TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MILJA VEHMAANPERÄ

LINEAARISEN KOKONAISLUKUOPTIMOINNIN HYÖDYNTÄMINEN LUKUJÄRJESTYSTEN MUODOSTAMISESSA

Diplomityö

Tarkastajat:
Yliopistonlehtori Simo Ali-Löytty ja
Yliopistotutkija Timo Hämäläinen
Tarkastajat ja aihe hyväksytty
Luonnontieteiden tiedekuntaneuvoston
kokouksessa 13.01.2016

TIIVISTELMÄ

MILJA VEHMAANPERÄ: Lineaarisen kokonaislukuoptimoinnin hyödyntäminen lukujärjestysten muodostamisessa
Tampereen teknillinen yliopisto
Diplomityö, 52 sivua, 16 liitesivua
Kesäkuu 2016
Teknis-luonnontieteellinen koulutusohjelma
Pääaine: Matematiikka
Tarkastajat: Yliopistonlehtori Simo Ali-Löytty ja Yliopistotutkija Timo Hämäläinen
Avainsanat: koulujen lukujärjestysongelma, yliopistojen lukujärjestysongelma

Automatisoitu aikataulujen luominen on haastava tutkimusala, johon kuuluvat myös koulujen ja yliopistojen lukujärjestykset. Tässä työssä luodaan yleiskatsaus lukujärjestysongelmaan sekä muodostetaan ja ratkaistaan esimerkkiongelma kahdella eri kustannusfunktiolla.

Lukujärjestysten muodostamisessa on kyse oppimistapahtumien ja resurssien eli opettajien, oppilasryhmien sekä opetustilojen jakamisesta eri aikaväleille, eli tunneille, tiettyjä rajoitteita noudattaen. Automatisoidussa lukujärjestysten luomisessa voidaan käyttää hyödyksi esimerkiksi lineaarista binäärilukuoptimointia. Tähän optimointimalliin luodaan rajoitteet ja kustannusfunktio, jonka arvo pyritään optimoimaan rajoitteita noudattaen. Rajoitteet voivat olla joko kovia tai pehmeitä ja ne voivat vaihdella tapauksittain. Kovien rajoitteiden on toteuduttava, jotta lukujärjestys voidaan muodostaa, kun taas kustannusfunktioon sisältyvien pehmeiden rajoitteiden avulla määritetään muodostettavan lukujärjestyksen laatu.

Eri pehmeille rajoitteille voidaan antaa erisuuruiset painoarvot sen mukaan, kuinka tärkeinä niiden toteutumista pidetään verrattuna muihin pehmeisiin rajoitteisiin. Tämän työn esimerkkitapauksessa kustannusfunktion avulla minimoidaan hyppytuntien määrää ja opetustilojen käytöstä aiheutuvia kustannuksia. Mikäli oppimistapahtumia on melko vähän, kuten tässä työssä, nämä kaksi optimoitavaa tekijää eivät riitä muodostamaan mielekkäitä lukujärjestyksiä. Lukujärjestyksiin muodostuu yksittäisen oppimistapahtuman päiviä, mikä ei ole toivottavaa. Täten laadukkaiden lukujärjestysten luomiseksi tarvitsee kustannusfunktioon sisällyttää myös muita pehmeitä rajoitteita.

ABSTRACT

MILJA VEHMAANPERÄ: Using integer linear programming in timetabling
Tampere University of Technology
Master of Science Thesis, 52 pages, 16 Appendix pages
June 2016
Master's Degree Programme in Science and Engineering
Major: Mathematics
Examiners: University Lecturer Simo Ali-Löytty and University Researcher Timo Hämäläinen
Keywords: school timetabling problem, university timetabling problem

Automated scheduling, including school and university timetabling, is a demanding field of research. This thesis gives an overview of the timetabling problem. Additionally, an example is formulated and solved with two different cost functions.

Timetabling aims to distribute learning events and resources like teachers, students and classrooms to different time slots while satisfying a set of constraints. An automated timetabling problem can be formulated, for example, as a linear binary integer program. That optimizing model includes constraints and a cost function and the aim is to optimize the cost function subject to these constraints. The constraints can be either hard or soft and they can differ according to the requirements of the case in question. The hard constraints have to be satisfied or the timetable cannot be generated. As for the soft constraints that are included into the cost function, the quality of a feasible timetable is determined by the fulfilment of them.

It is possible to set unequal weights for different soft constraints, depending on how important the fulfilment of them is compared to other soft constraints. In this thesis the number of free periods and the running costs of classrooms are minimized with the help of a cost function. If there are only a few learning events like in this thesis, these two soft constraints are not enough to generate practical timetables. In the case of few soft constraints, days with single lessons would be formed, which is not desirable. Thus, in order to create high-quality timetables, more soft constraints need to be included into the cost function.

ALKUSANAT

Tämä diplomityö on kirjoitettu opintojen ohessa Tampereen teknillisessä yliopistossa pääasiassa kevätlukukauden 2016 aikana. Olen kiitollinen mahdollisuudesta toteuttaa työ itselleni mielenkiintoisesta aiheesta. Tämä mahdollisuus teki kirjoitusurakasta huomattavasti mielekkäämmän.

Haluan kiittää Simo Ali-Löyttyä ja Timo Hämäläistä työni ohjaamisesta, kärsivällisyydestä sekä neuvoista kirjoitusprosessin aikana. Kiitokset haluan esittää myös Iinalle englannin kielen oikoluvusta. Lisäksi haluan kiittää pesäpalloyhteisöä, jonka osana on ollut etuoikeus ja kunnia saada olla. Heidän kanssaan olen jakanut suurenmoisia hetkiä, ja heidän ansiostaan elämässäni on ollut muutakin kuin opiskelu.

Suurimmat kiitokset haluan esittää perheelleni sekä lähimmille ystävilleni myötäelämisestä ja kannustuksesta kaikkien näiden vuosien aikana. Saamani tuen lisäksi olen oppinut heiltä myös sen, mikä elämässä on tärkeää.

Tampereella 21.05.2016

Milja Vehmaanperä

SISÄLLYS

1. Johdanto 1

2. Lukujärjestysongelma 3

2.1 Yleinen malli 3

2.2 Koulujen lukujärjestysongelma 4

2.3 Yliopistojen lukujärjestysongelma 9

3. Optimointimalli 13

3.1 Kokonaislukuoptimointi 14

3.2 Lukujärjestysongelman matemaattinen malli 17

3.2.1 Lukujärjestysongelman rajoitteet 19

3.2.2 Hyppytuntien määritys 24

3.2.3 Kustannusfunktion luominen 27

4. Erottelu- ja luokitteluanalyysi 29

5. Ongelman ratkaiseminen 32

5.1 Branch and bound –menetelmä 33

5.2 Simplex-menetelmä 35

6. Esimerkkiongelma 39

6.1 Esimerkkiongelman rajoitteet 40

6.2 Kustannusfunktio 44

6.3 Tulokset 44

7. Yhteenveto 48

Lähteet 50

A. Esimerkkitapauksen MATLAB-koodi 53

B. Esimerkkitapausten tulokset 65

B.1 Hyppytuntien optimointi, esimerkkiratkaisu 1 65

B.2 Hyppytuntien optimointi, esimerkkiratkaisu 2 66
B.3 Hyppytuntien ja tilakustannusten optimointi kun $C_t = 1{,}2$ 67
B.4 Hyppytuntien ja tilakustannusten optimointi kun $C_t = 2$ 68

LYHENTEET JA MERKINNÄT

BIO	biotekniikan kurssi
DIF	differentiaaliyhtälöiden kurssi
FYKE	fysikaalisen kemian kurssi
FYS	fysiikan seminaari –kurssi
ITC	International Timetabling Competition, kansainvälinen aikataulujen luomiseen keskittyvä kilpailu
LAB	laboratoriotyöturvallisuuden kurssi
LAFY	laajan fysiikan kurssi
LATEX	LaTeX-kurssi
MATLAB	Matrix Laboratory –niminen numeerisen laskennan tietokoneohjelmisto
MISTA	Multidisciplinary International Scheduling Conference: Theory & Applications, kansainvälinen konferenssisarja
OHJ	ohjelmoinnin kurssi
PATAT	Practice and Theory of Automated Timetabling, kansainvälinen konferenssisarja
TETA	teollisuustalouden kurssi

$\mathbf{A}$	optimointiongelman rajoiteyhtälöiden ja -epäyhtälöiden kertoimista muodostettu matriisi
$\mathbf{a}$	kertoimet erottelufunktiossa
$\hat{\boldsymbol{a}}_s$	matriisin $\mathbf{N}$ sarake s
$\hat{\boldsymbol{a}}_s^*$	matriisin $\mathbf{B}$ käänteismatriisin ja sarakkeen $\hat{\boldsymbol{a}}_s$ tulo
a	rajoitearvo optimointiongelmassa
a_j	lukujärjestyksen sisältämä tunti
$\mathbf{B}$	matriisin $\mathbf{A}$ ei-singulaarinen osa, lineaarisen optimointiongelman kanta
$\mathbf{B}^{-1}$	matriisin $\mathbf{B}$ käänteismatriisi
$\mathbf{b}$	lineaarisen kokonaislukuongelman rajoitteiden perusteella muodostuva vektori
C_h	hyppytunnin aiheuttama kustannus
C_t	opetustilan käytöstä aiheutuva kustannus
$\mathbf{c}$	lineaarisen kokonaislukuongelman kustannusfunktion kerroinvektori

$\mathbf{c_B}$	matriisia $\mathbf{B}$ vastaavat alkiot vektorissa $\mathbf{c}$
$\mathbf{c_N}$	matriisia $\mathbf{N}$ vastaavat alkiot vektorissa $\mathbf{c}$
$\hat{\mathbf{c}}_\mathbf{N}$	redusoidut kustannukset
c_{jk}	binäärimuuttuja, kertoo onko ryhmä k vapaa tunnilla j
$f(\mathbf{x}, \mathbf{y})$	minimoitava kustannusfunktio
$f(\mathbf{x}, \mathbf{y})^*$	optimointiongelman optimitulos
$g(\mathbf{x}, \mathbf{y})$	epäyhtälörajoitefunktio
H1-H5	harjoitustapahtumat
$h(\mathbf{x}, \mathbf{y})$	yhtälörajoitefunktio
$\mathbf{I}$	identiteettimatriisi
i	oppimistapahtumaa kuvaava indeksi
(i_1, i_2)	tapahtumapari, josta tapahtuma i_1 halutaan järjestää ennen tapahtumaa i_2
j	lukujärjestyksen tuntia kuvaava indeksi
j_i	tunti, jolla oppimistapahtuma i järjestetään
K	luokitteluanalyysin luokkien lukumäärä tai optimointiongelman käypä alue
k	oppilasryhmää kuvaava indeksi
κ	jokin opiskelijaryhmä joukossa R_i
L1, L2	luentokerrat
L	Lagrangen funktio
l_j	suurin mahdollinen luentojen lukumäärä tunnilla j
λ	konveksisuuteen liittyvä kerroin
λ_L	Lagrangen kerroin
lkm_i	oppimistapahtumaan i osallistuvien ryhmien lukumäärä
M_{nk}	binäärimuuttuja, kertoo onko havainto n luokassa k
m	matriisin $\mathbf{A}$ lineaarisesti riippumattomien pystyrivien lukumäärä
$\hat{\boldsymbol{\mu}}$	kaikkien havaintojen keskiarvo
$\hat{\boldsymbol{\mu}}_k$	luokan k keskiarvo
$\mathbf{N}$	matriisin $\mathbf{A}$ singulaarinen osa
N	havaintojen $\mathbf{x}_n$ lukumäärä erotteluanalyysissä
N_k	havaintojen lukumäärä luokassa k
n_i	oppimistapahtumien lukumäärä
n_{ik}	niiden oppimistapahtumien lukumäärä, joihin ryhmä k osallistuu
n_j	tuntien lukumäärä viikossa
n_k	oppilasryhmien lukumäärä
$\mathbf{O}$	optimointitehtävä

Ω	käypä joukko
p_{ijk}	kertoo, onko tapahtuma i ryhmälle k ennaltamäärätty tunnille j
R	oppilasryhmien joukko
R_i	ryhmät, jotka osallistuvat oppimistapahtumaan i
$\mathbb{R}$	reaalilukujen joukko
$\mathbb{R}^n$	n-ulotteinen reaaliavaruus
r_k	oppilasryhmä
S_q	kurssiryhmä, jonka kurssien luennot tulee järjestää eri tunneilla
s, r	matriisien $\mathbf{N}$ ja $\mathbf{B}$ indeksit
Σ_b	erotteluanalyysin luokkien välistä vaihtelua kuvaava kovarianssimatriisi
Σ_{tot}	kokonaiskovarianssimatriisi
Σ_w	erotteluanalyysin luokkien sisäistä vaihtelua kuvaava kovarianssimatriisi
T	oppimistapahtumien joukko
T_γ	joukko, joka sisältää oppimistapahtumaparit, joiden tapahtumat halutaan laittaa järjestykseen
T_k	oppimistapahtumat, joihin ryhmä k osallistuu
$\mathbf{c}^T$	vektorin $\mathbf{c}$ transpoosi
t_i	oppimistapahtuma
$u \in \mathbb{P}_v$	$u \in \{1, 2, \ldots, v\}$
u_k, v_k, y_k	Simplex-algoritmin lisämuuttujat
$\mathbf{X}$	sisältää havainnot $\mathbf{x}_n$
$\mathbf{x}, \mathbf{y}$	vektoreita
$\mathbf{x}$	lineaarisen optimointiongelman kantaratkaisu
$\mathbf{x}^*$	optimointiongelman optimiratkaisu
$\mathbf{x_B}$	matriisia $\mathbf{B}$ vastaavat alkiot, eli kantamuuttujat, kantaratkaisuvektorissa $\mathbf{x}$
$\mathbf{x_N}$	matriisia $\mathbf{N}$ vastaavat alkiot kantaratkaisuvektorissa $\mathbf{x}$
$\mathbf{x}_n$	havainto
$\mathbf{x}_n^{(k)}$	havainto n luokassa k
$\mathbf{x}_S$	Simplex-algoritmin pelivaramuuttujat
x, y	valintamuuttujia
$\mathbf{x}_{ijk}$	muuttujasta x_{ijk} luotu vektori
x_{ijk}	binäärimuuttuja, kertoo onko ryhmällä k oppimistapahtuma i tunnilla j
$\mathbf{Y}$	erottelufunktio, muuttujien lineaarikombinaatio

$\mathbf{y}_{jk}$	muuttujasta y_{jk} luotu vektori
y_{jk}	binäärimuuttuja, joka kertoo onko ryhmällä k hyppytunti tunnilla j
$\mathbb{Z}^n$	kokonaislukujen n-ulotteinen avaruus
$\mathbb{Z}^n_+$	positiivisten kokonaislukujen n-ulotteinen avaruus
$\mathbf{z}_{jk}$	muuttujasta z_{jk} luotu vektori
z_{jk}	binäärimuuttuja, joka kertoo onko ryhmällä k tuplahyppytunti tunneilla j ja $j+1$

1. JOHDANTO

Erilaisia aikatauluja käytetään koulujen ja yliopistojen lisäksi esimerkiksi liikenteessä, työvuorolistoissa, urheilutapahtumissa sekä eri urheilulajien sarjaohjelmissa. Tehokkaiden aikataulujen luominen on tärkeää, mutta haastavaa. Lisääntymässä on nimenomaan tietokoneavusteinen, automatisoitu aikataulujen luominen. Tämä tuo kuitenkin mukanaan sekä kvalitatiivisia että kvantitatiivisia haasteita, jotka täytyy ratkaista sekä teknisistä että käytännöllisistä näkökulmista, jotta aikataulujen automatisoitu muodostaminen tulee mahdolliseksi [21].

Kansainväliset konferenssisarjat PATAT (Practice and Theory of Automated Timetabling) [21] sekä MISTA (Multidisciplinary International Scheduling Conference: Theory & Applications) [15] kokoavat molemmat joka toinen vuosi yhteen tutkijoita ja asiantuntijoita jakamaan ideoitaan aikataulujen luomisen kehittämiseksi. Tavoitteena on kehittää tehokkaita ja käytännöllisiä ratkaisuja, jotka ovat kaikille osapuolille mahdollisimman mielekkäitä. Vaikka aihetta on tutkittu pitkään ja konferensseja pidetty jo kahden vuosikymmenen ajan, aihe on edelleen haastava. Tutkijat tekevät jatkuvasti työtä kehittääkseen tätä systeemiä. [21] Konferenssien lisäksi aikatauluongelmien ratkaisujen kehittämiseksi järjestetään kansainvälisiä kilpailuja, ITC (International Timetabling Competition), joiden kautta kaikilla kunnianhimoisilla aikataulujen luomisesta kiinnostuneilla on mahdollisuus päästä osaltaan edistämään tutkimus- ja kehitystyötä.

Koulujen ja yliopistojen lukujärjestykset voidaan jakaa tavallisiin kurssilukujärjestyksiin sekä tenttilukujärjestyksiin, ja nämä edelleen oppilaiden ja opettajien lukujärjestyksiin sekä opetustilakohtaisiin lukujärjestyksiin. [27] Tässä työssä rajoitutaan käsittelemään oppilaille tehtäviä koulujen ja yliopistojen kurssilukujärjestyksiä. Lukujärjestysten laatiminen voidaan periaatteessa hoitaa käsin, mutta kurssi- ja oppilasmäärän lisääntyessä tehtävä muuttuu yhä vaikeammaksi ja lopulta mahdottomaksi. Tämän vuoksi on kehitetty ohjelmia, jotka tekevät lukujärjestykset automaattisesti. Markkinoilla on useita erilaisia ohjelmia, joissa on käytetty eri al-

goritmeja ongelman ratkaisemiseen. Lukujärjestystä muodostettaessa tapauksesta luodaan matemaattinen malli, joka sisältää optimoitavan kustannusfunktion sekä rajoitteet. Tämä muodostettu optimointiongelma voidaan ratkaista erilaisten algoritmien avulla.

Luvussa 2 esitellään kirjallisuudessa esiintyvä yleinen lukujärjestysongelma sekä koulujen ja yliopistojen lukujärjestyksille tyypillisiä seikkoja. Luvussa 2 esitellään myös yleisimpiä käytettäviä rajoitteita sekä koulujen että yliopistojen malleissa. Tämän jälkeen luvussa 3 käsitellään lukujärjestysongelman matemaattista puolta. Ensin käydään läpi kokonaislukuoptimointimallin perusteita, minkä jälkeen muodostetaan lukujärjestysongelmalle rajoitteet matemaattisesti. Optimointiongelmassa rajoitteiden lisäksi tarvitaan myös kustannusfunktio, jonka muodostamista esitellään myös luvussa 3. Tässä työssä muodostetaan esimerkkiongelma, jossa tavoitteena on minimoida muun muassa hyppytuntien määrää. Jotta tästä voidaan luoda kustannusfunktio, tarvitaan avuksi erottelu- ja luokitteluanalyysiä, jota käsitellään luvussa 4.

Kun lukujärjestysongelmasta on muodostettu malli, ongelman ratkaisemiseksi tarvitaan työkaluja. Kaksi esimerkkiä, branch and bound –menetelmä sekä simplex-algoritmi, esitellään luvussa 5. Tämän jälkeen luvussa 6 esitellään tässä työssä muodostettu esimerkkitapaus, jolle myös luodaan lukujärjestykset MATLAB-ohjelmistoa hyväksi käyttäen. Lopuksi luvussa 7 kootaan yhteen työssä esitetyt havainnot ja saadut tulokset.

2. LUKUJÄRJESTYSONGELMA

Erilaisia lukujärjestysongelman yksityiskohtaisia ratkaisumalleja on esitelty kirjallisuudessa useita, mutta pohjimmaisin ajatus on niissä kaikissa kuitenkin sama. Lukujärjestysten luomisessa on yleisesti ottaen kyse siitä, että tietyt käytössä olevat resurssit sijoitetaan aikataulun sisältämiin aikaväleihin tiettyjä rajoitteita noudattaen.

Lähteestä riippuen lukujärjestysongelmassa käsitellään joko yksittäisiä resursseja tai resursseista muodostettavia resurssiryhmiä. Resurssien jakamista aikaväleihin säätelevät rajoitteet, jotka ovat joko kovia tai pehmeitä sen mukaan, kuinka tärkeinä ja välttämättöminä niiden toteutumista pidetään lukujärjestyksen muodostamisen kannalta. Lisäksi kovat ja pehmeät rajoitteet vaihtelevat riippuen siitä, mihin ja kenelle lukujärjestystä luodaan. [24]

2.1 Yleinen malli

Lukujärjestystä muodostettaessa oppimistapahtumat tulee tiettyjä rajoitteita noudattaen jakaa opettajille ja oppilaille sekä järjestettäviksi tiettyinä aikaväleinä [24], joita kutsutaan tunneiksi. Tunti ei kuitenkaan tarkoita, että aikaväli olisi pituudeltaan 60 minuuttia, vaan yleisimmin kouluissa ja yliopistoissa yksi tunti kestää 45 minuuttia. Joissain kouluissa ja ikäluokissa tunti voi kestää myös 75 minuuttia, mutta se on huomattavasti harvinaisempaa kuin 45 minuutin tunnit.

