Aalto-yliopisto
Perustieteiden korkeakoulu
Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Tuotantoprosessin optimaalinen aikataulutus

Kandidaatintyö
28.11.2014

Joona Kaivosoja

Aalto-yliopisto

AALTO-YLIOPISTO PERUSTIETEIDEN KORKEAKOULU PL 11000, 00076 Aalto http://www.aalto.fi	KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Joona Kaivosoja

Työn nimi: Tuotantoprosessin optimaalinen aikataulutus

Tutkinto-ohjelma: Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Pääaine: Systeemitieteet	Pääaineen koodi: F200-3

Vastuuopettaja: DI Ville Mäkelä

Ohjaaja: Professori Ahti Salo

Tiivistelmä:

Tässä työssä tutkitaan valmistus-pakkaus-prosessin optimaalista aikatauluttamista lineaarisen sekalukuoptimointimallin sekä neljän eri heuristiikan avulla. Tarkasteltava tuotantoprosessi kuvaa todellista tuotantosysteemiä, jossa tuote-erien väliset asetusajat koneilla riippuvat aikataulutettujen tuote-erien järjestyksestä. Tuotetyypistä riippuen jokainen tuote-erä käy tuotantoprosessissa läpi 1-2 valmistusvaihetta sekä yhden pakkausvaiheen. Jokaiselle tuotetyypille on jokaisessa vaiheessa määritetty joukko vaihtoehtoisia koneita, joilla vaihe voidaan suorittaa. Tuotantoprosessissa on yhteensä 9 konetta, ja erilaisia tuotetyyppejä on 8. Aikataulujen arviointikriteerinä on tuotanto-ohjelman läpäisyaika, eli aika, joka kuluu ohjelman ensimmäisen tuote-erän aloittamisesta viimeisen tuote-erän valmistumiseen. Tuotantoprosessin aikataulutusongelmasta formuloidaan lineaarinen sekalukuoptimointitehtävä. Käytettävät heuristiikat ovat: Monte Carlo –algoritmi, geneettinen algoritmi, sekä prioriteettisäännöt lyhyin prosessiaika (engl. shortest processing time, SPT) ja lyhyin asetusaika (engl. shortest setup time, SST).

Ratkaisualgoritmeja arvioidaan sekä niiden vaatiman laskenta-ajan että niiden antamien tulosten laadun perusteella. Tuotantoprosessin mallista luodaan yhteensä 130 testiongelmaa, joissa aikataulutettavia tuote-eriä on vähintään 4 ja enintään 16. Laskenta-ajat esitetään aikataulutettavien tuote-erien määrän funktiona, ja heuristisen menetelmien aikataulujen laatua arvioidaan suhteessa sekalukuoptimointimallin sekä SST-prioriteettisäännön tuloksiin.

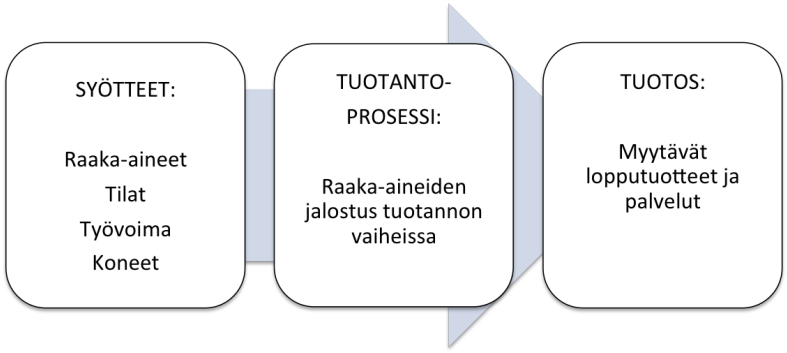
Päivämäärä: 28.11.2014	Kieli: Suomi	Sivumäärä: 30 + 3

Avainsanat: aikataulutus, läpäisyaika, MILP, geneettinen algoritmi, job-shop scheduling

Sisältö

1 Johdanto

Tuotantoprosessissa raaka-aine jalostetaan lopputuotteeksi. Tuotantoprosessi koostuu erilaisista vaiheista, jotka ovat välttämättömiä lopputuotteen tuottamiseksi. Prosessissa tuotteiden yksikkönä käytetään usein tuote-erää, ennalta määrätyn kokoista joukkoa yhdenlaista lopputuotetta, joka kulkee tuotantoprosessin läpi jakamattomana. Alasta riippumatta tuotantoprosessien eri vaiheet vaativat resursseja tuotannon mahdollistamiseksi. Kullakin prosessilla ja tuotteella on omat yksilölliset vaatimuksensa esimerkiksi tilojen, osaavan työvoiman, koneiden, raaka-aineiden, tai energian suhteen.



Kuva 1: Tuotantoprosessissa raaka-aineista jalostetaan muiden tuotannon resurssien avulla asiakkaalle myytävä lopputuote.

Harva tuote valmistetaan vain yhdessä vaiheessa, yhdessä paikassa, tai yhdenlaista resurssia käyttäen. Lisäksi harvassa tuotantolaitoksessa valmistetaan vain yhdenlaista tuotetta, vaan samaan aikaan valmistettavat erilaiset tuotteet kilpailevat yhteisistä tuotantoresursseista. Siten käytössä olevat ja tuotantoon tarvittavat resurssit on sovitettava ajallisesti yhteen. Tuotantoprosessin ohjaamista tukee tuotantoaikataulu, joka määrittää, milloin mitäkin tuote-eriä valmistetaan ja kuinka paljon resursseja eri vaiheissa käytetään.

Ominaista erilaisille prosesseille on, että vaihdettaessa tuotettavaa tuotetta kuluu ns. asetusaika ennen kuin toista tuotetta voidaan alkaa tekemään. Työpisteen työkalut on vaihdettava, tuotantotila puhdistettava, tai koneen osat vaihdettava seuraavaa tuotetta varten. Yleisesti asetusaikoja on kahdentyyppisiä, tuotejärjestyksestä riippuvaisia ja järjestyksestä riippumattomia. Järjestyksestä riippumattomat asetusajat voidaan tuotantoaikataulussa sisällyttää tuote-erien prosessiaikoihin, joten ne eivät monimutkaista aikataulutusongelmaa. Tuotejärjestyksestä riippuvat asetusajat täytyy aikatau-

lutuksessa kuitenkin käsitellä prosessiajoista irrallisina, mikä lisää aikataulutusongelman vaikeutta.

Asetusaikojen merkitys tuotantoprosessin tehokkuudelle vaihtelee teollisuusaloittain paljon. Esimerkiksi kemianteollisuuden prosesseissa asetusajat voivat olla erittäin merkittäviä suhteessa tuotteen prosessointiaikaan johtuen siitä, että edellisen tuoteen mahdolliset jäämät tuotantolaitteistossa täytyy puhdistaa tarkoin ennen seuraavan tuotteen valmistuksen aloittamista. Yksittäisen tuote-erän prosessiaika voi jopa olla lyhyempi kuin tuotetta edeltävä asetusaika. Tällöin myös tuotetta edeltävä asetusaika voi vaihdella voimakkaasti edellisestä tuotteesta riippuen, kahta samanlaista tuote-erää peräkkäin valmistettaessa asetusaika voi olla hyvin lyhyt.

Tuotantoaikataulun optimoinnin tavoitteena on minimoida tuotantoprosessin kokonaiskustannuksia suhteessa sen kapasiteettiin. Optimaalisessa aikataulussa erilaiset odotus-, asetus-, ja prosessiajat tuotantoprosessissa on minimoitu jakamalla tuotantoresurssit tarkoituksenmukaisesti. Mitä enemmän arvoa tuottavaa työtä ehditään tehdä aikayksikössä, sitä pienemmät keskimääräiset valmistuskustannukset tuote-erää kohti saavutetaan. Käytännössä tämä voisi esimerkiksi tarkoittaa ylitöiden vähentämistä, siirtymistä kolmivuorotyöstä kahteen vuoroon, tai tuotantoprosessin kapasiteetin kasvua.

Tuotantoaikataulua optimoitaessa aikataulun laatu voi monesti olla laskenta-aikaa tärkeämpi kriteeri. Esimerkiksi viikon tuotanto-ohjelman hienokuormitusta laadittaessa aikataulun ei tarvitse olla valmis sekunneissa, vaan muutamankin tunnin laskenta-aika voisi olla hyväksyttävissä. Pitkä laskenta-aika voi kuitenkin olla haitaksi, jos esimerkiksi aikatauluun joudutaan tekemään viime hetken muutoksia, aikataulu joudutaan uudelleenaikataulut-tamaan kesken tuotanto-ohjelman, tai tuotanto-ohjelman tietojen syöttämisessä tapahtuu virhe.

Tässä työssä rakennetaan malli todellista tuotantoprosessia kuvaavasta tuotanto-pakkaus -prosessista ja arvioidaan erilaisten optimointimenetelmien suoriutumista erikokoisten tuotanto-ohjelmien läpäisyaikojen minimointitehtävissä. Työssä käytettävät optimointimenetelmät ovat: lineaarinen sekalukuoptimointi, geneettinen algoritmi, SPT- ja SST-prioriteettisäännöt, sekä Monte Carlo -algoritmi. Tuotantoprosessissa tuote-erät käyvät läpi useita vaiheita, ja joitakin vaiheita voidaan suorittaa vaihtoehtoisilla koneilla. Asetusajat vaihtelevat tuotantoprosessissa paljon tuotejärjestyksestä riippuen.

2 Teoreettinen tausta ja edeltävä tutkimus

2.1 Job-shop aikataulutus

Yksi laskennallisesti vaikeimmista kombinatorisen optimoinnnin tehtävistä on niin kutsuttu job-shop scheduling (JSS) ongelma (Applegate and Cook, 1991). Konepajan aikataulutusongelmassa tehtävänä on aikatauluttaa joukko töitä konepajan eri koneille. Tehtävässä jokaiselle työlle on ennalta määrätty tietyt työvaiheet, eli koneet, sekä järjestys näille työvaiheille tuotteen valmistamiseksi. Yleisimmin optimointitehtävän tavoitteena on minimoida töiden valmistumisaikojen maksimi (Vela et al., 2010), eli tuotanto-ohjelman kokonaispituus konepajalla, mutta kohdefunktiona voidaan käyttää myös esimerkiksi valmistumisaikojen tai toteutuneiden asetuskustannusten summaa. Yksinkertainen yhden koneen job-shop ongelma on yhtenevä kuuluisan kauppamatkustajan ongelman kanssa, jossa kauppamatkustaja aloittaa ja lopettaa matkansa eri kaupungeista. Tehtävässä kauppamatkustaja voidaan tulkita koneeksi ja kaupungit aikataulutettaviksi töiksi. Job-shop ongelma ja sen yleistykset ovat siten kauppamatkustajan ongelman tavoin NP-kovia (Moghaddas and Houshmand, 2008).

Aikataulutusongelmassa asetusajat ja -kustannukset voivat joko riippua tuotteiden järjestyksestä tai olla näistä riippumattomia. Mikäli asetusajat oletetaan mallissa tuotteiden järjestyksestä riippumattomiksi, ne voidaan sisällyttää tuotteiden konekohtaisiin prosessointiaikoihin, jolloin asetusajat eivät ole eksplisiittisesti osa optimointitehtävää. Tämä oletus tehdäänkin suuressa osassa aikataulutustutkimuksia (Moghaddas and Houshmand, 2008). Job-shop ongelma, jossa asetusajat riippuvat tuotteiden järjestyksestä, formuloitiin jo yli kolme vuosikymmentä sitten, mutta ongelma on saanut vasta viime aikoina enemmän huomiota (Vela et al., 2010). Asetusaikojen eksplisiittinen käsittely lisää mallin kompleksisuutta, mutta monessa sovelluksessa asetusaikojen ja prosessointiaikojen erottaminen toisistaan antaa todenmukaisemman kuvan systeemistä. Eräässä tutkimuksessa todettiin, että noin 75%:ssa todellisista aikataulutusongelmista asetusaikoja pitäisi käsitellä erikseen prosessointiajoista (Allahverdi and Soroush, 2008).

