suorituskertaan 145:llä tilauksella, voidaan nähdä eron optimoidun ja alkuperäisen asettelun välillä tasoittuneen. Tulosten ero edelliseen suorituskertaan nähden oli enää 0,63 % simuloitaessa yli 30 kertaa suuremmalla aineistolla. Suuremmalla aineistolla nähdään myös satunnaisen asettelun toimivuuden heikkenevän merkittävästi suhteessa nykyiseen tai optimoituun sijoitteluun. Viimeisellä simulointikierroksella optimoitu versio on jopa 27 % parempi kuin satunnainen asettelu. Myös ero nykyisen ja satunnaisen sijoittelun välillä on odotetusti kasvanut 11,77 %:iin. Arvioitaessa puolueellisuuskorjauksen merkitystä lopputulokseen voidaan sen nähdä olleen vähäinen. Optimoidulla asettelulla korjausten b summa oli yhteensä 9625 yksikköä. Tällöin korjauksen b osuus on vain 0,82 % tilauskohtaisten kokonaiskustannusten T summasta.

```
Tiedosto luettavissa
simuloidaan keräilyä oletuspaikoilla..
..simuloidaan keräilyä satunnaisilla paikoilla..
..simuloidaan keräilyä optimaalisilla paikoilla..
simulointi on valmis. analysoidaan tuloksia..
4683  tilausta käsitelty:
alkuperäisillä keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  1354412
optimoiduilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  1172798
satunnaisilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä  1492446
optimoituihin paikkoihin on lisätty/vähennetty algoritmin puolueellisuudesta aiheutunut kustannus  9625
optimoitu versio on alkuperäistä parempi  15.485531182693002  prosenttia ja  27.25516244059078 parempi satunnaista asettelua
```

Kuva 13 Keräilyn simulointi 4683:lla asiakastilauksella.

Vaikka lopullinen tulos sisältää puolueellisuudesta aiheutuneet suorituskykyä alentavat lisäykset kokonaiskustannukseen, voidaan tulosta silti pitää merkittävänä. Paikkojen optimoinnilla voidaan päästä jopa 15 %:n säästöön kuljetussa matkassa, mikä on suoriksi kustannussäästöksi muunnettuna noin 4,54 % tavanomaisen varaston kaikista operatiivisista kustannuksista.

5 Metodit ja materiaalit

Seuraavassa kappaleessa esitellään toteutetun tapaustutkimuksen keskeisimmät metodit ja aineistonkeruun perusmenetelmät. Kappaleessa 5.1 tarkastellaan simulointia tutkimusmenetelmänä. Kappaleet 5.2 ja 5.3 käsittelevät aiemmin kuvatun (Kappale 4.3) simulointialgoritmin toimintaa käytännössä. Lopuksi arvioidaan tulosten luotettavuutta kappaleessa 5.4.

5.1 Simulointi tutkimusmenetelmänä

Tämä tutkielma on tutkimustavaltaan tapaustutkimus, jossa paneudutaan tarkastelemaan varaston keruutoimintojen tehokkuutta erilaisilla tuoteasetteluilla. Tavoitteena on tehdä uusia havaintoja ja selvittää miten erilainen tuoteasettelu mahdollisesti vaikuttaisi keräilyn kokonaistehokkuuteen. Tutkimuksen projektiosuudessa toteutetun optimointialgoritmin toimintaa arvioidaan simuloimalla sen tuottamaa asettelua tuotantoympäristöä vastaavalla aineistolla. Simulointi on laskennallinen tutkimusmenetelmä, jota käytetään työkaluna tutkimuksen toteutuksessa.

Tutkielman aineistonkeruun menetelminä ovat toimineet pääasiallisesti havainnointi sekä saatavilla oleva dokumentaatio. Dokumentaatiota ja muuta tutkimusainestoa on osin tuotettu jo ennen projektin aloittamista (Kappale 2.2). Havainnoinnin etuna on, että sen avulla voidaan saada välitöntä, suoraa tietoa yksilöiden, ryhmien tai organisaatioiden toiminnasta ja käyttäytymisestä (Hirsjärvi, Remes & Sajavaara, 2009, s. 213). Projektissa on hyödynnetty mahdollisuutta päästä seuraamaan käytännön toimintaa suoraan sen luonnolliseen ympäristöön, jossa tilanteet voivat keruutehtävän aikana muuttua nopeasti ja olla vaikeasti ennakoitavia. Hankittua tietoa on hyödynnetty simulointialgoritmin kehittämisessä, jotta se saadaan toimimaan mahdollisimman tarkkaan todellisuutta jäljittelevästi. Algoritmin toimintaperiaatetta tarkastellaan tarkemmin edempänä (Kappale 5.4).

5.1.1 Simulaatiomallinnuksen käyttö tutkimuksessa

Monet tutkimusmenetelmät, kuten kyselytutkimus, edellyttävät, että on tehtävä erilaisia oletuksia tutkittavan järjestelmän tarkasta syy-seurausluonteesta. Tutkimusmenetelmänä simulointi sen

sijaan sallii tutkijan tehdä oletuksia luonnostaan monimutkaisiin järjestelmiin sellaisenaan. Siinä missä monet muut tutkimusmetodit vastaavat kysymyksiin ”mitä tapahtui?, miten?, ja miksi?” auttaa simulointi vastaamaan kysymykseen ”mitä jos?”. Näin simuloinnin voidaan nähdä tekevän havaintoja ”siirtyen eteenpäin” tulevaisuuteen, siinä missä muut metodit usein katsovat taakse historiaan määritelläkseen mitä tapahtui ja miksi. (Dooley, 2002.)

Simuloinnin etuina voidaan pitää sitä, ettei mallinnusvaiheessa ole matemaattisten mallien rajoittavia ehtoja ja se voidaan toteuttaa hyvin suoraviivaisesti. Tämä mahdollistaa monimutkaistenkin järjestelmien suorituskyvyn arvioinnin. Haittapuolena on tulosten tuottamiseen tarvittava prosessointiaika ja niiden epätarkkuus vähäisillä simulointikerroilla. Tuloksien tarkkuus paranee mitä useampia simulointiajoja suoritetaan. Simuloinnista voi myös olla vaikea saada kokonaisnäkemystä ja sen optimointimahdollisuudet ovat rajalliset muihin menetelmiin verrattuna. (Teknillinen Korkeakoulu, 2020.)

Simulaatiomallit voidaan jakaa kolmeen eri koulukuntaan: systeemidynamiikkaan (jatkuvatoiminen), diskreettien tapahtumien simulaatioihin ja agenttipohjaisiin malleihin (Dooley, 2002). Kalatien (2016) esittelemässä kuvauksessa simulaatiomallien käytöstä tutkimusmenetelmänä (ks. Shannon, 1975), on hän todennut simuloinnin olevan prosessi, jossa suunnitellaan todellisen systeemin malli. Mallia käyttämällä voidaan siten tehdä kokeita systeemin ymmärtämiseksi ja/tai erilaisten strategioiden arvioimiseksi ilman riskiä todellisen ympäristön häiriintymisestä. Simulaatio auttaa tarkastelemaan järjestelmän toimintaa tilanteissa, joissa testausta ei voida suorittaa todellisessa järjestelmässä, testaukseen kuluva aika olisi liian pitkä tai toteuttaminen kallista.

Systeemidynamiikka menetelmän ydinajatuksena on, että järjestelmän sisäinen kausaalisuusrakenne määrittää ko. järjestelmän dynaamiset ominaisuudet. Menetelmän kehittäjä Jay W. Forresterin alkuperäisenä tarkoituksena oli mallintaa toimitusketjuja ja niihin liittyviä rajoitteita (Kalatie, 2016). Menetelmään kuuluu järjestelmän käyttäytymistä määrittävien keskeisten "tilamuuttujien" tunnistaminen ja näiden muuttujien liittäminen toisiinsa kytkettyjen differentiaaliyhtälöiden avulla. (Dooley, 2002.) Järjestelmä koostuu keskenään vuorovaikutuksessa olevista palautesilmukoista, jossa ulostulovirta toimii syötteenä jollekin toiselle vaiheelle. Palautesilmukoissa muuttujina ovat *tasot* (stock), jotka kuvaavat järjestelmän tilaa sekä *nopeudet* (flow), jotka muuttavat järjestelmän tilaa. Tällaiset palautesilmukat voivat olla tyypiltään joko rajoittavia tai vahvistavia. (Kalatie, 2016.)

Diskreettien tapahtumien malliin kuuluu simuloitavan järjestelmän mallintaminen kokonaisuutena, joka kehittyy ajan myötä resurssien saatavuuden ja tapahtumien käynnistymisen perusteella. (Dooley, 2002.) Tässä mallissa erityispiirteenä on ajan käsite, sillä mallissa aika

siirtyy tapahtumien ja niiden ajoituksen perusteella eteenpäin seuraavaan vaiheeseen. (Kalatie, 2016.)

Dooley (2002) listaa diskreetin mallin tunnusomaisiksi piirteiksi alun perin Averin Law:n ja David Keltonin vuonna 1982 julkaiseman listan seuraavista komponenteista:

- **Entiteetit** (Entities): passiivisia objekteja, joista järjestelmä muodostuu (esim. ihmisiä, osia tai dokumentteja).
- **Järjestelmän tilan muuttujat** (System state): tilamuuttuja, joka kuvaa järjestelmän tilaa annetulla hetkellä ja on usein liitetty johonkin tiettyyn entiteettiin.
- **Simulointikello** (Simulaton clock): merkitsee simuloidun ajan kulumista.
- **Tapahtumalista** (Event list): luettelo, joka määrittelee tulevaisuuden tapahtumat ja ajankohdan, jolloin ne tapahtuvat.
- **Tilastolliset datan kerääjät** (Statistical counters): käytetään tietojen keräämiseen simulointiajon aikana sekä historian tallentamiseen ja sen myöhempään analysointiin.
- **Alustusrutiini** (Initialization routine): keino valmistella malli kokeellista ajoa varten.
- **Ajoitusrutiini** (Timing routine): aliohjelma, joka hallitsee tapahtumalistan tapahtumia.
- **Tapahtumarutiini** (Event routine): aliohjelma kullekin erityyppiselle tapahtumalle, joka määrittää toiminnot (muiden tapahtumien luominen, tilamuuttujien muutos), jotka liittyvät tapahtuman käynnistämiseen.
- **Raportin generoija** (Report generator): raportoi tilastollisilta datan kerääjiltä saadut kokonaistulokset.
- **Pääohjelma** (Main program): ohjelma, joka koordinoi toimintaa simulointijärjestelmän kaikkien muiden osien välillä.

Käytännössä mallin entiteettinä voidaan pitää esimerkiksi keräystyön suorittavaa ihmistä. Entiteetit liikkuvat prosessin eri vaiheiden läpi, joissa ne voivat olla jonossa, viivästyä tai varata ja vapauttaa resursseja ennen prosessin päättymistä. Tällainen tapahtumapohjainen simulointi sopii hyvin esimerkiksi tuotantolaitosten toiminnan tai varaston tapahtumavirtojen mallinnukseen.

Mukaillen Kalatien (2016, s. 17) esittämää mallia pankin palvelujärjestelmästä voidaan samankaltainen prosessi tunnistaa myös varastossa tapahtuvasta keräilystä. Prosessi alkaa, kun keräilijä saa keräystehtävän ja lähtee liikkeelle kohti ensimmäistä listattua varastopaikkaa. Ensimmäisen paikan jälkeen keräilijä etenee joko seuraavalle tilauksessa olevalle keräilypaikalle tai takaisin varikolle riippuen siitä, onko kuormalava jo täyteen lastattu. Kummallakin prosessipisteellä on mahdollista huomioida myös viive-elementti (jono muista keräilijöistä johtuen). Varikolle voidaan lisäksi määrittää automaatioresurssi kuvaamaan lavan käärintää

kelmuun käärintäautomaatissa. Tämä resurssi vaikuttaa siihen montako keräilijää voidaan vastaanottaa samanaikaisesti. Lopulta keräilijä poistuu prosessista. Simulointi mahdollistaa useiden eri muutosten vaikutusten arvioinnin: automaatioresurssia voidaan muuttaa, keräilijöiden käyttämää lavakokoa voidaan vaihdella, mahdollisia jonotus- ja käsittelyaikojen parametreja voidaan säätää.

Agenttipohjaisessa mallissa simulointiin osallistuvat agentit, jotka yrittävät maksimoida hyödyllisyystoimintonsa vuorovaikutuksessa muiden agenttien ja resurssien kanssa. Agentin käyttäytyminen määritetään sulautetun skeeman avulla, joka on luonteeltaan sekä tulkitseva että toimintaan suuntautunut. Diskreetit tapahtuma- ja järjestelmädynamiikkamallit keskittyvät muuttujiin ja tapahtumiin, kun taas agenttipohjaiset simulointimallit keskittyvät järjestelmän osallistujiin ja niiden laajempaan kollektiiviseen käyttäytymiseen. (Dooley, 2002.) Agenttipohjaisessa mallissa agenttina eli osallistujana voidaan kuvata esimerkiksi ihmisiä, liiketoimintayksikköjä, ajoneuvoja tai muita, joille voidaan määrittää yksilöllisiä käyttäytymismalleja. Järjestelmän mallintaminen tapahtuu agenttitasolla, jolloin agenttien kollektiivisesta toiminnasta syntyy koko järjestelmän dynamiikka. Agentin ominaisuuksia voivat olla esimerkiksi ennakointikyky, reagointikyky, tilan hahmottamiskyky, kyky oppia, sosiaalinen kyvykkyys ja äly. (Kalatie, 2016.)

