Painettujen piirilevyjen ladontalinjojen tasapainotusongelmasta

TURUN YLIOPISTO
Informaatioteknologian laitos
Tietojenkäsittelytiede
Pro gradu -tutkielma
Jani Koskinen
2012

TURUN YLIOPISTO
Informaatioteknologian laitos

JANI KOSKINEN: Painettujen piirilevyjen ladontalinjojen tasapainotusongelmasta

Pro gradu -tutkielma, 124 s., 6 liites.
Tietojenkäsittelytiede
Kesäkuu 2012

Elektroniikkateollisuuteen kohdistuvat odotukset pakottavat valmistajia kehittämään pitkälle automatisoituja ja tehokkaita tuotantoratkaisuja. Ladontakoneiden monipuolistuminen vaikeuttaa kuitenkin tuotannon tehostamista.

Tämä tutkielma sisältää katsauksen nykyaikaiseen piirilevyladonnan tuotantoon ja siihen liittyvään töiden tasapainotusongelmaan. Piirilevyladonnan käsitteiden ja tasapainotusongelman perusperiaatteiden selvittämisen lisäksi pohditaan seikkoja, jotka muodostavat varsinaisen ratkaistavan ongelman. Näistä seikoista käsitellään tahtiajan minimointia, töiden sijoitusta usealle linjalle, töiden julkaisu- ja määräaikojen noudattamista sekä tehtäväjoukon dynaamisuutta eli työjoukon muuttumista kesken tuotannon.

Tässä työssä kerrotaan myös eri ratkaisumenetelmistä ja niiden soveltuvuudesta ongelmien ratkaisemiseen. Esimerkkien avulla tarkastellaan myös, miten osa menetelmistä toimii (esim. inkluusiopuun käyttö). Tämän työn lopuksi kehitetään yksinkertainen töidenjärjestelyalgoritmi usean linjan linjastolle ja tutkitaan testiajojen avulla sen soveltuvuutta eri tilanteille.

Asiasanat: pcb, piirivelyladonta, tasapainotus, töidenjärjestely

UNIVERSITY OF TURKU
Department of Information Technology

JANI KOSKINEN: Workload balancing of PCB manufacturing lines

Master's thesis, 124 p., 6 app. p.
Computer Science
June 2012

Expectations towards modern electronic industry are forcing manufacturers to develop highly automated and efficient production solutions. While component placement machines using surface mounted technology are becoming more versatile, it also makes production optimizing more difficult.

This thesis work reviews the basics of modern PCB manufacturing and work balancing problem related to it. In addition to introducing the basic concepts and principles of the balancing problem this work covers the issues that form the actual solvable problem. These issues include cyle time minimization, work distribution to production lines and acknowledging release times and deadlines but also considering the variability of job set. The variability allows new jobs to be added while production is ongoing.

This work covers some methods that can be used to solve the work balancing problem and consideres whether these methods are applicable in this thesis. Examples are used to clarify these methods (e.g. inclusion tree). In the final part of the thesis, simple algorithm is created for job balancing problem when multiple lines in production network are used. A number of test runs are used to evaluate the efficiency of the algorithm.

Keywords: job dispatching, load balancing, pcb, manufacturing

Sisältö

1 Johdanto **1**

2 Katsaus tuotantotekniikoihin **3**
2.1 Käsitteet . . . 3
2.2 Ladontaprosessi ja sen valmistelut . . . 4
2.3 Ladontaprosessien hallinta . . . 5
2.4 Tuotantotavat ja niiden vaikutukset töidenjärjestelyyn . . . 6
2.4.1 Yhden linjan malli (Single line) . . . 6
2.4.2 Rinnakkaismalli (Parallel lines) . . . 7
2.4.3 Sarjavalmistusmalli (Flow shop) . . . 7
2.4.4 Konepajamalli (Job shop) . . . 7
2.4.5 Toimitusketjumalli (Supply chain) . . . 8
2.5 Tuotannon ja sen järjestelyn tavoitteet . . . 9

3 Tuotantolinjojen tasapainoittaminen **11**
3.1 Käsitteet . . . 11
3.2 Yleiset ongelmatapaukset . . . 13
3.3 Muita ongelmatyyppejä . . . 14
3.3.1 Monta erilaista tuotetta . . . 15
3.3.2 Linjan tai linjaston hallinta . . . 16
3.3.3 Tehtäväaikojen muuttuvuus . . . 16
3.3.4 Tuotantolinjan ja linjaston muoto . . . 17
3.3.5 Tuotantolinjojen rinnakkaisuus . . . 18
3.3.6 Varustelu- ja prosessointivaihtoehdot . . . 20
3.3.7 Rajoitukset työn jaossa . . . 20
3.3.8 Tavoitteiden erilaisuus . . . 21
3.4 Linjojen tasapainoituksesta . . . 21

4 Tuotantolinjojen tasapainotusongelman ratkaisumenetelmistä **25**
4.1 Matemaattiset ohjelmointimenetelmät . . . 26
4.1.1 Lineaarinen ohjelmointi . . . 26
4.1.2 Kokonaislukuohjelmointi . . . 27
4.1.3 Epälineaarinen ohjelmointi . . . 27
4.1.4 Disjunktiivinen ohjelmointi . . . 28
4.2 Tarkat optimointimenetelmät . . . 28
4.2.1 Dynaaminen ohjelmointi . . . 29
4.2.2 IP-optimointi . . . 29
4.3 Heuristiset menetelmät . . . 31

4.3.1 Sääntöpohjaiset tekniikat . . . 32
4.3.2 Geneettiset algoritmit . . . 32
4.4 Rajoiteohjelmointimenetelmä . . . 35

5 Tuotantolinjojen tasapainotusongelmien ratkaiseminen 37
5.1 Yksi linja . . . 37
5.1.1 Ladontatehtävien jako koneille . . . 39
5.1.2 Komponenttien asettelustrategioista . . . 41
5.1.3 Minimiasettelu . . . 43
5.1.4 Rajoitukset komponenttien valinnassa . . . 47
5.2 Monen linjan linjasto . . . 48
5.2.1 Ryhmittelymenetelmät tasapainotuksessa . . . 50
5.2.2 Samankaltaisuusmitat . . . 51
5.2.3 Piirilevyryhmien muodostaminen . . . 52
5.2.4 Inkluusiopuun käyttö . . . 53
5.2.5 Esimerkki tasapainotuksesta . . . 55
5.2.6 Matemaattinen malli . . . 61
5.3 Tuotteiden sijoittelut linjoille . . . 68
5.4 Aloitus- ja määräaikojen noudattamisesta . . . 69
5.5 Jatkuva työmäärän kasvu . . . 72

6 Algoritmit 74
6.1 Järjestelmän mallinnuksesta . . . 74
6.2 Heuristinen tehtävänjakoalgoritmi . . . 75
6.2.1 Algoritmin toimintaperiaate . . . 75
6.2.2 Algoritmin vaiheet ja yksityiskohdat . . . 77
6.2.3 Pseudokoodi . . . 82
6.2.4 Algoritmin ongelmat ja parantaminen . . . 86
6.3 Geneettisen algoritmin käyttö . . . 87
6.3.1 Aakkoston kehittäminen ja operaatiot . . . 87
6.3.2 Hyvyysfunktion laatiminen . . . 88
6.3.3 Hybridialgoritmit . . . 89

7 Testiajot ja -tulokset 90
7.1 Rajoitukset ja yksinkertaistukset . . . 90
7.2 Testiajoissa käytetyt linjastot ja yhteiset syötteet . . . 91
7.3 Testiajo 1 . . . 93
7.3.1 Inkluusiopuu ja ryhmät . . . 93
7.3.2 Alustava töidenjärjestely . . . 96
7.3.3 Linjojen tasapainotus . . . 99
7.4 Testiajo 2 . . . 103

7.5 Testiajo 3 109
7.6 Ohjelma ja tulosten analysointi 120

8 Johtopäätökset **121**

Lähteet **122**

Liite 1: Testiajo 2:n piirilevyt **125**

Liite 2: Testiajo 2:n työt **126**

Liite 3: Testiajo 2:n lopulliset aikataulut **128**

1 Johdanto

Nykyajan elektroniikkateollisuudessa tuotanto on erittäin pitkälle automatisoitu. Elektroniikkateollisuuden päätuotteet eli piirilevyt kootaan nykyisin pitkälle automatisoiduilla linjastoilla. Nämä linjastot koostuvat useista ladontakoneista, jotka ovat korvanneet ihmiset jo kauan sitten ja suorittavat työn ylivertaisella nopeudella ja tarkkuudella. Elektroniikkateollisuuteen kohdistuu kuitenkin jatkuvasti kasvavia odotuksia. Osa näistä odotuksista kohdistuu nimenomaan valmistukseen, erityisesti hintaan ja nopeuteen. Näiden odotusten luomat paineet pakottavat valmistajat kehittämään tuotantoratkaisujaan aina pidemmälle ja pidemmälle, jotta tuotanto saataisiin mahdollisimman tehokkaaksi.

Ladontakoneet ovat kehittyneet niiden alkuajoista asti. Komponenttien ladontanopeus on kasvanut ja koneiden komponenttikapasiteetti on kasvanut. Lisäksi ladontakoneissa on nykyisin erikokoisia suuttimia, joilla kone pystyy latomaan niin isot piirit kuin pienimmätkin komponentit. Ladontakoneet ovat nykyään niin monipuolisia, että ainoastaan muutaman kappaleen prototyyppierät kannattaa tehdä käsin.

Ladontakoneiden monipuolisuus tuo uusia haasteita tuotannon suunnitteluun. Vaikka ladontakoneet pystyvät latomaan mitä erilaisimpia komponentteja, niiden komponenttien varastointikapasiteetti on silti rajallinen. Komponentit on yleisesti varastoitu keloihin ja kussakin ladontakoneessa on tietty määrä paikkoja keloille. On tärkeää, että oikeat komponentit ovat saatavilla oikeaan aikaan. Tämän mahdollistaa huolellinen tuotannon suunnittelu. Nykyajan ladontaprosessien suunnitteluun kuuluu niin tuotantolinjojen tasapainotus, töidenjärjestely kuin komponenttiasettelujen hallintakin.

Luvussa 2 käydään läpi tuotantotekniikoita ja niihin liittyviä asioita. Erityisesti käsitellään komponenttien ladontakoneita ja niistä muodostettavia linjastoja. Ladontakoneista esitellään niiden keskeisiä ominaisuuksia kuten ladonnan ja komponenttien hallintaan liittyviä asioita. Tuotantolinjastojen osalta käydään läpi eri konfiguraatiomallit ja käsitellään niiden vaikutuksia töidenjärjestelyyn. Tässä luvussa selvitetään tulevia lukuja ajatellen myös tuotannon ja sen järjestelyn tavoitteet sekä töidenjärjestelyongelmien perus-

teet.

Luku 3 käsittelee linjojen tasapainotusongelmaan liittyviä seikkoja. Ongelmaa käydään ensin läpi yleisesti ja sen jälkeen tutustutaan tasapainotusongelman eri versioihin. Samalla tutkitaan tarkemmin, miten monen linjan töidenjärjestely muuttuu, kun otetaan mukaan ennalta määrättyjä rajoitteita. Tällainen on esimerkiksi tiettyjen linjojen omistaminen tietylle tuotteelle tai työlle.

Luvussa 4 käydään läpi menetelmiä ja tekniikoita, jotka on todettu käyttökelpoisiksi ratkaistaessa tasapainotukseen ja töiden järjestelyyn liittyviä ongelmia. Tarkastelussa ovat matemaattiset ja rajoiteohjelmointimenetelmät, tarkat optimointimenetelmät sekä geneettiset algoritmit. Myös hyväksi havaittuja sääntöpohjaisia töidenjärjestelymenetelmiä käydään läpi.

Itse ongelma kuvataan luvussa 5. Ensin käsitellään yhden linjan tuotantoon liittyviä ongelmia, jonka jälkeen tutkitaan usean linjan linjastoa sekä joitakin piirilevyladontaan liittyviä rajoitteita. Tämän jälkeen selvitetään, miten huomioidaan tuotantoaikataulut ja massatuotantomahdollisuudet piirilevyladonnassa. Näin pyritään jalostamaan ongelmaa ja samalla kehittämään ratkaisumenetelmää kohti käyttökelpoista algoritmia.

Luvussa 6 etsitään ratkaisualgoritmia edellisessä luvussa esitetyn mallin mukaisesti. Tavoitteena on kehittää yksinkertainen heuristinen keino jakaa työt linjoille tavalla, joka pyrkii minimoimaan myöhästymiset. Lopulta kehitettyä algoritmia testataan käytännönläheisillä simulaatioilla sekä kuvataan saadut tulokset luvussa 7.

2 Katsaus tuotantotekniikoihin

Ladontakoneet ovat kehittyneet paljon viimeisen neljännesvuosisadan aikana. Niiden nopeus on kasvanut ja niiden käsittelemät komponentit ovat pienentyneet paljon. Lisäksi niiden tekniikka on monipuolistunut. Yksi kone saattaa esimerkiksi sisältää useita suuttimia erikokoisia komponentteja varten. Lisäksi ladontakone saattaa käyttää kahta ladontapäätä, jotka lähes yhtäaikaisesti latovat komponentteja samalle piirilevylle. Ladontakoneiden monipuolisuuden vuoksi käytettävien komponenttien koko saattaa vaihdella huomattavastikin. Tästä syystä myös komponenttikelojen koko, kuten myös niiden sisältämien komponenttien määrä, vaihtelee.

2.1 Käsitteet

Nykyisillä ladontakoneilla voidaan käsitellä sekä pintaliitoskomponentteja että levyn läpi juotettuja (radiaali-) komponentteja. Tässä työssä keskitymme tarkastelemaan pintaliitoskomponenttien ladontaa. Ladontaprosessissa komponentit sijoitetaan piirilevylle, jonka jälkeen levy kuumennetaan juotosuunissa. Kuumennettaessa piirilevyllä oleva juotostahna sulaa juottaen komponenttien kontaktipinnat (nastat) kiinni levyyn. Tämä juotoslankoja käyttävästä tekniikasta poikkeava tapa mahdollistaa monipuolisen kontaktipintojen sijoittelun. Ensinnäkin kontaktipinnat voidaan sijoittaa tiheästi. Monissa koteloissa kontaktit ovat sijoitettu siten, että käsin juottaminen on sangen vaikeaa. Toiseksi koko kotelon alapinta voidaan käyttää kontaktipintojen sijoittamiseen. Käsin juotettavaksi tällaiset kotelot ovat puolestaan jo mahdottomia.

Erilaisten pintaliitoskomponenttien kotelointi vaihtelee myös paljon. Kontaktipintojen sijainti ja määrä vaihtelee, kuten myös koteloiden mitat. Yhteistä on kuitenkin pintaliitoskomponenttien suorakulmainen muoto ja yläpinnan tasaisuus. Pienimmät ladontakoneissa käytetyt komponentit ovat toiminnaltaan yksinkertaisia, esim. vastuksia tai kondensaattoreita. Pienimpien komponenttien kotelon sivun pituus on vain alle puoli millimetriä, kun suurimpien komponenttien kohdalla puhutaan jo senttimetreistä.

Pintaliitoskomponentit pakataan usein muoviseen nauhaan, jossa on loke-

ro kullekin komponentille. Nauha lokeroineen on päällystetty ohuella muovisella suojakalvolla. Näin komponentit ovat tavallaan yksittäispakattuja. Nämä nauhat on kääritty syötinkeloihin. Kelojen avulla komponentit on nopeasti siirrettävissä ladontakoneen käyttöön. Yksi kela sisältää komponentteja sadoista jopa tuhansiin, komponentin koosta riippuen. Komponentin koko vaikuttaa myös nauhan leveyteen ja siten myös kelan leveyteen. Ladontakoneissa on tietty määrä kelapaikkoja. Leveämmät komponenttikelat vievät useamman paikan verrattuna kapeisiin keloihin. Lisäksi joissakin ladontakoneissa on mahdollista asettaa kaksi tai jopa kolme syötinkelaa samaan kelapaikkaan allekkain.

Komponentit siirretään kelojen nauhoista piirilevylle ladontapäitä käyttäen. Ladontapäät poimivat komponentit alipainetta käyttävien suuttimien avulla. Kuten aiemmin mainittiin, komponenttien koko (sivun pituus) vaihtelee alle millimetristä useisiin senttimetriin. Eri kokoiset komponentit vaativat eri kokoisen suuttimen. Pienimmät komponentit vaativat pienen imutehon ja pienen suuttimen, joka ei pysty imaisemaan komponenttia sisäänsä. Suurempien komponenttien käsittely onnistuu puolestaan paremmin käyttämällä isompaa imutehoa ja isompaa suuttinta, johon komponentti kiinnittyy varmemmin.

2.2 Ladontaprosessi ja sen valmistelut

Piirilevyjen tuotannossa useita ladontatöitä suoritetaan perättäisillä ja rinnakkaisilla ladontakoneilla. Työ koostuu erästä piirilevyjä, joihin on ladottava tietyt komponentit tarkalleen määrättyihin paikkoihin. Kukin työ voidaan suorittaa yhdellä tai useammalla ladontakoneella. Ladontakoneen komponenttikapasiteetti, ja sen soveltuvuus tietyn tyyppisten komponenttien ladontaan vaikuttavat siihen, voidaanko ladontatyö suorittaa kyseisellä koneella vai ei. Jos työtä ei voi suorittaa kyseisellä koneella, on kaksi vaihtoehtoa. Työtä voidaan jatkaa toisella koneella, jossa on vaadittavia komponentteja, tai koneeseen voidaan vaihtaa tarvittavat komponentit ja suuttimet, jotta työtä voidaan jatkaa.

Ladontakoneen komponenttien valinnassa tulee vastaan käsite *koneen aset-*

telu (engl. setup). Koneen asettelulla tarkoitetaan koneeseen valittujen komponenttityyppien joukkoa. Asettelu numero 1 voi esimerkiksi sisältää kymmenen erilaista vastusta (R_1 - R_{10}), viisi erilaista kondensaattoria (C_1 - C_5), viisi erilaista transistoria (T_1 - T_5) sekä kolme eri ic-piiriä (I_1 - I_3). Jos esimerkiksi työ A vaatii asettelu 1:n komponentit, voi työ B vaatia hieman tai jopa täysin erilaisen asettelun. Mikäli asettelut ovat riittävän samankaltaisia ja työt aiotaan tehdä samalla koneella, kannattaa koneiden asettelut mahdollisuuksien mukaan yhdistää. Tällöin molempien asettelujen yhdistelmä asetetaan koneeseen heti, jolloin työn vaihtaminen toiseen käy välittömästi ilman tuotantokatkosta. Jos kaksi tai useampi työ voidaan tehdä tietyllä asettelulla, kutsutaan töihin liittyviä piirilevyjä tuoteperheeksi. On kuitenkin huomattava, että ladontakoneissa on rajallinen määrä paikkoja komponenttikeloille ja ladontapäihin voidaan valita vain tietynlaiset suuttimet. Toisin sanoen on oletettavaa, että kaikkia eri asetteluja ei voida yhdistää. Lisäksi on huomattava, että asettelu käsittää ainoastaan komponenttien tyypit, ei komponenttien tarvittavia tai saatavilla olevia lukumääriä.

2.3 Ladontaprosessien hallinta

Prosessinhallinnan (ts. töidenjärjestelyn) kannalta kullekin työlle voidaan määrittää pysyviä ominaisuuksia, jotka eivät riipu töiden järjestelystä. Pinedo esittelee kirjassaan näitä ns. *staattisia* ominaisuuksia neljä kappaletta [Pin05].

Käytännössä yksi tärkeimmistä staattisista ominaisuuksista on työn suorittamiseen asetettu *takaraja* (engl. *due date* tai *deadline*). Mikäli työ valmistuu takarajan jälkeen, voidaan töiden järjestelyn ajatella ainakin jossain määrin epäonnistuneen, tai vaatimusten olleen liian tiukkoja.

Toinen aikataulutuksen kannalta tärkeä ominaisuus on *työn kesto*. Vaikka työn kestoon ei voida vaikuttaa töidenjärjestelyllä, on joskus mahdollista muokata työympäristöä siten, että työ saadaan tehtyä nopeammin. Ladontakoneiden maailmassa tämä saadaan aikaiseksi esim. optimoimalla ladontaprosessia.

Kolmas ominaisuus on *tärkeys* eli painoarvo. Tällä on esimerkiksi mah-

dollista varmistaa, että tärkeät työt myöhästyvät mahdollisimman vähän.

Neljäs ominaisuus on *saapumis-* tai *julkaisuaika*, joka määrittää ajan, jolloin työ on mahdollista aloittaa. Tämä siis asettaa rajoituksen työn alkamisajalle. Joissakin malleissa tätä ominaisuutta ei ole tarpeellista käyttää.

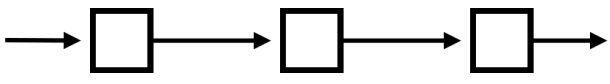
Pinedon mukaan [Pin05] tärkeimmät muuttuvat (eli *dynaamiset*) ominaisuudet ovat työn *aloitusaika* (engl. *start time*) ja työn *valmistumisaika* (engl. *completion time* tai *finish time*). Nämä ominaisuudet määräytyvät töidenjärjestelyn perusteella ja määrittävät lopullisesti töiden mahdollisten myöhästymisten määrän.

2.4 Tuotantotavat ja niiden vaikutukset töidenjärjestelyyn

Tuotannossa ladontakoneet on järjestetty tietyn konfiguraation mukaisesti. Tuotantoprosessi määrää suurelta osin koneiden sijoittelun tuotantoketjussa. Seuraavassa kuvataan Pinedon kirjassaan [Pin05] esittämät viisi linjaston konfiguraatioiden perustapausta. Käytetystä konfiguraatiosta riippuu, miten töidenjärjestelyongelmaa lähestytään.

2.4.1 Yhden linjan malli (Single line)

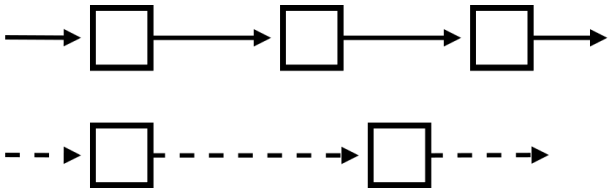
Yksinkertaisin ladontakoneiden konfiguraatio on yksittäinen linja, joka on esitetty kuvassa 1. Linja voi käsittää yhden tai useampia peräkkäisiä ladontakoneita ja työ siirtyy koneelta seuraavalle liukuhihnamaiseen tapaan. Yhden linjan mallissa ei luonnollisesti voida puhua linjojen tasapainotuksesta. Sen sijaan töiden järjestelyongelmat saattavat tulla kyseeseen, mikäli on tarkoitus tuottaa yhdellä koneella erilaisia tuotteita. On myös huomioitava, että tässä mallissa hitain kone määrää koko linjan tahdin ja samalla koko tuotannon tahdin.



Kuva 1: Yksittäislinja-malli

2.4.2 Rinnakkaismalli (Parallel lines)

Rinnakkaismallissa on useita linjoja rinnakkain ja kussakin linjassa yksi tai useampi kone peräkkäin. Tämä malli on esitetty kuvassa 2. Työ kulkee koneesta seuraavaan sitä linjaa pitkin, jolle työ on alun perin määrätty. Koska linjat ovat itsenäisiä, ne voivat edetä eri nopeuksilla. Tällöin nopeat ja hitaat työt tai koneet kannattaa eritellä eri linjoille odotusaikojen minimoimiseksi.



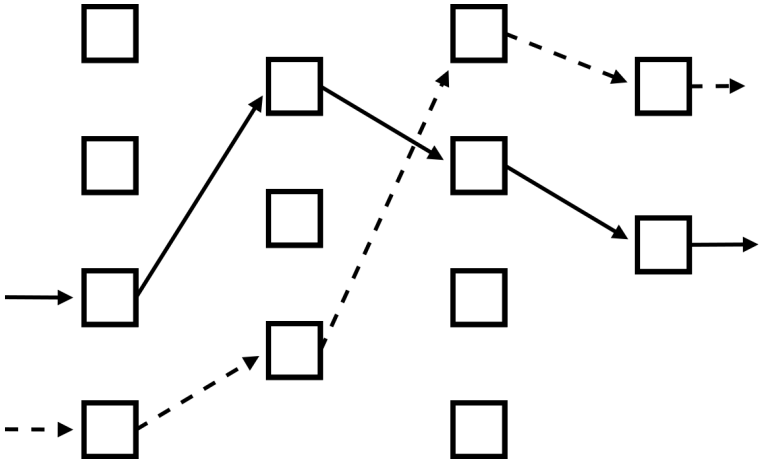
Kuva 2: Rinnakkaismalli

2.4.3 Sarjavalmistusmalli (Flow shop)

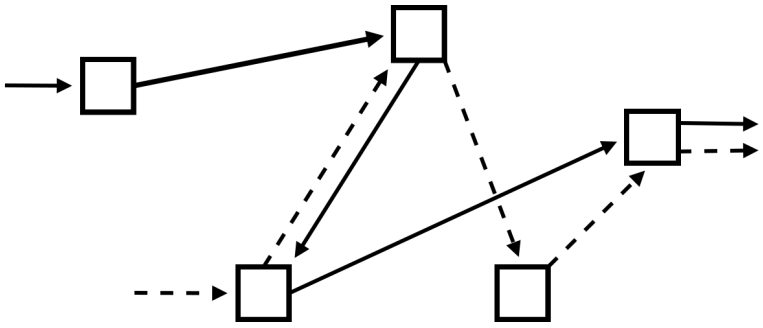
Flow shop -malli käsittää useita koneita, jotka on järjestetty tasoittain. Kullakin tasolla yksi tai useampi rinnakkaista konetta. Työ siirtyy tasolta toiselle eli jostakin tason a koneesta johonkin tason b koneelle ennalta määrättyjä reittejä pitkin. Jos siirtymisreitit eivät ole kiinnitettyjä tai jos työjärjestystä voidaan muuttaa dynaamisesti, puhutaan *flexible flow shopista*. Kuvassa 3 on esitetty flexible flow shopin malli.

2.4.4 Konepajamalli (Job shop)

Job-shop -mallissa eli työpajamallissa koneita ei ole järjestetty mitenkään. Mallissa ei ole ennalta määrättyjä tasoja tai tuotantolinjoja, kuten havaitaan kuvasta 4. Työ siirtyy koneelta toiselle sen mukaan, mitä työlle pitää tehdä ja mikä kyseiseen työhön kykenevä kone on vapaana. Tämä soveltuu töille, joiden osat voidaan tehdä (ainakin osittain) mielivaltaisessa järjestyksessä.



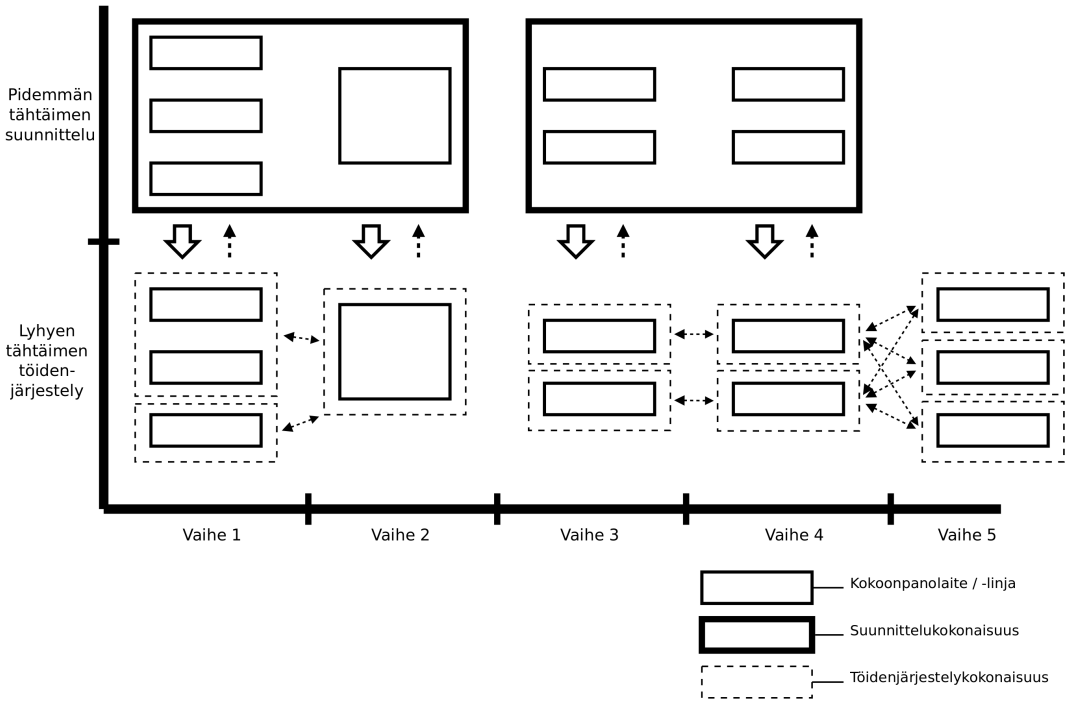
Kuva 3: Flex flow -malli [Pin05, s. 23]



Kuva 4: Job shop -malli [Pin05, s. 24]

2.4.5 Toimitusketjumalli (Supply chain)

Toimitusketju-mallilla tarkoitetaan isompaa tuotantoverkostoa, joka koostuu sarjavalmistus- tai konepajamallisista yksiköistä. Tämänkaltaisessa tuotantoverkostossa tuotannon suunnittelun, ja varsinkin aikataulujen suunnittelun, merkitys on erittäin tärkeä. Myös logistiikan eli komponenttien ja osien kuljetuksen valmistuspaikalta toiselle on toimittava hyvin. Kuten kuva 5 osoittaa, tuotannon suunnittelu ja aikataulutus voidaan erottaa toisistaan. Näin suunnittelu suoritetaan ylemmällä tasolla suurempia kokonaisuuksia ajatellen. Töiden aikataulutus puolestaan tehdään lähempänä tuotantolinjoja.



Kuva 5: Supply chain -malli [Pin05, s. 179]

2.5 Tuotannon ja sen järjestelyn tavoitteet

Piirilevytuotannossa, kuten monissa muissakin tuotantolajeissa, voi olla lukuisia erilaisia tavoitteita – aina laadusta tuotannon edullisuuteen. Mittareita, joilla tavoitteiden toteutumista mitataan, on vähintään yhtä monta. Seuraavassa kuvataan Pinedon [Pin05] mainitsemat yleisimmät tavoitteet. Tavoitteet liittyvät luonnollisesti nopeuteen ja tehokkuuteen. Nopeus on melko yksiselitteistä, mutta tehokkuus voi liittyä moneen seikkaan aina rahallisista asioista varastotilan käyttöön.

Osa tavoitteista liittyy suoraan linjastojen toimintaan. Tavoitteina voi olla mm. linjan tai linjaston suoritustehon maksimointi ja yhden tuotteen valmistamiseen käytettävän ajan minimointi. Nämä kaksi tavoitetta ovat läheisesti sidoksissa toisiinsa. Näillä tavoitteilla pyritään varmistamaan pienehköjen erien valmistuminen mahdollisimman nopeasti mm. käyttämällä useampaa valmistuslinjaa, mikäli mahdollista. Toisaalta tavoitteet voivat liit-

työ aikarajoihin. Tavoitteena tällöin on myöhästymisten minimointi eli pyritään suorittamaan kaikki työt mahdollisimman hyvin annettujen aikataulujen puitteissa. Eräs tavoite on minimoida koneiden asettelujen kustannukset. Luonnollisesti koneiden asettelujen kustannuksiin lasketaan aika, jonka asettelun vaihtaminen toiseen vaatii. Näihin kustannuksiin on laskettava myös mahdolliset materiaalikustannukset, joiden minimointi on myös oleellista.

Osa tuotantoprosessin tehostamista ovat varastointikustannusten minimointi. Näihin tavoitteisiin kuuluvat komponenttien varastointikustannusten lisäksi sekä keskeneräisten että valmiiden tuotteiden varastointikustannusten minimointi. Valmiiden tuotteiden pitäisi lähteä kokoonpanolinjalta mahdollisimman suoraan ja nopeasti kohti asiakasta. Lisäksi tuotteiden valmistusaika tulisi minimoida, jotta vältytään välivarastoinnilta. Samoin ylimääräisten komponenttien varastointia tulisi välttää. Komponenttien ja tuotteiden varastoinnin lisäksi kustannuksia kasvattavat niiden kuljetukset. Jos tuotanto on hajautettu useaan paikkaan, myös kuljetuskustannukset on syytä minimoida hyvällä ja järkevällä logistiikalla.

Tämän tutkielman aiheena on itse piirilevyjen ladonta ja siihen välittömästi liittyvät ongelmat. Näin ollen tutkielmassa keskitytään linjastojen suoritustehon maksimointiin. Tämä pitää sisällään koneiden asettelukustannusten minimoinnin, tuotteiden valmistusajan minimoinnin ja linjastojen käytön tasapainottamisen.

3 Tuotantolinjojen tasapainoittaminen

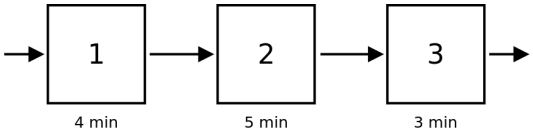
Tuotantolinjojen tasapainotuksessa (engl. *line balancing*) on kysymys työmäärän jakamisesta tasaisesti linjojen kesken ottaen huomioon ennalta määrätyt rajoitukset ja vaatimukset. Kuten edellä mainittiin, rajoituksia ja vaatimuksia ovat mm. töiden takarajat ja painoarvot. Näiden töihin ja tavoitteisiin liittyvien vaatimusten ja rajoitusten lisäksi itse tuotantolinjaston ominasuudet ovat ratkaisevassa asemassa. Linjaston konfiguraatio ja siihen kuuluvien koneiden ominaisuudet vaikuttavat ongelmaan. Näiden lisäksi koko tehtäväjoukon suorittamiselle asetaan jokin vaatimus, joka on tarkoitus optimoida. Tällainen vaatimus voi olla esim. ladontakoneiden asettelujen vaihtojen määrän minimointi tai yksinkertaisesti töiden myöhästymisten minimointi.

3.1 Käsitteet

Ensiksi on syytä selvittää termit, jotka liittyvät linjastoihin ja niiden tehokkuuden mittaamiseen. Seuraavassa käydään yleistä teoriaa läpi unohtamatta itse piirilevyladontaan liittyvää aihetta. Tästä syystä termejä kone (viitaten ladontakoneeseen) ja työpiste käytetään osittain rinnakkain.

Läpivirtausaika (engl. *flow time*) on kaikkien linjastojen koneiden tai työpisteiden yhteenlaskettu suoritusaika. Tämä vastaa yhden tuotteen valmistumisaikaa, mikäli kyseessä on yksinkertainen linja, joka ei esimerkiksi haaraudu. *Tahtiaika* (engl. *cycle time*) on suurin työaika, joka kuluu jollakin linjaston työpisteillä. Yksinkertaisen linjan tapauksessa tahtiaika ilmoittaa myös, kuinka usein uusi tuote valmistuu linjasta. Jos työpisteiden keskinäiset työajat poikkeavat paljon toisistaan, on seurauksena, että joidenkin koneiden tehollista käyttöaikaa käytetään odotteluun. *Edeltäjädiagrammi tai -graafi* (engl. *precedence graph/diagram*) puolestaan määrää järjestyksen, jossa työt tulee tehdä ja jossa töiden tulee siirtyä linjastolla.

Kuvassa 6 on yksinkertainen linja, jonka avulla on helppo antaa konkreettiset esimerkit edellä esitellyistä termeistä. Kuva itsessään on edeltäjägraafi, sillä se kertoo työjärjestyksen: ensin *työ 1*, sitten *työ 2* ja viimeisenä *työ 3*. Läpivirtausaika on kaikkien työpisteiden yhteenlaskettu työaika eli 12 minuuttia. Se on siis aika, joka käytetään linjastossa yhden tuotteen valmista-



Kuva 6: Yksinkertainen linja ja sen työpisteiden työajat

miseen. Tahtiaika puolestaan on suurin linjan työaika, joka on 5 minuuttia. Toisin sanoen joka 5. minuutti uusi tuote on valmiina linjan päätepisteessä.

Koneaika (engl. *station time*) on yhteenlaskettu aika, mikä joltakin tietyltä koneelta kuluu kaikkien sille määrättyjen tehtävien suorittamiseen. Koneaika koneelle k lasketaan $t(S_k) = \sum_{j \in S_k} t_j$, missä S_k on koneelle k määrättyjen töiden joukko. On huomattava, että tämä aika ei pidä sisällään mahdollisia taukoja, vaan sisältää ainoastaan itse työhön kuluneen ajan.

Tehokkuus (engl. *efficiency*) kertoo, kuinka hyvin linjan koneet ovat hyötykäytössä. Becker ja Scholl käsittelevät [BS06] yleisiä kokoonpanolinjojen ongelmia. Edellisten määritelmien lisäksi he antavat seuraavan kaavan tehokkuudelle:

$$E = \frac{T}{M \cdot C} \tag{1}$$

Kaavassa T tarkoittaa linjan työpisteiden suoritusaikojen summaa ($\sum T_i$). M tarkoittaa työpisteiden (tai koneiden) määrää ja C linjan tahtiaikaa. Kuvan 6 tehokkuudeksi saadaan siten kaavaa (1) käyttämällä:

$$E = \frac{4 + 5 + 3}{3 \cdot 5} = \frac{12}{15} = 0,8$$

Boysen, Fliedner ja Scholl [BFS06] mainitsevat, että kaavasta (1) on laskettavissa myös linjan työpisteiden yhteenlaskettu joutoaika:

$$T_{idle} = M \cdot C - T \tag{2}$$

Kuvan 6 linjan joutoajaksi saadaan tällöin:

$$T_{idle} = 3 \cdot 5 - 12 = 3$$

3.2 Yleiset ongelmatapaukset

Boysen, Fliedner ja Scholl [BFS06] käsittelevät tasapainotusongelmia ja ennen kaikkea niiden luokittelemista. Linjan tasapainotuksesta käytetään kirjallisuudessa lyhennettä *ALB* (*assembly line balancing*). Kun viitataan yksinkertaistettuun ongelma-asetteluun, käytetään lyhennettä *SALB* (*simple ALB*). Yksinkertaistetut tasapainotusongelman tapaukset ovat heidän mukaansa karkeasti jaettavissa neljään luokkaan niiden tavoitteiden mukaan:

- Käytettävien työpisteiden joutoaikaa yritetään minimoida tahtiajan funktiona (sama kuin työpisteiden määrän minimointi). Tämänkaltaisesta tasapainotusongelmasta käytetään merkintää *SALBP-1*.
- Tahtiaika minimoidaan käytössä olevien työpisteiden määrän funktiona. Tällaisia ongelmia merkitään lyhenteellä *SALBP-2*.
- Yritetään maksimoida linjan tehokkuutta (engl. *efficiency*), kun käytössä olevien työpisteiden määrä ja tahtiaika on tiedossa. Tehokkuutta maksimoivista ongelmista käytetään lyhennettä *SALBP-E*.
- Ongelmista, joissa yritetään saavuttaa järkevää tasapainoa (engl *feasible balance*) työpisteiden lukumäärän ja tahtiajan välillä, käytetään lyhennettä *SALBP-F*.

Edellämainitut yleiset tasapainotusongelmien tyypit perustuvat seuraaviin Boysenin, Fliednerin ja Schollin esittämään yhdeksään rajoittavaan oletamukseen.

1. Kyseessä on yhden tuotteen massatuotanto.
2. Työtehtäviin liittyvät valmistusprosessit ovat muuttumattomia. Toisin sanoen jokainen työtehtävä suoritetaan aina samalla ennaltamäärätyllä tavalla.
3. Koko linja on tahdistettu johonkin tiettyyn muuttumattomaan työtahtiin.

4. Linjan on oltava sarjamuotoinen ja suljettu. Tämä tarkoittaa, että linjassa ei ole rinnakkaisia koneita eikä linjaan voi syöttää mitään (esim. puolivalmiita tuotteita) linjan ensimmäisen työpisteen jälkeen.

5. Ennalta määrättyä töiden suoritusjärjestystä (järjestysdiagrammia) noudatetaan.

6. Kaikkien työaikojen oletetaan olevan ennalta määrättyjä vakioaikoja. Aikojen voidaan vaatia myös olevan kokonaislukuja optimoinnin helpottamiseksi.

7. Edeltäjägraafin oletetaan olevan ainoa rajoitus töiden määräämisessä koneille.

8. Oletetaan, että työt ovat jakamattomia eli atomisia. Mitään työtä ei voida jakaa useammalle työpisteelle samanaikaisesti suoritettavaksi.

9. Työpisteet ovat identtisiä. Ne pystyvät suoritumaan töistä esim. ilman ylimääräisiä taukoja yhtä hyvin kuin muutkin koneet.

SALB-ongelmien versioita on tutkittu laajalti. Edellä käsiteltiin vain ongelmien luokittelua, mutta näille ongelmille on kehitetty myös menetelmiä, jotka tuottaisivat tarkkoja ratkaisuja [SB06].

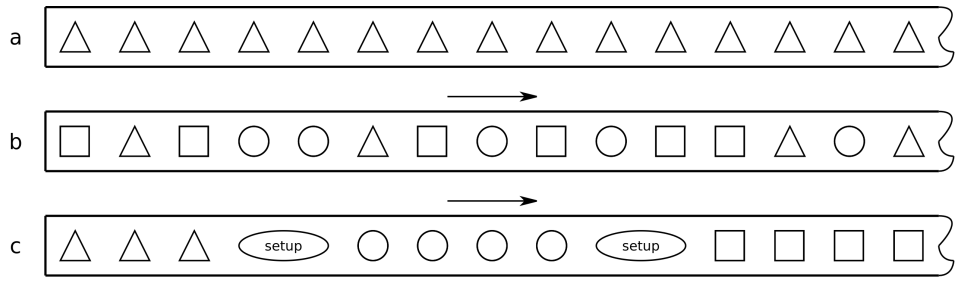
3.3 Muita ongelmatyyppejä

Edellä mainittu yksinkertaisten linjojen tasapainotusongelmien ryhmä on hyvin rajoittunut. Käytännössä linjastoille asetetaan usein vaatimuksia, joita ei voida toteuttaa ilman, että edellämainituista olettamuksista joustetaan tai luovutaan kokonaan. Boysen, Fliedner ja Scholl luokittelevat [BFS06] linjan tasapainoitusongelmat kahdeksaan luokkaan. Seuraavassa käydään piirilevyladonnan näkökulmasta läpi miten tosielämän olettamukset ja vaatimukset vaikuttavat näihin tapauksiin sekä tuotannon suunnitteluun.

3.3.1 Monta erilaista tuotetta

Piirilevyladonnassa tuotetaan harvoin vain yhtä ja samaa tuotetta tietyllä linjalla pitkiä aikoja. Harvassa elektroniikkatehtaassa voidaan pitää yhtäkään tuotantolinjaa varattuna vain tietylle massatuotteelle, vaan linjojen nopea mukautus ja uudelleenkonfigurointi mahdollistavat tehokkaan tuotannon.

Kun samalla linjalla on tarkoitus koota useita erilaisia tuotteita, on ajateltava tuotteiden samankaltaisuutta. Pieniä eroja sisältävien tuotteiden voidaan ajatella kuuluvan samaan tuoteperheeseen. Tietyn tuoteperheen tuotteet voidaan valmistaa samalla komponenttiasettelulla. Näin voidaan ilman linjaan tehtäviä muutoksia valmistaa monia tuotteita, jotka poikkeavat toisistaan vain vähän. Tällaista linjaa kutsutaan *sekatuotelinjaksi* (engl. *mixed-model line*) (kuva 7b).



Kuva 7: Tuotantolinjatyyppejä: yhden tuotteen linja (a), sekatuotelinja (b) ja monen tuotteen erätuotantolinja (c) [BS06]

Kun tuotteet eroavat toisistaan riittävästi, niiden valmistaminen suoritetaan ns. *tuote-erinä* (engl. *batch* tai *lot*), joiden välissä vaihdetaan koneiden asettelut seuraavaa tuotetta vastaavaksi. Eräajoja suorittavasta linjasta käytetään myös nimitystä *monen tuotteen tuotantolinja* (engl. *multi-model line*) ks. kuva 7c. Tämänkaltaisessa tuotannossa tiettyä tuotetta valmistetaan tarvittava määrä, minkä jälkeen suoritetaan valmistelut seuraavan tuotteen valmistusta varten. Elektroniikkakomponenttien ladontakoneissa tätä vastaa komponenttien *asettelu* (engl. *setup*) ladontakoneen *syöttimiin* (engl. *feeder*). Monen tuotteen linjan vastakohtana on luonnollisesti *yhden tuotteen linja* (engl. *single-model line*) (kuva 7a), joka on siis omistettu vain yhden tietyn tuotteen valmistamiseen.

Joka tapauksessa useamman eri tyyppisen tuotteen valmistaminen vaatii koneiden komponenttiasettelujen vaihtoa ja näin ollen siitä aiheutuvien kustannusten huomioimista. Tämä muuttaa linjojen tasapainotusongelmaa huomattavasti vaikeammaksi.