Job-shop ongelman muita ominaisuuksia ja oletuksia ovat yleisesti: prosessi- ja asetusaikojen vakioisuus, koneiden saatavuus riippumatta muiden koneiden kuormituksesta, yksikäsitteiset valmistusprosessit, ja töiden jakamattomuus. Aikataulutusongelmissa prosessi- ja asetusajat oletetaan yleisesti vakioiksi, eli riippumattomiksi tuotantotekijöistä, kuten koneiden kuormituksesta tai henkilöstön määrästä. Rajoittavaksi tekijäksi systeemissä oletetaan

usein konekapasiteetti osaavan henkilöstön tai raaka-aineiden sijaan, minkä vuoksi koneiden oletetaan olevan käytettävissä aina, kun niillä ei valmisteta jotakin tuotetta. Valmistusprosesseista oletetaan, että tuotteiden eri vaiheissa vaatimat koneet on etukäteen määritelty eikä tuotantovaiheissa ole vaihtoehtoisia koneita, ja ettei tuotteen valmistusta voida keskeyttää kesken minkään tuotantovaiheen. Yksittäiset artikkelit voivat kuitenkin poiketa näistä yleisistä oletuksista. Esimerkiksi Ruiz and Andrés-Romano (2011) käsittelevät artikkelissaan resurssien allokoinnista lineaarisesti riippuvaisia asetusaikoja rinnakkaisten, toisistaan riippumattomien, koneiden tapauksessa. Choi and Choi (2002) tarkastelevat artikkelissaan yleistettyä job-shop ongelmaa, jossa tuotteiden eri tuotantovaiheissa vaatimat koneet eivät ole yksiselitteisiä, vaan tuotantovaihe voidaan skeduloida joukolle vaihtoehtoisia koneita.

Konepajan aikataulutusongelma voidaan formuloida lineaarisena sekalukuoptimointitehtävänä (Applegate and Cook, 1991). Lineaarisen optimointitehtävän ratkaisuna saadaan globaali optimi aikataulutusongelmaan, mutta eksakteilla algoritmeilla voidaan ratkaista vain kooltaan ja kompleksisuudeltaan rajoitettuja ongelmia, koska algoritmien vaatimat ratkaisuajat kasvavat. Aikataulutusongelmien merkityksestä käytännön sovellutuksissa ja tieteellisessä tutkimuksessa johtuen ongelmien ratkaisemiseksi on kehitetty lukuisia menetelmiä. Vela et al. (2010) jaottelee erilaiset ratkaisumetodit kolmeen luokkaan: prioriteettisääntöihin (priority rules), eksakteihin menetelmiin, ja heuristiikkoihin. Prioriteettisääntöihin perustuvassa aikataulutuksessa koneelle valitaan seuraavaksi tuotteeksi tuote, jonka tunnusluku on aikataulutushetkellä vapaista tuotteista edullisin. Käytettyjä tunnuslukuja ovat esimerkiksi lyhyin prosessiaika (Shortest Prosessing Time, SPT), lyhyin asetusaika (Shortest Setup Time, SST), ja eniten prosessiaikaa jäljellä (Most WorK Remaining, MWKR). Näistä SPT on kenties suosituin, ja monien uusien menetelmien tehokkuutta verrataan usein SPT-sääntöön (Haupt, 1989). Menestyneimpiä eksakteja menetelmiä ovat erilaiset branch & bound -algoritmit (Vela et al., 2010). Heuristiikoista monia erilaisia geneettisiä algoritmejä, tabu-etsintä, paikallinen etsintä, sekä simuloitu jäähdytys -algoritmejä on hyödynnetty aikataulutusartikkeleissa. Algoritmejä on myös sekoitettu keskenään hybridialgoritmeiksi siten, että esimerkiksi geneettiseen algoritmiin on otettu elementtejä paikallinen etsintä -algoritmista tai prioriteettisäännöistä. (Bülbül and Kaminsky, 2013), (Moghaddas and Houshmand, 2008)

2.2 Lineaarinen sekalukuoptimointi

Lineaarisessa sekalukuoptimointitehtävässä (Mixed Integer Linear Programming, MILP) minimoitava kohdefunktio ja käypien ratkaisujen joukkoa rajaavat rajoitusehdot voidaan esittää lineaarisina funktioina päätösmuuttujista x, joista osa määritellään kokonaislukurajoitteisiksi. Yleisesti malli voidaan siis esittää muodossa:

$$\begin{aligned} \min \quad & c^T \cdot x \\ s.t. \quad & A \cdot x \leq b \\ & Aeq \cdot x = beq \\ & x \geq 0 \\ & y \subset x, \quad y \in \mathbb{Z}, \end{aligned}$$

jossa vektori c^T määrittää kohdefunktion, ja matriisit A ja Aeq sekä vektorit b ja beq määrittävät yhtälö- ja epäyhtälörajoitteet. Vektori x sisältää tehtävän päätösmuuttujat, joista osajoukko y on rajattu kokonaisluvuiksi.

Lineaariset sekalukutehtävät ovat yleistys lineaarisista optimointitehtävistä (Linear Programming, LP). Suuretkin LP-tehtävät voidaan nykyisin ratkaista tehokkaasti, mutta LP-tehtävänä voidaan formuloida MILP-tehtäviä paljon rajallisempi määrä tehtäviä. MILP-malleihin voidaan sisällyttää monipuolisesti esimerkiksi vaihtoehtoisia rajoituksia, positiivisen tuotantomäärän suuruudesta riippumattomia kiinteitä kustannuksia, tai paloittain lineaarisia kohdefunktioita. Lisäksi muuttujat voivat yleisesti esittää suureita, joille vain kokonaislukuarvot ovat mielekkäitä. MILP-mallien ratkaisu on kuitenkin haastavaa, eikä esimerkiksi MILP-tehtävinä formuloituja suurikokoisia aikataulutustehtäviä voida yleisesti ratkaista. (Chinneck, 2012)

Lineaaristen sekalukutehtävien ratkaisuun käytetään yleisimmin erilaisia *branch-and-bound* -algoritmeja. Algoritmien ensimmäinen vaihe MILP-ongelman ratkaisemiseksi on ratkaista alkuperäinen tehtävä ilman kokonaislukurajoituksia, eli tavanomaisena lineaarisena optimointitehtävänä, jolloin LP-relaksaation tuloksena saadaan alaraja kohdefunktion arvolle. Mikäli relaksaation ratkaisu toteuttaa alkuperäisen tehtävän kokonaislukurajoitteet, vastaa MILP-ongelman ratkaisu LP-relaksaation tulosta. Jos relaksaation tulos ei toteuta kokonaislukurajoitteita, valitaan jokin muuttuja x_i, optimiarvo LP-tehtävässä $\bar{x}_i$, joka ei toteuta kokonaislukurajoitetta. Nykyisestä LP-tehtävästä muodostetaan kaksi uutta alitehtävää, joihin lisätään kumpaankin yksi rajoite lisää. Toiseen lisätään rajoitus $x_i \leq k$ ja toiseen $x_i \geq k{+}1$, kun pätee $k < \bar{x}_i < k{+}1$, $k \in \mathbb{Z}$. Uusilla rajoituksilla x_i pakotetaan pois optimistaan, joka ei toteuta kokonaislukurajoitusta. Näin syntyviä alitehtäviä valitaan ratkaistavaksi, kun-

nes optimaalinen ratkaisu MILP-tehtävään löytyy. (Vielma, ilmestyy 2014) (Chinneck, 2012)

2.3 Geneettiset algoritmit

Geneettiset algoritmit ovat metaheurististen optimointimenetelmien luokka, joka jäljittelee luonnollisen evoluutioprosessin ominaisuuksia. Algoritmissa ratkaisujoukkoa, nk. kromosomipopulaatiota, pyritään asteittain parantamaan alistamalla yksittäiset ratkaisut, eli kromosomit, luonnonvalinnalle. Jokaista uutta kromosomisukupolvea muodostettaessa edellisen sukupolven sopivimmilla ratkaisuilla on suurin todennäköisyys päästä siirtämään ominaisuutensa uuteen sukupolveen. Luonnonvalinnasta selviytyneet kromosomit synnyttävät uuden populaation kromosomit geneettisen risteytysoperaattorin määräämällä tavalla. Kuten luonnossa, myös geneettisessä algoritmissa mutaatiot voivat kuitenkin vaikuttaa uuden sukupolven kromosomeihin. (Mesghouni et al., 1999)

Geneettistä algoritmia rakennettaessa täytyy tehdä monia päätöksiä algoritmin parametreihin ja rakenteisiin liittyen, joihin ratkaistavan ongelman ominaisuudet vaikuttavat. Ratkaisu täytyy esitää kromosomina risteytys- ja mutaatio-operaattoreita varten. Kromosomin perinteinen esitys on vektori, jonka alkiot, eli geenit, saavat binääriarvoja. Tämä on kuitenkin rajoittunut tapa kuvata monimutkaisia ratkaisuja, joten uusia esitystapoja on kehitetty vastaamaan haastavampien tehtävien, kuten aikataulutustehtävien, ominaisuuksia. (Mesghouni et al., 1999)

Kromosomisyntaksin ohella tärkeimpiä suunnittelupäätöksiä ovat geneettisten operaattoreiden, risteytys- ja mutaatio-operaattoreiden, toiminta. Risteytysoperaattori jäljittelee tapaa, jolla luonnossa osa kummankin vanhemman geeneistä siirtyy heidän jälkeläiselleen. Risteytysoperaattoreita on kehitetty monia erilaisia, ja sopivan operaattorin löytäminen ongelmatyypille saattaa vaatia testausta. Yksinkertaisimmassa risteytysoperaattorissa jälkeläinen saa kahdelta vanhemmalta kummaltakin ennalta määrätyn osan geeneistään, esimerkiksi isältään ensimmäisen puolikkaan geeneistään ja äidiltään jälkimmäisen puolikkaan. Mutaatio-operaattorin tehtävänä on geneettisessä algoritmissa, kuten luonnossakin, ylläpitää populaation geneettisen perimän monimuotoisuutta. Operaattori operoi yhtä satunnaista kromosomia ja muokkaa sen sisältöä määrätyllä tavalla. Aikataulutustehtävässä mutaatio saattaa esimerkiksi vaihtaa jollekin koneelle aikataulutettujen tuote-erien järjestyksen toisenlaiseksi. (Mesghouni et al., 1999)

3 Tutkimusongelma ja -menetelmät

Työn tutkimusongelmana oli soveltaa aiempia alan tutkimuksia, erityisesti kehitettyjä MILP-malleja, ja tutkia aikataulutettavan tuotanto-ohjelman koon vaikutusta mallin laskenta-aikaan. Lisäksi tutkittiin yksinkertaisten heuristiikkojen laskenta-aikoja sekä tulosten laatua.