Agenttipohjaisen mallinnuksen etuna on helppo tapa mallintaa yksilöllistä käyttäytymistä ja sitä, kuinka se vaikuttaa muihin. Käyttökohteita tällaiselle simuloinnille ovat esimerkiksi erilaiset yhteisöt ja organisaatiot. Malli sopii myös tilanteisiin, joissa mallinnettavat järjestelmät ovat liian monimutkaisia mallinnettavaksi perinteisillä menetelmillä ja/tai niissä on useita keskenään vuorovaikutuksessa olevia tekijöitä. (Kalatie, 2016.) Agenttipohjaisten mallien käytön lisääntyminen on luonnollinen seuraus tekoälyn yleistymisestä. (Dooley, 2002.) Tekoälyn tekemien päätösten ymmärtäminen ja niiden perustana olevan ympäristön simuloiminen kokonaisuudessaan tuskin on mielekästä, jos kiinnostavaa osaa toiminnasta voidaan simuloida agenttien avulla.

5.1.2 Menetelmän valinta

Kuten edellä esitetyistä malleista voidaan nähdä, on simulointi todellisuuden jäljittelyä, joka mahdollistaa ratkaisujen tai strategioiden testaamisen tuotantoympäristön kaltaisessa tilassa ilman riskiä tuotantoympäristön häiriintymisestä. Tässä tutkimuksessa simuloinnin toteuttaminen on perusteltua, koska uuden asettelun testaaminen tuotantoympäristössä voidaan nähdä olevan suuri

riski. Tuotesijoittelun muuttaminen ensimmäistä kertaa nykyisestä sijoittelusta optimointialgoritmin esittämäksi on kertaluontoinen, mutta kallis prosessi. Mikäli tästä ei voida nähdä olevan merkittävää hyötyä, niin muutosta ei kannata tehdä. Simuloinnilla mahdollinen hyöty voidaan osoittaa ennen kuin toteutusta tarvitsee tehdä reaalimaailmassa.

Projektin toteutuksessa simulointiin on käytetty diskreettiä simulointimallia, jossa ainoastaan tapahtuva keräily muuttaa annettua ympäristöä. Todellisuudessa ympäristö voidaan nähdä jatkuvatoimisena, sillä muiden keräilykoneiden sijainteja (mahdollinen este) on ääretön määrä ja näiden myötävaikutuksesta ympäristö voi muuttua epäedullisemmaksi ja esimerkiksi ruuhkaantua. Toteutuksen perustaksi on valittu diskreettien tapahtumien simulointimalli, sillä se soveltuu hyvin varaston tapahtumavirtojen simulointiin. Lopullisessa mallissa on käytetty kappaleessa 5.1.1 esitetyistä diskreetin mallin komponenteista seuraavia:

- Pääohjelma, joka yhdistää kaikki ohjelman osat toimivaksi kokonaisuudeksi.
- Entiteetit (keräilijä, varastopaikat).
- Tilastolliset datan kerääjät (kuljettu matka C, mahdollinen puolueellisuus b).
- Tapahtumalista (määrittää vaihtoehdot etenemiselle riippuen seuraavan paikan sijainnista).
- Alustusrutiini (alustaa kaikki paikat tyhjäksi ennen sijoittelun optimoinnin aloittamista).
- Raportin generoija (tulostaa lopuksi simuloinnin tulokset eri parametreilla ajettuna sekä ajokertojen väliset erot prosentteina).

Simulointi on valittu tutkimusmenetelmäksi myös siitä syystä, että alkuperäistä optimointi ongelmaa on vaikea saada matemaattisesti formaaliin muotoon niin, että yhteys todelliseen ympäristöön saadaan säilytettyä. Vastaavanlaisia haasteita oletetaan tulevan myös simulointia varten toteutettavissa etäisyysarvioissa varastopaikkojen välillä. Tällöin simulointi heuristiikkaa hyödyntäen mahdollistaa arvioinnin ilman matemaattisten mallien mahdollisia rajoitteita.

Kerättävät tuotteet voivat sijoittua keräilyalueelle hyvin moneen eri järjestykseen riippuen siitä mitä tuotteita on kulloinkin tilattu. Täydellisen etäisyysarvion, eli kaikkien reittivaihtoehtojen läpikäymisen ja niistä parhaan valitsemisen jokaiselle siirtymälle erikseen, olisi laskennallisesti erittäin tehoton vaihtoehto. Tämän kaltainen ratkaisu myös väistämättä sisältäisi suuren osan käytännön kannalta epärelevantteja yhteyksiä paikkojen välillä. Täydellisen etäisyysarvion sijasta etäisyyksien arviointiin on toteutettu tapausperusteinen heuristiikka (ks. kappale 5.3.), jossa etäisyyden laskentatapa riippuu siitä missä kohtaa keräilyaluetta käyttäjä on suhteessa seuraavaksi vierailtavaan varastopaikkaan. Näin ongelma voidaan ratkaista paljon nopeammin, mutta ratkaisu ei tällöin ole kaikkein paras.

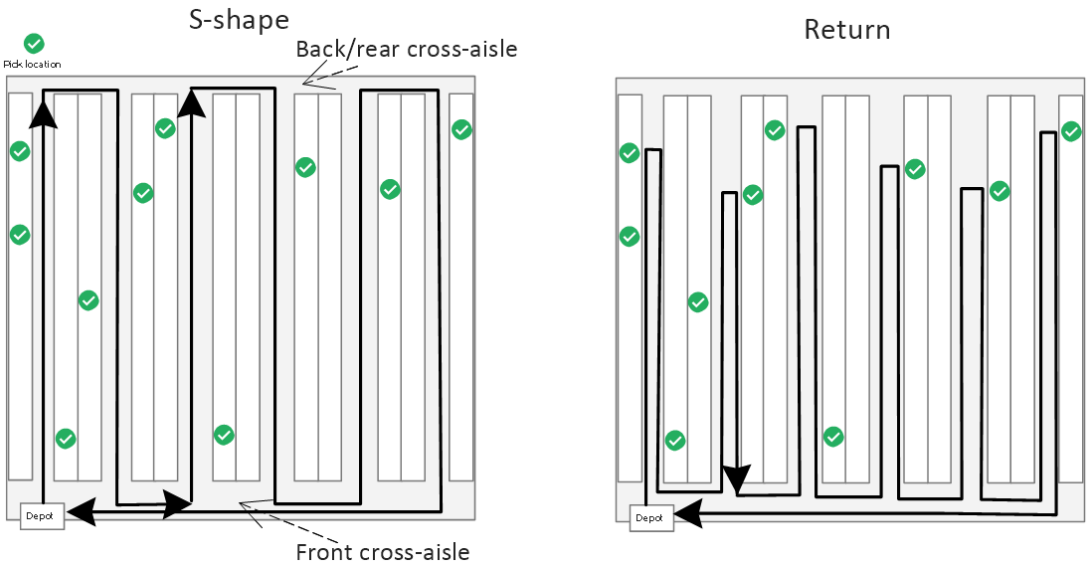
Simulointia varten toteutettu arviointiheuristiikka ei tuota täysin tarkkaa arviota varastopaikkojen välisestä etäisyydestä, mutta tuottaa riittävän tarkan arvion sen laskentanopeuteen nähden. Heuristiikka ei täten tuota ns. pätevää ratkaisua, joka säilyttäisi etäisyysarvion optimaalisuuden. Heuristisen funktion sanotaan olevan pätevä (admissible), mikäli se ei koskaan yliarvioi todellista kustannusta liikkua tutkittavien solmujen välillä (Torikka, 2015). Koska heuristinen arvio mittaa matkaa, jonka keräilijä kyseessä olevassa *tapauksessa* joutuu enintään kulkemaan, siten huomiotta voi jäädä joukko mahdollisesti lyhempiä siirtymiä. Optimaalinen ratkaisu huomioisi myös nämä vaihtoehdot, mutta se ei tässä tapauksessa ole relevanttia.

5.2 Simulointialgoritmin perusta

Kuten kappaleessa 4.3. kerrottiin, on simuloinnissa tarkastelun kohteena muuttuja *c*, joka kuvaa sitä kustannusta, minkä siirtymä keräilypaikalta seuraavalle aiheuttaa. Siirtymäkustannukset summataan jokaisen siirtymän osalta muuttujaan *C* ja keräystehtävän päätyttyä loppuun lisätään kustannus *e* siirtyä lopetusriviltä takaisin alkuun.

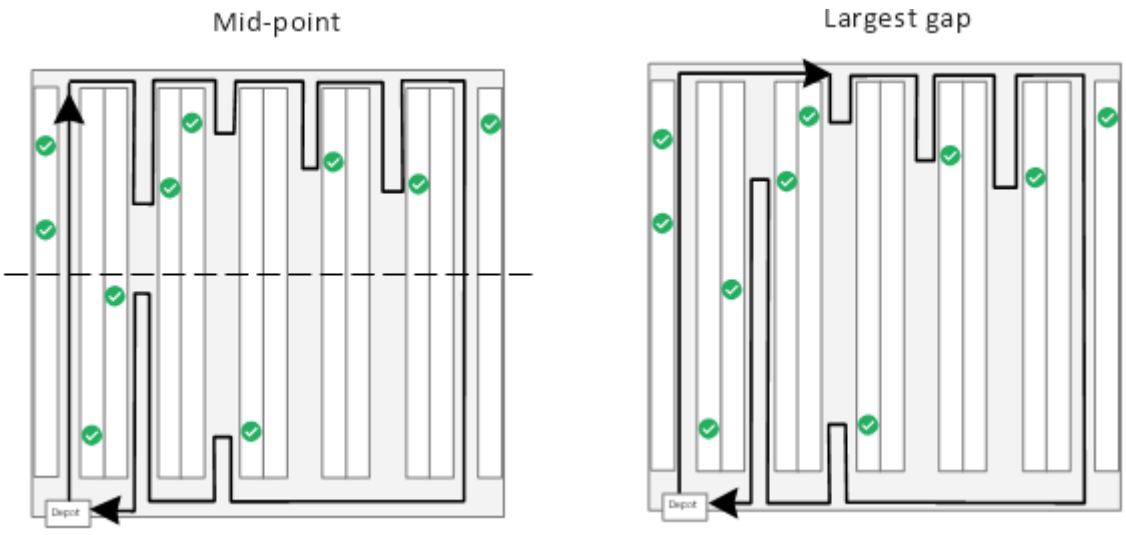
Tutkimuksen aiheesta löytyy runsaasti aiempia tutkimuksia, jotka usein tarkastelevat ongelmaa kuljettavan reitin optimoimisen kannalta. Myös näissä tapauksissa on tavoitteena minimoida kustannus C kullekin tilaukselle. Hyödyntäen aiempien tutkimusten periaatteita tehokkaasta reitityksestä, päätettiin reitti toteuttaa liikkumiskustannuksia simuloivan reititysheuristiikan avulla.

Mahdollisia reititysheuristiikkoja on olemassa useita, joista yksinkertaisimpina mainittakoon S-muoto (S-shape) ja paluu (Return) heuristiikat (Kuva 14). Näistä molemmat etenevät suoraviivaisesti varaston pohjapiirroksen mukaisesti tarkastaen vain yksinkertaisia sääntöjä. Esimerkiksi onko rivillä lainkaan tilattua tuotetta, jolloin se voidaan jättää väliin tai onko poimittu tuote rivin viimeinen, jolloin voidaan palata takaisin. Kuvan katkonaiset nuolet osoittavat poikittaiset käytävät varaston etu- ja takaosassa joiden lisäksi kuvassa esitetään keräystyön aloituspiste eli varikko (depot).



Kuva 14 Reititysheuristiikat S-muoto ja Paluu (mukaillen De Koster ym., 2007, s.19)

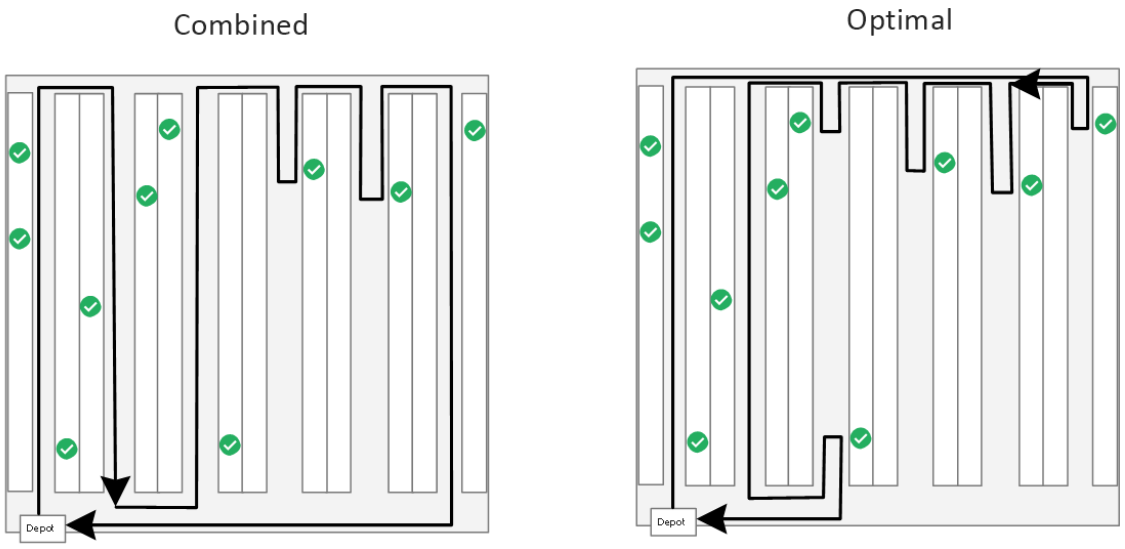
Myös keskipisteen (Mid-point) ja suurimman välimatkan (Largest gap) heuristiikat ovat mahdollisia, tällöin käytävää edetään korkeintaan puoliväliin asti tai suurimman välimatkan tapauksessa aina lähimpään mahdolliseen sijaintiin (Kuva 15).