3.3.2 Linjan tai linjaston hallinta

Tuotantolinjat voidaan jakaa kahteen luokkaan: *tahdistetut linjat* ja *tahdistamattomat linjat*. Tahdistetuissa linjoissa työ siirtyy kultakin koneelta seuraavalle ennalta määrätyn vakioajan välein. Tahdistamattomissa linjoissa työ siirtyy vasta, kun tarvittavat toiminnot on suoritettu ko. koneilla. Tämä saattaa aiheuttaa odottamattomia odotusaikoja, mikäli seuraava tai edeltävä kone ei ole vielä valmis, kun työn pitäisi siirtyä. Tahdistetujen linjojen tapauksessa odotusajat kuuluvat työsykliin ja ovat näin ennakoitavissa tai niitä ei tarvitse ottaa huomioon.

Boysen, Fliedner ja Scholl [BFS06] jakavat tahdistamattomat linjat edelleen synkronoituihin ja synkronoimattomiin linjoihin. Synkronoiduissa linjoissa työ siirtyy seuraavalle koneelle, kun kaikki kesken olevat työt on suoritettu. Näin siirrot tapahtuvat yhtäaikaisesti. Tahdistamattomissa linjoissa työ siirretään sen valmistuttua, jos seuraava kone on vapaa. Mikäli seuraavan koneen nykyinen työ on vielä kesken, joudutaan odottamaan. Odotusta voi myös tapahtua, kun koneen oma työ on valmis, mutta edeltävän koneen työ on vielä kesken. Edeltävän työn odottamisesta käytetään termiä *nälkiintyminen* (engl. *starving*). Odotuksiin voidaan vaikuttaa välivarastoinnilla, jossa työpisteiden välissä on muutaman työn kokoiset välivarastot tasoittamassa tuotannon kulkua.

3.3.3 Tehtäväaikojen muuttuvuus

Vaikka normaalisti suoritettavien tehtävien aikoja edellä kuvattiin vakioiksi, todellisuudessa ajoissa voi olla pientä vaihtelua. Tavallisten ja hyvin sujuvien ladontaoperaatioiden kohdalla nämä muutokset ovat lähes olemattomia eivätkä siten käytännössä vaikuta töiden järjestelyyn.

On kuitenkin mahdollista, että tehtäväajat vaihtelevat. Boysen, Fliedner

ja Scholl [BFS06] esittävät monia syitä tähän vaihteluun, joista kaikki eivät kuitenkaan vaikuta koneistettun ja automatisoidun ladontakoneympäristön tapauksessa. Ladontakoneiden osalta maininnan arvoisia muutoksia ovat oppiminen, työpistekohtaiset materiaaliviivet ja työkalujen vaihtoihin liittyvät viiveet.

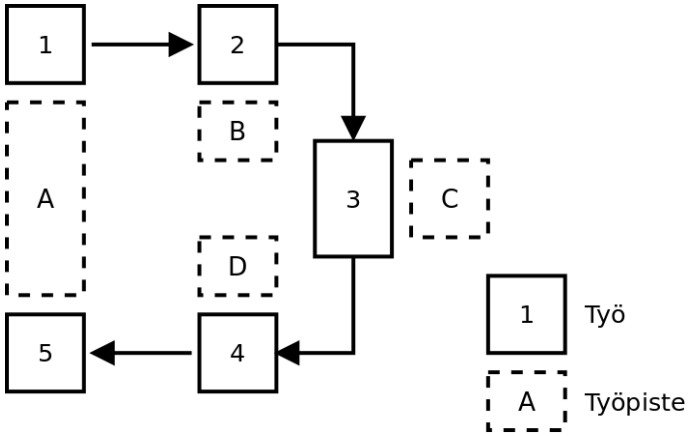
Oppimisella tarkoitetaan jonkin työn osa-alueen tehostamista eli toiminnan päivittämistä. Jonkin koneen ladontatehtävälle voidaan tällöin löytää nopeampi ratkaisu, joka voidaan ottaa käyttöön ehkä jopa kesken ladontaprosessin. Vaikka ladontaprosessin päivitys voi viedä hetken, itse tehtävään kulutettua työaikaa saadaan pienennettyä.

Materiaaliviiveille ja työkalujen vaihtoviiveille löytyy myös vastineet piirilevyladonnasta. Materiaaliviiveet liittyvät valmistusmateriaalin saatavuuteen. Ladontakoneympäristössä tällä tarkoitetaan tyhjän komponenttikelan vaihtoa uuteen. Tämä luonnollisesti aiheuttaa huomattavan viiveen. Eri kokoisia komponentteja varten puolestaan voidaan tarvita eri kokoisia suuttimia, joilla komponentteja siirretään komponenttikeloista piirilevylle. Suuttimen vaihto toiseen vie myös aikaa ja on tulkittavissa edellä mainittuna työkalun vaihtoviiveenä.

3.3.4 Tuotantolinjan ja linjaston muoto

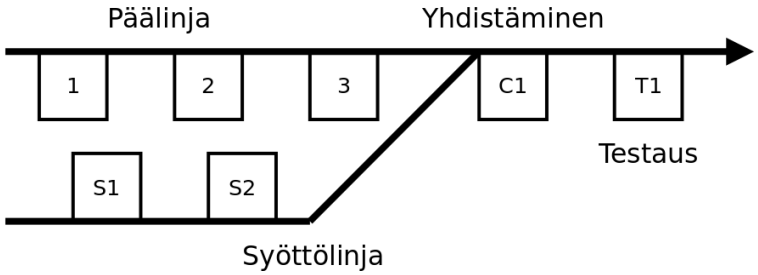
Tuotantolinjan ei tarvitse olla perinteisen suoraviivaisen liukuhihnan kaltainen. Boysen, Fliedner ja Scholl [BFS06] esittelevät kaksi vaihtoehtoa perinteiselle yksittäisen linjan mallille. Ensimmäinen vaihtoehto on ns. U-malli (kuva 8), jossa tuotteet palaavat samoille työpisteille tuotannon myöhemmässä vaiheessa. Piirilevytuotannossa tämänkaltaista linjamallia voidaan käyttää esim. 2-puolisten levyjen ladonnassa. Myös käsin työskentelyssä U-malli on käyttökelpoinen. Voisi esimerkiksi olla mahdollista, että sama työntekijä suorittaa jonkin pienen alkuvalmistelun ennen ladonnan alkua ja jonkin pienen viimeistelytoimenpiteen ladonnan jälkeen samalla työpisteellä. Tämä on mahdollista, jos työajat ovat tarpeeksi lyhyitä, jotta ne voidaan molemmat suorittaa tahtiajan puitteessa.

Toinen erikoisempi linjamalli on syöttölinjojen käyttö (kuva 9). Tässä



Kuva 8: U-mallin mukainen linja, jossa työpisteestä (A) tehdään useampi työ (1 ja 5).

mallissa toisessa linjassa tehdyt tuotteet tai sen osat ohjataan päälinjalle yhdistettäväksi varsinaiseen tuotteeseen. Tämän mallin soveltuvuus automatisoituun ladontakoneympäristöön on myös kyseenalainen. Sen sijaan esimerkiksi linjan loppuosassa käsin tehtävä lisäkortin asentaminen varsinaiselle piirilevylle perustelisi tämän linjamallin käytön. Tällöin lisäkortti ladottaisiin omalla linjallaan ja syötettäisiin toiselle linjalle kokoonpanoa varten.



Kuva 9: Esimerkki syöttölinjan käytöstä, jossa erillinen osa valmistetaan syöttölinjalla ja yhdistetään päälinjalla tehtyyn tuotteeseen.

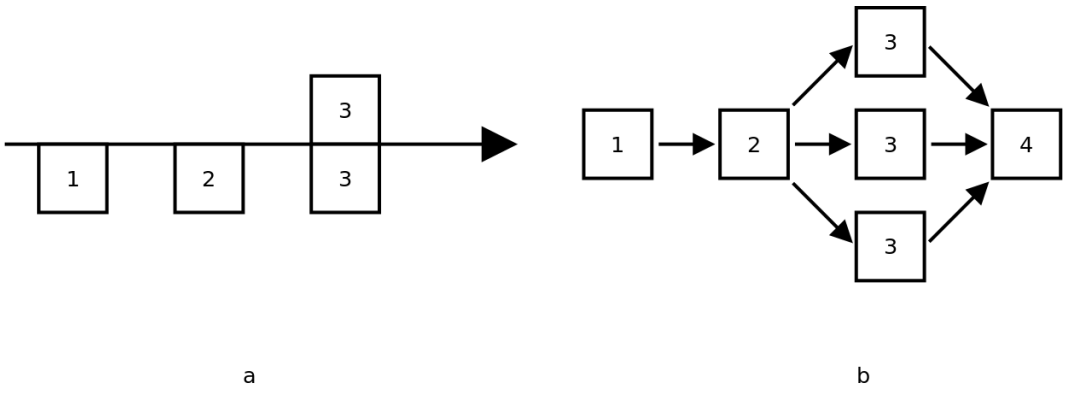
3.3.5 Tuotantolinjojen rinnakkaisuus

Luvussa 2.4.2 käsiteltiin Pinedon rinnakkaisten tuotantolinjojen mallia, jota myös Boysen, Fliedner ja Scholl [BFS06] esittävät linjojen tasapainotuksen ja

tuotannon tehostamiseksi. Kun valmistettavia tuotteita on useita, yksittäisen linjan aikataulutuksessa joudutaan tekemään kompromisseja. Jos linjasta siirretään muutama työpiste omaksi linjakseen, voidaan esimerkiksi yhden tuotteen valmistus siirtää kokonaisuudessaan omalle linjalleen.

Rinnakkaisuus ei välttämättä tarkoita useaa rinnakkaista linjaa. On nimittäin mahdollista asentaa myös rinnakkaiset työpisteet yhdelle linjalle (kuva 10a). Tämä edellyttää, että kyseisiä tehtäviä voi suorittaa rinnakkaisesti useampi tekijä. Vastaava tapaus esiintyy myös nykyaikaisissa ladontakoneissa, joissa on esimerkiksi kaksi ladontapäätä, jotka työskentelevät samanaikaisesti saman ladontatyön parissa [CWR+07].

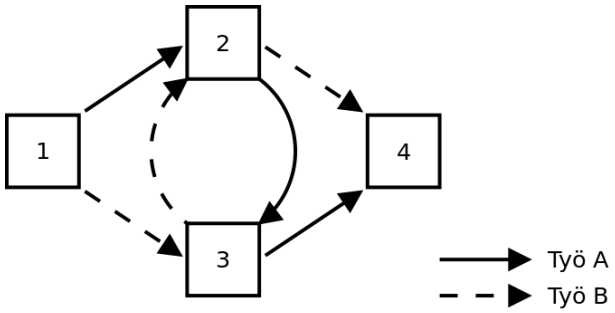
Boysen, Fliedner ja Scholl [BFS06] käsittelevät myös ns. työasemien rinnakkaistamista. Tämä eroaa edellisestä siten, että tässä tuotantolinja näyttää hetkellisesti jakautuvan kahdeksi tai useammaksi linjaksi. Tällöin kukin rinnakkaisista työasemista saa vuorollaan tuotteen käsiteltäväksi edelliseltä työpisteeltä (kuva 10b). Ladontakoneiden tapauksessa tämä vastaa sitä, että jokin kone olisi kahdennettu. Jos jokin ladontatehtävä vaatisi esim. kaksinkertaisen ajan verrattuna linjan tahtiaikaan, kannattaa kyseisen tehtävän ladontakoneen rinnalle tuoda toinen kone, jolla on identtinen konfiguraatio.



Kuva 10: Rinnakkaiset työpisteet (a) ja työasemat (b)

Boysenin, Fliednerin ja Schollin [BFS06] mukaan myös tehtävät olisivat rinnakkaistettavissa. Käytännössä tällä tarkoitetaan linjan edeltäjägraafin muuttamista. Tehtävien rinnakkaistaminen mahdollistetaan tekemällä graa-

fiin sykli, joka mahdollistaa tuotteen siirtämisen niin linjassa taaksepäin kuin myös työpisteen ohi. Ideana on valinnanmahdollisuus kahden tai useamman työn työjärjestyksessä. Tämä on esitetty kuvassa 11, jossa työn A kulku on merkitty yhtenäisellä ja työn B kulku katkonaisella viivalla.



Kuva 11: Työn rinnakkaistaminen edeltäjägraafin syklin avulla

3.3.6 Varustelu- ja prosessointivaihtoehdot

Boysenin, Fliednerin ja Schollin [BFS06] esittämät varustelu- ja prosessointivahtoehdot eivät yleensä ole oleellisia piirilevyladonnassa. Näillä tarkoitetaan toimenpiteitä, joita työpisteen puutteellinen varustelu (sekä osat että työkalut) tai työskentelyn muuttuminen aiheuttavat. Piirilevyladonnassa komponenttiasettelujen ja komponenttien tilapäinen loppuminen ovat kuitenkin rinnastettavissa puutteelliseen varusteluun.

3.3.7 Rajoitukset työn jaossa

Aiemmin todettiin edeltäjägraafin olevan aina rajoittamassa töiden jakamisesta työpisteille. Se osoittaa työtehtävien järjestyksen, jota tuotteen valmistamisessa tulee noudattaa. Tämän lisäksi Boysen, Fieldner ja Scholl [BFS06] esittelevät muita mahdollisia rajoituksia töiden jakamiselle.

Ensinnäkin työpisteet on voitu jakaa alueisiin. Jaon peruste voi olla niin resurssien saatavuus kuin ympäristötekijät, kuten lämpötila tai kosteus. Piirilevyladonnassa komponenttien varastointi saattaa olla eräs alueellinen tekijä, koska lisäkomponenttien saatavuuden pitää olla hyvä.

Toisena rajoituksena saattaa olla erikseen määrättyjen raja-arvojen ylittyminen. Esimerkiksi välivaraston koko saattaa rajoittaa uuden työn vastaanottamista. Uutta työtä ei voida poistaa työpisteestä, vaikka se olisi valmis, mikäli välivarastossa ei ole tilaa.

Kolmas rajoitus voi olla työtehtävän asettamat erikoisvaatimukset. Näitä voivat olla esimerkiksi käytössä olevat työkalut tai itse työpisteen ominaisuudet. Piirilevyladonnassa tämän tyyppisiä rajoituksia ovat esimerkiksi ladontakoneen ominaisuudet kuten ohjausohjelmiston versio ja koneen suutinvalikoima.

Neljäntenä mainitaan etäisyyksien aiheuttamat rajoitukset. Artikkelissa [BFS06] viitataan tapauksiin, joissa tuotteen lämpötila, kosteus tai jokin muu ominaisuus muuttuisi liikaa, kun tuotteen siirtäminen työpisteiden välillä kestää liian kauan. Näitä rajoituksia on piirilevyladonnassa yleensä vain vähän.

3.3.8 Tavoitteiden erilaisuus

Kuten edellä luvussa 2.5 mainittiin, tuotannossa ja sen järjestelyssä on monia eri tavoitteita, joiden toteutumista halutaan seurata. Kaikki tavoitteet liittyvät poikkeuksetta kustannusten minimointiin ja voiton maksimointiin. Se, miten tähän pyritään, on toinen asia. Monissa tehtaissa sisäisiä prosesseja kehitetään jatkuvasti. Tällöin saatetaan keskittyä esimerkiksi logistiikan toimivuuteen tai linjojen parhaimpaan mahdolliseen toimivuuteen. Vaikka ladontalinja sinänsä toimisi aivan hyvin, voidaan edellämainituilla tavoilla yrittää esimerkiksi vähentää koneasettelujen vaihtomääriä.

3.4 Linjojen tasapainoituksesta

Edellä mainittujen tavoitteiden lisäksi Boysen, Fieldner ja Scholl [BFS06] esittelevät idean *tasoitetuista työajoista* (engl. *smoothed station times*). Heidän mukaan kirjallisuudessa erotetaan *horisontaalinen ja vertikaalinen tasapainotus* (engl. *horizontal/vertical balancing*). Molemmat tasapainotusmenetelmät pyrkivät varmistamaan, että tuotteiden valmistus ei ylitä linjan tahtiaikaa.

Horisontaalisen tasapainotuksen ongelmia ovat käsitelleet Merengo, Nava ja Pozzetti [MNP99]. Lyhyesti sanottuna horisontaalinen tasapainotus tarkoittaa työpisteille määrättyjen työmäärien keskinäistä tasapainotusta. Toisin sanoen työpisteille jaettavien työkokonaisuuksien tulisi olla kestoltaan mahdollisimman samanlaisia. Vertikaalinen tasapainotus käsittää puolestaan kunkin linjan koneiden keskimääräisen työmäärän tasapainotuksen.

Kuvassa 12 on Merengon, Navan ja Pozzettin esimerkki tilanteesta, jossa liian suuri vaihtelu tehtäväajoissa saattaa aiheuttaa ongelmia. Kuvassa on esitetty yhden koneen ajankäyttö. On tarkoitus suorittaa vuorotellen 3 kappaletta työtä A ja 3 kappaletta työtä B. Kuvassa a työn A kesto on 1 minuutti, työn B kesto on 3 minuuttia. Yhtenäinen viiva kuvastaa itse työsuoritusta, katkoviiva työn vaihtamista toiseen ja pisteviiva käyttämätöntä aikaa. Pystyviivat kuvastavat työtahdin alkua ja loppua. Ensimmäiset minuutin mittaiset työt saadaan tehtyä hyvin. Myös ensimmäiset pidemmät työt saadaan tehtyä, mutta huomataan, että työn aloitus viivästyy aina minuutilla tahdin alkuun nähden. Tämä aiheuttaa tarpeeksi viivästystä, jotta kolmas työ B ei ehdi valmistua työtahdin loppuun mennessä.

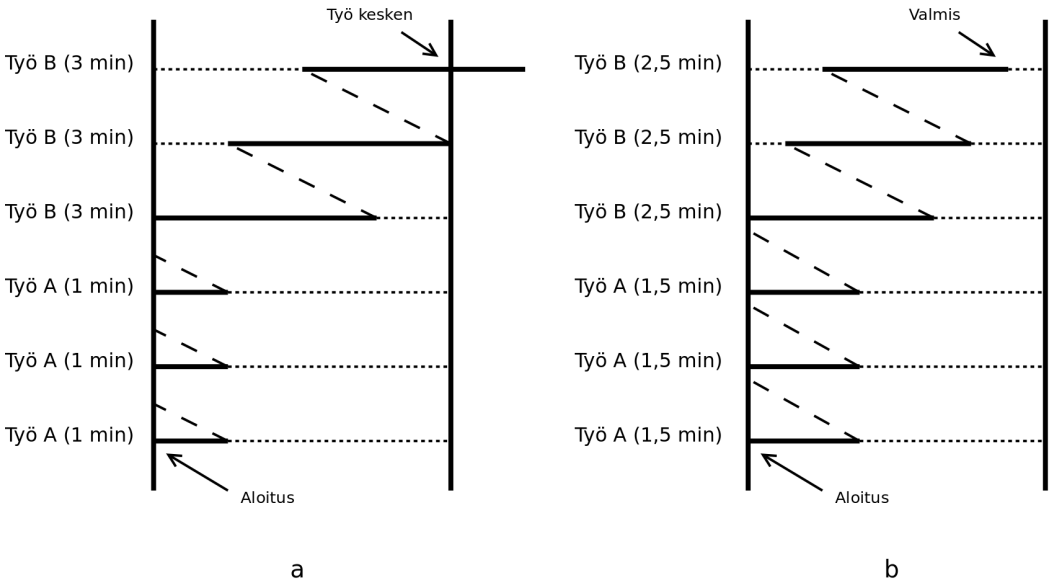
Kuvassa b kestoja on tasattu: A:n kesto on 1,5 minuuttia ja B:n kesto 2,5 minuuttia. Näin työ A ei aiheuta edelleenkään viivästystä eikä työ B aiheuta liikaa viivästystä ja viimeinenkin työ B saadaan valmiiksi ajoissa. Kuvassa a työjärjestys johtaa keskeneräiseen tuotteeseen, kun kuudetta tuotetta valmistetaan, vaikka kuvassa b kyseinen työjärjestys ei aiheuta ongelmia. On huomattava, että tämän kaltainen työajan muutos ei ole aina mahdollista.

Merengo, Nava ja Pozzetti [MNP99] antavat kaksi rajoitetta, joilla tutkia töihin käytettyä aikaa tahtiaikaan nähden. Ensinnäkin voidaan vaatia, että millään työpisteellä k ei minkään työn j aika ylitä tahtiaikaa C.

$$T_{jk} \leq C \quad \forall j, \forall k$$

Toinen rajoite tutkii keskimääräistä työaikaa:

$$\frac{1}{m} \sum_{j=1}^{m} T_{jk} \leq C \quad \forall k$$



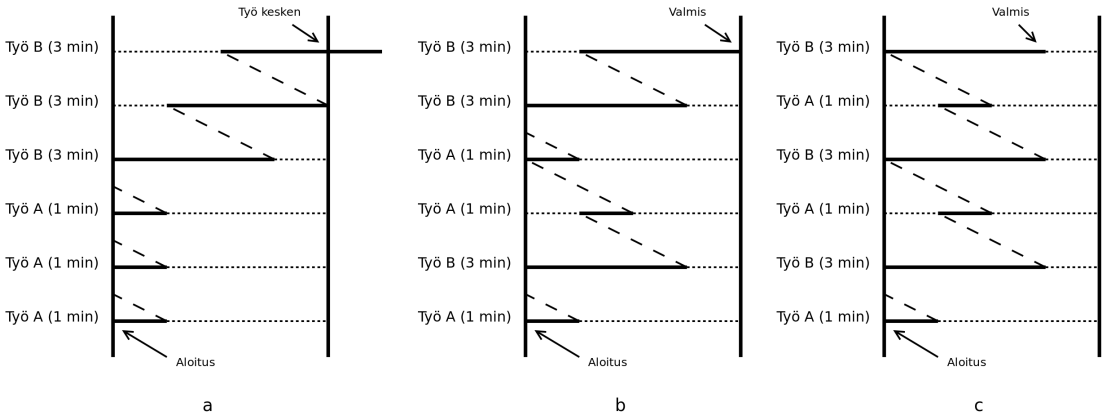
Kuva 12: Esimerkki horisontaalisen tasapainotuksen vaikutuksesta. Yhtenäinen viiva kuvaa työn suoritusta, katkoviiva siirtymistä töiden välillä ja pisteviiva käyttämätöntä aikaa. [MNP99]

On huomattava, että jälkimmäinen rajoite sallii (ainakin yksinään käytettynä) tahtiaikaa pidemmät työajat. Tämä on mahdollista vain, jos työpiste mahdollistaa pitkäkestoisen työskentelyn, jolloin tahtiajan ylitykset voidaan sallia yksittäistapauksina.

Vertikaalisessa tasapainotuksessa on kyse yhden linjan koneiden (tai työpisteiden) työmäärien tasapainottamisesta. Tarkoituksena on varmistaa, että töiden kestot ovat tarpeeksi vaihtelevia, jotta lyhyen ajan keskimääräinen tehtäväaika pysyisi mahdollisimman pienenä ja silti samansuuruisena. Tämä sallii paremmat mahdollisuudet selvitä ns. ruuhkahuipuista.

Kuva 13 noudattaa samaa esitystapaa kuin kuva 12. Kuvassa on tehtävänä sekä työtä A että työtä B, kutakin 3 kappaletta. Kohtien a, b ja c erona on eri työjärjestys.

Kohdassa a työjärjestys on A-A-A-B-B-B, mikä edellisen esimerkin tavoin johtaa keskeneräiseen työhön. Alussa keskimääräinen tehtäväaika on pieni, mutta lopussa suuri. Kahden peräkkäisen työn aikojen keskiarvo vaihtelee minuutista kolmeen minuuttiin. Kohdassa b töiden järjestystä on hieman se-



Kuva 13: Oikea työjärjestys mahdollistaa paremmin erikestoisten töiden suorittamisen [MNP99]

koitettu, jolloin lyhyen ajan keskimääräinen tehtäväaika ei vaihtele kuten a-kohdassa. Kohdassa c lyhytaikainen työn kestojen keskiarvo pysyy pienimpänä ja antaa siten eniten aikaa mahdollisille ennalta arvaamattomille tapauksille. Tässä kahden peräkkäisen työn ajan keskiarvo on muuttumaton 2 minuuttia.

Becker ja Scholl [BS06] esittelevät erään menetelmän vertikaaliselle tasapainotukselle. Ideana on minimoida ns. *tasaisuus-indeksi* (engl. *smoothness index*), jolle he antavat seuraavan kaavan:

$$SX = \sqrt{\sum_{k=1}^{M} (C - t(S_k))^2} \tag{3}$$

Kaavassa S_k viittaa koneelle k annettuun tehtäväjoukkoon, C linjan tahtiaikaan ja M koneiden määrään. $t(S_k)$ on koneen k ns. koneaika eli annetun tehtäväjoukon töiden suorittamiseen kuluva aika.

4 Tuotantolinjojen tasapainotusongelman ratkaisumenetelmistä

Tasapainotusongelman ratkaisuja on haettu usealla eri tavalla. Karkeasti jaoteltuna menetelmät voidaan jakaa tarkkoihin ja epätarkkoihin menetelmiin. Tarkat menetelmät tuottavat varmasti optimaalisen ratkaisun tasapainotusongelmaan. Epätarkat menetelmät sen sijaan eivät välttämättä tuota tasapainotukseen optimaalista ratkaisua, mutta löytävät yleensä ns. tarpeeksi hyvän ratkaisun. Tällainen ratkaisu on ominaisuuksiltaan käytännön kannalta tarpeeksi lähellä optimaalista ratkaisua. Lisäksi epätarkan menetelmän tuottama ratkaisu saattaa myös jopa olla optimaalinen — optimaalisuutta ei vain voida osoittaa.

Pinedo esittää [Pin05] neljä ratkaisumenetelmien ryhmää. Menetelmät jaetaan *matemaattisiin ohjelmointimenetelmiin* (engl. *mathematical programming methods*), *tarkkoihin optimointimenetelmiin* (engl. *exact optimization methods*), *heuristisiin menetelmiin* (engl. *heuristic methods*) sekä *rajoiteohjelmointimenetelmiin* (engl. *constraint programming methods*). Erityyppiset menetelmät ovat Pinedon mukaan myös yhdisteltävissä *risteymäksi* eli *hybridimenetelmiksi* (engl. *hybrid techniques*).

Pinedo esittää myös kolme arvostelukriteeriä menetelmille. Ensimmäinen kriteeri on *laatu*. Tämä määrittää kuinka lähelle optimaalista ratkaisua päästään. Toinen kriteeri on *ratkaisuaika*. Mikäli ratkaisun laskeminen kestää kauan, kyseistä menetelmää ei voida käytää reaaliaikaisesti, vaan ratkaisu on laskettava etukäteen. Kolmas kriteeri on kehittämisen *helppous*. Menetelmät, jotka ovat nopeasti kehitettävissä ja muokattavissa tuleville ongelmille, ovat luonnollisesti myös käyttökelpoisimpia.

Mikäli ei ole ennalta päätetty, mitä menetelmää käytetään, kannattaa selvittää käytettävissä olevan datan luonnetta. Pinedon mukaan sekä matemaattisen että rajoiteohjelmoinnin käyttö tilanteissa, joissa käytettävissä oleva data on epämääräistä ja muuttuvaa, saattaa olla hidasta. Tällaisiin tilanteisiin heuristiset menetelmät sopivat paremmin.

4.1 Matemaattiset ohjelmointimenetelmät

Eräs tapa ryhmitellä matemaattisia ohjelmointimenetelmiä on jakaa ne *lineaarisiin* (engl. *linear programming, LP*) ja *epälineaarisiin ohjelmointimenetelmiin* (engl. *nonlinear programming, NLP*). *Kokonaisluku-* eli ns. *IP-menetelmää* (engl. *integer programming*) on lineaarisen ohjelmoinnin menetelmä, jossa käytettävät muuttujat ovat ainoastaan kokonaislukuja. *Joukon ositus* (engl. *set partitioning*), *joukon peittäminen* (engl. *set covering*) ja *joukon pakkaus* (*set packing*) ovat lineaarisen ohjelmoinnin eräitä tärkeitä sovellutuksia. Näitä viimeksi mainittuja ongelmia ei kuitenkaan käsitellä tässä työssä, vaan keskitytään menetelmiin yleisellä tasolla.

4.1.1 Lineaarinen ohjelmointi

Lineaarisessa ohjelmoinnissa tavoite esitetään funktiona ja rajoitteet lineaarisena yhtälöryhmänä. Lisäksi määritellään muuttujien arvoalueet. Tavoitteena on yleensä jonkin ominaisuuden (esim. kokonaiskustannuksen) minimointi. Seuraavassa on yleinen esimerkki lineaarisen ohjelmoinnin yhtälöryhmästä. Minimoitava tavoitefunktio on muotoa

$$min(c_1x_1 + c_2x_2 + \ldots + c_nx_n)$$

ja rajoitteet sekä muuttujien arvoalueet ovat:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\leq b_m \\ x_j &\geq 0, j = 1 \ldots n \end{aligned} \tag{4}$$

Muuttujia $x_1, \ldots, x_n$ kutsutaan *päätösmuuttujiksi*. Vakioiden $c_1, \ldots, c_n$ muodostamaa vektoria $\bar{c}$ kutsutaan *kustannusvektoriksi* (engl. *cost vector*). Vakioiden $a_{1j}, \ldots, a_{mj}$ muodostamaa vektoria $\bar{a}$ kutsutaan puolestaan *aktiviteettivektoriksi* j (engl. *activity vector* j) ja vektoria $\bar{b}$, jonka muodostavat $b_1, \ldots, b_m$, kutsutaan *resurssivektoriksi* (engl. *resource vector*). Näitä käyt-

tämällä ylläolevalle esitykselle saadaan seuraava matriisimuotoinen esitys:

$$\begin{aligned} &\text{Min } \bar{c}\bar{x}, \text{ kun} \\ &\qquad \mathbf{A}\bar{x} \leq \bar{b} \\ &\qquad \bar{x} \geq 0 \quad . \end{aligned}$$

Tämänkaltaisen lineaarisen yhtälöryhmän ratkaisemiseksi on olemassa useita tehokkaita algoritmeja, kuten *simplex*-algoritmit ja *sisäpiste*-metodit (engl. *interior point*), joista tärkeimmäksi mainitaan *Karmarkar*-algoritmi [Pin05]. Simplex-algoritmeista ei tiedetä, ovatko ne polynomiaikaisia, mutta Karmarkar-algoritmin sen sijaan tiedetään toimivan polynomisessa ajassa.

4.1.2 Kokonaislukuohjelmointi

Mikäli vaaditaan, että päätösmuuttujien $x_1, \ldots, x_n$ arvot ovat kokonaislukuja, puhutaan IP-menetelmästä eli kokonaislukuohjelmoinnista. Jos puolestaan vaaditaan, että vain osa muuttujista on kokonaislukuja, on kyse *MIP-menetelmästä* eli *sekalukuoptimoinnista* (engl. *mixed integer programming*). Kuten Pinedo mainitsee [Pin05], on tärkeää huomata, että IP- ja MIP-menetelmille ei ole olemassa tehokasta polynomiaikaista ratkaisualgoritmia — toisin kuin tavalliselle LP-menetelmälle, jossa muuttujat ovat jatkuvia.

4.1.3 Epälineaarinen ohjelmointi

Epälineaarinen ohjelmointi on lineaarisen ohjelmointimenetelmän yleistys, jossa sallitaan sekä tavoitefunktion että rajoitusten $(x_1, \ldots, x_n)$ olevan epälineaarisia. Eräs yleinen esimerkki epälineaarisesta ongelmasta on seuraavanlainen:

$$\begin{aligned} &Min(g(x_1, \ldots, x_n)), \text{ kun} \\ &\qquad f_1(x_1, \ldots, x_n) = 0 \\ &\qquad\qquad \vdots \\ &\qquad f_m(x_1, \ldots, x_n) = 0 \end{aligned} \tag{5}$$

Sekä tavoitefunktion että rajoitteiden on täytettävä tietyt vaatimukset, jotka varmistavat, että mahdollinen ratkaisu on varmistettavissa optimaali-

seksi. Näitä vaatimuksia kutsutaan *Kuhn-Tuckerin ehdoiksi* [Pin05].

Epälineaarisen ohjelmoinnin ongelmiin on monia ratkaisumenetelmiä. Tärkeimmiksi menetelmiksi Pinedo mainitsee *gradientti-menetelmän* ja ns. *penalty & barrier -menetelmän*. Tietyissä tapauksissa, kuten kaavan (5) tapauksessa, ongelma voidaan muuntaa yhdeksi tavoitefunktioksi:

Minimoi:
$$g(x_1, \ldots, x_n) + \lambda_1 f_1(x_1, \ldots, x_n) + \cdots + \lambda_m f_m(x_1, \ldots, x_n)$$

Tämänkaltaisen funktion osittaisderivointi ja yhdistäminen alkuperäisiin rajoituksiin (5) tuottaa $n+m$ tuntematonta $n+m$ yhtälössä ja on siten melko helposti ratkaistavissa.

4.1.4 Disjunktiivinen ohjelmointi

Matemaattisten ohjelmointimenetelmien rajoitteet ovat jaettavissa *konjuktiivisiin* ja *disjunktiivisiin* (engl. *conjuctive/disjunctive*) [Pin05]. Edellä esitettyjen esimerkkien rajoitteet, kaavat (4) ja (5), ovat konjuktiivista muotoa, koska minimoitavan funktion ratkaisu edellyttää kaikkien rajoitteiden samanaikaista toteuttamista. Rajoiteryhmää kutsutaan disjunktiiviseksi, jos ainakin yhden rajoitteen on toteuduttava, mutta ei välttämättä kaikkien. Disjunktiivisten rajoitteiden toteutuminen on siis osittain ehdollista.

4.2 Tarkat optimointimenetelmät

Tässä työssä tarkastellaan töiden järjestelyongelmaa, jossa on mukana sekä töiden aloitus- että määräajat. Tällainen töidenjärjestelyongelma on NP-vaikea. NP-vaikeille ongelmille ei tunneta polynomiaikaista ratkaisualgoritmia, joka tuottaisi optimaalisen ratkaisun [KT06]. Pinedo esittää [Pin05] kaksi menetelmätyyppiä, jotka soveltuvat NP-vaikeille ongelmille ja samalla tuottavat optimaalisen ratkaisun. Nämä menetelmätyypit ovat *dynaamiset ohjelmointimenetelmät* ja *kokonaislukuohjelmoinnin optimointimenetelmät*. On kuitenkin huomattava, että vaikka tarkkojen optimointimenetelmien tuottamat ratkaisut ovat optimaalisia, niiden tuottaminen saattaa olla liian hidasta käytännön sovelluksiin.

4.2.1 Dynaaminen ohjelmointi

Kun ongelmalle ei löydy polynomiaikaista ratkaisumenetelmää (kuten edellä esitetty kokonaislukuohjelmointi), on ehkä mahdollista käyttää dynaamista ohjelmointia. Tässä menetelmässä on piirteitä sekä *hajoita ja hallitse* -tekniikasta että raa'an voiman ns. *brute-force* -tekniikasta. Dynaamisessa ohjelmoinnissa yhdistetään pienempien osaongelmien ratkaisuja isommiksi kokonaisuuksiksi, mutta samalla käydään läpi erittäin iso joukko mahdollisia ratkaisuja [KT06]. Pinedon [Pin05] mukaan dynaaminen ohjelmointi soveltuu ainakin periaatteessa sekä NP-vaikeille ongelmille että polynomisessa ajassa ratkeaville ongelmille.

Dynaaminen ohjelmointi koostuu kolmen tyyppisistä lausekkeista: *alustuslausekkeista*, *rekursiolausekkeista* ja *optimaalisuuden mittauslausekkeista*. Alustus pitää sisällään systeemin tiedot ja rekursiolauseke huolehtii ongelman jakamisen pienempiin osiin. Optimaalisuuden mittauslauseke puolestaan määrittää, kuinka hyvä jokin ratkaisu tai sen osa on. Usein tämä tarkoittaa lukua, jota yritetään minimoida tai maksimoida. Dynaamista ohjelmointia voi suorittaa eteen- tai taaksepäin muuttamalla rekursioyhtälöä sopivasti.

4.2.2 IP-optimointi

Kuten aiemmin todettiin, kokonaislukuohjelmoinnin ongelmille ei tunneta polynomiaikaista ratkaisumenetelmää. Pinedo kertoo [Pin05], että parhaimmat menetelmät kokonaislukuongelman ratkaisemiseksi ovat

- ns. *branch-and-bound* -menetelmät
- tasonleikkaus-menetelmät
- yhdistelmätekniikat

Branch-and-bound -tekniikoissa (lyh. BB) ongelma jaetaan osaongelmiin, joten ne muistuttavat toiminnaltaan hajoita-ja-hallitse -tekniikkaa. BB-tekniikassa käytetään lisäksi kokonaislukurajoitteiden lievennystä sekä uusien rajoitteiden lisäämistä. Tasonleikkaus-menetelmät perustuvat myös rajoitteiden lisäämiseen, mutta pääajatuksena on ratkaista ongelma lineaarisen ohjelmoin-

nin kautta. Hybridi-menetelmissä nimensä mukaisesti yhdistetään ominaisuuksia useista eri menetelmistä. On mahdollista yhdistää esimerkiksi sekä branch-and-bound että tasonleikkaus-menetelmä.

Pinedon [Pin05] mukaan kokonaislukuohjelmoinnissa ongelma ratkaistaan yleensä ns. *branch-and-bound* -tekniikalla. Tekniikan ideana on relaksoida kokonaislukuongelma lineaariseksi ongelmaksi, jolloin ratkaisun ei tarvitse olla kokonaisluku. Kokonaislukuratkaisujen etsintää voidaan tämän jälkeen rajoittaa saatujen reaalilukuratkaisujen avulla. Pinedo merkitsee lineaariseksi ongelmaksi relaksoidun ongelman ratkaisuvektoria termillä $\bar{x}^0$. Jos $\bar{x}^0$-vektorin arvot ovat kokonaislukuja, se on myös kokonaislukuohjelmoinnin ratkaisu. Muussa tapauksessa $\bar{c}\bar{x}^0$ antaa kuitenkin alarajan optimaaliselle ratkaisulle.

Kun saatu ratkaisu ei koostu kokonaisluvuista, jaetaan ongelma kahteen osaongelmaan antaen kummallekin toisensa poissulkevat rajoitteet. Pinedo esittää tilanteen seuraavasti. Otetaan jokin $\bar{x}^0$:n kokonaislukujen joukosta poikkeava arvo x_j, joka on reaaliluku, toisin sanoen $x_j = r$. Osaongelmalle 1 (merk. SP(1)) alkuperäistä kokonaislukuongelmaa muokataan tällöin antamalla lisärajoite $x_j \leq \lfloor r \rfloor$. Toinen osaongelma (merk. SP(2)) on myös kopio alkuperäisestä kokonaislukuongelmasta, mutta lisärajoitteena on $x_j \geq \lceil r \rceil$. Näin saatujen osaongelmien rajoitteita relaksoidaan jälleen siten, että ratkaisujen ei tarvitse olla kokonaislukuja. Mikäli osaongelman ratkaisuksi ei näin saada kokonaislukua, suoritetaan uusi jako, jolloin saadaan osaongelmat SP(1,1) ja SP(1,2). Näin jatketaan, kunnes löytyy kokonaislukuratkaisu tai saatu ratkaisu ei toteuta annettua uutta rajoitetta. Näin saaduista ratkaisuista valitaan lopuksi paras.

Edellä esitetyn yksinkertaisen tekniikan lisäksi on kehitetty hienostuneempia menetelmiä, jotka ovat osoittaneet käyttökelpoisuutensa myös käytännössä [Pin05].

- Lagrangen relaksaatiotekniikka (engl. *Lagrangian relaxation*)
- branch-and-cut -menetelmä
- branch-and-price -menetelmä

Lagrangen relaksointi on tekniikka, jolla branch-and-bound -menetelmässä saadaan tarkempia alarajoja mahdollisille ratkaisuille. Tässä tekniikassa relaksoidaan itse ongelman rajoitteita eikä niinkään ratkaisun kokonaislukuihin kuulumista.

Branch-and-cut -menetelmä on yhdistelmätekniikka, jossa yhdistetään branch-and-bound -menetelmä edellä mainittuun tasonleikkaus-menetelmään. Tässä menetelmässä kunkin osaongelman alaraja etsitään tasonleikkausalgoritmia käyttäen. Tarkoituksena on tuottaa tarkempia arvioita kokonaislukuratkaisua varten.

Branch-and-price -menetelmä soveltuu kokonaislukuongelmille, joissa on erityisen paljon muuttujia. Tämän menetelmän osaongelmissa otetaan huomioon aina vain muuttujien jokin osajoukko. Muuttujia arvioidaan sen perusteella, mihin osaongelmiin ne kuuluvat ja miten niitä sisältävät osaongelmat ratkeavat. Pinedon mukaan branch-and-cut -menetelmät sopivat rinnakkaisten koneiden töidenjärjestelyongelmien ratkaisemiseen. [Pin05]

4.3 Heuristiset menetelmät

On monia linjan tasapainotusongelmia ja töidenjärjestelyongelmia, jotka ovat erityisen vaikeita ja kuuluvat NP-vaativuusluokkaan. Vaikka ongelma olisi muotoiltavissa esimerkiksi edellä esitetylle kokonaislukuohjelmoinnin menetelmälle, näille ongelmille ei tunneta polynomiaikaista ratkaisumenetelmää.

Heuristiset menetelmät luottavat ennalta hyväksitodettuihin keinoihin, joilla etsitään ongelmaan kelvollinen ratkaisu. Näitä menetelmiä käytetään, kun optimaalisen ratkaisun etsiminen veisi kauan tai tarvitsisi kohtuuttomasti resursseja. Heuristisiin menetelmiin kuuluu niin perinteisiä hyväksihavaittuja algoritmitekniikoita, sumean logiikan käyttöä kuin myös ns. evolutionäärisiä algoritmeja. Esimerkiksi töiden järjestelyssä on useita yksinkertaisia heuristisia töiden järjestelymenetelmiä ja ongelmaksi muodostuu lähinnä oikean menetelmän valinta.

Heuristiset töiden järjestelymenetelmät voidaan jakaa *staattisiin* ja *dynaamisiin* menetelmiin. Staattiset menetelmät eivät riipu ajasta. Niissä käytetään vain töiden ja koneiden ominaisuuksia päätöksenteossa. Staattisia me-

netelmiä kutsutaan usein myös *sääntöpohjaisiksi menetelmiksi*, koska järjestely tehdään tietyn ennalta sovitun ja muuttumattoman säännön mukaan. Dynaamisissa menetelmissä aika on olennaisena tekijänä, kun esimerkiksi tutkitaan jäljellä olevaa aikaa tai seuraavaa lähintä takarajaa, johon mennessä jokin työ pitäisi suorittaa. Menetelmät voidaan myös jakaa paikallisiin ja ei-paikallisiin menetelmiin, joissa paikalliset menetelmät käyttävät ainoastaan kyseisen työn ja siihen liittyvän koneen ominaisuuksia, kun ei-paikalliset menetelmät käyttävät myös muiden koneiden (kuten tehtäväjonojen) tietoja.

4.3.1 Sääntöpohjaiset tekniikat

On olemassa monia yleisesti hyväksi havaittuja töiden järjestelytekniikoita. Näitä kutsutaan *sääntöpohjaisiksi tekniikoiksi*, koska ne perustuvat johonkin tiettyyn sääntöön, jonka mukaan järjestelyalgoritmi toimii. Menetelmien tavoite vaihtelee. Jokin menetelmistä pyrkii minimoimaan myöhästymiset, kun toista voidaan käyttää rinnakkaisten linjojen työmäärien tasaamiseen.

Taulukossa 1 on lueteltu Pinedon esittelemät [Pin05] yleisimmät sääntöpohjaiset töidenjärjestelytekniikat. Ne on jaettu kolmeen ryhmään, joista ensimmäisen ryhmän menetelmät riippuvat töiden saatavuusajoista ja vaadituista valmistumisajoista. Toisen ryhmän menetelmien toiminta perustuu töiden suoritusaikoihin sekä mahdollisiin ennaltamäärättyihin työjärjestyksiin. Viimeinen ryhmä koostuu sekalaisista menetelmistä.

4.3.2 Geneettiset algoritmit

Evoluutionääristen algoritmien tunnetuin ja käytetyin joukko on *geneettiset algoritmit*. Näissä algoritmeissa ongelmanratkaisuun sekoitetaan genetiikasta tuttuja peruskäsitteitä, kuten *yksilön kelpoisuus*, *ominaisuuksien periytyminen* ja *mutaatio*. Tavoite on kehittää satunnaisesti luodusta ratkaisujoukosta parempi ratkaisu simulaation ja kelpoisuusvalintojen avulla. Geneettisiä algoritmeja on tutkittu paljon ([VHN03][Bal11][RK06]) ja aihe on kirjallisuudessa hyvin katettu. Koza, Bennett, Andre ja Keane [KBAK99] antavat toimintaperiaatteen geneettiselle algoritmille kuvassa 14.