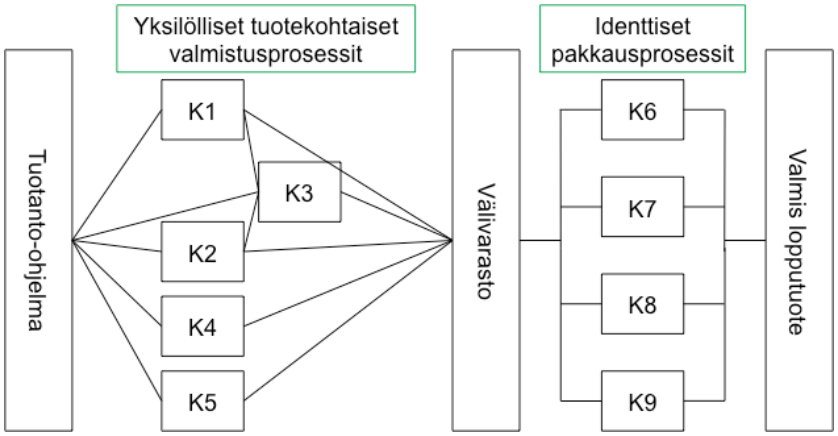
Aikataulutusalgoritmien toimintaa tutkitaan todellista tuotantoympäristöä kuvaavan testiongelman avulla. Luvussa 3.1 määritellään testiongelman yleiset ominaisuudet ja luvussa 3.2 formuloidaan testiongelmaa kuvaava MILP-malli. Työssä käytettävien heuristisen menetelmien sovellutukset kuvataan luvussa 3.3.

3.1 Testiongelman määrittely

Tässä työssä mallinnetaan kaksivaiheinen tuotantosysteemi, joka kuvaa valmistus-pakkaus-prosessia. Yleiskuva tuotantosysteemistä on esitetty kuvassa 2. Systeemiin kuuluu yhteensä 9 konetta, joista koneet 1-5 ovat osa valmistusprosessia ja koneet 6-9 osa pakkausprosessia. Systeemin lävitse voi kulkea kahdeksaa erilaista tuotetta (tuotteet A-H). Valmistusprosessissa jokainen tuoteerä käy tuotekohtaisesti läpi 1-2 vaihetta, joista jokaiselle on määritelty 1-2 koneen mahdollinen konejoukko. Kukin kone voi kuitenkin olla tuotteelle osa vain yhtä sen vaihetta ja vaihdeiden järjestys on määrätty. Pakkausprosessissa kukin tuote käy läpi yhden vaiheen. Pakkauksessa koneet 6-9 ovat identtisiä, ja mikä tahansa tuote voidaan aikatauluttaa mille tahansa koneelle. Valmistusvaiheen ja pakkausvaiheen välissä tuotteet voidaan välivarastoida ilman erilliskustannuksia tai rajoituksia, joten tuotteen valmistuttua valmistusvaiheesta sen ei tarvitse välittömästi siirtyä pakkausprosessiin. Tuotteen tulee kuitenkin valmistua valmistusprosessista ennen kuin sen pakkausprosessi voi alkaa. Tuotteiden erilaiset tuotantoprosessit on esitetty tuotekohtaisesti taulukossa 1. Taulukosta voidaan myös nähdä, että tuotteet voidaan tuotantoprosessiensa perusteella jakaa kuuteen eri tuotekategoriaan tuotantovaiheiden ja niissä mahdollisten koneiden mukaisesti.

Jokaiselle tuotteelle on mallissa määritelty prosessiajat ja asetusajat konekohtaisesti. Prosessiaika $p_{i,k}$ kuvaa tuotteen i prosessointiin kuluvaa aikaa koneella k. Prosessiajat on esitetty liitteen A taulukossa 3. Asetusajat ovat mallissa tuotteiden järjestyksestä riippuvaisia ja ne on esitetty liitteen A taulukoissa 4, 5, 6, 7, 8, ja 9. Asetusaika $s_{i,j,k}$ kuvaa tuotevaihtoon käytettävää aikaa, kun tuotteesta i siirrytään valmistamaan tuotetta j koneella k. Sekä

prosessiajat että asetusajat on mallissa oletettu vakioiksi, eli niihin ei sisälly epävarmuutta.



Kuva 2: Havainnekuva testiongelman tuotantosysteemistä ja eri tuotteiden mahdollisista poluista tuotantosysteemin lävitse.

Taulukko 1: Tuotteiden tuotantoprosessit valmistus- ja pakkausvaiheissa. Sarakkeissa on esitetty koneet, joilla tuotantoon kuuluvan prosessin voi suorittaa tietyllä tuotteella.

Tuotteet	Valmistusprosessi		Pakkausprosessi
	1. vaihe	2. vaihe	1. vaihe
A	5	-	6/7/8/9
B	4	-	6/7/8/9
C, E	1/2	-	6/7/8/9
D	3	-	6/7/8/9
F	1	-	6/7/8/9
G, H	1/2	3	6/7/8/9

Todellinen tuotantosysteemi aloittaa harvoin tuotannon puhtaalta pöydältä. Mallissa tuotantosysteemin nykyinen tuotantotilanne alustetaan parametreilla R_k ja T_k, jotka määrittelevät koneelle k sen edellisen tuotanto-ohjelman viimeisen tuotteen tyypin ja tuotteen valmistumisajan. Parametrin R_k avulla voidaan määrittää koneelle k ensimmäiseksi aikataulutetun tuotteen aloitusta edeltävän asetusajan pituus ja parametrin T_k avulla tämän asetusajan aikaisin mahdollinen aloitusaika. Nykyisen tuotantotilanteen alustaminen skedulointitehtävään tekee mallista paremmin todellisuutta vastaavan ja mahdollistaa mallin käytön myös tilanteissa, joissa tuotanto-ohjelma joudutaan uudelleenaikatauluttamaan osittain.

Muita testiongelmaan liittyviä oletuksia ovat: töiden jakamattomuus, tuotevaihtojen erotettavuus tuotantovaiheista, ja materiaalin sekä työvoiman saatavuus. Töiden jakamattomuus tarkoittaa sitä, ettei työtä esimerkiksi pakkausvaiheessa voida jakaa kahdelle koneelle työstettäväksi samaan aikaan. Toisaalta vaaditaan myös, että aloitettu työ jatkuu keskeyttämättömänä loppuun asti. Tuotevaihdoista oletetaan, että vapaana oleva kone voidaan valmistella seuraavaa tuotetta varten, vaikka tuote ei vielä olisi valmistunut edellisestä vaiheestaan. Kapasiteettirajoitukseksi työssä oletettiin ainoastaan konekapasiteetti, materiaalin ja työvoiman oletetaan olevan saatavilla tarvittaessa.

3.2 Lineaarinen sekalukuoptimointimalli

Työn yhtenä tavoitteena oli tutkia testiongelman ratkaisemista lineaarisen sekalukuoptimointimallin (Mixed Integer Linear Programming, MILP) muodossa. Lineaarisessa sekalukuoptimointimallissa kaikki rajoitusehdot ja kohdefunktio ovat lineaarisia kuvauksia päätösmuuttujista, joista osa voi olla kokonaislukurajoitteisia. Optimointimallin formulointi on kokonaisuudessaan esitetty alla. Mallin päätösmuuttujia ovat: tuotteen i aloitusaika $t_{i,k}$ koneella k, aloitusaikojen maksimi C_{max}, sekä binäärimuuttujat $x_{i,j,k}$. Binäärimuuttuja $x_{i,j,k}$ saa arvon 1, kun tuote i edeltää tuotetta j koneella k, ja saa muissa tapauksissa arvon 0. Optimointitehtävän kohdefunktio on tuotanto-ohjelman läpimenoaika systeemistä, eli tuotteiden valmistumisaikojen maksimi C_{max}. Tehtävän formuloinnissa käytetyt merkinnät on esitetty taulukossa 2.

Taulukko 2: MILP-mallin formulaatiossa käytetyt merkinnät sekä niiden selitykset.

Merkintä	Selitys
I	Aikataulutettavien tuotteiden i joukko
I_k	Koneelle k aikataulutettavien tuotteiden joukko
$K_{i,h}$	Vaihtoehtoisten koneiden k joukko tuotteen i vaiheessa h
H_i	Tuotteen i tuotantovaiheiden joukko
$H_{i,k}$	Tuotteen i vaihe koneella k
$t_{i,k}$	Tuotteen i aloitusaika koneella k
$x_{i,j,k}$	Binäärimuuttuja: tuote i edeltää tuotetta j koneella k
R	Dummy-tuotteiden joukko, joka alustaa nykyisen tuotantotilanteen
R_k	Dummy-tuote koneella k
T_k	Dummy-tuotteen valmistuksen loppumisaika koneella k
M	Suuri positiivinen luku

Lineaarisen sekalukuoptimointimallin formuloinnin perustana on käytetty artikkeleita Moghaddas and Houshmand (2008), Ruiz and Andrés-Romano (2011) ja Choi and Choi (2002). Moghaddas ja Houshmand käsittelevät artikkelissaan JSS-ongelmaa järjestyksestä riippuvaisilla asetusajoilla ja erittelevät yksityiskohtaisesti malliin sisältyvät oletukset. Ruiz ja Andrés-Romano formuloivat MILP-mallin rinnakkaisille koneille järjestyksistä riippuvaisilla asetuajoilla. Choin ja Choin artikkelissa käsitellään yleistettyä JSS-ongelmaa järjestyksistä riippuvaisilla asetusajoilla, joka kuvaa testiongelmaa vastaavaa ympäristöä, jossa tuotteille voidaan määritellä eri vaiheisiin vaihtoehtoisia koneita. Tässä työssä testiongelma on kuitenkin formuloitu yhdistämällä elementtejä kahdesta ensimmäisestä artikkelista siten, että vaihtoehtoisten koneiden tapauksissa valinnat on mallinnettu samoja muuttujia käyttäen, kuin joilla tuotteiden peräkkäisyyksiä mallinnetaan. Choin ja Choin artikkelissa vaihtoehtoisten koneiden valintaa mallinnetaan omilla erillisillä muuttujilla, mikä lisää mallin muuttujien määrää. Työssä ei kuitenkaan esitetä vertailua eri formulaatioiden mahdollisista eroista laskentatehokkuudessa.

$$\min \quad C_{max}$$
$$x_{i,j,k}, t_{i,k}, C_{max}$$
$$s.t.$$

$$\sum_{k \in K_{j,h}} \sum_{i \in \{R,I\}} x_{i,j,k} = 1, \forall j \in I, \forall h \in H_j \tag{1}$$

$$\sum_{k \in K_{i,h}} \sum_{j \in I} x_{i,j,k} \leq 1, \forall i \in I, \forall h \in H_i \tag{2}$$

$$\sum_{j \in I_k} x_{R_k,j,k} \leq 1, \forall k \in K \tag{3}$$

$$\sum_{w \in \{R,I_k\}, w \neq i,j} x_{w,i,k} \geq x_{i,j,k}, \forall k \in K_i, \forall i,j \in I, i \neq j \tag{4}$$

$$t_{i,k} + p_{i,k} + s_{i,j,k} \leq t_{j,k} + (1 - x_{i,j,k})M, \forall k \in K, \forall i,j \in I, i \neq j \tag{5}$$

$$T_k + s_{R_k,j,k} \leq t_{j,k} + (1 - x_{R_k,j,k})M, \forall k \in K, \forall j \in I \tag{6}$$

$$t_{j,k} + p_{j,k} \leq t_{j,w} + (1 - \sum_{i \in \{R,I\}} x_{i,j,k})M + (1 - \sum_{i \in \{R,I\}} x_{i,j,w})M$$
$$, \forall j \in I, \forall k, w \in K_j | H_{j,w} = H_{j,k} + 1 \tag{7}$$