Kuva 15 Reititysheuristiikat Keskipiste ja Suurin välimatka (mukaillen De Koster ym., 2007, s.19)

Dynaamisen ohjelmoinnin avulla eri heuristiikkoja voidaan yhdistellä (Combined) ja tehdä päätös kunkin käytävän kohdalla erikseen (De Koster ym., 2007, s.19). Tarkasteltaessa eri vaihtoehtoja

käytännön näkökulmasta, on selvää, että esimerkin kaltainen optimaalinen (Optimal) ratkaisu ei ole tutkittavassa tapauksessa käytännöllinen sen monimutkaisuuden takia (Kuva 16). Sen sijaan etäisyydenarviointiheuristiikka on toteutettu yhdistelmänä s-muotoista-, suurimman välimatkan- sekä täysin räätälöityä heuristiikkaa, joka pyrkii kuvaamaan varastossa tapahtuvaa keräilyprosessia mahdollisimman totuudenmukaisesti.



Kuva 16 Reititysheuristiikat Yhdistelmä ja Optimaalinen (mukaillen De Koster ym., 2007, s.19)

5.3 Simuloinnin toimintaperiaate

Kuten jo kappaleessa 4.3 todetaan, on tärkeää huomata, että simulointi olettaa varatun paikan löytyvän jokaiselle tuotteelle jostakin keräilyradalta. Ellei näin ole, on tuote optimoinnin kannalta menekiltään merkityksettömän pieni, jolloin se voidaan sijoittaa keräilyradan ulkopuolella sijaitseviin pienempiin hyllyihin tai kappaleen 4.3 kuvaamalla tavalla paikkaan 322. Tällaisia pienen menekin tuotteita voivat olla esimerkiksi kertaostettavat tuotteet, kausipakkaukset tai hintaluokaltaan selkeästi tavallista kalliimmat tuotteet.

Simuloinnin toimintaperiaate perustuu nykytilan ja tavoitetilan väliseen arviointiin. Simuloinnin tuloksena saatava tieto auttaa arvioimaan onko asetettu tavoite kannattava. Simuloinnissa pyritään mallintamaan todellista toimintaympäristöä ja sen rajoitteita niiden relevanteilta osin.

Toimintaympäristön simuloimiseksi on sinisenä esitetty keräilyalue (Liite 1.) muutettu vektorimuotoon, jossa jokainen sijainti on muutettu vastaamaan kyseisen paikan numeroa kokonaislukuna. Jokaiselle paikalle on määritetty loppukustannus, joka kuvaa sitä, kuinka kaukana kyseessä oleva paikka on keräilyn aloituspaikasta. Arvot ovat välillä 1-69, jossa 69

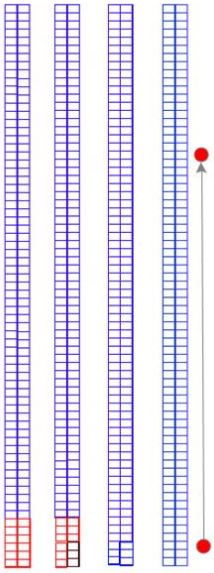
merkitsee käytäväpuoliskon viimeistä paikkaa (Kuva 17). Tämä kustannus tarkastetaan keräystyön päättyessä, jolloin se määrittää matkan palata takaisin lähtöpaikkaan. Kolmas keräilypaikkaan liitetty piirre on käytävän numero, jota käytetään mahdollisten oikoreittien arvioinnissa siirryttäessä keräilypaikkojen välillä. Keräilyalueen rajoittuessa neljään kaksipuoleiseen hyllyriviin on käytäviä tällöin käytössä 8. Tätä piirrettä käytetään myös rajaamaan pois käytöstä poistetut paikat asettamalla käytäväksi numero 99, jota ei tutkita simuloinnin suorituksen aikana. Käytöstä poistettujen paikkojen pitäminen mukana simuloinnissa on välttämätöntä pidettäessä paikkojen numerointi katkeamattomana sarjana. Tämä rajoite tulee valitusta ohjelmointitavasta.

location	end cost	row
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
...	...	...
64	64	1
65	65	1
66	66	1
67	67	1
68	68	1
69	69	1
70	69	2
71	68	2
72	67	2
73	66	2
74	65	2
75	64	2
76	63	2
77	62	2
78	61	2
79	60	2
80	59	2

Kuva 17 Paluukustannuksen (end cost) muutos varastopaikkojen ja käytävien suhteen.

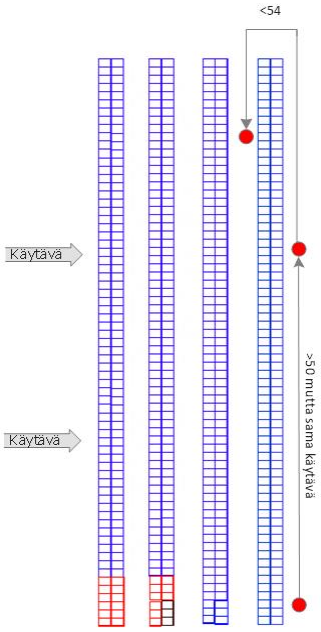
Simuloinnissa käytetty arviointiheuristiikka laskee pessimistisen arvion sille, mikä todellisuudessa kuljettava matka edellisestä paikasta sillä hetkellä tutkittavaan paikkaan olisi. Arviointi on pessimistinen vain siltä osin, että se voi arvioida reitin todellista pidemmäksi yhden käsiteltävän tapauksen sisällä. Toimintaperiaate perustuu etäisyysarvioihin, joissa huomioidaan mahdolliset poikittaiskäytäviä hyödyntävät oikoreitit. Poikittaiset käytävät jakavat rivin kolmeen erisuureen osaan (16+20+27), jotka eroavat mitaltaan korkeintaan 11 varastopaikan verran. Käytännössä algoritmi huomioi seuraavaksi kuvaillut tapaukset.

Sekä edeltävä että nyt tutkittu paikka sijaitsevat samalla rivillä ja samalla puolella käytävää. Tällöin kustannus c lasketaan vähentämällä nykyinen paikka sitä edeltävästä paikasta. Tämä voidaan tehdä riippumatta siitä kuinka pitkä kuljettava matka on, sillä reitillä ei voi olla oikoreittiä (Kuva 18).



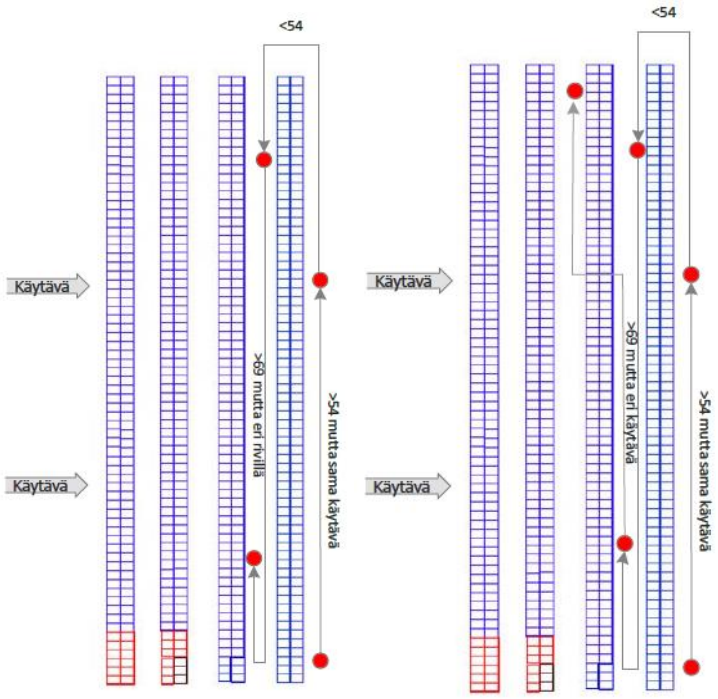
Kuva 18 Tapaus 1: tutkittava paikka ja sitä edeltänyt paikka ovat samalla rivillä.

Tapauksessa, jossa seuraavana keräiltävä tilausrivi on seuraavassa käytävävälissä, on huomioitava mahdolliset oikoreitit. Jos välimatka nykyisen ja edeltävän paikan välillä on alle 54 yksikköä, ei tarjolla ole lyhempää reittiä, joka olisi käytännössä järkevä. Yksi poikittaiskäytävien muodostama lohko siirryttävien käytävien välissä voi sisältää korkeintaan 54 (27+27) mahdollista paikkaa. Välimatkan ollessa 54 tai alle, on kustannus ensimmäisen tapauksen mukainen ja etenemistä jatketaan ilman oikoreitin käyttämistä (Kuva 19).



Kuva 19 Tapaus 2: tutkittava paikka ja sitä edeltänyt paikka ovat eri riveillä, mutta alle 54 yksikön päässä toisistaan.

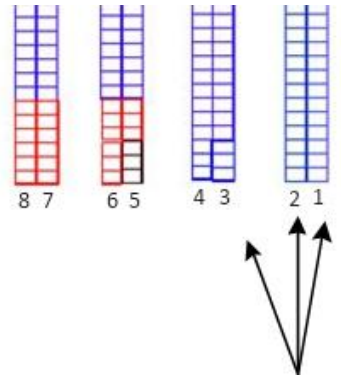
Mikäli välimatka on tätä pidempi tuotteiden ollessa eri puolilla käytävää, on lähes aina olemassa myös nopeampi reitti. Sanotaan, että tässä esimerkissä laskennallinen etäisyys on 76 yksikköä, mutta seuraava poimittava tuote on viereisellä rivillä samassa käytävävälissä (Kuva 20). Tässä tapauksessa etäisyysarviota voidaan korjata alaspäin, koska todellisuudessa on aina olemassa laskennallista välimatkaa lyhempi reitti. Laskennallisella välimatkalla tarkoitetaan varastopaikkojen numeroiden välistä erotusta. Oletetaan, että kyseessä on huonoin mahdollinen tapaus, jolloin tutkittavan rivin viimeisestä tuotteesta siirrytään seuraavan rivin ensimmäiseen tuotteeseen. Tässä tapauksessa kustannusta ei voida vähentää oikoreittien avulla, vaan se on 69 yksikköä. Jos seuraavalla käytävällä oleva tuote on yli 69 yksikön päässä (käytävän mitta) tarkoittaa se, että välikäytäviä hyödyntäen maksimietäisyys tuotteeseen on korkeintaan 69 yksikköä. Tässä tapauksessa pitää huomioida, että välikäytäviä käytettäessä on hetkellisesti sallittua liikkua kulkusuunnan vastaisesti. Tämän perusteella voimme laskea kustannusta asettamalla se 69:ään 76:n sijasta.



Kuva 20 Tapaus 3: tutkittava paikka ja sitä edeltänyt paikka ovat joko eri rivillä tai seuraavalla käytävällä, mutta yli 69 yksikön päässä toisistaan.

Tapaus, jossa etäisyys seuraavaan tuotteeseen on enemmän kuin yksi käytäväväli tarkoittaa, että käymättömät rivit voidaan jättää pois laskuista ja siten vähentää niiden laskennallinen kustannus ko. siirtymän kustannuksesta. Rivien välisiä etäisyyksiä kuvaavissa matriiseissa (Kuva 22) vihreät solut kuvaavat jo edellä käsiteltyjä tapauksia, jolloin seuraava keräiltävä tuote on joko samalla tai

sitä seuraavalla rivillä (seuraava rivi – kuljettava rivi < 3). Myös kahden hyllyrivin ero voidaan lukea käsitellyksi tapaukseksi, koska kuljettavan etäisyyden kannalta se on yhtä kaukana kuin yhden hyllypuoliskon päässä oleva rivi. Käytännössä samalla maksimikustannuksella voidaan tutkia toinen puoli nykyistä hyllyä tai seuraavan hyllyvälin lähempi hyllypuolisko (Kuva 21).

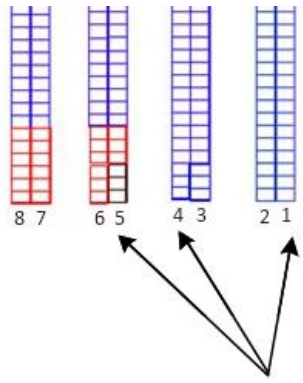


Kuva 21 Hyllyrivit 1-3.

	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7
2	1	0	1	2	3	4	5	6
3	2	1	0	1	2	3	4	5
4	3	2	1	0	1	2	3	4
5	4	3	2	1	0	1	2	3
6	5	4	3	2	1	0	1	2
7	6	5	4	3	2	1	0	1
8	7	6	5	4	3	2	1	0

Kuva 22 Hyllyrivien etäisyysmatriisi.