Taulukko 1: Yleisimmät sääntöpohjaiset skedulointimenetelmät [Pin05]

Lyhenne	Englanninkielinen nimitys
ERD	Earliest Release Date first
EDD	Earliest Due Date first
MS	Minimum Slack first
LPT	Longest Processing Time first
SPT	Shortest Processing Time first
WSPT	Weighted Shortest Processing Time first
CP	Critical Path
LNS	Largest Number of Successors
SIRO	Service In Random Order
SST	Shortest Setup Time first
LFJ	Least Flexible Job first
SQNO	Shortest Queue at the Next Operation

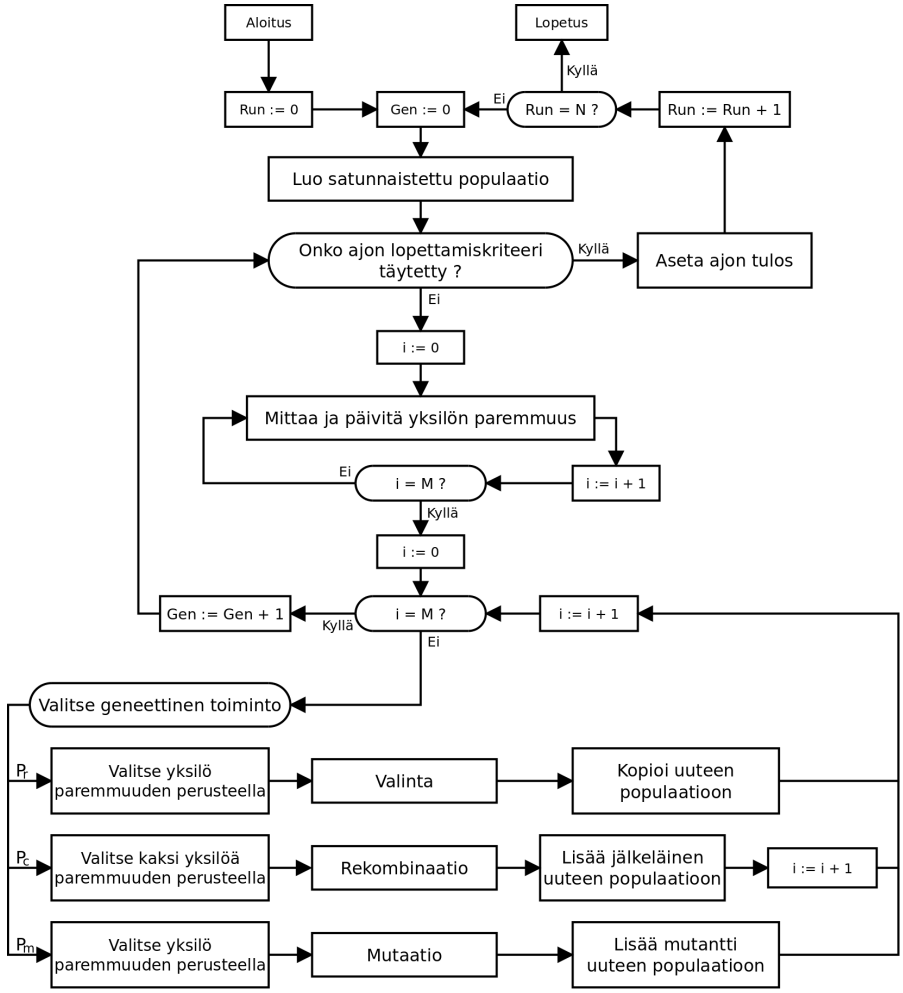
Algoritmi alkaa ensimmäisen sukupolven eli aloitusgeneraation luomisella. Tätä kutsutaan usein *0-generaatio*:ksi ja se sisältää M kappaletta satunnaisesti luotuja ratkaisuehdokkaita eli *yksilöitä*. Kussakin yksilössä käytetään ennaltasovittua *koodaustapaa*, jolla yksilön ominaisuudet tallennetaan. Koodaustapa koostuu ns. *merkistöstä*, joista muodostetaan tietyn pituinen merkkijono. Esimerkiksi, jos käytössä on merkkijonon pituus $L = 3$ ja merkistön koko $K = 2$, voidaan tehdä merkkijono "011".

Itse algoritmi koostuu kolmesta vaiheesta, joita toistetaan, kunnes vaiheen 1 ehdot toteutuvat.

1. Tarkastetaan, ovatko sukupolven edustajat riittävän hyviä, tai onko generoitujen sukupolvien lukumäärän yläraja (G) saavutettu. Koska kyseessä on satunnaistettu alkutilanne, on periaatteessa mahdollista että jo 0-generaatio tuottaa tarvittavan hyvän ratkaisun.

2. Mitataan kunkin populaatioon kuuluvan ratkaisun *sopivuus* (engl. *fitness*). Tähän käytetään ns. *sopivuusfunktiota*, joka on toteutettu sen mukaan, mitä ominaisuuksia ratkaisulta halutaan.

3. Luodaan uusi sukupolvi eli generaatio. Edellisestä populaatiosta valitaan sopivuuden perusteella yksilöitä, joille suoritetaan jokin geneet-



Kuva 14: Kaavio geneettisen algoritmin toiminnasta [KBAK99, s. 23]

tinen operaatio. Suoritettava operaatio valitaan satunnaisesti, joskin todennäköisyydet ovat määritettävissä. Operaatioita ovat *valinta*, *risteymä* (eli *rekombinaatio*) ja *mutaatio*.

Geneettisten algoritmien eräänä merkittävänä erona on niissä tarvittavan tiedon vähyys. Geneettisiä algoritmeja voidaan käyttää ongelman ratkaisemiseen, vaikka itse ongelman rakenteesta ei olisi täydellistä kuvaa. Lisäksi geneettiset algoritmit tuottavat yleensä melko hyviä tuloksia ja niiden toteutus on suhteellisen yksinkertaista. Koska genettisessä algoritmissa on kyseessä satunnaistetun datan satunnaistettu käsittely, on tilastollisista syistä syytä

suorittaa ajo tarpeeksi monta kertaa. Tämän vuoksi menetelmän huonona puolena on yleensä pitkät laskenta-ajat. [Pin05]

4.4 Rajoiteohjelmointimenetelmä

Vaikka matemaattisissa ohjelmointimenetelmissä käytetään rajoitteita, tarkoitetaan *rajoiteohjelmointimenetelmällä* hieman laajempaa käsitettä. Pinedo antaa [Pin05] lyhyen esittelyn tästä menetelmästä ja kertoo, että toisin kuin matemaattisen ohjelmoinnin tapauksessa, rajoiteohjelmoinnin juuret ovat tietojenkäsittelytieteessä ja tekoälyn kehittämisessä. Itse rajoitteiden esittäminen muistuttaa edellä esitettyjä matemaattisia esitystapoja, mutta on niitä vapaampi. Rajoiteohjelmoinnissa ongelma koostuu muuttujista $x_1, \ldots, x_n$ ja niiden mahdollisista arvoista. Merkitään, että joukko $\mathcal{D}_j$ koostuu muuttujalle x_j sallituista arvoista. Itse rajoite määritellään muuttujan arvon kuulumisena johonkin ennalta määrättyyn arvojoukkoon. Tämä voidaan kuvata siten, että muuttujat $x_1, \ldots, x_n$ kuuluvat johonkin joukon $\mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n$ osajoukkoon $\mathcal{S}$. Rajoite voidaan kuvata matemaattisena funktiona, kuten esimerkiksi

$$f(x_1, \ldots, x_n) = 1,$$

jos ja vain jos rajoite toteutuu. Näin saadaan *rajoitteen toteutumisongelmalle* (engl. *constraint satisfaction problem (CSP)*) seuraava määritelmä:

$$\begin{aligned} f_i(x_1, \ldots, x_n) &= 1, \qquad i = 1, \ldots, m \\ x_j &\in \mathcal{D}_j, \qquad j = 1, \ldots, n \end{aligned} \tag{6}$$

Pinedo huomauttaa myös, että rajoiteohjelmointimenetelmässä ei käytetä tavoitefunktiota, jonka arvoa yritetään minimoida tai maksimoida. Kyseessä on ns. *kelpoisuusongelma* (engl. *feasibility problem*), jossa riittää, että muuttujat ovat kelvollisia eli toteuttavat niille annetut rajoitteet. Huomion arvoista on myös, että rajoitteiden tyyppi voi vaihdella. Luvun ei tarvitse löytyä joltakin lukuväliltä, vaan rajoite voi olla myös jokin looginen operaatio tai jonkin joukon mahtavuuteen liittyvä ominaisuus.

Rajoiteohjelmointi antaa vapaammat kädet ongelman mallinnukselle verrattuna matemaattiseen ongelmanmallinnukseen. Tästä syystä LP- ja IP-ongelmien ratkaisumenetelmät eivät välttämättä sovellu rajoiteongelmien ratkaisemiseen. Bockmayr ja Pisaruk ovat kuitenkin tutkineet, kuinka LP-, IP- ja MIP-ratkaisumenetelmiä voidaan yhdistää rajoiteohjelmointiin [BP01] [BP06]. He ehdottavat rajoiteohjelmoinnin käyttöä epäpätevien ratkaisujen eliminoimiseksi ja branch-and-cut -menetelmää leikkausten löytämiseksi.

Rajoiteohjelmointia varten on kehitetty jopa ohjelmointikieliä. Pinedo mainitsee näistä kaksi. Toinen on *Prolog* ja toinen on *Optimization Programming Language (OPL)*. Prolog ei kuulu yleisimpiin käytettyihin ohjelmointikieliin — lähinnä rajoittuneen soveltuvuutensa vuoksi. OPL sen sijaan on kehitetty myöhemmin ja se soveltuu sekä optimointiongelmien ratkaisuun kuin myös rajoiteongelmien ratkaisuun. [Pin05]

5 Tuotantolinjojen tasapainotusongelmien ratkaiseminen

Nykyajan piirilevyladonnan töidenjärjestely on yllättävän monimutkaista. Kuten edellä on todettu, ladontalinjojen käyttöön vaikuttavat komponenttiasettelut, linjojen omistaminen tietyille tuotteille sekä töiden määräajat. Kaikki nämä osaongelmat on yhdistettävä, jotta saadaan optimaaliset aikataulut ja komponenttiasettelut kullekin linjalle ja koneelle.

Tämän ongelman ratkaiseminen jaetaan tässä tutkielmassa neljään vaiheeseen. Ratkaiseminen aloitetaan yksinkertaisesta tapauksesta ja lisämääreitä lisätään vähitellen. Vaiheet ovat:

1. Yhden linjan tapaus, jossa keskitytään komponenttiasettelujen järkevään hallintaan ja pyritään suorittamaan työt mahdollisimman nopeasti.

2. Ongelmaan lisätään muut linjat. Tällöin asettelu- ja töidenjärjestelyongelmaan lisätään linjojen tasapainotus.

3. Osa linjoista omistetaan tietyille tuotteille, joita valmistetaan suuria määriä verrattuna muihin tuotteisiin.

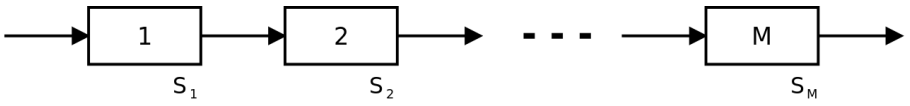
4. Ladontatöiden aloitusajat ja takarajat otetaan käyttöön.

Kun ongelmaa aloitetaan ratkaisemaan yksi osaongelma kerrallaan, on mahdollista, että lisättävät määreet muuttavat koko lähestymistapaa. Esimerkiksi töiden takarajojen tuominen mukaan muuttaa koko ongelman tavoitteen töiden nopeasta suorittamisesta myöhästymisten minimointiin.

5.1 Yksi linja

Kun tarkastellaan vain yhtä linjaa, voidaan keskittyä suorittamaan ladontatyöt mahdollisimman nopeasti ja sujuvasti. Tämä mahdollistetaan käyttämällä jokainen linjan kone mahdollisimman hyvin hyödyksi. Hyötykäyttö maksimoidaan järjestelemällä työt siten, että linjan joka koneella on yhtä pitkäkestoinen työ. Näin jokaisen koneen työn kesto on mahdollisimman lähellä

luvussa 3.1 mainittua linjan tahtiaikaa ja koneiden joutoaika jää mahdollisimman pieneksi.



Kuva 15: Yksi tuotantolinja, jossa M konetta

Lähdetään tarkastelemaan ongelmaa kuvan 15 osoittamasta tilanteesta, jossa on käytössä yksi tuotantolinja. Linjalla on M konetta, joista kullakin on oma komponenttiasettelunsa S_m. Oletetaan yksinkertaisuuden vuoksi, että komponentit voidaa latoa missä järjestyksessä tahansa, eikä komponenttien sijoituspaikka levyllä vaikuta nopeuteen. Tosielämässä myös komponenttien sijoittelu vaikuttaa ladonta-aikaan. Ladonta on hitaampaa, mikäli ladontapään matka komponenttikelalta sijoituspaikkaan on pitkä. Koska tämän tutkielman puitteissa ei olla kiinnostuneita komponenttien sijoittelusta piirilevyllä, voidaan lisäksi olettaa, että kunkin yksittäisen komponentin sijoitus piirilevylle kestää keskimäärin yhtä kauan. Seuraavassa tästä ajasta käytetään merkintää t_c. Jos komponentin ladonta-aika ei riipu sen tyypistä, voidaan ladottavat komponentit sijoittaa siten, että kullekin koneelle tulee yhtä paljon komponentteja ladottavaksi. Tässä on kuitenkin huomioitava eri komponenttityyppien vaatimat suuttimet. Toisin sanoen eri suuttimia vaativat komponentit on syytä sijoittaa eri koneille. Mikäli koneiden kelakapasiteetti on riittävä, tämä riittää jo ongelman ratkaisemiseksi.

On huomattava, että itse ladonta-aika on vain osa koko työajasta. Suuri osa koko työajasta kuluu alkuvalmistelujen tekemiseen. Tästä syystä piirilevyt ladotaan usein ns. *eräajona* (engl. *batch* tai *lot*). On kuitenkin mahdollista, että tuotteita on useita erilaisia ja vaadittujen komponenttityyppien yhteismäärä on suuri. Tällöin alkuvalmistelut suoritetaan eräkohtaisesti ladontaerien välissä, jolloin itse ladontaan käytettävän ajan osuutta saadaan kasvatettua. Myös eri komponenttityyppien suuri määrä ja ladontakoneiden kapasiteettirajoitukset puoltavat eräajon käyttöä. Seuraavassa tutkitaan yhden linjan käyttöä eräajossa.

5.1.1 Ladontatehtävien jako koneille

Ladontatehtävien jako-ongelman tarkastelu aloitetaan vaatimuksista eli piirilevyjen vaatimista komponenteista. Ladottavia piirilevytyyppejä on J kappaletta. Komponenttityyppejä on p kappaletta, ja ne merkitään $T_1, \ldots, T_p$. Merkitään matriisiin A komponenttien tarve piirilevytyypeittäin. Tällöin $a_{jt} = x$ ilmaisee, että piirilevyyn j tarvitaan x kappaletta komponenttia t.

Selvennetään asiaa ensin yksinkertaisen esimerkin avulla. Tehtävänä on latoa kolmea konetta käyttämällä kolmeen erityyppiseen piirilevyyn (100 kpl kutakin) komponentteja T taulukon 2 mukaisesti.

Taulukko 2: Kolmen esimerkkilevyn vaatimat komponenttimäärät

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Levy 1	8	8	5	10	10					
Levy 2		3	2		1	5	5	10		4
Levy 3	12	10	2				12	6	6	4

Komponenttien vaatimusmatriisi A:

$$A = \begin{pmatrix} 8 & 8 & 5 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 5 & 5 & 10 & 0 & 4 \\ 12 & 10 & 2 & 0 & 0 & 0 & 12 & 6 & 6 & 4 \end{pmatrix} \tag{7}$$

Laskemalla komponenttien määrät yhteen saadaan levylle 1 yhteensä 41 komponenttia, levylle 2 30 ja levylle 3 52 komponenttia. Kun kutakin levyä tehdään 100 kappaletta, ladottavien komponenttien kokonaismäärä on $100 \cdot (41+30+52) = 12300$. Oletetaan, että levyjen ladonta suoritetaan eräajona yksi levytyyppi kerrallaan. Jotta linjan tahtiaika saadaan pysymään mahdollisimman pienenä, on syytä tasoittaa koneiden työajat. Optimitilanteessa ensimmäistä levyä ladottaessa tahtiajaksi saataisiin $C_{opt1} = \lceil t_{total}/M \rceil \cdot t_c = \lceil 41/3 \rceil t_c = 14t_c$. Toisin sanoen kullakin koneella olisi noin 14 komponenttia ladottavana. Vastaavasti levyille 2 ja 3 saadaan optimaaliset tahtiajat $C_{opt2} = 10t_c$ ja $C_{opt3} = 18t_c$.

Kun komponenttityypit jaetaan koneiden kesken, pyritään samalla tasoittamaan ladottavien komponenttien määrä konetta kohti. Mikäli koneissa on tilaa ylimääräisille komponenttityypeille, voidaan töitä tasoittaa edelleen jakamalla tiettyjen komponenttien ladonta usealle koneelle. Jos kuitenkin kutakin komponenttityyppiä ladotaan ainoastaan yhdellä koneella, tähän ideaalitilanteeseen ei välttämättä päästä. Tässä esimerkissä pyritään säästämään kelapaikkojen käyttöä, eikä jaeta samaa komponenttityyppiä usealle koneelle.

Työaikoja tasattaessa voidaan käyttää luvussa 4.3.1 mainittua *longest processing time first*-metodia hyväksi. Tällä menetelmällä jäljellä olevista töistä valitaan pitkäkestoisin ja se määrätään koneelle, jolla on valintahetkellä vähiten töitä. Luonnollisesti tässä esimerkissä työn kesto vastaa komponenttien määrää. Yhdeksi vaihtoehdoksi tällöin muodostuu asettelu, jossa komponentit T_1 ja T_2 ladotaan koneella 1, T_3 ja T_4 koneella 2 ja jättää T_5-komponentit koneen 3 tehtäväksi. Näin koneiden suoritusajoiksi saadaan $t_1 = 16t_c$, $t_2 = 15t_c$ ja $t_3 = 10t_c$ (ks. taul. 3). Näistä suurin määrää linjan tahdin eli linjan tahtiaika ensimmäistä levysarjaa ladottaessa on $C_1 = 16t_c$.

Vastaavasti levyille 2 ja 3 saadaan taulukon 3 mukainen komponenttijako sekä tahtiajat $C_2 = 10t_c$ ja $C_3 = 18t_c$.

Taulukko 3: Levyjen ladontatehtävien jako sekä koneiden työajat

		Komponentit	Työaika
Levy 1	Kone 1	T_1, T_2	16
	Kone 2	T_3, T_4	15
	Kone 3	T_5	10
		Komponentit	Työaika
Levy 2	Kone 1	T_8	10
	Kone 2	T_5, T_6, T_{10}	10
	Kone 3	T_2, T_3, T_7	10
		Komponentit	Työaika
Levy 3	Kone 1	T_1, T_9	18
	Kone 2	T_7, T_{10}	16
	Kone 3	T_2, T_3, T_8	18

Huomataan, että levyjen 2 ja 3 kohdalla päästään edellä laskettuihin parhaisiin mahdollisiin tahtiaikoihin. Levyn 1 tapauksessa 16 yksikön tahtiaika

jää optimiajasta 2 yksikköä eli 14%. Tämä on kokonaisuudessaan melko vähän.

Edellä ollut esimerkki on idealistinen, sillä ladontakoneilla on myös ominaisuuksia, jotka vaikuttavat töiden järkevään sijoitteluun ja nämä ominaisuudet joudutaan myös ottamaan huomioon. Näitä ominaisuuksia ovat *nopeus*, *kelakapasiteetti* ja *suutinvalikoima*. Nopeat koneet pystyvät suoriutumaan ladontatehtävästä nopeammin. Suuri kelakapasiteetti mahdollistaa suuren komponenttivalikoiman. Suutinvalikoimasta riippuu, mitä komponentteja koneella voidaa latoa. Näihin ladontakoneiden ominaisuuksiin keskitytään usean linjan tarkastelussa luvussa 5.2.

5.1.2 Komponenttien asettelustrategioista

Kun koneiden keskinäiset tehtäväjaot on selvitetty, kiinnitetään huomio järjestykseen, jolla levyt ladotaan. Jokaisen komponenttityypin vaihtaminen toiseen vie aikaa. Komponenttityyppien joukkoa kutsutaan siis *komponenttiasetteluksi* (engl. *setup*).

Yilmaz, Grunow, Günther ja Yapan [YGGY07] antavat neljä eri strategiaa asetteluiden käytölle. Seuraavassa esitellään lyhyesti nämä strategiat ja tarkastellaan niistä kahta tarkemmin — minimiasettelustrategiaa ja ryhmittelystrategiaa.

- **Minimiasettelustrategia (engl. *minimum setup*)**
  Minimiasettelustrategia keskittyy minimoimaan ajan, joka kuluu asettelujen vaihtoon, kun ladonnassa siirrytään piirilevystä toiseen [CC03] [SSJN06] [SSJ+03].

- **Yksittäisasettelun strategia (engl. *unique setup*)**
  Yksittäisasettelustrategiassa kiinnitetään huomiota myös itse ladontakoneen ominaisuuksiin. Tässä strategiassa asettelu tehdään kullekin levytyypille erikseen sekä pyritään sijoittamaan komponenttikelat siten, että koneen ladontapäiden työ helpottuisi mahdollisimman paljon. Esimerkiksi ladontapäiden kulkemaa matkaa voidaan minimoida, jos tiedetään tietyn tyyppisten komponenttien vaadittu sijainti piirilevyllä.

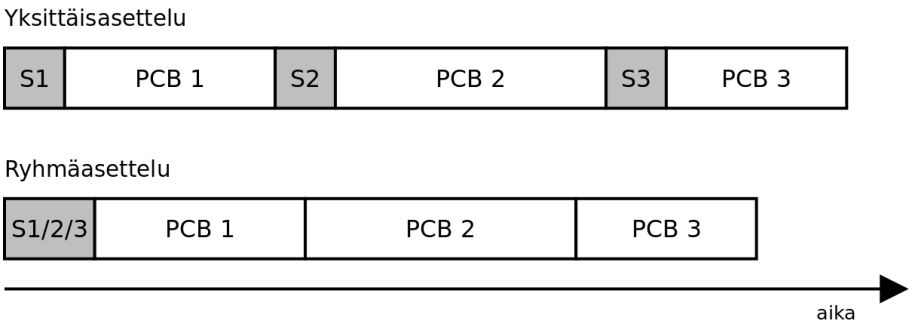
Tämänkaltaisesta strategiasta on hyötyä, kun yhdellä linjalla tuotetaan suuria määriä samaa tuotetta.

- **Osittainen asettelustrategia (engl. *partial/fixed setup*)**
  Tässä strategiassa komponenttityypit jaetaan yhteisiin ja piirilevykohtaisiin. Ideana on pitää yhteiset komponentit koko ajan koneissa ja vaihtaa levykohtaiset komponentit, kun levytyyppi vaihtuu.

- **Ryhmäasettelustrategia (engl. *group setup*)**
  Ryhmäasettelustrategia muistuttaa minimiasettelustrategiaa. Tätä käytetään, kun piirilevyt on jaettu levytyypeittäin ryhmiin. Tässä strategiassa yritetään minimoida aikaa, joka kuluu asettelun vaihtoon piirilevyperheen vaihtuessa. [KHJN02] [SJP+98] [SSJ+03] [YGGY07]

Strategian valitsemiseen vaikuttaa lähinnä linjaston koko, ladottavien piirilevyjen määrät sekä niiden erilaisuudet. Ylimazin, Grunowin, Güntherin ja Yapanin [YGGY07] mukaan minimiasettelustrategia sopii pienille ladontapajoille, joissa ladontatehtävät vaihtuvat usein. Sen sijaan yksittäisasettelun strategia soveltuu suureen massatuotantoon linjan koosta riippumatta.

Osittaisen asettelustrategian idea on jättää osa komponenteista paikalleen ajan säästämiseksi. Tämä asettelustrategia on yleisesti käytössä ja hyväksi todettu. Lisäksi tässä strategiassa ns. *syötinkärryjen* (engl. *feeder trolley* käyttö on tehokasta. Syötinkärryjen ideana on mahdollistaa useiden kelojen vaihto samanaikaisesti. Ylimaz, Grunow, Günther ja Yapan [YGGY07] kuitenkin kritisoivat osittaista asettelustrategiaa. Heidän mukaan strategiasta saatava hyöty jää pieneksi, jos on käytössä syötinkärryt. Tämän lisäksi, vaikka kärryjä ei käytettäisi, strategian noudattaminen on monimutkaista, koska päätös kelan vaihtamiseen riippuu aina seuraavasta levytyypistä. Näin ollen ladontatöiden järjestelyn kannalta vaikutukset asetteluihin eivät ulotu vain edellisen piirilevytyypin komponenttivaatimuksiin, vaan myös sitä edeltäviin levytyyppeihin. Näitä ongelmia ei ryhmäasettelustrategialla ole, joka Ylimazin, Grunowin, Güntherin ja Yapanin [YGGY07] mukaan on käyttökelpoisin lähestymistapa, kun tuotantomäärät ovat keskinkertaisia ja tuotteiden vaihtuvuus on vaihteleva.



Kuva 16: Kolmen levyerän valmistuksen kesto yksittäisellä ja ryhmäasettelulla [YGGY07, s.875]

Kuvassa 16 nähdään ero yksittäisen ja ryhmäasettelun välillä. Kuvan tilanteessa levyt 1–3 kuuluvat samaan ryhmään. Yksittäisessä asettelustrategiassa asettelu suoritetaan jokaiselle levylle erikseen, kun ryhmäasettelustrategiassa asettelu suoritetaan kullekin ryhmälle. Yleisesti ottaen levyryhmän vaatiman asettelun tekemiseen kuluva aika on pienempi kuin levyjen erikseen vaatimien asettelujen teko. Ryhmäasettelun vaatima aika ei voi olla suurempi, ja yhtäsuurikin se on vain, jos mitkään komponenteista eivät ole levyjen kesken yhteisiä. Lisäksi aikaa voi säästyä, jos tuotantoa ei tarvitse pysäyttää niin usein asettelujen vaihtoa varten.

Koska yksittäisasettelussa käytettyä kelasijoittelun optimointia ei voida sellaisenaan käyttää ryhmäasettelussa, voi yksittäisten komponenttien ladonta-aika kasvaa. Tällöin kokonaisladonta-aika voi myös kasvaa, mikäli ladottavien erien koot ovat suuria. Tässä tutkielmassa ei kuitenkaan käsitellä kelasijoittelun optimointia eikä komponenttien sijoittelupaikkoja piirilevyllä.

5.1.3 Minimiasettelu

Minimiasettelustrategian periaatteena on pitää asettelujen vaihdot mahdollisimman nopeina. Tällöin otetaan huomioon, mitä komponentteja koneisiin on jo ladattu ja mitä komponentteja on seuraavaa levyä varten ladattava. Yksinkertaisuuden vuoksi seuraavassa oletetaan, että asettelun vaihto ei ole inkrementaalinen ts. koneeseen ei jätetä komponentteja, joita ei tarvita.

Van Hop ja Nagarur esittävät [VHN03] käsitteen *samankaltaisuus* (engl.

similarity). Samankaltaisuus mittaa paljonko kahdessa komponenttiasettelussa on yhteisiä alkioita, eli komponenttityyppejä. Joukko-opillisesti voidaan puhua kahden joukon leikkauksen mahtavuudesta. Van Hop ja Nagarur käyttävät tätä käsitettä ryhmäasettelussa, mutta sen pohjalta voidaan yrittää minimoida myös minimiasettelustrategian vaihtoaikoja.

Periaattena on, että asettelun vaihto toiseen samankaltaiseen asetteluun käy nopeasti. Merkitään piirilevyjen 1–3 vaatimia asetteluja seuraavasti:

$$S_1 = \{T_1, T_2, T_3, T_4, T_5\}$$
$$S_2 = \{T_2, T_3, T_5, T_6, T_7, T_8, T_{10}\}$$
$$S_3 = \{T_1, T_2, T_3, T_7, T_8, T_9, T_{10}\}$$

Kokonaisvaihtoaika liittyy kuitenkin siihen, kuinka monta komponenttityyppiä joudutaan vaihtamaan. Toisin sanoen asettelujen ero kertoo vaihtoon kuluvan ajan. Merkitään samankaltaisuusfunktiota merkinnällä *Sim* ja erilaisuutta merkinnällä *Diff*, jolloin näiden joukkojen keskinäiset samankaltaisuudet ja erot voidaan ilmoittaa seuraavasti:

$$Sim(S_1, S_2) = 3 \qquad Diff(S_1, S_2) = 6$$
$$Sim(S_1, S_3) = 3 \qquad Diff(S_1, S_3) = 6$$
$$Sim(S_2, S_3) = 5 \qquad Diff(S_2, S_3) = 4$$

Näistä arvoista huomataan, että viimeksi mainittua asetteluvaihtoa (levyjen 2 ja 3 välillä) kannattaa käyttää. Toisin sanoen levyt 2 ja 3 kannattaa valmistaa peräkkäin. Näin ollen hyviä asettelujärjestyksiä ovat mm. $S_1 \Rightarrow S_2 \Rightarrow S_3$ tai $S_2 \Rightarrow S_3 \Rightarrow S_1$. Koska kaksi ensimmäistä samanlaisuusarvoa ovat yhtäsuuret, niiden asetteluvaihtojen järjestyksillä ei ole väliä. Arvoista pystytään myös päättelemään, että huonoimmat asettelujärjestykset ovat niitä, jotka kayttävät vain kahta ensimmäistä yhdistelmää.

On syytä huomata, että asettelujen optimointi on yleensä ristiriidassa töiden järjestelyn kanssa. Näin on varsinkin tässä tapauksessa, kun ladontaa ei suoriteta vain yhdellä koneella, vaan monta konetta käsittävällä linjalla. Taulukosta 3 huomataan, että aiemmin hyväksi ja nopeaksi havaittu komponenttijako aiheuttaa myös paljon komponenttivaihtoja. Tahtiaikojen

minimointiin käytetty LPT-menetelmä nimittäin sijoittaa eri levyjen yhteisiä komponentteja eri koneille. Esimerkiksi levyn 1 tapauksessa komponentit T_2 ja T_3 sijaitsevat koneilla 1 ja 2, mutta levyn 2 tapauksessa ne sijaitsevat molemmat koneella 3. Näin ollen joudutaan valitsemaan, halutaanko mahdollisimman pienet tahtiajat vai mahdollisimman vähän muutoksia asetteluiden kesken.

Seuraavassa tutkitaan, kuinka paljon jouduttaisiin hyvistä tahtiajoista tinkimään, mikäli pyritään minimoimaan asettelujen vaihdot. Oletetaan, että yhden komponenttityypin vaihtoon (engl. *feeder change*) kuluu aika t_f ja valitaan hyvä asettelujärjestys $S_1 \Rightarrow S_2 \Rightarrow S_3$. Piirilevyjen kokonaisladonta-aika (t_J) saadaan, kun lasketaan yhteen asettelujen ja ladontatehtävien vaatimat ajat:

$$t_J = \sum_{j=1}^{J} t(S_{j-1} \Rightarrow S_j) + t_j \tag{8}$$

Kaavan (8) summassa ensimmäinen termi kuvaa aikaa, joka kuluu kun siirrytään asettelusta toiseen ja toinen itse levyn ladonnan aikaa. Asettelu S_0:n ajatellaan vastaavan tyhjää konetta. Tällöin saattaa olla hyvä aloittaa asettelulla, johon kuuluu mahdollisimman vähän komponentteja. Levyerän ladonta-aika lasketaan levyjen ja koneiden lukumäärien (N_J ja M) sekä tahtiajan (C) avulla kaavalla:

$$t_j = (N_J + (M - 1)) \cdot C \tag{9}$$

Kun pyritään minimoimaan tahtiajat, esimerkin ladontatyön kokonaisajaksi saadaan:

$$\begin{aligned} t_J &= t(S_0 \Rightarrow S_1) + t_{J1} + t(S_1 \Rightarrow S_2) + t_{J2} + t(S_2 \Rightarrow S_3) + t_{J3} \\ &= 5 \cdot t_f + (100 + (3 - 1)) \cdot C_1 + \\ &\quad 7 \cdot t_f + (100 + (3 - 1)) \cdot C_2 + \\ &\quad 4 \cdot t_f + (100 + (3 - 1)) \cdot C_3 \\ &= 16 \cdot t_f + 4488 \cdot t_c \end{aligned}$$

jossa t_c on yhden komponentin ladonta-aika, ja t_f yhden komponenttityypin vaihtoaika. Komponenttien vaihtoajat on määritetty seuraavasti:

- Koska S_0 on tyhjä, asennetaan aluksi 5 komponenttityyppiä.
- Levyä 2 varten joudutaan lisäämään koneisiin yhteensä jopa 7 komponenttia – lähinnä koska komponentteja siirretään toisiin koneisiin.
- Levyä 3 varten lisätään 4 komponenttia.
- Komponenttien poistamista ei lasketa, ainoastaan komponenttityypin lisäys.

Kun vuorostaan minimoidaan komponenttivaihtojen määrä, tilanne muuttuu seuraavasti. Suoritetaan ensimmäisen levyn asettelu, kuten edellä. Seuraavissa asetteluvaihdoissa jo asennetut komponenttityypit jätetään siihen koneeseen, jossa ne jo ovat, ja vain uudet komponentit sijoitetaan niin, että tahtiaika minimoidaan. Näin ollen, kuten taulukosta 2 nähdään, levyjen 1 ja 2 yhteiset komponentit ovat T_2, T_3 ja T_5. Kun siirrytään levyn 2 ladontaan, nämä komponentit pysyvät vastaavasti koneissa 1, 2 ja 3. LPT-menetelmää (*longest processing time first*) käyttäen loput levyn 2 vaatimat komponentit jaetaan seuraavasti: koneelle 1: T_7, koneelle 2: T_6 ja T_{10} sekä koneelle 3: T_8. Vastaavasti vaihdettaessa levyn 3 ladontaan, koneisiin jätetään komponentit T_2, T_3, T_7, T_8 ja T_{10}, ja lisätään komponentit T_1 ja T_9. Näin saadut optimaaliset asettelut ja niiden mukaiset ladonta-ajat on esitetty taulukossa 4.

Lasketaan nyt piirilevyjen kokonaisladonta-ajat uudelleen.

$$\begin{aligned} t_J &= t(S_0 \Rightarrow S_1) + t_{J1} + t(S_1 \Rightarrow S_2) + t_{J2} + t(S_2 \Rightarrow S_3) + t_{J3} \\ &= 5 \cdot t_f + (100 + (3-1)) \cdot C_1 + \\ &\quad 4 \cdot t_f + (100 + (3-1)) \cdot C_2 + \\ &\quad 2 \cdot t_f + (100 + (3-1)) \cdot C_3 \\ &= 11 \cdot t_f + 4998 \cdot t_c \end{aligned}$$

Taulukosta 5 nähdään näiden kahden tavoitteen ero. Kun pyritään minimoimaan vaihtojen määrä, tahtiajat kasvavat jonkin verran. Vaihtojen määrä sen sijaan putoaa. Kun kokonaisladonta-aika halutaan mahdollisimman pieneksi, ratkaisevaksi tekijäksi nousee yhden komponentin ladonta-ajan ja komponenttityypin vaihtoajan suhde.

Taulukko 4: Levyjen ladontatehtävien jako sekä koneiden työajat, kun asetteluvaihtojen määrä on minimoitu

Levy 1

	Komponentit	Työaika
Kone 1	T_1, T_2	16
Kone 2	T_3, T_4	15
Kone 3	T_5	10

Levy 2

	Komponentit	Työaika
Kone 1	T_2, T_7	8
Kone 2	T_3, T_6, T_{10}	11
Kone 3	T_5, T_8	11

Levy 3

	Komponentit	Työaika
Kone 1	T_2, T_7	22
Kone 2	T_1, T_3, T_{10}	18
Kone 3	T_8, T_9	12

5.1.4 Rajoitukset komponenttien valinnassa

Optimaalisten komponenttiasettelujen toteuttamista saattaa rajoittaa kaksi seikkaa: koneiden kelakapasiteetit ja komponenttien vaatimat suuttimet. Ladontakoneissa on tietty määrä *kelapaikkoja* (engl. *slot*). Yksi kela voi viedä useamman kelapaikan, sillä isommat komponentit vaativat leveämmän kelan.

Ison komponentin ladonta vaatii todennäköisesti myös erilaisen suuttimen verrattuna pienempään komponenttiin. Suuttimien vaihto vie oman aikansa. Jotkin nykyiset koneet osaavat itse vaihtaa suuttimen, mutta käsin vaihdettaessa aikaa saattaa kulua jopa 15–30 minuuttia. Yleisesti ottaen suuttimen vaihtoa pyritään välttämään. Suurempi ongelma suuttimien kohdalla on niiden olemassa olo. Ei voida olettaa, että linja koostuisi samanlaisista tai samanikäisistä koneista. Tällöin kaikissa koneissa ei välttämättä ole suuttimia, joita joidenkin komponenttien ladontaan tarvittaisiin.

Joissakin koneissa on käytössä ns. modulaariset syöttimet, jotka ovat verrattavissa syöttökärryihin. Nämä syöttimet mahdollistavat useamman komponentin yhtäaikaisen asettamisen, mikä nopeuttaa asettelua. Toisaalta tietyn komponentin sitominen tiettyyn moduliin voi rajoittaa modulien käytön joustavuutta.

Taulukko 5: Tahtiajan ja asetteluvaihtojen minimoinnin erot

Tahtiajat

	Min.tahtiajat	Min.vaihdot
Levy 1	16	16
Levy 2	10	11
Levy 3	18	22

Vaihtojen määrät

	Min.tahtiajat	Min.vaihdot
	16	11

Kokonaisladonta-aika

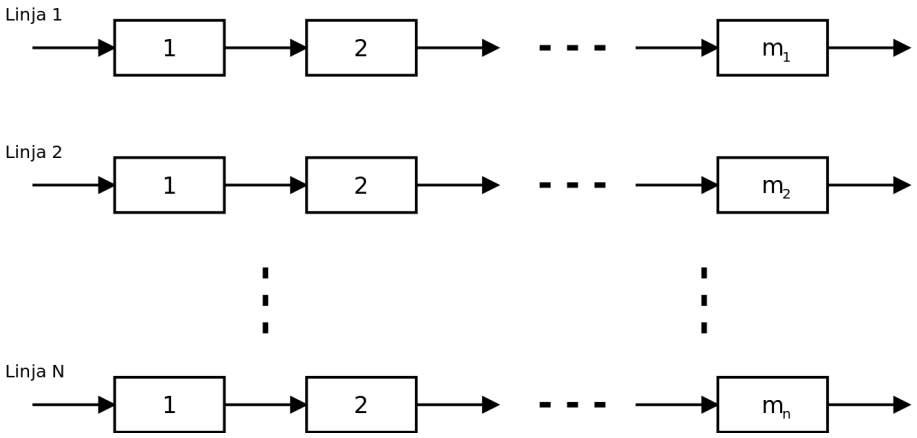
	Min.tahtiajat	Min.vaihdot
	$16 \cdot t_f + 4488 \cdot t_c$	$11 \cdot t_f + 4998 \cdot t_c$

5.2 Monen linjan linjasto

Vaikka monen linjan käyttöönotto saattaa vaikuttaa tuotantoa ja sen suunnittelua helpottavalta ratkaisulta, on usean linjan tehokas käyttö sangen haastavaa. Linjojen tasapainotus- ja aikataulutusongelmaa se mutkistaa merkittävästi. Seuraavassa keskitytään tutkimaan mahdollisimman geneeristä linjastoa, jossa kukin linja voi käsittää eri määrän koneita. Koneilla voi myös olla toisistaan poikkeavat resurssit, kuten kelakapasiteetti, suuttimet ja nopeus. Tällöin linjasto koostuu n:stä linjasta, joista kukin pitää sisällään jonkin määrän (m_i) ladontakoneita (kuva 17).

Kun käytetään useaa linjaa, kannattaa kullekin linjalle sijoittaa keskenään samankaltaisia tuotteita. Tämä tarkoittaa tuotteiden eli piirilevyjen jakamista tuoteperheisiin. Ladottavat piirilevyt voidaan jakaa tuoteperheisiin jo yhden linjan tapauksessa, mutta usean linjan käytössä jaosta hyödytään enemmän.

Aloitetaan ongelman selvittäminen tutkimalla alkutilannetta. Käytettävästä linjastosta tiedetään seuraavat asiat:



Kuva 17: n linjaa, jossa kussakin m_i konetta, missä $i = 1 \ldots n$

- Linjojen määrä n
- Koneiden määrä kussakin linjassa m_i $(i = 1, 2, \ldots, n)$
- Koneen j komponenttikelakapasiteetti C_{ij} linjassa i $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, m_i)$
- Koneen nopeus R_{ij}
- Koneen j suutinkokoelma O_{ijs}, jossa O_{ijs} on 1, jos suutin s löytyy linjan i koneesta j

Suoritettavien ladontatöiden joukosta tiedetään seuraavat asiat:

- Ladontatöiden eli erilaisten ladottavien levytyyppien lukumäärä J
- Kunkin ladontatyön levymäärä N_j
- Työhön j tarvittavien komponenttityyppien joukko T_j
- Vaadittujen komponenttien lukumäärät A_{jt}, joka ilmoittaa montako kappaletta työ j vaatii komponenttityypppiä t

Kustakin komponenttityypistä tiedetään seuraavat asiat:

- Komponenttityypin t koko eli komponenttikelan leveys W_t, joka vaikuttaa konekapasiteetin käyttöön.

- Komponentin t ladontaan vaadittu suutin: S_t

Monet edellä esitetyistä muuttujista voidaan määritellä uudelleen, jotta ne sopivat paremmin ongelman ratkaisemiseen. Kun käytetään matemaattiseen ohjelmointiin sopivia merkintöjä, muuttujista määritellään usein ns. totuusmuuttujat. Nämä totuusmuuttujat ovat matriiseja, joiden alkion arvot ovat 1 tai 0. Esim. komponenttityypin ladontaan vaadittu suutin voidaan esittää matriisina S, jossa S_{ts} on 1, jos komponenttityyppi t vaatii suuttimen s.

Tavoitteena on suorittaa annettujen ladontatöiden joukko mahdollisimman nopeasti eli minimoida ladontatöiden kokonaisaika. Tavoite on jaettavissa osiin. Aikaa kuluu sekä itse ladontatyöhön että komponenttiasettelujen vaihtoon. Ladottavien komponenttien määrä on ennalta määrätty. Näin ollen itse ladonnan kestoon vaikuttavat ainoastaan koneiden mahdolliset nopeuserot. Asettelujen ja suuttimien vaihtoon kuluva aika on merkittävän suuri ja on täten ratkaiseva, kun minimoidaan kokonaisaikaa. Asettelujen vaihtoaika sisältää kunkin komponenttityypin vaihdon ajan sekä itse asettelun vaihtoprosessin vaatiman ajan. Kun asetteluvaihto suoritetaan, joudutaan linja pysäyttämään ja tekemään myös muita aikaa vieviä alustustoimenpiteitä. Näin ollen kannattaa pyrkiä pitämään myös asettelujen vaihtokerrat pieninä.

5.2.1 Ryhmittelymenetelmät tasapainotuksessa

Ryhmittelymenetelmiä ja niihin perustuvia algoritmeja on tutkittu runsaasti [SJP+98], [KHJN02], [YGGY07] ja [SSJN06]. Ryhmittelyn pääajatuksena on jakaa työt ryhmiin niiden keskinäisten samankaltaisuuksien ja käytössä olevien linjojen ominaisuuksien perusteella. Tällöin pyritään siihen, että kussakin ryhmässä piirilevyt vaativat mahdollisimman samankaltaisia komponentteja. Oletetaan lisäksi, että kukin levy esiintyy ainoastaan yhdessä ryhmässä. Eräs tapa on jakaa levyt yhtä moneen ryhmään kuin linjastossa on rinnakkaisia linjoja. Piirilevyjen *ryhmityksessä* (engl. *grouping*) on tarkoitus maksimoida piirilevyjen yhteisten komponenttityyppien määrä kunkin ryhmän sisällä, ts. maksimoida samankaltaisuus. Samankaltaisuuksien lisäksi optimaalisten ryhmien kokoon ja levyihin voivat vaikuttaa myös la-

dottavien levyjen lukumäärät. Levymäärien huomioon ottamiseen palataan myöhemmin.