$$t_{i,k} + p_{i,k} \leq C_{max}, \forall i \in I, \forall k \in K_i \quad (8)$$

$$x_{i,j,k} \in \{0,1\}, \forall i,j,k \quad (9)$$

$$t_{i,k} \geq 0, \forall i,k \quad (10)$$

$$T_k = 0, \forall k \quad (11)$$

Mallin formulaatiossa määritellyt rajoitusehdot määräävät mallinnettavan systeemin toiminnan. Rajoitusehto 1 määrittää, että jokaista tuotetta $j \in I$ edeltää tasan yksi tuote sen jokaisessa tuotantovaiheessa. Ehto 2 sen sijaan määrää, että jokaista tuotetta sen jokaisessa vaiheessa seuraa enintään yksi tuote. Tähän liittyen ehto 3 rajoittaa, että jokaisen koneen dummy-tuotetta seuraa enintään yksi tuote. Jokaiselle koneelle on mallissa määritelty oma dummy-tuote. Näiden tuotteiden tarkoituksena on kuvata systeemin tila, johon se jää edellisen tuotanto-ohjelman jäljiltä, jolloin ensimmäisiksi aikataulutettaville tuotteille voidaan määritellä oikeat asetusajat ja koneiden vapautumisaikaa uuden ohjelman käyttöön voidaan hallita. Tässä työssä kaikkien koneiden kuitenkin oletetaan yksinkertaisuuden vuoksi vapautuvan samaa aikaa uudelle ohjelmalle, mikä on ilmaistu rajoitusehdossa 11. Rajoitusehto 4 linkittää tuotteet järjestykseen vaatimalla, että tuotteella i, jota prosessoidaan koneella k, täytyy olla koneella k tuote w, jota i välittömästi seuraa. Rajoitusehto 5 määrittää tuotteiden aloitusajat siten, että koneella valmistetaan kerrallaan vain yhtä tuotetta ja ehto 6 määrittää aikaisimman aloituksen tuotteille konekohtaisten alustusten mukaisesti. Rajoitusehdon 7 tarkoitus on pitää huolta, että tuotteet aikataulutetaan eri vaiheille oikeassa järjestyksessä. Suuren positiivisen luvun M kertoimet tekevät rajoituksesta merkityksettömän niissä tapauksissa, kun tuotetta ei ole aikataulutettu koneelle k tai w. Ehto 8 määrittää C_{max}:in tuotteiden valmistumisaikojen maksimiksi.

3.3 Implementoidut heuristiset menetelmät

Työssä sovellettiin muutamia yksinkertaisia heuristiikkoja testiongelman ratkaisemiseksi. Käytetyt menetelmät olivat: Monte Carlo -algoritmi, geneettinen algoritmi, sekä prioriteettisäännöt SPT ja SST. Monte Carlo -algoritmi on yksinkertainen arvausalgoritmi. Geneettinen algoritmi perustuu aikataulupopulaation evolutiiviseen kehitykseen. Prioriteettisäännöt luovat aikataulun esimerkiksi tuote-erien prosessointi- ja asetuaikoihin pohjautuvan yksinkertaisen säännön mukaan ilman stokastisuutta. Algoritmien sovellutukset testiongelman ratkaisemiseksi on esitelty seuraavissa alakappaleissa.

3.3.1 Monte Carlo -algoritmi

Monte Carlo -algoritmit ovat algoritmeja, jotka tuottavat epävarman tuloksen deterministisessä ajassa. Työssä toteutetun Monte Carlo -algoritmin toimintaperiaate on hyvin yksinkertainen. Algoritmi arpoo suuren, ennalta määrätyn, määrän kelpoisia aikatauluja ja palauttaa näistä aikatauluista läpäisyajaltaan parhaan.

Implementoidun Monte Carlo -algoritmin pseudokoodi on esitetty alla:

0. $C_{max} = \text{Inf}$, $x = [\ ]$, $n = 100000$.
1. Käy läpi lista tuotteita ja lisää jokainen tuote tarvittaville koneille tuotetyypin mukaan.
   - Jos valmistusvaiheessa tuotteella on vaihtoehtoisia koneita, arvo aikataulutettava kone.
2. Järjestä koneille lisätyt tuotteet konekohtaisesti satunnaiseen järjestykseen, tallenna muodostettu aikataulu x'.
3. Määritä aikataulun x' läpäisyaika C'_{max}.
4. Jos $C'_{max} < C_{max}$, niin $C_{max} = C'_{max}$ ja $x = x'$.
5. Toista vaiheet 1-4 n kertaa.
6. Palauta C_{max} ja x.

Odotusarvoisesti algoritmin antama tulos on sitä parempi mitä pidempi aika algoritmille annetaan laskentaan. Työssä arvottavien aikataulujen määränä käytetään 100 000 aikataulua, jotta kaikkien testiongelmien laskentaan kuluva aika pysyy lyhyehkönä. Yksittäistä ongelmaa ratkaistaessa toistojen määrää voidaan kuitenkin sallitun laskenta-ajan puitteissa kasvattaa rajatta, sillä toistojen lisääminen ei lisää algoritmin muistin tarvetta.

3.3.2 Geneettinen algoritmi

Työssä käytetty geneettinen algoritmi muodostui perinteisistä geneettisen algoritmin osista: kromosomipopulaatiosta, risteytysoperaattorista, mutaatiooperaattorista sekä sopivuusfunktioon perustuvasta valinnasta. Kromosomipopulaation kooksi valittiin työssä 1000 kromosomia ja sukupolvien määräksi 100 sukupolvea. Populaation koon ja sukupolvien määrän tulo sovitettiin

Monte Carlo -algoritmissa käytettyyn aikataulujen määrään (100 000) algoritmien vertailtavuuden parantamiseksi. Populaation koko valittiin sukupolvien määrää suuremmaksi, jotta populaatioista saatiin monimuotoisia, sillä sen uskottiin tuottavan laadukkaampia yksittäisiä tuloksia.

Geneettisen algoritmin toteutuksessa hyödynnettiin aikataulutettavan systeemin toiminnasta saatuja tietoja. Systeemin ominaisuudet vaikuttivat erityisesti kromosomisyntaksin määritelmään sekä risteytysoperaattorin toimintaan. Algoritmia muotoiltaessa todettiin, että aikataulun kromosomiesityksen rakentaminen koneen tarkkuudella oli monimutkaista ja kromosomeille tehdyt risteytykset vaativat uuden aikataulun kelpoisuuden säilyttämiseksi monivaiheisen korjausoperaattorin. Työssä päädyttiinkiin kromosomiesitykseen, jossa jokainen kromosomi koostuu kahdesta osasta: valmistusaikataulusta ja pakkausaikataulusta. Risteytysoperaattoriksi valittiin yhden pisteen risteytys, joka tässä sovelluksessa merkitsi sitä, että uusi kromosomi peri toiselta vanhemmalta kromosomin alkuosan (valmistusaikataulun) ja toiselta vanhemmalta kromosomin loppuosan (pakkausaikataulun). Valittu esitys oli yksinkertainen ja kuvasi systeemin valmistuprosessien ja pakkausprosessien irrallisuutta. Lisäksi oletettiin, että aikataulut, joiden toisen tai molempien prosessien aikataulut olivat mielekkäästi järjestyneet, erottuisivat valintaprosessissa keskimääräisistä aikatauluista ja siten entistä parempia aikatauluja voitaisiin muodostaa yhdistämällä erillisten prosessien mielekkäitä aikatauluja.

Valinta ja lisääntyminen toteutettiin käyttämällä kromosomin määrittämän aikataulun läpäisyajan käänteislukua sopivuusfunktiona. Valinnassa määriteltiin, että vain populaation sopivuusfunktioiden keskiarvoa paremmilla kromosomeilla oli mahdollisuus lisääntymiseen, koska aikataulujen arpomisen oletettiin luovan suuri määrä heikkolaatuisia aikatauluja. Näiden keskiarvoa parempien kromosomien sopivuusfunktioiden summa normeerattiin ykköseksi vanhempien arvonnan yksinkertaistamiseksi. Vanhemmat arvottiin siis normeerattuja sopivuusfunktioita painottaen pari kerrallaan ja arvottuun pariin sovellettiin määriteltyä risteytysoperaattoria yhden jälkeläisen tuottamiseksi. Vanhempia arvottiin jokaisessa sukupolvessa yhteensä 500 kappaletta. Uusi populaatio koostui sadasta edellisen sukupolven parhaasta kromosomista, 500 edellisen populaation vanhempien tuottamasta jälkeläisestä, sekä 400 uudesta arvotusta kromosomista.

Mutaatio-operaattoriksi määriteltiin toiminto, joka järjesti annetun aikataulun annetun koneen tuotteet satunnaiseen järjestykseen. Lisääntymisen jälkeen kaikkien 600 edellisestä populaatiosta periytyneen aikataulun jokaista konetta testattiin mutaation toteutumiseksi. Mikäli välille $[0, 1]$ tasaisesti ja-

kautuneesta jakaumasta arvottiin kynnysarvoa $P(mutaatio) = 0,1$ pienempi luku, sovellettiin aikataulun koneeseen mutaatio-operaattoria.

Implementoidun geneettisen algoritmin pseudokoodi on esitetty alla:

0. $x = [\,], C_{max} = [\,], Populaatio = 1000, Sukupolvet = 100, P(mutaatio) = 0.1, Parit = 500, Parhaat = 100.$
1. Luo alkupopulaatio kokoa *Populaatio* arpomalla ja määritä populaation sopivuusfunktiot.
2. Muodosta *Parit* määrä pareja lisääntymistä varten arpomalla kromosomeja sopivuusfunktiolla painotetusta jakaumasta.
3. Sovella risteytysoperaattoria jokaiseen pariin.
4. Muodosta uusi populaatio edellisen populaation parhaista yksilöistä (*Parhaat* kappaletta) sekä uusista jälkeläisistä (*Parit* kappaletta), ja täytä loppupopulaatio uusilla arvotuilla yksilöillä.
5. Suorita mutaatio koneelle todennäköisyydellä $P(mutaatio)$.
6. Määritä populaation sopivuusfunktiot.
7. Toista vaiheet 2-6 *Sukupolvet* kertaa.
8. Palauta paras aikataulu x ja sitä vastaava läpäisyaika C_{max}.

3.3.3 Prioriteettisäännöt

Työssä tutkittiin aikataulutusongelman ratkaisua myös prioriteettisääntöjen SPT (Shortest Processing Time) ja SST (Shortest Setup Time) sovelluksilla. Prioriteettisääntöihin perustuvassa aikataulutuksessa koneelle aikataulutetaan vapaana olevista tuote-eristä aina se, jonka prioriteetti on korkein ennalta määrätyllä säännöllä mitattuna. SPT-säännössä korkeimman prioriteetin saa tuote-erä, jonka prosessointi koneella kestää lyhyimmän ajan. Testiongelmaan SPT-sääntöä sovellettiin siten, että prosessointiaikaa mitattiin prosessiajan ja asetusajan summana. SST-säännössä korkeimman prioriteetin saa tuote-erä, jonka asetusaika koneella on lyhyin.

Prioriteettisääntöjen sovellutuksessa tuotteita vapautettiin aikataulutettaviksi koneiden jonoihin niiden valmistuessa edellisestä vaiheestaan. Seuraavaksi aikataulutettava tuote-erä valittiin aina kaikkien jonoissa olevien tuote-erien joukosta sen mukaan, millä tuote-erällä oli korkein prioriteetti. Vapaal-

le koneelle aikataulutettiin aina tuote-erä, mikäli kyseisen koneen jonossa oli yksi tai useampi tuote.