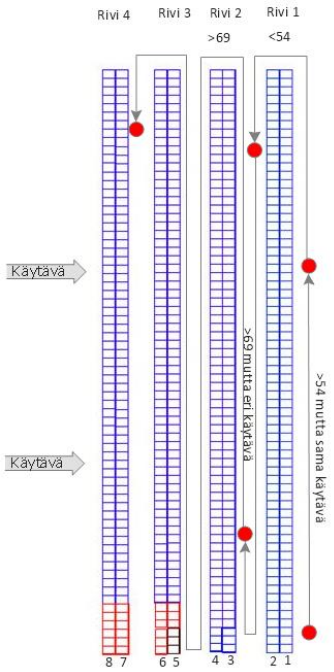
Eron ollessa 3 tai 4 riviä (seuraava rivi – kuljettava rivi = 3 tai 4), voidaan silloin varmuudella sanoa, että yksi hyllyväli voidaan jättää tutkimatta ja näin ollen kustannuksesta voidaan vähentää 138 (69+69) yksikköä (Kuva 23). Noudatettaessa S-mallista heuristiikkaa ilman ehtoa tutkimatta jättämisestä olisi hyllyvälissä vierailtu turhan takia (Kuva 25).



Kuva 23 Tutkittavat rivit 1,4 ja 5.

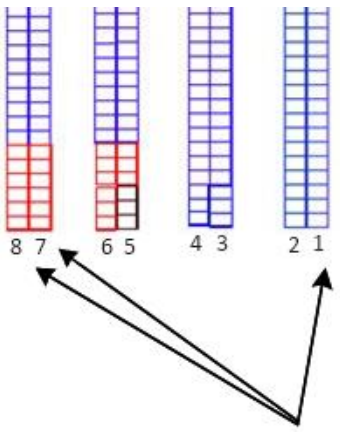
	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7
2	1	0	1	2	3	4	5	6
3	2	1	0	1	2	3	4	5
4	3	2	1	0	1	2	3	4
5	4	3	2	1	0	1	2	3
6	5	4	3	2	1	0	1	2
7	6	5	4	3	2	1	0	1
8	7	6	5	4	3	2	1	0

Kuva 24 Hyllyrivien etäisyysmatriisi.



Kuva 25 Tapaus 4: Kolmannessa käytävävälissä vieraillaan tarpeettomasti.

Vastaavasti eron ollessa 5 tai 6 hyllyrivin puolta, voidaan varmuudella jättää tutkimatta kaksi käytävävälіä. Tapauksessa 5 kustannuksesta vähennetään 276 yksikköä (4*69), koska kaksi käytävävälіä voidaan jättää tutkimatta. Eron ollessa yli 6 hyllyrivin puolta on kyseessä tapaus 6, jossa voidaan vähentää 3:n käytävävälin kustannus 414 (6*69). Keräilyalueen hyllypuoliskojen lukumäärän ollessa korkeintaan 8 ei muita mahdollisia tapauksia tarvitse tutkia. Simuloitaessa 20 satunnaista tilausta, tavattiin tapaukset 1 ja 2, mutta ei kertaakaan tapausta, jossa useita hyllyvälejä olisi jätetty väliin.



Kuva 26 Tutkittavat rivit 1,7 ja 8.

0	1	2	3	4	5	6	7	8
1	0	1	2	3	4	5	6	7
2	1	0	1	2	3	4	5	6
3	2	1	0	1	2	3	4	5
4	3	2	1	0	1	2	3	4
5	4	3	2	1	0	1	2	3
6	5	4	3	2	1	0	1	2
7	6	5	4	3	2	1	0	1
8	7	6	5	4	3	2	1	0

Kuva 27 Hyllyrivien etäisyysmatriisi.

Simuloinnissa käytetty arviointiheuristiikka on osin pessimistinen sen salliessa todellista pidemmän etäisyysarvion yhden käytävä kolmanneksen sisällä. Tämä vaikuttaa kokonaiskustannukseen tapauksessa jossa keräystehtävä päättyy tähän kolmannekseen ja tuote löytyykin arvioitua lähempää, jolloin voidaan palata takaisin. Lyhimmän mahdollisen siirtymän käyttämistä tärkeämpää arvioinnissa on se, että todellinen reitti ei voi koskaan olla pidempi, kuin arvioitu reitin pituus. Siinä missä lyhimmän reitin ongelmassa pätevyyden vaatimus on, ettei heuristinen arvio saa koskaan olla pidempi kuin todellinen reitti tai muuten ei voida sanoa lyhimmän reitin löytyneen, on tässä tapauksessa ongelma päinvastainen. Jos todellinen reitti voisi olla pidempi kuin simuloitu reitti, ei optimoidun sijoittelun toimivuutta voitaisi arvioida puolueettomasti. Täten heuristiikkaa voidaan pitää pätevänä tätä tehtävää varten. Heuristinen arvio ei ole tarpeettoman pessimistinen, sillä arvioinnissa otetaan huomioon myös mahdolliset oikoreitit sekä tapaukset, joissa useita käytäviä voi esiintyä ilman yhtään keräiltävää yksikköä. Tällöin ne voidaan vähentää etäisyysarviosta. Käytännössä nämä tapaukset, joissa ensimmäinen keräiltävä yksikkö sijoittuu radan alkuun ja seuraavat aivan sen loppuun, ovat hyvin harvinaisia.

Suoritettava algoritmi on ”epätietoinen” ympäröivästä tilasta, koska se pystyy näkemään ainoastaan kulloinkin tutkittavan ja sitä edeltäneen paikan. Simuloinnin suorittamiseksi on luotu raportti alkuperäisistä keräilypaikkojen varauksista kaikille varastopaikoille, jotka ovat mukana optimoinnissa. Tätä alustavaa järjestystä on käytetty ensimmäisessä simulointiajossa ja sen tulokset on tallennettu tuloksia keräävään muuttujaan. Tämän jälkeen simulointi päättyy eikä samaan tilaan enää palata takaisin. Myös optimoidulle tuoteasettelulle suoritetun ajon tulokset tallennetaan vastaavaan datan kerääjään jatkokäsittelyä varten.

Simulointiin on lisätty stokastisuutta sekoittamalla paikat ja ajamalla simulointi vielä uudelleen satunnaisille paikoille. Tämä havainnollistaa tilannetta, jossa tuotteet olisi sijoitettu keräilyalueelle täysin satunnaisesti, esimerkiksi varastoon saapumisen perusteella. Tässä mallissa tuotteiden sijoitteluun kiinnitetään huomiota vain siltä osin, että tuote löytyy aina jostakin keräilyalueelta eikä samalle tuotteelle ole käytössä useita varastopaikkoja.

Simulointiaineistona on käytetty kuukauden ajalta satunnaisesti valittuja myyntitilauksia. Aineistossa on yhteensä 108158 tilausriviä, joista simulointiin on käytetty 45191 riviä. Tämä vastaa 4683 kappaletta asiakastilauksia.

5.4 Tutkimuksen luotettavuuden arviointi

Tässä luvussa arvioidaan tämän tutkielman luotettavuutta. Tutkielman ollessa projektiluontoinen, luokitellaan se ensisijaisesti tapaustutkimukseksi. Tapaustutkimuksessa yksittäistapauksia pyritään tutkimaan niiden luonnollisessa ympäristössään. Tällaiset kuvailevat menetelmät eivät välttämättä pyri selittämään ilmiöiden välisiä yhteyksiä, testaamaan hypoteeseja tai tekemään ennusteita. Tavoitteena on pikemminkin tutkimuskohteen ominaispiirteiden systemaattinen, tarkka ja totuudenmukainen kuvailu. Tapaustutkimuksen teko ei rajoita käytettävissä olevia menetelmävalintoja. On kuitenkin olennaista, että tutkittava tapaus muodostaa jonkinlaisen kokonaisuuden, kuten tässäkin tutkielmassa käy ilmi. (Saaranen-Kauppinen & Puusniekka, 2006.) Aineiston hankintaan ja analysointiin käytetyt tutkimusmenetelmät ovat tyypiltään laadullisia eli kvalitatiivisia. Laadullisen tutkimuksen päämääränä on ennen kaikkea laajentaa ja tuoda uusia näkemyksiä vähän tutkitusta ilmiöstä (Naakka, 2018).

Tapaustutkimuksen valitseminen menetelmäksi on perusteltua, koska tutkimuksen kohdetta halutaan analysoida syvällisesti ja huomioida siihen vahvasti liittyvä konteksti (toimintaympäristö). Laadullisella tutkimuksella pyritään ymmärtämään varastossa tapahtuvaan keräilyyn vaikuttavia ominaisuuksia ja merkityksiä kokonaisvaltaisesti. Varsinaisten optimointitulosten arviointiin hyödynnetään simulointia eri toteutusten puolueettoman vertailun toteuttamiseksi. Laadullisen tutkimuksen luotettavuutta mitataan usein seuraavilla kriteereillä: tutkimuksen uskottavuus (credibility), siirrettävyys (transferability), riippuvuus (dependability) ja vahvistettavuus (confirmability) (Tuomi & Sarajärvi, 2009). Tämän tutkimuksen uskottavuus perustuu muun muassa laajaan kirjallisuuskatsaukseen, joka on toteutettu tutkittavan aiheen ympäriltä ja johon valitut toteutuksen menetelmät perustuvat.

Tutkielman siirrettävyys tutkimuskontekstin ulkopuoliseen vastaavaan kontekstiin riippuu vahvasti siitä, miten samankaltaisia ympäristöt ovat. Projektissa toteutetut algoritmit ja datan esivalmistelu ovat räätälöityjä kohdevaraston käyttöön, eivätkä siten ole siirrettävissä täysin uuteen ympäristöön ilman merkittäviä muutoksia. Siirrettävyys tutkimusympäristöstä yrityksen tuotantoympäristöön on mahdollista verrattain pienillä muutoksilla. Haasteen tuovat muutokset, jotka on toteutettu tuotantoympäristöön tutkimuksen aloittamisajankohdan jälkeen. Sekä algoritmien että aineiston käsittelyyn on hyödynnetty tutkijan laajaa henkilökohtaista tuntemusta kyseisestä ympäristöstä usean vuoden ajalta. Tämä voi osaltaan vaikeuttaa tutkimuksen siirrettävyyttä, mikäli sen toteuttaisi joku muu kuin tämän tutkielman toteuttaja.

Laadullisessa tutkimuksessa tulosten pysyvyyttä voidaan arvioida tutkielman riippuvuuden kautta. Tällä tarkoitetaan sitä, että tutkimus on toteutettu tieteellisen tutkimuksen toteuttamista

yleisesti ohjaavin periaattein. Arviossa tutkijan tulee ottaa huomioon paitsi erilaiset ulkoiset vaihtelua aiheuttavat tekijät, myös tutkimuksesta ja ilmiöstä itsestään johtuvat tekijät. (Tuomi & Sarajärvi, 2009.) Tutkielman luotettavuuden arviointiin on käytetty apuna haastatteluja, joihin on osallistunut kohdevaraston henkilöstöä sekä johdon että työntekijöiden osalta. Haastattelut ovat olleet vapaamuotoisia ja niissä pyritty tunnistamaan mahdollisia käytännön haasteita, joihin tutkimuksessa ei lähtökohtaisesti ole otettu kantaa. Oleellisimpina asioina esiin nousivat seuraavat aiheet:

- Onko poikkeustapausten havainnointiin käytetty aika ollut riittävää ja onko siten keräilyn mallinnuksessa huomioitu erot yksilöiden käyttäytymisen välillä?
- Tilausten yhdisteleminen on käytännössä mahdollista monella eri tavalla. Tulisiko algoritmin huomioida myös tilausyhdistelmät ja mitkä ovat niiden vaikutukset kokonaisuuden kannalta?
- Simulointiaineiston määrä on suhteellisen pieni verrattuna koko vuoden kattavaan aineistoon. Materiaalista ei siten käy ilmi mikä olisi kausivaihteluiden vaikutus lopputulokseen?

Haastattelujen avulla kerättyä tietoa on pyritty mahdollisuuksien mukaan huomioimaan projektin toteutuksessa. Kuitenkin edellä mainitut seikat ovat asioita, jotka vaativat laajempaa jatkotutkimusta niiden merkitsevyyden selvittämiseksi. Aihetta on käsitelty tarkemmin luvussa 7.1. Tutkimusaiheesta on melko vähän tieteellistä tutkimusta, jota kartoittavalla laadullisella tutkimuksella pyrittiin hankkimaan lisää. Koska toimintaympäristö on tällaisessa tapauksessa täysin uniikki, on toteutuksen käytännön soveltuvuudesta ja siihen liitetyistä haasteista haettu monipuolisempia näkemyksiä juuri haastattelujen avulla.

Tutkimuksessa pureudutaan vaiheittain tutkittavan ilmiön laajemmista teemoista suppeampiin ja jokaisen luvun kohdalla on kerrottu, mitä seuraavaksi tullaan käsittelemään. Asiat on pyritty esittämään riittävän seikkaperäisesti, jotta lukija pystyy seuraamaan tutkijan päättelyä ja arvioimaan sitä. Myös lähdekirjallisuudessa on pyritty keskittymään mahdollisimman tuoreisiin ja ajan tasalla oleviin lähteisiin.

6 Tulokset

Tässä osassa käydään läpi tutkielman pääasialliset löydökset ja poikkeamien vaikutukset tuloksiin sekä arvioidaan projektin onnistumista.

6.1 Tulokset lukuina

Tuloksien osalta on oleellista tietää, että kaikki eri sijoitteluvaihtoehdot on arvioitu käyttämällä samaa simulointimallia ja heuristiikkaa eri simulointikierroksilla.