5.2.2 Samankaltaisuusmitat

Samankaltaisuuden mittaamiseen on kehitetty useita menetelmiä. Yksinkertaisin menetelmä on laskea yhteisten komponenttityyppien lukumäärä [VHN03]. Tämä ei kuitenkaan kerro koko tilannetta. Usein on syytä tietää, kuinka suuri osa komponenttityypeistä on yhteisiä, pelkän lukumäärän sijaan.

Kun halutaan ottaa huomioon myös komponenttityyppien kokonaismäärä, puhutaan *samanlaisuuskertoimista* (engl. *similarity coefficients*). Nämä ovat suhdelukuja, jotka määrittelevät yleisesti ottaen yhteisten komponenttien osuuden jostakin kokonaismäärästä. Näiden suhdelukujen laskemiseen on kuitenkin monia menetelmiä, joiden lausekkeita määriteltäessä käytetään yleisesti taulukon 6 merkintöjä. Luku a ilmaisee levyjen i ja j yhteisten komponenttityyppien määrän ja d puolestaan niiden komponenttien määrän, jotka eivät esiinny kummassakaan. Luvut b ja c vastaavat komponenttimääriä, jotka esiintyvät vastaavasti ainoastaan levyssä i tai j.

Taulukko 6: Yleinen merkintätapa komponenttityyppien esiintymisestä levyissä

		Levy j	
		esiintyy	ei esiinny
Levy i	esiintyy	a	b
	ei esiinny	c	d

Shafer ja Rogers [SR93] antavat mm. seuraavia esimerkkejä suhdelukujen laskentamenetelmistä: *Russell and Rao*, *Simple matching*, *Jaccard*, *Dice* ja *Sokal and Sneath*. Menetelmät eroavat toisistaan painotuksien osalta sekä miten komponenttityyppien lukumäärät suhteutetaan toisiinsa. Osa näistä on esitetty taulukossa 7, ja osaan keskitytään seuraavassa tarkemmin.

Yilmaz, Grunow, Günther ja Yapan [YGGY07] keskittyvät ensin *Jaccardin samanlaisuuskertoimeen* (engl. *Jaccard's similarity coefficient*), jonka

Taulukko 7: Samanlaisuuskertoimien laskentamenetelmiä [SR93]

Russell & Rao	$\frac{a}{a+b+c+d}$
Dice	$\frac{2a}{2a+b+c}$
Sokal & Sneath 1	$\frac{2(a+d)}{2(a+d)+b+c}$
Sokal & Sneath 2	$\frac{a}{a+2(b+c)}$

mainitaan olevan kuuluisin samankaltaisuuden mittauskeino. Jaccardin samanlaisuuskerroin levyjen i ja j välillä määritellään seuraavasti:

$$Sim_J(i,j) = \frac{a}{a+b+c}$$

eli yhteisten komponenttityyppien määrä suhteessa levyjen yhteisesti sisältämien komponenttityyppien määrään.

Jaccardin samanlaisuuskerroin ei kuitenkaan ota huomioon levyjen tuotannossa tarvittavien komponenttityyppien kokonaismäärää, eikä ehkä täten ole paras piirilevyladonnan ryhmien selvittämiseen [YGGY07]. Tästä syystä seuraavassa esitellään piirilevyladonnalle *yksinkertainen samanlaisuuskerroin* (engl. *simple matching coefficient*):

$$Sim_{SM}(i,j) = \frac{a+d}{a+b+c+d}$$

Yksinkertaisen samanlaisuuskertoimen käytön etuna on komponenttityyppien vaihtelun väheneminen ryhmän sisällä [YGGY07].

5.2.3 Piirilevyryhmien muodostaminen

Yilmazin, Grunowin, Güntherin ja Yapanin [YGGY07] mukaan piirilevyladonnassa kannattaa huomioida, kuinka piirilevyjen komponenttityyppien joukot sisältyvät toisiinsa. Tämän vuoksi piirilevyryhmien muodostamiseen

ehdotetaan ns. *sisällyttämismenetelmää*. Sisällyttämismenetelmästä on kehitetty kaksi eri versioita, *erottava* ja *lisäävä*. Erottavan version lähtökohtana on yksi kaikki levyt kattava ryhmä, josta erotetaan ryhmiä niiden erilaisuuksien ja toisiinsa kuulumattomuuden perusteella. Lisäävä versio menetelmästä toimii päin vaistoin eli yhdistää ryhmiä niiden samanlaisuuden ja toisiinsa kuuluvuuden perusteella. Sekä erotuksessa että yhdistetämisessä otetaan huomioon linjojen kapasiteetti- ja suutinrajoitukset. Yilmazin, Grunowin, Güntherin ja Yapanin [YGGY07] mukaan lisäävä sisällytystekniikka antaa parhaan tuloksen, kun mukana on rajoittavia tekijöitä, kuten koneiden kapasiteettirajoituksia.

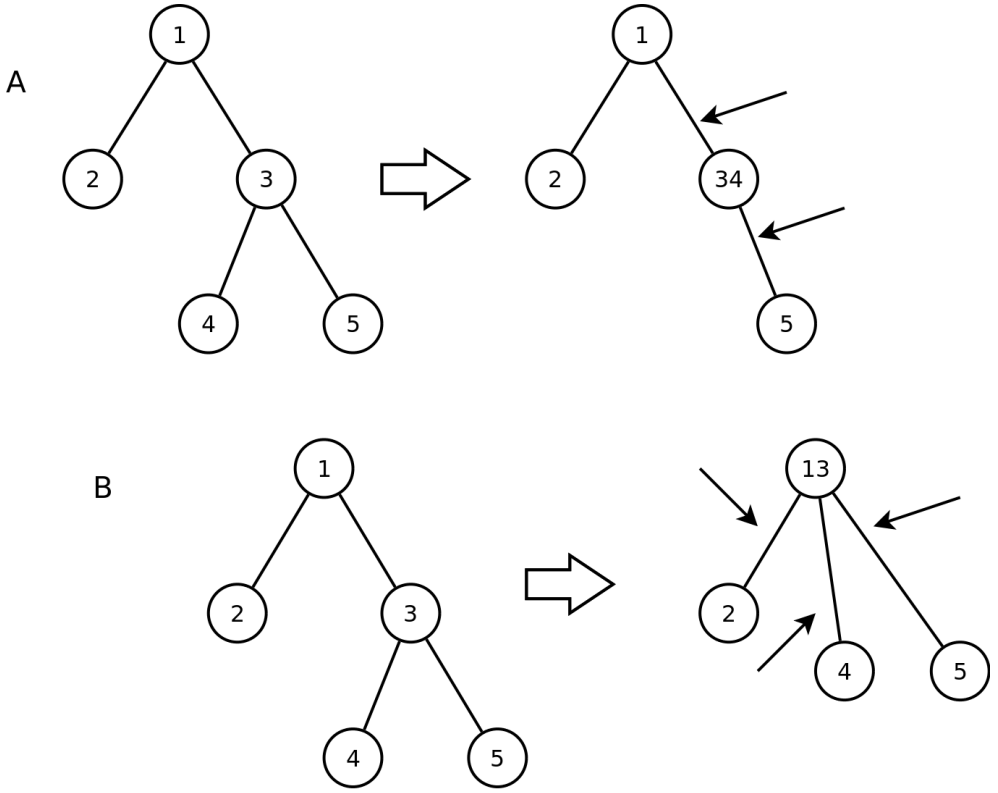
5.2.4 Inkluusiopuun käyttö

Lisäävässä sisällyttämismenetelmässä on tavoitteena rakentaa ns. *inkluusiopuu*, joka osoittaa miten komponenttien tyyppijoukot sisältyvät toisiinsa. Näistä alkujoukoista muodostetaan sitten isompia kokonaisuuksia kokoonpanolinjoja varten. Ensin piirilevyt lajitellaan niiden sisältämien komponenttityyppien määrän perusteella laskevaan järjestykseen. Levy, jolla on eniten komponenttityyppejä, muodostaa puun juurisolmun. Muut levyt liitetään puuhun laskevassa järjestyksessä. Solmu, jonka pojaksi uuden levyn solmu liitetään, valitaan komponenttijoukkojen sisältyvyyden mukaan. Sisältyvyys mitataan ns. *epäsymmetristä sisältyvyysmittaria* (engl. *asymmetric inclusion measure*) käyttäen [YGGY07]. Tämä kertoo, miten komponenttijoukko j sisältyy joukkoon i. Epäsymmetrinen sisältyvyysmittari määritetään seuraavasti:

$$IM(i,j) = \frac{a}{a+c} \quad , \tag{10}$$

jossa a ja c on määritelty taulukon 6 mukaan.

Kun inkluusiopuu on valmis, kukin levy muodostaa aluksi oman ryhmänsä. Seuraavaksi aloitetaan ryhmien yhdistäminen sisältyvyyden perusteella. Tämä tapahtuu yhdistämällä ryhmät, joiden kesken laskettu sisältyvyys on suurin, yhdeksi solmuksi. Näiden kummankin solmun lapsisolmut liitetään juuri muodostettuun solmuun (ks. kuva 18). Yhdistämisen jälkeen välittömät sisältyvyyskertoimet lasketaan uudestaan. Piirilevyryhmien yhdistämi-



Kuva 18: Inkluusiopuun ryhmien yhdistäminen. A: ryhmät 3 ja 4 yhdistetään, B: ryhmät 1 ja 3 yhdistetään. Nuolet osoittavat uudelleen laskettavat sisältyvyyssuhteet.

sessä on otettava huomioon myös rajoitteita. Ensinnäkin kokonaisladontaaika ei saa kasvaa. Toiseksi ladonnan on oltava mahdollinen. Kukin ryhmä on sijoitettava linjalle, jonka koneisiin on mahdollista sijoittaa vaaditut komponentit sekä kapasiteettien että vaadittujen suuttimien osalta. Mikäli ryhmän komponenttityyppien yhteenlasketut koot ylittävät linjan koneiden komponenttikapasiteetin ei yhdistämistä voida tehdä. Jos ryhmän komponenttien vaatima kapasiteetti saavuttaa linjan koneiden kapasiteetin siten, ettei yhtään ryhmää voida lisätä, katsotaan ryhmän olevan valmis ja se poistetaan puusta. Näin jatketaan, kunnes kaikki keskenään yhdistettävät ryhmät on muodostettu.

Inkluusiomenetelmä sellaisenaan ei välttämättä tuota parasta mahdollista ratkaisua, sillä kaikkia seikkoja ei oteta huomioon. On mahdollista, että

menetelmän käyttämä yhdistämisjärjestys ei ole optimaalinen. Näin lopullisten ryhmien väliset kokoerot voivat ovat hyvinkin suuria. Mikäli inkluusiomenetelmä ei tuota riittävän tasapainoista ratkaisua, voidaan kokeilla vaihtaa sisältyvyysfunktio toiseen. Eri funktiot ottavat eri komponenttimäärät huomioon eri tavoin.

Myös itse inkluusiomenetelmälle on kehitetty parannuksia. Smed et. al. esittävät sekä *vaihto-operaation* että *yhdistämisoperaation* [SJP+98]. Vaihto-operaatio käy läpi kaikkien ryhmien työt ja suorittaa vaihtoja, jotka tuottavat paremman ratkaisun. Yhdistämisoperaatio yhdistää ryhmiä esim. komponenttierojen perusteella. Tämän operaation käyttö inkluusiomenettelyn jälkeen tuottaa poikkeuksetta ylisuuria ja sopimattomia ryhmiä. Siksi operaatioon kuuluu, että sopimattomasta ryhmästä poistetaan töitä toisiin ryhmiin, jotta ryhmästä saadaan jälleen kelvollinen.

5.2.5 Esimerkki tasapainotuksesta

Seuraavaksi käsitellään yksityiskohtaisesti sisällytystekniikka käyttämällä edellisen luvun esimerkkiä hieman laajennetussa muodossa, jossa kapasiteetti- ja suutinrajoitukset on otettu mukaan. Taulukko 8 sisältää piirilevyjen sisältämien komponenttityyppien määrät. Levyjen määrä on kasvatettu seitsemään ja komponenttityyppien määrä 15:een. Ladottavien levyjen määrä pysyy samana eli ladottavana on 100 kappaletta kutakin levyä. Asettelu- ja ladontaaikojen laskemista on sen sijaan hieman yksinkertaistettu, jotta ei eksyttäisi itse ryhmittelyideasta.

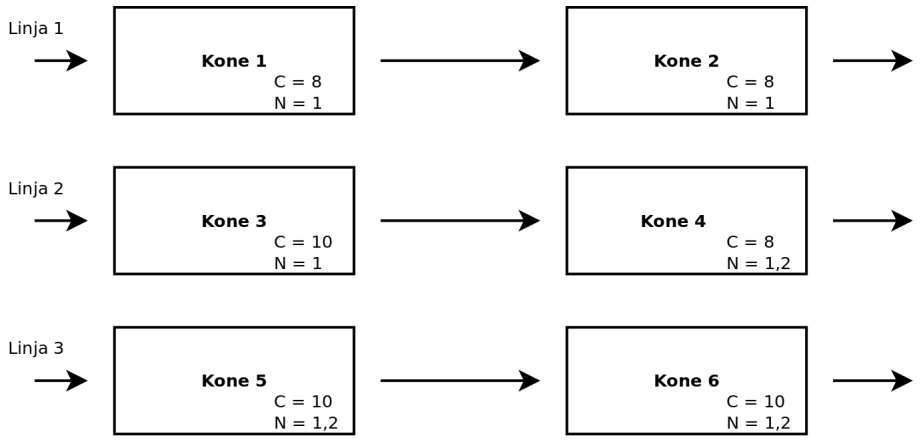
Taulukko 8: Seitsemän esimerkkilevyn vaatimat komponenttimäärät.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}
Levy 1	8	8	15	10	10				4		14	3	13	9	
Levy 2		3	2		1	5	5	10		4					
Levy 3	12	10	2				12	6	6	4					
Levy 4			12	4					16	8		10		4	
Levy 5			16	10					8	12					8
Levy 6	8						4	14	12		9	3	3		2
Levy 7			16	10		4	4	8	8			4			6

Komponentit 1-10 ovat yhden yksikön levyisissä keloissa, komponentit

11-13 ovat kahden yksikön levyisissä keloissa ja komponenttien 14 ja 15 kelat vaativat koneesta kolme kelapaikkaa. Komponenteista ainoastaan tyyppi 15 vaatii suuttimen numero 2, kun muut ovat ladottavissa suuttimella 1.

Kokoonpanolinjastona tässä esimerkissä on kuvan 19 mukainen konfiguraatio. Linjasto koostuu kolmesta linjasta, joissa kussakin on kaksi konetta. Koneiden kapasiteetit komponenttikelalokeroina ilmaistuna ovat linjalla 1: 8 ja 8; linjalla 2: 10 ja 8; linjalla 3: 10 ja 10. Jokaisessa koneessa on suutin numero 1, mutta suutin numero 2 löytyy vain koneista 4, 5 ja 6.



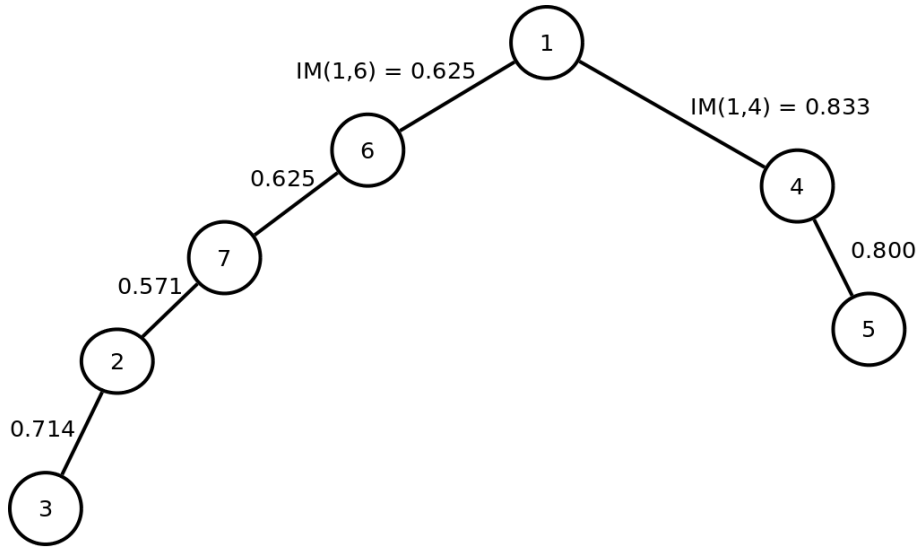
Kuva 19: Linjaston koneiden kapasiteetit komponenttikelalokeroina (C) ja suutin vaihtoehdot (N)

Taulukko 9: Piirilevyt järjestettynä komponenttityyppien määrän mukaan

Levy 1	10
Levy 6	8
Levy 7	8
Levy 2	7
Levy 3	7
Levy 4	6
Levy 5	5

Taulukon 9 mukaan sisällytyspuun juurisolmuksi tulee levy 1, johon levy 6 liitetään. Laskettaessa levyn 7 paikkaa joudutaan laskemaan sen sisälty-

vyys sekä levyyn 1 että 6 kaavan (10) mukaan. Sisältyvyydet ovat vastaavasti 0,500 ja 0,625, minkä perusteella levy 7 sisältyy paremmin levyyn 6. Mikäli parhaaksi liittämiskohdaksi on useita ehdokkaita, valitaan se, missä on vähiten komponentteja. Näin käy kun liitetään levyä 5, jolle vaihtoehtoja ovat sekä 7 että 4. Näistä levy 4 sisältää vähemmän komponenttityyppejä, jolloin se valitaan kohteeksi. Tätä menetelmää jatkaen saadaan lopuksi kuvan 20 mukainen puu.



Kuva 20: Inkluusiopuun rakenne ja kaavan (10) mukaiset sisältymiskertoimet.

Ryhmiä yhdistettäessä valitaan ne kaksi ryhmää, joiden välillä inkluusiopuun mukaan on suurin yhteenkuuluvuus. Mikäli edellä mainitut yhdistämisvaatimukset (eli kokonaisladonta-ajan pieneneminen ja ladonnan mahdollisuus) toteutuvat, puun kaksi solmua yhdistetään yhdeksi. Tällöin on laskettava uuden juuri muodostetun ryhmän yhteenkuuluvuudet ympäröiviin ryhmiin. Jos yhdistämisvaatimukset eivät toteudu, merkitään yhteenkuuluvuudeksi -1, jotta yhdistämistä ei yritettäisi uudestaan. Tätä jatketaan, kunnes yhdistämiset eivät enää ole mahdollisia [YGGY07]. Yhdistämiset voidaan myös lopettaa silloin, kun saadaan toivottu määrä ryhmiä, esim. kullekin linjalle omansa.

Yhdistämisvaatimuksista kelakapasiteettien laskeminen ja suuttimien vaatiminen linjalta on yksinkertaista. On huomattava, että linjaa voidaan pitää varattuna, jos jokin ryhmä on mahdollista latoa ainoastaan kyseisellä linjalla. Tällöin myöhemmin syntyvät ryhmät eivät voi käyttää kyseistä linjaa.

Kokonaisladonta-ajan laskeminen on hankalaa, kun kaikkia levyjä ei ole jaettu ryhmille. Kun kaksi ryhmää yhdistetään, saattaa koko piirilevyjoukkojen jako linjoille muuttua. Koska kyseessä on ladonta-ajan muuttumissuunnan tutkiminen, voidaan laskenta yksinkertaistaa koskemaan vain yhdistettäviä ryhmiä. Tällöin arvioidaan muutosta laskemalla ensin uudelle ryhmälle alkuasettelun sekä ladonnan vaatima aika mahdollisine suutinvaihtoineen. Tätä aikaa verrataan siihen aikaan, mikä kuluisi, jos ryhmien levyt ladottaisiin peräkkäin yksittäisasetteluja käyttäen. On muistettava huomioida myös asetteluista johtuvat kiinteät aikakustannukset eli verrata myös asettelukertojen määrää. Peräkkäin ladottaessa aloitetaan levystä (tai ryhmästä), johon kuuluu enemmän komponenttityyppejä, eli ts. on inkluusiopuussa ylempänä. Vertailtavaksi ajaksi otetaan lyhin aika, ts. nopeimmin tehtävistä suoriutuvan linjan aika. Tämä vertailumenetelmä ei ole paras mahdollinen, mutta se voi estää esim. hitaan suuttimen vaihdon.

Kuvasta 20 havaitaan, että ensimmäisenä yritetään muodostaa levyistä 1 ja 4 uusi ryhmä. Merkitään uutta ryhmää R_{14}. Uuden ryhmän komponentit vaativat yhteensä 16 kelapaikkaa. Näin ollen ryhmä on ladottavissa kaikilla kolmella linjalla, koska kapasiteetitrajoitukset eivät ylity ja suutinta numero 2 ei tarvita.

Ladonta-aikaa arvioitaessa suoritetaan ensin asettelu ja ladonta levylle, jossa on enemmän komponenttityyppejä ($t_s + 10 \cdot t_f + 94 \cdot 100 \cdot t_c$). Tässä t_s on itse asettelukerran vaatima aika, t_f on yhden komponenttityypin vaihtoaika ja t_c yhden komponentin ladonta-aika. Sitten lisätään tarvittavat komponentit toista levyä varten sekä ladotaan komponentit ($t_s + 1 \cdot t_f + 54 \cdot 100 \cdot t_c$). Näin saatua kokonaisaikaa ($2 \cdot t_s + 11 \cdot t_f + 148 \cdot 100 \cdot t_c$) verrataan ryhmän R_{14} vaatimaan aikaan ($t_s + 11 \cdot t_f + 148 \cdot 100 \cdot t_c$). Tässä on tilannetta yksinkertaistettu siten, että lasketaan ainoastaan lisättävistä komponenttityypeistä tuleva aika, jolloin ainoaksi eroksi tulee asettelukertojen määrä. Tilanne siis yksinkertaistuu lopulta asettelukertojen minimointiin, jolloin ryhmien yhdis-

täminen on aina ajallisesti kannattavaa.

Kun lasketaan uuden ryhmän kuuluvuudet ryhmiin 6 ja 5, saadaan arvot $0,625$ ja $0,800$, joiden mukaan seuraavaksi yritetään liittää R_{14} ryhmään 5. Näin saadun ryhmän R_{145} levyjen komponentit vaativat yhteensä 19 kelapaikkaa. Ottaen huomioon myös suuttimen 2 tarpeellisuuden (komponenttityypppi 15) on ryhmän kolme levyä mahdollista latoa vain linjalla 3.

Näin jatkaen saadaan lopullisiksi ryhmiksi R_{145}, R_{23} ja R_{67}. Näiden ryhmien ominaisuudet on esitetty taulukossa 10, jossa on mainittu ryhmän sisältämien komponenttien kokonaismäärä, vaadittu kapasiteetti linjan koneilta sekä linjat, joilla kyseisen ryhmän työt voidaan suorittaa. Näistä ominaisuuksista huomataan, että jako on melko epätasainen komponenttimäärien suhteen. Tämä tarkoittaa, että ryhmän R_{145} ladonta kestää pisimpään. Komponenttimäärän lisäksi kyseisen ryhmän ladontaa hidastaa komponentin 15 vaatima suutin, mikä joudutaan vaihtamaan kesken ladonnan. Tämä suuttimen vaihto hidastaa myös ryhmän R_{67} ladontaa.

Taulukko 10: Inkluusiomenetelmällä saatujen ryhmien ominaisuuksia (epäsymmetrinen ja Jaccardin sisältyvyys)

Ryhmä	Komp. määrä	Kapasiteetti	Mahd. linjat
R_{145}	202	19	3
R_{23}	82	9	1,2,3
R_{67}	115	16	2,3

Taulukko 11: Inkluusiomenetelmällä saatujen ryhmien ominaisuuksia (yksinkertainen sisältyvyys)

Ryhmä	Komp. määrä	Kapasiteetti	Mahd. linjat
R_1	94	15	1,2,3
R_6	55	13	2,3
R_{23457}	197	19	3

Mikäli inkluusiopuun rakentamisessa ja ryhmien yhdistämisessä käytetään Jaccardin sisältyvyyttä (s. 52), saadaan sama tulos. Sen sijaan käytet-

täessa yksinkertaista sisältyvyyttä (s. 52), konekapasiteetit tulevat tasaisemmin käytetyiksi, kuten nähdään taulukosta 11. Ladottavien komponenttien määrässä on silti suuria eroja linjojen välillä. On helppo havaita, että ryhmästä R_{23457} voisi yrittää siirtää töitä toisiin ryhmiin, jotka sisältävät vain yhden työn kukin. Näin ollen edellä mainittujen vaihto-operaatioiden yrittäminen lienee kannattavaa.

Tutkitaan epäsymmetrisellä menetelmällä suoritettua ryhmäjakoa (taulukko 10) ja yritetään tasoittaa ryhmiä arvioimalla ladonta-aikoja. Nykyiset ladonta-ajat koostuvat asettelukerrasta (t_s), yksittäisten komponenttien asettelusta (t_f), mahdollisesta suuttimen vaihdosta (t_n) sekä komponenttien ladonta-ajasta (t_c). Oletetaan, että suuttimen vaihto on optimoitu siten, että vaihto tapahtuu vain kerran kerran ladottavaa levyä kohden. Ts. ladonta aloitetaan sillä suuttimella, mikä koneeseen edellisen levyn jälkeen jäi. Lisäksi oletetaan, että kumpikin linjan koneista latoo puolet komponenteista, ja että linjalla 3 riittää, kun suutinta vaihdetaan ainoastaan toisessa koneessa.

Jotta nämä ajat olisivat vertailukelpoisia, on tiedettävä ainakin niiden keskinäiset suhteet. Koneiden ladontanopeudet ovat kymmeniä tuhansia komponentteja tunnissa. Suuttimen vaihto, ainakin käsin tehtynä, on puolestaan hidasta.

Arvioidaan näitä aikoja suoraan: $t_c = 10ms$, $t_s = 60s$, $t_f = 15s$, $t_n = 120s$. Näin saadaan alustaville ryhmille seuraavat aika-arviot:

$$\begin{aligned} R_{145}\text{:} &\quad t_s + 12 \cdot t_f + 10100 \cdot t_c + 100 \cdot t_n &&= 13250s \\ R_{23}\text{:} &\quad t_s + 9 \cdot t_f + 4100 \cdot t_c &&= 605s \\ R_{67}\text{:} &\quad t_s + 11 \cdot t_f + 5750 \cdot t_c + 100 \cdot t_n &&= 12800s \end{aligned}$$

Vaikka suuttimen vaihtoaika on arvioitu optimistisesti, sen vaikutus ladonta-aikoihin on silti erittäin suuri. Aikoja tasatessa on kaksi vaihtoehtoa. Joko suuttimen vaihdon vaatimat levyt (5, 6 ja 7) yritetään sijoittaa tasaisesti vaihtoon pystyville linjoille tai ne yritetään pitää yhdessä ryhmässä.

Ensimmäisessä tapauksessa ryhmästä R_{145} yritetään siirtää levyt 1 ja 4 ryhmään R_{23}. Koska uusi ryhmä, johon levy siirretään, on tarkoitus ajaa linjalla 1 (ei tarvetta toiselle suuttimelle), on käytössä oleva kelakapasiteetti 16 yksikköä. Tämä estää levyn 1 liittämisen ryhmään. Ainoastaan levy 4 voi-

daan liittää mukaan, jolloin vaadittu kelakapasiteetti on 15. Tämän vaihdon vaikutus on nähtävissä taulukon 12 ensimmäisessä osassa.

Taulukko 12: Yhden levyn vaihdon vaikutukset ladonta-aikoihin

1:	R_{15}	$t_s + 12 \cdot t_f + 7400 \cdot t_c + t_n$	$= 13085s$
	R_{234}	$t_s + 12 \cdot t_f + 6800 \cdot t_c$	$= 920s$
	R_{67}	$t_s + 11 \cdot t_f + 5750 \cdot t_c + t_n$	$= 12800s$
2:	R_{14}	$t_s + 11 \cdot t_f + 7400 \cdot t_c$	$= 965s$
	R_{23}	$t_s + 9 \cdot t_f + 4100 \cdot t_c$	$= 605s$
	R_{567}	$t_s + 12 \cdot t_f + 8450 \cdot t_c + t_n$	$= 13085s$

Jälkimmäinen vaihtoehto on yrittää yhdistää kaikki levyt, joiden ladonta vaatii suuttimen vaihtoa. Näin ladonnan hitaus pystytään keskittämään yhdelle linjalle. Suurin hyöty tästä saadaan, jos ilman suutinvaihtoja ladottavia levyjä on paljon. Tässä esimerkissä ryhmä R_{567} on mahdollinen, sillä sen vaatima kelakapasiteetti on 17, jolloin se on ladottavissa linjoilla 2 ja 3. Näin muille linjoille jäisi ryhmät R_{14} ja R_{23}. Tämän vaihdon vaikutus näkyy taulukon 12 jälkimmäisessä osassa.

Näin yksinkertaistetussa esimerkissä ei kyetä helposti jakamaan suuria ladonta-aikoja. Huomion arvoista on kuitenkin, että linjaston suurin ladonta-aika ei juurikaan muutu levyvaihtojen seurauksena. Alkuperäisestä yli 3,5 tunnin arviosta putoaa pois vain hieman alle 3 minuuttia. Tässä esimerkissä levyjen ryhmien välisten vaihtojen vaikutus on siis olematon, mutta käytännön tilanteissa ne saattavat olla merkittäviä. Kuten tämä esimerkki osoitti, voidaan suuret ladonta-ajat keskittää yhdelle linjalle, jolloin muut linjat suoriutuvat töistään nopeammin ja mahdollistavat näin uusien töiden aloittamisen aikaisemmin. Tämänkaltainen tulos on nähtävissä taulukon 12 jälkimmäisessä osassa.

5.2.6 Matemaattinen malli

Edellä esitetyn heuristisen menetelmän lisäksi ongelmaa voi lähestyä matemaattisen esitystavan kautta. Ratkaisuvaihtoehtoina ovat tällöin matemaat-

tinen ohjelmointi, tarkat optimointimenetelmät ja rajoiteohjelmointi. Seuraavassa pyritään laatimaan ongelmasta tarkka malli matemaattisesti ja siten tarkastella, mitä seikkoja on syytä ottaa huomioon tämän kaltaisia ratkaisumenetelmiä käyttäessä.

Kehitetään joukko totuusmuuttujia, joiden avulla voidaan asioita ja suhteita ilmaista binäärisesti, arvojen 0 ja 1 avulla. Tilanteet, joissa normaalisti käytettäisiin ehtolauseita, voidaan nyt esittää kerto- ja yhteenlaskujen avulla tai muodostamalla yhtälöryhmiä. Näiden muuttujien lisäksi käytetään tavallisempia muuttujia, jotka esittävät mm. komponenttien lukumääriä, ladontakoneiden kelakapasiteetteja sekä erilaisia aikamääreitä. Muuttujien ja indeksien merkinnässä käytetään taulukon 13 mukaisia merkintöjä sekä tässä että seuraavassa luvussa. Merkintätapa on pääosin itse kehitetty, mutta noudattaa artikkeleiden [VHN03] ja [CWR+07] merkintätapoja mm. indekseille ja lukumäärille. Kaikki muuttujat on kerätty samaan taulukkoon, vaikka joihinkin viitataan vasta seuraavissa luvuissa.

Taulukko 13: Luettelo käytetyistä symboleista

Indeksit	
g	piirilevyryhmä
j	ladontatyö/levy
k	komponenttityyppi
m	ladontakone
n	linja
o	suutin

Vektorit ja matriisit

A_{jk} Komponenttien vaatimusmatriisi. Alkio $A_{jk} = 1$, jos levy j vaatii komponenttityypin k, muutoin 0.

B_{jg} Ryhmään kuulumisen esittävä matriisi. Alkio $B_{jg} = 1$, jos levy j kuuluu ryhmään g, muutoin 0.

C_{nm} Kelakapasiteettimatriisi. Alkio C_{nm} ilmaisee linjan n koneen m kelakapasiteetin.

I_{jk} Komponenttityyppien inkluusiomatriisi. Alkio I_{jk} ilmaisee, kuinka monta tyypin k komponenttia levyyn j on ladottava.

L_{gn} Levyryhmien sijoittelumatriisi. Alkio $L_{gn} = 1$, jos ryhmä g on sijoitettu linjalle n, muutoin 0.

O_{nmo} Suuttimien saatavuusmatriisi. Tämän kolmiulotteisen matriisin alkio $O_{nmo} = 1$, jos suutin numero o löytyy linjan n koneesta m, muutoin 0.

P_{knm} Ladontamatriisi, joka kertoo kuinka monta kappaletta linjan n kone m latoo komponenttityyppiä k yhdelle levylle.

Q_{nm} Ladontapäiden määrä linjan n koneessa m. Tässä työssä keskitytään ainoastaan yhden ladontapään koneisiin eli Q-matriisia ei käytetä.

R_{ko} Suuttimien vaatimusmatriisi. Alkio $R_{ko} = 1$, mikäli komponenttityypin k ladontaan vaaditaan suutin numero o, muutoin 0.

S_{knm} Asettelumatriisi. $S_{knm} = 1$, jos komponentti k on asetettu linjan n koneeseen m.

V_{nmo} Suutinten sopivuusmatriisi. $V_{om} = 1$, jos suutin o voidaan asentaa linjan n koneeseen m. Jatkossa oletetaan, että tämä sisältyy matriisiin O, joten myöskään tätä matriisia ei käytetä.

W_k Komponenttityyppien koon ilmaiseva vektori. W_k ilmaisee komponenttityypin k vaatiman kelan leveyden.

Päätösmuuttujat

X_{kn} Komponentin vaatimusmatriisi. Alkio $X_{kn} = 1$, jos linjalla n vaaditaan komponenttityyppi k, muutoin 0.

Y_{nmo} Suuttimen vaatimusmuuttuja. $Y_{nmo} = 1$, jos linjan n koneeseen m on vaihdettava suutin o, muulloin 0.

Lukumäärät ja ajat

B	Työhön kuuluvien levyjen määrä.
G	Työryhmien lukumäärä.
J	Töiden lukumäärä eli ladottavan levyn eräkoko.
K	Komponenttityyppien lukumäärä.
M_n	Koneiden määrä linjalla n.
N	Linjojen määrä.
t_c	Yhden komponentin ladontaan kuluva aika (riippuu koneesta).
t_{Cn}	Linjan n tahtiaika.
t_o	Suuttimen vaihtoon kuluva aika (jos kone vaihtaa).
t_{setup}	Asettelun viemä kokonaisaika.

Lineaarisia lausekkeita sisältävät matemaattiset optimointiongelmat ovat ratkaistavissa olemassa olevilla optimointiohjelmistoilla. Tästä syystä tuotantolinjojen tasapainotusongelma on syytä pyrkiä muuttamaan lineaariseksi systeemiksi. Esimerkiksi kapasiteettirajoitteet linjoille voidaan ilmaista summina. Aloitetaan asettamalla linjalle n seuraava rajoite, jonka mukaan ryhmään kuuluvien komponenttityyppien kokojen summa ei saa ylittää linjan koneiden kapasiteettien summaa.

$$\sum_{k \in G_n} W_k \leq \sum_{m=1}^{M_n} C_{nm}$$

Epäyhtälön vasemmalla puolella on linjan n töiden ryhmään G_n kuuluvien komponenttien k kelojen leveyksien summa ja oikealla puolella linjan koneiden kelakapasiteettien summa.

Siirrytään 0/1-päätösmuuttujien käyttöön. Määritellään ensin uusi komponenttivaatimukset ilmaiseva muuttuja seuraavasti:

$$X: \quad X_{kn} = \begin{cases} 1, & \text{kun } \sum_{g}^{G} \sum_{j}^{J} L_{gn} B_{jg} A_{jk} \geq 0 \\ 0, & \text{muutoin} \end{cases}$$

Näin saadaan varmistettua, että kaikki töiden komponentit on jaettu, ja

kapasiteetit ovat riittävät. Kun käytetään edellä esitettyjä 0/1-päätösmuuttujia, saadaan jokaiselle linjalle n toteutettava yhtälö:

$$\sum_{k=1}^{K}(W_k \cdot X_{kn}) \leq \sum_{m}^{M_n} C_{nm} \quad ,\forall n$$

Tässä komponenttivaatimus kerrotaan kyseisen komponentin kelan leveydellä ja verrataan näin saatua summaa linjan koneiden yhteenlaskettuun kapasiteettiin.

On kuitenkin huomattava, että edellä esitetty rajoite ei ole riittävän tiukka. Pelkkien summien laskenta ei riitä, kun on kyse kelakapasiteettien käytöstä. On nimittäin mahdollista, että leveiden kelojen kokojen summa täsmää linjan koneiden kapasiteettien summaan, mutta asettelu on silti mahdoton. Kuvitellaan tilanne, jossa linjassa on kolme konetta, joiden kunkin kapasiteetti on 10 kelayksikköä. Mikäli ladottavia komponenttityyppejä on 10 ja kukin niistä toimitetaan 3 yksikköä leveässä kelassa, yllä oleva yhtälö toteutuu. Kuitenkin kuhunkin koneeseen mahtuu 3 yksikon keloja vain 3. Näin ainoastaan 9 tämän kokoista komponenttia voidaan asettaa linjan koneisiin.

Jotta varmistetaan, että asettelu on mahdollinen, otettiin käyttöön asettelumatriisi S_{knm}. Matriisin alkion arvo on 1 jos komponentti k on asetettu linjan n koneeseen m. Muutoin arvo on 0. Tätä matriisia käyttäen voidaan käyttää selvittämään, onko asettelu mahdollinen, eli onko valituille komponenteille tilaa koneissa. Seuraavalla kaavalla lasketaan tiettyyn koneeseen asetettujen komponenttien kelojen leveydet yhteen ja vaaditaan, että summa on korkeintaan koneen kelakapasiteetin suuruinen.

$$\sum_{k=1}^{K}(S_{knm} \cdot W_k) \leq C_{nm} \quad ,\forall m$$

Matriisia S käyttäen voidaan myös varmistua, että kaikki tarvittavat komponentit löytyvät linjan koneista. Aloitetaan yhden linjan tapauksesta. Kukin linjalle määrättyjen töiden vaatimista komponenteista on löydyttävä vähintään yhdestä linjan koneesta. Toisin sanoen kunkin komponentin esiintymistaajuus linjan koneiden asetteluissa on oltava vähintään yhtä suuri kuin

komponentin vaadittu määrä, joka on 1 tai 0 riippuen komponentin tarpeellisuudesta. Näin saadaan seuraava yhtälö kullekin linjalle:

$$X_{kn} \leq \sum_{m=1}^{M_n} S_{knm} \quad , \forall k$$

Kun edellä mainitut rajoitteet ovat voimassa yhtäaikaa, on mahdollista asettaa tarvittavat komponentit linjan koneisiin ja suorittaa tarvittavat ladontatyöt.

Edellä keskityttiin komponentteihin liittyviin rajoitteisiin. Suuttimille saadaan vastaavat rajoitteet mm. korvaamalla komponenttien tarpeen ilmaiseva matriisi A suuttimien tarpeen ilmaisevalla matriisilla R. Samoin asettelumatriisin S lisäksi määritellään suuttimien saatavuusmatriisilla O. Suuttimien osalta voidaan esittää seuraava rajoite linjalle n sijoitettujen töiden komponenteille k:

$$\sum_{m=1}^{M} \sum_{o=1}^{O} R_{ko} O_{nmo} \geq 1 \quad , \forall k$$

Tässä siis vaaditaan että ainakin yhdestä linjan n koneesta löytyy haluttu suutin o. Tällöin on mahdollista käyttää linjaa haluttuun työhön.

Ottamalla mukaan asettelumatriisi S voidaan varmistaa, että kuhunkin koneeseen on asennettu komponenttien vaatimat suuttimet. Seuraava rajoite vaatii, että ladontaan vaadittava suutin löytyy koneesta. Epäyhtälön vasemman puolen arvo on 0 jos komponentti ei kuulu asetteluun tai komponentti ei vaadi kyseistä suutinta.

$$S_{knm} R_{to} \leq O_{nmo} \quad , \forall o, k$$

Kaikkia edellämainittuja rajoitteita käyttämällä voidaan varmistaa, että

- linjaa voidaan käyttää sille osoitettujen töiden ladontaan
- linjalle osoitettujen töiden vaatimat komponentit on mahdollista asettaa koneisiin
- koneissa on vaaditut suuttimet komponenttien ladontaan.

Lopullisena tavoitteena on, kuten aiemminkin, minimoida kokonaisladontaaika. Tämä koostuu kunkin linjan asettelu- ja työaikojen summasta. Linjan n työajan määrää suurelta osin linjan tahtiaika (t_{Cn}) ja linjalla ladottavien levyjen määrä (B_j). Yhden ladontatyön vaatimaa kokonaisaikaa voidaan kuvata kaavalla

$$t_j = t_{setup} + t_{Cn} \cdot (B_j + M_n - 1)$$

Linjan tahtiaika (t_{Cn})määräytyy sen hitaimman koneen mukaan. Koneen käyttämään työaikaan vaikuttavat sillä ladottavien komponenttien määrä sekä suuttimen vaihdot, mikäli sellaiset ovat tarpeen. Äärimmäisen yksinkertaistettuna minimoitava linjan tahtiaika voidaan ilmaista seuraavasti:

$$t_{Cn} = max(\sum t_c + \sum t_o), \quad \forall m \tag{11}$$

Otetaan käyttöön matriisi P, joka on laskettavissa matriisien I ja X avulla. Tästä matriisista ilmenee ladottavien komponenttien määrä konekohtaisesti. On huomattava, että tämän matriisin sisältö vaihtelee ladottavan levyn tai levyryhmän mukaan. Tätä matriisia voidaan käyttää laskettaessa koneen ladontaan kuluvaa työaikaa, jolloin kerrotaan yhden komponentin ladontaan kuluva aika ladottavien komponenttien lukumäärällä:

$$\sum t_c \implies t_c \cdot \sum_{k=1}^{K} P_{knm}$$

Suuttimien vaihdot voidaan laskea käyttämällä matriiseja S ja R. Seuraavassa summataan kutakin suutinta käyttävät ja asettelusta löytyvät komponentit. Määritetään 0/1-päätösmuuttuja Y:

$$Y: \quad Y_{nmo} = \begin{cases} 1, & \text{kun } \sum_{k=1}^{K}(S_{knm} \cdot R_{ko}) \geq 0 \\ 0, & \text{muutoin} \end{cases}$$

ja näin saadaan:

$$\sum t_o \implies t_o \cdot \sum_{o=1}^{O} Y_{nmo})$$

Kun komponenttien ladonnan sekä suuttimien vaihtojen ajat yhdistetään saadaan linjan n minimoitava tahtiaika muutettua kaavasta (11) kaavan (12) muotoon.

$$t_{Cn} = max \left[t_c \cdot \sum_{k=1}^{K} P_{knm} + t_o \cdot \sum_{o=1}^{O} Y_{nmo} \right] \quad , m = 1, \ldots, M \qquad (12)$$

5.3 Tuotteiden sijoittelut linjoille

Tuotantolinja voidaan varata tuotteelle, jota valmistetaan moninkertainen määrä muihin tuotteisiin verrattuna. Tällöin puhutaan massatuotannosta. Kun jokin ladontalinja on varattu tuotteelle, ei asetteluvaihtoja tarvitse tehdä alkuasettelun jälkeen.

Massatuotannon lisäksi linjan poikkeuksellisen käytön syynä voivat olla lukuisat pienet työt. Mikäli ladontatöiden joukossa on paljon pieniä töitä, nämä voivat aiheuttaa epätasaisia tahtiaikoja linjoille. Kun pienet työt sijoitetaan tietylle linjalle, voidaan muilla linjoilla suorittaa normaaleja ladontatöitä ilman, että lyhyt ladontatyö häiritsee töiden jakoa. Tällaisesta linjasta käytetään nimitystä *prototyyppi-linja* ja sellaista käytetään usein pienten prototyyppierien ladontaan.

Tässä tutkielmassa ei oteta kantaa, milloin tuotteelle kannattaa osoittaa oma linjansa. Sen sijaan seuraavassa pyritään määrittämään, millainen linja kannattaa varata tuotteelle.

Tuotannon nopeus perustuu suurelta osin asetteluvaihtojen minimointiin. Massatuotannossa asetteluvaihtoja kesken jonkin piirilevyn tuotantoerän ei voida sallia, sillä tämä tarkoittaisi tuotteiden käsittelemistä kahteen (tai jopa useampaan) kertaan. Myöskään resurssien tuhlaaminen ei ole järkevää. Massatuotannossa varattavaksi linjaksi kannattaa valita linja, joka suoriutuu juuri ja juuri kyseisen tuotteen ladonnasta.

Resurssien käyttöä voidaan mitata sekä kelakapasiteetin että käytettyjen suuttimien avulla. Mitä pienemmäksi vapaa kelakapasiteetti jää asettelun jälkeen, sitä vähemmän resursseja tuhlataan. Sama pätee käyttämättö-

miin suuttimiin. Koneiden nopeus linjalla on puolestaan prioriteettikysymys. Mikäli vaaditaan mahdollisimman nopeaa massatuotantoa, kannattaa myös koneiden nopeuksiin kiinnittää huomiota. Muutoin kannattaa linjat valita siten, että muiden töiden vaatimille asetteluille jää mahdollisimman joustavat mahdollisuudet.