Prioriteettisäännön pseudokoodi on esitetty alla:

0. $C_{max} = \text{Inf}$, $x = [\ ]$, $t = 0$.
1. Lisää kaikki tuotteet ensimmäisten vaiheidensa kaikkien koneiden jonoon.
2. Tarkista onko hetkellä t vapaata konetta.
   - Jos on vapaa kone, valitse koneelle aikataulutettava tuote-erä prioriteettisäännön perusteella, lisää tuote-erä aikatauluun x, ja poista tuote-erä jonoista.
   - Jos ei vapaata konetta, edistä aikaa t seuraavan koneen vapautumiseen ja lisää valmistuva tuote tarvittaessa seuraavan vaiheen jonoihin.
3. Toista vaihetta 2 kunnes kaikkien tuotteiden kaikki vaiheet lisätty aikatauluun x.
4. Määritä aikataulun x läpäisyaika C_{max}.
5. Palauta aikataulu ja läpäisyaika.

4 Tulokset

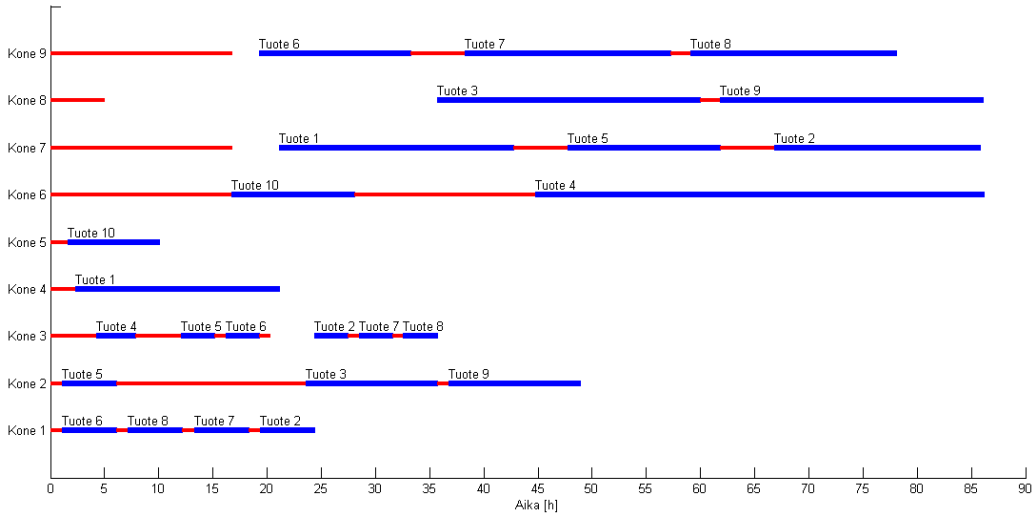
Tässä työssä eri ratkaisualgoritmien arvioinnin perustana käytettiin kahta kriteeriä: mallin antaman tuloksen laatua sekä laskenta-aikaa. Algoritmien arviointia varten rakennettiin joukko testiongelmia kuvaamaan erikokoisten tuotanto-ohjelmien aikataulutusta. Testiongelmia määritettiin yhteensä 130 kappaletta, 10 kappaletta jokaiselle tuotanto-ohjelman koolle 4-16 tuote-erää. Testiongelmat muodostettiin arpomalla kussakin ongelmassa aikataulutettavat erilaiset tuotetyypit.

Kaikki laskennat suoritettiin samalla tietokoneella (Intel Core 2 Duo 2,93 GHz; 2,0 GB RAM) käyttäen Matlab R2014a -ohjelmistoa. Laskenta-ajat mitattiin Matlabin *tic*- ja *toc*-funktioita käyttäen siten, että ajanotto aloitettiin, kun muuttujien ja parametrien alustus lähtötietojen perusteella oli tehty ja lopetettiin algoritmin ilmoittaessa tuloksen. Laskenta-ajan yksikkönä käytettiin geneettisen ja Monte Carlo -algoritmin tapauksessa sekuntia [s], prioriteettisääntöjen kanssa millisekuntia [ms], ja MILP-mallin kanssa minuuttia [min].

Heuristisen algoritmien antamien tulosten laatua mitattiin vertaamalla niitä sekä MILP-mallin että SST-prioriteettisäännön antamiin tuloksiin. MILP-mallilla saatiin globaalit optimit kaikille tuotanto-ohjelmille, joiden koko oli alle kahdeksan tuotetta. Kahdeksan tuotteen tuotanto-ohjelmista kuudelle kymmenestä saatiin optimaalinen tulos ja yhdeksän tuotteen ohjelmista ei yhdellekään. Kahdeksan ja yhdeksän tuotteen ohjelmille MILP-mallilla 360 minuutin laskenta-ajalla saatuja tuloksia käytettiin kuitenkin tässä tarkastelussa heuristisen menetelmien vertailukohtana, sillä tuloksien arvioitiin riittävällä tarkkuudella edustavan globaalia optimia. Koska MILP-mallin tuloksista saatiin vertailutuloksia vain pienille tehtäville, verrattiin heuristisen menetelmien tuloksia myös SST-säännön antamiin tuloksiin. SST-säännön tuloksien laadun hajonta oli kuitenkin suurta, minkä vuoksi tulosten tulkinnassa painotettiin vertailua MILP-mallin tuloksiin. Heuristisen menetelmien tulosten ja vertailuratkaisuiden erotus on kuvaajissa esitetty prosenttiosuutena vertailuratkaisusta.

Työssä luotiin funktio aikataulujen esittämiseksi matlab-kuvaajina. Kuvaaja on tehokas ja selkeä tapa aikataulun havainnollistamiseksi. Esimerkki funktiolla luodusta kuvaajasta on esitetty kuvassa 3, joka kuvaa erästä kymmenen tuotteen testiongelmaan geneettisellä algoritmilla saatua aikataulua. Kuvaajassa vaaka-akselilla esitetään aikaa tuotanto-ohjelman aloittamisesta ja pystyakselille on kuvattu tuotantosysteemin eri koneet. Paksulla sinisellä janalla on kuvattu tuote-erän prosessointiin käytetty aika koneella ja ohuem-

malla punaisella janalla tuotevaihtoon käytetty aika. Tuotteiden aloitusajat koneilla on piirretty aikaisimpaan mahdolliseen aloitusaikaan.



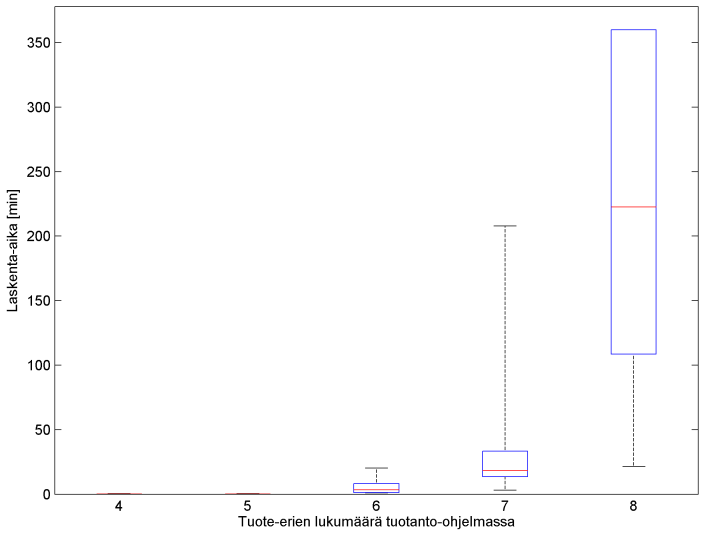
Kuva 3: Kymmenen tuotteen tuotanto-ohjelman aikataulu esitettynä Matlab-kuvaajana. Sininen väri merkitsee tuotteen prosessointia koneella ja punainen väri asetusaikaa.

4.1 MILP-mallin arviointi

MILP-mallin ratkaisemiseen käytettiin Matlabin uutta, Optimization Toolbox:in vuoden 2014 versiossa esiteltyä, *intlinprog*-funktiota. Laskennassa käytetyt funktiolle määritellyt asetukset olivat funktion oletusasetukset, poikkeuksena: 'Display','off', 'HeuristicsMaxNodes', 100, 'MaxTime', 21600, 'CutGenMaxIter', 25,'MaxNodes', 1e8. Funktion laskenta-aika per tehtävä oli rajoitettu 360 minuuttiin. Lisäksi ruudun päivitys oli estetty ja muutamien asetusten maksimikokoa oli kasvatettu oletusasetuksista testiongelmien laskennallisen haastavuuden vuoksi. Tarkoituksena oli ensisijaisesti tutkia mallin vaatiman laskenta-ajan kehitystä erikokoisilla tuotanto-ohjelmilla, sillä ratketessaan MILP-malli antaa globaalin optimin.

Laskenta-aikojen kehittyminen suhteessa aikataulutettavan tuotanto-ohjelman kokoon on esitetty boxplot-kuvaajan avulla kuvassa 4. Boxplot-kuvaajassa mediaanitulosta esittää punainen viiva laatikon sisällä, ensimmäistä ja kolmatta kvartiilia (25 % ja 75 % tuloksista) esittävät laatikon päädyt, ja viiksien päät merkkaavat tulosten minimiä ja maksimia. Kahdeksan tuotteen kokoisista tuotanto-ohjelman optimointitehtävistä vain kuusi kymmenestä rat-

kesi asetuksissa määritellyn maksimiajan ja maksimipisteiden puitteissa. Tuloksia tulkitessa täytyy siis huomioida, että tuotanto-ohjelman koon 8 osalta tulokset ovat todellisia arvoja pienempiä rajoitetusta laskenta-ajasta johtuen. Yhdeksän tuotteen tehtävistä yksikään ei ratkennut optimaalisuuteen asti asetusten rajoissa.



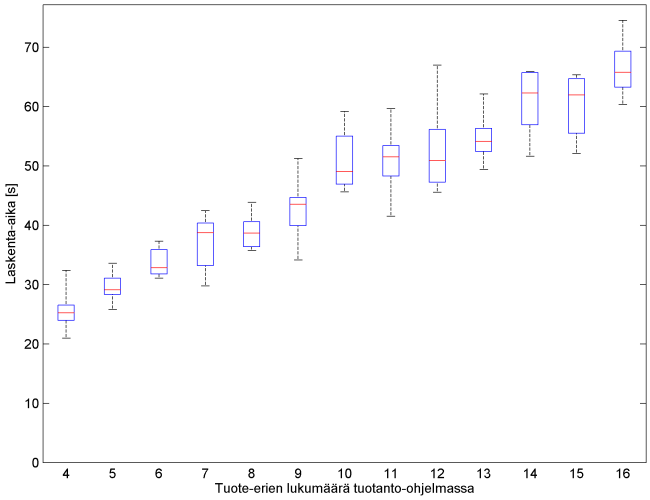
Kuva 4: MILP-mallin laskenta-ajat erikokoisilla testiongelmilla.

Tuloksista voidaan todeta, että testiongelman laskenta-aika MILP-mallia käyttäen kasvaa eksponentiaalisesti tuotanto-ohjelman koon kasvaessa. Tulosten mukaan malli ei käytännössä sovellu kuin pienten, alle kahdeksan tuotteen, tuotanto-ohjelmien aikatauluttamiseen. Laskenta-ajoissa huomionarvoista on myös se, että kaikilla tuotanto-ohjelmien kooilla maksimiaika on moninkertainen minimiaikaan nähden, yhtä kokoa lukuun ottamatta kaikissa yli kymmenkertainen. Suuren hajonnan laskenta-ajoissa selittävät todennäköisesti matlabin käyttämän *branch-and-bound* -algoritmin säännöt, joiden perusteella algoritmi valitsee seuraavaksi ratkaistavia alitehtäviä. Hajonnan tarkempaa syytä ei tässä työssä kuitenkaan selvitetä.