Simuloinnissa käytettyjen tilausten määrä	**Alkuperäinen asettelu (kokonaiskustannus)**	**Satunnainen asettelu (kokonaiskustannus)**	**Optimoitu asettelu (kokonaiskustannus)**
100 kpl	30438	31696	24929
100 kpl (uutta tilausta)	29932	32975	23006
2000 kpl	678963	633032	563859
2000 kpl (uudelleen simulointi samalle aineistolle)	678963	639737	563859
4683 kpl	1354412	1494194	1172798

Taulukko 2 Simuloinnin tulokset lukuina

100 tilausta: kokonaiskustannus oli alkuperäisellä asettelulla 22,10 % optimoitua lajittelua korkeampi ja satunnaisella lajittelulla 27,15 % optimoitua korkeampi.

100 uutta tilausta: kokonaiskustannus oli alkuperäisellä asettelulla 30,11 % optimoitua lajittelua korkeampi ja satunnaisella lajittelulla 43,33 % optimoitua korkeampi. Eri materiaalilla simuloitaessa voidaan silminnähden havaita varianssin olevan suurta. Tästä johtuen simulointiaineistoa tulee kasvattaa seuraavia kierroksia varten.

2000 tilausta: kokonaiskustannus oli alkuperäisellä asettelulla 20,41 % optimoitua lajittelua korkeampi ja satunnaisella lajittelulla vain 12,27 % optimoitua korkeampi. Mielenkiintoinen havainto on, että tuotteiden satunnainen sijoittelu näyttää tällä kertaa olevan parempi kuin alkuperäinen. Tässä tapauksessa ainesto on syytä käsitellä uudestaan, jotta sijoittelun satunnaisuus voidaan vahvistaa.

2000 tilausta uudelleen simuloituna: kokonaiskustannus oli alkuperäisellä asettelulla edelleen 20,41 % optimoitua lajittelua korkeampi ja satunnaisella lajittelulla 13,46 % optimoitua korkeampi. Käytettäessä samaa aineistoa uudelleen, nähdään odotetusti muutos ainoastaan satunnaisessa sijoittelussa. Tulosten suuren vaihtelun vuoksi on syytä suorittaa simulointi uudestaan suuremmalla aineistolla.

4683 tilausta: kokonaiskustannus oli alkuperäisellä asettelulla 15,49 % optimoitua lajittelua korkeampi ja satunnaisella lajittelulla 27,40 % optimoitua korkeampi. Simuloitaessa yhä suuremmalla aineistolla, eivät tulokset enää muuttuneet yhtä radikaalisti aiempiin kierroksiin verrattuna. Simulointi päätettiin lopettaa tässä vaiheessa, koska havaittujen muutosten ei katsottu enää olevan merkittäviä.

6.2 Poikkeamien vaikutukset tuloksiin

2000 tilauksen joukosta löytyi yhteensä 142 tuotetta, joille ei löytynyt paikkaa optimoitujen paikkojen joukosta. 34,51 % (49kpl) näistä tuotteista olivat keskimääräiseltä myynniltään 0-17 laatikkoa kuukaudessa. Loput paikattomista tuotteista eli 65,49% (92kpl) olivat volyymiltaan 17-81 laatikkoa kuukaudessa. Määriä voidaan pitää optimoinnin kannalta lähes merkityksettöminä, sillä ko. kuukauden kokonaisvolyymi oli n. 500 000 laatikkoa, jolloin suurimman poisjääneen tuotteen menekki olisi 0,02% koko kuukauden toimituksista. Optimoidusta asettelusta pois jääneistä tuotteista 139:lle löytyi kuitenkin paikka alkuperäisessä asettelussa. Nämä tuotteet käsitellään simuloinnissa kappaleen 4.3. kuvaamalla tavalla, jotta mahdollisen puolueellisuuden vaikutus saadaan huomioitua arvioinnissa. Loput 3 tuotetta puuttuvat molemmista asetteluista, joten ne on rajattu pois simuloinnista.

Jotta arvioinnista saadaan riittävän puolueeton, on tehty oletus siitä, että kaikille tuotteille löytyy paikka radalta laskettaessa keräyksen kustannus sitten alkuperäisellä tai optimoidulla sijoittelulla. Tällä olettamuksella on tarkastettu missä varastopaikassa tilausriviltä löytyvä tuote on nykyisessä sijoittelussa. Mikäli tuotteelle ei löydy paikkaa nykyiseltä radalta rajataan se ulos käsittelystä yllä esitetyn matalan menekin perusteella. Näin ollen on tilauksilta suodatettu pois käsittelystä

alkuperäisestä sijoittelusta puuttuvat tuotteet. Mikäli optimointialgoritmi lisää sellaisen tuotteen keräysalueelle ei sitä varastopaikkaa huomioida simuloinnissa, koska tuote on rajattu pois tilauksilta. Tämä voi aiheuttaa simuloinnissa puolueellisuutta alkuperäisen asettelun suuntaan, koska optimoidulla asettelulla voi tässä harvinaisessa tapauksessa olla käytössä paikka, jota ei koskaan tutkita sen tuotteen tuotteen puuttuessa alkuperäisestä asettelusta. Edellä mainittujen 2000 tilauksen joukossa tällaisia tuotteita ei löytynyt ainuttakaan.

On kuitenkin huomattava, että optimointia voidaan helposti laajentaa koskemaan myös laajempaa aluetta tai tarpeen kasvaessa myös erillisille hyllyille voidaan suorittaa optimointi, ottaen mukaan vain tuotteet ja paikat, jotka jäivät nyt toteutetun optimoinnin ulkopuolelle. Vähäisen menekin tuotteet, joille ei löytynyt sopivaa paikkaa sijoitettaisiin käytännössä hyllyjen päädyssä oleville matalammille hyllyille, joissa on useita kerroksia. Nämä eivät ole osa keräilyrataa, mutta sijaitsevat radan alussa ja lopussa, jolloin tuotteet ovat helposti matkalla poimittavissa eikä keräystä voi suorittaa niitä ohittamatta.

Sekä nykyisessä käytännössä että lisähyllyyn perustuvassa vaihtoehdossa, tuotteet olisivat saatavilla sellaisella etäisyydellä, joka ei ainakaan kasvata nykyistä kustannusarviota, vaan ne poimitaan mukaan radan varrelta paikoista, jotka keräilijä ohittaa väistämättä. Toki ongelmaksi voi muodostua hyllyjen kapasiteetti, mikäli aktiivisten tuotteiden määrä kasvaa rajusti, jolloin tuotteet joudutaan sijoittamaan kauemmaksi radasta ja varastoimaan lattialla. Tämä väistämättä vaikuttaisi myös keräyskustannuksiin. Lisäksi on olemassa mahdollisuus, että asiakas tilaa tuotetta, jota ei ole asetettu millekään paikalle tai jonka myyntiennuste on 0 eikä sitä siten ole sijoitettu mihinkään. Nämä tapaukset ovat kuitenkin kokonaisuuden kannalta merkityksettömiä poikkeuksia, jotka eivät vaikuta oleellisesti simuloinnissa saatuihin tuloksiin.

6.3 Projektin onnistuminen

Projektilla ei ollut onnistumisen kriteereinä taloudellisia tai aikataulullisia mittareita. Projektin toteutuksen onnistumista voidaan kuitenkin arvioida sen toimitusten perusteella, suhteessa projektin tavoitteeseen. Projekti onnistui vastaamaan kysymykseen voiko keräilyprosessia tehostaa optimoimalla tuotteiden sijoittelua nykyisellä keräilyalueella. Simuloinnissa saatujen tulosten valossa tämä on hyvinkin mahdollista ja se antaa lupaavan lähtökohdan varaston kehitystyön aloittamiselle. Projektissa toimitettiin onnistuneesti myös prosentuaaliset arviot siitä, kuinka merkittävä hyöty uudella sijoittelulla voidaan saavuttaa nykytilanteeseen verrattuna. Optimoinnilla saavutettu hyöty on myös onnistuttu muuttamaan potentiaalisiksi

kustannussäästöiksi. Kuten tuloksista käy ilmi, optimoitu asettelu voi vähentää liikkumiseen käytettyä matkaa nykyiseen verrattuna jopa 15% ja täysin satunnaiseen asetteluun verrattuna jopa 27%. 15 prosenttiyksikön säästö kuljetussa matkassa on kustannussäästöiksi muunnettuna noin 4,5 % keskimääräisen varaston kaikista operatiivisista kustannuksista.

7 Johtopäätökset ja yhteenveto

Kohdeyrityksen varasto ja sen keruualue ovat laajat kattaen satoja lavapaikkoja. Optimaalisen keräilyreitin tai tuotesijoittelun suunnittelu ilman teknistä toteutusta voi tällöin osoittautua hankalaksi. Tuotesijoitteluun suunniteltu optimointialgoritmi mahdollistaa valittujen parametrien mukaan optimoidun keräilyreitin suunnittelun. Käytännön toteutuksessa tulisi ottaa huomioon myös sijoittelun ylläpidettävyys. Tämä voi olla haasteellista volyymiperusteisessa mallissa, johon vaikuttavat esimerkiksi kausittaiset vaihtelut.

Käsiteltävän tuoteportfolion homogeenisyys mahdollistaa koko keräilyalueen kattavan volyymiperusteisen sijoittelun käyttöönoton. Keräilyalueelle voidaan varastoida tuotteet niiden laatikkokoon ja kiertonopeuksien mukaiseen järjestykseen. Tämän järjestyksen kertoo toteutettu algoritmi. Historiallisesti katsottuna esimerkkiyrityksen kysyntä vaihtelee vuodenajoittain suuresti, joten optimoitua järjestystä tulee uudelleenarvioida säännöllisesti, aina ennusteiden muuttuessa.

Tutkielman tavoitteena oli tarkastella eri optimointimallien ja menetelmien soveltuvuutta keräystyössä kuljetun matkan optimointiin. Aluksi tutustuttiin varastossa liikkumiseen ja saatavilla olevaan materiaaliin sekä siitä tunnistettaviin piirteisiin. Keräilyprosessia ja tuotteiden sijoittelumenetelmiä tutkittiin yleisesti eri näkökulmista. Keräilyn todettiin muodostavan ison osan varaston kustannuksista ja keräilyyn kuluvasta ajasta suurimman osan menevän liikkumiseen paikkojen välillä. Tuotteiden sijoitteluvaihtoehdoista volyymiin perustuvan sijoittelumenetelmän todettiin olevan tehokkain keino vähentää keskimääräistä liikkumiskustannusta. Tehokkaan volyymipohjaisen sijoittelun löytämisen lisäksi ongelmaa tarkasteltiin myös reitin optimoinnin näkökulmasta. Ensin kauppamatkustajan ongelman, sen käänteisen version sekä muiden muunnelmien kautta, samalla tutkien millaisia ratkaisuja ongelmiin on löydetty. Ratkaisuun sopivista menetelmistä esiteltiin neljä erilaista vaihtoehtoa ja niiden soveltuvuus eri tilanteissa. Näitä vaihtoehtoja olivat heuristiset menetelmät, neuroverkot, rivijärjestyksen optimointi sekä paritusalgoritmit.

Toteutusvaihtoehtojen selvittämisen jälkeen esiteltiin valittu menetelmä ja sen oletetut vaikutukset lopulliseen ratkaisuun. Mahdollisuus tehdä paljon esivalmisteluja jo tunnetulle datalle johti siihen, että toteutuksessa päädyttiin hyödyntämään perinteistä algoritmiikkaa ja käsittelemään aineistoa listoina. Osoittautui myös, ettei neuroverkkoja hyödyntävän ratkaisun sinänsä raskasta rakennetta tarvittu, vaikka sillä oletetusti voitaisiin parantaa algoritmin suorituskykyä muuntamalla se ns. mustan laatikon ongelmaksi. Tämä olisi tullut kyseeseen,

mikäli algoritmi olisi toteutettu esimerkiksi matemaattisena mallina, jonka aikakompleksisuus olisi osoittautunut prosessoinnin pullonkaulaksi.

Projektiosuudessa tarkasteltiin kohdeyrityksen varastoa, nimikkeitä, keräilyprosessin nykytilaa ja keräilyreitin optimoinnin problematiikkaa käytännön näkökulmasta. Havaittiin, että nimikkeet voidaan automaattisesti jakaa keräilyalueelle volyymiperusteisesti, mutta vain siten, että ne täyttävät käytännön sanelemat ehdot muun muassa pakkauskoon mukaisesta lajittelusta. Käytännön toteutukseen pureuduttiin tarkemmin neljännessä osassa, jossa käsiteltiin optimointialgoritmin toimintaa ja sen testaamista simuloinnilla. Simulointiin sovellettiin heuristisista menetelmistä juonnettua mallia, jossa lasketaan etäisyysarvio liikkumiseen käytettävästä matkasta jokaisen tilausrivin kohdalla. Arviossa on huomioitu nykyinen ympäristö ja siinä liikkumista rajoittavat tekijät. Samaa heuristiikkaa käytetään simuloitaessa eri malleja, jotta saadaan aikaan keskenään vertailukelpoiset tulokset nykyisen, satunnaisen ja optimoidun sijoittelun välille. Tällä tavalla voidaan arvioida algoritmin toimintaa juuri kyseisessä varastoympäristössä.