Kaikkia niitä tekijöitä, joita tarvitaan oppimistapahtuman, eli oppitunnin, luennon tai harjoitustapahtuman järjestämiseen, kutsutaan resursseiksi. Lukujärjestysongelmassa resursseina voidaan pitää opettajia, oppilasryhmiä sekä luokkatiloja. [24] Kunkin tyyppisiä resursseja voidaan pitää omana resurssiryhmänään, eli esimerkiksi kaikki opettajat muodostavat yhden resurssiryhmän. Kukin tapahtuma vaatii onnistuakseen resurssin jokaisesta resurssiryhmästä. [27] Joissain tilanteissa voidaan myös käyttää yhteen tapahtumaan useaa resurssia samasta resurssiryhmästä, mutta

näitä muutamaa poikkeustapausta käsitellään tarkemmin koulujen ja yliopistojen lukujärjestysongelmien yhteydessä luvuissa 2.2 ja 2.3.

Yleensä lukujärjestykset laaditaan viikon ajalle ja samaa lukujärjestystä toistetaan viikoittain tietyn aikaperiodin ajan. Esimerkiksi lukiossa lukuvuoden aikana on lukiosta riippuen joko 5 tai 6 jaksoa, kun taas yliopistoissa periodeja on 4. Viikon opetuspäivät jaetaan ajallisesti yhtä pitkiin tunteihin, joille oppimistapahtumat ja käytettävissä olevat resurssit tulee jakaa. [27]

Kovat ja pehmeät rajoitteet rajoittavat sitä, miten tuntien jako on sallittua tehdä. Kovat rajoitteet ovat rajoitteita, joiden on ehdotonta toteutua, jotta lukujärjestys voidaan muodostaa ja se on käyttökelpoinen. [28] Vaikka rajoitteet vaihtelevat tapauksittain, yleensä kovat rajoitteet estävät sen, että mikään yksittäinen resurssi ei voi olla samaan aikaan kahdessa eri paikassa [20]. Toisaalta myös saman resurssin edustajat eivät voi olla samaan aikaan samassa paikassa, eli tietyllä tunnilla saman resurssiryhmän edustajia voi osallistua tiettyyn tapahtumaan vain yksi [24]. Myös tähän on poikkeustapauksia, joita käsitellään myöhemmin koulujen ja yliopistojen lukujärjestysongelmien yhteydessä luvuissa 2.2 ja 2.3.

Pehmeät rajoitteet ovat puolestaan rajoitteita, joiden ei ole pakko toteutua, jotta lukujärjestyksestä saadaan käypä. Kuitenkin mitä useampaa pehmeää rajoitetta voidaan noudattaa, sitä parempilaatuinen lukujärjestys saadaan luotua [28]. Joskus osa pehmeistä rajoitteista voi olla ristiriidassa keskenään, jolloin kaikki niistä eivät voi toteutua samaan aikaan. Yleensä pehmeät rajoitteet on kuitenkin määritelty niin, että ne voivat kaikki toteutua. [24]

Koulujen lukujärjestysongelmassa pyritään siihen, että kukaan opettaja ei opeta kahta luokkaa samaan aikaan, ja päinvastoin, ettei mikään luokka osallistu kahden opettajan pitämälle oppitunnille samaan aikaan. Yliopistojen kurssilukujärjestysongelmassa sen sijaan tavoitteena on pyrkiä minimoimaan päällekkäisyys sellaisilta kursseilta, joilla on yhteisiä opiskelijoita, ja lisäksi ettei kukaan opettaja opeta samaan aikaan kahdessa eri paikassa. [16]

2.2 Koulujen lukujärjestysongelma

Koulujen lukujärjestysongelmalle tyypillistä on se, että luokkatilojen koot on mitoitettu siten, että kaikki oppilasryhmät mahtuvat kaikkiin opetustiloihin [28]. Ala-

asteella, kun opetusta antavat pääasiassa luokanopettajat, on tyypillistä, että jokaisella luokalla on oma luokkatilansa, eli niin kutsuttu kotiluokka. Lisäksi joissakin oppiaineissa opetus tapahtuu niille tarkoitetuissa erikoisluokissa. [24] Esimerkiksi musiikin ja liikunnan oppitunnit pidetään yleensä juuri niille tarkoitetuissa tiloissa. Myös muun muassa biologian, käsitöiden ja kotitalouden oppitunneille on usein varattu erikoisluokat.

Yläasteelle ja lukioon siirryttyä opetusta antavat aineenopettajat, joilla kullakin on pitämiään oppitunteja varten tietty luokkatila. Eli sekä ala-asteella että yläasteella ja lukiossa tieto oppituntiin käytettävästä luokkatilasta sisältyy joko oppilasryhmän tai opettajan määritykseen. Näistä seikoista johtuen luokkatilojen valintaan ei tarvitse koulujen lukujärjestysongelmassa kiinnittää erikseen huomiota.

Peruskoulun alemmilla luokilla kukin oppilas voi kuulua vain yhteen luokkaan, jolloin oppilailla ei voi olla oppituntien päällekkäisyyksiä. Tällöin lukujärjestyksiä muodostettaessa käsitellään luokka-opettaja –pareja [5, 24], eli opettajista ja luokista muodostetaan pareja, jotka jaetaan eri tunneille.

Ylemmillä luokilla valinnaiset aineet lisääntyvät, jolloin luokkajaot eivät ole enää yksikäsitteiset vaan riippuvat oppiaineesta. [5] Tällöin oppilasryhmät sisältävät eri oppilaat kullakin kurssilla, mutta tästä ei juurikaan aiheudu haittaa, sillä valinnaiset kurssit järjestetään siten, ettei päällekkäisyydet ole mahdollisia. Valinnaisaineiden kurssit ryhmitellään ja kunkin kurssiryhmän kaikki kurssit järjestetään samaan aikaan keskenään, mutta eri aikaan eri kurssiryhmiin kuuluvien kurssien kanssa. Kun oppilas saa valita jokaisesta ryhmästä ainoastaan yhden kurssin, päällekkäisyydet ovat mahdottomia.

Kuten yleisen mallin tapauksessa luvussa 2.1 mainittiin, on mahdollista käyttää yhdelle tunnille useampaa saman resurssiryhmän resurssia. Yleensä tämä tarkoittaa peruskoulujen ja lukioiden tapauksissa sitä, että jotkin oppiaineet pidetään useammalle luokalle samaan aikaa. Useissa kouluissa yhdistetään useampi luokka esimerkiksi liikuntatuntien ajaksi. Lisäksi luokan jakamista ryhmiin tapahtuu ainakin liikunnan tunneille, joilla tytöt ja pojat ovat useimmiten omina ryhminään.

Toisin kuin useiden maiden ja korkeakoulujen tapauksissa, Suomessa peruskouluissa ja lukioissa joka päivä on varattu aika ruokatunnille. Ruokatunti ei kuitenkaan yleensä kestä yhtä kauaa kuin muut tunnit, vaan on yleensä pituudeltaan korkeintaan puoli tuntia. Tämä saa aikaan sen, että tunneilla ennen ja jälkeen ruokatunnin

on eri alkamisajat. Toisin sanoen, jos ennen ruokatuntia tunnit alkavat tasatunnein, ruokailun jälkeen tuntien alkamisajat ovat joitakin muita kuin tasatunteja.

Yleisesti ottaen lukujärjestyksiä muodostettaessa tarkoituksena on siis jakaa opettajasta, oppimistapahtumasta ja oppilasryhmästä koostuvat kolmikot eri ajankohtina järjestettäville tunneille. [28] Tilanteesta riippuen joko kyseinen opettaja tai oppilasryhmä määrittää luokkatilan, jossa oppitunti pidetään. Resurssien ja oppimistapahtumien sijoittelussa huomioon otettavista kovista rajoitteista yleisimpiä ovat seuraavat [20]

- Kukin oppilasryhmä voi osallistua vain yhteen oppimistapahtumaan samaan aikaan.
- Kukin opettaja voi opettaa vain yhdessä oppimistapahtumassa samaan aikaan.
- Kullekin oppilasryhmälle tulee järjestää vaadittu määrä oppimistapahtumia kustakin oppiaineesta.

Lisäksi lähes kaikissa ulkomaalaisten koulujen tapauksissa vaaditaan kovana rajoitteena, että kunkin opettajan tulee opettaa vaadittu lukumäärä oppitunteja kullekin luokalle [20]. Suomessa tämä ehto on kuitenkin kyseenalainen, sillä riittää, että kukin opettaja opettaa tietyn määrän tunteja, ja kullekin oppilasryhmälle tulee opetettua kutakin oppiainetta oikea määrä. Kovaksi rajoitteeksi ei ole tarpeen asettaa sitä, kuka tunnit luokille pitää tai kenelle opettaja tuntinsa pitää, kunhan vaaditut tuntimäärät täyttyvät.

Yleensä [20] lukujärjestyksen muodostamisen taustalla ovat edellä luetellut kovat rajoitteet. Näiden lisäksi tapauksesta riippuen muita yleisimpiä kovia rajoitteita ovat seuraavat [20]

- Kutakin luokkatilaa voidaan käyttää vain yhteen oppimistapahtumaan samaan aikaan. Tämä on oleellisinta erikoisluokkien tapauksissa, sillä useat oppilasryhmät käyttävät samaa tilaa. Muissa tapauksissa tätä ei Suomessa tarvitse ottaa huomioon, sillä luokkatilan valinta sisältyy oppilasryhmien tai opettajien valintaan.
- Kukin opettaja voi pitää oppitunteja ainoastaan niinä ajankohtina, kun hän on käytettävissä koululla. Tämä on asianmukainen rajoite erityisesti sellaisissa tapauksissa, joissa jotkin opettajat opettavat useammassa koulussa.

- Oppilasryhmien jakaminen ja yhdistäminen on mahdollista tietyissä oppiaineissa kuten liikunnassa.
- Jos koulupäivän tulee päättyä tiettyyn aikaan, sen jälkeen ei voi enää olla oppimistapahtumia.
- Jotkin oppimistapahtumat pitää tai päin vastoin ei pidä järjestää tietyillä tunneilla. Esimerkiksi liikuntatunteja ei välttämättä haluta pitää heti ruokailun jälkeen. Tämä olisi ainakin Suomessa todennäköisesti korkeintaan pehmeä rajoite.

Kovien rajoitteiden lisäksi lukujärjestysongelmissa käytetään pehmeitä rajoitteita, joiden toteutuminen puolestaan määrittää lukujärjestyksen laadun. Myös pehmeät rajoitteet saattavat vaihdella suurestikin eri tapausten välillä. Lisäksi on mahdollista, että samat rajoitteet ovat joissain tapauksissa kovia ja joissain pehmeitä. Esimerkkejä tällaisista rajoitteista ovat seuraavat [14, 20, 24]

- Oppilailla ei saa olla keskellä päivää hyppytunteja. Erityisesti ala-asteella tämä on ehdoton rajoite, mutta yläasteella ja lukiossa hyppytunnit tulevat mahdollisiksi aineiden valinnaisuuden vuoksi. Jos hyppytunteja kuitenkin tulee, ne pyritään sijoittamaan päivän alkuun tai loppuun.
- Opettajilla on mahdollisuus hyppytunteihin keskellä päivää, mutta niiden määrä pyritään kuitenkin minimoimaan, jolloin saadaan aikaan mahdollisimman kompakti lukujärjestys.
- Opettajilla on tietty määrä opetettavia tunteja viikossa tai päivässä. Suomessa tuntiopettajilla vaaditut tunnit koskevat koko lukuvuotta. Tällöin on mahdollista, että esimerkiksi jossain jaksossa on enemmän opetusta kuin jossain toisessa. Vaikka opetustuntien lukumäärä viikossa ei välttämättä olekaan vakio koko vuoden ajan, kuitenkin jo lukujärjestystä muodostettaessa tiedetään, kuinka monta tuntia kukin opettaja viikossa opettaa.
- Oppilaiden saman aineen oppimistapahtumat jakautuvat opetuspäiville mahdollisimman tasaisesti.
- Oppilailla ei ole saman aineen oppimistapahtumia useampaa kertaa saman päivän aikana, eikä välttämättä edes peräkkäisinä päivinä. Tästä ovat poikkeuksena kaksois- tai kolmoistunnit, joita halutaan pitää joistakin oppiaineista.

- Tietylle oppilasryhmälle ei järjestetä saman aineen oppimistapahtumia päivän viimeisellä ja seuraavan päivän ensimmäisellä tunnilla.

Lisäksi esimerkkejä muista mahdollisista pehmeistä rajoitteista koulujen lukujärjestysongelmissa ovat seuraavat [14, 19, 20, 28]

- Kunkin opettajan opetustuntien kokonaismäärä jakautuu mahdollisimman tasaisesti niiden päivien välillä kun he ovat käytettävissä koululla.
- Osa opettajista opettaa mieluiten joinain tiettyinä ajankohtina.
- Opettajien mahdolliset hyppytunnit jakautuvat mahdollisimman tasaisesti kaikille opettajille. Tässä täytyy kuitenkin huomioida se, että opettajilla saattaa olla opetusta eri tuntimääriä viikossa, jolloin hyppytuntien määrääkään ei voi vain jakaa tasan kaikille.
- Koulupäivä alkaa niin myöhään kuin mahdollista.
- Koulupäivä päättyy niin aikaisin kuin mahdollista.

Koulupäivien alkamiseen ja loppumiseen liittyvät rajoitteet ovat suoraan yhteydessä hyppytuntien lukumäärään. Jos lukujärjestyksestä saadaan luotua kompakti kokonaisuus, ja viikon aikana olevat tunnit on jaettu koko viikolle tasaisesti, kunakin päivänä oppilaat osallistuvat tietylle määrälle tunteja. Tämä tuntien lukumäärä päivässä ratkaisee käytännössä sen, moneltako päivän on mahdollista päättyä. Hyppytunnit sen sijaan lykkäävät koulupäivän päättymisaikaa myöhemmäksi, jolloin rajoite päivän päättymisestä mahdollisimman aikaisin ei toteudu.

Suomessa alemmilla luokilla lukujärjestysongelma vastaa melko lailla yleistä koulujen lukujärjestysongelmaa. Sen sijaan ylemmillä luokilla ja erityisesti lukiossa malli on koulujen ja yliopistojen mallin kombinaatio johtuen suuresta määrästä valinnaisia oppiaineita ja kursseja. [19] Osa kursseista on pakollisia kaikille, jolloin tilanteen mallintamisessa voidaan käyttää koulujen lukujärjestysongelmaa. Kun taas käsitellään valinnaisia kursseja, tilanteen mallintamiseen käytetään yliopistojen lukujärjestysongelmaa.

2.3 Yliopistojen lukujärjestysongelma

Yliopistojen lukujärjestysongelmassa tarkoituksena on sijoittaa yleensä viikon ajalle jokaisen kurssin luennot tiettyyn määrään opetustiloja ja tunteja. [26] Periodi on yliopistoissa se ajanjakso, jonka ajan saatua lukujärjestystä tyypillisesti toistetaan. Lukujärjestyksen luonnissa olennaista on se, että kursseilla, joilla on samoja opiskelijoita, olisi mahdollisimman vähän päällekkäisyyksiä. [24] Kuka opettaja luennoi mitäkin kurssia, on päätetty jo etukäteen, jolloin sitä ei tarvitse ottaa enää huomioon lukujärjestystä muodostettaessa. [11]

Kouluissa on tyypillistä, että sama opettaja opettaa useaa oppiainetta, mutta joissakin yliopistoissa, esimerkiksi Rooman yliopistossa sama luennoitsija luennoi yleensä vain yhtä kurssia. [24] Sellaisissa tapauksissa, joissa tietty luennoitsija luennoi vain yhtä kurssia, ei tarvitse ottaa huomioon sitä, että luennoitsija voi luennoida vain yhdessä paikassa kerrallaan. Suomessa tämä sen sijaan täytyy ottaa huomioon, sillä yksi luennoitsija voi luennoida useampaakin kurssia. Yleensä kuitenkin kukin luennoitsija luennoi samat kurssit vuosittain, ellei muutoksiin ole erityisesti tarvetta.

Yleisen mallin ja koulujen lukujärjestysongelman yhteydessä on mainittu useamman saman resurssiryhmän resurssin käytöstä yhdessä oppimistapahtumassa. Yleensä jokaisesta resurssiryhmästä käytetään kullekin oppimistapahtumalle vain yhtä resurssia, mutta kouluissa poikkeustapauksena tästä oli oppilasryhmien yhdistely joillekin oppitunneille. Yliopistoissa poikkeuksen aiheuttaa sen sijaan useamman opettajan osallistuminen samaan oppimistapahtumaan. Esimerkiksi laskuharjoitustapahtumissa voi olla useampikin opettaja kerrallaan, jos kurssin opiskelijamäärä niin vaatii.

Toisin kuin koulujen lukujärjestysongelmassa, jossa käytettävät luokkatilat määräytyvät opettajien tai oppilasryhmien mukaan, yliopistoissa opetustilojen kokoon ja vapaanaoloon on kiinnitettävä enemmän huomiota [16]. Eri kursseille osallistuvat ryhmät ovat eri kokoisia, ja jokaiselle samaan aikaan järjestettävälle oppimistapahtumalle tulee olla oikean kokoinen tila vapaana. Opetustilan valinnassa olennaista ei ole niinkään se, kuinka monta opiskelijaa kurssille on ilmoittautunut, vaan se, kuinka moni opiskelija osallistuu oppimistapahtumiin. Tarkkaa lukumäärää ei toki voida tietää etukäteen, mutta hyvin suuntaa antavaa tietoa saadaan aikaisempien vuosien kokemuksista.

Koulujen lukujärjestysongelmassa yhtenä perusajatuksena on se, että oppilas voi kuulua vain yhteen oppilasryhmään. Yliopistojen lukujärjestysongelmassa eroavai-

suutena on se, että yliopistoissa ei ole vakituisia oppilasryhmiä, vaan opiskelijat osallistuvat kursseille, joita on käynnissä useita periodin aikana. [16] Tällöin joudutaan huomioimaan se, ettei tiettyjen kurssien oppimistapahtumat ole päällekkäin.

Yliopistoissa yleensä kukin laitos laatii omat lukujärjestyksensä tarjoamistaan kursseista. Tällöin kurssien päällekkäisyyksiä koskevat rajoitteet pystytään huomioimaan vain saman laitoksen järjestämien kurssien osalta. Kun jokin laitos haluaa minimoida kurssiensa oppimistapahtumien päällekkäisyyden myös muiden laitosten oppimistapahtumien kanssa, laitoksen on vain pyrittävä huomioimaan mahdollisimman hyvin muiden laitosten lukujärjestykset. [16]

Samoin kuin koulujen lukujärjestysongelmassa, myös yliopistojen lukujärjestysongelmaa ratkaistaessa otetaan huomioon sekä kovia että pehmeitä rajoitteita. Kovien rajoitteiden on toteuduttava, jotta lukujärjestys on käyttökelpoinen. [1] Yliopistoissa lukujärjestyksiä luotaessa yleisimpiä kovia rajoitteita ovat seuraavat [1, 2, 9, 11, 13, 16, 26]

- Kukaan opiskelija ei voi osallistua kahteen eri oppimistapahtumaan samaan aikaan.
- Luennoitsija voi luennoida vain yhdessä oppimistapahtumassa kerrallaan. Kuten aiemmin jo mainittiin, tämä on tärkeä rajoite erityisesti sellaisissa yliopistoissa, joissa sama luennoitsija luennoi useita eri kursseja saman periodin aikana.
- Luento voidaan järjestää vain vapaana olevassa opetustilassa. Kahta eri oppimistapahtumaa ei siis voida järjestää samanaikaisesti samassa tilassa, vaan resurssien tulee riittää kaikille samanaikaisille oppimistapahtumille.
- Opetustilassa on oltava tarpeeksi istumapaikkoja kyseisen oppimistapahtuman osallistujamäärälle.
- Opetustilan varustelun tulee vastata kurssin vaatimuksia [1, 23]. Tämä koskee Suomessa oikeastaan vain luento- ja harjoitustapahtumien jakoa niille tarkoitettuihin tiloihin sekä kursseja, jotka sisältävät työskentelyä laboratoriossa.
- Jokainen oppimistapahtuma kestää yhden tunnin eli yliopistoissa tyypillisesti 45 minuuttia [13]. Luentoja pidetään kuitenkin yleensä aina kaksi peräkkäin, jolloin luentoja lasketaan pidetyksi kaksi. Joskus luentoja sisältyy viikkoon

kolme, jolloin kaksi luentoa voidaan pitää peräkkäin, mutta yksi joudutaan luennoimaan yksittäisenä luentona.

- Saman kurssin luentoja saa olla korkeintaan kaksi päivässä [13].