Prototyyppilinjalla suoritetaan paljon asettelun vaihtoja, jolloin koneiden ladontanopeudet eivät hallitse kokonaisladonta-aikaa. Tällöin tämän linjan koneiksi riittävät hitaammatkin koneet. Pääpaino näillä koneilla on monipuolisuudessa. Niiden on yhdessä pystyttävä latomaan kaikkia mahdollisia komponentteja, jotta linja suoriutuisi kaikista mahdollisista prototyyppilevyistä.

Koska tämänkaltaisen erillistuotannon vaatimukset ovat yksinkertaiset verrattuna piirilevyjen ryhmittelyyn sekä asetteluaikojen minimointiin nähden, kannattaa näille omistettavien linjojen valinta suorittaa tuotannon suunnittelun aikaisessa vaiheessa. Kun erillistuotantolinjat on valittu, suoritetaan edellisessä luvussa käsitelty linjojen tasapainotus ryhmittelyn avulla. Mikäli linjojen tasapainotus ei tuota tulosta, on mahdollista, että erillistuotantoon on valittu väärät linjat. On myös mahdollista, että erillistuotantoon pyritään valitsemaan liian monta linjaa. Jos esim. massatuotettavia levyjä on liian monta linjojen lukumäärään nähden, muut ladontatyöt kärsivät. Näin jäljelle jäävien linjojen koneiden kelakapasiteetit eivät riitä sisältämään tarvittavia komponentteja ja kaikki ladontatyöt eivät olisi tehtävissä yhdellä linjaston läpiviennillä.

5.4 Aloitus- ja määräaikojen noudattamisesta

Töiden aloitus- ja määräajat tuovat lisähaastetta töidenjärjestelyongelmaan. Töillä on lähes poikkeuksetta *valmistumismääräaika* (engl. *deadline*), jolloin sen odotetaan olevan valmis. Lisäksi on mahdollista, että töitä ei voida aloittaa ennen tiettyä hetkeä. Näin voi käydä, jos tarvittavat levyt tai komponentit saapuvat vasta tiettynä ajankohtana. Tätä aikaa kutsutaan *julkaisuajaksi* (engl. *release time*).

Näiden vaikutukset matemaattisesti määriteltyihin rajoitteisiin ei ole suu-

ri. Rajoitteita tulee lisää aloitusaikojen muodossa ja minimoitavaksi funktioksi vaihtuu töiden myöhästymisten minimointi. Tällöin eräs mahdollisuus on minimoida myöhästymisten summafunktiota. Tästä funktiosta on kaksi versiota sen mukaan, hyödytäänkö ajoissa olemisesta eli negatiivisesta myöhästymisestä, vai ei. Nämä versiot ovat *lateness* ja *tardiness*, joihin palataan myöhemmin tässä työssä.

$$\text{tardiness: } \sum_{j=1}^{J} max(0, d_j - f(j)) \quad , \quad \text{lateness: } \sum_{j=1}^{J} d_j - f(j)$$

Matemaattisen ohjelmoinnin, rajoiteohjelmoinnin tai geneettisten algoritmien yhteydessä nämä eivät vaikuta suuresti muuhun kuin algoritmien suoritusten kestoaikaan.

Heuristisissa menetelmissä aloitus- ja määräaikojen noudattaminen muodostaa suurempia ongelmia. Aiemmin pyrittiin jakamaan työt linjoille komponenttivaihtojen minimoimiseksi. Työt, jotka on suoritettava nopeasti, kannattaa myös jakaa rinnakkaisille linjoille. Nämä kaksi tavoitetta saattavat olla ristiriidassa keskenään. Liiallinen keskittyminen töiden jakoon komponenteittain saattaa johtaa pitkiin myöhästymisiin. Toisaalta myöhästymisten liiallinen estäminen saattaa johtaa toisten töiden kasvaviin myöhästymisiin.

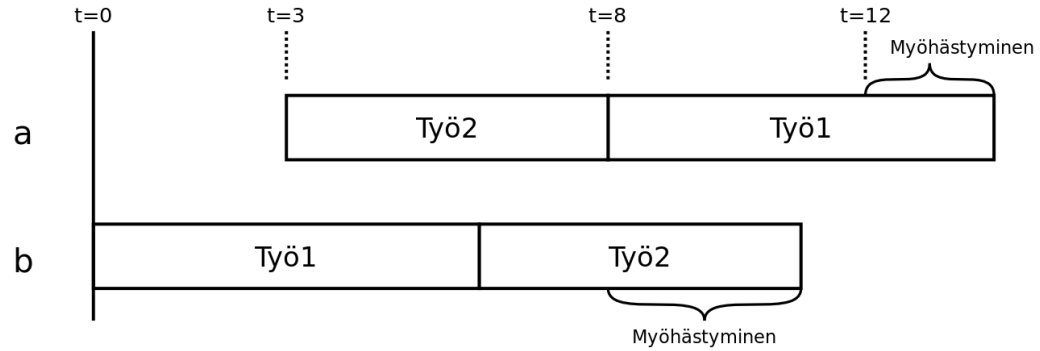
Tässä on ratkaistavana kaksi ongelmaa samanaikaisesti: töiden nopea suoritus oikealla työnjaolla sekä määräaikojen noudattaminen. Suoraviivainen keino kahden ongelman ratkaisemiseen on vuorottelu [Saw02]. Muodostetaan tai keksitään toiselle ongelmalle jokin ratkaisu, minkä jälkeen ratkaistaan toinen ongelma. Siihen, kumpi ratkaistaan ensin, ei ole selkeää vastausta. Molemmat vaihtoehdot on syytä kokeilla ja valita parempi.

On myös mahdollista jatkaa näiden ongelmien ratkomista vuorotellen. Tämä menetelmä ei välttämättä vie ratkaisua aina parempaan suuntaan. Ei myöskään voida olettaa, että ratkaisuiteraatioiden tulokset lähestyisivät jotakin tiettyä ratkaisua.

Eräs tehokas tapa tämän kaltaisten usean ongelman ratkaisemiseen (engl. *multiple criteria decision making*) on *sakotuskäytäntö* (engl. *cost penalty method*). Tässä menetelmässä kaikki ei-toivotut tilanteet pisteytetään eli niille

määrätään ns. sakko. Esimerkiksi jokaisesta myöhästymisminuutista voidaan antaa työlle muutama piste sakkoa. Tärkeille töille voidaan määrätä isompi sakko. Tavoitteeksi jää tällöin minimoida saadun sakon määrä.

Jotta töidenjärjestely käyttäisi kaikkia linjoja mahdollisimman tasaisesti, voidaan sakkoa määrätä myös ladontakoneiden käyttämättömyydestä. Tällöin tästä määrättävä sakko on yleensä huomattavasti pienempi verrattuna myöhästymissakkoon. Näin toimimalla voidaan antaa koneiden olla jopa käyttämättömänä hetken, jotta tärkeä työ saadaan nopeasti valmistettavaksi ja tehdyksi. Tämänkaltainen tilanne on esitetty kuvassa 21a, jossa vaihtoehtona on koneiden jatkuva käytössä pitäminen (21b).



Kuva 21: Erot töiden myöhästymisissä, kun tärkeää työtä odotetaan (a) tai pyritään pitämään koneet varattuina (b).

Kuvasta huomataan, että lyhyehkö odottaminen saattaa pienentää myöhästymisiä. Oletetaan, että tärkeä työ (*Työ2*) voidaan aloittaa hetkellä $t = 3$ ja sen on oltava valmis heti kun mahdollista. Koska työn kesto on 5 aikayksikköä, saadaan tällöin määräajaksi $d_2 = 8$. *Työ1* ei ole erityisen tärkeä, mutta sen tekeminen voidaan aloittaa heti. Sen kesto on 6 yksikköä ja määräajaksi sovittu $d_1 = 12$.

Kuvassa 21a ainoastaan *Työ1* myöhästyy 2 yksikköä. Kuvassa 21b ainoastaan *Työ2* myöhästyy 3 yksikköä. Näiden tilanteiden eron määrää se, kuinka tärkeän työn myöhästymistä painotetaan, ja miten koneiden käyttämättömyydestä sakotetaan. Oletetaan, että käytössä on seuraavat sakot:

- 1 piste/aikayksikkö, kun kone on käyttämättä

- 2 pistettä/aikayksikkö, kun työ on myöhässä
- 3 pistettä/aikayksikkö, kun tärkeä työ on myöhässä.

Tällöin kuvan 21a tapauksessa sakko on 7 pistettä ja kuvan 21b tapauksessa 9 pistettä. Tilanne muuttuu vielä enemmän tapauksen a hyväksi, mikäli keskitytään pelkästään myöhästymisiin. Käytännössä myöhästymisten minimointi on usein pääasiallinen prioriteetti. On kuitenkin syytä huomata, että vaikka tapaus a on näiden kahden työn tapauksessa parempi, se voi viivästyttää seuraavien töiden alkamista, ja sitä kautta myös töiden valmistumista. Näin ollen pitäisi aina tutkia tilanne kaikkien tiedossa olevien töiden osalta.

5.5 Jatkuva työmäärän kasvu

Käytännön tilanteissa töiden järjestykset eivät ole pysyviä. Tehtävien töiden joukko muuttuu jatkuvasti, kun uusia tilauksia saapuu. Uudet tärkeät tilaukset saattavat mennä muiden edelle. Myös uusien töiden mahdolliset poikkeavat aloitusajat vaikuttavat ladonnan suunnitteluun. Kaikki nämä aiheuttavat jatkuvaa muutosta tulevien töiden työjärjestykseen.

Tuleviin töihin, joista ei tiedetä mitään, on mahdotonta varautua. Niinpä vaihtoehdoksi jää selvittää uusi optimaalinen työjako uuden tiedon perusteella. Tämä tarkoittaa kokonaan uuden työjärjestyksen laskemista, mikä saattaa olla iso työ. Tästä syystä on tehtävä kompromisseja, jotta laskentaa saadaan nopeutettua.

Eräs vaihtoehto on käyttää heuristisia menetelmiä selvittämään, onko juuri tulleista töistä joitakin syytä sijoittaa työjonon eteen. Tätä voidaan arvioida määräaikojen mukaan, prioriteettien mukaan tai kiireellisyyden mukaan. Tässä kiireellisyydellä tarkoitetaan määräajan ja aikaisimman mahdollisen aloitusajan erotusta. Mikäli uusista töistä havaitaan tällaisia töitä, suoritetaan laskenta uudestaan, joko kokonaan tai osin.

Voidaan myös pitää lyhyen ja pitkän aikavälin suunnittelut erillään. Tässä mallissa pidetään yllä pitkän tähtäimen suurpiirteistä aikataulua sekä tarkkaa lyhyen tähtäimen aikataulua. Suurpiirteistä aikataulua laskettaessa ei oteta kaikkia muuttujia huomioon, jolloin sen laskeminen on nopeampaa. Tarkempi

eli lyhyen tähtäimen aikataulun laskenta ottaa kaikki muuttujat huomioon, mutta laskenta nopeutuu, kun laskentaa ei tarvitse ajallisesti ulottaa pitkälle tulevaisuuteen. On myös mahdollista ns. lukita ajallisesti lähimmät työt, mikä saattaa helpottaa lyhyen tähtäimen laskentaa. Tämä tosin estää viime hetkillä tehtäväksi tulevien töiden aloittamisen ajoissa.

6 Algoritmit

Ladontatöiden tasapainotusongelman ratkaisemiseksi on kehitetty lukuisia algoritmeja. Valmiin ratkaisun sijaan tässä tutkielmassa yritetään kehittää omaa ratkaisua edellä esitettyjen mietintöjen pohjalta. Algoritmin kehityksessä yritetään pitäytyä yksinkertaisuudessa. Lähtökohdaksi valitaan heuristinen algoritmi, jolla saataisiin kullekin työlle etsittyä oikea linja ja kullekin linjalle oikea työjärjestys. Algoritmin tavoitteeksi otetaan töiden todellisten myöhästymisten summan minimointi.

Tässä luvussa selvitetään aluksi algoritmin päävaiheet, minkä jälkeen paneudutaan teknisiin ja toiminnallisiin yksityiskohtiin. Luvun lopuksi tarkastellaan lisäksi, kuinka geneettistä algoritmia voitaisiin käyttää ongelman ratkaisuun ja selvitetään seikkoja, jotka on syytä ottaa huomioon geneettistä algoritmia käyttäessä.

6.1 Järjestelmän mallinnuksesta

Linjasto koostuu yhdestä tai useammasta linjasta, joista kukin koostuu yhdestä tai useammasta koneesta. Kustakin koneesta tiedetään sen ladontanopeus, kelakapasiteetti, asetetut komponentit sekä tiedot asennetuista suuttimista ja suuttimista, jotka on mahdollista asentaa koneeseen. Koska kaikkiin koneisiin ei voida asentaa kaikkia mahdollisia suutintyyppejä, joudutaan konekohtaisesti pitämään kirjaa, mitkä suuttimet on asennettu sekä mitkä suuttimet on mahdollista asentaa koneeseen. Komponenttien asettelussa mahdollisesti käytettäviä syöttökärryjä ei mallinneta. Niiden käyttöä voitaisiin periaatteessa simuloida muuttamalla sekä yhden komponentin asetteluaikaa että asettelukohtaista kokonaisaikaa.

Ladontatyö käsittää tässä mallinnuksessa yhden piirilevytyypin ja sen valmistusmäärän. Lisäksi työlle on määritelty aloitus- ja määräaika. Yksinkertaisuuden vuoksi oletetaan, että kaikki työt ovat samanarvoisia, eli töitä ei erikseen voi määrittää muita kiireisemmiksi tai tärkeämmiksi.

Kukin piirilevytyyppi käsittää siihen kuuluvien komponenttien kokoelman. Komponenttikokoelma puolestaan sisältää listan kokoelmaan kuuluvista komponenttityypeistä sekä niiden lukumäärästä. Komponenttien mallin-

nus käsittää komponenttikelan leveyden sekä ladontaan vaaditun suuttimen. Suuttimet puolestaan mallinnetaan yksinkertaisesti numeroimalla suuttimet.

6.2 Heuristinen tehtävänjakoalgoritmi

Heuristisesta algoritmia kehitettäessä pyritään antamaan yleispätevä menetelmä ladontatöiden jakamiseen siten, että se tuottaa hyvän tuloksen riippumatta tehtävien ladontatöiden sekä käytettävän linjaston ominaisuuksista. Edellisessä luvussa algoritmin suunnittelun perusajatuksena käytettiin inkluusiopuuta. Puun avulla saatua jakoa yritettiin sen jälkeen parantaa vaihtotekniikalla. Kyseessä oli kuitenkin yksinkertaistettu tilanne, jossa suurin ongelmanaiheuttaja oli suutinvalinnat. Yksinkertaistetussa mallissa riitti mahdollisimman hyvä ryhmäjako samankaltaisuuden perusteella. Tosiasiassa töiden määräajat ja suoritusnopeus vaikuttavat eniten myöhästymisiin, jolloin samankaltaisuutta ei voida käyttää ainoana, tai edes pääasiallisena, mittarina. Mikäli samankaltaisuus olisi ryhmittelyn ainoa peruste, saattaisivat työt, jotka tulisi suorittaa nopeasti, sijoittua samalle linjalle peräkkäin tehtäviksi. Nopeasti tehtävien töiden pitäisi olla eri ryhmissä sijoitettuna rinnakkaisille linjoille. Tästä syystä tämän algoritmin kehityksessä töiden määräajat ja suoritusnopeudet ovat ensisijaisia kriteereitä.

6.2.1 Algoritmin toimintaperiaate

Aluksi edellä mainittu ns. "nopeasti tehtävä työ"on syytä määritellä paremmin. Määritellään työlle jo luvussa 5.5 mainittu ominaisuus: *kiireellisyys* (engl. *urgency*). Se kertoo, kuinka monta komponenttia tiettyä aikayksikköä kohden on ladottava, jotta työ valmistuisi määräaikaansa mennessä. Toisin sanoen kiireellisyys ilmaisee vaaditun ladontanopeuden. Mikäli vaadittu ladontanopeus ylittää mahdollisen ladontanopeuden, on myöhästyminen väistämätöntä. Kiireellisyys on esitetty kaavassa (13). Se määritetään jakamalla työn j ladottavien komponenttien määrä (P_j) määräaikaan (d_j) jäljellä olevalla ajalla.

$$U_j = \frac{P_j}{d_j - t} \tag{13}$$

Kiireellisyys suhteuttaa määräajan läheisyyden vaadittuun työmäärään, jolloin se on kuvaavampi kuin pelkkä määräaika.

Kiireellisyyden määrittelyn lisäksi on syytä erotella myöhästymisten määritelmät. Tässä työssä viitataan kahteen eri myöhästymiseen. Ensimmäinen on *todellinen* eli *absoluuttinen myöhästyminen* (engl. *tardiness*). Se on aina ei-negatiivinen ja ilmoittaa, kuinka paljon jokin työ tai linja on myöhässä. Toisesta käytetään nimitystä *suhteellinen myöhästyminen* tai vain *myöhästyminen*. Tämä voi olla myös negatiivinen, jos työ on ajoissa. Tämän englanninkielinen nimitys on *lateness*.

Algoritmia kehitettäessä pyritään säilyttämään järjestelmän dynaamisuus. Tällä tarkoitetaan mahdollisuutta työlistan muuttumiseen kesken tuotannon. Jos tehtävien töiden lista muuttuu, on syytä mahdollistaa lähitulevaisuuden työaikataulujen nopea päivitys. Tästä syystä algoritmi aloitetaan lajittelemalla työt määräajan mukaan.

Seuraava vaihe on töiden ryhmittely ja jako linjoille. Ryhmittelyn perusteena käytetään epäsymmetristä sisältyvyyttä. Kun ryhmät on jaettu linjoille, suoritetaan töiden järjestely ja linjojen tasapainotus siten, että myöhästymiset saadaan minimoitua.

Tässä kehitetty algoritmi ei yritäkään taata optimaalista lopputulosta, mutta sen löytämät aikataulut todennäköisesti sisältävät melko vähän myöhästymisiä. Algoritmin vaiheet ovat:

SimpleScheduleCreator:

1. Tulevat työt lajitellaan määräaikojen mukaan.

2. Käsittelemättömistä töistä valitaan erä töitä käsiteltäväksi. Valintaperusteena voidaan käyttää esim. töiden lukumäärää tai valmistumismääräaikaa. Aikataulutettavaksi voidaan valita myös kaikki järjestelmään tulleet työt.

3. Valitut työt jaetaan linjoille ryhmittelyn avulla käyttäen inkluusiopuumenetelmää.

4. Kunkin linjan työjärjestys optimoidaan. Tällä varmistetaan, että kullakin linjalla työt ovat optimaalisessa suoritusjärjestyksessä.

5. Linjojen työmäärä tasapainotetaan niiden myöhästymisten ja kiireellisyyksien perusteella. Perusperiaatteena on siirtää eniten myöhästymisiä sisältävältä linjalta eniten myöhässä oleva työ linjalle, jolla on vähiten työtä tehtävänä.

Tämänkaltainen algoritmi ei rajoita ongelman dynaamisuutta. Toisin sanoen uusia töitä voidaan lisätä tuotantojärjestelmään tuotannon ollessa kesken, kunhan töiden vaikutukset aikatauluihin selvitetään. Tämä tarkoittaa algoritmin suorittamista uudelleen ainakin kiireisimmille töille heti, kun uudet työt on lisätty listaan.

Kun aikataulutus uusitaan, otetaan töiden ryhmittelyyn mukaan myös kyseisellä hetkellä tekeillä olevat työt. On tosin huomattava, että kyseiset työt ovat jo tekeillä, joten ne pysyvät eri ryhmissä eikä niitä ei voi siirtää toisille linjoille. Seuraavassa tutkitaan tarkemmin algoritmin eri vaiheita sekä esitetään tekniset ratkaisut yksityiskohtaisemmin.

6.2.2 Algoritmin vaiheet ja yksityiskohdat

Suoritettavat työt talletetaan linkitetyyn listaan, joka pidetään kasvavassa järjestyksessä määräajan mukaan. Tämä tarkoittaa, että järjestelmään lisättävälle työlle etsitään välittömästi oikea paikka listassa. Näin ensimmäistä vaihetta ei tarvitse erikseen suorittaa.

Toisessa vaiheessa valitaan käsiteltävien töiden joukko. Kuten edellä mainittiin, valinta voi perustua joko töiden lukumäärään tai määräaikoihin. Tässä vaiheessa lajitellaan valitut työt ryhmittelyä varten levyjen sisältämien komponenttityyppien lukumäärän mukaan. Lisäksi lasketaan töiden kokonaiskomponenttimäärät valmiiksi mahdollisia osituksia varten. Kokonaiskomponenttimäärällä tarkoitetaan työn valmistumiseksi ladottavien komponenttien lukumäärää. Kokonaismäärä on siis ladottavien levyjen määrä ker-

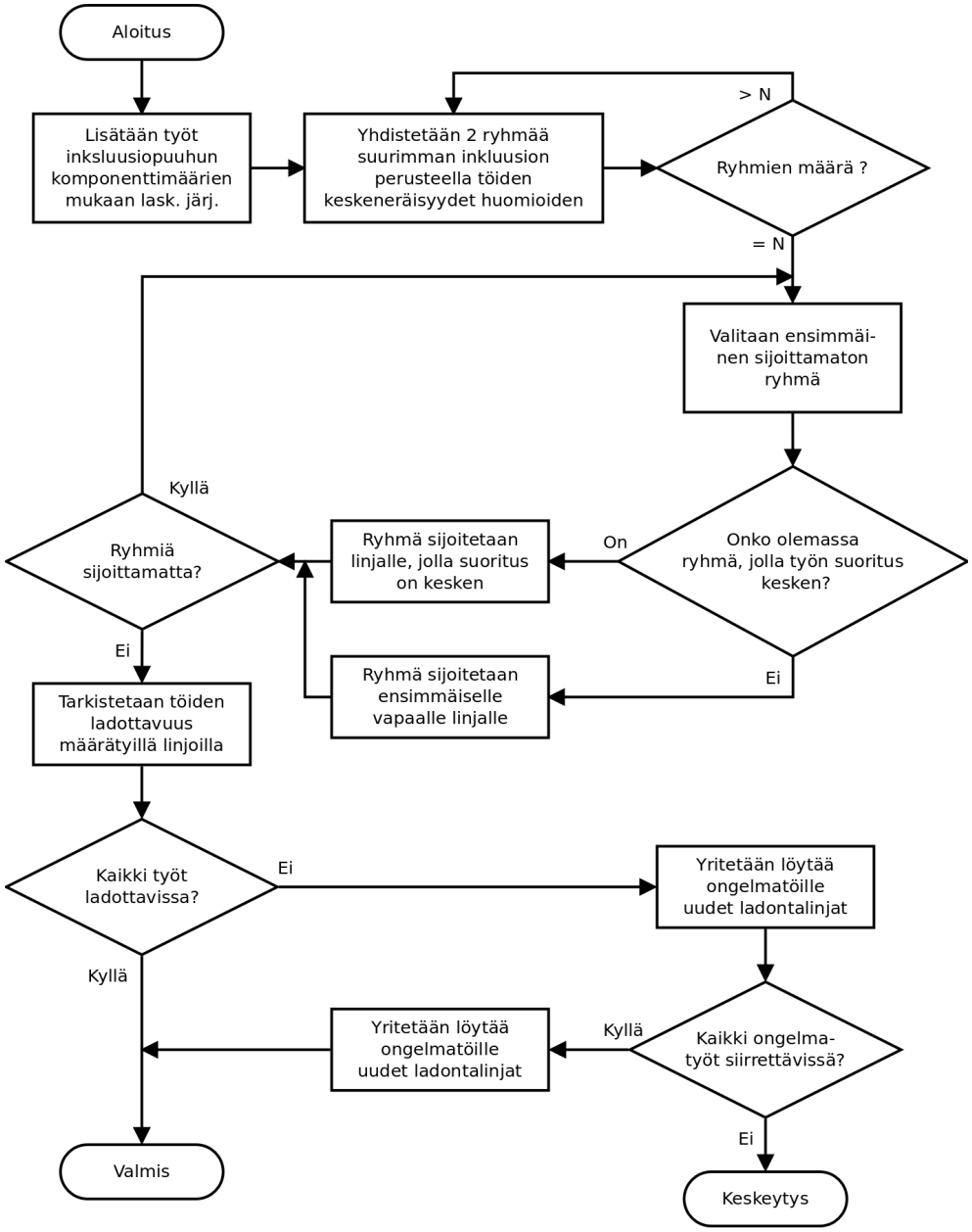
rottuna yhden levyn komponenttimäärällä:

$$P_{total} = P_j \cdot \sum_{k=1}^{K} I_{jk}$$

On kuitenkin huomattava, että aikataulutuksen päivittäminen vain lähitulevaisuuden töille mitätöi myöhempien töiden aikataulut. Toisin sanoen: aiemmin pitkällä tähtäimellä lasketut aikataulutukset joudutaan kuitenkin päivittämään ennen kyseisten töiden aloittamista, koska lähitulevaisuuden aikataulujen vaikutusta näihin töihin ei enää tunneta. Todennäköisesti paras tulos saadaan ottamalla kaikki työt mukaan. Tämän vuoksi myös toisen vaiheen, eli valinnan, tarpeellisuus on kyseenalainen.

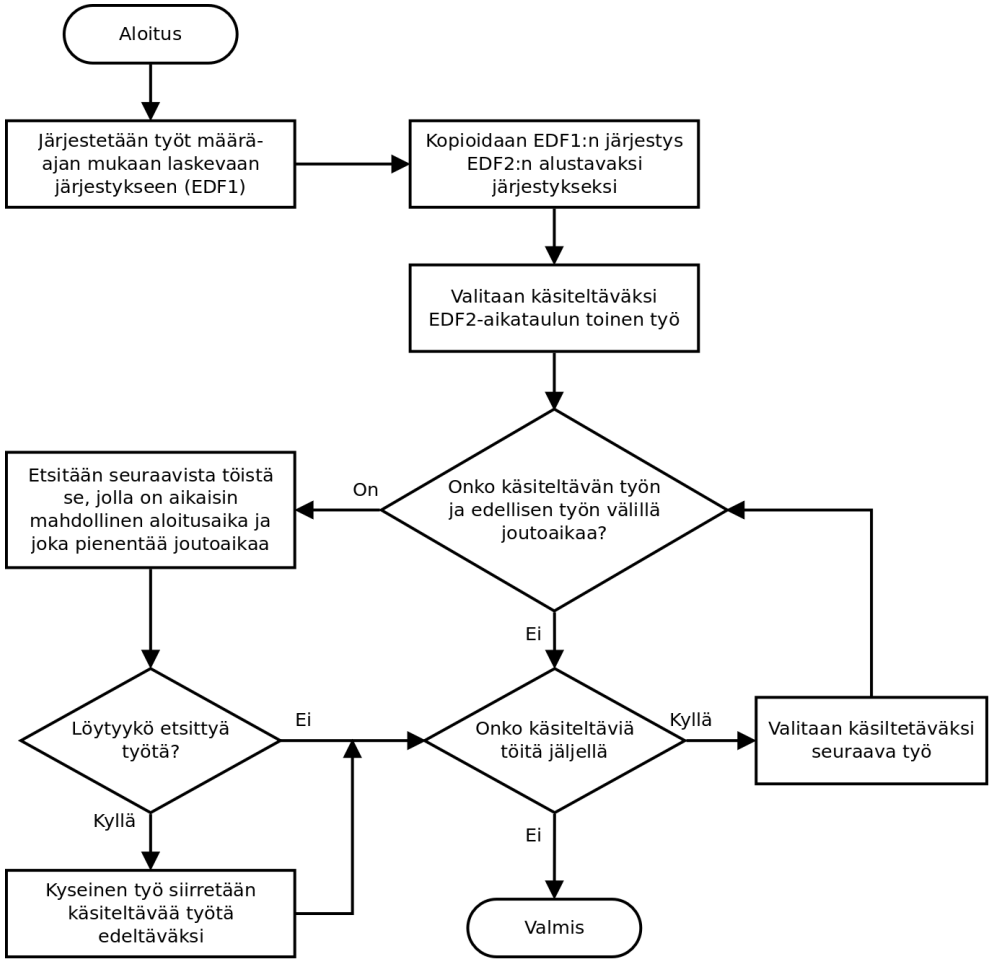
Kohdassa 3 työt ryhmitetään inkluusiopuumenetelmää käyttäen. Kuten aiemmin (luvussa 5.2.4) esitettiin, inkluusiopuun muodostaminen aloitetaan töistä, joiden komponenttityyppien lukumäärä on suurin. Muut työt liitetään puuhun lukumäärän mukaan laskevassa järjestyksessä. Ryhmät yhdistetään sisältyvyyksien perusteella, joita vertailtaessa tässä käytetään epäsymmetristä sisältyvyysmittaria (kaava (10) s. 53). Tässä vaiheessa on huomioitava, että kahta ryhmää ei voida yhdistää, jos kumpikin sisältää tekeillä olevan työn. Ryhmiä yhdistetään kunnes niiden lukumäärä on sama kuin tuotantolinjojen määrä. Kun ryhmät on muodostettu, poistetaan ns. mahdottomat tilanteet. Tällainen tilanne muodostuu, jos jotain linjalle määrättyä työtä ei voida suorittaa komponentti- tai suutinrajoitusten vuoksi. Työt, jotka aiheuttavat mahdottoman tilanteen, siirretään ryhmään, jossa ladonta on mahdollista. Kohta 3 on esitetty vuokaaviona kuvassa 22.

Kun selvitetään, onko työ mahdollista latoa linjalla, työtä varten yritetään luoda komponenttiasettelu. Yksinkertaisuuden vuoksi käytetään yksittäisasettelua. Komponenttiasettelun luominen aloitetaan eniten tilaa vaativasta komponenttityypistä (suurin kelan leveys), joka sijoitetaan linjan ensimmäiseen koneeseen, jossa on tilaa, ja jossa on mahdollista käyttää komponentin vaatimaa suutinta. Komponenttien sijoitusten jälkeen suoritetaan koneiden työaikojen tasapainotus linjan tahtiajan minimoimiseksi. Työaikoja laskettaessa otetaan huomioon asennettavien komponenttien ja suuttimien



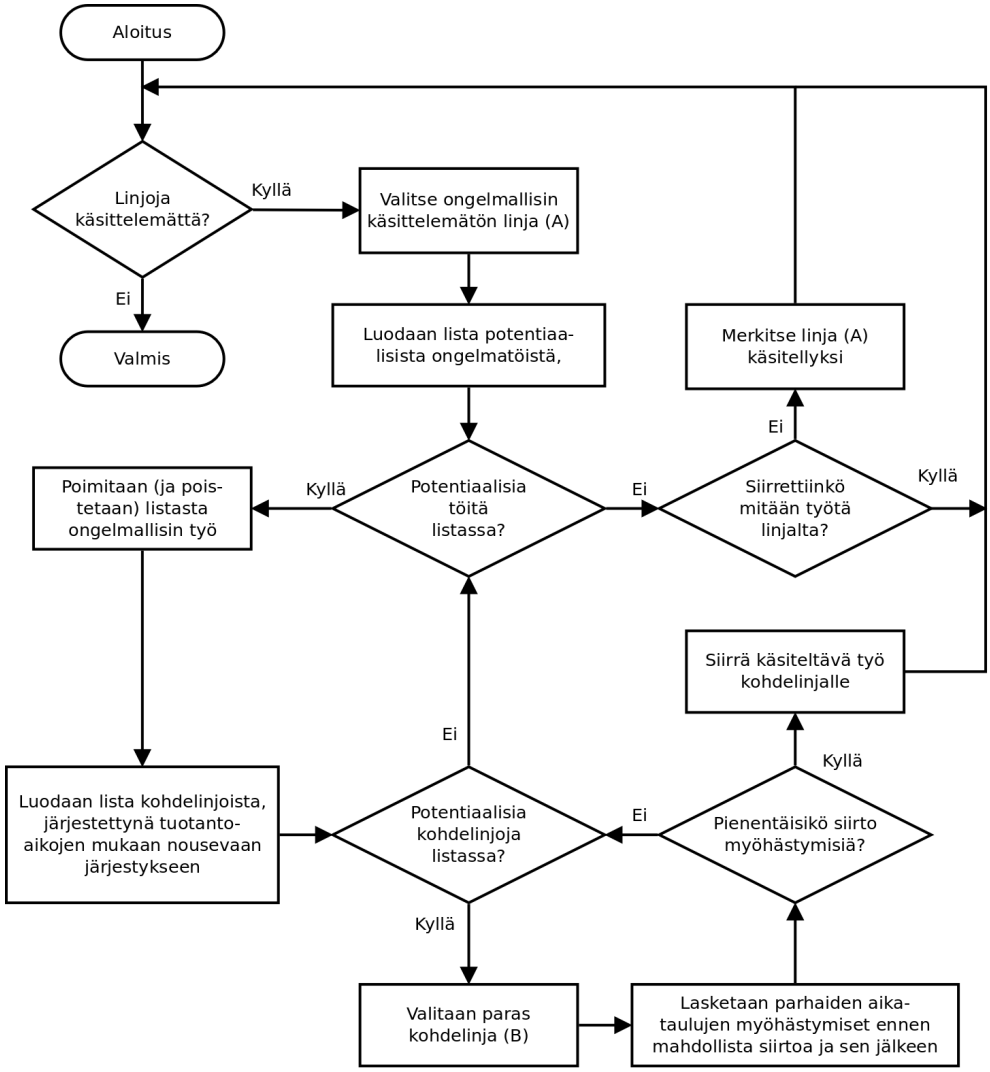
Kuva 22: Töiden ryhmittely ja työryhmien linjoille sijoittaminen.

määrä, ladottavien komponenttien määrä sekä koneen ladontanopeus. Itse tasapainotus tapahtuu siirtämällä komponenttityyppien ladonnat koneelta toiselle siten, että koneiden työaikojen poikekavuus linjan keskiarvosta sadaan minimoitua.



Kuva 23: Linjan työjärjestyksen optimointi.

Kohta 4 aloitetaan aloitetaan jakamalla isot työt osiin. Iso työ määritellään tutkimalla kunkin työn kokonaiskomponenttimäärää suhteessa töiden keskiarvoon. Jos ylitys on tietyn suuruinen, työ jaetaan kahteen osaan, joilla on sama levy, mutta puolitettu ladontamäärä. Mahdollisten jakamisten jälkeen etsitään kunkin linjan töille optimaalinen suoritusjärjestys. Ensin suoritetaan kullekin linjalle työjärjestyksen optimointi käyttäen tavallista EDF-algoritmia. Tämän jälkeen muodostetaan vaihtoehtoinen järjestys, jossa joutoajat on pyritty minimoimaan kuvan 23 mukaan. Näin saaduista kahdesta aikatauluista valitaan linjan alustavaksi aikatauluksi se, jonka myöhästyminen on pienempi. Vertailukriteerinä käytetään todellista myöhästymistä. Mi-



Kuva 24: Linjojen työmäärien tasapainotus.

käli aikataulujen todelliset myöhästymiset ovat yhtä suuret, verrataan suhteellisia myöhästymisiä.

Kohta 5 käsittää linjojen tasapainotuksen. Tämän kohdan vuokaavio on esitetty kuvassa 24. Tasapainotus perustuu ryhmien ja töiden myöhästymisten, komponenttimäärien sekä kiireellisyyksien vertailuun.

Ensin valitaan ns. ongelmallisin linja, jota ei ole vielä käsitelty. Ongelmallisuuden perusteena käytetään myöhästymisiä, ensisijaisesti todellista ja sen jälkeen suhteellista. Negatiivinen myöhästyminen lasketaan tässä linjan

tai työn eduksi. Ongelmalliselta linjalta kerätään lista siirrettävistä töistä. Tähän listaan kerätään kaikki siirrettävissä olevat työt järjestettynä ensisijaisesti myöhästymisen perusteella ja toissijaisesti kiireellisyyden perusteella.

Tämän jälkeen listataan potentiaaliset kohdelinjat siirrettävälle työlle. Yritettävien kohdelinjojen järjestys määräytyy linjojen tämänhetkisten työmäärien mukaan. Ensimmäisenä yritetään linjaa, jolla olevat työt vievät vähiten aikaa. Siirron kannattavuuden määrittämiseksi luodaan uudet aikataulutukset, joille suoritetaan kuvan 23 optimoinnit. Mikäli ongelma- ja kohdelinjan yhteenlasketut myöhästymiset pienenevät, on siirto kannattava, ja työn siirto suoritetaan.

Mikäli siirto linjalle ei ole kannattava, yritetään listan seuraavaa kohdelinjaa. Mikäli siirto millekään kohdelinjalle ei pienennä myöhästymisiä, valitaan seuraava ongelmallinen työ siirron kohteeksi ja yritetään kohdelinjoja uudelleen. Jos mikään ongelmallisen linjan töiden siirroista ei ole kannattava, merkitään linja käsitellyksi. Tällöin siirrytään seuraavaan ongelmalliseen linjaan ja toistetaan edellä mainitut vertailut siirtopäätöksen tekemiseksi. Jos siirto on kannattava, siirto suoritetaan. Tasapainotus on valmis, kun miltään linjalta ei kannata siirtää mitään työtä muuhun linjaan.

6.2.3 Pseudokoodi

Seuraavassa kehitetään algoritmille pseudokoodi. Lähestymistapana käytetään ns. top-down -menetelmää, eli pseudokoodin kehittäminen aloitetaan ylimmältä tasolta, minkä jälkeen määritetään käytettävien funktioiden toiminnallisuus. Päätaso sisältää kolme funktiokutsua edellä esitettyjen kolmen vaiheen (3–5) mukaan: ryhmittely, sijoitus ja tasapainotus.

Algoritmi 1 Skeduloijan päärakenne

```
Input:  Jobs - list of jobs to schedule
        Network - production network
function SimpleScheduleCreator(jobs, network)
    jobgroups ← CreateJobGroups(jobs, network.linecount)
    schedules ← AssignJobGroups(jobgroups, network)
    schedules ← BalanceSchedules(schedules)
    return schedules
end function
```

Päätason ensimmäinen vaihe on töiden ryhmittely, joka kuvataan algoritmissa 2. Ryhmitelyssä työt lisätään tyhjään inkluusiopuuhun alkaen työstä, joka sisältää eniten erilaisia komponenttityyppejä. Tämän jälkeen inkluusiopuun ryhmiä yhdistetään, kunnes ryhmiä on jäljellä haluttu määrä.

Ryhmien sijoittaminen linjoille aloitetaan ryhmistä, joihin kuuluu keskeneräinen työ. Nämä ryhmät sijoitetaan niille linjoille, joilla työ on kesken. Loput ryhmät sijoitetaan jäljellä oleville linjoille satunnaisesti. Kullekin linjalle kehitetään alustava aikataulu käyttäen sekä tavallista että muunneltua EDF-algoritmia, minkä jälkeen isot työt jaetaan osiin. Tämä on kuvattu algoritmissa 3

Linjojen tasapainotusfunktio on kuvattu algoritmissa 4. Se koostuu lukuisista sisäkkäisistä silmukoista, joilla yritetään ongelmallisuusjärjestyksessä siirtää ongelmallisilta linjoilta ongelmallisimpia töitä vähiten ongelmallisille linjoille.

Algoritmi 2 Töiden ryhmittely

```
Input: jobs - list of jobs
       count - desired number of groups
function CreateJobGroups(jobs, count)
    Sort Jobs according amount of used component types.
    Create empty inclusion tree.
    for all job in Jobs do
        Add job to inclusion tree
    end for
    while inclusiontree.groupcount > count do
        combine groups with largest inclusion value
    end while
    return list of jobgroups
end function
```

Algoritmi 3 Töiden sijoittaminen linjoille

```
Input: jobgroups - list of job groups
       network - network containing production lines
function AssignJobGroups(jobgroups, network)
    for all group in jobgroups do
        if group.hasJobInProgress = True then
            Assign group to line where job is in progress.
            Create initial schedule for line with EDF-algorithms
            Split large jobs in group.
        end if
    end for
    for all group in jobgroups do
        if group.hasJobInProgress = False then
            Assign group to first free line.
            Create initial schedule for line with EDF-algorithms
            Split large jobs in group.
        end if
    end for
    return initial schedules
end function
```

Algoritmi 4 Linjojen tasapainotus

```
Input: schedules - initial production line schedules
function BalanceSchedules(schedules)
    while unfinished production lines exists do
        problem_line ← unfinished line with max tardiness or lateness
        potential_jobs ← not yet started jobs
        for all job in potential_jobs do
            if job.finishtime > job.deadline then
                job.isLate ← True
            end if
        end for
        Sort late jobs and and move to beginning of list.
        Sort remaining jobs according urgency.
        to_lines ← other lines sorted according production time
        while problem_jobs.count > 0 and job_moved = false do
            job ← problem_jobs.popFirst()
            while to_lines.count > 0 do
                to_line ← to_lines.popFirst()
                Create alternative schedules where job is moved to to_line
                if alternative schedules are better then
                    schedules ← alternative schedules
                    job_moved ← True
                end if
            end while
        end while
        if job_moved = false then
            problem_line.finished ← True
        end if
    end while
    return schedules
end function
```

6.2.4 Algoritmin ongelmat ja parantaminen

Kehitetyn algoritmin suurin ongelma on todennäköisesti sen yksinkertaisuus. Se perustuu loogiseen järkeilyyn ja siinä käytetyt tekniikat ovat yksinkertaisia, josta syystä asiat tehdään yksi kerrallaan ja toistuvasti. Tästä syystä algoritmin suoritus saattaa kestää kauan. Lisäksi tuotettavat aikataulut eivät ole optimaalisia. Seuraavassa kerrotaan muutama parannusehdotus, joilla algoritmia voitaisiin melko helposti parantaa.

Algoritmi jakaa työt osiin kokonaiskomponenttimäärien perusteella. Jakamisperusteena voitaisiin myös käyttää kaavassa (13) esitettyä kiireellisyyttä. Tällöin esimerkiksi jo hieman myöhässä oleva työ olisi mahdollista jakaa kahdelle linjalle ja näin estää sen myöhästyminen.

Algoritmissa käytetään yksinkertaisuuden vuoksi yksittäisasettelua. Koska algoritmissa käytetään ryhmittelyä, parempaan tulokseen päästäisiin ryhmäasetteluja käyttäen. Tätä voidaan yrittää simuloida ottamalla huomioon ainoastaan niiden komponenttien vaihtoajat, joita ei ole jo asennettuna koneeseen. Myös ryhmittelyperusteena käytettävän epäsymmetrisen inkluusion toimivuus kannattaa kyseenalaistaa. Kuten luvussa 5.2.2 mainittiin, on olemassa monia inkluusiomittareita.

Tutkittaessa tasapainotusfunktiota (algoritmi 4), huomataan eräs mahdollinen ongelma. Kun linja on merkitty käsitellyksi, sitä ei enää voida ajatella ongelmalliseksi. Voi kuitenkin olla mahdollista, että työn siirron vuoksi jo käsitelty linja muuttuu ongelmalliseksi, kun työ siirretään sille. Jos linjalta tässä yhteydessä poistettaisiin käsitelty-leima, voisi jokin työ jäädä siirtymään edestakaisin kahden tai useamman linjan välille. Tällöin algoritmin päättyminen olisi varmistettava muilla tavoin.

Seuraavaksi tarkastellaan tahtiajan optimointia. Edellä esittiin menetelmä, jossa työmäärien tasoituksessa komponenttityyppien sijoituksia siirretään koneelta toiselle. Tasoitus olisi optimaalisempi, jos usealla koneella voisi latoa samaa komponenttityyppiä. Tämän menetelmän käyttökelpoisuutta saattaa kuitenkin rajoittaa ladontakoneiden kela- ja suutinkapasiteetit sekä ylimääräisiin asennuksiin kuluva aika.

Yksi hyvä keino, jolla algoritmista saataisiin monipuolisempi, olisi sako-

tuskäytännön mukaan ottaminen. Tämä mahdollistaisi töiden prioriteettien huomioon ottamisen, erilaisten myöhästymisten painottamisen ja ehkä myös erilaisten tavoitefunktioiden mukaan ottamisen. Eräs tällainen uusi tavoitefunktio voisi olla luvussa 3.4 esitetty tasaisuusindeksi.

6.3 Geneettisen algoritmin käyttö

Geneettisten algoritmien käytöllä on mahdollista saada kohtuullisen hyviä ratkaisuja ilman, että varsinaista ongelmaa tunnetaan kunnolla. Geneettisen algoritmin käytön ongelmana on kuitenkin kehittää hyvä ja käyttökelpoinen aakkosto sekä laatia ratkaisun hyvyyttä mittaava funktio. Seuraavassa mietitään, mitä asioita pitää ottaa huomioon tämän ongelman ratkaisemisessa, jos käytetään geneettistä algoritmia.

6.3.1 Aakkoston kehittäminen ja operaatiot

Geneettisessä algoritmissa muuttujien tieto on sisällytetty *kromosomeihin*, jotka koostuvat *geeneistä*. Tiedon sisällyttämistapaa kutsutaan koodaukseksi. Koodaustapoja on useita, joilla kullakin on omat hyvät ja huonot puolensa [SB06].