Viidennessä osassa käsiteltiin tutkimusmenetelmiä yleisesti ja etenkin simuloinnin käyttämistä tulosten arvioinnissa. Viimeisessä luvussa arvioitiin tutkimuksen luotettavuutta ja sitä mitkä seikat siihen vaikuttavat. Kuudes osa tiivistää tutkielman tulokset luvuiksi, käyden samalla läpi tutkielman pääasialliset löydökset. Tässä osassa arvioidaan myös poikkeamien vaikutuksia tuloksiin ja sitä, miten projektissa onnistuttiin. Optimoinnilla saavutettu 15 % parannus alkuperäiseen sijoitteluun on parempi kuin mitä osattiin odottaa. Erityisesti, kun ottaa huomioon, että täysin satunnaisen sijoittelun ja nykyisen sijoittelun välinen ero on vain noin 12 % nykyisen sijoittelun hyväksi. Siten tuloksista voidaan tulkita, että siirryttäessä algoritmin ehdottamaan sijoitteluun, voidaan saada jopa vastaavaa hyötyä, kuin siirryttäessä satunnaisesta mallista varaston nykyiseen tilanteeseen.

7.1 Jatkotutkimuskohteet

Tämän tutkielman ja siihen liittyvän projektin jatkotutkimuskohteet tulevat pääosin projektin aikana tunnistetuista puutteista ja/tai projektin laajuudesta pois rajatuista asioista. Projektin aikana huomattiin, että jo verrattain pienetkin muutokset volyymissä vaikuttavat voimakkaasti tuotteiden sijoittumiseen radalla. Jotta optimointi toimisi käytännön kannalta parhaiten, vaatii se säännöllistä seurantaa ja tuotteiden uudelleen järjestelyä. Ajatuksena on, että uudet ja menekiltään kasvavat tuotteet vaihdetaan ketterästi paremmille paikoille, samalla kun menekiltään hiipuvat ja kokonaan lopetetut tuotteet siirtyvät huonommille paikoille tai kokonaan pois keräilyalueelta. Tarvittavia

vaihtoja varten olisi luontevaa toteuttaa työkalu, jota käytettäisiin esimerkiksi kerran kuukaudessa tai aina ennusteiden päivittyessä. Siihen, mitkä ovat oikeat raja-arvot laukaisemaan varastopaikan vaihtotarpeen, tarvitaan lisätutkimusta. Ei ole mielekästä muuttaa järjestystä, mikäli tuotteen kuukausivolyymi muuttuu esimerkiksi prosentin ylös- tai alaspäin, vaikka viereinen tuote olisi silloin oikeutettu paremmalle paikalle.

Kehitetylle algoritmille löytyy moninaisia jatkokehityskohteita. Potentiaalisimpina kehityskohteina voisivat toimia esimerkiksi eri myyntikanavien tai koko varaston kierron huomioiminen osana sijoittelun optimointia. Mallista voisi saada enemmän hyötyä, mikäli se pystyisi ottamaan huomioon lisämääreitä, kuten saapuvan tavaran sijoittuminen suhteessa lähtevien tavaroiden käyttämään keräilypaikkaan. Jatkokehittämällä voidaan siten huomioida myös keräilyalueen ulkopuoliset hyllyrivit, joissa tavaraa varastoidaan ja josta se siirretään edelleen keräilyalueelle.

Tässä tutkimuksessa keskityttiin arvioimaan tuotteiden sijoittelun vaikutusta liikkumiseen käytettyyn aikaan. Erityisesti sitä, miten tuotteiden sijoittelulla annetulle reitille on vaikutusta kuljettavaan matkaan ja näin ollen myös keräystehtävään käytettyyn aikaan. Jatkoa ajatellen voisi olla hyödyllistä tutkia, miten kahdesta kerroksesta tapahtuva keräily, erilaiset hyllyratkaisut, keräystehtävien koko, aloitus- ja lopetus pisteiden sijainti ym. paikalliset muutokset vaikuttaisivat keräilyyn käytettyyn aikaan ja radan tuotesijoitteluun.

Optimoinnissa tulisi huomioida tuotekanavakohtainen myynti, jotta keräilyalue voitaisiin järjestää tukemaan eri kanaviin kohdistuvaa myyntiä tehokkaammin. Esimerkkitapauksessa, jossa asiakas tilaa vientitoimituksen, olisi vientiin suunnatut tuotteet sijoitettu tietylle alueelle. Myös yksittäisen paikan kapasiteetin huomioiminen sekä sovelluskohdekohtaisten rajoitteiden ja hienosäädön lisääminen jo optimointivaiheessa olisi mahdollista toteuttaa.

Viimeisenä jatkokehityskohteena mainittakoon ohjelman laajennus graafisella käyttöliittymällä python-flask-html tai python-django-rest toteutuksella. Graafinen käyttöliittymä lisäisi huomattavasti käyttömukavuutta ja olisi loppukäyttäjän kannalta helpommin omaksuttavissa. Tässä vaiheessa voisi toteuttaa myös mahdollisuuden ennustaa optimaalinen paikka vain yhdelle uudelle tuotteelle sille annettujen parametrien mukaisesti.

Käytännössä tätä toteutusta voisi laajentaa hyvin pitkälle, mutta on hankala arvioida mikä on käytännön kannalta yleistyvyyden raja, jottei pieni muutos reaalimaailmassa riko koko ohjelman toimintaperiaatteita. Lisäksi haasteen aiheuttaa jatkuvat muutokset varaston pohjapiirroksessa. Aktiivisella ja jatkuvalla kehittämisellä voitaisiin haasteeseen vastata ja saada aiempaa parempia tuloksia.

Lähteet

Anaconda, 2019. Anaconda Distribution - The World's Most Popular Python/R Data Science Platform. Luettavissa: https://www.anaconda.com/distribution/. Haettu 01.11.2019.

Baka, B., 2017. Python Data Structures and Algorithms. Published by Packt Publishing Ltd, Livery Place, Birmingham, UK.

Bartholdi, J. J. & Hackman, S. T., 2014. Warehouse & Distribution Science. Georgia Institute of Technology, School of Industrial and Systems Engineering, The Supply Chain and Logistics Institute. August 22 2011, latest release: version 0.95.

Benson, D., 2020. Location Numbering in Warehouses. Luettavissa: http://www.wearethepractitioners.com/index.php/topics/warehouse-operations/functions-terms/location-numbering-warehouses. Haettu 22.3.2020.

Boschetti, A. & Massaron, L., 2016. Python Data Science Essentials. Published by Packt Publishing Ltd., Birmingham, UK, Second Edition p. 25-26.

Bottani E., Montanari R. & Rinaldi M., 2017. A flexible tool for warehouse design and picking optimization. Italy, XXII Summer School "Francesco Turco" – Industrial Systems Engineering. Luettavissa: http://www.summerschool-aidi.it/edition-2017/cms/extra/papers/16-%20Bottani%20et%20al-with-numbers.pdf. Haettu 01.11.2019.

Chiarandini, M., 2008. HEURISTICS AND LOCAL SEARCH ALGORITHMS FOR COMBINATORIAL OPTIMZATION. SDU, Department of Mathematics and Computer Science, Lecture 5, Construction Heuristics, Luettavissa: https://imada.sdu.dk/~marco/Teaching/Fall2008/DM811/Slides/DM811-lec5-2x2.pdf. Haettu 6.4.2020.

Chung, Y. & Demange, M., 2008. Some Inverse Traveling Salesman Problems. Elsevier B.V., Electronic Notes in Discrete Mathematics 30 (2008) 9–14, doi:10.1016/j.endm.2008.01.003.

De Koster, R., Le-Duc, T. & Roodbergen, K.J., 2007. Design and control of warehouse order picking: a literature review. European Journal of Operational Research 182(2), p. 481-501.

Dooley, K., 2002. "Simulation research methods," Companion to Organizations. Joel Baum (ed.), London: Blackwell, p. 829-848.

Gale, D. & Shapley, L. S., 1962. College Admissions and the Stability of Marriage. American Mathematical Monthly. 69 (1): 9–14. Doi: 10.2307/2312726. JSTOR 2312726.

Hansen, P., Mladenović, N. & Perez, J.A.M., 2010. Variable neighbourhood search: methods and applications. Annals of Operations Research. 175: p. 367–407.

Henderson, D., Jacobson, S. H. & Johnson, A. W., 2006. The Theory and Practice of Simulated Annealing. ResearchGate, DOI: 10.1007/0-306-48056-5_10, Luettavissa:

https://www.researchgate.net/publication/225260290_The_Theory_and_Practice_of_Simulated_Annealing. Haettu 31.10.2020.

Hirsjärvi, S., Remes, P. & Sajavaara, P., 2009. Tutki ja kirjoita. 15. uudistettu painos, Kariston Kirjapaino Oy, Hämeenlinna.

Holste, C., 2009. Logistics News: Designing the Most Effective Order Pick Routing in the DC. Luettavissa: http://www.scdigest.com/assets/Experts/Holste_09-07-02.php. Haettu 01.11.2019.

HUB Logistics, 2009. Hankintahetki. Julkaisu 3/2009 s.4-5. Luettavissa: https://issuu.com/hublogistics/docs/2009_hankintahetki_3. Haettu 10.11.2019.

JavaTPoint, 2020. Hill Climbing Algorithm in Artificial Intelligence. Luettavissa: https://www.javatpoint.com/hill-climbing-algorithm-in-ai. Haettu 8.1.2020.

Jünger, M., Reinelt, G. & Rinaldi, G., 1995. Chapter 4 The Traveling Salesman Problem. Handbooks in Operations Research and Management Science volume 7: Network Models, North-Holland Elsevier, 1995.

Kalatie, T., 2016. Simulaatiomallien hyödyntäminen varastotoiminnoissa. Kandidaatintyö. Lappeenrannan teknillinen yliopisto, tuotantotalous.

Kazuo, I. & Shuichi, M., 2008. "A Survey of the Stable Marriage Problem and Its Variants" (PDF). International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008): p. 131–136. doi:10.1109/ICKS.2008.7. HDL: 2433/226940. ISBN 978-0-7695-3128-1.

Karhunen, J., Pouri, R., & Santala, J., 2004. Kuljetukset ja varastointi – järjestelmät, kalusto ja toimintaperiaatteet. Suomen Logistiikkayhdistys ry.

Kivinen, J., 2008. Tietorakenteet. Luentoviikko 11, University of Helsinki. Luettavissa: https://www.cs.helsinki.fi/u/jkivinen/opetus/tira/k08/viikko11.pdf. Haettu 11.7.2020.

Le-Duc, T. & De Koster, R., 2007. Travel time estimation and order batching in a 2-block warehouse. European Journal of Operational Research. Luettavissa: https: //www.sciencedirect.com/science/article/abs/pii/S0377221705006338. Haettu 15.11.2019.

Logistiikan maailma, 2020a. PIENTAVARAKERUU JA AUTOMAATIO. Luettavissa: https://www.logistiikanmaailma.fi/huolinta-terminaalit/logistiikkakeskus/pientavarakeruu-ja-automaatio/. Haettu 18.11.2020.

Logistiikan maailma, 2020b. VARASTONHALLINTAJÄRJESTELMÄT. Luettavissa: http://www.logistiikanmaailma.fi/logistiikka/ohjausjarjestelmat/varastonhallintajarjestelmat/. Haettu 13.4.2020.

Luotonen, M., 2017. KERÄILYTEHOKKUUDEN OPTIMOINTI – 3PL-varaston kehittäminen. Turku AMK, Opinnäytetyö (AMK) Auto- ja kuljetustekniikka.

MHI, 2019. Order Picking: Pick Sequencing and Batching. Material Handling Industry. Luettavissa: www.mhi.org/downloads/learning/cicmhe/lectures/orderpicking.ppt. Haettu 01.11.2019.

Miettinen, K., 2009. ”Miksi kompromissi on parempi kuin optimi? Uusia monitavoiteoptimoinnin menetelmiä päätöksen tekoon.”. Luettavissa: https://docplayer.fi/16248143-Miksi-kompromissi-on-parempi-kuin-optimi-uusia-monitavoiteoptimoinnin-menetelmia-paatoksentekoon.html. Haettu 01.4.2020.

Moeller, K., 2011. Increasing warehouse order picking performance by sequence optimization. Procedia - Social and Behavioral Sciences., vol. 20, p. 177-185. Luettavissa: https://www.sciencedirect.com/science/article/pii/S1877042811014030. Haettu 10.12.2019.

Naakka, S., 2018. ”BIG DATA KILPAILUETUNA SUORAHAKU-KONSULTOINNIN EHDOKASHAUISSA - Enemmän, nopeammin ja parempia IT-osaajia big datalla?”. Markkinoinnin pro gradu –tutkielma, Turun kauppakorkeakoulu.

Ognjanovski, G., 2019. Everything you need to know about Neural Networks and Backpropagation — Machine Learning Easy and Fun. Towards data science. Luettavissa: https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a. Haettu 31.10.2020.

Petersen, C.G., Aase, G.R. & Heiser D.R., 2004. Improving order-picking performance through the implementation of class-based storage. International Journal of Physical Distribution & Logistics Management; 2004; 34, 7/8.

Puranen, T., 2012. Reitinoptimointiongelma, sen variantit ja ratkaisumenetelmät. University of Jyväskylä. Luettavissa: http://users.jyu.fi/~jhaka/ldo/TIEA382kalvot_tuukka.pdf. Haettu 7.1.2020.

Python, 2019. The Python Standard Library. Luettavissa: https://docs.python.org/3.7/library/. Haettu 10.12.2019.