4.2 Monte Carlo -algoritmin arviointi

Monte Carlo -algoritmissa toistoiksi yhtä testiongelmaa kohti määritettiin 100 000 toistoa. Algoritmin laskenta-ajat suhteessa aikataulutettavan tuotanto-

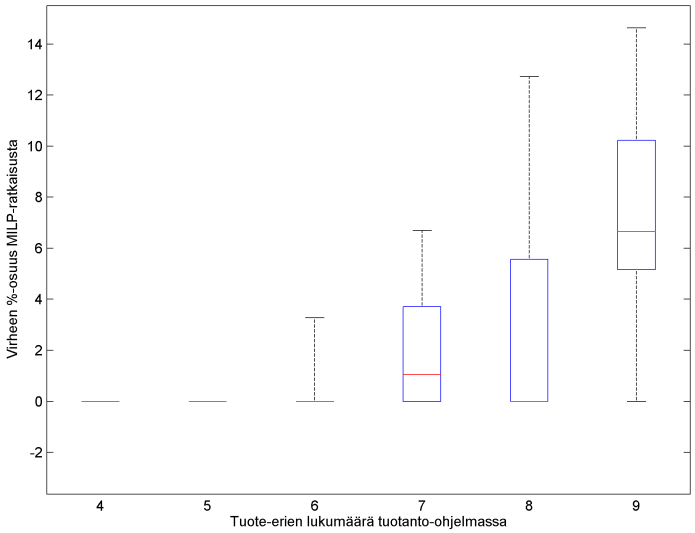
ohjelman kokoon on esitetty boxplot-kuvaajana kuvassa 5. Laskenta-ajat nousevat melko lineaarisesti tuotteiden lisääntyessä koko otoksen matkalta. Maksimi- ja minimiajat poikkeavat hieman toisistaan, mutta maksimiajat eivät millään tuotanto-ohjelman koolla ole kaksinkertaisia minimiaikoihin verrattuna. Kuvaajan perusteella voida arvioida laskenta-aikojen hajonnan kasvavan hieman aikataulutettavien tuotteiden määrän lisääntyessä.



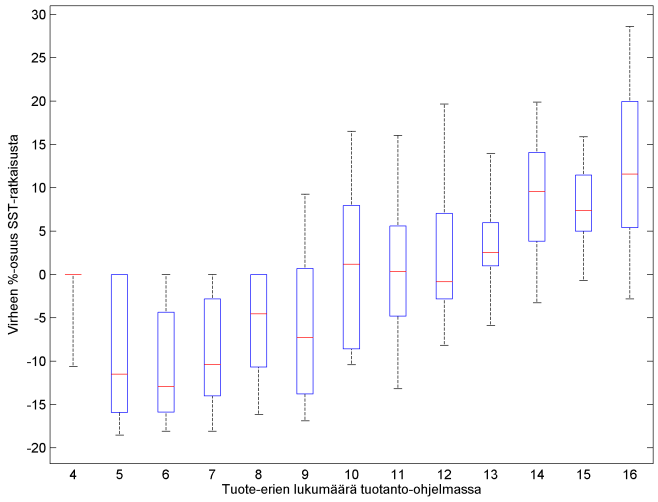
Kuva 5: Monte Carlo -algoritmin laskenta-aika aikataulutettavan tuotanto-ohjelman koon funktiona.

Monte Carlo -algoritmin tulosten ja MILP-mallin tulosten suhteellinen erotus on esitetty kuvassa 6. Kuvasta voidaan nähdä, että sekä virhe että virheen hajonta kasvavat aikataulutettavan tuotanto-ohjelman koon kasvaessa. Maksimivirhe kasvaa jyrkästi heti tuote-erien lukumäärästä 5 alkaen. Kolmannen kvartiilin kasvu on maksimivirhettä loivempaa. Jokaisella tuotanto-ohjelman koolla 4-8 algoritmilla saavutettiin vähintään neljässä testiongelmassa kymmenestä MILP-ratkaisun kanssa yhtä hyvä ratkaisu, ja koolla 9 yhdessä tapauksessa. Virheiden suuresta hajonnasta johtuen algoritmin tuloksen laatua on hyvin vaikea ennakoida, sekä jälkikäteen arvioida ilman tietoa todellisesta optimiratkaisusta.

Monte Carlo -algoritmin tulosten ja SST-prioriteettisäännön tulosten suhteellinen erotus on esitetty kuvassa 7. Erotuksen hajonta on hyvin suurta. Kuvaajasta voidaan kuitenkin nähdä, että tuotanto-ohjelman koosta 13 tuote-erää lähtien Monte Carlo -algoritmi alkaa keskimäärin tuottaa SST-sääntöä huonompia tuloksia.



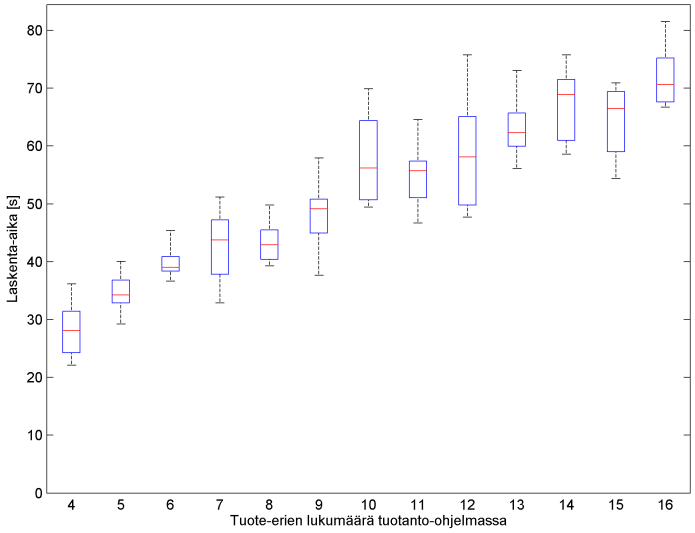
Kuva 6: Monte Carlo -algoritmin tuloksen suhteellinen erotus MILP-mallin tuloksesta aikataulutettavan tuotanto-ohjelman koon funktiona.



Kuva 7: Monte Carlo -algoritmin tuloksen suhteellinen erotus SST-prioriteettisäännön tuloksesta aikataulutettavan tuotanto-ohjelman koon funktiona.

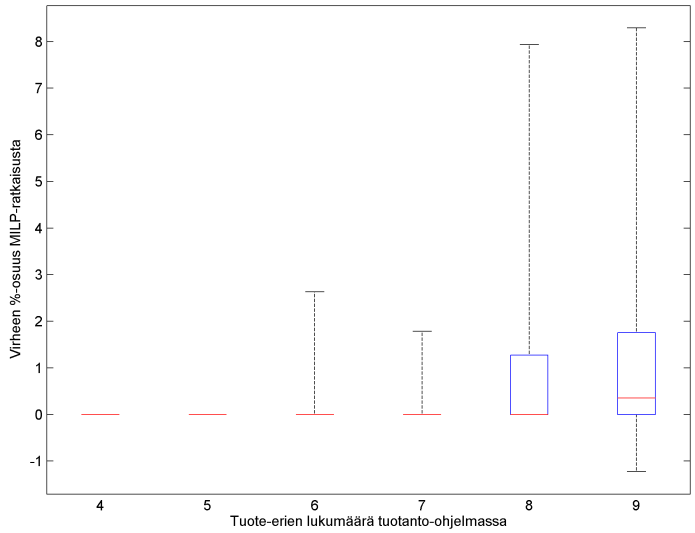
4.3 Geneettisen algoritmin arviointi

Geneettisen algoritmin laskenta-ajat suhteessa tuotanto-ohjelman kokoon on esitetty kuvassa 8. Mediaaniajoista voidaan nähdä, että laskenta-aika kasvaa lineaarisesti aikataulutettavan tuotanto-ohjelman koon kasvaessa. Maksimi- ja minimiajat pysyttelevät melko lähellä mediaaniaikoja, eikä aikojen hajonta kasva merkittävästi tuotanto-ohjelman koon kasvaessa. Geneettisen algoritmin laskenta-ajat ovat tasaisesti noin 5 sekuntia pidemmät, kuin Monte Carlo -algoritmilla.

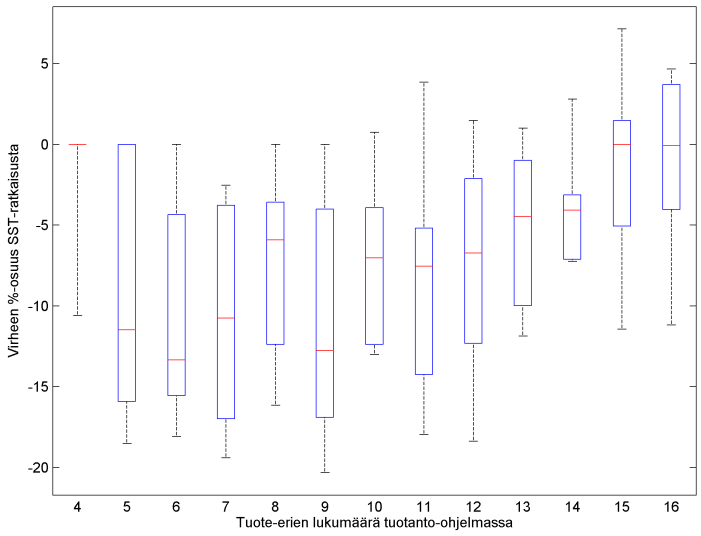


Kuva 8: Geneettisen algoritmin laskenta-aika aikataulutettavan tuotanto-ohjelman koon funktiona.

Geneettisen algoritmin ratkaisujen ja MILP-mallin ratkaisujen suhteellinen erotus on esitetty kuvassa 9. Käsitellyssä otoksessa, jonka suurimmat tuotanto-ohjelmat ovat kooltaan yhdeksän tuotetta, geneettisen algoritmin virheen keskiarvo pysyi hyvällä, maksimissaan 2,5 %, tasolla. Maksimivirhe nousee epätasaisesti ja melko voimakkaasti tuotanto-ohjelmien koon kasvaessa. Kasvu ei kuitenkaan ole yhtä voimakasta, kuin Monte Carlo -algoritmin tapauksessa. Mediaani virhe on 0 % ko'oilla 4-8 tuote-erää, ja koolla 9 alle 0,5 %. Yhdessä yhdeksän tuote-erän aikataulutusongelmassa geneettinen algoritmi löysi jopa epäoptimaalista MILP-ratkaisua paremman ratkaisun. Kokonaisuudessaan algoritmin virheen hajonta kasvaa tuontanto-ohjelman koon kasvaessa, mutta aineiston perusteella suurin osa ratkaisuista pysyy pienen vir-



Kuva 9: Geneettisen algoritmin antaman tuloksen ja MILP-mallin tuloksen suhteellinen erotus aikataulutettavan tuotanto-ohjelman koon funktiona.



Kuva 10: Geneettisen algoritmin antaman tuloksen ja SST-säännön tuloksen suhteellinen erotus aikataulutettavan tuotanto-ohjelman koon funktiona.