Yliopistojen lukujärjestysongelmissa käytettäviä yleisimpiä pehmeitä rajoitteita ovat puolestaan seuraavat

- Opiskelijoille pyritään luomaan mahdollisimman kompaktit lukujärjestykset eli mahdollisimman vähän hyppytunteja keskelle päivää [13].
- Saman kurssin luennot järjestetään aina samassa luentosalissa ja samaan kellonaikaan [13, 16]. Kaksi periodia kestävien kurssien kohdalla on tässä poikkeuksia, sillä ei voida taata, että kellonaika ja luentosali pysyvät samana myös uuden periodin lukujärjestyksessä.
- Opetustilakohtaiset lukujärjestykset ovat mahdollisimman kompakteja [13], eli resurssit käytetään mahdollisimman tehokkaasti hyödyksi.
- Joillakin luennoitsijoilla on ajankohtia, jolloin he haluavat mieluiten luennoida [13, 16, 23]. Jotkut suosivat opetusajoissa esimerkiksi iltapäiväaikoja ja välttävät aikaisia tai vastaavasti myöhäisiä ajankohtia.
- Joillakin luennoitsijoilla on tietyt luentosalit, joissa he haluavat mieluiten luennoida [16, 23]. Jotkin luentosalit voivat olla esimerkiksi sijainniltaa mieluisampia kuin toiset.
- Luentojen kokonaismäärä päivässä voidaan rajata [13].
- Opiskelijoilla peräkkäisten oppimistapahtumien opetustilat ovat mahdollisimman lähekkäin, jolloin siirtyminen oppimistapahtumasta toiseen on sujuvaa [13]. Tämä on olennaista erityisesti yliopistoissa, joissa opetustilat ovat kaukana toisistaan, jolloin siirtymiseen opetustilojen välillä kuluu merkittävästi aikaa.
- Joillain kursseilla on enemmän kuin yksi luennoitsija [13], jolloin pitää huomioida useamman henkilön vapaanaolo kyseisellä tunnilla.
- Opiskelijoilla ei ole luentoja päivän viimeisillä tunneilla [1, 2, 8, 26]. Tämä rajoite johtaa suoraan siihen, ettei luennoitsijoillakaan ole luennoitavia kursseja päivien viimeisillä tunneilla.

- Opiskelijoilla ei ole enempää kuin kahden kurssin luentoja peräkkäin [1, 2, 8, 26]. Tämä tarkoittaa käytännössä sitä, että joko päivässä on maksimissaan vain 4 luentoa tai sitten opiskelijoille tulee hyppytunteja.
- Opiskelijoilla ei ole yhtä yksittäistä oppimistapahtumaa päivässä [1, 2, 8, 26].
- Jokin kurssi tulee järjestää ennen jotakin toista kurssia, tai vastaavasti vasta jonkin toisen kurssin jälkeen [16]. Tässä on kyse kurssien jakamisesta eri periodeihin, millä pyritään huomioimaan esitietovaatimusten täyttyminen ja jatkokurssien oikea järjestys.

3. OPTIMOINTIMALLI

Matemaattinen optimointi on parhaan mahdollisen ratkaisun etsimistä kaikkien mahdollisten ratkaisujen joukosta. Ongelmasta luodun matemaattisen mallin muuttujat on sidottu toisiinsa analyyttisillä yhtälöillä tai epäyhtälöillä, joita kutsutaan rajoitteiksi. Kaikki muuttujien arvot, jotka toteuttavat rajoitteet, muodostavat kaikkien mahdollisten ratkaisujen joukon. Näiden ratkaisujen joukosta voidaan etsiä paras mahdollinen ratkaisu tilanteesta riippuen joko minimoimalla tai maksimoimalla halutusta optimointikriteeristä muodostettava kohdefunktio. Optimointikriteerinä voi olla esimerkiksi kustannus tai aika, joka halutaan minimoida, tai tuotettava voitto, joka puolestaan halutaan maksimoida. Tavallisesti optimointi suoritetaan yhteen kriteeriin kerrallaan. On kuitenkin mahdollista optimoida myös useampaa kriteeriä kerrallaan, jolloin näistä muodostettava vektori toimii kohdefunktiona. [25]

Optimointiongelmat voidaan jakaa muuttujien ja rajoitefunktioiden tyyppien mukaan erilaisiin optimointiongelmiin. Jos jako tehdään muuttujien tyypin mukaan, optimointiongelmat voivat olla joko jatkuvia, kokonaislukuongelmia tai sekalukuongelmia. Jatkuvissa ongelmissa muuttujat voivat olla mitä tahansa reaalilukuja, kun taas kokonaislukuoptimoinnissa muuttujat voivat nimensä mukaisesti olla ainoastaan kokonaislukuja. Jos kokonaisluvut ovat kaikki binäärilukuja, kyseessä on binäärioptimointi. Sekalukuoptimoinnissa puolestaan osa muuttujista on reaalilukuja ja osa kokonaislukuja. [25]

Jos jako tehdään mallissa olevien funktioiden tyyppien mukaan, optimointiongelmat voidaan jakaa lineaariseen ja epälineaariseen optimointiin. Lineaarisesta optimoinnista on kyse silloin, kun kaikki ongelmassa mukana olevat funktiot ovat affiineja. Jos taas mukana on yksikin epälineaarinen funktio, kyseessä on epälineaarisen optimoinnin ongelma. Nämä kahden eri ominaisuuden mukaan jaotellut optimointiongelmat voidaan yhdistää, jolloin saadaan esimerkiksi lineaarinen kokonaislukuoptimointi, lineaarinen sekalukuoptimointi ja epälineaarinen sekalukuoptimointi. [25]

Yleisen epälineaarisen sekalukuoptimointiongelman muoto on [12]

$$\begin{aligned} &\min f(\mathbf{x}, \mathbf{y}) \\ &h_i(\mathbf{x}, \mathbf{y}) = 0, \quad i = 1, \ldots, r \\ &g_j(\mathbf{x}, \mathbf{y}) \leq 0, \quad j = 1, \ldots, s \\ &\mathbf{x} \in \mathbb{Z}^n, \quad \mathbf{y} \in \mathbb{R}^q. \end{aligned} \tag{3.1}$$

Ongelma voidaan muuttaa kokonaislukuongelmaksi määräämällä, että $q = 0$, eli $\mathbf{x} \in \mathbb{Z}^n$ ja $\mathbf{y} \in \mathbb{R}^0$, ja jatkuvaksi taas asettamalla $n = 0$ ja $q = 1$, eli $\mathbf{x} \in \mathbb{Z}^0$ ja $\mathbf{y} \in \mathbb{R}$.

Epälineaarisen sekalukuongelman käypä joukko Ω on

$$\Omega = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^n \times \mathbb{R}^q \mid h_i(\mathbf{x}, \mathbf{y}) = 0, \forall i = 1, \ldots, r \text{ ja } g_j(\mathbf{x}, \mathbf{y}) \leq 0, \forall j = 1, \ldots, s\}, \tag{3.2}$$

joka sisältää optimointiongelman käyvät ratkaisut. Lineaarinen ongelma sisältyy epälineaarisen ongelmaan, eli lineaarinen ongelma on yllä esitellyn ongelman erikoistapaus [25]. Yleinen muoto voidaan myös esittää aina minimointiongelmana, sillä maksimointiongelma saadaan muutettua minimointiongelmaksi [25]

$$\max f(\mathbf{x}, \mathbf{y}) = -\min(-f(\mathbf{x}, \mathbf{y})). \tag{3.3}$$

Jokainen $(\mathbf{x}, \mathbf{y}) \in \Omega$ on käypä ratkaisu, ja tarkoituksena on löytää tästä joukosta se ratkaisu $(\mathbf{x}, \mathbf{y})^*$, joka minimoi kohdefunktion. Tätä ratkaisua kutsutaan optimiratkaisuksi, ja ongelman optimitulos on $f(\mathbf{x}, \mathbf{y})^*$.

3.1 Kokonaislukuoptimointi

Kokonaislukuongelmassa kaikkien muuttujien on oltava kokonaislukuja. Jos nämä kokonaisluvut ovat erityisesti joko 0 tai 1, kyseessä on binäärioptimointi. Binäärioptimointia käytetään tyypillisesti silloin, kun tehdään valintoja kahden vaihtoehdon väliltä. Tällöin toinen päätös antaa muuttujalle arvon 0 ja toinen arvon 1.

Lineaarinen kokonaislukuongelma voidaan esittää muodossa [12]

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ & \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \\ & \mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2 \\ & \mathbf{x} \geq 0. \end{aligned} \tag{3.4}$$

Tässä tapauksessa ongelman käypä joukko on

$$\Omega = \{\mathbf{x} \in \mathbb{Z}_+^n \mid \mathbf{A}_1\mathbf{x} = \mathbf{b}_1, \quad \mathbf{A}_2\mathbf{x} \leq \mathbf{b}_2, \quad \mathbf{x} \geq 0\}. \tag{3.5}$$

Kokonaislukuoptimoinnissa ongelman muotoilun voi usein tehdä monella eri tavalla. Muotoilun valinnalla on kuitenkin merkitystä siihen, kuinka vaikea ongelma on ratkaista. Mitä suppeammaksi kokonaislukumuuttujien joukko alunperin määrätään, sitä helpompi ongelma on ratkaista. [17]

Kokonaislukuoptimoinnilla voidaan esittää kätevästi erilaisia rajoitteita. Rajoiteyhtälöitä saa olla useita, jolloin näistä muodostetaan yhtälöryhmä. Rajoitteiden muotoilussa käytetään usein binääristä valintamuuttujaa, joka saa arvonsa sen mukaan kumpi mahdollisista vaihtoehdoista tapahtuu [3]. Toisin sanottuna

$$x = \begin{cases} 1 & \text{jos valitaan vaihtoehto 1} \\ 0 & \text{jos valitaan vaihtoehto 2.} \end{cases}$$

Näiden valintamuuttujien avulla voidaan esimerkiksi valita yksi tai useampi vaihtoehto jostakin joukosta. Jos halutaan valita a määrä tekijöitä tarkasteltavasta joukosta, otetaan a rajoitteeksi. Epäyhtälö- tai yhtäsuuruusmerkillä määritetään se, halutaanko valita tasan, vähintään vai enintään a kappaletta tekijöitä. [3] Otetaan käyttöön merkintä $u \in \mathbb{P}_v$ ilmaisemaan, että $u \in \{1, \ldots, v\}$. Jos esimerkiksi halutaan, että korkeintaan a kappaletta valintamuuttujista saa arvon 1, määrätään

$$\sum_{i=1}^{n} x_i \leq a \qquad i \in \mathbb{P}_n. \tag{3.6}$$

Jos kahden tilanteen välillä on ekvivalenssi $x \Leftrightarrow y$, se voidaan mallintaa ehdolla [3]

$$x - y = 0, \tag{3.7}$$

jolloin joko molemmat tilanteet tapahtuvat tai kumpikaan ei tapahdu. Vastaavasti implikaatio $x \Rightarrow y$ voidaan esittää

$$x \leq y. \tag{3.8}$$

Kokonaislukuoptimoinnin sovelluksista suuri osa on nimettyjä standardimalleja, jotka ovat käyttökelpoisia monenlaisissa ongelmissa riippumatta siitä, mihin ne on alun perin luotu. Esimerkkejä tällaisista standardimalleista ovat muun muassa selkäreppuongelma, kauppamatkustajaongelma sekä järjestelyongelma [3, 17].

Selkäreppuongelmassa retkeilijän tulee valita reppuunsa mahdollisimman hyödylliset tavarat siten, että repun massa ei ylitä määrättyä painorajoitusta [17]. Ongelma voidaan esittää binäärimallilla, jolloin mukaan pakattava tavara saa arvon 1. Lisäksi jokaisella tavaralla on tietty massa ja tavarasta saatava hyötyarvo. Ongelmassa on tarkoituksena maksimoida tavaroista saatava hyöty huomioimalla asetettu painorajoite. [3] Ongelmasta on olemassa myös sellainen versio, jossa reppuun voidaan pakata useampia samaa tyyppiä olevia tavaroita, esimerkiksi taskulamppuja. Tällöin ongelma muuttuu binäärisestä ongelmasta kokonaislukuongelmaksi.

Kauppamatkustajaongelman periaate on kiertää ennaltamääritetyt paikkakunnat ja palata takaisin kotipaikkakunnalle. Jokaisella paikkakunnalla saa vierailla ainoastaan kerran, ja tehtävänä on löytää reitti, joka minimoi kuljetun kokonaismatkan. Kauppamatkustajaogelma on yksi käytetyimmistä kokonaislukuoptimoinnin standardimalleista. Mallia voidaan käyttää esimerkiksi tilanteissa, joissa tehtävänä on suorittaa joukko operaatioita jossakin järjestyksessä siten, että kustannukset minimoituvat. [17]

Kolmas esimerkki kokonaislukuoptimoinnin standardimalleista on järjestelyongelma, jossa muodostetaan pareja eri ryhmiin kuuluvista kohteista. Eri ryhmät voivat olla esimerkiksi naiset ja miehet, työt ja käytettävät koneet, tai vaikkapa oppilasryhmät ja opettajat. [17]

3.2 Lukujärjestysongelman matemaattinen malli

Esitetään seuraavaksi sekä koulujen että yliopistojen lukujärjestysongelmien mallit matemaattisesti. Olkoon opiskelijaryhmien joukko $R = \{r_1, r_2, \ldots, r_{n_k}\}$, jolloin ryhmien lukumäärä on n_k. Kouluissa nämä ryhmät ovat tavallisesti luokkia, ja yliopistossa ryhmät voivat muodostaa esimerkiksi saman opintosuunnan samana vuonna aloittaneet opiskelijat. Resurssit tulee jakaa käytettäviksi tunneille $\{a_1, a_2, \ldots, a_{n_j}\}$, jossa n_j on viikossa olevien tuntien kokonaismäärä, joka riippuu siitä, kuinka pitkiä tuntien halutaan olevan. Opiskelijaryhmien lisäksi muut tunneille jaettavat resurssit ovat oppimistapahtumat, kouluissa oppitunnit ja yliopistoissa luennot ja harjoitustapahtumat. Olkoon oppimistapahtumien joukko $T = \{t_1, t_2, \ldots, t_{n_i}\}$, jossa n_i on kaikkien oppimistapahtumien lukumäärä. Oppimistapahtumien joukkoon kuuluu siis kunkin aineen kukin yksittäinen tapahtuma. Opettajat ja luennoitsijat kullekin oppimistapahtumalle määrätään jo ennen resurssien jakoa, jolloin niitä ei tarvitse itse ongelmanratkaisussa ottaa huomioon. Jos opettajia halutaan joihinkin oppimistapahtumiin useampia, myös se määrätään etukäteen.

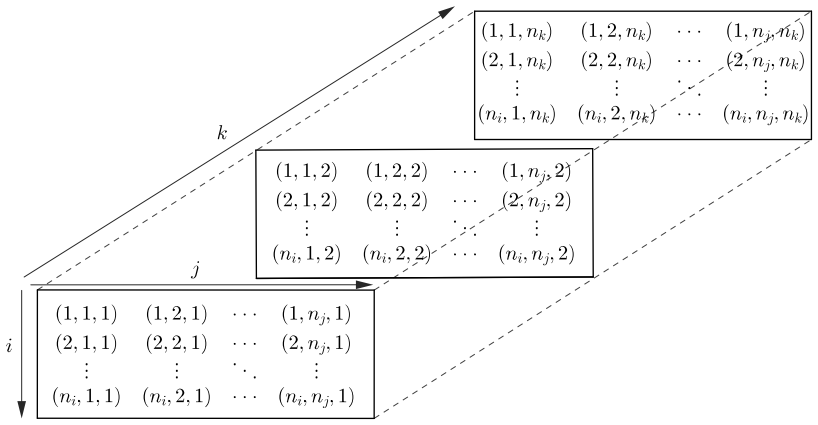
Koska sama ryhmä osallistuu viikon aikana useisiin oppimistapahtumiin, on määritettävä ne oppimistapahtumat, jotka tulee järjestää eri aikaan. Määritetään joukko $\{S_1, S_2, \ldots, S_p\}$, jossa kaikki ryhmään S_q kuuluvat oppimistapahtumat on järjestettävä eri aikoina. Lisäksi kaikki ryhmät eivät osallistu kaikkiin samoihin oppimistapahtumiin, joten on otettava käyttöön vielä kaksi uutta muuttujaa. Joukko $T_k \subset T$ sisältää ne kaikki n_{ik} oppimistapahtumaa, joihin ryhmä k osallistuu, ja joukko $R_i \subset R$ sisältää ne ryhmät, jotka osallistuvat tapahtumaan i.

Käytetään jatkossa oppimistapahtumista, tunneista sekä opiskelijaryhmistä vain indeksejä i, j ja k, ja otetaan käyttöön binäärinen muuttuja

$$x_{ijk} = \begin{cases} 1 & \text{jos oppimistapahtuma } i \text{ pidetään tunnilla } j \text{ ryhmälle } k \\ 0 & \text{muulloin,} \end{cases}$$

jossa $i \in \mathbb{P}_{n_i}, \quad j \in \mathbb{P}_{n_j}, \quad k \in \mathbb{P}_{n_k}$. Kuvassa 3.1 on havainnollistettu muuttujan olevan itse asiassa 3-ulotteinen taulukko. Muuttujan alaindeksi ijk osoittaa taulukon olevan 3-ulotteinen, joten käytetään tästä eteenpäin muuttujasta pelkkää nimitystä taulukko.

Koska kaikki ryhmät eivät välttämättä kaikkina ajankohtina voi osallistua oppimis-



Kuva 3.1 *Muuttujaa x_{ijk} voidaan kuvata 3-ulotteisella taulukolla, jossa indeksi i kuvaa oppimistapahtumia, indeksi j tunteja ja indeksi k opiskelijaryhmiä.*

tapahtumiin, otetaan käyttöön myös binäärimuuttuja

$$c_{jk} = \begin{cases} 1 & \text{jos ryhmä } k \text{ on vapaa tunnilla } j \\ 0 & \text{muulloin.} \end{cases}$$

Tulevaa hyppytuntien optimointia varten määritellään myös kaksi hyppytuntimuuttujaa. Määritellään ensin muuttuja yksittäiselle hyppytunnille

$$y_{jk} = \begin{cases} 1 & \text{jos ryhmällä } k \text{ on hyppytunti tunnilla } j \\ 0 & \text{muulloin.} \end{cases}$$

Tämän työn esimerkkitapauksessa yksi oppimistapahtuma sisältää kaksi 45 minuutin tuntia, eli periaatteessa yhteensä 2 tuntia. Vaikka oppimistapahtuma onkin ajassa mitattuna pituudeltaan 2 tuntia, kutsutaan sitä silti vain tunniksi. Toisin sanottuna siis yhden oppimistapahtuman kestoa kutsutaan nyt tunniksi sen ajallisesta pituudesta riippumatta. Hyppytuntimuuttuja y_{jk} kuvaa siis sitä, onko ryhmällä k yksittäinen hyppytunti vai ei. Kuten edellä määriteltiin, tämä yksittäinen hyppytunti on kuitenkin ajassa mitattuna 2 tuntia pitkä. Toinen vaihtoehto on, että hyppytun-

teja on kahden oppimistapahtuman verran. Tällöin puhutaan tuplahyppytunnista, joka kestää siis ajassa mitattuna 4 tuntia. Tätä tuplahyppytuntia varten otetaan käyttöön muuttuja

$$z_{jk} = \begin{cases} 1 & \text{jos ryhmällä } k \text{ on tuplahyppytunti tunneilla } j \text{ ja } j+1 \\ 0 & \text{muulloin.} \end{cases}$$

Nämä kaksi muuttujaa y_{jk} ja z_{jk} ovat taulukoita, joiden dimensioiden määrän alaindeksit osoittavat olevan 2. Kaikki kolme yllä määriteltyä muuttujaa x_{ijk}, y_{jk} ja z_{jk} muodostavat lineaarisen kokonaislukuongelman (3.4) ratkaisun $\mathbf{x}$. Muuttujat ovat esitetty havainnollisuuden vuoksi taulukoiden muodossa, mutta ongelmaa ratkaistaessa taulukot kerätään vektoreiksi $\mathbf{x}_{ijk}$, $\mathbf{y}_{jk}$ ja $\mathbf{z}_{jk}$. Näistä kolmesta vektorista muodostetaan vektori $\mathbf{x} = [\mathbf{x}_{ijk}^T, \mathbf{y}_{jk}^T, \mathbf{z}_{jk}^T]^T$.

3.2.1 Lukujärjestysongelman rajoitteet

Seuraavissa rajoitteissa käsitellään ainoastaan muuttujaa x_{ijk}. Muuttujat y_{jk} ja z_{jk} koskevat vain hyppytunteja, joten nyt käsiteltävissä rajoitteissa niiden arvoiksi tulee automaattisesti 0. Tästä syystä hyppytuntimuuttujia ei käsitellä tässä luvussa lainkaan, vaan niitä koskevat rajoitteet käsitellään myöhemmin luvussa 3.2.2.