Roodbergen, K.J. & De Koster, R., 2000. Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research 133 (2001) p. 32-42.

Roodbergen, K.J. & De Koster, R., 2001. Routing methods for warehouses with multiple cross aisles. International Journal of Production Research 39(9), p. 1865-1883.

Rouse M., 2015. garbage in, garbage out (GIGO). Luettavissa: https://searchsoftwarequality.techtarget.com/definition/garbage-in-garbage-out. Haettu 10.12.2019.

Ruohonen, K., 2013. Graafiteoria, Tampereen teknillinen yliopisto, Luettavissa: http://math.tut.fi/~ruohonen/GT.pdf. Haettu 11.7.2020.

Saaranen-Kauppinen, A. & Puusniekka, A., 2006. KvaliMOTV - Menetelmäopetuksen tietovaranto [verkkojulkaisu]. Tampere: Yhteiskuntatieteellinen tietoarkisto [ylläpitäjä ja tuottaja]. Luettavissa: https://www.fsd.tuni.fi/menetelmaopetus/>. Haettu 11.7.2020.

Shannon, R., 1975. Systems simulation: the art and science. USA, Prentice-Hall.

Seward, C., 2015. Optimizing Warehouse Operations with Machine Learning on GPUs. Luettavissa: https://devblogs.nvidia.com/optimizing-warehouse-operations-machine-learning-gpus/. Haettu 3.1.2019.

Spyder, 2019. The Scientific Python Development Environment. Luettavissa: https://www.spyder-ide.org. Haettu 10.12.2019.

Teknillinen Korkeakoulu, 2020. ”Simulointi - luento11.ppt.”. Liikenneteorian perusteet – Kevät 2006, Tietoverkkolaboratorio, Aalto-yliopisto. Luettavissa: http://www.netlab.tkk.fi/opetus/s381145/k06/luennot/luento11_1.pdf. Haettu 20.9.2020.

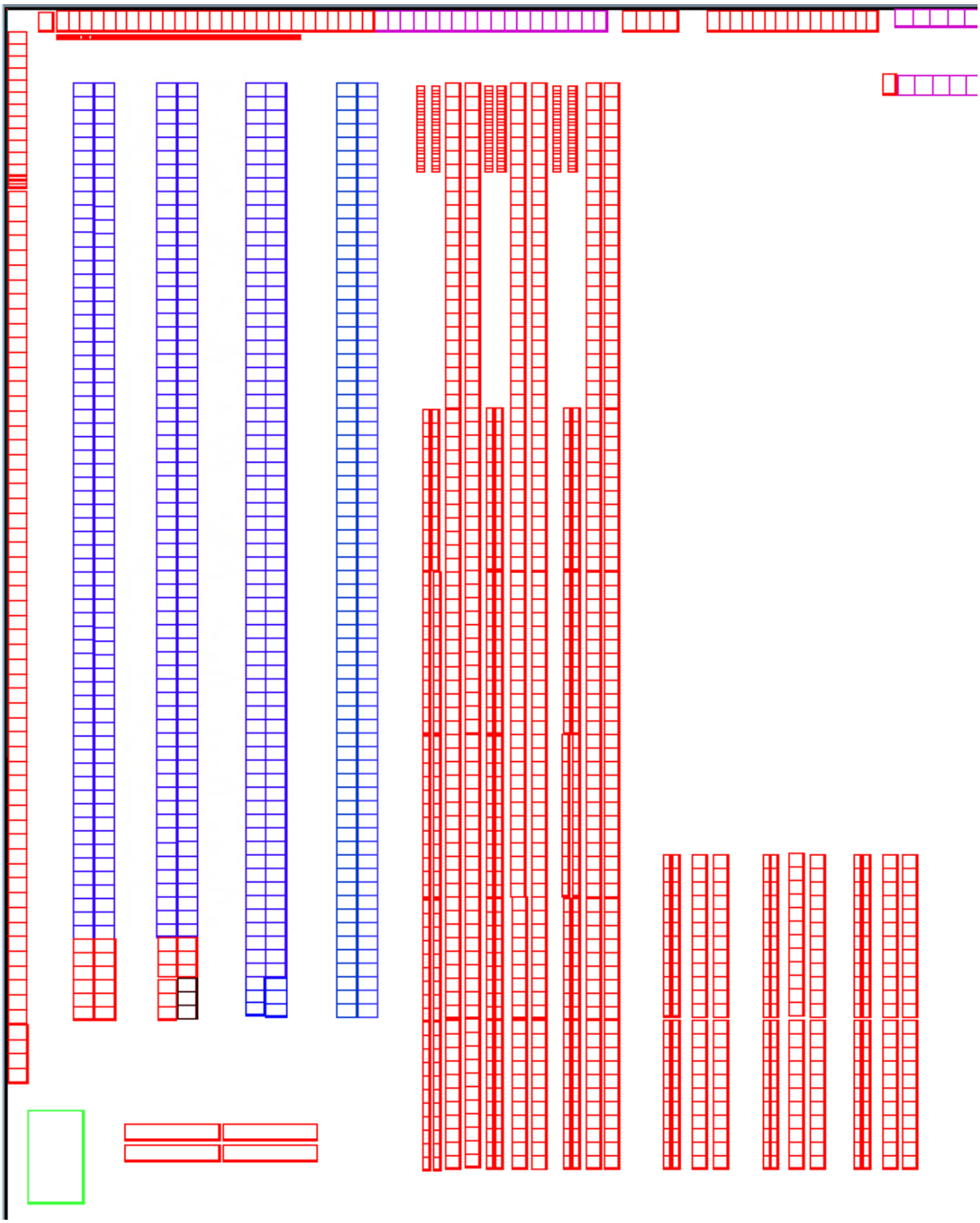
Torikka, O., 2015. A*-algoritmi ja siihen pohjautuvat muistirajoitetut heuristiset reitinhakualgoritmit. Pro gradu –tutkielma, ITÄ-SUOMEN YLIOPISTO Tietojenkäsittelytieteen laitos.

Tuomi, J. & Sarajärvi, A., 2009. Laadullinen tutkimus ja sisällönanalyysi. 6. painos. Kustannusosakeyhtiö Tammi, Helsinki.

Universiteit Utrecht, 2019. The Travelling Salesperson Problem – Algorithms and Networks. Luettavissa: cs.www.uu.nl › docs › vakken › an-tsp. Haettu 3.1.2020.

Liitteet

Liite 1: Varastonhallintajärjestelmän visualisoima pohjapiirros



Liite 2: Paikkojen sijoittelu algoritmin toteutus python koodina

```
"""
Created on Wed May  2 09:35:37 2019

@author: mahfi
"""

# loading libraries
import pandas as pd
from matplotlib import pyplot as plt
import csv
import time
from contextlib import contextmanager


timestamp = time.asctime( time.localtime(time.time()) )
HEADER = "Logfile created - Matti Ahonen \n"
FOOTER = "\nLogfile end \n"
FOOTER2 = (timestamp)
counter = 1
filename = "Logfile"
str_filename = filename,counter


class Main:

    #=========================================
    # Initialize the class                   #
    #=========================================


    def __init__(self): # alustusmetodi muuttaa funktiot metodeiksi "Initializer"
        #Print document details
        print(__doc__)
        plt.style.use('default')
        #self.main = main.lataa_nimikkeet()
```

```
#=============================================
# Handle exceptions                          #
#=============================================

try:
    content = open("..\WarehouseOptimization\input\AX_item.csv").read()

except IOError:  # Works in both Python 2 & 3
    print("Tiedostoa ei pystytä lukemaan!")
try:
    content = open("..\WarehouseOptimization\input\Item_Volume.csv").read()

except IOError:  # Works in both Python 2 & 3
    print("Tiedostoa ei pystytä lukemaan!")
try:
    content = open("..\WarehouseOptimization\input\locations3.csv").read()

except IOError:  # Works in both Python 2 & 3
    print("Tiedostoa ei pystytä lukemaan!")


#=====================================================
# Create logfile about the usage of the program      #
#=====================================================
@contextmanager
def new_log_file(name): #takes obj as an argument
        try:
            logname = name
            f = open(logname, 'w')
            f.write(HEADER)
            yield f #yied rather than return tells for Python this is generator
        finally:
            f.write(FOOTER)
            f.write(FOOTER2)
            print("Log file created ", timestamp)
            f.close()



def import_file(tiedostonimi,names):
    print(tiedostonimi," loaded")
    item_ds = pd.read_csv(tiedostonimi, header=None,delimiter=';',names = names, skiprows=1)
    return item_ds


#=====================================================
# Set constants                                      #
#=====================================================

#column names
item_names = ['Id','Item number','Item group','Item type','Alcohol vol','Litres','Pallet quantity','Weight','Weight2','Sales unit','Activity Class Code','Client group']
volume_names = ['Id','Item number','Volume','ABC','PWeight','PWeight2','Size']
location_names = ['Id','Location','Alias','Qty','Postfix','Weight','MinCaseSize','MaxLoading (1-3)','Channel','MultiPallet']


#item_dataset = import_file('..\WarehouseOptimization\input\AX_item.csv',item_names)
volume_dataset = import_file('..\WarehouseOptimization\input\Item_Volume2.csv',volume_names)
location_dataset = import_file('..\WarehouseOptimization\input\locations3.csv',location_names)
```

```
#=======================================================
# Executing                                            #
#=======================================================

#Create logfile
with new_log_file(filename) as file:
            file.write('File imported: item_dataset')
            #file.write(str(item_dataset))
            file.write(str(volume_dataset))
            file.write(str(location_dataset))
```

```
main = Main() # luo luokan instanssi
location_dataset = main.location_dataset
volume_dataset = main.volume_dataset
#järjestä varastopaikat painotuksen ja sijainnin mukaiseen järjestykseen
sorted_locations = location_dataset.sort_values(by=['Weight','Location'],axis=0, ascending=False, inplace=False, kind='quicksort', na_position='last')
#järjestä nimikkeet painotuksen ja koon mukaiseen järjestykseen.
sorted_items = volume_dataset.sort_values(by=['PWeight2','Size'],axis=0, ascending=False, inplace=False, kind='quicksort', na_position='last')
#print(sorted_items)
#print(sorted_locations[0:10])

optimal_locs =[]
optimal_items =[]
optimal_weights = []
triples = []
print("paikkojen optimointi aloitettu...")
```

```
for i in range(len(sorted_locations)): #käy läpi kaikki painoarvon mukaan järjestetyt varastopaikat
    #print("location under inspection: ",sorted_locations.iloc[i,1])
    j = 0
    while True: #kullekkin paikalle kokeillaan

        koko = sorted_items.iloc[j,6] #rivi j sarake Size
        if(sorted_locations.iloc[i,6] <= koko and sorted_items.iloc[j,1] != '999999'): #jos paikan minimikoko on sama tai pienempi kuin tuotteen koko, se kelpaa paikalle
            optimal_loc = sorted_locations.iloc[i,1]
            optimal_item = sorted_items.iloc[j,1] #lisää optimaalisten paikkojen listaan nimike j paikalle i
            optimal_item_weight = sorted_items.iloc[j,5] #PWeight2 kentästä painostus
            triple = optimal_loc,optimal_item,optimal_item_weight
            optimal_locs.append(optimal_loc)
            optimal_items.append(optimal_item)
            optimal_weights.append(optimal_item_weight)
            triples.append(triple) #sijainti, optiminimike, painoarvo samassa tarkastusta varten
            #sorted_items = sorted_items.drop([j]) #poista nimike käsiteltävistä nimikkeistä jotta läpikäynti nopeutuu
            sorted_items = sorted_items.replace({optimal_item : '999999'}) #korvataan jo käsitelty nimike numerolla 999999 jotta sitä ei enää käsitellä uudelleen
            #print("Optimaalinen tuote löydetty indexillä  ",j)
            j += 1 #kasvata j:tä yhdellä
            break
        elif j == len(sorted_items)-1: #Lopetetaan jos kaikki nimikkeet on testattu, mutta mikään ei kelpaa
            optimal_loc = sorted_locations.iloc[i,2] #tallennetaan apumuuttujaan tutkittava paikka
            optimal_loc = "tyhja ", optimal_loc #merkataan paikka tyhjäksi, ei sopivaa nimikettä
            optimal_locs.append(optimal_loc)
            print("Paikalle ei ole olemassa sopivaa nimikettä  ",j ,optimal_loc)
            break
        else:
            j += 1 #kasvata j:tä yhdellä
```

```
print("...paikkojen optimointi on valmis, käsiteltyjä paikkoja ", len(optimal_locs), " kpl")
print("optimaalisia nimikkeitä löytynyt",len(optimal_items)," kpl")
```

```
with open('..\WarehouseOptimization\output\optimal_loc_item.csv', mode='w') as csv_file:
    fieldnames = ['Location', 'SKU', 'Weight']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

    writer.writeheader()
    k = 0
    for i in range(len(optimal_items)):
        #writer.writerow({'Location': optimal_locs[k], 'SKU': optimal_items[k], 'Weight': ' '})
        writer.writerow({'Location': optimal_locs[k], 'SKU': optimal_items[k], 'Weight': optimal_weights[k]})
        k += 1
```

Liite 3: Testitilausten jakaminen omiin .csv tiedostoihin

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 25 14:48:04 2019

@author: mahfi
"""
import pandas as pd
try:
        content = open("..\input\AllOrders.csv").read()
        print("Tiedosto luettavissa")

except IOError:
        print("Tiedostoa ei pystytä lukemaan!")