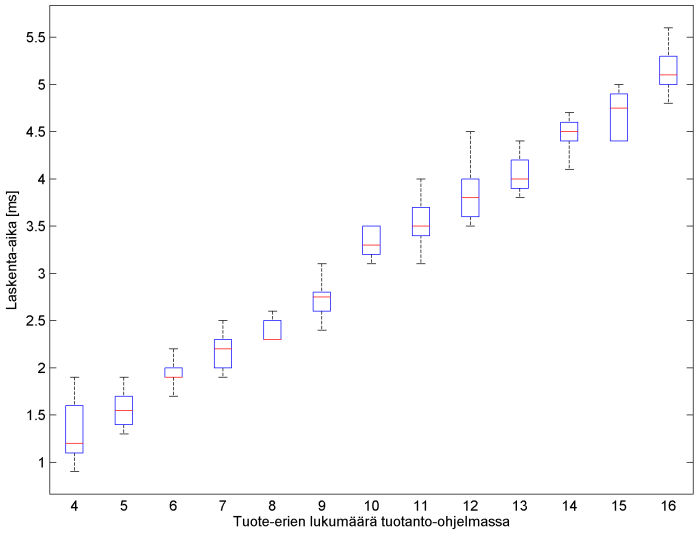
hemarginaalin sisällä. Otosta suurempien tuotanto-ohjelmien ratkaisujen laatua on kuitenkin vaikea arvioida aineiston pohjalta. Maksimivirheet edustavat yksittäisten ongelmien ratkaisuja, ja useampien koepisteiden kerääminen koetta toistamalla tasaisi todennäköisesti maksimivirheen epätasaista nousua ja antaisi paremman kuvan algoritmin tulosten luotettavuudesta.

Geneettisen algoritmin tulosten ja SST-prioriteettisäännön tulosten suhteellinen erotus on esitetty kuvassa 10. Suhteellisten erotusten hajonta on merkittävää, mistä johtuen tarkkoja johtopäätösiä on vaikea tehdä. Otoksesta käy kuitenkin ilmi, että keskiarvoisesti geneettinen algoritmi antaa SST-sääntöä paremman tuloksen kaikilla testin tuotanto-ohjelmien ko'oilla. Tuotanto-ohjelman koon edelleen kasvaessa olisi kuitenkin odotettavaa, että geneettisen algoritmin tulokset muuttuisivat SST-säännön tuloksia huonommiksi.

4.4 Prioriteettisääntöjen arviointi

SPT-prioriteettisäännön laskenta-ajat aikataulutettavan tuotanto-ohjelman koon funktiona on esitetty kuvassa 11. Kuvaaja esittää myös SST-säännön laskenta-aikoja, sillä algoritmien erot liittyvät ainoastaan tuote-erän prioriteetin laskentatapaan. SPT-säännössä prioriteetti määritettiin asetusajan ja prosessiajan summana, SST-säännössä käytettiin pelkkää asetusaikaa. Kuvaajasta voidaan nähdä, että laskenta-aikojen mediaani nousee lineaarisesti tuotanto-ohjelmien koon kasvaessa. Maksimi- ja minimilaskenta-ajat pysyvät erittäin lähellä keskiarvoista laskenta-aikaa eikä hajonnan kasvusta tuotanto-ohjelman koon kasvaessa ole viitteitä. Muiden menetelmien laskenta-aikoihin verrattuna prioriteettisääntöjen laskenta-ajat ovat erittäin lyhyitä. Keskiarvoisesti tulos saadaan alle neljässä millisekunnissa, kun geneettisen ja Monte Carlo -algoritmin keskimääräinen laskenta-aika on noin minuutin.

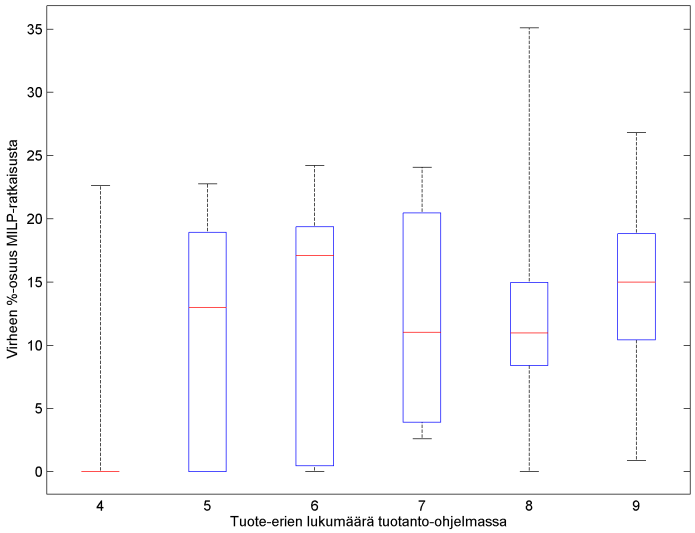
SPT-prioriteettisäännön antamien tulosten ja MILP-mallin tulosten suhteellinen erotus on esitetty kuvassa 12. Muista algoritmeistä poiketen SPT-säännön mediaanivirheessä ei ole havaittavissa selvää nousevaa trendiä, vaan mediaanivirhe pysyy noin 15 prosentissa. Koko otoksessa algoritmin maksimivirheet ovat erittäin suuria muihin menetelmiin verrattuna. Toisaalta minimivirheet pysyvät lähellä nollaa prosenttia. Algoritmin tulosten hajonta on erittäin suurta, mutta muista algoritmeistä poiketen hajonta näyttäisi pienentyvän tuotanto-ohjelman koon kasvaessa. SST-prioriteettisäännön tulosten ja MILP-mallin tulosten suhteellinen erotus on esitetty kuvassa 13. Tulokset näyttävät lähes identtisiltä SPT-säännön tuloksiin verrattuna, vain tulosten maksimivirhe on yhdessä tapauksessa SPT-sääntöä huomattavasti pienempi. Prioriteettisääntöjen ratkaisujen laatu näyttäisi riippuvan vah-



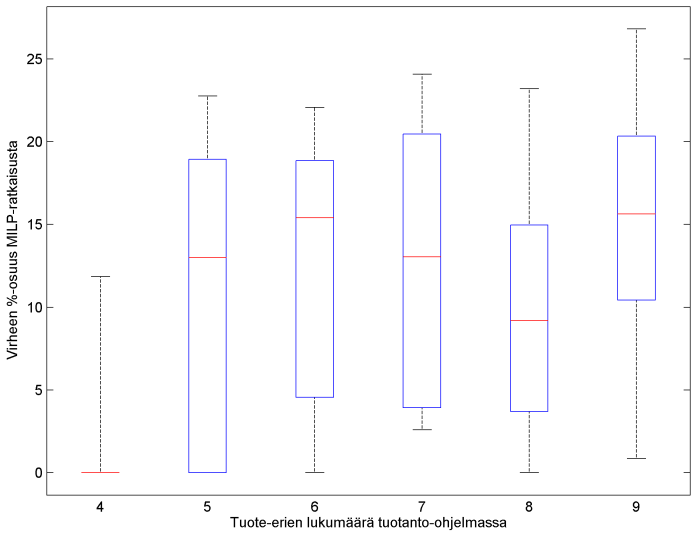
Kuva 11: SPT-säännön laskenta-aika aikataulutettavan tuotanto-ohjelman koon funktiona.

vasti aikataulutettavista tuote-eristä, sillä algoritmit itsessään ovat täysin deterministisiä. Testiongelmissa yksittäisillä aikataulutuspäätöksillä voi voimakkaasti vaihtelevista asetusajoista johtuen olla merkittävä vaikutus koko aikataulun laatuun.

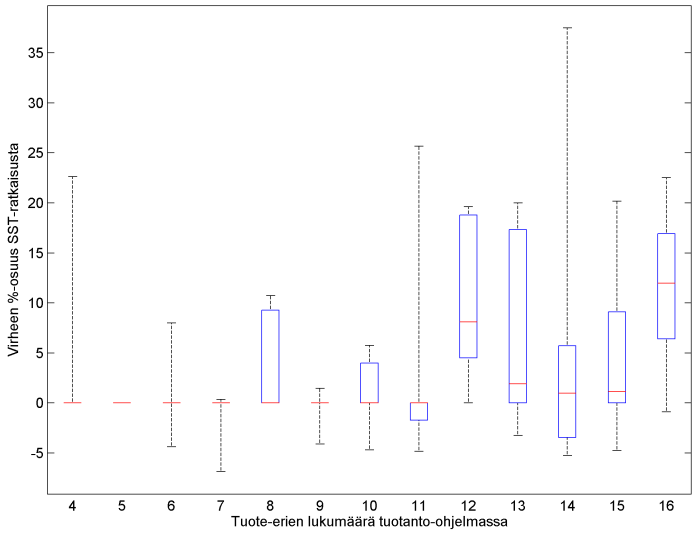
SPT-prioriteettisäännön tulosten suhteellinen ero SST-prioriteettisäännön tuloksiin on esitetty kuvassa 14. Kuvaajan tulokset vastaavat pienten tehtävien osalta kuvista 12 ja 13 tehtyjä päätelmiä prioriteettisääntöjen melko tasaisesta suoriutumisesta. Tuotanto-ohjelmien koon kasvaessa SST-sääntö erottuu SPT-sääntöä selkeästi paremmin suoriutuvana. Yhdessäkään tehtävässä SPT-sääntö ei ole yli viittä prosenttia parempi kuin SST-sääntö, mutta SST-sääntö on jopa 35 % SPT-sääntöä parempi. Tulosten hajonta on kuitenkin suurta, eikä mediaanierotuksessa ole havaittavissa selkeää trendiä. Kuvaajan tuloksilla perusteltiin SST-prioriteettisäännön valinta benchmarkheuristiikaksi SPT-säännön sijasta.



Kuva 12: SPT-säännön antaman tuloksen suhteellinen erotus MILP-mallin tuloksesta aikataulutettavan tuotanto-ohjelman koon funktiona.



Kuva 13: SST-säännön antaman tuloksen suhteellinen erotus MILP-mallin tuloksesta aikataulutettavan tuotanto-ohjelman koon funktiona.



Kuva 14: SPT-säännön antaman tuloksen suhteellinen erotus SST-säännön tuloksesta aikataulutettavan tuotanto-ohjelman koon funktiona.

5 Tarkastelu ja johtopäätökset

Työssä määriteltiin valmistus-pakkaus-prosessille aikataulutusongelma ja tutkittiin testiongelman ratkaisemista erilaisilla tuotteilla ja tuote-erien määrillä. Aikataulutusongelman ratkaisemiseen käytettiin lineaarista sekalukuoptimointimallia sekä neljää heuristiikkaa: Monte Carlo -algoritmia, geneettistä algoritmia, sekä SPT- ja SST-prioriteettisääntöjä. Eri ratkaisumenetelmiä arvioitiin niiden laskenta-aikojen sekä ratkaisujen laatujen perusteella.

Työssä havaittiin, että lineaarisen sekalukuoptimointimallin laskenta-aika kasvoi eksponentiaalisesti aikataulutettavan tuotanto-ohjelman koon kasvaessa. Työssä suurimmat optimaalisuuteen asti ratkaistut tuotanto-ohjelmat olivat kooltaan 8 tuotetta, joka on todellisissa sovelluksissa pieni tuotemäärä. Laskenta-ajan kehitys vastasi kirjallisuudessa esitettyjä arvioita lineaarisen sekalukumallin riittämättömyydestä käytännöllisen kokoisten ongelmien ratkonnassa. MILP-mallin etuna on kuitenkin se, että optimaalinen tulos on taattu ratkaistavissa oleville tehtäväko'oille. Lisäksi MILP-malliin on helppo tarvittaessa lisätä uusia rajoituksia, jotka koskevat yksittäisten tuotteiden valmistumisia tai aloituksia eri tuotantovaiheissa, esimerkiksi jos jokin tuote täytyy saada määräaikaan menneessä valmiiksi tai tietyn tuotteen raaka-aineet ovat saatavilla vasta määritellyn ajan jälkeen.

Monte Carlo -algoritmin parhaita puolia olivat sen yksinkertainen implementointi sekä lineaarisesti kasvava laskenta-aika. Algoritmin rakenne ei itsessään suosinut minkäänlaisia aikatauluja, vaan tuotetut aikataulut olivat aidosti satunnaisia. Myös algoritmin virittäminen oli yksinkertaista, sillä ainoa muutettava parametri oli luotavien aikataulujen määrä.