Kaikista oleellisinta on, että kukaan opettaja ei pidä enempää oppimistapahtumia kuin viikossa on tunteja yhteensä, ja toisaalta ettei millään ryhmällä ole enempää oppimistapahtumia viikossa kuin on tunteja. Opettajien suurin sallittu tuntimäärä otetaan huomioon jo siinä vaiheessa, kun oppimistapahtumat jaetaan opettajille. Oleellista on myös se, että kukaan ei voi olla kahdessa eri paikassa samaan aikaan. Tämä johtaa myös siihen, ettei millään ryhmällä voi olla viikossa enempää oppimistapahtumia kuin tunteja on yhteensä. Jotta jokainen ryhmä osallistuu tietyllä tunnilla korkeintaan yhteen oppimistapahtumaan, on oltava

$$\sum_{i=1}^{n_i} x_{ijk} \leq c_{jk} \qquad \forall j \in \mathbb{P}_{n_j},\ \forall k \in \mathbb{P}_{n_k}. \tag{3.9}$$

Tämä rajoite (3.9) ottaa huomioon siis myös sen, ettei kaikki oppilasryhmät voi välttämättä osallistua oppimistapahtumiin kaikkina tunteina. Jos rajoitteessa (3.9) muuttuja c_{jk} saa arvon 1, jokainen ryhmä osallistuu tietyllä tunnilla korkeintaan

yhteen oppimistapahtumaan. Jos taas ryhmä ei ole kyseisellä tunnilla käytettävissä, alkion arvo nolla saa suoraan aikaan sen, että tapahtuma ei ole kyseisellä tunnilla mahdollinen. Tästä rajoitteesta seuraa suoraan se, ettei millään ryhmällä voi olla useampaa oppimistapahtumaa viikossa kuin on tunteja yhteensä.

Lisäksi rajoitteissa täytyy huomioida se, että oppimistapahtumia, joilla on samoja opiskelijoita, ei järjestetä samaan aikaan. Esimerkiksi jos ryhmään S_1 kuuluvat oppimistapahtumat t_1 ja t_2, niin nämä kaksi oppimistapahtumaa tulee järjestää eri aikaan. Muuttujan c_{jk} ollessa 1, jokaisella ryhmällä on rajoitteen (3.9) mukaan kullakin tunnilla korkeintaan 1 oppimistapahtuma, joten vaatimus tapahtumien t_1 ja t_2 eriaikaisesta järjestämisestä sisältyy kyseiseen rajoitteeseen. Jos sama opettaja opettaa useampia kursseja, on otettava huomioon myös se, ettei saman opettajan kurssien oppimistapahtumia voida järjestää samaan aikaan. Tämä tapaus käydään tarkemmin läpi myöhemmin rajoitteen (3.14) yhteydessä.

Jotta kukin oppimistapahtuma järjestetään ainoastaan kerran kullekin ryhmälle, vaaditaan, että

$$\sum_{j=1}^{n_j} x_{ijk} = 1 \qquad \forall k \in \mathbb{P}_{n_k},\ \forall i \in T_k. \tag{3.10}$$

Koska ehdon (3.10) on toteuduttava kaikilla muuttujan $i \in T_k$ arvoilla, ehto määrää myös sen, että kukin ryhmä osallistuu oikeaan määrään oppimistapahtumia.

Kouluissa halutaan yleensä, että vain yksi ryhmä osallistuu yhteen oppimistapahtumaan. Joskus voidaan kuitenkin haluta, että samaan aikaan järjestettävään oppimistapahtumaan osallistuu kaksi tai useampia ryhmiä. Otetaan käyttöön muuttuja lkm_i, joka kertoo kyseiseen oppimistapahtumaan i osallistuvien ryhmien lukumäärän. Tällöin oppimistapahtumaan i saadaan vaadittua haluttu määrä ryhmiä rajoitteella

$$\sum_{k=1}^{n_k} \sum_{j=1}^{n_j} x_{ijk} = \mathrm{lkm}_i \qquad \forall i \in \mathbb{P}_{n_i}. \tag{3.11}$$

Edellä rajoitteissa (3.10) ja (3.11) on määrätty, että kukin oppimistapahtuma järjestetään vain kerran kullekin ryhmälle, ja toisaalta, että kyseiseen oppimistapahtumaan osallistuu haluttu määrä ryhmiä. Lisäksi tarvitaan vielä rajoite, joka määrää, että kyseinen oppimistapahtuma on kaikilla ryhmillä samaan aikaan. Lukujärjestystä muodostettaessa voidaan myös haluta, että jotkut tapahtumat järjestetään muu-

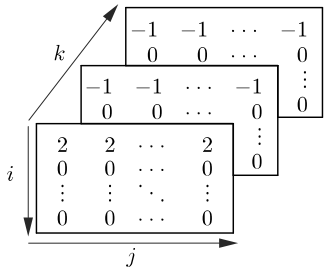
ten vaan samaan aikaan. Nämä molemmat tapaukset voidaan vaatia samanlaisella rajoitteella.

Käytännössä muodostettava rajoite tarkoittaa sitä, että muuttujataulukossa x_{ijk} tiettyyn tapahtumaan osallistuvat ryhmät joko kaikki osallistuvat oppimistapahtumaan tiettynä tuntina tai sitten mikään ryhmistä ei osallistu. Tämä tarkoittaa käytännössä sitä, että tapahtumaa vastaava muuttuja x_{ijk} on joko kaikilla tapahtumaan osallistuvilla ryhmillä 0 tai kaikilla 1. Tämä saadaan rajoitteeksi siten, että nollataulukkoon vaihdetaan sopivat kertoimet tiettyä tapahtumaa vastaaviin kaikkiin tunteihin sekä kaikkiin osallistuviin ryhmiin ja vaaditaan että summaus näiden ryhmien suhteen tuottaa arvoksi 0. Rajoite (3.10) ottaa jo huomioon sen, ettei samaa tapahtumaa pidetä samalle ryhmälle montaa kertaa, joten riittää, että nyt tarkastellaan vain ryhmiä. Oppimistapahtuman järjestäminen kaikille halutuille ryhmille samaan aikaan voidaan vaatia matemaattisesti

$$(\mathrm{lkm}_i - 1)\, x_{ij\kappa} - \sum_{k \in R_i \setminus \kappa} x_{ijk} = 0 \qquad \forall j \in \mathbb{P}_{n_j}, \tag{3.12}$$

jossa $\kappa \in R_i$ on jokin ryhmä joukossa R_i.

Jos esimerkiksi tapahtumaan i osallistuu 3 ryhmää, taulukkoon lisättävät kertoimet ovat näille kolmelle ryhmälle 2,-1 ja -1, jolloin summaksi saadaan nolla ainoastaan jos kaikki muuttujat ovat joko 0 tai kaikki ovat 1. Kertoimien sijoittelua on havainnollistettu kuvassa 3.2.



Kuva 3.2 *Kun tapahtuma halutaan järjestetää samaan aikaan kaikille kurssille osallistuville ryhmille, muuttujan x_{ijk} taulukkoon lisätään kertoimet siten, että rajoitteen toteutumiseksi kaikkien muuttujien on oltava joko 0 tai kaikkien 1.*

Rajoite (3.12) pätee myös silloin, jos ainoastaan yksi ryhmä osallistuu johonkin tapahtumaan, mutta rajoitetta ei tässä tapauksessa tarvita välttämättä lainkaan. Tässä esitetty tapa on vain yksi mahdollisuus vaatia, että jotkin tapahtumat järjestetään samaan aikaan kaikille osallistuville ryhmille. Toinen mahdollisuus on esimerkiksi poistaa taulukosta näitä tiettyjä tapahtumia vastaavat alkiot kaikilta muilta paitsi yhdeltä ryhmältä. Tällöin käsitellään 3-ulotteisen taulukon sijasta pelkkää matriisia, jolloin muuttujien määrä saadaan vähenemään.

Mikäli käsitellään ryhmien sijaan tapahtumia, eli halutaan järjestää tietyt tapahtumat samaan aikaan, kertoimet asetetaan ryhmien sijaan kyseisille tapahtumille. Tällöin joko kaikki tapahtumat järjestetään tietyllä tunnilla tai sitten yhtäkään kyseisistä tapahtumista ei järjestetä silloin.

Jos jonkin oppimistapahtuman ajankohta on ennaltamäärätty, voidaan se vaatia ehdolla

$$x_{ijk} \geq p_{ijk} \qquad i \in \mathbb{P}_{n_i},\ j \in \mathbb{P}_{n_j},\ k \in \mathbb{P}_{n_k}, \tag{3.13}$$

jossa

$$p_{ijk} = \begin{cases} 1 & \text{jos tapahtuma } i\text{, johon ryhmä } k \text{ osallistuu, on ennaltamäärätty ajalle } j \\ 0 & \text{muulloin.} \end{cases}$$

Otetaan huomioon myös se, että samaan aikaan ei voida järjestää useampaa kurssia kuin luokkatiloja on vapaana. Olkoon l_j suurin mahdollinen määrä, joka oppimistapahtumia voidaan järjestää samaan aikaan. Yliopistossa tämä tarkoittaa käytännössä vapaana olevien opetustilojen lukumäärää tunnilla j. Tällöin voidaan määrätä

$$\sum_{k=1}^{n_k} \sum_{i=1}^{n_i} \frac{1}{\mathrm{lkm}_i} \cdot x_{ijk} \leq l_j \qquad \forall j \in \mathbb{P}_{n_j}. \tag{3.14}$$

Saman opettajan opettaessa useaa kurssia, on otettava huomioon se, ettei opettaja voi olla kuin yhdessä paikassa samaan aikaan. Tämä voidaan vaatia rajoitteella (3.14) siten, että muuttujan l_j tilalle vaihdetaan arvo 1 osoittamaan, että tapahtumista korkeintaan 1 voidaan järjestää tietyllä tunnilla. Lisäksi ei tutkita kaikkia oppimistapahtumia, vaan ainoastaan ne tapahtumat i, joita sama opettaja opettaa.

Koska oppimistapahtumiin kuuluu samasta aineesta tai kurssista sekä luentoja että

harjoitustapahtumia, on joissain tapauksissa tärkeää määrittää myös oppimistapahtumien keskinäinen toteutusjärjestys. Voidaan esimerkiksi haluta, että tietyn kurssin luennot järjestetään ennen laskuharjoitustapahtumaa. Sen lisäksi että luennot halutaan järjestää ennen harjoituksia, on mahdollista myös järjestää saman kurssin luennot tiettyyn järjestykseen. Joskus voidaan myös haluta eri kurssien luennot tiettyyn järjestykseen. Esimerkiksi jos fysiikan kurssilla käytetään jotakin matemaattista työkalua, on hyvä, että asia ehditään käydä jo ennen fysiikan luentoa matikan luennolla läpi.

Yksi mahdollinen tapa toteuttaa tapahtumien järjestyksen vaatiminen on muodostaa järjestettävistä oppimistapahtumista pareja (i_1, i_2), joissa tapahtuma i_1 halutaan järjestää ennen tapahtumaa i_2. Joukko $T_\gamma \subset T \times T$ sisältää nämä oppimistapahtumaparit.

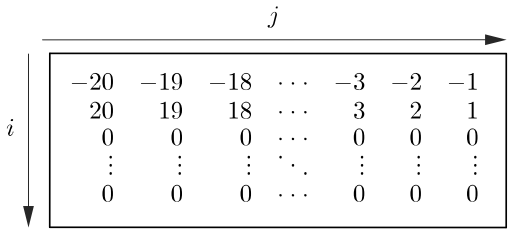
Rajoite (3.10) määrää, että kukin tapahtuma järjestetään kullekin ryhmälle vain kerran, eli kumpikin oppimistapahtuma saa yhden kertoimen sen mukaan, millä tunnilla oppimistapahtuma järjestetään. Määrätään tapahtuman i_1 kerroin negatiiviseksi ja tapahtuman i_2 positiiviseksi. Vaaditaan sitten, että näiden kertoimien summan on oltava negatiivinen. Jotta summa olisi negatiivinen, tapahtuman i_1 kertoimen itseisarvon tulee olla suurempi kuin tapahtuman i_2. Näiden vaatimusten toteutuessa tapahtuma i_1 järjestetään välttämättä ennen tapahtumaa i_2.

Muuttuja j_i kertoo monentenako tuntina oppimistapahtuma i järjestetään. Nyt rajoite kahden tapahtuman järjestykseen laittamiseksi voidaan kirjoittaa

$$\begin{gathered} -1 \cdot (-j_{i_1} + n_j + 1)x_{i_1jk} + (-j_{i_2} + n_j + 1)x_{i_2jk} < 0, \\ \forall k \in \mathbb{P}_{n_k}, \quad i_1, i_2 \in T_\gamma, \quad j \in \mathbb{P}_{n_j}. \end{gathered} \tag{3.15}$$

Tällä rajoitteella on siis mahdollista järjestää halutut tapahtumat tiettyyn järjestykseen. Samankaltaisella rajoitteella on mahdollista myös esimerkiksi määrätä halutut tapahtumat eri päiville tai vastaavasti jotkin tietyt tapahtumat järjestettäviksi peräkkäin. Kuvassa 3.3 havainnollistetaan kertoimia yhden ryhmän osalta tapaukselle, jossa viikko on jaettu 20 tuntiin, eli jokainen tunti on pituudeltaan 2 tuntia.

Todellisuudessa monilla kursseilla laskuharjoitukset kulkevat luentoja viikon myöhässä, jolloin edellisen viikon luentoasiat käsitellään laskuharjoituksissa vasta seuraavalla viikolla. Tällöin ei ole väliä, vaikka laskuharjoitustapahtuma olisikin viikon



***Kuva** 3.3 Kun halutaan järjestää kaksi tapahtumaa tietyssä järjestyksessä keskenään, kyseisten tapahtumien kohdalle kaikille tunneille annetaan kertoimet siten, että tapahtuman järjestysajankohdista määräytyvien kertoimien summa tulee negatiiviseksi silloin, kun haluttu tapahtuma järjestetään ensin. Kuvassa tapahtuma 1 halutaan järjestää ennen tapahtumaa 2, joten tapahtumaa 1 vastaavat kertoimet ovat negatiivisia.*

ensimmäinen oppimistapahtuma kyseiseltä kurssilta.

3.2.2 Hyppytuntien määritys

Lukujärjestysongelmassa yksi mahdollinen optimoitava asia on hyppytuntien lukumäärä. Tavoitteena on löytää rajoitteet, joilla hyppytuntimuuttujille saadaan arvot. Kun nämä arvot on saatu selville, voidaan luoda kustannusfunktio, jonka avulla hyppytuntien määrä pystytään minimoimaan.

Tämän työn esimerkkitapauksessa päivässä on 4 tuntia, joille oppimistapahtumia voidaan sijoittaa. Jokainen näistä tunneista on kestoltaan 2 tuntia. Kun päivän ensimmäistä eikä viimeistä tuntia lasketa hyppytunniksi, mahdollisia hyppytunteja voi olla päivän kahdella keskimmäisellä tunnilla. Toisin sanottuna hyppytunteja voi olla joko 1 tai 2, eli ajassa mitattuna joko 2 tai 4 tuntia.

Tutkitaan ensin yksittäisiä, eli ajalliselta kestoltaan 2 tuntia pitkiä hyppytunteja. Tutkitaan jokainen mahdollisen hyppytunnin paikka erikseen. Ryhmällä k on oppimistapahtuma tunnilla j, jos

$$\sum_{i=1}^{n_i} x_{ijk} = 1. \tag{3.16}$$

Jos ryhmällä k on hyppytunti tunnilla j, se tarkoittaa, että ryhmällä k on opetusta tunneilla $j-1$ ja $j+1$, mutta ei tunnilla j. Käytetään muuttujaa y_{jk}, joka määriteltiin aiemmin luvussa 3.2. Muuttuja siis saa arvon 1, mikäli ryhmällä k on hyppytunti tunnilla j ja arvon 0, mikäli hyppytuntia ei ole. Muuttuja y_{jk} riippuu siis muuttujan

x_{ijk} arvoista, joten näiden kahden muuttujan välille on luotava yhteys rajoitteiden avulla.

Lasketaan eri tunneille summat

$$\sum_{i=1}^{n_i} x_{ijk} \qquad \forall j \in \mathbb{P}_{n_j},\ \forall k \in \mathbb{P}_{n_k}, \tag{3.17}$$

jolloin kaikki mahdolliset kolmen tunnin kombinaatiot ovat 000, 001, 010, 100, 101, 110, 011 ja 111. Tavoitteena on siis löytää lineaarinen rajoite, jonka avulla voidaan löytää hyppytunnin sisältävä kombinaatio 101 näiden muiden kombinaatioiden joukosta. Yksi tapa löytää oikea lineaarinen rajoite on käyttää apuna luokitteluanalyysiä, jota on käsitelty kappaleessa 4. Luokitteluanalyysi voidaan suorittaa esimerkiksi MATLAB-ohjelmistoa apuna käyttäen.

Luokitteluanalyysista saadaan tuloksena parametrit, joiden avulla rajoitteet voidaan muodostaa. Rajoitteella

$$\left(\sum_{i=1}^{n_i} x_{i(j-1)k} - \sum_{i=1}^{n_i} x_{ijk} + \sum_{i=1}^{n_i} x_{i(j+1)k} \right) - y_{jk} \leq 1 \tag{3.18}$$

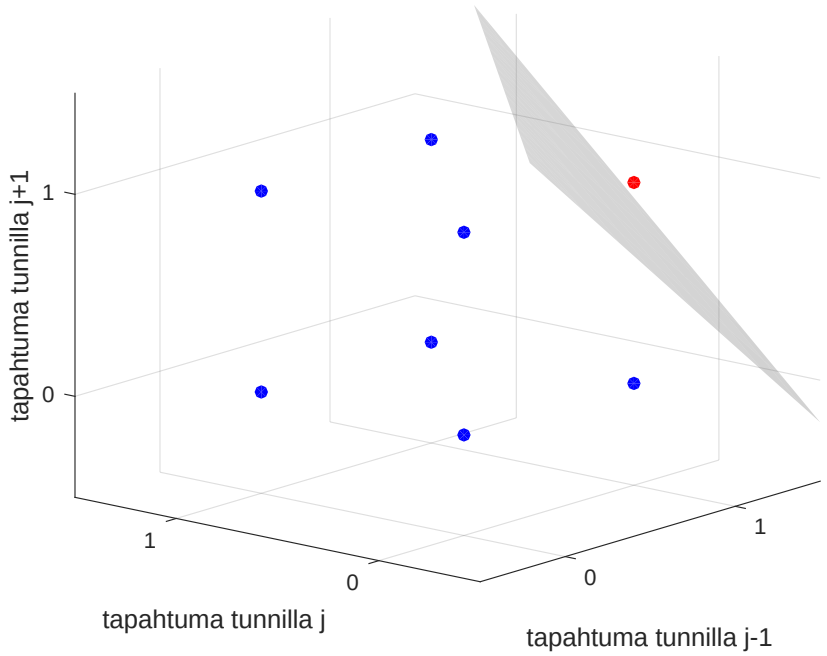
taataan, että muuttuja y_{jk} on 1, mikäli ryhmällä k on hyppytunti tunnilla j. Rajoite

$$\left(-\sum_{i=1}^{n_i} x_{i(j-1)k} + \sum_{i=1}^{n_i} x_{ijk} - \sum_{i=1}^{n_i} x_{i(j+1)k} \right) + 3y_{jk} \leq 1 \tag{3.19}$$

puolestaan takaa, että muuttuja y_{jk} on 0, mikäli ryhmällä k ei ole hyppytuntia tunnilla j.

Luokittelijaa havainnollistetaan kuvassa 3.4. Luokittelijana toimii taso, jonka toiselle puolelle jää ainoastaan erotettavaa kombinaatiota kuvaava piste. Tason parametrit saadaan myös luokitteluanalyysin tuloksena.

Edellä tarkasteltiin siis ainoastaan yhden hyppytunnin tilannetta. Tuplahyppytunnin, eli ajallisesti 4 tuntia pitkän hyppytunnin, käsittely etenee samalla tavoin kuin yksittäisen hyppytunnin tapauksessa. Tällöin tieto tuplahyppytunnista sisältyy muuttujaan z_{jk}. Rajoite



Kuva 3.4 *Kolmen tunnin eri kombinaatiot ja luokittelijana toimiva taso.*

$$\left(2\sum_{i=1}^{n_i} x_{i(j-1)k} - 2\sum_{i=1}^{n_i} x_{ijk} - 2\sum_{i=1}^{n_i} x_{i(j+1)k} + 2\sum_{i=1}^{n_i} x_{i(j+2)k}\right) - z_{jk} - z_{(j+1)k} \leq 2 \tag{3.20}$$

takaa, että mikäli ryhmällä k on tuplahyppytunti, eli hyppytunti sekä tunnilla j että $j+1$, muuttujat z_{jk} sekä $z_{(j+1)k}$ saavat arvon 1. Muuttujan z_{jk} arvon määrittämiseen tarvitaan jälleen myös toinen rajoite. Rajoitteella

$$\left(-2\sum_{i=1}^{n_i} x_{i(j-1)k} + 2\sum_{i=1}^{n_i} x_{ijk} + 2\sum_{i=1}^{n_i} x_{i(j+1)k} - 2\sum_{i=1}^{n_i} x_{i(j+2)k}\right) + 8z_{jk} \leq 4 \tag{3.21}$$

vaaditaan, että mikäli tuplahyppytuntia ei ole, muuttuja z_{jk} saa arvon 0. Samanlaisella rajoitteella saadaan arvo 0 myös muuttujalle $z_{(j+1)k}$.