names = ['Location','Item','Order']
orders=[]
csv = pd.read_csv('..\input\AllOrders.csv', delimiter=';',names =names,skiprows=1)

#================================================#
#POIMITAAN JOKAINEN TILAUSNUMERO LISTALLE        #
#================================================#

order_to_save = 0
j =0

for i in range(len(csv)):
    order_to_check = csv['Order'][j]
    j+=1
    if order_to_check != order_to_save:
        orders.append(order_to_check) #tuntematon tilausnumero lisätään listaan
        order_to_save = order_to_check #tallennetusta tulee se mitä tarkastetaan


#=============================================================================#
#LUODAAN LISTAN PERUSTEELLA JOKAISESTA TILAUKSESTA OMA CSV-TIEDOSTO          #
#=============================================================================#

print("tilausnumerot listattu")
print("käsitellään tilauksia..")
for order in orders:
    order = order
    csv_order = csv[csv['Order'] == order]
    name = str(order)+".csv"
    csv_order.to_csv(name, index=False, sep=';')
```

Liite 4: Simulointialgoritmi

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np
from sklearn import utils

import input
try:
        content = open("..\input\LocationSimulation.csv").read()
        print("Tiedosto luettavissa")

except IOError:  # Works in both Python 2 & 3
        print("Tiedostoa ei pystytä lukemaan!")


names = ['Location','endcost','row']
locations = pd.read_csv('..\input\LocationSimulation.csv',header=None,delimiter=';',names =names,skiprows=1)#tarvitaan simulointiin

#================================TUTKITTAVAT TAPAUKSET=============================================================================#
#Case1:tutkittava paikka ja edellinen paikka samalla rivillä => kustannus = tutkittavapaikka - edellinen paikka                    #
#Case2:tutkittava paikka ja edellinen paikka eri rivillä (seuraava rivi) => laske kustannus kuten yllä, jos kustannus alle 69 kustannus = kustannus  #
#                                                  muuten kustannus = 69                                                            #
#Case3:tutkittava paikka ja edellinen paikka eri rivillä => seuraava rivi - kuljettava rivi = < 3 : ei tehdä mitään. yllä olevat tapaukset kattavat  #
#Case4:tutkittava paikka ja edellinen paikka eri rivillä => seuraava rivi - kuljettava rivi = 3 tai 4 : kustannus on kustannus - 138               #
#Case5:tutkittava paikka ja edellinen paikka eri rivillä => seuraava rivi - kuljettava rivi = 5 tai 6 : kustannus on kustannus - 276               #
#Case6:tutkittava paikka ja edellinen paikka eri rivillä => seuraava rivi - kuljettava rivi = > 6 : kustannus on kustannus - 414                #
#==================================================================================================================================#

#=================================================#
#LOPPUKUSTANNUKSEN LASKEMINEN                     #
#=================================================#

def calculate_end_cost(last_loc, locations):
    x = 0
    for j in range(len(locations)):
        if locations.iloc[j,0] == last_loc:
            #print("paikka löydetty")
            x = locations.iloc[j,1]
        else:
            j+= 1
    return x
```

```python
#================================================#
#YKSITTÄISEN RIVIN KUSTANNUKSEN LASKEMINEN       #
#================================================#
def calculate_cost(current_loc,last_loc,locations): #parametrina laskettu välimatka paikasta seuraavaan
    cost_new = current_loc-last_loc #Lähtökustannus
    #print("Lähtökustannus: ",cost_new) #Case1
    row_current_loc = 0 #alustetaan nykyinen tutkittava käytävä ykköseksi, koska rivien numerointi alkaa siitä
    row_last_loc = 1
    #jotta tiedetään missä kohtaa rataa paikat sijaitsevat, joudutaan kaikki paikat tutkimaan
    for j in range(len(locations)): #käydään läpi kukin paikka
        if locations.iloc[j,0] == last_loc: #jos paikka j on viimeisin tutkittu paikka
            row_last_loc = locations.iloc[j,2] #silloin viimeisimmän paikan käytävän puoli on row_last_loc

        if locations.iloc[j,0] == current_loc: #jos paikka j on sillä hetkellä tutkittava paikka
            row_current_loc = locations.iloc[j,2] #silloin tutkittavan paikan käytävän puoli on row_current_loc

            #koska ylempi if on aina totta ennen nykyistä paikkaa (ei palata taaksepäin) voidaan tässä kohtaa laske käytävien välinen etäisyys
            #print("käytävät edellinen : ",  row_last_loc, "nyt tutkitaan", row_current_loc)
            diff= row_current_loc - row_last_loc #Lasketaan käytävien välimatka vähentämälle edellinen käytävä nykyisestä.
            #print("ero:",diff)
            if diff == 0: #case1
                cost_new = cost_new
                #print("Case1")
            elif diff == 1 or 2 : #case2 & 3
                if cost_new > 69:
                    cost_new = 69
                else: cost_new = cost_new
                #print("Case2")
            elif diff == 3 or 4: #case 4
                cost_new == cost_new-138
                #print("Case3")
            elif diff == 5 or 6: #case 5
                cost_new == cost_new-276
                #print("Case4")

            elif diff > 6: #case 6
                cost_new == cost_new-414
                #print("Case5")
            else: print("virhe")

    #print("palautetaan kustannus: ", cost_new)
    #tuotteet joille ei löydy paikkaa keräilyradan kaikista paikoista, voidaan olettaa olevan volyymiltaan niin pieniä,
    #jotta ne voidaan sijoittaa radan ulkopuolisiin pienempiin hyllyihin tai käsitellä pullokeräilynä.

    return cost_new

#================================================================#
#SIMULOIDAAN KERÄILY JA TALLENNETAAN RIVIKOHTAISET KUSTANNUKSET#
#================================================================#
def simulate_picking(order):
    cost_table =  [] #alustetaan kustannukset aina nollaksi
    last_loc = 0
    for i in range(len(order)):
        current_loc = order.iloc[i,0] #tutkittava paikka
        cost_new = calculate_cost(current_loc,last_loc,locations)
        last_loc = order.iloc[i,0] #asetetaan tutkittu paikka edelliseksi paikaksi
        #print(last_loc)
        cost_table.append(cost_new)

    end_cost = calculate_end_cost(last_loc, locations) #funktio laskee kustannuksen palata lähtöpaikkaan
    #print("loppukustannus (paluu lähtöpaikkaan): ", end_cost)
    total_cost = sum(cost_table)+end_cost
    #print("kokonaiskustannus on ", total_cost)
    return total_cost
```

```
#=========================================================================#
#SEKOITA KERÄILYPAIKAT SATUNNAISEEN JÄRJESTYKSEEN KOKO RADAN PITUUDELTA#
#=========================================================================#

def shuffle_order(order):
    for i in range(len(order)):order.iloc[i,0] =np.random.randint(1,627) #sekoita paikat randomilla
    order.sort_values(by='Location',inplace=True) #järjestä keräilyjärjestykseen
    return order
```

```
#=============================================================================#
#OHJAA TILAUKSELLA OLEVAT TUOTTEET OPTIMAALISILLE PAIKOILLE JOTKA LASKETTU AIEMMIN    #
#=============================================================================#

def optimize_locations(x, optimal_locations):
    new_order = []
    bias = []
    for i in range(len(x)):
        item = x.iloc[i,1] #tuote
        loc = x.iloc[i,0] #paikka
        j = 0
        #print("paikka ennen optimointia, ",item, loc)
        while j < 551:    #tutkitaan paikkoja kunnes vastaava nimike löytyy 552 on paikkojen määrä
            if item == optimal_locations.iloc[j,1] and loc != optimal_locations.iloc[j,0] :
                #print("uusi paikka löydetty ",optimal_locations.iloc[j,0])
                new_order.append(int(optimal_locations.iloc[j,0])) #muista konvertoida kaikki palautettavat paikat numeroiksi jotta niitä voidaan laskea.
                break
            elif j < (len(optimal_locations)-1):
                j+=1
        else:
            ero = 0
            ero = 322-loc #lasketaan paljonko on ero alkuperäisen ja uuden "lisähyllyn" välillä
            bias.append(int(ero))
            #new_order.append(int(loc))
            new_order.append(int('322'))
    #print(sum(bias),"puolueellinen yksikköä optimoidun algoritmin suuntaan")
    bias = sum(bias)
    return new_order,bias;
```

Liite 5: Simuloinnin suorittaminen

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 25 09:05:17 2019

@author: mahfi
"""
from simulation.simulate_picking import shuffle_order
from simulation.simulate_picking import simulate_picking
from simulation.simulate_picking import optimize_locations
import pandas as pd
import glob
import os


names_optm = ['Location','item','weight']
order_names = ['Location','item','item_name']


optimal_locations = pd.read_csv('..\output\optimal_loc_item.csv',header=None,delimiter=',',names =names_optm,skiprows=1)#optimaaliset paikat jotta tilauksille voidaan muuttaa uusi paikka
optimal_locations['Location'] = optimal_locations.Location.str.replace("[({'PABC]", "") #poistetaan paikoista etu- ja takaosan kirjaimet jotka eivät kuvaa järjestystä. "P001A"

orig_loc = []
optim_loc = []
random_loc = []
sum_of_bias = []
earlier=0
#path=r'C:\Users\mahfi\WarehouseOptimization\input\orders'
path='C:/Users/mahfi/Desktop/Koulu/Gradu/WarehouseOptimization/input/orders'
all_files = glob.glob(os.path.join(path, "*.csv")) #lista jossa kaikki .csv tiedostot
```

```
"SIMULOI KERÄILYÄ TODELLISILLA TILAUKSILLA JA PAIKOILLA"
print("simuloidaan keräilyä oletuspaikoilla..")
for file in all_files: #tehdään simulointi kaikille tiedostoille kansiossa
    file_name = os.path.splitext(os.path.basename(file))[0]
    file_name = "/"+file_name+".csv"
    order1 = pd.read_csv(path+file_name,header=None,delimiter=';',names =order_names,skiprows=1,encoding = 'unicode_escape')
    order1.sort_values(by='Location',inplace=True)
    normal = simulate_picking(order1)

    #print("normaali versio: ",normal) #tulos aina 607 tällä tilauksella
    orig_loc.append(normal)
```

```
"SIMULOI KERÄILYÄ SEKOITETUILLA PAIKOILLA"
print("..simuloidaan keräilyä satunnaisilla paikoilla..")
for file in all_files:
    file_name = os.path.splitext(os.path.basename(file))[0]
    file_name = "/"+file_name+".csv"
    order2 = pd.read_csv(path+file_name,header=None,delimiter=';',names =order_names,skiprows=1,encoding = 'unicode_escape')
    shuffled_order = shuffle_order(order2)
    shuffled = simulate_picking(shuffled_order)
    #print("sekoitettu versio: ",shuffled) #tulos vaihtelee
    random_loc.append(shuffled)
```

```
"SIMULOI KERÄILYÄ OPTIMOIDUILLA PAIKOILLA"
print("..simuloidaan keräilyä optimaalisilla paikoilla..")
for file in all_files:
    file_name = os.path.splitext(os.path.basename(file))[0]
    file_name = "/"+file_name+".csv"
    order3 = pd.read_csv(path+file_name,header=None,delimiter=';',names =order_names,skiprows=1,encoding = 'unicode_escape')
    optimized_order,bias = optimize_locations(order3,optimal_locations)
    optimized_order = pd.DataFrame(optimized_order, columns=['Location'])  #laita uudet paikat data frameen
    optimized_order.sort_values(by='Location',inplace=True) #järjestä paikat suuruusjärjestykseen
    optimal = simulate_picking(optimized_order)
    #print("algoritmi on puolueellinen optimoinnin suuntaan ",bias,"yksikköä tällä tilauksella")
    #print("optimoitu versio: ",optimal)
    optim_loc.append(optimal)
    sum_of_bias.append(bias)
```

```
#================================================#
#SUMMATAAN SIMULOINTITULOKSET                    #
#================================================#
print("simulointi on valmis. analysoidaan tuloksia..")
bias = sum(sum_of_bias)
a = sum(orig_loc)
b = sum(optim_loc)+bias
c = sum(random_loc)

print(len(all_files)," tilausta käsitelty:")
print("alkuperäisillä keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä ",a)
print("optimoiduilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä ",b)
print("satunnaisilla keräilypaikoilla kustannus kerätä kaikki tilaukset on yhteensä ",c)
print("optimoituihin paikkoihin on lisätty/vähennetty algoritmin puolueellisuudesta aiheutunut kustannus ", bias)

total = a+b+c
orig = a/total*100
optim = b/total*100
random = c/total*100

opros = ((orig-optim)/optim)*100
rpros = ((random-optim)/optim)*100
print ("optimoitu versio on alkuperäistä parempi ",opros, " prosenttia ja ", rpros, "parempi satunnaista asettelua")
```

Liite 6: Tiedostot

main.py = paikkojen optimoinnin toteutus.

simulate_picking.py = simulointiin käytettävät funktiot ja heuristiikan toteutus.

run_simulation.py = simuloinnin suorittaminen.

Item_volume2.csv = optimointiin käytetyt nimikkeet.

locations3.csv = optimointiin käytetyt paikat esiprosessoinnin ja painotuksen jälkeen.

optimal_loc_item.csv = optimoinnin tuloksena syntyvä csv-tiedosto optimaalisista tuotteista kullekin paikalle.

LocationSimulation.csv = keräilyradan esitys tekstimuodossa simulointia varten sisältäen paikkojen rivipuoliskon sekä paluukustannuksen takaisin alkuun.