Tavallisessa koodaustavassa kromosomi määritellään vektorina, jonka kukin alkio a_j kertoo, millä linjalla työ j suoritetaan. Tämä on yksinkertaisin ja yleisesti käytetyin koodaustapa, mutta tavalliset risteymä- ja mutaatiooperaatiot tuottavat helposti epäpäteviä tuloksia. Näitä voidaan välttää sakotusmenetelmän ja heuristiikan avulla. [SB06] [RK06]

Toinen koodaustapa on järjestyskoodaus. Tässä tavassa kromosomeihin on tallennettu töiden suoritusjärjestyksiä. Schollin ja Beckerin mukaan on kuitenkin osoitettu, että tämän kaltainen koodaus ei tuota yksiselitteisiä ratkaisuja [SB06]. Johonkin ratkaisuun voidaan päätyä useasta erilaisesta työjärjestyksestä ja tietty työjärjestys voi johtaa useaan eri ratkaisuun. Tämä ei ole välttämättä huono asia, mutta kuitenkin huomion arvoinen seikka. Koska työjärjestystä ei voi rikkoa mielivaltaisesti, järjestyskoodauksessa käytetään hienostuneempia risteytystekniikoita, esim. *kahden pisteen risteytystä* [SB06].

Muita koodaustapoja ovat ryhmäkoodaus ja epäsuorat koodaustavat. Ryhmäkoodauksessa kromosomit koostuvat kahdesta osasta. Ensimmäinen osa sisältää tiedon työstä, kuten tavallisessa koodaustavassa, mutta lisäksi mukana on ryhmittelyyn liittyvä osa, joka sisältää geenin kullekin linjalle. Myös tämä menetelmä vaatii monimutkaisempia menetelmiä geneettisille operaatioille. Epäsuorat koodaustavat sisältävät tavat, joissa geeneillä ei viitata suoraan töihin tai suoritusjärjestyksiin. Viittaukset voivat käsittää esim. töiden prioriteetteihin liittyviä tietoja. Schollin ja Beckerin mukaan tämän kaltaisia tapoja käytettäessä ratkaisujen kelvollisuus on helpommin saavutettavissa. Tämän lisäksi on mahdollista käyttää heuristisia menetelmiä ml. tila-haku (engl. *local search*), joilla on mahdollista parantaa ratkaisun kyvykkyyttä. [SB06]

6.3.2 Hyvyysfunktion laatiminen

Geneettisen algoritmin tuottamille ratkaisuille on laskettava niiden hyvyys eli *fitness*. Miten kyvykkyys määritellään, riippuu täysin tuotantojärjestelmästä. Kyvykkyysmittarina voidaan käyttää mm. myöhästymisten minimaalisuutta, tuotteen valmistumisnopeutta ja erilaisia tasaisuus-indeksejä esim. kaava (3)(s. 24). Kyvykkyysfuntio voi olla hyvinkin monimutkainen ja sisältää usean ominaisuuden mittamista. [SB06] [Bal11]

Seuraavassa esitetään esimerkki äärimmilleen yksinkertaistetusta myöhästymiset minimoivasta funktiosta. Funktiossa lasketaan yhteen kullekin linjalle sijoitettujen töiden myöhästymiset.

$$\sum_{n}^{N} \sum_{j}^{J_n} max(0, d_j - f_j)$$

Tässä on syytä huomata, että negatiivista myöhästymistä eli ennen määräaikaa valmistumista ei huomioida. Mukaan lasketut negatiiviset myöhästymiset kumoaisivat todelliset myöhästymiset ja näin vääristäisivät lopullista tulosta.

6.3.3 Hybridialgoritmit

Yleensä geneettisiä algoritmeja käytettäessä suositaan ns. hybridi-algoritmeja. Hybridi-algoritmeissa yhdistetään heuristisia menetelmiä geneettisen algoritmin käyttöön. Yleisiä yhdistettäviä menetelmiä ovat *tabu*- ja *local search* -menetelmät, joilla pyritään parantamaan saatujen ratkaisuehdokkaiden kyvykkyyttä. [SSJN06][SB06][SSJ+03]

7 Testiajot ja -tulokset

Tässä luvussa tutkitaan, miten edellä kehitetty algoritmi toimii käytännössä käyttäen kolmea eri testiajoa. Ensin tarkastellaan algoritmin toimintaa yksityiskohtaisemmin pienellä työmäärällä. Tämän jälkeen tutkitaan, minkälaisia tuloksia algoritmi kehittää suurista työmääristä. Viimeiseksi mallinnetaan dynaaminen ympäristö, jossa töitä tuodaan tuotantojärjestelmään vaihtelevasti kesken tuotannon.

7.1 Rajoitukset ja yksinkertaistukset

Koska tässä työssä keskitytään töiden ryhmittelyn ja työmäärien tasapainottamisen periaatteisiin, on mallinnusta hieman yksinkertaistettu. Tämä ilmenee mm. ajoissa, joilla ladontakoneiden operaatioita mallinnetaan. Esimerkiksi komponenttityypin vaihtoajan ja suuttimen asennusajan ajatellaan olevan samat kaikille ladontakoneille. Lisäksi ladontakoneen suorittaman suuttimenvaihdon oletetaan olevan vakio koneesta riippumatta. Näiden lisäksi asetteluvaihdon aiheuttaman tuotantokatkoksen kustannusajat ovat samat kaikille linjoille.

Edellä mainittujen yksinkertaistusten lisäksi joitakin tuotannon yksityiskohtia jätetään mallintamatta, kuten edellisessä luvussa mainittiin. Varasuuttimien käyttöä ei mallinneta ts. kukin komponenttityyppi voidaan latoa vain yhdellä tietyllä suuttimella. Myöskään syöttökärryjen käyttöä ei mallinneta. Näin ollen kukin komponenttivaihto suoritetaan erikseen ja niiden vaatimat ajat lasketaan yhteen.

Asettelu perustuu yksittäisasetteluun. Ladontakoneissa jo olevat komponentit otetaan kuitenkin huomioon, joten ryhmittelystä saatava hyöty otetaan huomioon. Ryhmäasettelu tuottaisi todennäköisesti parempia tuloksia, mutta tämän työn keskittyessä tasapainotusongelmaan tyydytään tässä yksinkertaisempaan ratkaisuun.

Itse ladontakoneiden asettelun mallinnusta on myös yksinkertaistettu. Asennettujen kelojen paikkoja ei huomioida, vaan kiinnitetään huomiota ainoastaan kelojen leveyksiin ja koneiden kelakapasiteetteihin. Toisin sanoen päällekkäin asetettavia komponenttikeloja ei ole mallinnettu. Myöskään vo-

lyymituotteiden hallintaa ei ole mallinnettu, vaan oletetaan, että volyymituotteille on osoitettu omat tuotantolinjat jo ennen aikataulutusta.

7.2 Testiajoissa käytetyt linjastot ja yhteiset syötteet

Kussakin testiajossa käytetään tuotantolinjastoa, joka koostuu viiden koneen tuotantolinjoista. Ensimmäisessä ja kolmannessa ajossa on käytössä kolme tuotantolinjaa ja toisessa ajossa viisi. Ladontakoneista on mallinnettu mm. niiden ladontanopeus ja kelakapasiteetti. Ladontanopeus ilmaisee, kuinka monta komponenttia kone kykenee latomaan minuutissa. Kelakapasiteetti määrää asennettavien komponenttien kelojen yhteenlasketun maksimileveyden. Näiden lisäksi on mallinnettu suutinkapasiteetti ja asennettavissa olevat suuttimet. Suutinkapasiteetti määrää montako suutinta voi ladontakoneeseen olla yhtäaikaa asennettuna. Asennettavien suuttimien lista puolestaan kertoo, mitä suuttimia kyseiseen koneeseen voidaan asentaa.

Simuloinnissa käytetään myös yhteisiä aikamääreitä. Nämä on esitetty taulukossa 14. *Linjan uudelleenkäynnistysaika* koostuu tuotantolinjan alasajon ja uudelleenkäynnistyksen vaatimista ajoista. *Komponentin asennusaika* on aika, joka kuluu, kun yhden koneen yksi komponenttityyppi vaihdetaan toiseen. *Asettelun kiinteä vaihtoaika* koostuu muista asetteluun liittyvistä toimista. *Suuttimen asennusaika* tarkoittaa aikaa, joka kuluu yhden uuden suuttimen asentamiseen johonkin koneesesen. *Suuttimen vaihtoajalla* tarkoitetaan aikaa, joka koneelta kuluu, kun se vaihtaa ladontaan käytettävän suuttimen toiseen.

Taulukko 14: Simuloinnissa käytetyt ajat sekunteina.

Simulointiaika	$t(s)$
Linjan uudelleenkäynnistys	120
Asettelun kiinteä vaihto	60
Komponentin asennus	15
Suuttimen asennus	300
Suuttimen vaihto	2

Kaikissa testiajoissa ladottavat levyt perustuvat 102 komponenttityyppiin, joiden ladontaan käytetään 28 erilaista suutinta. Kustakin komponenttityypistä tiedetään ladontaan vaadittavan suuttimen lisäksi komponenttikelan leveys. Nämä tiedot on esitetty kaikista mallinnuksessa käytettävistä komponenteista taulukossa 15.

Taulukko 15: Taulukko käytetyistä komponenttityypeistä, niiden vaatimien kelojen leveydestä sekä niiden ladontaan vaadittavista suuttimista.

Komp.	Leveys	Suutin
R01	1	1
R02	1	1
R03	1	1
R04	1	1
R05	1	1
R06	1	1
R07	1	1
R08	1	1
R09	1	1
R10	1	1
R11	1	1
R12	1	1
R13	1	2
R14	1	2
R15	1	2
R16	1	2
R17	1	2
R18	1	2
R19	1	2
R20	1	2
R21	1	2
R22	1	2
R23	1	2
R24	1	2
R25	1	2
R26	1	2
R27	1	1
R28	1	1
R29	1	1
R30	1	1
R31	2	2
R32	2	2
R33	1	1
R34	3	3
R35	3	3
R36	3	3
R37	3	3
R38	2	2
R39	2	3
R40	2	3
R41	2	3
R42	2	3
R43	1	3
R44	1	3
R45	1	3

Komp.	Leveys	Suutin
L01	1	10
L02	1	10
L03	1	11
L04	1	11
L05	1	11
L06	1	12
L07	1	12
L08	2	12
L09	2	13
L10	1	13
L11	2	13
L12	2	14
L13	2	13
L14	1	14
L15	3	14
C01	1	15
C02	1	15
C03	1	15
C04	1	16
C05	1	16
C06	1	16
C07	1	16
C08	1	16
C09	2	17
C10	2	17
C11	2	17
C12	2	18
C13	2	18
C14	2	18
C15	2	18
C16	2	19
C17	2	19
C18	2	19
C19	2	20
C20	2	20
C21	2	20
C22	2	20
C23	2	20
C24	3	21
C25	3	21

Komp.	Leveys	Suutin
IC1	2	26
IC2	1	26
IC3	2	26
IC4	3	27
IC5	2	27
IC6	3	27
IC7	4	28
IC8	3	28
D1	1	4
D2	1	4
D3	1	5
D4	2	5
D5	2	6
Q1	1	7
Q2	1	8
Q3	2	9
Q4	3	9

7.3 Testiajo 1

Ensimmäisessä testiajossa käytetään taulukossa 16 esitettyjä levyjä. Levyjä on yhteensä 8 kappaletta. Levykokoelmassa on kaksi varsinaista levyä, joiden lisäksi on kaksi varianttia toisesta levystä ja neljä varianttia toisesta. Variantit sisältävät pääosin samoja komponentteja toisiinsa ja varsinaiseen levyyn verrattuna.

Taulukko 16: Testiajon levyjoukko koostuu kahdesta levystä (A ja B) ja niiden varianteista.

Levy	Komponenttien määrä	Komponenttityyppien määrä
Motherboard_A	209	46
Motherboard_A.v1	233	44
Motherboard_A.v2	241	44
Motherboard_B	164	52
Motherboard_B.v1	244	60
Motherboard_B.v2	165	46
Motherboard_B.v3	156	45
Motherboard_B.v4	136	38

7.3.1 Inkluusiopuu ja ryhmät

Töiden tulevaa ryhmittelyä varten tarvitaan levyjen keskinäiset sisältyvyydet, joiden laskemiseen käytetään aiemmin esitettyä epäsymmetristä sisältyvyyttä. Tämän mukaan lasketut sisältyvyydet on esitetty taulukossa 17.

Tässä ensimmäisessä ajossa on 10 aikataulutettavaa työtä. Työt perustuvat edellä esitettyihin levyihin, joiden valmistettavat kappalemäärät vaihtelevat neljästä yli 2000:een. Taulukko 18 sisältää näistä töistä julkaisuajan, valmistumisen takarajan, valmistettavan levyn sekä levyjen määrän. Näiden lisäksi taulukossa on mainittu kustakin työstä ladottavien komponenttien kokonaismäärä (KKM). Kuhunkin työhön viitataan kirjaimella, jotta aikataulutuksen ja linjojen tasapainotuksen seuraaminen olisi selkeämpää.

Edellä kehitetty algoritmi alkaa inkluusiopuun muodostamisella. Työt lisätään inkluusiopuuhun niihin liittyvän levyn komponenttityyppien luku-

Taulukko 17: Levyjen keskinäiset sisältyvyydet laskettuna kaavan (10) epäsymmetristä sisältyvyysmittaria käyttäen.

IM(i,j): $j \setminus i$	*Motherboard_A*	*Motherboard_A.v1*	*Motherboard_A.v2*	*Motherboard_B*	*Motherboard_B.v1*	*Motherboard_B.v2*	*Motherboard_B.v3*	*Motherboard_B.v4*
Motherboard_A	1,000	0,848	0,848	0,348	0,478	0,348	0,348	0,261
Motherboard_A.v1	0,886	1,000	0,818	0,364	0,500	0,364	0,364	0,273
Motherboard_A.v2	0,886	0,818	1,000	0,409	0,545	0,386	0,432	0,341
Motherboard_B	0,308	0,308	0,346	1,000	0,923	0,865	0,808	0,712
Motherboard_B.v1	0,367	0,367	0,400	0,800	1,000	0,700	0,683	0,600
Motherboard_B.v2	0,348	0,348	0,370	0,978	0,913	1,000	0,783	0,761
Motherboard_B.v3	0,356	0,356	0,422	0,933	0,911	0,800	1,000	0,689
Motherboard_B.v4	0,316	0,316	0,395	0,974	0,947	0,921	0,816	1,000

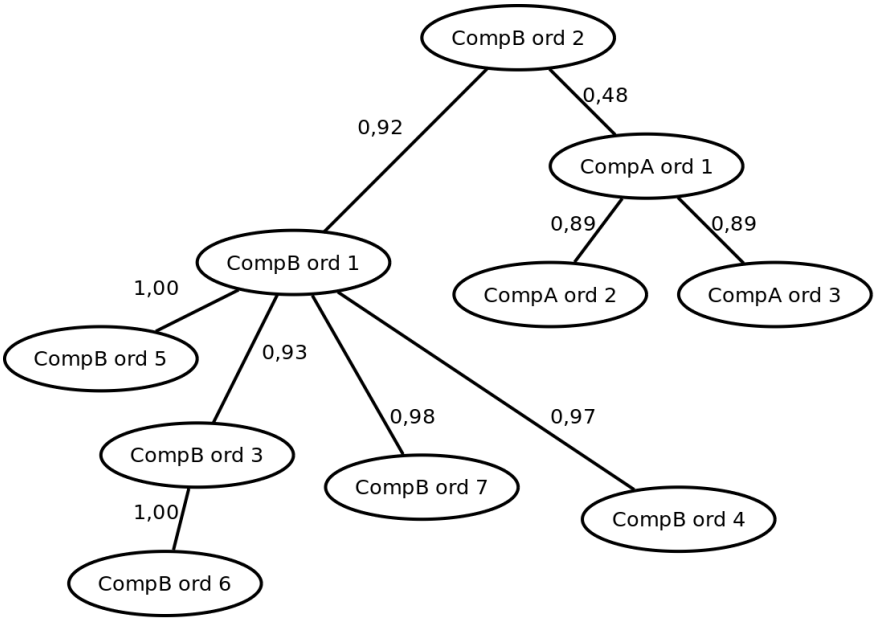
Taulukko 18: Ensimmäisessä ajossa aikataulutettavat työt ja niiden ominaisuudet.

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(A) Company A order 1	04.06.2012 08:30	06.06.2012 19:00	Motherboard_A	1000	209000
(B) Company A order 2	04.06.2012 12:00	07.06.2012 06:00	Motherboard_A.v1	120	27960
(C) Company A order 3	05.06.2012 08:00	07.06.2012 12:00	Motherboard_A.v2	800	192800
(D) Company B order 1	04.06.2012 15:00	07.06.2012 12:00	Motherboard_B	2200	360800
(E) Company B order 2	05.06.2012 12:00	07.06.2012 17:00	Motherboard_B.v1	300	73200
(F) Company B order 3	07.06.2012 10:00	08.06.2012 16:00	Motherboard_B.v3	550	85800
(G) Company B order 4	04.06.2012 18:00	06.06.2012 18:00	Motherboard_B.v4	4	544
(H) Company B order 5	06.06.2012 15:00	07.06.2012 12:00	Motherboard_B	1900	311600
(I) Company B order 6	06.06.2012 09:00	07.06.2012 18:00	Motherboard_B.v3	700	109200
(J) Company B order 7	06.06.2012 08:00	08.06.2012 14:00	Motherboard_B.v2	3100	511500

*KKM – Komponenttien kokonaismäärä

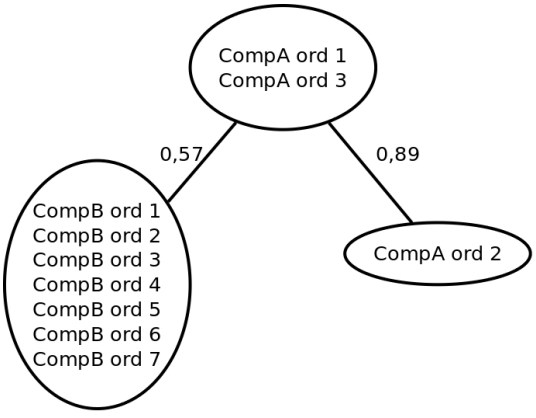
määrän mukaan laskevassa järjestyksessä. Toisin sanoen työ, jonka levy sisältää eniten erilaisia komponenttityyppejä, lisätään ensin. Lopputuloksena on kuvan 25 mukainen puu.

Puuta tarkastellessa huomio kiinnittyy yllättävän pieneen sisältyvyyskertoimeen 0.48, kun kertoimet muuten ovat vähintään 0.88. Tämä johtuu työn *Company A order 1* aikaisesta lisäyshetkestä. Kun kyseinen työ lisätään puuhun, puu sisältää vain 3 työtä, joissa käytetään vain kahta levyä. Kumpikin näistä levyistä eroaa melkoisesti lisättävän työn levystä, mikä johtaa pieneen sisältyvyyskertoimeen.



Kuva 25: Inkluusiopuu ja sisältyvyydet ennen ryhmien yhdistämistä.

Koska tässä ensimmäisessä ajossa on käytössä kolme linjaa, yhdistetään inkluusiopuun ryhmiä, kunnes ryhmiä on kolme kappaletta. Nämä lopputuloksena saadut kolme ryhmää on esitetty kuvassa 26. Kuvasta nähdään kuinka epätasaisesti työt jakautuvat. Voidaankin todeta, että pelkkä ryhmittely voi johtaa huonoon tulokseen tasapainotuksen osalta.



Kuva 26: Inkluusiopuu ryhmien yhdistämisen jälkeen.

Inkluusiopuuta käyttämällä saadut ryhmät sijoitetaan nyt linjaston tuotantolinjoille. Linjasto koostuu kolmesta koneesta, joiden ominaisuudet on esitetty taulukossa 19. Koska töiden suoritusta ei ole vielä aloitettu, sijoitetaan ryhmät satunnaisesti linjoille.

Taulukko 19: Linjaston ladontakoneet ja niiden ominaisuudet. Nopeus on ilmoitettu keskimäärin minuutissa ladottavien komponenttien määränä (kpl/min). Kapasiteetti on koneen kelakapasiteetti kelayksikköinä (ky).

Linja 1

	M_1A	M_1B	M_1C	M_1D	M_1E
Nopeus (kpl/min)	450	450	450	450	250
Kapasiteetti (ky)	30	30	30	30	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22 26	20 21 22 26	20 21 22 26	

Linja 2

	M_2A	M_2B	M_2C	M_2D	M_2E
Nopeus (kpl/min)	600	600	600	600	250
Kapasiteetti (ky)	25	25	25	25	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22 26	20 21 22 26	20 21 22	

Linja 3

	M_3A	M_3B	M_3C	M_3D	M_3E
Nopeus (kpl/min)	500	500	500	500	380
Kapasiteetti (ky)	28	28	28	28	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22	20 21 22 26	20 21 22 26	

7.3.2 Alustava töidenjärjestely

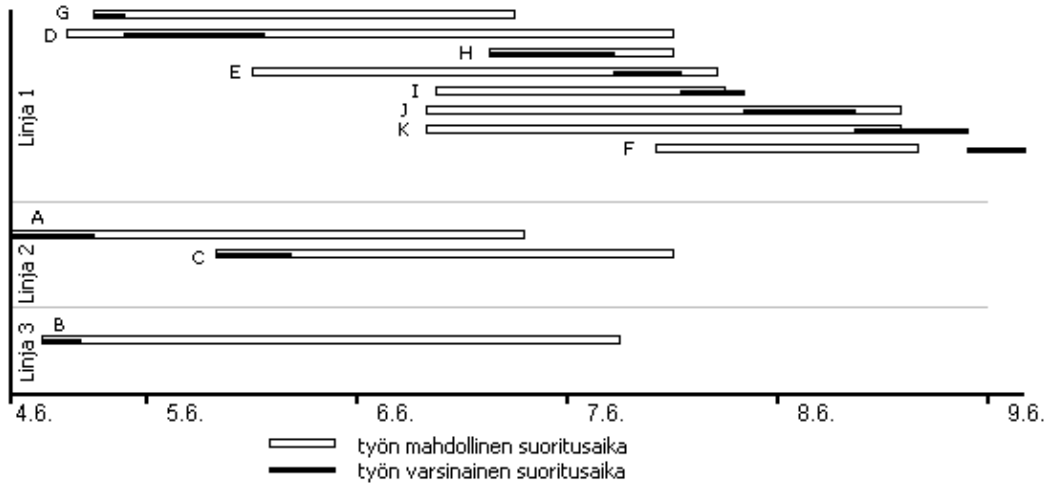
Sijoitusten jälkeen käydään töiden komponenttimäärät läpi ja jaetaan isot työt osiin. Työ on iso, jos sen komponenttimäärä ylittää raja-arvon. Raja-arvoksi on määritelty 200%:a töiden komponenttimäärien keskiarvosta. Taulukon 18 töiden komponenttikeskiarvo on 188240.4, jolloin raja-arvoksi saadaan 376480.8. Tehtävistä töistä ainoastaan *Company B order 7*:n komponenttimäärä ylittää tämän arvon. Tällöin se jaetaan kahteen osaan ja jälkim-

mäisen työn nimeen liitetään tunnus ".pt2". Tähän uuteen työhön viitataan aikatauluissa kirjaimella *K*.

Töiden osiin jakamisen jälkeen kullakin linjalla suoritetaan optimoinnit. Optimointi aloitetaan järjestämällä työt nousevaan järjestykseen määräaikojen mukaan. Toisin sanoen käytetään EDF-algoritmia (*earliest deadline first*), jolla saadaan osaltaan minimoitua myöhästymisiä [KT06].

Järjestämisen jälkeen suunnitellaan komponenttiasettelut. Asettelut pyritään tasapainottamaan siten, että kullakin linjan koneella komponenttien ladonta veisi yhtä paljon aikaa. Tällöin linjan tahtiaika saadaan mahdollisimman pieneksi ja sitä kautta tuotanto nopeaksi. Kun asettelut ovat selvillä, voidaan laskea kunkin työn vaatimat tuotantoajat, ja siten myös kunkin työn aloitus- ja valmistumisajat.

Näin saadut linjakohtaiset tuotantoaikataulut on esitetty taulukossa 20 ja havainnollisemmin kuvassa 27. Kuvassa ontto suorakaide kuvastaa työn ns. tuotantoikkunaa, joka on työn julkaisuajan ja määräajan välinen aika. Musta palkki kuvaa itse työn suorittamiseen kuluvaa aikaa. Tässä aikataulujen esittämistä on yksinkertaistettu siten, että tuotantoajat sisältävät myös asetteluihin ym. tuotannollisiin asioihin kuluvat ajat kuten esim. linjan uudelleenkäynnistämiset.



Kuva 27: Alkuperäisen EDF-algoritmin tuottamat aikataulut linjakohtaisesti.

Taulukko 20: EDF-algoritmin tuottamat aikataulut linjakohtaisesti.

Linja 1

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company B order 4	04.06.2012 18:00	04.06.2012 18:00	04.06.2012 21:24	06.06.2012 18:00
Company B order 1	04.06.2012 15:00	04.06.2012 21:24	05.06.2012 13:19	07.06.2012 12:00
Company B order 5	06.06.2012 15:00	06.06.2012 15:00	07.06.2012 05:22	07.06.2012 12:00
Company B order 2	05.06.2012 12:00	07.06.2012 05:22	07.06.2012 12:50	07.06.2012 17:00
Company B order 6	06.06.2012 09:00	07.06.2012 12:50	07.06.2012 20:00	07.06.2012 18:00
Company B order 7	06.06.2012 08:00	07.06.2012 20:00	08.06.2012 08:46	08.06.2012 14:00
Company B order 7.pt2	06.06.2012 08:00	08.06.2012 08:46	08.06.2012 21:32	08.06.2012 14:00
Company B order 3	07.06.2012 10:00	08.06.2012 21:32	09.06.2012 04:01	08.06.2012 16:00

Linja 2

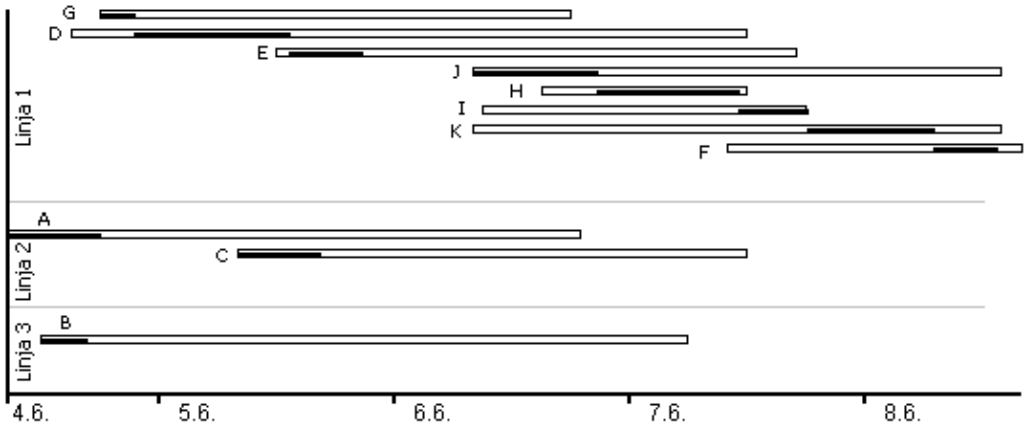
Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 1	04.06.2012 08:30	04.06.2012 08:30	04.06.2012 18:02	06.06.2012 19:00
Company A order 3	05.06.2012 08:00	05.06.2012 08:00	05.06.2012 16:30	07.06.2012 12:00

Linja 3

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 2	04.06.2012 12:00	04.06.2012 12:00	04.06.2012 16:31	07.06.2012 06:00

Vaikka töille on asetettu melko väljät tuotantoikkunat, linjalla 1 viimeiset työt myöhästyvät. Tähän on osasyynä töiden järjestämisessä käytetty EDF-algoritmi, joka ottaa huomioon vain määräajat. Tästä syystä aikataulu sisältää paljon joutoaikaa. Ylimääräisen joutoajan välttämiseksi EDF-algoritmista ajetaan myös muunnettu versio, jolla pyritään minimoimaan joutoaika kuten kuvassa 21b. Jos muunnetussa algoritmissa esiintyy joutoaikaa, joutoaikaa seuraava työ korvataan työllä, jolla on aikaisin julkaisuaika. Näin saadaan töille vaihtoehtoinen aikataulu, joka on esitetty kuvassa 28 ja tarkemmin taulukossa 22.

Vaihtoehtoinen aikataulu tuo eroa ainoastaan linjan 1 osalta. Ero on kuitenkin huomattava. Tavallisen EDF-algoritmin jälkeen myöhässä olevien töiden yhteenlaskettu myöhästyminen on lähes 22 tuntia ja viimeinen työ valmistuu 9.6. aamuyöllä. Joutoajan minimoivan EDF-algoritmin jälkeen vastaava myöhastyminen on vain n. 18 minuuttia ja viimeinen työ valmistuu 8.6. iltapäivällä. Inkluusiopuun avulla tapahtunut töiden ryhmittely jakaa työ hyvin epätasaisesti linjoille, joten seuraavaksi siirrytään linjojen työmäärien tasapainottamiseen.



Kuva 28: Muunnetulla EDF-algoritmilla saatu vaihtoehtoinen aikataulu, jossa joutoaikaa pyritty vähentämään.

Taulukko 21: Muunnetun EDF-algoritmin tuottamat aikataulut.

Linja 1

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company B order 4	04.06.2012 18:00	04.06.2012 18:00	04.06.2012 21:24	06.06.2012 18:00
Company B order 1	04.06.2012 15:00	04.06.2012 21:24	05.06.2012 13:19	07.06.2012 12:00
Company B order 2	05.06.2012 12:00	05.06.2012 13:19	05.06.2012 20:47	07.06.2012 17:00
Company B order 7	06.06.2012 08:00	06.06.2012 08:00	06.06.2012 20:45	08.06.2012 14:00
Company B order 5	06.06.2012 15:00	06.06.2012 20:45	07.06.2012 11:08	07.06.2012 12:00
Company B order 6	06.06.2012 09:00	07.06.2012 11:08	07.06.2012 18:18	07.06.2012 18:00
Company B order 7.pt2	06.06.2012 08:00	07.06.2012 18:18	08.06.2012 07:04	08.06.2012 14:00
Company B order 3	07.06.2012 10:00	08.06.2012 07:04	08.06.2012 13:33	08.06.2012 16:00

Linja 2

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 1	04.06.2012 08:30	04.06.2012 08:30	04.06.2012 18:02	06.06.2012 19:00
Company A order 3	05.06.2012 08:00	05.06.2012 08:00	05.06.2012 16:30	07.06.2012 12:00

Linja 3

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 2	04.06.2012 12:00	04.06.2012 12:00	04.06.2012 16:31	07.06.2012 06:00

7.3.3 Linjojen tasapainotus

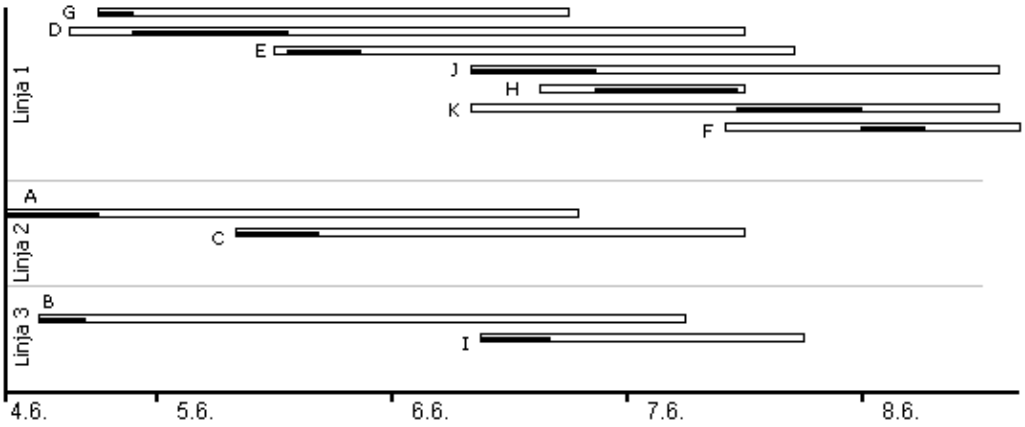
Linjojen tasapainotuksen lähtökohdaksi otetaan edellä saatujen EDF-algoritmien aikataulutuksista parempi. Ensiksi valitaan linjoista ongelmallisin. Kriteereinä linjan ongelmallisuudelle käytetään ensin todellista (tai absoluuttista) myöhästymistä (engl. *tardiness*) ja sitten myöhästymistä, joka ottaa huomioon myös työn valmistumisen ennen määräaikaa (engl. *lateness*). Ongel-

mallisesta linjasta (eli tässä linja 1) valitaan ongelmallisin työ siirrettäväksi. Tällainen on työ, joka on eniten myöhässä. Jos mikään työ ei ole myöhässä, siirrettäväksi työksi valitaan kiireellisin työ (ks. kaava (13), s. 75).

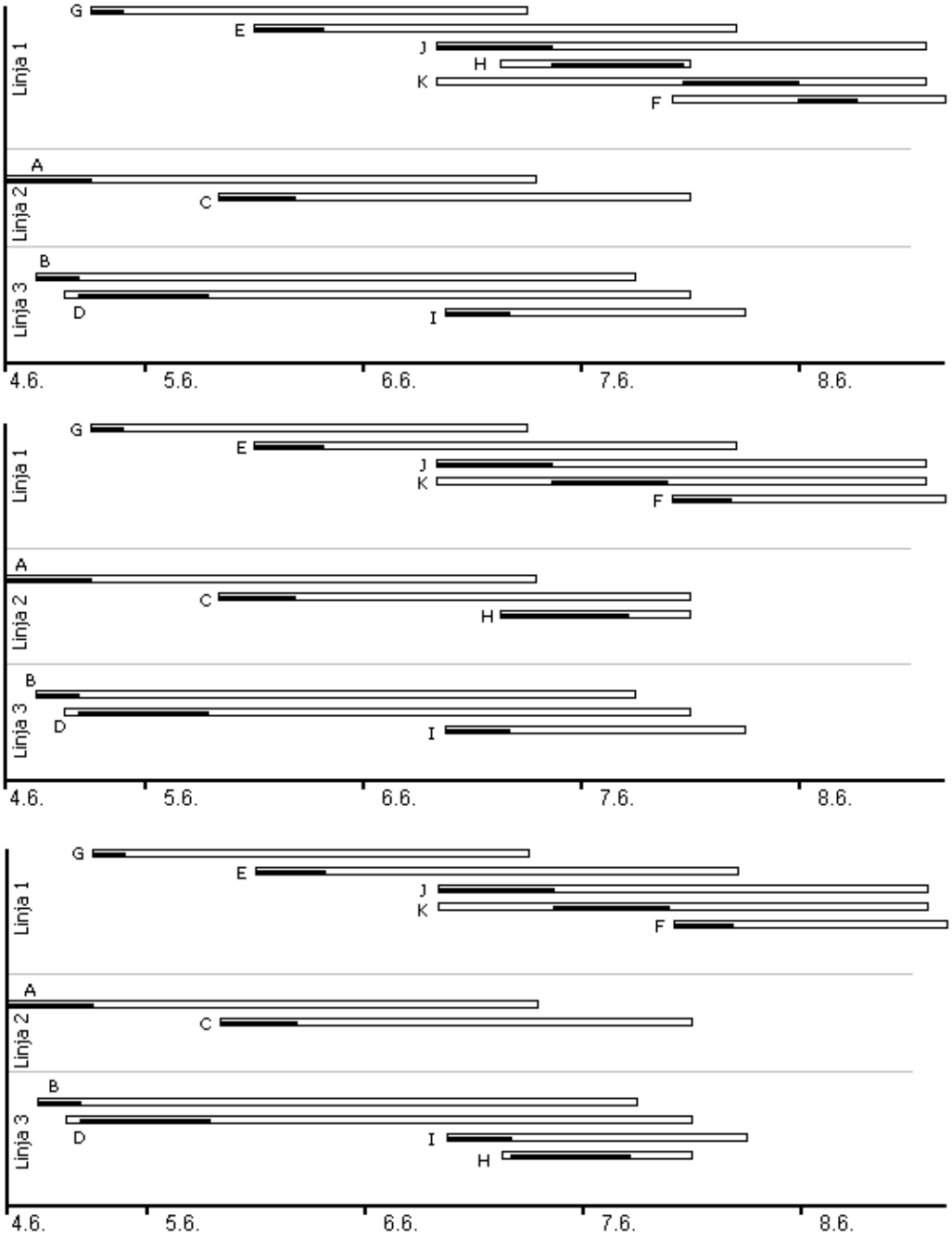
Ns. ensisijaiseksi kohdelinjaksi valitaan linja, jolle on määrätty vähiten tuotantoaikaa. Tällöin voidaan olettaa, että kyseisellä linjalla olisi eniten aikaa lisätuotannolle. Jos työn siirto ei ole kannattava, kokeillaan toista linjaa. Jos työn siirto mihinkään linjaan ei kannata, kokeillaan siirtää seuraava työ. Näin jatketaan kunnes ongelmalliset linjat on käyty läpi.

Työn siirron kannattavuuden tutkimiseksi aikataulutus tehdään uudestaan sekä ongelmalliselle että kohdelinjalle. Tähän käytetään EDF-algoritmin molempia versioita, joiden tuottamia aikatauluja arvioidaan absoluuttisen myöhästymisen vähenemisellä, tuotannon valmistumisen aikaistumisella sekä yksittäisten töiden aikaistumisella. Työn siirto on kannattava, jos työ on sopiva ja lisäksi todellinen myöhästyminen (tardiness) pienenee tai suhteellinen myöhästyminen (lateness) pienenee. Seuraava kuvasarja näyttää, miten aikataulut muuttuvat linjojen tasapainotuksen edetessä.

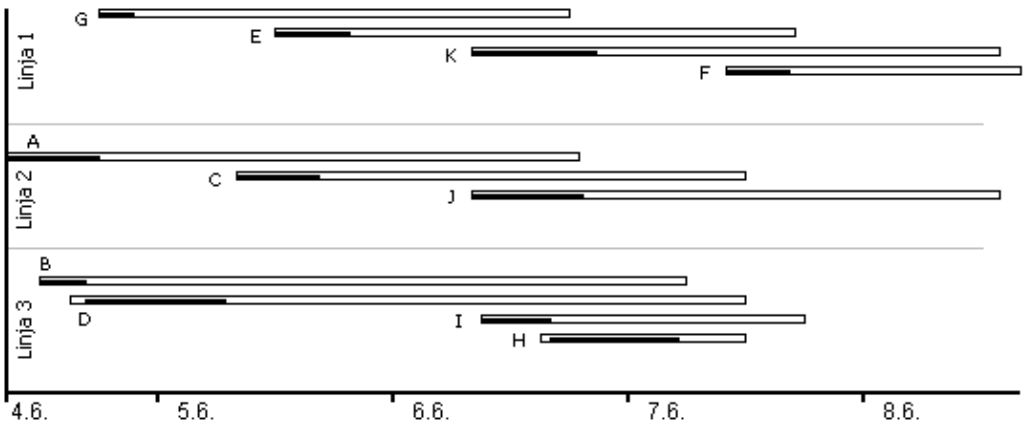
Ensimmäiseksi siirrettäväksi työksi valitaan työ *I*, koska se on linjan 1 ainoa myöhästyvä työ. Kohdelinjaksi valitaan linja 3, koska sille on määrätty vähiten töitä ajallisesti laskettuna. Tämän jälkeen siirretään töitä niiden kiireellisyyksien perusteella. Siirrettävät työt ovat järjestyksessä: *I*, *D*, *H*, *H* uudelleen ja *J*. Näin päästään kuvan 31 tilanteeseen.



Kuva 29: Linjojen aikataulut ensimmäisen työn siirron jälkeen.



Kuva 30: Linjojen aikataulujen muuttuminen töiden siirtojen myötä.



Kuva 31: Lopulliset aikataulut tasapainotuksen jälkeen.

Taulukko 22: Lopulliset aikataulut.

Linja 1

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company B order 4	04.06.2012 18:00	04.06.2012 18:00	04.06.2012 21:24	06.06.2012 18:00
Company B order 2	05.06.2012 12:00	05.06.2012 12:00	05.06.2012 19:28	07.06.2012 17:00
Company B order 7.pt2	06.06.2012 08:00	06.06.2012 08:00	06.06.2012 20:45	08.06.2012 14:00
Company B order 3	07.06.2012 10:00	07.06.2012 10:00	07.06.2012 16:29	08.06.2012 16:00

Linja 2

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 1	04.06.2012 08:30	04.06.2012 08:30	04.06.2012 18:02	06.06.2012 19:00
Company A order 3	05.06.2012 08:00	05.06.2012 08:00	05.06.2012 16:30	07.06.2012 12:00
Company B order 7	06.06.2012 08:00	06.06.2012 08:00	06.06.2012 19:17	08.06.2012 14:00

Linja 3

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
Company A order 2	04.06.2012 12:00	04.06.2012 12:00	04.06.2012 16:31	07.06.2012 06:00
Company B order 1	04.06.2012 15:00	04.06.2012 16:31	05.06.2012 06:55	07.06.2012 12:00
Company B order 6	06.06.2012 09:00	06.06.2012 09:00	06.06.2012 16:06	07.06.2012 18:00
Company B order 5	06.06.2012 15:00	06.06.2012 16:06	07.06.2012 05:09	07.06.2012 12:00

7.4 Testiajo 2

Toisella testiajolla pyritään selvittämään, miten algoritmi toimii isolla työmäärällä. Testiajon työt perustuvat taulukossa 27 (ks. liite 1) esitettyihin 38 piirilevyyn, joiden komponenttimäärät vaihtelevat muutamasta kymmenestä yli 1200:aan. Aikataulutettavia töitä on tässä ajossa 100 kappaletta. Niiden alkamisajat sijoittuvat neljän viikon ajalle. Nämä työt on esitetty taulukossa 28 (ks. liite 2).

Toiseen testiajoon on tehty yksi muutos edelliseen testiajoon verrattuna. Nyt työn komponenttimäärän pitää ylittää keskiarvo nelinkertaisesti, jotta työ jaettaisiin osiin. Tähän on syynä töiden määrä ja esitystapa, joka yritetään tässä pitää selkeänä. Liiallinen töiden osittaminen voi johtaa töiden lukumäärän huomattavaan kasvuun, jolloin aikataulujen esittäminen saattaa olla sekavaa. Todellisuudessa ositusperusteet ovat monimutkaisempia, joihin vaikuttaa työkuorman lisäksi niin linjojen kuin töidenkin lukumäärät.

Tässä testiajossa käytetty linjasto on esitetty taulukossa 23. Linjasto sisältää viisi linjaa, jossa kussakin on viisi ladontakonetta. Kolme ensimmäistä linjaa ovat identtisiä edellisen testiajon linjojen kanssa. Ja kuten edellisessä testiajossa, kullakin linjalla on 1–2 ominaisuuksiltaan muista poikkeavaa konetta.

Taulukko 23: Linjaston ladontakoneet ja niiden ominaisuudet. Nopeus on ilmoitettu keskimäärin minuutissa ladottavien komponenttien määränä (kpl/min). Kapasiteetti on koneen kelakapasiteetti kelayksikköinä (ky).

Linja 1

	M_1A	M_1B	M_1C	M_1D	M_1E
Nopeus (kpl/min)	450	450	450	450	250
Kapasiteetti (ky)	30	30	30	30	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22 26	20 21 22 26	20 21 22 26	

Linja 2

	M_2A	M_2B	M_2C	M_2D	M_2E
Nopeus (kpl/min)	600	600	600	600	250
Kapasiteetti (ky)	25	25	25	25	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22 26	20 21 22 26	20 21 22	

Linja 3

	M_3A	M_3B	M_3C	M_3D	M_3E
Nopeus (kpl/min)	500	500	500	500	380
Kapasiteetti (ky)	28	28	28	28	35
Suutinkapasiteetti	8	8	8	8	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 8 9 13 14
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	15 16 17 18 23
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	24 25 27 28
	20 21 22 26	20 21 22	20 21 22 26	20 21 22 26	

Linja 4

	M_4A	M_4B	M_4C	M_4D	M_4E
Nopeus (kpl/min)	350	350	350	350	200
Kapasiteetti (ky)	30	30	30	30	40
Suutinkapasiteetti	10	10	10	10	12
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 8 9 13
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	14 15 16 17 18
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	19 23 24 25 26
	20 21 22 26	20 21 22 26	20 21 22 26	20 21 22 26	27 28

Linja 5

	M_5A	M_5B	M_5C	M_5D	M_5E
Nopeus (kpl/min)	380	380	380	300	300
Kapasiteetti (ky)	28	28	28	32	32
Suutinkapasiteetti	10	10	10	15	15
Suuttimet	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 4 5 6 7 8	1 2 3 8 9 10 16	1 2 3 8 9 10 16
	10 11 12 13 14	10 11 12 13 14	10 11 12 13 14	17 18 22 23 24	17 18 22 23 24
	15 16 17 18 19	15 16 17 18 19	15 16 17 18 19	25 26 27 28	25 26 27 28
	20 21 22 26	20 21 22 26	20 21 22 26		

Kun työt on ryhmitelty, suoritetaan isojen töiden osittaminen. Tässä tapauksessa vain neljä työtä jaetaan osiin. Tämä johtuu jakamispäätökseen käytettävästä korkeasta raja-arvosta. Jaettavat työt ovat *CompD_job_09*, *CompH_job_01*, *CompH_job_05* ja *CompH_job_07*. Tämän jälkeen työt sijoitetaan linjoille. Kuvasta 32 huomataan, että edellisen testiajon tapaan työt jakautuvat hyvin epätasaisesti. Inkluusion käyttö ryhmittelyssä johtaa siihen, että suurten komponenttimäärien työt sijoitetaan helposti samaan ryhmään. Tämä epätasaisuus johtaisi tasapainottamattomassa aikataulutuksessa yli 80 päivän myöhästymiseen. Yksittäisten töiden myöhästymisten summa, jota pyritään minimoimaan, on merkittävästi tätäkin suurempi.