Geneettinen algoritmi oli työssä suoritettujen testien perusteella kaikkein toimivin tapa testiongelman ratkaisemiseen. Sen laskenta-aika kasvoi lineaarisesti ongelman kokoon nähden ja testien perusteella ratkaisujen suhteellinen virhe pysyi hyvin maltillisena. Geneettistä algoritmia implementoitaessa jouduttiin kuitenkin tekemään monta valintaa heuristiikan rakenteesta ja parametreista, jotka vaikuttavat algoritmin suorituskykyyn. Todennäköisesti algoritmi olisi siis viritettävissä paremmin toimivaksi, mutta viritysongelma on monimutkainen parametrien ja rakenteellisten ratkaisujen määrästä ja mahdollisista yhteisvaikutuksista johtuen.

SPT-prioriteettisääntö esitettiin kirjallisuudessa yhtenä benchmark-heuristiikkana, johon uusia heuristiikkoja usein verrataan. Työssä SPT-sääntö todettiin kuitenkin hyvin epävarmaksi menetelmäksi testiongelman ratkaisemiseen. SPT-sääntöä paremmaksi prioriteettisäännöksi todettiin SST-sääntö,

joka huomioi vain tuote-erän asetusajan. Prioriteettisääntöjen laskenta-ajat olivat muihin menetelmiin verrattuna erittäin lyhyitä, joten säännöistä voisi olla mahdollista kehittää erilaisia hybridiversioita, joissa esimerkiksi tasatilanteissa tai tuotteiden allokoinnissa vaihtoehtoisille koneille hyödynnettäisiin arvontaa ja toistoa. Arpomalla aikataulutettava kone vaihtoehtoisia koneita sisältävissä tuotantovaiheissa testiongelma pelkistyisi standardiksi JSS-ongelmaksi tuotejärjestyksestä riippuvaisilla asetusajoilla, jonka ratkaisemiseen prioriteettisäännöt voisivat soveltua testiongelmaa luontevammin.

Optimointialgoritmien tulosten käytettävyys todellisissa aikataulutusongelmissa riippuu laskenta-ajan ja tuloksen laadun lisäksi mallin perustana olevien oletusten soveltuvuudesta käytännön tuotantosysteemiin. Tässä työssä oletettiin esimerkiksi, ettei koneiden saatavuus riipu työvoiman saatavuudesta ja, että asetus- ja prosessiajat ovat deterministisiä. Todellisen systeemin lisärajoitukset tarkoittaisivat, ettei optimointialgoritmin tulosta voitaisi sellaisenaan käyttää tuotannon ohjaamiseen. Aikataulun laatija voisi kuitenkin mahdollisesti käyttää yksinkertaistetun systeemin antamaa aikataulua monimutkaisemman systeemin aikataulun laatimisessa eräänlaisena alkuarvauksena. Yksinkertaistetun systeemin optimiaikataulu on myös todellisen systeemin optimiaikataulu, jos aikataulu toteuttaa todellisen systeemin asettamat lisärajoitukset. Toisaalta esimerkiksi todellisen systeemin työvoimarajoitteista on pitkällä tähtäimellä mahdollista päästä eroon palkkaamalla lisää osaavaa työvoimaa. Tuotantosysteemin kokonaisoptimiin vaikuttavat erityisesti systeemin toiminnalla asetetut tavoitteet sekä sen kustannusrakenne.

Työssä erityiseksi ongelmaksi havaittiin heuristiikkojen ratkaisujen laadun arviointi suurilla tuotanto-ohjelmilla. Pienet ohjelmat voitiin ratkaista MILP-mallilla optimaalisuuteen saakka, jolloin heuristiikkojen vertailu oli helppoa suhteessa optimitulokseen. Suuremmilla ohjelmilla benchmark-tuloksena käytettiin SST-prioriteettisääntöä, mutta vertailusta ei voitu tehdä tarkkoja päätelmiä tulosten suuren hajonnan takia. Kirjallisuuden mukaan SPT-sääntöä käytetään usein vertailutuloksena, mutta työssä päädyttiin käyttämään vertailutuloksena SST-sääntöä sen laadukkaampien tulosten takia. Testiongelma oli luonteeltaan haastava, sillä yksittäisillä aikataulutuspäätöksillä saattoi olla voimakkaasti vaihtelevien asetusaikojen takia merkittävä vaikutus koko tuotanto-ohjelman pituuteen. Toisaalta juuri tästä syystä toimivan aikataulutusalgoritmin löytäminen testiongelmalle olisi mielekästä.

Mahdollisia mielenkiintoisia jatkotutkimuksia testiongelman ratkaisemiseksi voisivat olla uudenlaisten heuristiikkojen soveltaminen, hybridialgoritmien rakentaminen redusoidun tehtävän ratkaisemiseksi, sekä geneettisen algoritmin parametrien virittäminen. Uusista heuristiikoista tabu etsintä, jota Choi

and Choi (2002) käyttivät benchmark-heuristiikkana, ja simuloitu jäähdytys vaikuttivat lupaavimmilta. Hybridialgoritmeista MILP-sovellutukset, joissa tuotanto-ohjelma optimoidaan lyhyissä pätkissä, sekä prioriteettisääntösovellukset, joissa ongelma redusoidaan standardiksi JSS-ongelmaksi, olisivat kiinnostavia tutkimuksen kohteita. Kiinnostavaa olisi myös tutkia, miten työssä implementoidun geneettisen algoritmin toimintaa voitaisiin parantaa algoritmin parametrejä virittämällä.

Mikäli työn tuloksia sovellettaisiin sellaisinaan testiongelman kuvaaman tuotantoprosessin ohjaamiseen, kannattaisi aikataulun optimointiin testien perusteella käyttää geneettistä algoritmia. Todennäköisesti geneettisen algoritmin tulosten laatu paranisi hieman populaation kokoa ja sukupolvien määrää kasvattamalla. Populaation koko ja sukupolvien määrä määräävät myös algoritmin laskenta-ajan. Käytännön sovellutuksessa pitäisi siis tasapainottaa käyttäjän tarpeet suhteessa ratkaisun laatuun ja laskenta-aikaan. Käytännössä kannattaisi lisäksi aina laskea myös SST- ja SPT-sääntöjen antamat aikataulut, sillä prioriteettisäännöt antavat toisinaan geneettistä algoritmia paremman ratkaisun ja niiden laskenta-ajat ovat erittäin lyhyitä. Prosessin ohjaamiseen voitaisiin tällöin valita paras aikataulu kolmesta kandidaattiaikataulusta.

Viitteet

A. Allahverdi and H.M. Soroush. The significance of reducing setup times/setup costs. *European Journal of Operational Research*, 187:978–984, 2008.

D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, 3(2):149–156, 1991.

K. Bülbül and P. Kaminsky. A linear programming-based method for job shop scheduling. *Journal of Scheduling*, 16:161–183, 2013.

J.W. Chinneck. Practical optimization: A gentle introduction, 2012. URL `www.sce.carleton.ca/faculty/chinneck/po.html`.

I.-C. Choi and D.-S. Choi. A local search algorithm for jobshop scheduling problems with alternative operations and sequence-dependent setup times. *Computers and Industrial Engineering*, 42:43–58, 2002.

R. Haupt. A survey of priority rule-based scheduling. *OR Spektrum*, 11:3–16, 1989.

K. Mesghouni, P. Pesin, D. Trentesaux, S. Hammadi, C. Tahon, and P. Borne. Hybrid approach to decision-making for job shop scheduling. *Production Planning Control*, 10(7):690–706, 1999.

R. Moghaddas and M. Houshmand. Job-shop scheduling problem with sequence dependent setup times. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2:1546–1552, 2008.

R. Ruiz and C. Andrés-Romano. Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 57:777–794, 2011.

C.R. Vela, R. Varela, and M.A. González. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16:139–165, 2010.

J.P. Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, ilmestyy 2014.

Liite A: Tuotantoprosessien parametrit

Prosessointiajat

Taulukko 3: Testiongelman tuotteiden i prosessointiajat $p_{i,k}$ koneilla k. Prosessointiaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei rivin tuotetta voi valmistaa sarakkeen koneella.

Tuote \ Kone	Kone 1	Kone 2	Kone 3	Kone 4	Kone 5	Koneet 6, 7, 8, 9
A	-	-	-	-	8,5	11,4
B	-	-	-	18,8	-	21,7
C	12,1	12,1	-	-	-	24,3
D	-	-	3,6	-	-	41,4
E	6,1	6,1	-	-	-	17,9
F	5,0	-	-	-	-	14,7
G	5,0	5,0	3,1	-	-	14,0
H	5,0	5,0	3,1	-	-	19,0

Asetusajat

Taulukko 4: Testiongelman tuotteiden asetusajat $s_{i,j,1}$ koneella 1, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-
C	-	-	1,1	-	17,5	17,5	17,5	17,5
D	-	-	-	-	-	-	-	-
E	-	-	17,5	-	1,1	17,5	17,5	17,5
F	-	-	17,5	-	17,5	1,1	17,5	17,5
G	-	-	17,5	-	17,5	17,5	1,1	1,1
H	-	-	17,5	-	17,5	17,5	1,1	1,1

Taulukko 5: Testiongelman tuotteiden asetusajat $s_{i,j,2}$ koneella 2, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-
C	-	-	1,1	-	17,5	17,5	17,5	17,5
D	-	-	-	-	-	-	-	-
E	-	-	17,5	-	1,1	17,5	17,5	17,5
F	-	-	-	-	-	-	-	-
G	-	-	17,5	-	17,5	17,5	1,1	1,1
H	-	-	17,5	-	17,5	17,5	1,1	1,1

Taulukko 6: Testiongelman tuotteiden asetusajat $s_{i,j,3}$ koneella 3, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-
C	-	-	-	-	-	-	-	-
D	-	-	-	1,0	-	-	4,3	4,3
E	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-
G	-	-	-	4,3	-	-	1,0	1,0
H	-	-	-	4,3	-	-	1,0	1,0

Taulukko 7: Testiongelman tuotteiden asetusajat $s_{i,j,4}$ koneella 4, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	-	-	-	-	-	-	-	-
B	-	2,3	-	-	-	-	-	-
C	-	-	-	-	-	-	-	-
D	-	-	-	-	-	-	-	-
E	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-
G	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	-	-

Taulukko 8: Testiongelman tuotteiden asetusajat $s_{i,j,5}$ koneella 5, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	1,6	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-
C	-	-	-	-	-	-	-	-
D	-	-	-	-	-	-	-	-
E	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-
G	-	-	-	-	-	-	-	-
H	-	-	-	-	-	-	-	-

Taulukko 9: Testiongelman tuotteiden asetusajat $s_{i,j,k}$ koneilla $k \in \{6, 7, 8, 9\}$, kun tuotetta i seuraa tuote j. Asetusaikojen yksikkö on tunti [h]. Merkintä "-" taulukon solussa tarkoittaa, ettei kyseinen tuotevaihto ole mahdollinen koneella.

Tuoteeesta i \ Tuotteeseen j	A	B	C	D	E	F	G	H
A	1,8	16,7	16,7	16,7	16,7	16,7	16,7	16,7
B	16,7	1,8	5,0	16,7	16,7	16,7	5,0	5,0
C	16,7	5,0	1,8	16,7	16,7	16,7	16,7	16,7
D	16,7	16,7	16,7	1,8	16,7	16,7	5,0	5,0
E	16,7	16,7	16,7	16,7	1,8	16,7	16,7	16,7
F	16,7	16,7	16,7	16,7	16,7	1,8	16,7	16,7
G	16,7	16,7	16,7	16,7	16,7	16,7	1,8	5,0
H	16,7	16,7	16,7	16,7	16,7	16,7	5,0	1,8