3.2.3 Kustannusfunktion luominen

Pehmeistä rajoitteista ei muodosteta samanlaisia rajoite-ehtoja kuin kovista rajoitteista, sillä niiden toteutumatta jääminen ei estä lukujärjestyksen muodostamista. Pehmeiden rajoitteiden avulla tarkastellaan sen sijaan muodostettavien lukujärjestysten laatua. Lukujärjestystä muodostettaessa pehmeistä rajoitteista muodostetaan kustannusfunktio, jolle etsitään optimaalinen arvo siten, että kaikki kovat rajoitteet toteutuvat. [26]

Pehmeitä rajoitteita käsiteltäessä noudatetaan sakkomenetelmää, jolloin rajoitteen jäätyä toteutumatta siitä aiheutuu tietty sakko. Pehmeitä rajoitteita voi olla useita, ja on myös mahdollista, että niillä on erisuuruiset sakot sen mukaan, kuinka suuri painoarvo kunkin pehmeän rajoitteen toteutumisella on. Tällöin kohdefunktio voidaan esittää kaikkien rajoitteiden aiheuttamien sakkojen summana. [16]

Pehmeitä rajoitteita valitessa on myös mahdollista ottaa opettajien ja oppilaiden toiveita huomioon. Tällöin kuitenkin kustakin pehmeästä rajoitteesta tulevan sakon suuruus suhteessa muihin rajoitteisiin tulee miettiä. Lukujärjestysongelmassa voidaan esimerkiksi optimoida kaikkien oppimistapahtumien mahdollisimman tasaista jakautumista koko viikon ajalle tai pyrkiä minimoimaan yksittäisten oppimistapahtumien määrää viikon aikana. Opiskelijoille voidaan myös toivoa mahdollisimman paljon vapaapäiviä, jolloin siis täysin opetuksesta vapaiden päivien määrä pyritään maksimoimaan. Toinen vaihtoehto on esimerkiksi maksimoida vapaiden iltapäivätuntien tai vastaavasti aamutuntien määrää. Lisäksi opettajilla tai luennoitsijoilla saattaa olla toiveita oppimistapahtumien järjestämisajankohtiin. Edellä luetellut pehmeät rajoitteet ovat vain esimerkkejä, sillä kustannusfunktioon voi valita optimoitavaksi lähes mitä vaan.

Tässä työssä muodostettavassa esimerkkiongelmassa on lukujärjestyksen muodostamisen ohella tarkoitus minimoida hyppytuntien määrää viikon aikana. Toisessa esimerkkitapauksessa minimoidaan hyppytuntien lisäksi myös tilojen vuokrista aiheutuvia kustannuksia. Tavoitteena on siis luoda hyppytunneista ja tilakustannuksista kustannusfunktio, jolle etsitään minimiarvo. Kuten edellä on todettu, sekä hyppytunnille että tilakustannuksille voidaan antaa eri kustannus, joita merkitään nyt C_h ja C_t. Jotta hyppytuntien määrä voidaan laskea ja sen jälkeen minimoida, on ensin löydettävä hyppytunnit muiden tuntien joukosta.

Hyppytuntien etsimistä erottelu- ja luokitteluanalyysin avulla on käsitelty edellä lu-

vussa 3.2.2, jossa on määritelty, miten muuttujille y_{jk} ja z_{jk} saadaan arvot sen mukaan, onko kyseessä hyppytunti tai tuplahyppytunti vai ei. Hyppytuntien määrän minimoimiseksi muodostetaan kustannusfunktio $\mathbf{c}^T\mathbf{x}$ lineaarisen kokonaislukuongelman (3.4) mukaisesti. Vektori $\mathbf{c}$ sisältää kertoimet, jotka vastaavat vektorin $\mathbf{x}$ alkioita. Kun optimoidaan ainoastaan hyppytuntien määrää, vektoriin $\mathbf{c}$ laitetaan halutut kertoimet C_h ainoastaan hyppytuntivektoreita $\mathbf{y}_{jk}$ ja $\mathbf{z}_{jk}$ vastaaviin alkioihin ja kaikki muut alkiot nolliksi. Tällöin, jos jokin alkio hyppytuntivektorissa on 1, eli jollakin tunnilla on hyppytunti, kustannusfunktion arvo kasvaa halutun kertoimen C_h verran. Vektorin $\mathbf{x}_{ijk}$ alkion ollessa 1 kustannusfunktion arvo ei kuitenkaan kasva, sillä kertoimena on 0. Kun halutaan hyppytuntien lisäksi optimoida myös tilakustannuksia, vektoriin $\mathbf{c}$ lisätään lisäksi vektoria $\mathbf{x}_{ijk}$ vastaaviin alkioihin halutut kertoimet C_t.

4. EROTTELU- JA LUOKITTELUANALYYSI

Diskriminantti- eli erotteluanalyysi on tilastollinen menetelmä, jonka avulla pyritään erottamaan erilaiset tunnetut populaatiot toisistaan. Tavoitteena on löytää muuttujakombinaatio, jolla saadaan aikaan mahdollisimman suuri luokkien välinen vaihtelu suhteessa luokkien sisäiseen vaihteluun. Erotteluanalyysiin liittyy kiinteästi myös luokittelu, jossa erotteluanalyysistä saatuja tuloksia voidaan käyttää uuden havainnon sijoittamisessa johonkin jo olemassa olevaan luokkaan. [7]

Lähtökohtana erotteluanalyysissä ovat muuttujat, joiden ajatellaan eroavan eri luokkien välillä. Analyysissä on tarkoitus etsiä näiden muuttujien yhdistelmiä, joilla luokkien väliset erot saadaan mahdollisimman suuriksi. Näitä muuttujien yhdistelmiä, eli lineaarikombinaatioita kutsutaan erottelufunktioiksi. Jos eri luokkia on K kappaletta, erottelufunktioita voidaan muodostaa korkeintaan $K-1$ kappaletta.

Luokkien jakaumien parametreja ei yleensä tunneta tarkasti, vaan ne on arvioitava. Määritellään binäärimuuttuja

$$M_{nk} = \begin{cases} 1 & \text{jos havainto } n \text{ on luokassa } k \\ 0 & \text{muulloin.} \end{cases}$$

Määritetään sitten arvio luokan k keskiarvolle, joka saadaan kaavalla

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{n=1}^{N} M_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} M_{nk}}, \tag{4.1}$$

jossa N on havaintojen $\mathbf{x}_n$ kokonaismäärä. Luokkien sisäistä vaihtelua voidaan kuvata kovarianssimatriisilla [7]

$$\Sigma_w = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{n=1}^{N_k} \left(\mathbf{x}_n^{(k)} - \hat{\boldsymbol{\mu}}_k\right) \left(\mathbf{x}_n^{(k)} - \hat{\boldsymbol{\mu}}_k\right)^T, \tag{4.2}$$

jossa $\mathbf{x}_n^{(k)}$ on havainto n luokassa k, K luokkien kokonaismäärä ja N_k havaintojen lukumäärä luokassa k. Eri luokkien välinen kovarianssi voidaan puolestaan esittää [7]

$$\Sigma_b = \frac{1}{K-1} \sum_{k=1}^{K} N_k \left(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}\right) \left(\hat{\boldsymbol{\mu}}_k - \hat{\boldsymbol{\mu}}\right)^T, \tag{4.3}$$

jossa $\hat{\boldsymbol{\mu}}$ on kaikkien havaintojen yhteinen keskiarvo. Kokonaiskovarianssimatriisi saadaan summaamalla

$$\Sigma_{tot} = \Sigma_w + \Sigma_b. \tag{4.4}$$

Erottelufunktioksi etsitään lineaarikombinaatio $\mathbf{Y} = \mathbf{a}^T\mathbf{X}$, jossa $\mathbf{X}$ sisältää havainnot $\mathbf{x}_n$. Koska erotteluanalyysissä on tarkoituksena jakaa dataa tunnettuihin luokkiin, samalla maksimoiden suhdetta luokkien välisen hajonnan ja luokkien sisäisen hajonnan välillä, erottelufunktio muodostetaan siten, että saadaan

$$\max_{\mathbf{a}} \frac{\mathbf{a}^T\Sigma_b\mathbf{a}}{\mathbf{a}^T\Sigma_w\mathbf{a}}. \tag{4.5}$$

Tämä voidaan ratkaista rajoitettuna optimointiongelmana Lagrangen menetelmällä. Tällöin ehtona on, että $\mathbf{a}^T\Sigma_w\mathbf{a} = 1$. Yleisesti optimointiongelmalle [12]

$$\begin{aligned} &\max f(\mathbf{x}) \\ &\quad h(\mathbf{x}) = b \end{aligned} \tag{4.6}$$

voidaan kirjoittaa Lagrangen funktio [12]

$$L(\mathbf{x}, \lambda_L) = f(\mathbf{x}) + \lambda_L \cdot (b - h(\mathbf{x})), \tag{4.7}$$

jossa $\lambda_L \in \mathbb{R}$ on Lagrangen kerroin. Lagrangen menetelmällä optimointiongelmalle voidaan löytää käypä alue Lagrangen funktion ääriarvopisteessä, jossa $\nabla L = 0$.

Nyt optimoitavana on siis funktio (4.5) ehdolla $\mathbf{a}^T\Sigma_w\mathbf{a} = 1$. Tällöin

$$\begin{aligned} f(\mathbf{a}) &= \mathbf{a}^T\Sigma_b\mathbf{a} \\ h(\mathbf{a}) &= \mathbf{a}^T\Sigma_w\mathbf{a} \\ b &= 1. \end{aligned}$$

Nyt voidaan kirjoittaa Lagrangen funktio

$$L(\mathbf{a}) = \mathbf{a}^T\Sigma_b\mathbf{a} + \lambda_L\left(1 - \mathbf{a}^T\Sigma_w\mathbf{a}\right). \tag{4.8}$$

Derivoidaan tämä ja ratkaistaan sen nollakohta

$$\begin{aligned} \nabla L(\mathbf{a}) &= 2\Sigma_b\mathbf{a} - 2\lambda_L\Sigma_w\mathbf{a} = 0 \\ \Leftrightarrow \Sigma_b\mathbf{a} &= \lambda_L\Sigma_w\mathbf{a} && (4.9) \\ \Rightarrow \Sigma_w^{-1}\Sigma_b\mathbf{a} &= \lambda_L\mathbf{a}. && (4.10) \end{aligned}$$

Ratkaistaan ominaisarvoyhtälön (4.10) ominaisarvot ja niitä vastaavat ominaisvektorit. Kerrotaan yhtälö (4.9) vasemmalta vektorilla $\mathbf{a}^T$ ja käytetään lisäksi ehtoa $\mathbf{a}^T\Sigma_w\mathbf{a} = 1$, jolloin saadaan

$$\begin{aligned} \Sigma_b\mathbf{a} &= \lambda_L\Sigma_w\mathbf{a} \\ \Rightarrow \mathbf{a}^T\Sigma_b\mathbf{a} &= \lambda_L\mathbf{a}^T\Sigma_w\mathbf{a} \\ \Rightarrow \mathbf{a}^T\Sigma_b\mathbf{a} &= \lambda_L, && (4.11) \end{aligned}$$

eli funktion $f(\mathbf{a})$ arvo on sama kuin ominaisarvo. Tällöin haettu maksimiratkaisu on suurinta ominaisarvoa vastaava ominaisvektori.

5. ONGELMAN RATKAISEMINEN

Kokonaislukuoptimoinnin ongelmassa muuttujien mahdollisia arvoja on vähemmän kuin tavallisessa lineaarisessa optimointiongelmassa, minkä vuoksi voisi kuvitella, että kokonaislukuongelman ratkaiseminen olisi helpompaa kuin lineaarisen ongelman. Tämä ei kuitenkaan pidä paikkaansa, sillä ei ole olemassa mitään helppoa tapaa selvittää, onko löydetty piste optimiratkaisu. Piste voidaan todeta lokaaliksi optimiksi tutkimalla kohdefunktion arvoa viereisissä kokonaislukupisteissä, mutta toisin kuin lineaarisessa optimoinnissa, tämä ei kuitenkaan takaa pisteen olevan globaali optimi. [17]

Ongelman ratkaisemiseksi voisi tulla mieleen hakea ratkaisu ensin ilman kokonaislukurajoitetta, ja tämän jälkeen pyöristää saatu tulos lähimpään kokonaislukuun. Tämä ei kuitenkaan tuota ongelmalle oikeaa ratkaisua, vaan saatu optimi voi olla hyvin kaukana oikeasta optimista. [17]

Pienille ongelmille voidaan hakea ratkaisu käymällä kaikki vaihtoehdot läpi ja valitsemalla niistä se, joka antaa kohdefunktiolle parhaan arvon. Tällaista menetelmää kutsutaan täydellisen läpikäynnin menetelmäksi. [22] Käytännön tehtävät ovat kuitenkin usein melko suuria, jolloin tästä menetelmästä tulee käyttökelvoton. Tästä johtuen on täytynyt kehittää myös muita menetelmiä ongelman ratkaisemiseksi.

Yleisin ja luotettavimpana pidetty menetelmä lineaarisen kokonaislukuongelman ratkaisuun on branch and bound –algoritmi [3, 17]. Muita ratkaisualgoritmeja ovat esimerkiksi branch and cut [17], local search [22], lokaalisten hakualgoritmien luokkaan kuuluva simulated annealing (simuloitu jäähdytys) [22], tabu-search [17] sekä geneettiset algoritmit [17].

5.1 Branch and bound –menetelmä

Lineaarisen kokonaislukuongelman ratkaisemisessa käytettävä branch and bound –menetelmä on esimerkki implisiittisen läpikäynnin menetelmästä, jossa periaatteessa käydään läpi kaikki käyvät ratkaisut, mutta niitä ei kaikkia kuitenkaan konkreettisesti lasketa [17]. Menetelmässä muodostetaan alkuperäisestä ongelmasta aliongelmia, joiden ratkaisuista saadaan alkuperäiselle lineaariselle ongelmalle ylä- ja alarajoja. Näitä saatuja ylä- ja alarajoja voidaan parannella uusien aliongelmien avulla.

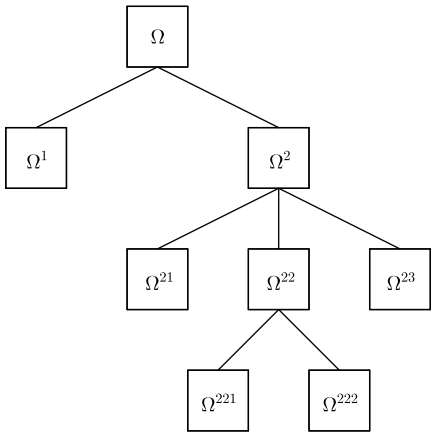
Relaksaatiossa on kyse ongelman käyvän alueen muuttamisesta ongelman ratkaisemisen helpottamiseksi [10]. Tämä voidaan tehdä esimerkiksi rajoitteita muuttamalla tai poistamalla. Verrataan kahta optimointitehtävää [10]

$$\mathbf{O}_1: \quad \min_{\mathbf{x}\in\Omega_1} f(\mathbf{x}) \qquad \text{ja} \qquad \mathbf{O}_2: \quad \min_{\mathbf{x}\in\Omega_2} f(\mathbf{x}).$$

Molemmissa optimointitehtävissä $\mathbf{O}_1$ ja $\mathbf{O}_2$ on samat kohdefunktiot, mutta käyvät alueet ovat molemmissa tehtävissä erit. Tällöin ongelmaa $\mathbf{O}_1$ sanotaan ongelman $\mathbf{O}_2$ relaksaatioksi jos $\Omega_2 \subset \Omega_1$ [10]. Branch and bound –menetelmää käytettäessä lineaarinen kokonaislukuongelma muutetaan ensin jatkuvaksi lineaariseksi ongelmaksi. Olkoon Ω tämän jatkuvan ongelman käypä joukko. Tämä joukko voidaan jakaa osiin $\Omega^i, i = 1, \ldots, k$ siten, että Ω on näiden osien yhdiste. Jakamista voidaan jatkaa edelleen, jolloin Ω^i jaetaan osiin $\Omega^{ij}, j = 1, \ldots, k_i$ ja niin edelleen. Näin aliongelmista muodostuu kuvassa 5.1 esitetty läpikäyntipuu, jota ei implisiittisessä läpikäynnissä tarvitse tutkia kokonaan. Sellaista osaa, jota ei tarvitse tutkia tarkasti, sanotaan karsituksi [17].

Branch and bound –menetelmässä aliongelmista muodostetaan alkuperäisen ongelman relaksaatioita rajoitteita muuttamalla. Tällöin kaikkia aliongelmia ei tarvitse käydä tarkasti läpi, vaan niistä käsitellään tarkasti vain lupaavalta näyttävät osat [10, 17]. Jos jonkin aliongelman ratkaisu on käypä ratkaisu, aliongelmalla ei ole ratkaisua tai ratkaisu antaa kohdefunktiolle heikomman arvon kuin jokin edellinen aliongelma, kyseisen aliongelman aliongelmia ei tarvitse käsitellä [17], eli kyseinen osa karsitaan.

Kun ratkaistaan kokonaislukuongelmaa branch and bound –menetelmällä, ongelma muutetaan ensin jatkuvaksi ongelmaksi ja ratkaistaan se. Jos ratkaisuksi saadaan kokonaislukuratkaisu, ratkaisun etsiminen lopetetaan. Jos osana ratkaisua on yksi-



Kuva 5.1 *Lineaarisen kokonaislukuongelman ratkaisemisessa voidaan hyödyntää relaksaatiota. Jatkuvaksi ja lineaariseksi muutetun alkuperäisen ongelman käypä joukko voidaan jakaa aliongelmiin ja muodostuneet aliongelmat jälleen uusiin aliongelmiin. Tästä aliongelmiin jaosta muodostuu kuvan mukainen läpikäyntipuu.*

kin kokonaisluvusta poikkeava luku, ratkaisun etsimistä jatketaan. Tällöin ratkaisun sallittu alue jaetaan uusien rajoitteiden avulla kahteen osaan, joista kumpaankaan aiempi ratkaisu ei kuulu. Näin saadaan kaksi aliongelmaa, joihin haetaan ratkaisua. Näin jatketaan kunnes ratkaistavia aliongelmia ei enää ole. Tämän jälkeen aliongelmien sallituista kokonaislukuratkaisuista valitaan kohdefunktiolle parhaimman arvon antava optimaalinen ratkaisu. [18]

Kun löydetään ensimmäinen sallittu kokonaislukuratkaisu, siitä saadaan maksimointitehtävässä kohdefunktion alaraja ja minimointitehtävässä vastaavasti yläraja. Jos maksimointitehtävässä minkä tahansa aliongelman ratkaisu antaa kohdefunktiolle alarajaa pienemmän arvon, tätä aliongelmaa ei enää tarvitse jakaa uusiksi aliongelmiksi, vaan sen käsittely voidaan lopettaa. Vastaavasti minimointiongelmassa minkään ylärajaa suurempaa kohdefunktion arvoa antavaa aliongelmaa ei jaeta uusiksi aliongelmiksi. Jos algoritmin edetessä löytyy parempi sallittu kokonaislukuratkaisu, yläraja ja alaraja voivat toki muuttua. Paremman ratkaisun löydyttyä muistetaan kuitenkin, että tätä aliongelmaa ei enää jaeta uusiksi aliongelmiksi, vaan tehtävää jatketaan mahdollisista muista aliongelmista kunnes kaikki on käyty loppuun. [18]

Menetelmän nopeuteen ja muistitilan tarpeeseen vaikuttaa aliongelmien käsittelyjärjestys. Seuraava käsiteltävä aliongelma voidaan valita esimerkiksi parhaan kohdefunktion arvon perusteella, eli maksimointitehtävässä suurimman ja minimointitehtävässä pienimmän arvon mukaan. Jaettavan aliongelman lisäksi pitää valita muuttuja, jolle uudet rajoitteet annetaan, eli mitä muuttujaa käytetään uusien aliongelmien muodostamiseen. Täksi muuttujaksi voidaan valita esimerkiksi se, jonka desimaaliosa on kauimpana viereisistä kokonaislukuarvoista. [18]

Branch and bound –menetelmä on siis tapa löytää lineaariselle kokonaislukuongelmalle ratkaisu. Menetelmän edetessä käsitellään kuitenkin lineaarisia aliongelmia, joille täytyy myös löytää ratkaisu. Ratkaisut näihin lineaarisiin aliongelmiin voidaan etsiä esimerkiksi simplex-menetelmällä.