Tasapainotuksen jälkeen saadaan taulukossa 29 (ks. liite 3) esitetyt linjakohtaiset aikataulut. Nämä aikataulut on esitetty myös kuvassa 33. Kuvasta nähdään, että algoritmi ei tuota täydellistä aikataulua. Tasapainotuksen jälkeen myöhästyneisyyttä havaitaan vain muutaman työn osalta, mutta myöhästymisten määrä on kuitenkin melko iso.

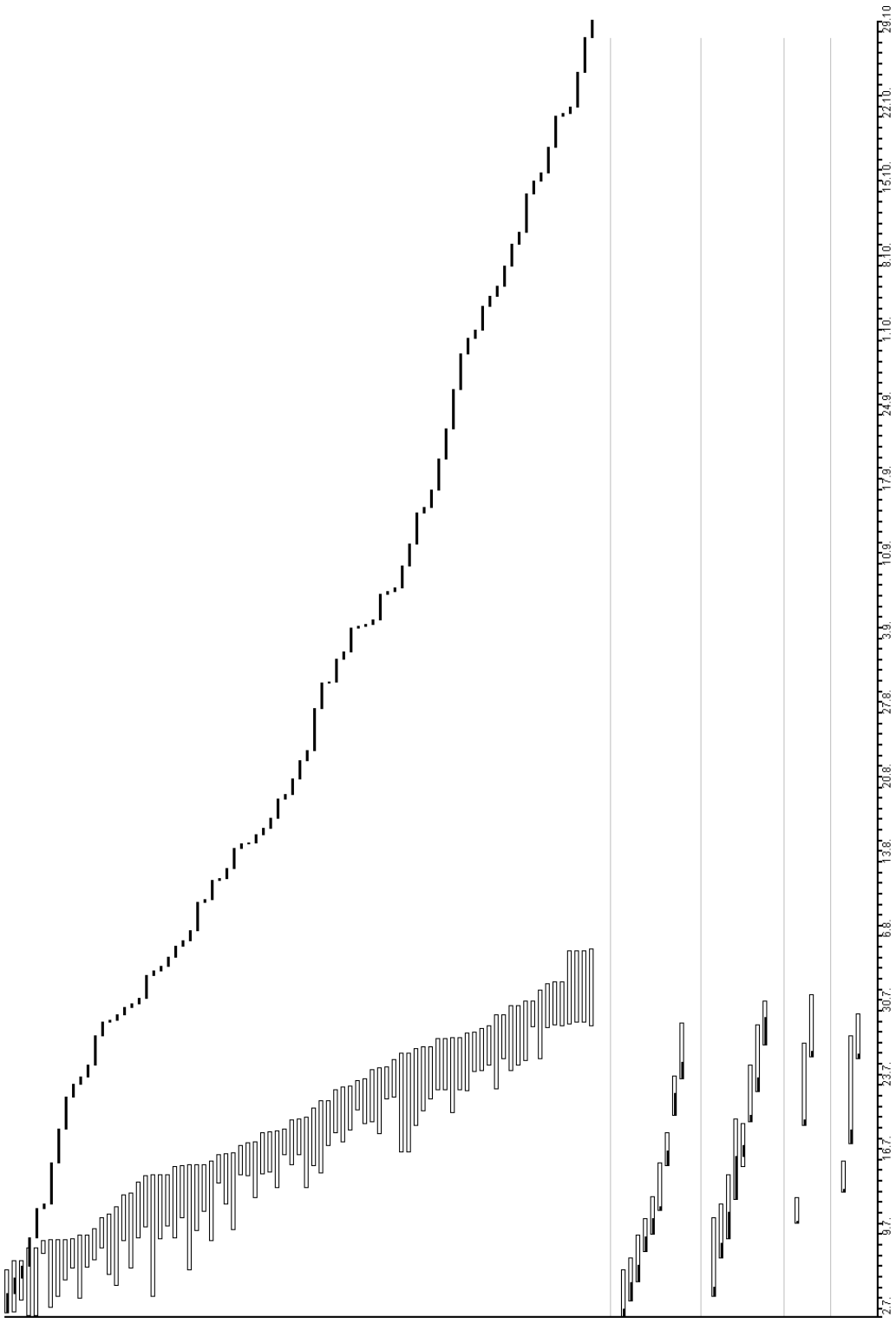
Ongelmallisimmaksi on jäänyt linja 2, jonka loppuvaiheen työt myöhästyvät paljon. Kuvaa tarkastellessa huomio kiinnittyy muutamaan kiintoisaan seikkaan. Ensinnäkin ajalla 23.–26.7. on runsaasti töitä tehtävänä. Tästä syystä myöhästymisiä on odotettavissa. Toinen kiinnostava asia on linjan 2 viimeiset työt. Näistä huomionarvoisin on ehkä kolmanneksi viimeisin työ, joka myöhästyy poikkeuksellisen pitkästä aikaikkunasta huolimatta. Kolmas seikka on linjan 1 joutoaika 18.7., jolloin linja on käyttämättä yli 19 tuntia. Kyseisen hetken jälkeen mikään linjan 2 töistä ei ala julkaisuhetkellä eli tuotannon aikaistaminen olisi ehkä mahdollista ja töiden siirto voisi näin olla perusteltua.

Kuvaan 33 on merkitty kaksi työtä, jotka molemmat ovat potentiaalisia siirtoehdokkaita. Molemmat työt ovat melko lyhyitä ja linjan 1 joutoaika osuu kummankin työn toivottuun suoritusaikaan. Työt ovat *ExpA_job_2* ja *CompG_job_03*. Tutkittaessa taulukon 23 tietoja havaitaan, että linjan 1 koneet ovat komponenttikapasiteettinsa myötä monipuolisempia linjan 2 koneisiin nähden. Näin ollen kyseiset työt olisi mahdollista suorittaa myös linjalla 1. Linjan 1 koneet ovat kuitenkin 25% hitaampia linjan 2 koneisiin nähden, mikä aiheuttaa arviolta 5–6 tunnin lisäajan työn suoritukseen. Tästä linjal-

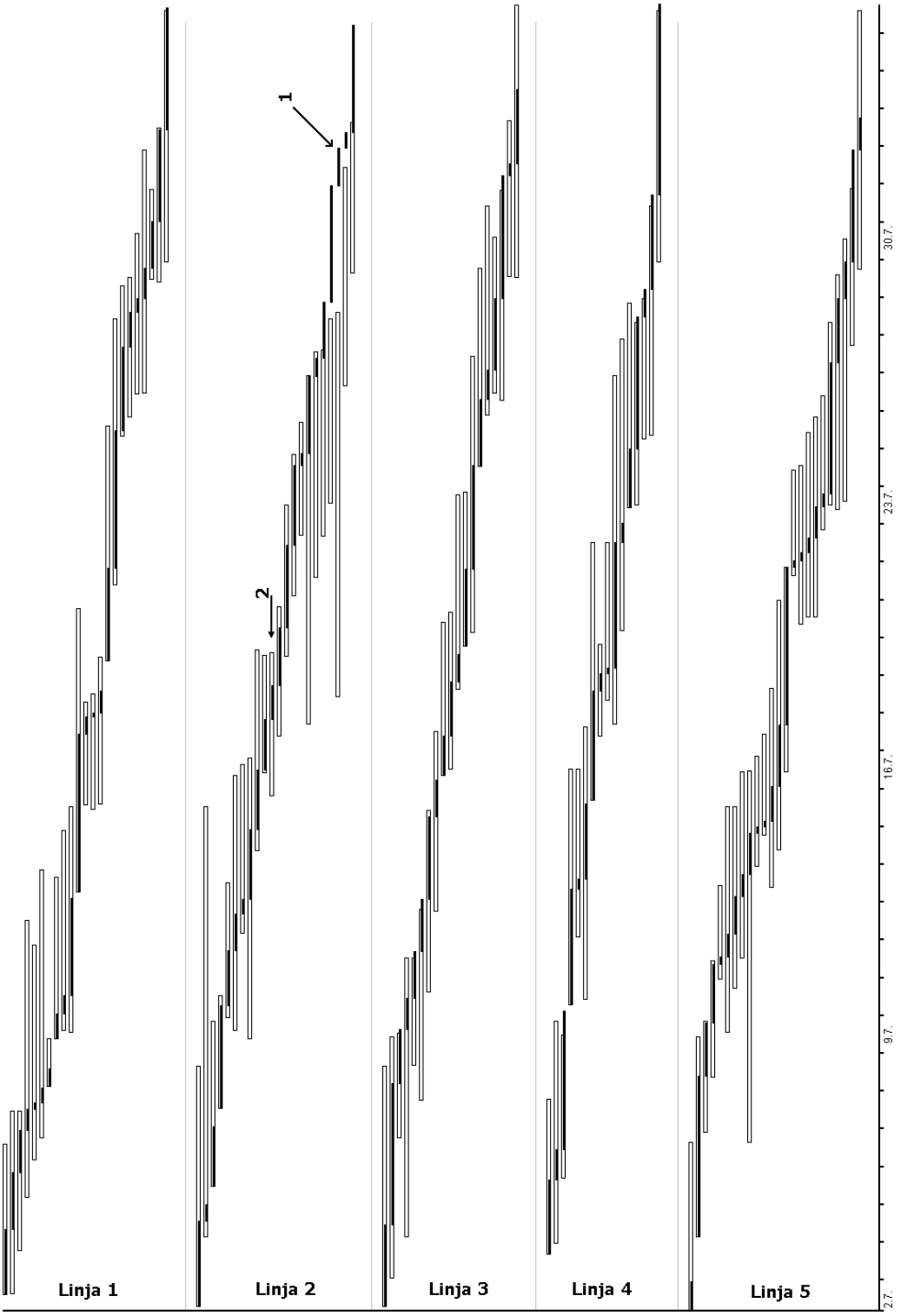
le 1 aiheutuva lisämyöhästyminen on kuitenkin hyvin pieni verrattuna yli 20 tunnin myöhästymisen pienenemiseen linjalla 2. Myös muita vastaavanlaisia tapauksia voidaan löytää kuvasta.

Taulukko 24: Linjojen töiden yhteenlasketut myöhästymiset ja joutoajat aikataulutuksen jälkeen.

	Myöhästymiset	Joutoaika
Linja 1	1 h 47 min 26 s	43 h 51 min 23 s
Linja 2	303 h 46 min 20 s	23 h 18 min 18 s
Linja 3	22 h 10 min 21 s	7 h 55 min 25 s
Linja 4	34 h 29 min 57 s	16 h 19 min 58 s
Linja 5	23 h 59 min 54 s	28 h 34 min 59 s



Kuva 32: Testiajon 2 työt ja aikataulut ennen tasapainotusta.



Kuva 33: Testiajon 2 aikataulut tasapainotettuina.

7.5 Testiajo 3

Kolmannessa testiajossa keskitytään aikataulutuksen dynaamisuuteen. Tavoitteena on tutkia, miten algoritmi ottaa huomioon kesken tuotannon lisättävät uudet työt. Jotta algoritmin toimintaa pystytään seuraamaan helpommin, suoritetaan simulaatiota pienessä mittakaavassa. Lähtökohtana käytetään ensimmäistä testiajoa niin töiden kuin linjaston suhteen. Tässä on käytössä sama 3 linjan linjasto kuin ensimmäisessä testiajossa (ks. taul. 19). Seurattavuuden vuoksi töiden kokonaismäärä pidetään melko pienenä kuten myös simuloinnin aikaväli. Töiden lisääminen suoritetaan usein (kahden päivän välein) ja simulaatiota suoritetaan lyhyt aika (noin viikon ajan).

Testiajossa käytettävät työt perustuvat edellisen tapaan taulukon 27 piirilevyihin — kaikkia ei kuitenkaan käytetä. Aloitustilanteeksi otetaan testiajo 1:n tilanne, joka on esitetty uudestaan taulukossa 25 ja kuvassa 34. Taulukosta nähdään, että ensimmänen työ aloitetaan 4.6. klo 8:30. Ensimmäiset lisätyöt lisätään seuraavan päivän aamuna klo 8:00. Tästä eteenpäin töitä lisätään joka toinen päivä klo 8:00. Kerrallaan lisättävien töiden määrä on 5–8 ja niiden ominaisuudet perustuvat pääosin satunnaisuuteen, joskin joitakin ominaisuuksia on korjattu, jotta tilanteet muistuttaisivat tosielämän tilanteita. Nämä lisättävät työt on esitetty taulukossa 26.

Seuraavassa selvitetään kuvaparien avulla, miten algoritmi suorittaa töiden aikataulutuksen. Kuvaparin yläkuva osoittaa tilanteen, johon työt lisätään, ja alakuva lisäyksen ja uuden aikataulutuksen jälkeisen tilanteen. Kukin kuva käsittää 7 päivän mittaisen aikavälin alkaen tarkastelupäivän aamusta klo 8:00. Jotta töiden sijoittelumuutokset olisivat helpommin seurattavissa, merkitään työt ensimmäisen testiajon tavoin. Samoin selkeyden vuoksi linjajärjestys pysyy samana, vaikka niiden merkinnöistä luovutaan ensimmäisten kuvien jälkeen. Aloitustilanteen työt merkitään *0A, 0B, ...*, ensimmäiset lisäykset *1A, 1B, ...* ja niin edelleen. Mikäli työ jaetaan osiin, merkitään ensimmäistä puoliskoa lisämääreellä *1* ja jälkimmäistä lisämääreellä *2*, esim. *1B2* ja *3A1*.

Taulukko 25: Linjojen aikataulut ennen töiden lisäystä.

Linja 1

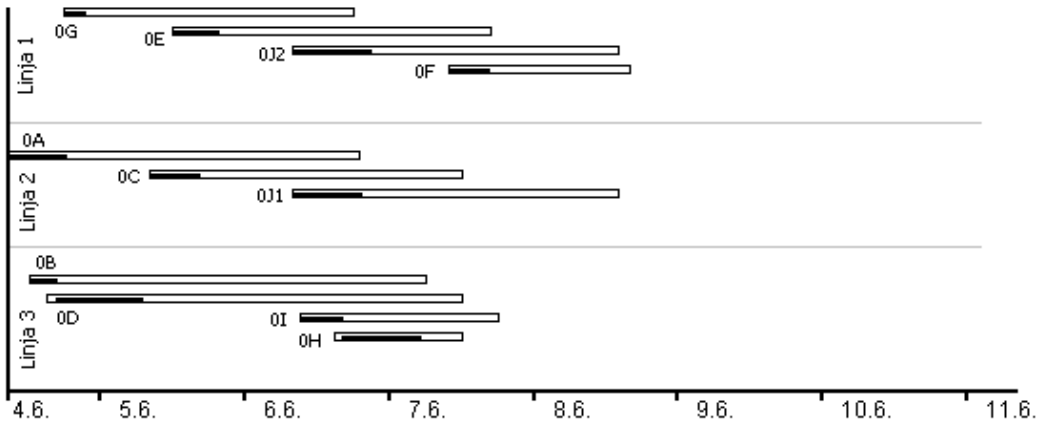
Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
(0G) Company B order 4	04.06.2012 18:00	04.06.2012 18:00	04.06.2012 21:24	06.06.2012 18:00
(0E) Company B order 2	05.06.2012 12:00	05.06.2012 12:00	05.06.2012 19:28	07.06.2012 17:00
(0J2) Company B order 7.pt2	06.06.2012 08:00	06.06.2012 08:00	06.06.2012 20:45	08.06.2012 14:00
(0F) Company B order 3	07.06.2012 10:00	07.06.2012 10:00	07.06.2012 16:29	08.06.2012 16:00

Linja 2

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
(0A) Company A order 1	04.06.2012 08:30	04.06.2012 08:30	04.06.2012 18:02	06.06.2012 19:00
(0C) Company A order 3	05.06.2012 08:00	05.06.2012 08:00	05.06.2012 16:30	07.06.2012 12:00
(0J1) Company B order 7	06.06.2012 08:00	06.06.2012 08:00	06.06.2012 19:17	08.06.2012 14:00

Linja 3

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
(0B) Company A order 2	04.06.2012 12:00	04.06.2012 12:00	04.06.2012 16:31	07.06.2012 06:00
(0D) Company B order 1	04.06.2012 15:00	04.06.2012 16:31	05.06.2012 06:55	07.06.2012 12:00
(0I) Company B order 6	06.06.2012 09:00	06.06.2012 09:00	06.06.2012 16:06	07.06.2012 18:00
(0H) Company B order 5	06.06.2012 15:00	06.06.2012 16:06	07.06.2012 05:09	07.06.2012 12:00



Kuva 34: Testiajon 3 työt ja aikataulut ennen töiden aloitusta 4.6. klo 8:30.

Taulukko 26: Lista testiajo 3:een lisättävistä töistä.

5.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(1A) Company C order 1	07.06.2012 08:00	10.06.2012 08:00	Motherboard_C	4000	1088000
(1B) Company C order 2	08.06.2012 12:00	11.06.2012 16:00	Motherboard_C.v1	800	183200
(1C) Company C order 3	09.06.2012 14:00	11.06.2012 21:20	Motherboard_C.v2	4000	1060000
(1D) Company D order 1	07.06.2012 06:00	12.06.2012 14:00	Motherboard_D	4500	2002500
(1E) Company D order 2	07.06.2012 16:00	10.06.2012 18:00	Motherboard_D.v1	3200	1296000
(1F) Company E order 1	05.06.2012 11:20	11.06.2012 09:30	Motherboard_E	3200	1984000
(1G) Company E order 2	05.06.2012 11:30	07.06.2012 22:40	Motherboard_E.v1	2500	1617500

7.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(2A) Company A order 4	08.06.2012 10:00	11.06.2012 12:00	Motherboard_A.v2	3000	723000
(2B) Company D order 3	07.06.2012 12:00	10.06.2012 22:00	Motherboard_D	7000	3115000
(2C) Company D order 4	07.06.2012 19:00	12.06.2012 09:10	Motherboard_D.v2	2500	1135000
(2D) Company D order 5	09.06.2012 18:00	13.06.2012 08:00	Motherboard_D.v1	1400	567000
(2E) Company E order 3	09.06.2012 18:00	13.06.2012 20:00	Motherboard_E.v2	2000	1108000
(2F) Company E order 4	10.06.2012 08:00	11.06.2012 15:00	Motherboard_E.v3	500	340500
(2G) Company F order 1	10.06.2012 16:00	14.06.2012 10:00	Motherboard_F	3600	306000
(2H) Company F order 2	10.06.2012 22:00	13.06.2012 09:00	Motherboard_F.v1	3800	288800

9.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(3A) Company A order 5	10.06.2012 15:00	14.06.2012 11:00	Motherboard_A.v1	6800	1584400
(3B) Company A order 6	14.06.2012 16:00	18.06.2012 18:00	Motherboard_A	3000	627000
(3C) Company B order 8	11.06.2012 15:00	14.06.2012 08:00	Motherboard_B.v4	4800	652800
(3D) Company B order 9	12.06.2012 12:00	17.06.2012 18:00	Motherboard_B	8000	1312000
(3E) Company C order 4	13.06.2012 13:30	17.06.2012 09:00	Motherboard_C.v3	4000	1160000

11.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(4A) Company C order 5	12.06.2012 08:00	15.06.2012 08:00	Motherboard_C.v2	6000	1590000
(4B) Company D order 6	13.06.2012 03:00	16.06.2012 22:00	Motherboard_D	2000	890000
(4C) Company E order 5	12.06.2012 16:30	16.06.2012 18:00	Motherboard_E	1400	868000
(4D) Company F order 3	12.06.2012 23:30	14.06.2012 10:00	Motherboard_F	5500	467500
(4E) Company F order 4	13.06.2012 12:40	14.06.2012 06:00	Motherboard_F.v2	600	46200
(4F) Company F order 5	14.06.2012 20:30	17.06.2012 22:00	Motherboard_F	12000	1020000

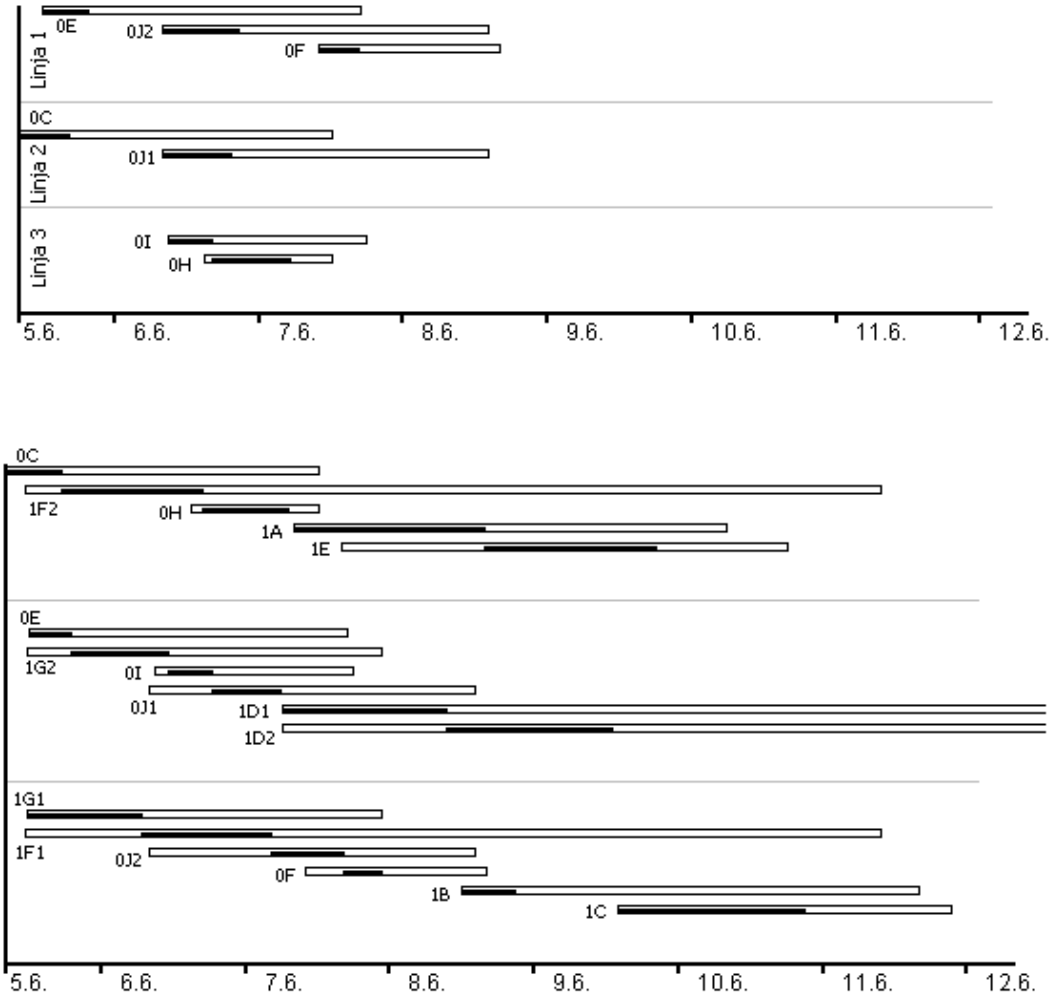
13.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(5A) Company A order 7	14.06.2012 18:00	16.06.2012 14:00	Motherboard_A	800	167200
(5B) Company A order 8	14.06.2012 21:30	16.06.2012 14:00	Motherboard_A.v2	2000	482000
(5C) Company A order 9	15.06.2012 18:00	16.06.2012 14:00	Motherboard_A.v1	1600	372800
(5D) Company B order 10	15.06.2012 18:30	18.06.2012 06:00	Motherboard_B.v4	2400	326400
(5E) Company C order 6	14.06.2012 09:00	16.06.2012 09:00	Motherboard_C	700	190400
(5F) Company C order 7	14.06.2012 09:00	16.06.2012 14:00	Motherboard_C.v1	4500	1030500
(5G) Company E order 6	15.06.2012 13:30	17.06.2012 07:15	Motherboard_E.v3	740	503940
(5H) Company E order 7	17.06.2012 18:00	19.06.2012 20:00	Motherboard_E	1200	744000

15.6. 8:00

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
(6A) Company C order 8	15.06.2012 13:00	18.06.2012 15:00	Motherboard_C.v3	2000	580000
(6B) Company D order 7	16.06.2012 23:00	20.06.2012 06:00	Motherboard_D.v2	820	372280
(6C) Company E order 8	17.06.2012 04:00	20.06.2012 10:00	Motherboard_E.v1	2500	1617500
(6D) Company F order 8	15.06.2012 15:00	19.06.2012 18:00	Motherboard_F	2400	204000
(6E) Company F order 9	16.06.2012 19:30	18.06.2012 18:00	Motherboard_F.v1	2300	174800

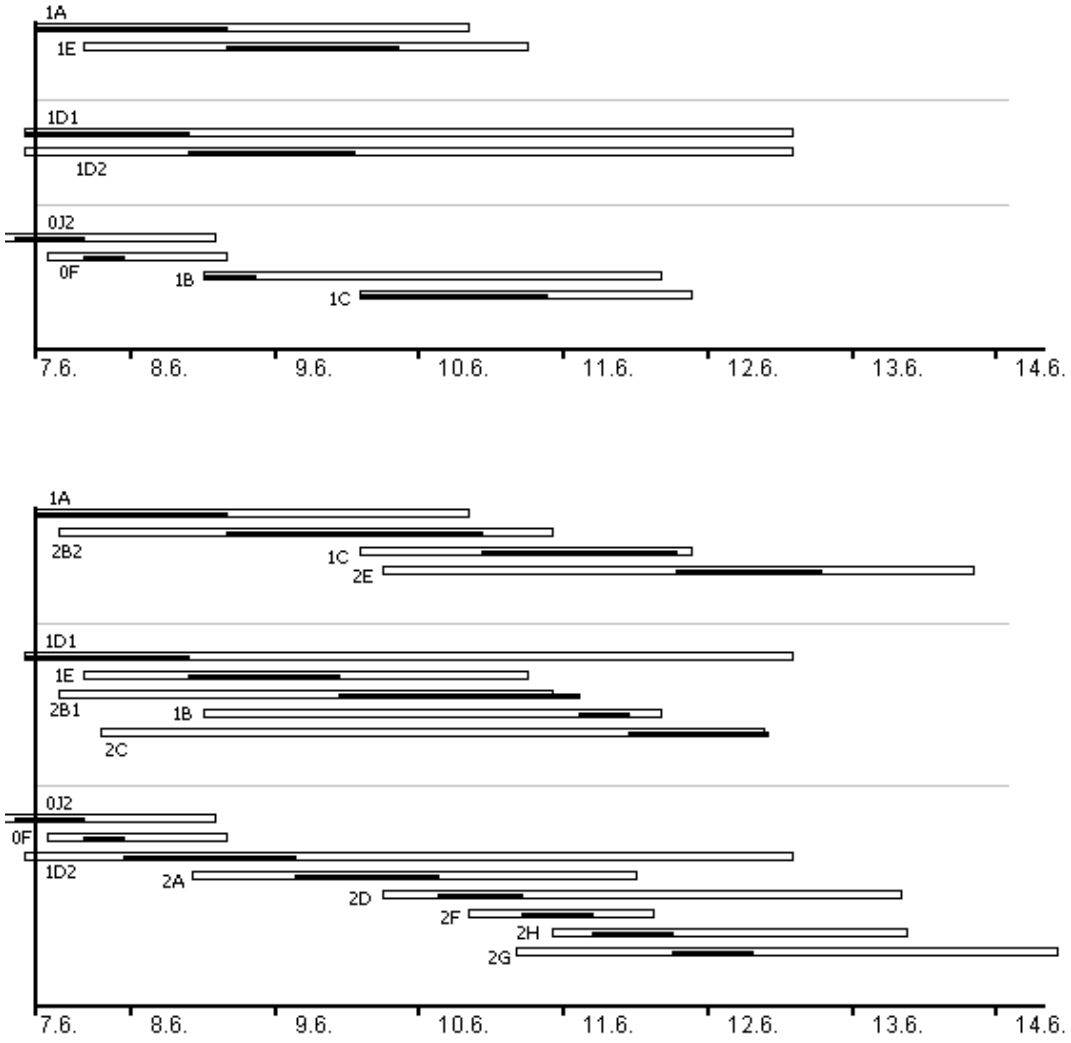
*KKM – Komponenttien kokonaismäärä



Kuva 35: Ensimmäisten töiden lisäys ja aikataulutus 5.6.

Kuvan 35 yläkuva näyttää tilanteen 5.6. klo 8:00. Kun aikataulutettavaksi lisätään ensimmäiset seitsemän työtä, saadaan töiden järjestelyn jälkeen alakuvan mukainen aikataulu. On syytä huomata, että työ *0C* on aikataulutettu alkavaksi klo 8:00 ja sitä ei ole vielä merkitty aloitetuksi. Tästä syystä se on mahdollista siirtää linjalta toiselle.

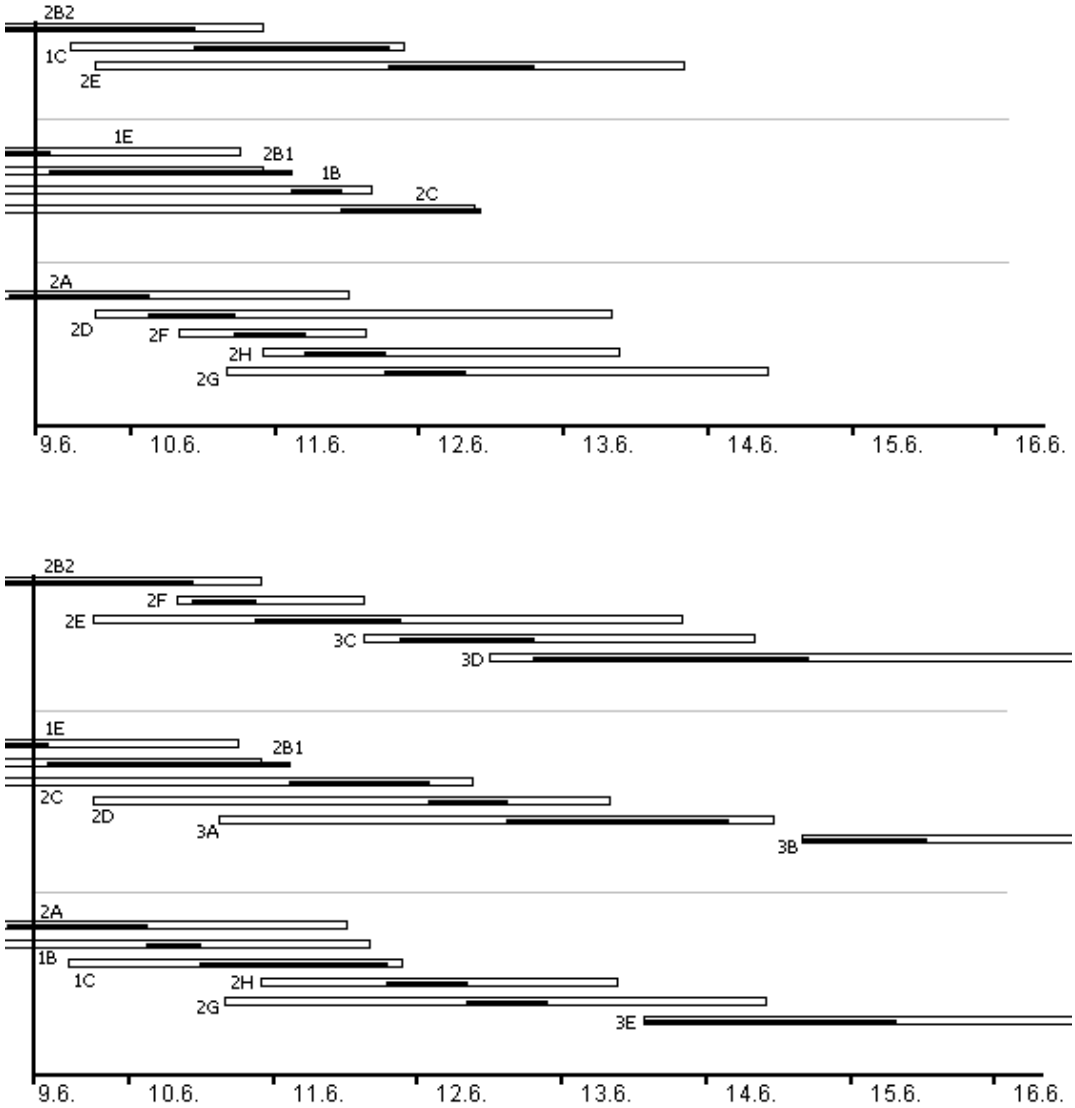
Lisättävistä töistä kolme on ositettu kahteen osaan (*1D*, *1F* ja *1G*). Näistä kahden osat on jaettu eri linjoille, kun yhden molemmat puolet pysyvät samalla linjalla — ainakin tässä vaiheessa.



Kuva 36: Aikataulut toisen lisäyksen yhteydessä 7.6.

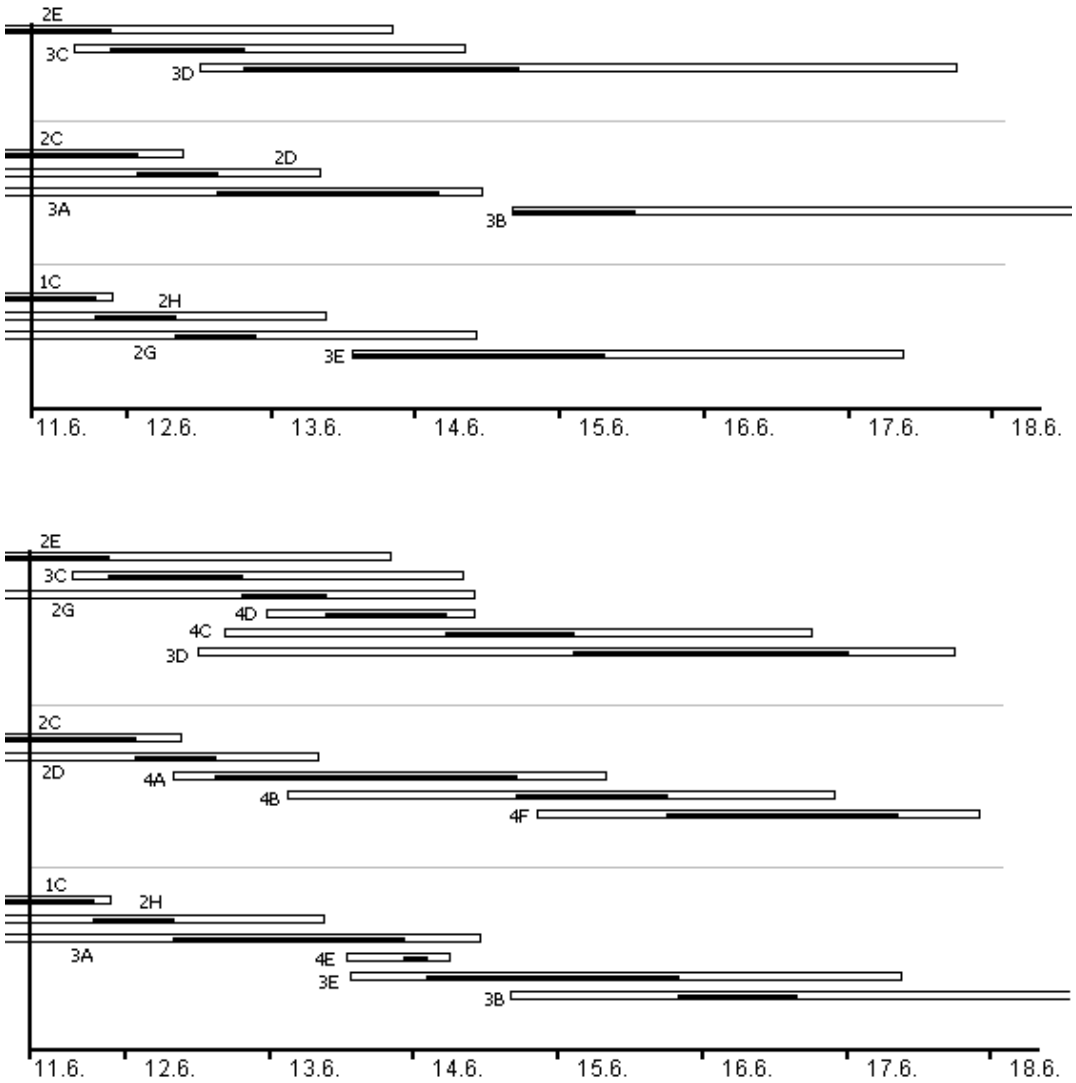
Kuvan 36 yläkuva näyttää tilanteen aamulla 7.6. klo 8:00. Nyt jokaisella linjalla on työ tekeillä. Näitä töitä ei tulla siirtämään aikataulutuksen aikana.

Töiden lisäämisen ja aikataulutuksen jälkeen havaitaan, että linjan 3 joutoajat on saatu käytettyä hyödyksi. Myös työn *1D* puoliskot ovat nyt sijoitettuina eri linjoille. Lisäksi havaitaan ensimmäiset myöhästymiset. Toisella linjalla kaksi työtä (*2B1* ja *2C*) on myöhässä yhteensä yli 5 tuntia. Silmämääräisesti arvioituna osa linjan 3 töistä olisi siirrettävissä myöhemmäksi. Toisin sanoen tasapainotuksen toimivuudessa olisi tämän mukaan parantamisen varaa.



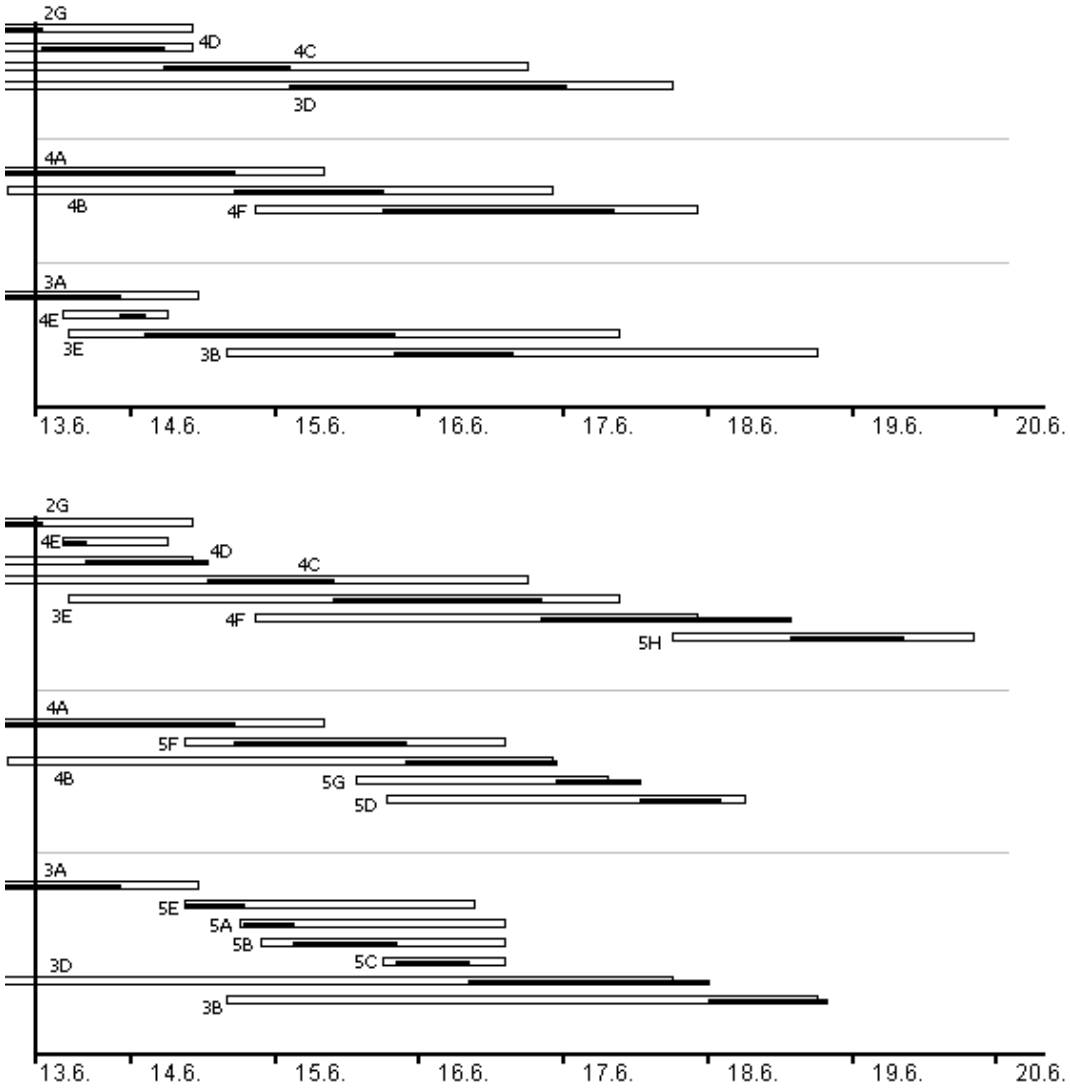
Kuva 37: Aikataulut 9.6.

9.6. uusien töiden myötä suoritettava aikataulutus poistaa työn *2C* myöhästymisen siirtämälle sen toiselle linjalle (kuva 37). Muutenkin uusi aikataulu näyttää väljemmältä, mutta syynä on todennäköisesti se, että tässä vaiheessa lisättiin ainoastaan viisi työtä.



Kuva 38: Aikataulut 11.6.

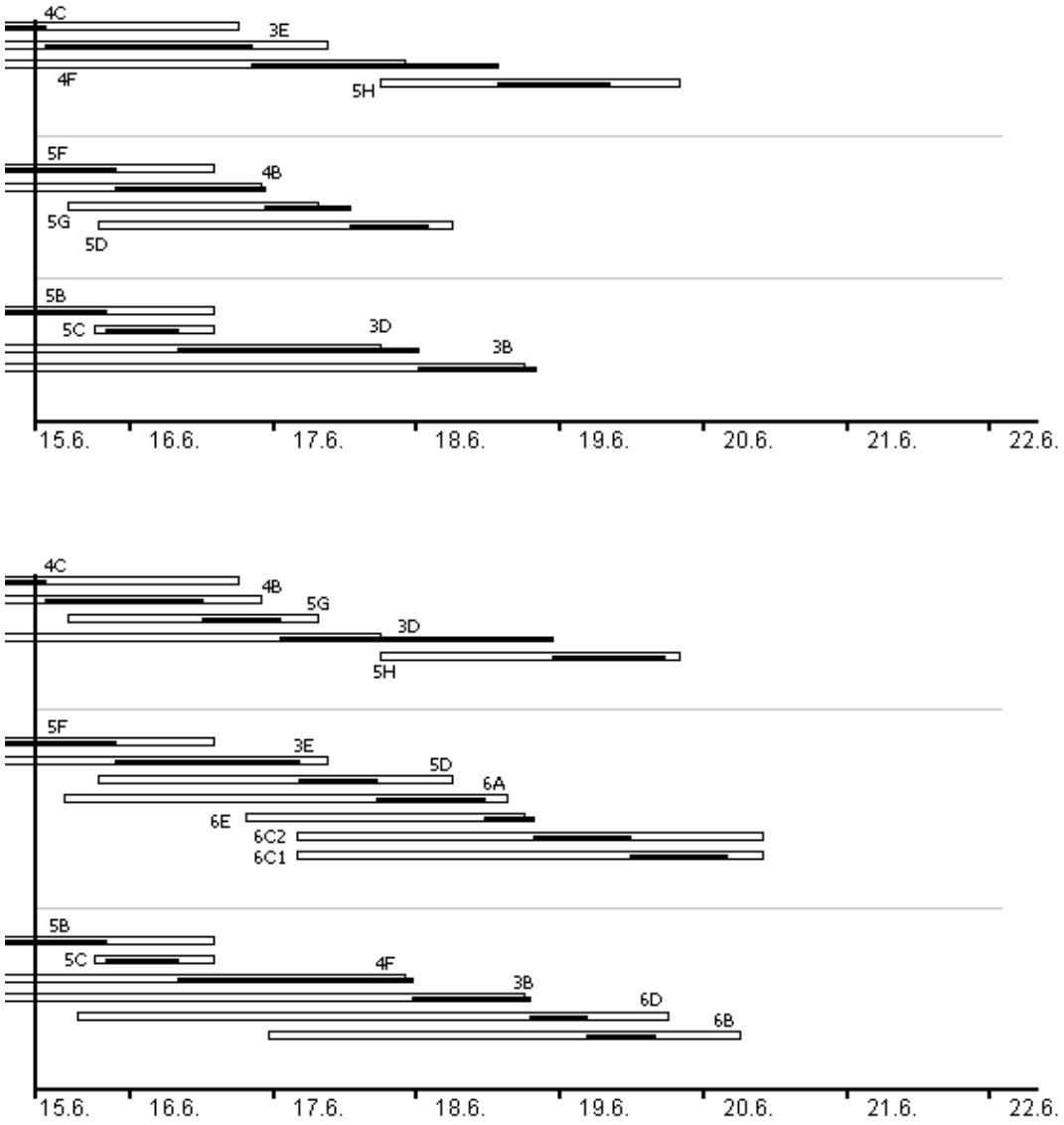
Kuvassa 38 lisätään kuusi työtä edellä luotuun väljään aikatauluun. Tuloksena on toimiva aikataulutus ilman myöhästymisiä.