5.2 Simplex-menetelmä

Simplex-menetelmä on käytetyin lineaarisen standardimuotoisen optimointiongelman ratkaisualgoritmi [6]. Menetelmän perusajatus on kulkea ongelman käyvän alueen reunaa pitkin ääripisteestä toiseen, kunnes löydetään optimaalinen ratkaisu [18]. Käyvän alueen muoto ratkeaa muuttujien määrän mukaan. Yleisesti jos muuttujia on m kappaletta, käypä alue on m-ulotteinen moni- tai hypertahokas. Alueen läpikäynti aloitetaan hypertahokkaan yhdestä kulmapisteestä, eli kantaratkaisusta, josta tarkastelua jatketaan johonkin sellaiseen naapurikulmaan, joka on lähempänä optimiratkaisua. Tällä tavoin edetään, kunnes löydetään optimaalinen kantaratkaisu. [18] Optimiratkaisu on siis lineaarisia rajoitteita sisältävän ongelman käyvän alueen nurkkapiste.

Lineaarisen standardimuodossa olevan optimointiongelman käypä alue K on konveksi, eli kaikille $x, y \in K$ ja $\lambda \in [0, 1]$ myös $\lambda x + (1 - \lambda)y \in K$. Alueen kärkipiste on sellainen piste, joka ei kuulu millekään joukon kahden muun pisteen välille vedetylle suoralle janalle. Toisin sanottuna konveksin joukon $K \in \mathbb{R}^n$ kärkipiste on piste x, jos ei ole olemassa kahta erillistä pistettä $y, z \in K$ siten, että $x = \lambda z + (1 - \lambda)y$ jollakin $\lambda \in (0, 1)$. [12]

Simplex-menetelmä soveltuu lineaarisen optimointiongelman ratkaisuun, jossa käytetään edellistä ratkaisua seuraavan ratkaisun etsimisen lähtökohtana. Tämä poikkeaa esimerkiksi useista sisäpistemenetelmistä, jotka joutuvat aloittamaan ratkaisun etsimisen joka kerta alusta. Sopivalla alkuarvojen valinnalla voi siis olla suurikin

merkitys ongelman ratkaisunopeuteen. [6]

Simplex-menetelmää voidaan käyttää siis standardimuodossa olevalle lineaariselle optimointiongelmalle. Tehtävä ei kuitenkaan välttämättä ole aluksi standardimuodossa, joten ensimmäiseksi ongelma täytyy muuttaa sellaiseksi. Yleinen standardimuotoinen lineaarinen ongelma on muotoa [12]

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0, \end{aligned} \tag{5.1}$$

jossa $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ ja $\mathbf{b} \in \mathbb{R}^m$. Standardimuotoinen ongelma on lineaarisen kokonaislukuongelman (3.4) kaltainen, mutta siitä puuttuvat epäyhtälörajoitteet. Jotta ongelma saadaan muutettua standardimuotoon, maksimointitehtävä on ensin muutettava minimointitehtäväksi yhtälön (3.3) mukaan. Tämän lisäksi epäyhtälöt on muutettava yhtälöiksi käyttämällä niin kutsuttuja pelivara- ja ylijäämämuuttujia. Nämä samat lisätyt muuttujat on lisättävä myös kohdefunktioon. Jotta nämä ylimääräiset muuttujat eivät vaikuta ratkaisuun, niiden kertoimiksi laitetaan kohdefunktiossa 0. Jos lisäksi jokin alkuperäinen muuttuja x_k voi olla negatiivinen, se voidaan kirjoittaa uudelleen muotoon $x_k = u_k - v_k$, jossa $u_k, v_k \geq 0$. Nämä uudet muuttujat sijoitetaan alkuperäisen muuttujan paikalle ongelmaan. Rajoite $x_k \leq 0$ voidaan korvata muuttujalla $y_k \geq 0$ lisäämällä ongelmaan muuttuja $y_k = -x_k$. Lisäksi rajoite $x_k \geq h$ voidaan korvata muuttujalla $y_k \geq 0$ siten, että $y_k = x_k - h$. [18]

Kun ongelma on saatu standardimuotoon, seuraava tehtävä on löytää jokin käypä kantaratkaisu ongelmalle. Tähän kantaratkaisuun kuuluvat yleensä pelkästään ylijäämä- ja pelivaramuuttujat, ja lisäksi kohdefunktion arvo on tällöin 0. [18] Olkoon $\mathrm{rank}(\mathbf{A}) = m$, eli matriisissa $\mathbf{A}$ on m kappaletta lineaarisesti riippumattomia pystyrivejä. Olkoon lisäksi m ensimmäistä pystyriviä nämä matriisin $\mathbf{A}$ lineaarisesti riippumattomat pystyrivit. Siten voidaan jakaa matriisi $\mathbf{A}$ kahteen osaan, ja kirjoittaa $\mathbf{A} = [\mathbf{B} \quad \mathbf{N}]$, jossa $\mathbf{B}$ on ei-singulaarinen $m \times m$ –matriisi. Vektori $\mathbf{x}$ voidaan jakaa vastaavasti

$$\mathbf{x} = \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix}.$$

Nyt voidaan kirjoittaa standardimuodon (5.1) yhtälö $\mathbf{A}\mathbf{x} = \mathbf{b}$ uuteen muotoon

$\mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b}$, josta edelleen ratkaistuna saadaan $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{Nx_N}$. Kantaratkaisu $\mathbf{x}$ lineaariselle optimointiongelmalle saadaan, kun $\mathbf{x_N} = 0$, jolloin

$$\mathbf{x} = \begin{bmatrix} \mathbf{B}^{-1}\mathbf{b} \\ 0 \end{bmatrix}.$$

Kantaratkaisu on lisäksi käypä kantaratkaisu, jos $\mathbf{x_B} \geq 0$. Toisaalta, vektori $\mathbf{x} \in \mathbb{R}^n$ on käypä kantaratkaisu jos ja vain jos se on lineaarisen optimointiongelman käyvän alueen kärkipiste. Matriisia $\mathbf{B}$ kutsutaan kannaksi ja vektorin $\mathbf{x_B}$ komponentteja kantamuuttujiksi. [12]

Ensimmäinen käypä kanta voidaan löytää lisäämällä alkuperäiseen, muotoa $\mathbf{A}_1\mathbf{x} \leq \mathbf{b}$ ja $\mathbf{x} \geq 0$ ja olevaan ongelmaan pelivaramuuttujat $\mathbf{x}_S$, jolloin saadaan

$$\mathbf{x} = \mathbf{A}_1\mathbf{x} + \mathbf{x}_S = \mathbf{A}_1\mathbf{x} + \mathbf{I}\mathbf{x}_S = [\mathbf{A}_1 \quad \mathbf{I}] \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_S \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_S \end{bmatrix},$$

jossa $\mathbf{A} = [\mathbf{A}_1 \quad \mathbf{I}]$. Tällöin ensimmäiseksi käyväksi kannaksi voidaan valita $\mathbf{B} = \mathbf{I}$, ja käyvät kantamuuttujat ovat tällöin pelivaramuuttujat $\mathbf{x}_S$, koska $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b} = \mathbf{b} \geq 0$. [12]

Simplex-algoritmissa siis valitaan aluksi ensimmäinen käypä kantaratkaisu $\mathbf{B}_0$, jota pidetään ensin käypänä kantana $\mathbf{B}$. Seuraavaksi ratkaistaan $\mathbf{x_B}$ yhtälöstä $\mathbf{Bx_B} = \mathbf{b}$, eli $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$. Tästä saadaan sen hetkinen kantaratkaisu $\mathbf{x} = [\mathbf{x_B}^{\mathbf{T}} \quad \mathbf{0}^{\mathbf{T}}]^{\mathbf{T}}$. Tämän jälkeen voidaan laskea redusoidut kustannukset $\hat{\mathbf{c}}_\mathbf{N}^\mathbf{T} = \mathbf{c}_\mathbf{N}^\mathbf{T} - \mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{N}$, jossa $\mathbf{B}^{-1}\mathbf{N}$ saadaan ratkaisemalla yhtälö $\mathbf{BX} = \mathbf{N}$. Jos saatu redusoitu kustannus $\hat{\mathbf{c}}_\mathbf{N} \leq 0$, ollaan löydetty optimaalinen kanta. Jos ei, niin valitaan indeksi s siten, että saadun redusoidun kustannusvektorin alkio s on negatiivinen. Poimitaan sitten matriisista $\mathbf{N}$ sarake s ja merkitään sitä $\hat{\boldsymbol{a}}_s$. Tämän jälkeen lasketaan $\hat{\boldsymbol{a}}_s^* = \mathbf{B}^{-1}\hat{\boldsymbol{a}}_\boldsymbol{s}$. Jos tuloksesi saadaan $\hat{\boldsymbol{a}}_s^* \leq 0$, voidaan tarkastelu lopettaa, sillä ongelma on rajoittamaton. [12] Muussa tapauksessa tarkastelua jatketaan laskemalla

$$\min_{\hat{\boldsymbol{a}}_{s,j}^* > 0} \frac{\mathbf{x}_{\mathbf{B},j}}{\hat{\boldsymbol{a}}_{s,j}^*} = \frac{\mathbf{x}_{\mathbf{B},r}}{\hat{\boldsymbol{a}}_{s,r}^*},$$

jossa j ja r vektoreiden alaindekseissä kuvaavat kyseisen vektorin alkioita. Tämän jälkeen korvataan matriisin $\mathbf{B}$ sarake r matriisin $\mathbf{N}$ sarakkeella s, ja aloitetaan ite-

raatiokierros alusta. Tällöin kantaan tuodaan muuttuja $\mathbf{x}_{\mathbf{N},s}$ ja vastaavasti poistetaan kantamuuttuja $\mathbf{x}_{\mathbf{B},r}$. Tämän jälkeen uutta iteraatiokierrosta jatketaan ratkaisemalla uuden kannan avulla uusi kantaratkaisu. Tätä jatketaan kunnes löydetään optimaalinen ratkaisu, tai kunnes on suoritettu etukäteen päätetty määrä iteraatiokierroksia. [18]

6. ESIMERKKIONGELMA

Muodostetaan yksinkertainen esimerkkitapaus, jolle luodaan lukujärjestykset käyttäen MATLAB-ohjelmistoa. Esimerkissä tutkitaan myös, miten opetustilojen määrän muutos ja eri kustannusfunktiot vaikuttavat ongelman ratkaisuna saataviin lukujärjestyksiin. Esimerkkitapauksessa on 5 opiskelijaryhmää, jotka kaikki opiskelevat Tampereen teknillisessä yliopistossa teknis-luonnontieteellisessä koulutusohjelmassa. Lukujärjestykset luodaan kaikkiaan yhdeksästä kurssista, joista kaksi ovat sellaisia, joille kaikki ryhmät osallistuvat, ja loput sellaisia, joille vain osa ryhmistä osallistuu.

Opiskelijaryhmät on jaettu pääaineiden mukaan siten, että ryhmät 1 ja 2 opiskelevat matematiikkaa, ryhmät 3 ja 4 fysiikkaa sekä ryhmä 5 kemiaa. Kukin ryhmä osallistuu periodin aikana kahdelle tai kolmelle oman pääaineensa kurssille sekä kahdelle kaikille pakolliselle kurssille. Kukin kurssi sisältää tietyn määrän oppimistapahtumia, joihin sisältyvät sekä luennot että mahdolliset harjoitustapahtumat. Kullekin luennolle osallistuvat kaikki kyseiselle kurssille osallistuvat ryhmät, mutta kuhunkin harjoitustapahtumaan osallistuu vain yksi ryhmä. Kurssit, niiden oppimistapahtumien lukumäärät viikossa sekä kullekin kurssille osallistuvat ryhmät on esitelty taulukossa 6.1. Kuten taulukosta ilmenee, viikon aikana on yhteensä 33 oppimistapahtumaa, joista 15 on luentoja ja loput 18 harjoitustapahtumia.

Oppimistapahtumat sijoitellaan lukujärjestykseen aikavälille klo 8–16 kullekin arkipäivälle. Jokainen oppimistapahtuma sisältää kaksi 45 minuutin opetusosiota, joiden välissä on 15 minuutin tauko. Lisäksi kukin oppimistapahtuma järjestetään yhdellä tunnilla, mutta kuten luvussa 2.1 todettiin, tunti ei aina ole pituudeltaan 60 minuuttia. Tässä esimerkkitapauksessa kukin tunti on pituudeltaan 120 minuuttia, jolloin jokaisena arkipäivänä on 4 tuntia ja koko viikolla 20 tuntia, joiden aikana oppimistapahtumat tulee järjestää.

Taulukko 6.1 *Esimerkkiongelman kurssit, oppimistapahtumien lukumäärät sekä kursseille osallistuvat opiskelijaryhmät.*

kurssi	luentojen lkm	harjoitusten lkm	ryhmät
ohjelmointi	2	5	1,2,3,4,5
teollisuustalous	1	5	1,2,3,4,5
differentiaaliyhtälöt	2	2	1,2
LaTeX -kurssi	1	2	1,2
laaja fysiikka	2	2	3,4
fysiikan seminaari	1	0	3,4
fysikaalinen kemia	2	1	5
laboratoriotyöturvallisuus	2	0	5
biotekniikka	2	1	5

6.1 Esimerkkiongelman rajoitteet

Kuten aiemmin kappaleessa 2 todettiin, lukujärjestys muodostetaan tiettyjä rajoitteita noudattaen. Rajoitteissa täytyy vaatia, että kukin oppimistapahtuma järjestetään vain kerran ja että se pidetään oikealle määrälle ryhmiä. Rajoitteen (3.11) mukaisesti tällöin on käsiteltävä jokainen oppimistapahtuma erikseen, joten yhtälörajoitteita muodostuu $n_i = 33$ kappaletta. Tämä rajoite on esitetty MATLAB-koodina liitteessä A riveillä 58–70.

Tämän lisäksi on vaadittava, että tapahtumat, joihin osallistuu useampia ryhmiä, tulee järjestää samaan aikaan kaikille osallistuville ryhmille. Tässä tapauksessa nämä tapahtumat ovat ainoastaan luentoja, sillä kuhunkin harjoitustapahtumaan osallistuu vain yksi ryhmä. Kursseja, joille osallistuu useampia ryhmiä ovat ohjelmointi, teollisuustalous sekä 2 matematiikan ja 2 fysiikan kurssia. Näiltä kursseilta rajoitteissa tulee huomioida kukin erillinen luento. Lisäksi, koska ei tiedetä, milloin luennot järjestetään, rajoitteissa on käsiteltävä kaikki tunnit rajoitteen (3.12) mukaisesti. Tällöin saadaan yhteensä $(\textit{luentojen lkm}) \cdot n_j = (2+1+2+1+2+1) \cdot 20 = 180$ yhtälörajoitetta, joiden muodostuminen on esitetty liitteen A riveillä 74–105.

Lukujärjestystä muodostettaessa on oltava tiedossa myös se, mitkä ryhmät osallistuvat millekin kurssille. Tämä tieto ilmaistaan rajoitteen muodossa kääntäen siten, että määrätään, mihin oppimistapahtumiin kukin ryhmä ei osallistu. Toisin sanottuna ryhmät 3,4 ja 5 eivät osallistu matematiikan kursseille, ryhmät 1,2 ja 5 fysiikan kursseille eivätkä ryhmät 1,2,3 ja 4 kemian kursseille. Liitteen A riveiltä 115–128 nähdään, miten näistä jokaisesta tapauksesta muodostetaan 1 yhtälörajoite. Tällöin kyseisestä tapauksesta muodostuu yhteensä 3 yhtälörajoitetta.

Kursseilla, joille osallistuu useita ryhmiä, harjoitustapahtumia on useita, mutta jokainen ryhmä osallistuu niistä vain yhteen. Lisäksi jokaiseen harjoitustapahtumaan osallistuu vain yksi ryhmä. Tämän vuoksi rajoitteella täytyy vaatia myös se, että kaikille ryhmille järjestetään oikea määrä kunkin kurssin harjoitustapahtumia. Tätä rajoitetta tarvitaan ohjelmoinnin, teollisuustalouden, kahdelle matematiikan kurssille sekä yhdelle fysiikan kurssille. Käytetään perustana rajoitetta (3.10) ja kirjoitetaan liitteen A riveille 139–167 rajoitteet koodin muodossa. Jokaisen käsiteltävän kurssin kutakin osallistuvaa ryhmää kohti saadaan 1 yhtälörajoite, jolloin yhtälörajoitteita saadaan (*käsiteltäville kursseille osallistuvien ryhmien lkm*) $= 5+5+2+2+2 = 16$ kappaletta.

Yksi tavoiteltava asia lukujärjestyksissä on se, ettei niillä kursseilla, joille tietty opiskelijaryhmä osallistuu, ole päällekkäisyyksiä. Tämä taataan rajoitteella (3.9), joka vaatii, että jokaisella ryhmällä voi olla kullakin tunnilla korkeintaan yksi oppimistapahtuma. Reaalimaailmassa tämä ei aina voi toteutua, vaan on väistämätöntä, että osalla kursseista on päällekkäisyyksiä ainakin joidenkin opiskelijoiden lukujärjestyksissä. Tässä esimerkkitapauksessa päällekkäisyyksien välttäminen on mahdollista, sillä ongelma on suhteellisen pieni. Jotta saadaan taattua, ettei millään ryhmällä ole samalla tunnilla useata oppimistapahtumaa, tulee rajoitteissa käsitellä kaikki ryhmät ja jokaisen ryhmän kohdalla kaikki tunnit, kuten riveiltä 173–180 liitteestä A ilmenee. Tällöin epäyhtälörajoitteita saadaan yhteensä $n_k \cdot n_j = 5 \cdot 20 = 100$ kappaletta.

Lukujärjestystä muodostettaessa on tärkeää huomioida kurssin opettajien ja opiskelijaryhmien lisäksi myös tilojen vapaanaolo rajoitteen (3.14) mukaisesti. Tässä esimerkissä on käytettävissä 1 luentosali ja 2 harjoitussalia, mikä rajoittaa sitä, kuinka monta luentoa tai harjoitustapahtumaa voidaan järjestää samaan aikaan. Molemmissa tapauksissa rajoitteita muodostettaessa on käsiteltävä kaikki tunnit, jolloin epäyhtälörajoitteita muodostuu yhteensä $n_j + n_j = 20 + 20 = 40$ kappaletta. Harjoitus- ja luentosalien määrää koskevat rajoitteet on esitetty liitteen A riveillä 186–196 ja 204–214. Esimerkissä tarkastellaan yhtenä tapauksena myös sitä, miten harjoitussalien lukumäärän vähentäminen vaikuttaa saataviin lukujärjestyksiin. Tämä muutos ei kuitenkaan vaikuta ongelman rajoitteiden määrään.

Kuten yleensä lukujärjestysongelmissa, myös tässä esimerkissä oppimistapahtumien luennoitsijat ja harjoitusten pitäjät on päätetty etukäteen, joten lukujärjestyksiä muodostettaessa niitä ei tarvitse huomioida. Tässä esimerkkitapauksessa kaikkia

kursseja luennoivat eri henkilöt, mutta saman kurssin kaikki laskuharjoitustapahtumat pitää sama henkilö. Tällöin on otettava rajoitteita luotaessa huomioon se, ettei useita saman kurssin laskuharjoitustapahtumia voi järjestää samaan aikaan. Tämän rajoitteen pohjana voidaan käyttää rajoitetta (3.14), jossa muuttuja $l_j = 1$. Lisäksi rajoitteessa ei summata kaikkia ryhmiä ja kaikkia tapahtumia, vaan ainoastaan käsiteltävät harjoitustapahtumat ja niihin osallistuvat ryhmät. Rajoitteet on esitetty koodin muodossa liitteessä A riveillä 223–259. Kursseja, joilla on useampia harjoitustapahtumia on tässä esimerkkitapauksessa viisi: ohjelmointi, teollisuustalous, differentiaaliyhtälöt, LaTeX-kurssi ja laaja fysiikka. Näistä jokaiselle pitää muodostaa rajoite erikseen. Lisäksi, koska ei tiedetä, millä tunnilla tapahtumat järjestetään, jokaisessa rajoitteessa tulee käydä kaikki 20 tuntia läpi. Tästä muodostuvat rajoitteet ovat epäyhtälörajoitteita ja niitä muodostuu yhteensä $(\textit{käsiteltävien kurssien lkm}) \cdot n_j = 5 \cdot 20 = 100$ kappaletta.

Edellä lueteltujen rajoitteiden lisäksi halutaan, että saman kurssin luennot järjestetään eri päivinä. Tämä rajoite tarvitaan jokaiselle kurssille, jolla on 2 luentoa viikossa. Näitä kursseja on yhteensä 6 kappaletta, eli kaikki muut, paitsi teollisuustalous, LaTeX-kurssi ja fysiikan seminaari. Kuten liitteen A riveiltä 266–298 ilmenee, jokaista käsiteltävää kurssia koskevassa rajoitteessa pitää käydä kaikki 5 opetuspäivää läpi. Tällöin epäyhtälörajoitteita muodostuu käsiteltävien kurssien osalta yhteensä $(\textit{käsiteltävien kurssien lkm}) \cdot (\textit{opetuspäivien lkm}) = 6 \cdot 5 = 30$ kappaletta.