Kuva 39: Aikataulut viidennen lisäyksen jälkeen 13.6.

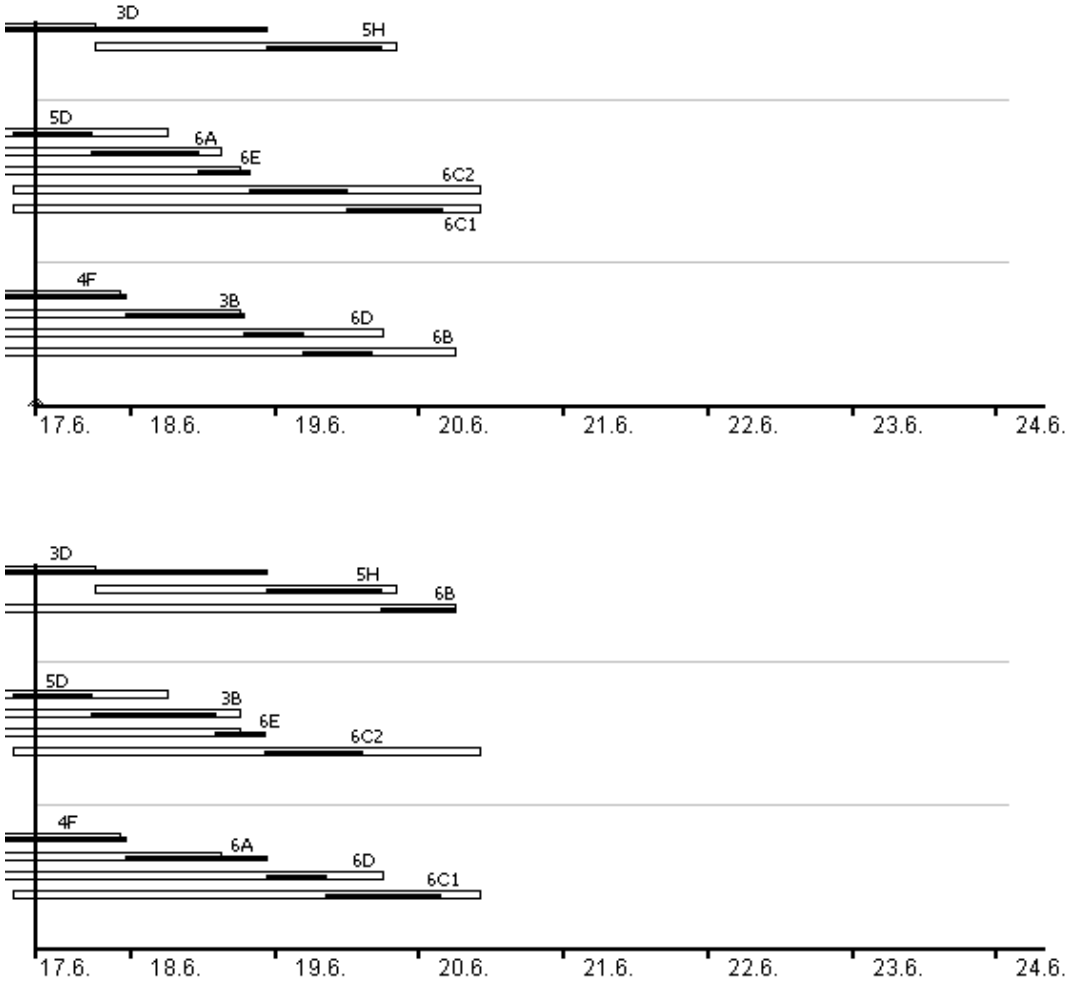
Kuva 39 osoittaa, kuinka 13.6. lisättävät 8 työtä aiheuttavat ongelmia algoritmille. Lisättävät työt eivät ole erityisen kiireellisiä, mutta niiden tuotantoikkunat osuvat paljolti päällekkäin sekä toistensa että jo aikataulutettujen töiden kanssa. Myöhästymisiä voidaan siis odottaa, mutta seurauksena on yhteensä jopa 32 tuntia myöhästymisiä.

Huomiota herättää työn *4E* siirto linjalta 3 linjalle 1. Tämä siirto jättää linjalle 3 joutoaikaa, mutta aiheuttaa samalla myöhästymistä linjalla 1.



Kuva 40: Viimeisen töiden lisäys 15.6.

Viimeisessä lisäyksessä aikataulutettavaksi lisätään 5 työtä. Nämä viimeiset työt on suoritettava melko nopeasti, mikä johtaa myöhästymisten kasvuun melkein 3 tunnilla. Myöhästymiset keskittyvät linjalle 1, mutta linjojen työmäärät on melko hyvin tasattu.

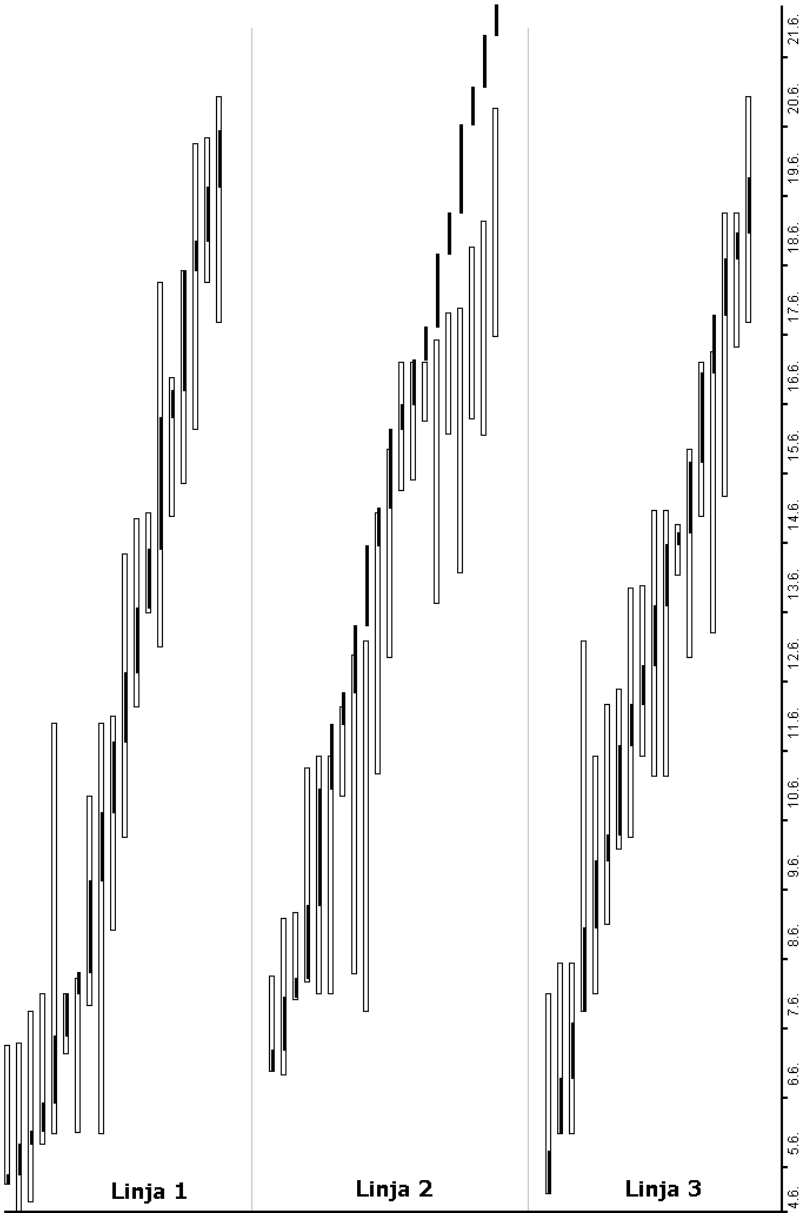


Kuva 41: Aikataulut uusinta-aikataulutuksen jälkeen.

17.6. suoritetaan vielä yksi aikataulutus, jotta aloittamattomien töiden myöhästymistä voitaisiin pienentää. Itse asiassa vaikutus on päinvastainen: kokonaismyöhästyminen kasvaa yli 9 tunnilla.

Algoritmia tutkimalla tähän löydetään kaksi mahdollista syytä. Ensinnäkin uutta linjastoaikataulua ei verrata vanhaan, vaan sen oletetaan olevan edellistä parempi. Kun selvitetään työn siirron kannattavuutta, aikataulujen vertailu tehdään vain linjakohtaisesti. Toinen syy on työn siirron kohdelinjan valintaperuste, joka on linjan tuotantoajan määrä. Ensisijaisena valintana tulisi käyttää linjaa, jolla on vähiten myöhästymisiä.

Suoritetaan vielä yksi vertailu. Kun työt lisätään erissä uuden aikataulu-



Kuva 42: Luotu aikataulu, kun kaikki työt aikataulutetaan kerralla

tuksen kera, kuten edellä, viimeinen työ valmistuu 20.6. Kuvassa 42 nähdään tulos, jos kaikki testiajon työt aikataulutettaisiin yhdellä kertaa. Huomataan, että aikatauluttamalla kaikki kerralla viimeinen työ valmistuu vasta 21.6. Lisäksi linjalla 2 havaitaan aikataulun lopussa useita töitä, jotka voitaisiin siir-

tää esim. linjalle 3. Tähän huonoon tulokseen on syynä se, että kun linja merkitään käsitellyksi, se merkitään pysyvästi. Kun linja 2 on merkitty käsitellyksi ja se on silti potentiaalisena kohdelinjana, saattaa töitä kerääntyä kuvan osoittamalla tavalla. On selvää, että aikataulut kannattaa päivittää aika ajoin, myös tuotannon ollessa käynnissä.

7.6 Ohjelma ja tulosten analysointi

Testiajot on suoritettu erikseen tätä tutkielmaa varten tehdyllä simulointiohjelmalla. Ohjelmointikielenä on käytetty Javaa. Testiajot on ajettu Windows XP -käyttöjärjestelmällä käyttäen AMD Athlon 64 -prosessoria (X2 4200+ 2.2 GHz). Testiajoissa käytetyn keskusmuistin määrä on ollut 2GB ja ohjelma on ajettu Eclipse-kehitysympäristössä.

Kuten tasapainotusongelman luonteesta voi olettaa, testiajojen simulointiajat kasvavat runsaasti kaytettävän testidatan ja käytettävien linjojen lisääntyessä. Testiajo 1:n simulointiaika oli n. 3 sekuntia (10 työtä ja 3 linjaa), kun testiajo 2:n aika oli yli 2 minuuttia (100 työtä ja 5 linjaa).

Testiajoja tutkimalla on selvää, että edellä kehitetty algoritmi ei ole täydellinen. Edellisessä luvussa ja luvussa 6.2.4 löydettiin monta parannettavaa seikkaa. Huomion arvoista on se, miten käyttökelpoisia aikataulut ovat algoritmin yksinkertaisuuteen nähden, sillä algoritmin kehitykseen on käytetty vain yksinkertaista matematiikkaa yksinkertaisen logiikan apuna.

Ajamalla lisää testiajoja, voidaan algoritmista todennäköisesti löytää myös muita heikkoja kohtia. Tässä käytetyt skenaariot, jotka ovat melko todenmukaisia, osoittavat kuitenkin algoritmin toimivan suhteellisen hyvin.

8 Johtopäätökset

Tässä tutkielmassa luotiin yleiskatsaus nykyajan piirilevyladonnan tuotantotekniikoihin ja ongelmiin sekä kehitettiin yksinkertainen menetelmä ja algoritmi ladontalinjojen tasapainotukseen. Ensin tutkittiin itse ladontaprosessia sekä erilaisia tuotantotapoja ja töidenjärjestelyjä. Luvussa 3 keskityttiin tuotantolinjojen työmäärien tasapainotukseen käyden läpi erilaisia linjastoja ja muita ladonnan tasapainotukseen liittyviä seikkoja. Neljännessä luvussa tutkittiin tasapainotusongelmaan kehitettyjä ratkaisumenetelmiä. Luvussa 5 käytiin läpi, miten näitä ratkaisumenetelmiä voidaan soveltaa käytäntöön. Tutkiminen aloitettiin yksinkertaisesta yhden linjan tapauksesta edeten monen linjan tasapainotukseen, jossa töiden julkaisu- ja määräajat on otettava myös huomioon. Avuksi otettiin esimerkkejä, joiden avulla pystyttiin paremmin kuvaamaan ongelmaa.

Luvussa 6 kehitettiin algoritmi melko yksinkertaisesti mallinnettuun tuotantojärjestelmään. Vaikka lähtökohdaksi valittiin heuristinen logiikkaan perustuva menetelmä, ei geneettisen algoritmin käyttöä unohdettu kokonaan. Kehitetyn algoritmin toiminnallisuutta testattiin luvussa 7 kolmen testitapauksen avulla. Ensin tutkittiin tasapainotuksen toimintaa pienellä työmäärällä käyttäen pientä linjastoa. Sitten selvitettiin algoritmin toiminnallisuus suurella työmäärällä. Lopuksi arvioitiin algoritmin dynaamisuutta lisäten työmäärään lisää töitä kahden päivän välein ja tutkimalla näin saatuja aikatauluja. Kehitetty algoritmi osoittautui yksinkertaisuudestaan huolimatta mukautuvaksi ja tuloksiltaan suhteellisen hyväksi.

Ladontatöiden tasapainotusongelma on monitahoinen ongelma, jota on tutkittu paljon, ja jonka ratkaisemiseksi on kehitetty runsaasti erilaisia tekniikoita. Voidaan ehkä sanoa, että ladontalinjojen tasapainotusongelma on jollain tasolla ratkaistu tyydyttävästi. Tulevaisuuden haaste on luoda yhä parempia ratkaisumenetelmiä.

Viitteet

[Bal11] Savas Balin. Non-identical parallel machine scheduling using genetic algorithm. *Expert Systems with Applications*, 38:6814–6821, 2011.

[BFS06] N. Boysen, M. Fliedner, and A. Scholl. A classification of assembly line balancing problems. *Jenaer Schriften zur Wirtschaftswissenschaft*, (12), 2006.

[BP01] Alexander Bockmayr and Nicolai Pisaruk. Solving assembly line balancing problems by combining IP and CP. *Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001. Cornell University Library (arXiv.org: cs/0106002v2).

[BP06] Alexander Bockmayr and Nicolai Pisaruk. Detecting infeasibility and generating cuts for mixed integer programming using constraint. *Computers & Operation Research*, 33:2777–2786, 2006.

[BS06] C. Becker and A. Scholl. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168:694–715, 2006.

[CC03] Wei-Shing Chen and Chiuh-Cheng Chyu. A minimum setup strategy for sequencing PCBs with multi-slot feeders. *Integrated Manufacturing Systems*, 14/3:255–267, 2003.

[CWR+07] Nilanjan Choudhury, Eilbert Wilhelm, Brijesh Rao, Jonathan Gott, and Nikhilesh Khotekar. Process planning for circuit card assembly on a series of dual head placement machines. *European Journal of Operational Research*, 182:626–639, 2007.

[KBAK99] John R. Koza, Forrest H. Bennet, David Andre, and Martin A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufman Publishers, 1999.

[KHJN02] Timo Knuutila, Mika Hirvikorpi, Mika Johnsson, and Olli Nevalainen. Grouping PCB assembly jobs with typed component feeder units. *TUCS Technical Report No 460*, 2002.

[KT06] J. Kleinberg and É. Tardos. *Algorithm Design*. Addison Wesley, 2006.

[MNP99] C. Merengo, F. Nava, and A. Pozzetti. Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835–2860, 1999.

[Pin05] Michael L. Pinedo. *Planning and Scheduling in Manufacturing and Services*. Springer New York, 2005.

[RK06] Patricia A. Randall and Mary E. Kurz. Effectiveness of adaptive crossover procedures for a genetic algorithm to schedule unrelated parallel machines with setups. *International Journal of Operations Research*, 4(1):1–10, 2006.

[Saw02] Tadeusz Sawik. Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. *European Journal of Operational Research*, 143:115–124, 2002.

[SB06] Armin Scholl and Christian Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168:666–693, 2006.

[SJP+98] Jouni Smed, Mika Johnsson, Mikko Puranen, Timo Leipälä, and Olli Nevalainen. Job grouping in surface mounted component printing. *TUCS Technical Report No 196*, 1998.

[SR93] S.M. Shafer and D.F. Rogers. Similarity and distance measures for cellular manufacturing. part I. A survey. *International Journal of Production Research*, 31(5):1133–1142, 1993.

[SSJ+03] Jouni Smed, Kari Salonen, Mika Johnsson, Tommi Johtela, and Olli S. Nevalainen. Grouping PCBs with minimum feeder chan-

ges. *The International Journal of Flexible Manufacturing Systems*, 15:19–35, 2003.

[SSJN06] K. Salonen, J. Smed, M. Johnsson, and O. Nevalainen. Grouping and sequencing PCB assembly jobs with minimum feeder setups. *Robotics and Computer-Integrated Manufacturing*, 22:297–305, 2006.

[VHN03] N. Van Hop and N.N. Nagarur. The scheduling problem of pcbs for multiple non-identical parallel machines. *European Journal of Operational Research 158*, pages 577–594, 2003.

[YGGY07] I.O. Yilmaz, M. Grunow, H.O. Günther, and C. Yapan. Development of group setup strategies for makespan minimisation in PCB assembly. *International Journal of Production Research*, 45(4):871–897, 2007.

Liite 1: Testiajo 2:n piirilevyt

Taulukko 27: Testiajon töissä käytettävät piirilevyt.

Levy	Komponenttien määrä	Komponenttityyppien määrä
Motherboard_A	209	46
Motherboard_A.v1	233	44
Motherboard_A.v2	241	44
Motherboard_B	164	52
Motherboard_B.v1	244	60
Motherboard_B.v2	165	46
Motherboard_B.v3	156	45
Motherboard_B.v4	136	38
Motherboard_C	272	54
Motherboard_C.v1	229	43
Motherboard_C.v2	265	52
Motherboard_C.v3	290	57
Motherboard_D	445	52
Motherboard_D.v1	405	46
Motherboard_D.v2	454	47
Motherboard_E	620	56
Motherboard_E.v1	647	56
Motherboard_E.v2	554	55
Motherboard_E.v3	681	61
Motherboard_E.v4	540	52
Motherboard_F	85	25
Motherboard_F.v1	76	23
Motherboard_F.v2	77	22
Motherboard_G	989	50
Motherboard_G.v1	998	54
Motherboard_G.v2	992	54
Motherboard_H	1212	65
Motherboard_H.v1	1267	70
Motherboard_H.v2	1268	73
ExpBoard_A	59	20
ExpBoard_B	64	26
ExpBoard_C	60	27
ExpBoard_D	48	17
ExpBoard_E	38	13
ExpBoard_F	96	28
Powerboard_A	128	24
Powerboard_B	100	25
Powerboard_C	73	17

Liite 2: Testiajo 2:n työt

Taulukko 28: Lista testiajo 2:n aikataulutettavista töistä.

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
CompA_job_01	04.07.2012 02:50	11.07.2012 12:00	Motherboard_A	3000	627000
CompA_job_02	07.07.2012 17:50	12.07.2012 19:00	Motherboard_A.v1	5800	1351400
CompA_job_03	09.07.2012 13:00	15.07.2012 12:00	Motherboard_A.v2	8400	2024400
CompA_job_04	13.07.2012 06:00	20.07.2012 18:00	Motherboard_A.v2	14000	3374000
CompA_job_05	16.07.2012 08:00	20.07.2012 09:00	Motherboard_A	4000	836000
CompA_job_06	20.07.2012 13:00	25.07.2012 19:30	Motherboard_A.v1	2000	466000
CompA_job_07	23.07.2012 09:00	29.07.2012 14:00	Motherboard_A	5000	1045000
CompA_job_08	27.07.2012 17:15	31.07.2012 21:00	Motherboard_A.v2	8800	2120800
CompB_job_01	02.07.2012 04:00	06.07.2012 15:00	Motherboard_B	2560	419840
CompB_job_02	03.07.2012 16:00	07.07.2012 18:00	Motherboard_B.v1	6000	1464000
CompB_job_03	05.07.2012 10:40	09.07.2012 20:00	Motherboard_B.v2	7300	1204500
CompB_job_04	08.07.2012 08:20	11.07.2012 10:00	Motherboard_B.v3	5800	904800
CompB_job_05	09.07.2012 22:00	13.07.2012 12:00	Motherboard_B	6000	984000
CompB_job_06	12.07.2012 03:50	16.07.2012 15:00	Motherboard_B.v4	1400	190400
CompB_job_07	16.07.2012 09:30	19.07.2012 12:00	Motherboard_B.v1	4500	1098000
CompB_job_08	21.07.2012 02:00	24.07.2012 20:00	Motherboard_B.v3	10000	1560000
CompB_job_09	24.07.2012 12:20	29.07.2012 18:00	Motherboard_B.v2	7500	1237500
CompB_job_10	25.07.2012 21:00	31.07.2012 10:00	Motherboard_G.v2	1300	1289600
CompC_job_01	02.07.2012 15:00	07.07.2012 11:00	Motherboard_C	4600	1251200
CompC_job_02	03.07.2012 18:10	07.07.2012 11:00	Motherboard_C.v1	3575	818675
CompC_job_03	06.07.2012 15:00	16.07.2012 11:00	Motherboard_C.v2	2510	665150
CompC_job_04	09.07.2012 14:00	16.07.2012 08:00	Motherboard_C.v3	2880	835200
CompC_job_05	13.07.2012 08:50	18.07.2012 15:00	Motherboard_C.v2	2030	537950
CompC_job_06	17.07.2012 09:00	20.07.2012 19:00	Motherboard_C.v3	5000	1450000
CompC_job_07	19.07.2012 09:00	25.07.2012 14:00	Motherboard_C	8000	2176000
CompC_job_08	23.07.2012 10:00	28.07.2012 20:00	Motherboard_C.v1	4800	1099200
CompC_job_09	25.07.2012 08:00	31.07.2012 10:00	Motherboard_C.v2	6400	1696000
CompC_job_10	29.07.2012 12:10	05.08.2012 17:30	Motherboard_C.v3	4600	1334000
CompD_job_01	03.07.2012 00:40	09.07.2012 10:00	Motherboard_D	8000	3560000
CompD_job_02	06.07.2012 21:20	09.07.2012 20:00	Motherboard_D.v1	3000	1215000
CompD_job_03	08.07.2012 02:20	09.07.2012 09:00	Motherboard_D.v2	750	340500
CompD_job_04	14.07.2012 08:00	19.07.2012 16:00	Motherboard_D.v1	5200	2106000
CompD_job_05	16.07.2012 12:00	20.07.2012 15:30	Motherboard_D	2800	1246000
CompD_job_06	20.07.2012 03:00	27.07.2012 10:00	Motherboard_D.v2	7000	3178000
CompD_job_07	22.07.2012 16:00	27.07.2012 14:00	Motherboard_D.v1	4800	1944000
CompD_job_08	25.07.2012 07:20	29.07.2012 07:00	Motherboard_D	4480	1993600
CompD_job_09	29.07.2012 22:00	05.08.2012 14:00	Motherboard_H.v2	8500	10778000
CompE_job_01	02.07.2012 14:00	06.07.2012 14:00	Motherboard_E	3200	1984000
CompE_job_02	05.07.2012 16:10	09.07.2012 11:00	Motherboard_E.v1	5600	3623200
CompE_job_03	06.07.2012 17:40	09.07.2012 12:00	Motherboard_E.v2	2500	1385000
CompE_job_04	08.07.2012 16:00	11.07.2012 12:00	Motherboard_E.v1	2200	1423400
CompE_job_05	11.07.2012 12:00	16.07.2012 10:00	Motherboard_E	640	396800
CompE_job_06	14.07.2012 08:30	20.07.2012 23:00	Motherboard_E.v3	1700	1157700
CompE_job_07	16.07.2012 10:10	21.07.2012 20:00	Motherboard_E.v4	8200	4428000
CompE_job_08	19.07.2012 18:00	23.07.2012 20:00	Motherboard_E	4200	2604000
CompE_job_09	23.07.2012 13:00	28.07.2012 10:00	Motherboard_E.v2	8000	4432000
CompE_job_10	26.07.2012 06:20	31.07.2012 19:30	Motherboard_E	7000	4340000

*KKM – Komponenttien kokonaismäärä

Työ	Julkaisuaika	Takaraja	Levy	Levy kpl	KKM*
CompF_job_01	03.07.2012 23:00	09.07.2012 20:00	Motherboard_F	4500	382500
CompF_job_02	06.07.2012 17:50	13.07.2012 20:00	Motherboard_F.v1	2400	182400
CompF_job_03	10.07.2012 06:20	16.07.2012 12:00	Motherboard_F	19000	1615000
CompF_job_04	10.07.2012 17:00	15.07.2012 12:00	Motherboard_F.v2	6000	462000
CompF_job_05	12.07.2012 18:00	17.07.2012 12:00	Motherboard_F.v1	7200	547200
CompF_job_06	17.07.2012 09:00	19.07.2012 19:00	Motherboard_F	2300	195500
CompF_job_07	20.07.2012 08:00	24.07.2012 13:00	Motherboard_F.v2	900	69300
CompF_job_08	23.07.2012 14:20	30.07.2012 13:00	Motherboard_F.v1	7200	547200
CompF_job_09	29.07.2012 12:40	02.08.2012 16:00	Motherboard_F	1800	153000
CompG_job_01	04.07.2012 03:00	09.07.2012 10:00	Motherboard_G	5000	4935000
CompG_job_02	10.07.2012 09:40	17.07.2012 15:00	Motherboard_G.v1	2940	2910600
CompG_job_03	15.07.2012 18:50	19.07.2012 14:00	Motherboard_G.v2	1500	1419000
CompG_job_04	19.07.2012 11:40	23.07.2012 12:00	Motherboard_G.v1	3400	3366000
CompG_job_05	22.07.2012 20:00	26.07.2012 09:00	Motherboard_G.v2	240	227040
CompG_job_06	25.07.2012 20:00	29.07.2012 12:00	Motherboard_G	1200	1184400
CompG_job_07	26.07.2012 11:00	30.07.2012 14:00	Motherboard_G.v1	3000	2970000
CompG_job_08	29.07.2012 09:30	02.08.2012 11:30	Motherboard_G	4000	3956000
CompH_job_01	02.07.2012 06:20	08.07.2012 15:00	Motherboard_H	6500	7878000
CompH_job_02	07.07.2012 12:40	10.07.2012 12:00	Motherboard_H.v1	3800	4814600
CompH_job_03	10.07.2012 14:30	15.07.2012 10:00	Motherboard_H.v2	2750	3487000
CompH_job_04	15.07.2012 16:20	22.07.2012 12:00	Motherboard_H	2850	3454200
CompH_job_05	17.07.2012 16:40	26.07.2012 22:00	Motherboard_H.v1	5500	6968500
CompH_job_06	21.07.2012 09:00	28.07.2012 10:00	Motherboard_H.v2	4800	6086400
CompH_job_07	23.07.2012 12:00	28.07.2012 08:00	Motherboard_H	7000	8484000
CompH_job_08	29.07.2012 15:00	02.08.2012 15:00	Motherboard_H.v1	4000	5068000
ExpA_job_1	13.07.2012 22:30	16.07.2012 20:00	Expboard_A	850	50150
ExpA_job_2	18.07.2012 10:00	28.07.2012 14:00	Expboard_A	10000	590000
ExpA_job_3	26.07.2012 10:00	30.07.2012 16:00	Expboard_A	2900	171100
ExpB_job_1	12.07.2012 01:20	16.07.2012 12:00	Expboard_B	1300	83200
ExpB_job_2	20.07.2012 13:00	25.07.2012 10:00	Expboard_B	2400	153600
ExpB_job_3	26.07.2012 11:00	01.08.2012 21:00	Expboard_B	6000	384000
ExpC_job_1	10.07.2012 22:30	13.07.2012 10:00	Expboard_C	700	42000
ExpC_job_2	20.07.2012 04:00	27.07.2012 21:00	Expboard_C	2830	169800
ExpC_job_3	26.07.2012 15:30	01.08.2012 10:00	Expboard_C	2500	150000
ExpD_job_1	15.07.2012 10:40	18.07.2012 12:00	Expboard_D	450	21600
ExpD_job_2	21.07.2012 15:00	24.07.2012 10:00	Expboard_D	1210	58080
ExpE_job_1	14.07.2012 17:50	17.07.2012 10:00	Expboard_E	1100	41800
ExpE_job_2	18.07.2012 07:30	22.07.2012 12:00	Expboard_E	1200	45600
ExpE_job_3	04.07.2012 03:00	15.07.2012 12:00	Expboard_E	4820	183160
ExpF_job_1	15.07.2012 14:00	19.07.2012 11:00	Expboard_F	3500	336000
ExpF_job_2	18.07.2012 14:30	23.07.2012 18:00	Expboard_F	4200	403200
PsuA_job_1	09.07.2012 09:00	16.07.2012 19:00	Powerboard_A	12000	1536000
PsuA_job_2	09.07.2012 13:00	15.07.2012 12:00	Powerboard_A	2300	294400
PsuA_job_3	22.07.2012 16:40	25.07.2012 16:00	Powerboard_A	1460	186880
PsuA_job_4	29.07.2012 11:00	31.07.2012 20:00	Powerboard_A	8000	1024000
PsuB_job_1	05.07.2012 04:00	12.07.2012 12:00	Powerboard_B	3300	330000
PsuB_job_2	09.07.2012 09:00	13.07.2012 15:00	Powerboard_B	4000	400000
PsuB_job_3	21.07.2012 13:30	27.07.2012 13:00	Powerboard_B	2550	255000
PsuB_job_4	29.07.2012 17:15	05.08.2012 14:00	Powerboard_B	3500	350000
PsuC_job_1	06.07.2012 04:00	11.07.2012 20:00	Powerboard_C	850	62050
PsuC_job_2	09.07.2012 14:00	14.07.2012 21:00	Powerboard_C	2950	215350
PsuC_job_3	15.07.2012 13:30	18.07.2012 06:30	Powerboard_C	2750	200750
PsuC_job_4	25.07.2012 06:00	28.07.2012 23:00	Powerboard_C	5000	365000

*KKM – Komponenttien kokonaismäärä

Liite 3: Testiajo 2:n lopulliset aikataulut

Taulukko 29: Lopulliset aikataulut.

Linja 1

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
CompE_job_01	02.07.2012 14:00	02.07.2012 14:00	04.07.2012 07:44	06.07.2012 14:00
CompC_job_01	02.07.2012 15:00	04.07.2012 07:44	05.07.2012 19:32	07.07.2012 11:00
CompC_job_02	03.07.2012 18:10	05.07.2012 19:32	06.07.2012 22:41	07.07.2012 11:00
PsuB_job_1	05.07.2012 04:00	06.07.2012 22:41	07.07.2012 11:48	12.07.2012 12:00
PsuC_job_1	06.07.2012 04:00	07.07.2012 11:48	07.07.2012 16:13	11.07.2012 20:00
CompF_job_02	06.07.2012 17:50	07.07.2012 16:13	08.07.2012 01:19	13.07.2012 20:00
CompD_job_03	08.07.2012 02:20	08.07.2012 02:20	08.07.2012 13:47	09.07.2012 09:00
PsuB_job_2	09.07.2012 09:00	09.07.2012 09:00	10.07.2012 00:26	13.07.2012 15:00
PsuC_job_2	09.07.2012 14:00	10.07.2012 00:26	10.07.2012 11:55	14.07.2012 21:00
CompA_job_03	09.07.2012 13:00	10.07.2012 11:55	13.07.2012 01:43	15.07.2012 12:00
CompA_job_04	13.07.2012 06:00	13.07.2012 06:00	17.07.2012 10:22	20.07.2012 18:00
PsuC_job_3	15.07.2012 13:30	17.07.2012 10:22	17.07.2012 21:11	18.07.2012 06:30
ExpD_job_1	15.07.2012 10:40	17.07.2012 21:11	17.07.2012 23:43	18.07.2012 12:00
ExpF_job_1	15.07.2012 14:00	17.07.2012 23:43	18.07.2012 13:38	19.07.2012 11:00
CompC_job_07	19.07.2012 09:00	19.07.2012 09:00	21.07.2012 19:43	25.07.2012 14:00
CompH_job_06	21.07.2012 09:00	21.07.2012 19:43	25.07.2012 10:45	28.07.2012 10:00
CompD_job_08	25.07.2012 07:20	25.07.2012 10:45	27.07.2012 16:01	29.07.2012 07:00
CompG_job_06	25.07.2012 20:00	27.07.2012 16:01	28.07.2012 14:14	29.07.2012 12:00
ExpA_job_3	26.07.2012 10:00	28.07.2012 14:14	28.07.2012 23:01	30.07.2012 16:00
ExpB_job_3	26.07.2012 11:00	28.07.2012 23:01	29.07.2012 18:02	01.08.2012 21:00
PsuA_job_4	29.07.2012 11:00	29.07.2012 18:02	30.07.2012 23:34	31.07.2012 20:00
CompG_job_08	29.07.2012 09:30	30.07.2012 23:34	02.08.2012 09:58	02.08.2012 11:30
CompD_job_09	29.07.2012 22:00	02.08.2012 09:58	05.08.2012 15:47	05.08.2012 14:00

Linja 2

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
CompH_job_01	02.07.2012 06:20	02.07.2012 06:20	04.07.2012 12:58	08.07.2012 15:00
ExpE_job_3	04.07.2012 03:00	04.07.2012 12:58	04.07.2012 23:24	15.07.2012 12:00
CompB_job_03	05.07.2012 10:40	05.07.2012 10:40	07.07.2012 00:37	09.07.2012 20:00
CompH_job_02	07.07.2012 12:40	07.07.2012 12:40	10.07.2012 05:50	10.07.2012 12:00
CompB_job_05	09.07.2012 22:00	10.07.2012 05:50	11.07.2012 16:47	13.07.2012 12:00
CompC_job_04	09.07.2012 14:00	11.07.2012 16:47	12.07.2012 16:10	16.07.2012 08:00
CompB_job_06	12.07.2012 03:50	12.07.2012 16:10	13.07.2012 01:19	16.07.2012 15:00
PsuA_job_1	09.07.2012 09:00	13.07.2012 01:19	14.07.2012 21:29	16.07.2012 19:00
CompD_job_04	14.07.2012 08:00	14.07.2012 21:29	16.07.2012 11:23	19.07.2012 16:00
CompB_job_07	16.07.2012 09:30	16.07.2012 11:23	17.07.2012 19:13	19.07.2012 12:00
CompG_job_03	15.07.2012 18:50	17.07.2012 19:13	18.07.2012 16:56	19.07.2012 14:00
CompC_job_06	17.07.2012 09:00	18.07.2012 16:56	20.07.2012 05:48	20.07.2012 19:00
CompG_job_04	19.07.2012 11:40	20.07.2012 05:48	22.07.2012 10:23	23.07.2012 12:00
CompB_job_08	21.07.2012 02:00	22.07.2012 10:23	24.07.2012 13:04	24.07.2012 20:00
PsuA_job_3	22.07.2012 16:40	24.07.2012 13:04	24.07.2012 20:20	25.07.2012 16:00
CompH_job_05	17.07.2012 16:40	24.07.2012 20:20	26.07.2012 21:14	26.07.2012 22:00
PsuB_job_3	21.07.2012 13:30	26.07.2012 21:14	27.07.2012 09:00	27.07.2012 13:00
CompD_job_07	22.07.2012 16:00	27.07.2012 09:00	28.07.2012 20:18	27.07.2012 14:00
CompE_job_09	23.07.2012 13:00	28.07.2012 20:18	31.07.2012 22:45	28.07.2012 10:00
ExpA_job_2	18.07.2012 10:00	31.07.2012 22:45	01.08.2012 22:34	28.07.2012 14:00
ExpC_job_3	26.07.2012 15:30	01.08.2012 22:34	02.08.2012 08:25	01.08.2012 10:00
CompH_job_08	29.07.2012 15:00	02.08.2012 08:25	05.08.2012 04:42	02.08.2012 15:00

Linja 3

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
CompH_job_01.pt2	02.07.2012 06:20	02.07.2012 06:20	04.07.2012 10:27	08.07.2012 15:00
CompD_job_01	03.07.2012 00:40	04.07.2012 10:27	08.07.2012 04:26	09.07.2012 10:00
CompE_job_03	06.07.2012 17:40	08.07.2012 04:26	09.07.2012 14:35	09.07.2012 12:00
CompA_job_01	04.07.2012 02:50	09.07.2012 14:35	10.07.2012 10:15	11.07.2012 12:00
CompE_job_04	08.07.2012 16:00	10.07.2012 10:15	11.07.2012 16:02	11.07.2012 12:00
CompA_job_02	07.07.2012 17:50	11.07.2012 16:02	13.07.2012 01:12	12.07.2012 19:00
CompH_job_03	10.07.2012 14:30	13.07.2012 01:12	15.07.2012 05:43	15.07.2012 10:00
CompF_job_05	12.07.2012 18:00	15.07.2012 05:43	16.07.2012 05:10	17.07.2012 12:00
CompA_job_05	16.07.2012 08:00	16.07.2012 08:00	17.07.2012 08:51	20.07.2012 09:00
CompD_job_05	16.07.2012 12:00	17.07.2012 08:51	18.07.2012 19:21	20.07.2012 15:30
ExpF_job_2	18.07.2012 14:30	18.07.2012 19:21	19.07.2012 12:54	23.07.2012 18:00
CompE_job_08	19.07.2012 18:00	19.07.2012 18:00	21.07.2012 18:50	23.07.2012 20:00
CompD_job_06	20.07.2012 03:00	21.07.2012 18:50	24.07.2012 12:37	27.07.2012 10:00
CompB_job_09	24.07.2012 12:20	24.07.2012 12:37	26.07.2012 06:33	29.07.2012 18:00
CompB_job_10	25.07.2012 21:00	26.07.2012 06:33	27.07.2012 01:19	31.07.2012 10:00
CompG_job_07	26.07.2012 11:00	27.07.2012 01:19	28.07.2012 22:46	30.07.2012 14:00
CompE_job_10	26.07.2012 06:20	28.07.2012 22:46	01.08.2012 04:49	31.07.2012 19:30
CompF_job_09	29.07.2012 12:40	01.08.2012 04:49	01.08.2012 12:37	02.08.2012 16:00
CompC_job_10	29.07.2012 12:10	01.08.2012 12:37	03.08.2012 11:55	05.08.2012 17:30

Linja 4

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
CompB_job_02	03.07.2012 16:00	03.07.2012 16:00	05.07.2012 15:08	07.07.2012 18:00
CompF_job_01	03.07.2012 23:00	05.07.2012 15:08	06.07.2012 10:14	09.07.2012 20:00
CompE_job_02	05.07.2012 16:10	06.07.2012 10:14	10.07.2012 02:27	09.07.2012 11:00
CompF_job_03	10.07.2012 06:20	10.07.2012 06:20	13.07.2012 07:45	16.07.2012 12:00
ExpB_job_1	12.07.2012 01:20	13.07.2012 07:45	13.07.2012 14:11	16.07.2012 12:00
CompG_job_02	10.07.2012 09:40	13.07.2012 14:11	15.07.2012 13:53	17.07.2012 15:00
CompH_job_04	15.07.2012 16:20	15.07.2012 16:20	18.07.2012 13:37	22.07.2012 12:00
CompF_job_06	17.07.2012 09:00	18.07.2012 13:37	19.07.2012 00:29	19.07.2012 19:00
ExpE_job_2	18.07.2012 07:30	19.07.2012 00:29	19.07.2012 04:12	22.07.2012 12:00
CompH_job_05.pt2	17.07.2012 16:40	19.07.2012 04:12	22.07.2012 12:10	26.07.2012 22:00
ExpC_job_2	20.07.2012 04:00	22.07.2012 12:10	22.07.2012 23:59	27.07.2012 21:00
CompC_job_08	23.07.2012 10:00	23.07.2012 10:00	24.07.2012 23:31	28.07.2012 20:00
CompH_job_07	23.07.2012 12:00	24.07.2012 23:31	28.07.2012 11:17	28.07.2012 08:00
PsuC_job_4	25.07.2012 06:00	28.07.2012 11:17	29.07.2012 04:31	28.07.2012 23:00
CompC_job_09	25.07.2012 08:00	29.07.2012 04:31	31.07.2012 16:37	31.07.2012 10:00
CompD_job_09.pt2	29.07.2012 22:00	31.07.2012 16:37	05.08.2012 17:35	05.08.2012 14:00

Linja 5

Työ	Julkaisuaika	Aloitusaika	Valmistumisaika	Määräaika
CompB_job_01	02.07.2012 04:00	02.07.2012 04:00	02.07.2012 22:25	06.07.2012 15:00
CompG_job_01	04.07.2012 03:00	04.07.2012 03:00	08.07.2012 08:57	09.07.2012 10:00
CompD_job_02	06.07.2012 21:20	08.07.2012 08:57	09.07.2012 18:34	09.07.2012 20:00
CompB_job_04	08.07.2012 08:20	09.07.2012 18:34	11.07.2012 08:05	11.07.2012 10:00
ExpC_job_1	10.07.2012 22:30	11.07.2012 08:05	11.07.2012 12:34	13.07.2012 10:00
PsuA_job_2	09.07.2012 13:00	11.07.2012 12:34	12.07.2012 03:19	15.07.2012 12:00
CompF_job_04	10.07.2012 17:00	12.07.2012 03:19	13.07.2012 02:59	15.07.2012 12:00
CompE_job_05	11.07.2012 12:00	13.07.2012 02:59	13.07.2012 17:06	16.07.2012 10:00
CompC_job_03	06.07.2012 15:00	13.07.2012 17:06	14.07.2012 19:06	16.07.2012 11:00
ExpA_job_1	13.07.2012 22:30	14.07.2012 19:06	14.07.2012 23:22	16.07.2012 20:00
ExpE_job_1	14.07.2012 17:50	14.07.2012 23:22	15.07.2012 02:45	17.07.2012 10:00
CompC_job_05	13.07.2012 08:50	15.07.2012 02:45	16.07.2012 00:40	18.07.2012 15:00
CompE_job_06	14.07.2012 08:30	16.07.2012 00:40	17.07.2012 16:08	20.07.2012 23:00
CompE_job_07	16.07.2012 10:10	17.07.2012 16:08	21.07.2012 19:59	21.07.2012 20:00
ExpD_job_2	21.07.2012 15:00	21.07.2012 19:59	22.07.2012 00:11	24.07.2012 10:00
CompF_job_07	20.07.2012 08:00	22.07.2012 00:11	22.07.2012 05:26	24.07.2012 13:00
ExpB_job_2	20.07.2012 13:00	22.07.2012 05:26	22.07.2012 15:02	25.07.2012 10:00
CompA_job_06	20.07.2012 13:00	22.07.2012 15:02	23.07.2012 10:31	25.07.2012 19:30
CompG_job_05	22.07.2012 20:00	23.07.2012 10:31	23.07.2012 18:40	26.07.2012 09:00
CompH_job_07.pt2	23.07.2012 12:00	23.07.2012 18:40	27.07.2012 05:52	28.07.2012 08:00
CompA_job_07	23.07.2012 09:00	27.07.2012 05:52	28.07.2012 22:38	29.07.2012 14:00
CompF_job_08	23.07.2012 14:20	28.07.2012 22:38	29.07.2012 22:05	30.07.2012 13:00
CompA_job_08	27.07.2012 17:15	29.07.2012 22:05	01.08.2012 20:59	31.07.2012 21:00
PsuB_job_4	29.07.2012 17:15	01.08.2012 20:59	02.08.2012 17:40	05.08.2012 14:00