Sen lisäksi että saman kurssin luennot halutaan järjestää eri päivinä, saman kurssin luennot halutaan järjestää myös ennen kyseisen kurssin laskuharjoitustapahtumia. Järjestykseen halutuista kursseista muodostetaan luvun 3.2.1 määritysten mukaan pareja, joita muodostuu tässä esimerkissä 24 kappaletta. Lisäksi rajoitteita muodostettaessa jokaisen tapahtumaparin kohdalla käsitellään kaikki ryhmät. Tällöin epäyhtälörajoitteita muodostuu $(\textit{tapahtumaparien lkm}) \cdot n_k = 24 \cdot 5 = 120$ kappaletta. Tämä rajoitteen (3.15) mukainen käsittely on esitetty liitteessä A riveillä 304–317.

Kuten luvussa 3.2.1 matemaattisen mallin rajoitteita käsiteltäessä todettiin, kaikki opettajat eivät välttämättä ole käytettävissä kaikilla tunneilla. Koska ohjelmoinnin kurssin luennoitsija ja harjoitusten pitäjä ovat estyneitä pitämään opetusta maanantaisin ja perjantaisin, tässä esimerkissä otetaan huomioon se, ettei ohjelmoinnin kurssin oppimistapahtumia voida järjestää niinä päivinä. Perustana tässä voidaan käyttää rajoitetta (3.9), mutta rajoitteena on opiskelijaryhmien sijaan opettajien va-

paana olo. Rajoite on esitetty liitteen A riveillä 324–326, ja tilanteesta muodostuu 1 epäyhtälörajoite.

Lopuksi rajoitteista käsitellään hyppytuntimuuttujien arvot määrittävät rajoitteet. Jotta hyppytuntimuuttujille saadaan arvo 0 tai 1 sen mukaan, onko hyppytuntia vai ei, tarvitaan rajoitteita (3.18)–(3.21), jotka on esitelty luvussa 3.2.2. Yksittäiset hyppytunnit ja tuplahyppytunnit käsitellään rajoitteita muodostettaessa erikseen. Kun molemmissa määritellään, ettei päivien ensimmäiset eivätkä viimeiset tunnit voi olla hyppytunteja, muodostuu yhteensä 2 yhtälörajoitetta. Nämä vaatimukset on esitetty liitteen A riveillä 351–354 ja 399–402. Tämän lisäksi yksittäisiä hyppytunteja käsiteltäessä käsitellään kaikki ryhmät ja kaikkien päivien 2 keskimmäistä tuntia, jolloin hyppytunteja on mahdollista olla. Lisäksi jokaista ryhmää ja tuntia kohti tulee 2 rajoitetta, jotka antavat muuttujalle arvon 0 tai 1 sen mukaan, onko kyseessä hyppytunti vai ei. Tällöin epäyhtälörajoitteita saadaan liitteen A rivien 357–378 mukaisesti yhteensä $n_k \cdot (\textit{mahdollisten hyppytuntien lkm viikossa}) \cdot 2 = 5 \cdot 10 \cdot 2 = 100$ kappaletta.

Samoin käsitellään tuplahyppytunnit. Kunakin päivänä on mahdollista olla 1 tuplahyppytunti, eli viikossa niitä on mahdollista olla 5 jokaisella ryhmällä. Lisäksi jokaista ryhmää ja tuplahyppytuntia kohti saadaan 3 epäyhtälörajoitetta. Yksi rajoite määrää, että jos päivällä on tuplahyppytunti, muuttuja saa niillä tunneilla arvon 1. Kaksi muuta rajoitetta puolestaan määräävät kyseisten kahden tunnin kohdalle muuttujaan arvon 0, mikäli kyseessä ei ole tuplahyppytunti. Tällöin hyppytuntimuuttujan arvon määrittämisestä muodostuu epäyhtälörajoitteita yhteensä $n_k \cdot (\textit{mahdollisten tuplahyppytuntien lkm viikossa}) \cdot 3 = 5 \cdot 5 \cdot 3 = 75$. Tuplahyppytuntimuuttujan arvot määräävät rajoitteet on esitetty koodina liitteen A riveillä 408–438.

Jokaisesta edellä esitellystä rajoitteesta muodostuu yksi rivi joko lineaarisen kokonaislukuongelman (3.4) matriisiin $\mathbf{A}_1$ tai $\mathbf{A}_2$ sen mukaan, onko kyseessä yhtälö- vai epäyhtälörajoite. Yhtälörajoitteet sisältävään matriisiin $\mathbf{A}_1$ saadaan tällöin $33 + 60 + 120 + 3 + 16 + 2 = 234$ riviä ja epäyhtälörajoitteet sisältävään matriisiin $\mathbf{A}_2$ puolestaan $100 + 1 + 30 + 120 + 40 + 100 + 100 + 75 = 566$ riviä. Rajoitteita tässä esimerkkiongelmassa on yhteensä siis $234 + 566 = 800$ kappaletta.

Vektoriin $\mathbf{x}$ puolestaan on kerätty peräkkäin kaikki taulukoiden x_{ijk}, y_{jk} ja z_{jk} alkiot. Taulukon x_{ijk} alkioiden määrä on $n_i \cdot n_j \cdot n_k = 33 \cdot 20 \cdot 5 = 3300$. Sekä taulukon y_{jk} että z_{jk} alkioiden määrä on $n_k \cdot n_j = 5 \cdot 20 = 100$, eli jokaisen ryhmän kohdalla käsitellään

kaikki tunnit. Täten vektorin $\mathbf{x}$ pituudeksi saadaan $3300 + 100 + 100 = 3500$.

6.2 Kustannusfunktio

Tässä esimerkissä käytetään yhteensä kahta eri kustannusfunktiota. Kustannusfunktion muodostuksen perusasiat on käsitelty aiemmin luvussa 3.2.3. Ensimmäisessä tapauksessa lukujärjestys halutaan muodostaa siten, että opiskelijoiden lukujärjestyksiin muodostuu mahdollisimman vähän hyppytunteja. Kuten hyppytuntien käsittelyssä luvussa 3.2.2 määrättiin, päivien ensimmäisiä eikä viimeisiä tunteja lasketa hyppytunneiksi. Tässä esimerkissä jokainen hyppytunti, riippumatta siitä, onko kyseessä yksittäinen hyppytunti vai toinen tuplahyppytunnin tunneista, aiheuttaa kustannuksen $C_h = 1$. Koska jokaisena opetuspäivänä voi olla 2 hyppytuntia, mahdollisista hyppytunneista koituvien kustannusten maksimimäärä on $2 \cdot (\textit{opetuspäivien lkm viikossa}) \cdot C_h = 2 \cdot 5 \cdot 1 = 10$.

Esimerkissä testataan myös tilakustannusten vaikutusta ratkaisuna saataviin lukujärjestyksiin. Opetustilojen käytöstä aiheutuu kustannuksia, jotka riippuvat ajankohdista, jolloin opetustiloja käytetään. Keskellä päivää, eli klo 10–14 järjestettävillä oppimistapahtumilla kustannukset ovat suuremmat kuin päivien ensimmäisillä ja viimeisillä tunneilla. Tästä syystä toisessa tapauksessa pyritään hyppytuntien määrän ohella minimoimaan myös opetustilojen käytöstä aiheutuvia kustannuksia. Lisäksi esimerkissä testataan kahta eri vaihtoehtoa tilakustannuksen suuruudelle. Jokaisen hyppytunnin aiheuttama kustannus C_h on edelleen 1, ja jokainen välillä klo 10–14 järjestettävä oppimistapahtuma aiheuttaa joko kustannuksen $C_t = 1{,}2$ tai $C_t = 2$. Jos samaan oppimistapahtumaan osallistuu useita ryhmiä, kustannus jaetaan jokaiselle osallistuvalle ryhmälle siten, että kustakin klo 10–14 järjestetystä oppimistapahtumasta aiheutuu sama kustannus riippumatta siitä, kuinka monta ryhmää siihen osallistuu.

6.3 Tulokset

Esimerkkiongelman ratkaisut etsitään MATLAB R2015b –ohjelmistolla tietokoneella, jonka suoritin on AMD A8-7100. Liitteessä A on esitelty MATLAB-koodi, joka sisältää luvuissa 6.1 ja 6.2 esitellyt rajoitteet sekä kustannusfunktion. MATLAB luo ryhmille automaattisesti lukujärjestykset näitä rajoitteita noudattaen ja antaa myös kyseisten lukujärjestysten muodostaman kustannusfunktion arvon.

Haetaan ensin ratkaisu ongelmalle, jossa minimoitavana on ainoastaan hyppytuntien kokonaismäärä. MATLAB löytää ongelmalle ratkaisun noin 10 sekunnissa, eikä lukujärjestyksiin muodostu yhtään hyppytuntia. Lisäksi huomataan, että ratkaisu ei ole yksikäsitteinen, vaan ratkaisuja löytyy jopa tuhansia. Liitteissä B.1 ja B.2 on esitetty kahden ratkaisun mukaiset lukujärjestykset. Useita eri ratkaisuja voidaan etsiä esimerkiksi lisäämällä kustannusfunktioon pieni satunnaislukukerroin. Yksi esimerkki muodostuvista lukujärjestyksistä on esitetty myös taulukossa 6.2.

Taulukko 6.2 *Ryhmän 1 lukujärjestys tapauksessa, jossa on käytössä 2 harjoitussalia. Lyhenteellä TETA merkitään teollisuustalouden kurssia, lyhenteellä OHJ ohjelmoinnin ja lyhenteellä DIF differentiaaliyhtälöiden kurssia. Tunnusten perässä oleva L tarkoittaa luentoa ja H harjoitustapahtumaa.*

	ma	ti	ke	to	pe
8-10	DIF L1	TETA L1	OHJ L2		
10-12		OHJ L1		OHJ H2	DIF H1
12-14		LATEX L1		TETA H1	
14-16		LATEX H1		DIF L2	

Liitteessä B.1 esitellyissä, hyppytuntien optimoinnin ensimmäisen ratkaisun lukujärjestyksissä ryhmillä 3 ja 4 on yksi kokonainen vapaapäivä viikon aikana. Muuten oppimistapahtumat ovat jakautuneet kaikilla ryhmillä koko viikon ajalle, kuitenkin niin, että kaikilla ryhmillä on vähintään yksi yksittäisen oppimistapahtuman päivä. Liitteessä B.2 olevien toisen esimerkkiratkaisun lukujärjestyksissä sen sijaan millään ryhmällä ei ole vapaapäiviä viikon aikana. Lisäksi toisen ratkaisun lukujärjestyksissä yksittäisen oppimistapahtuman päiviä on kaikilla ryhmillä yhteensä 12, kun ensimmäisen ratkaisun lukujärjestyksissä niitä on vain 7.

Siirrytään seuraavaksi tapaukseen, jossa hyppytuntien lisäksi minimoidaan myös tilakustannuksia. Tällöin ongelman ratkaisuaika nousee 10 sekunnista jopa 2 tuntiin. Koska päivien kahdella keskimmäisellä tunnilla pidetyistä oppimistapahtumista koituva kustannus on suurempi kuin yhdestä hyppytunnista, saataviin lukujärjestyksiin muodostuu myös hyppytunteja. Kun tilakustannus C_t on 1,2 jokaiselta klo 10–14 järjestetyltä oppimistapahtumalta, ratkaisun lukujärjestyksiin muodostuu viikon ajalle yhteensä 1 hyppytunti ryhmälle 1. Sen sijaan klo 10–14 järjestettäviä oppimistapahtumia on 12. Ratkaisun kokonaiskustannukseksi muodostuu tällöin $(\textit{tapahtumien lkm klo 10–14}) \cdot C_t + (\textit{hyppytuntien lkm}) \cdot C_h = 12 \cdot 1{,}2 + 1 \cdot 1 = 15{,}4$. Saadut lukujärjestykset on esitetty liitteessä B.3.

Kun tilakustannusta C_t kasvatetaan arvoon 2, sekä hyppytuntien että keskellä päivää järjestettyjen oppimistapahtumien määrät lisääntyvät. Saaduista lukujärjestyksistä ryhmän 1 lukujärjestys on esitetty taulukossa 6.3 ja lisäksi kaikkien ryhmien lukujärjestykset liitteessä B.4. Lukujärjestyksiin muodostuu yhteensä 4 hyppytuntia, joihin sisältyvät 2 yksittäistä ja yksi tuplahyppytunti. Lisäksi klo 10–14 järjestettäviä oppimistapahtumia on yhteensä 15. Tällöin saadun ratkaisun kokonaiskustannus on $(\textit{tapahtumien lkm klo 10–14}) \cdot C_t + (\textit{hyppytuntien lkm}) \cdot C_h = 15 \cdot 2 + 4 \cdot 1 = 34$.

Taulukko 6.3 *Ryhmän 1 lukujärjestys tapauksessa, jossa optimoidaan hyppytuntien lisäksi opetustilojen käytöstä aiheutuvia kustannuksia, ja kunkin välillä klo 10–14 järjestetyn oppimistapahtuman kustannus on 2.*

	ma	ti	ke	to	pe
8-10	DIF L1			LATEX H2	DIF H2
10-12	LATEX L1	OHJ L1	OHJ L2		
12-14			OHJ H5	TETA L1	
14-16				DIF L2	TETA H2

Tämän työn esimerkkitapauksessa sekä hyppytuntien että keskellä päivää järjestettävien oppimistapahtumien lukumäärät lisääntyvät kasvatettaessa tilakustannuksia. Lisäksi molemmissa hyppytuntien lukumäärän lisäksi tilakustannuksia minimoivassa tapauksessa lukujärjestyksiin muodostuu useita yksittäisen oppimistapahtuman päiviä. Tilakustannuksen ollessa 1,2 kaikille ryhmille muodostuu yhteensä 9 yksittäisen oppimistapahtuman päivää. Kun tilakustannus on 2, ryhmällä 4 on yksi vapaa päivä ja yksittäisten oppimistapahtumien päiviä muodostuu yhteensä 7.

Tilakustannuksen ollessa joko 1,2 tai 2 on edullisempaa järjestää usean ryhmän oppimistapahtumat keskipäivällä, kuin että kaikilla ryhmillä olisi tällöin hyppytunti. Tilakustannusten ollessa 1,2 on edullisempaa myös järjestää 2 ryhmää sisältäviä oppimistapahtumia keskellä päivää kuin määrätä molemmille ryhmille hyppytunnit. Sen sijaan tilakustannuksen ollessa 2 oppimistapahtumien, joihin osallistuu 2 ryhmää, järjestäminen keskellä päivää aiheuttaa saman kustannuksen kuin kahden ryhmän hyppytunnit. Vaikka keskipäivällä ei järjestettäisikään usean ryhmän oppimistapahtumaa, se ei kuitenkaan tarkoita sitä, että kaikilla ryhmillä olisi silloin hyppytunti. Kuten liitteiden lukujärjestyksistä nähdään, keskellä päivää järjestetään myös harjoitustapahtumia ja luentoja, joihin osallistuu vain yksi ryhmä.

Edellä saadut lukujärjestykset muodostettiin siis ehdolla, jossa käytössä on 1 luento-

sali ja 2 harjoitussalia. Muutetaan rajoitteita siten, että kaikki muut rajoitteet pysyvät ennallaan, mutta käytössä on 2 harjoitussalin sijaan vain 1 harjoitussali. Tällöin MATLAB ei löydä ongelmalle ratkaisua, vaikka se yrittää etsiä sitä 10 tuntia. Kun ongelmalle etsitään ratkaisua MATLABin sijaan optimointiohjelma Gurobilla [4], varmistuu se, että ongelmalle ei ole olemassa käypää ratkaisua kyseisillä rajoitteilla. Toinen vaihtoehto tämän toteamiseen on lisätä MATLABissa ratkaisijan optioihin komento `optimoptions(options,'IntegerPreprocess','advanced')`, jolloin ongelman esikäsittely paranee. Komento ei kuitenkaan toimi MATLAB R2015b –versiossa, joten tällöin myös ohjelmisto tulee päivittää MATLAB R2016a –versioon.

Vaihdettaessa ohjelmiston uudempaan versioon huomataan, että se vaikuttaa saatuihin tuloksiin. Myös esimerkiksi rajoitteiden järjestyksen muuttaminen ja ratkaisuajan tai sallittujen iteraatiokierrosten muuttaminen vaikuttavat saataviin ratkaisuihin. Tässä lukujärjestysongelmassa lokaaleja maksimeja on paljon, joten pienikin muutos jossakin ratkaisun vaiheessa voi muuttaa saatua lopputulosta. Tässä työssä ratkaisujen etsinnässä on käytetty vain tiettyjä ratkaisijan optioita ja MATLABin versiota, joten optioita ja ohjelmistoa muuttamalla on mahdollista saada lukuisia erilaisia tuloksia tähän esimerkkiongelmaan.

7. YHTEENVETO

Tässä työssä käytiin läpi koulujen ja yliopistojen lukujärjestysongelmien perusteita. Ongelman ratkaisemiseksi muodostettiin matemaattinen optimointimalli sisältäen rajoitteet ja kustannusfunktion. Mallin muodostamisessa käytettiin hyödyksi lineaarista kokonaislukuoptimointia ja yleisesti tähän liittyen esiteltiin kokonaislukuongelmien ratkaisemisessa käytettävät branch and bound –algoritmi sekä simplex-algoritmi. Työssä esiteltiin myös kustannusfunktion luomista tapauksessa, jossa minimoitavana on hyppytuntien lukumäärä. Lopuksi muodostettiin pieni esimerkkitapaus, jossa viidelle opiskelijaryhmälle luotiin lukujärjestykset tiettyjä rajoitteita noudattaen. Lukujärjestyksiä muodostettaessa käytettiin myös kahta eri kustannusfunktiota, joista toisella minimoitiin hyppytuntien määrää ja toisella sen lisäksi tilakustannusten suuruutta.

Pelkkää hyppytuntien määrää minimoitaessa ratkaisu esimerkkiongelmalle löytyi melko nopeasti, mutta kun kustannusfunktioon otettiin mukaan myös tilakustannukset, ratkaisun etsimiseen käytetty aika kasvoi huomattavasti. Lisäksi vaikka esimerkkiongelma oli melko pieni, ratkaisun löytyminen ei ollut itsestäänselvää. Tämä vahvistaa osaltaan sitä, että lukujärjestysongelma on tutkimusaiheena haastava.

Muodostuneita lukujärjestyksiä tutkiessa herää kysymys, mitä itseasiassa kannattaa optimoida, jotta saadaan luotua mahdollisimman hyviä lukujärjestyksiä. Vaikka MATLAB löytää rajoitteet toteuttavat ratkaisut sekä hyppytuntien määrää että tilakustannuksia minimoitaessa, huomataan, että jokaisen ryhmän jokaiseen muodostettuun lukujärjestykseen muodostuu melko paljon yksittäisen oppimistapahtuman päiviä. Tietenkin on otettava huomioon se, ettei muodostuneet lukujärjestykset ole yksikäsitteisiä, ja että mikäli esimerkiksi vaihtaa rajoitteiden järjestystä koodissa tai sallittujen iteraatiokierrosten lukumäärää, myös ratkaisu muuttuu. Kuitenkaan työssä saadut esimerkkitapauksen lukujärjestykset eivät ole kovin kompakteja.

Vaikka työssä saadaan muodostettua rajoitteet toteuttavat lukujärjestykset, lopputulokset ovat hieman puutteellisia. Pienissä tapauksissa yksi vaihtoehto on tuottaa

ensin lukujärjestykset MATLABia hyödyntäen, minkä jälkeen käsin muokata lukujärjestyksistä ongelmallisimmat yksityiskohdat pois. Tämän työn esimerkkitapauksissa käsitellään kuitenkin vain tiettyjä rajoitteita ja kohdefunktioita, joten myös rajoitteiden lisäämistä tai kohdefunktion muokkaamista kannattaa kokeilla.

Jatkotutkimuksena ongelmaan voisi lisätä rajoitteita tai muuttaa kustannusfunktiota siten, että saataisiin yksittäisen oppimistapahtuman päivien määrä pienemmäksi. Tämän lisäksi mahdollisuutena olisi myös esimerkiksi pyrkiä jakamaan oppimistapahtumat mahdollisimman tasaisesti viikon ajalle. Ainakin osalle opiskelijoista vapaapäivät ovat mieluisia, joten yksi mahdollisuus olisi pyrkiä tekemään lukujärjestyksistä mahdollisimman kompaktit, jolloin vapaapäiviä syntyisi.

Jatkotutkimuksessa käytettävä MATLAB-koodi kannattaisi rakentaa siten, että perustiedot annetaan koodin alussa, minkä jälkeen rajoitteet muodostuvat automaattisesti muuttujien mukaan. Tällöin esimerkiksi yhden oppimistapahtuman lisääminen ongelmaan ei pakota päivittämään kaikkia rajoitteita käsin. Lisäksi määrittämällä etukäteen mille harjoituskerralle kukin ryhmä osallistuu, ratkaisujen lukumäärää saadaan pienennettyä merkittävästi.

LÄHTEET

[1] S. Abdullah, E. K. Burke, B. McCollum. Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In: K. F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R. F. Hartl, M. Reimann. Metaheuristics, Progress in Complex Systems Optimization. 2007. pp. 153-169.

[2] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, M. A. Awadallah. A Modified Artificial Bee Colony Algorithm for Post-enrolment Course Timetabling. In: Y. Tan, Y. Shi, H. Mo. Advances in Swarm Intelligence. 4th International Conference, ICSI 2013, Harbin, China, June 12-15, 2013. pp. 377-386.

[3] H. Ehtamo. Luento 7: Kokonaislukuoptimointi. Luentomoniste. Systeemianalyysin laboratorio. Aalto-yliopisto. 2007. [viitattu 19.02.2016]. Saatavissa: http://salserver.org.aalto.fi/vanhat_sivut/Opinnot/Mat-2.2105/Luennot/luento7.pdf

[4] Gurobi Optimization. [viitattu 17.05.2016]. Saatavissa: http://www.gurobi.com/index

[5] P. de Haan, R. Landman, G. Post, H. Ruizenaar. A Case Study for Timetabling in a Dutch Secondary School. In: E. K. Burke, H. Rudová. Practice and Theory of Automated Timetabling VI. 6th International Conference, PATAT 2006, Brno, Czech Republic, August 30-September 1, 2006. pp. 267-279.

[6] J. Haataja. Optimointitehtävien ratkaiseminen. CSC-Tieteellinen laskenta Oy. Helsinki. 2004. [viitattu 13.02.2016]. Saatavissa: http://butler.cc.tut.fi/~trantala/opetus/files/LAFY-7204200.Laskennallinen.fysiikka/Haataja-Optimointitehta%CC%88vienRatkaiseminen.pdf

[7] A. J. Izenman. Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning. New York. USA. 2008. pp. 237-280.

[8] S. N. Jat, S. Yang. A Guided Search Genetic Algorithm for the University Course Timetabling Problem. Proceedings of the 4th Multidisciplinary International Scheduling Conference, Theory and Applications (MISTA 2009), August 10-12, 2009, Dublin, Ireland. 2009. pp. 180-191.

[9] A. S-A. Jorge, C. Martín, P. Héctor, A. S. Marco. Comparison of Metaheuristic Algorithmswith a Methodology of Design for the Evaluation of Hard Constraints over the Course Timetabling Problem. In: O. Castillo, P. Melin, J. Kacprzyk. Recent Advances on Hybrid Intelligent Systems. 2013. pp. 289-302.

[10] M. Laaksonen. Matemaattinen analyysi. Luentomoniste. Teknillinen tiedekunta. Vaasan yliopisto. 2003. [viitattu: 09.01.2016]. Saatavissa: `http://lipas.uwasa.fi/~mla/orms1010/majob1p.pdf`

[11] J. Y-T. Leung. Handbook of scheduling: Algorithms, models, and performance analysis. USA. 2004. ch. 45.

[12] D. G. Luenberger, Y. Ye. Linear and Nonlinear Programming. Stanford University. Stanford. USA. 2008.

[13] S. A. MirHassani, F. Habibi. Solution approaches to the course timetabling problem. In: Artificial Intelligence Review, vol. 39, issue 2. 2013. pp.133-149.

[14] C. N. Moschopoulos, C. E. Alexakos, C. Dosi, G. N. Beligiannis, S. D. Likothanassis. A User-Friendly Evolutionary Tool for High-School Timetabling. In: C. Koutsojannis, S. Sirmakessis, Tools and Applications with Artificial Intelligence. 2009. pp. 149-162.

[15] Multidisciplinary International Scheduling Conference: Theory & Applications. [viitattu 03.05.2016]. Saatavissa: `http://www.schedulingconference.org/`

[16] T. Müller. Constraint-based Timetabling. Ph.D. Thesis. Faculty of Mathematics and Physics. Charles University in Prague. 2005. Saatavissa: `http://www.unitime.org/papers/phd05.pdf`

[17] M. M. Mäkelä. Matemaattinen optimointi II. Opintomoniste. Matematiikan ja tilastotieteen laitos. Turun yliopisto. 2015. [viitattu: 13.01.2016]. Saatavissa: `https://www.utu.fi/fi/yksikot/sci/yksikot/mattil/opiskelu/2013-14-kurssit/Documents/Matemaattinen%20optimointi%20II.pdf`

[18] G. L. Nemhauser, L. A. Wolsey. Integer and Combinatorial Optimization. 1988. pp. 355-367.

[19] K. Nurmi, J. Kyngäs. A Framework for School Timetabling Problem. Proceedings of the 3th Multidisciplinary International Scheduling Conference, Theory and Applications (MISTA 2007), August 28-31, 2007, Paris, France. 2007. pp. 386-393.

[20] N. Pillay. A survey of school timetabling research. In: Springer US, Annals of Operations Research, vol. 218, issue 1. 2014. pp. 261-293.

[21] Practice and Theory of Automated Timetabling. [viitattu: 03.05.2016]. Saatavissa: `http://www.patatconference.org/`

[22] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stützle. A comparison of the performance of different metaheuristics on the timetabling problem. In: E. Burke, P. De Causmaecker. Practice and Theory of Automated Timetabling IV. 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002. pp. 329-351.

[23] H. Rudová, K. Murray. University Course Timetabling with Soft Constraints. In: E. Burke, P. De Causmaecker. Practice and Theory of Automated Timetabling IV. 4th International Conference, PATAT 2002, Gent, Belgium, August 21-23, 2002. pp. 310-328.

[24] A. Schaerf. A Survey of Automated Timetabling. In: Kluwer Academic Publishers, Artificial Intelligence Review, An International Science and Engineering Journal, vol.13, issue 2. 1999. pp. 87-127.

[25] R. Silvennoinen. Matemaattinen optimointiteoria 2. Luentomoniste, Luento 1. TTY. 2004. [viitattu: 17.1.2016]. Saatavissa: `http://matriisi.ee.tut.fi/courses/73125/Luento1.pdf`

[26] H. Turabieh, S. Abdullah, B. McCollum, P. McMullan. Fish Swarm Intelligent Algorithm for the Course Timetabling Problem. In: J. Yu, S. Greco, P. Lingras, G. Wang, A. Skowron. Rough Set and Knowledge Technology. 5th International Conference, RSKT 2010, Beijing, China, October 15-17, 2010. pp. 588-595.

[27] Unitime, University Timetabling Comprehensive Academic Scheduling Solutions. [viitattu:13.12.2016]. Saatavissa: `http://www.unitime.org/`

[28] I. X. Tassopoulos, G. N. Beligiannis. Using particle swarm optimization to solve effectively the school timetabling problem. In: Soft Computing, vol. 16, Issue 7. 2012. pp. 1229-1252.

A. ESIMERKKITAPAUKSEN MATLAB-KOODI

```
close all
clear all

ni = 33; % oppimistapahtumien lukumäärä
nj = 20; % tuntien lukumäärä viikossa
nk = 5;  % opiskelijaryhmien lukumäärä

Oppimistapahtumat = {...
'OHJ L1'; 'OHJ L2';...
'OHJ H1'; 'OHJ H2';'OHJ H3';'OHJ H4';'OHJ H5';...
'TETA L1';...
'TETA H1'; 'TETA H2'; 'TETA H3'; 'TETA H4';'TETA H5';...
'DIF L1'; 'DIF L2';...
'DIF H1'; 'DIF H2';...
'LATEX L1';...
'LATEX H1';'LATEX H2';...
'LAFY L1'; 'LAFY L2';...
'LAFY H1'; 'LAFY H2';...
'FYS L1';...
'FYKE L1'; 'FYKE L2';...
'FYKE H1';...
'LAB L1';'LAB L2';
'BIO L1';'BIO L2';
'BIO H1'};

Aikavalit = {'ma 8-10';'ma 10-12';'ma 12-14';'ma 14-16';...
    'ti 8-10';'ti 10-12';'ti 12-14';'ti 14-16';...
    'ke 8-10';'ke 10-12';'ke 12-14';'ke 14-16';...
    'to 8-10';'to 10-12';'to 12-14';'to 14-16';...
    'pe 8-10';'pe 10-12';'pe 12-14';'pe 14-16'};

Ryhmat = {'Ryhma 1';'Ryhma 2';'Ryhma 3';'Ryhma 4';'Ryhma 5'};

% Apumuuttuja, joka on oletusarvoisesti muuttujan x_{ijk}
```

```
% kokoinen nollataulukko:
xa = zeros(ni,nj,nk);

% Muuttujan x_{ijk} koko on:
n = length(xa(:));
% Epäyhtälörajoitteiden alkumäärittely. Näihin lisätään koodissa
% määriteltävät epäyhtälörajoitteet:
A = zeros(0,n);
b = zeros(0,1);
% Yhtälörajoitteiden alkumäärittely. Näihin lisätään koodissa
% määriteltävät yhtälörajoitteet:
Aeq = zeros(0,n);
beq = zeros(0,1);
% Muuttujan x ylä- ja alaraja (muuttuja on binäärinen):
lb = zeros(n,1);
ub = ones(n,1);
% Kertoo alkiot, jotka ovat kokonaislukurajoitteita:
intcon = (1:n)';

%% Muodostetaan rajoitteet

% Oppimistapahtumat pidetään oikealle määrälle ryhmiä vain
% kerran. Rajoitteen (3.11) mukaisesti:
for ii = 1:ni
    % Käsitellään jokainen oppimistapahtuma erikseen. Apumuuttujaan
    % asetetaan kyseistä tapahtumaa vastaaville kaikille tunneille
    % sekä kaikille ryhmille kertoimet 1:
    apu = [zeros(ii-1,nj,nk);ones(1,nj,nk);zeros(ni-ii,nj,nk)];
    % Lisätään yhtälörajoitteisiin tämä rajoite:
    Aeq = [Aeq;(apu(:))'];
    % Jokaisesta oppimistapahtumasta muodostuu yksi rivi
    % yhtälörajoitematriisiin.
end
% Kuhunkin oppimistapahtumaan osallistuvien ryhmien lukumäärät:
beq = ...
    [beq;5;5;ones(5,1);5;ones(5,1);2;2;ones(2,1);2;ones(2,1);2;2;...
    ones(2,1);2;1;1;1;1;1;1;1;1];

% OHJ ja TETA luennot, eli tapahtumat 1,2 ja 8 ovat samaan
% aikaan kaikilla ryhmillä:
for ii = [1:2 8]
    for jj = 1:nj
        apu = xa;
        % Apumuuttujaan lisätään käsiteltävän oppimistapahtuman ja
```

```
78          % tunnin kohdalle kaikille tapahtumiin osallistuville
79          % ryhmille kertoimet rajoitteen (3.12) mukaisesti:
80          apu(ii,jj,:)=[4 -1 -1 -1 -1];
81          Aeq = [Aeq;(apu(:))'];
82          beq = [beq;0]; % Summan vaaditaan olevan 0.
83      end
84  end
85  % DIF ja LATEX luennot, eli tapahtumat 14,15 ja 18 ovat samaan
86  % aikaan ryhmillä 1 ja 2, eli ryhmillä, jotka osallistuvat
87  % kyseisille kursseille:
88  for ii = [14:15 18]
89      for jj = 1:nj
90          apu = xa;
91          apu(ii,jj,:)=[1 -1 0 0 0];
92          Aeq = [Aeq;(apu(:))'];
93          beq = [beq;0];
94      end
95  end
96  % LAFY ja FYS luennot, eli tapahtumat 21,22 ja 25 ovat samaan aikaan
97  % ryhmillä 3 ja 4:
98  for ii = [21:22 25]
99      for jj = 1:nj
100         apu = xa;
101         apu(ii,jj,:)=[0 0 1 -1 0];
102         Aeq = [Aeq;(apu(:))'];
103         beq = [beq;0];
104     end
105 end
106
107 % Määritellään seuraavaksi, mitkä ryhmät osallistuvat millekin
108 % kurssille. Tämä tehdään siten, että vaaditaan, mille kursseille
109 % mitkäkin ryhmät eivät osallistu. Apumuuttujaan laitetaan
110 % kertoimet 1 niiden oppimistapahtumien kohdalle, joihin ryhmät
111 % eivät osallistu.
112
113 % Ryhmilla 3,4 ja 5 ei ole kursseja DIF ja LATEX, eli
114 % tapahtumia 14-20:
115 apu = cat(3, ...
        zeros(ni,nj,2),[zeros(13,nj,3);ones(7,nj,3);zeros(13,nj,3)]);
116 Aeq = [Aeq;(apu(:))'];
117 beq = [beq;0];
118 % Ryhmillä 1-4 ei ole kursseja FYKE, LAB ja BIO, eli
119 % tapahtumia 26-33:
120 apu = cat(3,[zeros(25,nj,4);ones(8,nj,4)],zeros(ni,nj,1));
```

```
121 Aeq = [Aeq;(apu(:))'];
122 beq = [beq;0];
123 % Ryhmillä 1,2 ja 5 ei ole kursseja LAFY ja FYS, eli
124 % tapahtumia 21:25
125 apu = cat(3,[zeros(20,nj,2);ones(5,nj,2);zeros(8,nj,2)],...
126     zeros(ni,nj,2),[zeros(20,nj,1);ones(5,nj,1);zeros(8,nj,1)]);
127 Aeq = [Aeq;(apu(:))'];
128 beq = [beq;0];
129
130 % Seuraavaksi vaaditaan, että kaikilla ryhmillä on oikea määrä
131 % harjoitustapahtumia. Käydään läpi harjoitustapahtumat kaikilta
132 % niiltä kursseilta, joilla on useampi harjoitustapahtuma, mutta
133 % jokaiseen niistä osallistuu vain yksi ryhmä. Apumuuttujaan
134 % lisätään kerroin 1 käsiteltävän ryhmän kaikille tunneille, jotka
135 % vastaavat kyseisiä tapahtumia. Perustana voidaan käyttää
136 % rajoitetta (3.10):
137
138 % Ryhmillä 1-5 on oikea määrä OHJ harjoitustapahtumia:
139 for kk = 1:nk
140     apu = cat(3,zeros(ni,nj,kk-1),[zeros(2,nj); ones(5,nj); ...
            zeros(26,nj)],zeros(ni,nj,nk-kk));
141     Aeq = [Aeq;(apu(:))'];
142     beq = [beq;1];
143 end
144 % Ryhmillä 1 ja 2 on oikea määrä DIF harjoitustapahtumia:
145 for kk = 1:2
146     apu = cat(3,zeros(ni,nj,kk-1),[zeros(15,nj); ones(2,nj); ...
            zeros(16,nj)],zeros(ni,nj,nk-kk));
147     Aeq = [Aeq;(apu(:))'];
148     beq = [beq;1];
149 end
150 % Ryhmillä 1 ja 2 on oikea määrä LATEX harjoitustapahtumia:
151 for kk = 1:2
152     apu = cat(3,zeros(ni,nj,kk-1),[zeros(18,nj); ...
            ones(2,nj);zeros(13,nj)],zeros(ni,nj,nk-kk));
153     Aeq = [Aeq;(apu(:))'];
154     beq = [beq;1];
155 end
156 % Ryhmillä 1-5 on oikea määrä TETA harjoitustapahtumia:
157 for kk = 1:5
158     apu = cat(3,zeros(ni,nj,kk-1),[zeros(8,nj); ones(5,nj); ...
            zeros(20,nj)],zeros(ni,nj,nk-kk));
159     Aeq = [Aeq;(apu(:))'];
160     beq = [beq;1];
```

```
161 end
162 % Ryhmillä 3 ja 4 on oikea määrä LAFY harjoitustapahtumia:
163 for kk = 3:4
164     apu = cat(3,zeros(ni,nj,kk-1),[zeros(22,nj); ...
            ones(2,nj);zeros(9,nj)],zeros(ni,nj,nk-kk));
165     Aeq = [Aeq;(apu(:))'];
166     beq = [beq;1];
167 end
168
169 % Jokaisella ryhmällä on kullakin tunnilla korkeintaan yksi
170 % oppimistapahtuma. Rajoitteen (3.9) mukaisesti muuttujaan lisätään
171 % käsiteltävän ryhmän ja tunnin kohdalle kaikkiin tapahtumiin
172 % kerroin 1:
173 for jj = 1:nj
174 for kk = 1:nk
175     apu = xa;
176     apu(:,jj,kk)=ones(ni,1);
177     A = [A;(apu(:))'];
178     b = [b;1];
179 end
180 end
181
182 % Harjoitustapahtumia on korkeintaan niin monta samaan aikaan,
183 % kuin harjoitussaleja on käytössä. Apumuuttujassa sijoitetaan
184 % käsiteltävän tunnin kohdalle kerroin 1 kaikille harjoitus-
185 % tapahtumille. Käytetään hyödyksi rajoitetta (3.14).
186 hslkm = 2;
187
188 % Jos harjoitussalien lukumääräksi vaihtaa 1, ongelmalla ei ole
189 % ratkaisua.
190 for jj = 1:nj
191     apu = [zeros(ni,jj-1), ...
            [0;0;ones(5,1);0;ones(5,1);0;0;1;1;0;1;1;...
192          0;0;1;1;0;0;0;1;0;0;0;0;1], zeros(ni,nj-jj)];
193     apu2 = cat(3,apu,apu,apu,apu,apu);
194     A = [A;(apu2(:))'];
195     b = [b;hslkm];
196 end
197
198 % Luentoja voi olla korkeintaan niin monta samaan aikaan, kuin
199 % luentosaleja on käytössä. Apumuuttujilla sijoitetaan ryhmille
200 % käsiteltävän tunnin kohdalle kerroin 1 kaikille luennoille, joihin
201 % ryhmä osallistuu. Koska ryhmät 2 ja 4 käyvät samat luennot kuin
202 % ryhmät 1 ja 3, niin niille kertoimia ei laiteta. Myös tässä
```

```
% rajoitteessa hyödynnetään rajoitetta (3.14).
lslkm=1;
for jj = 1:nj
    apu1 = [zeros(ni,jj-1), ...
        [zeros(13,1);1;1;0;0;1;zeros(15,1)],zeros(ni,nj-jj)];
    apu2 = [zeros(ni,nj)];
    apu3 = [zeros(ni,jj-1), ...
        [zeros(20,1);1;1;0;0;1;zeros(8,1)],zeros(ni,nj-jj)];
    apu4 = [zeros(ni,nj)];
    apu5 = [zeros(ni,jj-1), ...
        [zeros(25,1);1;1;0;1;1;1;1;0],zeros(ni,nj-jj)];
    apu6 = cat(3,apu1,apu2,apu3,apu4,apu5);
    A = [A;(apu6(:))'];
    b = [b;lslkm];
end

% Vaaditaan seuraavaksi, että saman kurssin harjoitustapahtumia
% ei voida järjestää useita samaan aikaan, sillä niissä kaikissa
% on sama ohjaaja. Apumuuttujaan asetetaan käsiteltävän tunnin
% kohdalle kerroin 1 kaikkiin niihin tapahtumiin, jotka on
% järjestettävä eri tunneilla. Käytetään pohjana rajoitetta (3.14).

% OHJ harjoitustapahtumat (3-7) eri tunneilla:
for jj = 1:nj
    apu1 = [zeros(ni,jj-1), [0;0;ones(5,1);zeros(26,1)], ...
        zeros(ni,nj-jj)];
    apu2 = cat(3,apu1,apu1,apu1,apu1,apu1);
    A = [A;(apu2(:))'];
    b = [b;1];
end
% TETA harjoitustapahtumat (9-13) eri tunneilla:
for jj = 1:nj
    apu1 = [zeros(ni,jj-1), ...
        [zeros(8,1);ones(5,1);zeros(20,1)],zeros(ni,nj-jj)];
    apu2 = cat(3,apu1,apu1,apu1,apu1,apu1);
    A = [A;(apu2(:))'];
    b = [b;1];
end
% DIF harjoitustapahtumat (16-17) eri tunneilla:
for jj = 1:nj
    apu1 = [zeros(ni,jj-1), ...
        [zeros(15,1);ones(2,1);zeros(16,1)],zeros(ni,nj-jj)];
    apu2 = [zeros(ni,nj)];
    apu3 = cat(3,apu1,apu1,apu2,apu2,apu2);
```

```
241     A = [A;(apu3(:))'];
242     b = [b;1];
243 end
244 % LATEX harjoitustapahtumat (19-20) eri tunneilla:
245 for jj = 1:nj
246     apu1 = [zeros(ni,jj-1), ...
            [zeros(18,1);ones(2,1);zeros(13,1)],zeros(ni,nj-jj)];
247     apu2 = [zeros(ni,nj)];
248     apu3 = cat(3,apu1,apu1,apu2,apu2,apu2);
249     A = [A;(apu3(:))'];
250     b = [b;1];
251 end
252 % LAFY harjoitustapahtumat (23-24) eri tunneilla:
253 for jj = 1:nj
254     apu1 = [zeros(ni,jj-1), ...
            [zeros(22,1);ones(2,1);zeros(9,1)],zeros(ni,nj-jj)];
255     apu2 = [zeros(ni,nj)];
256     apu3 = cat(3,apu2,apu2,apu1,apu1,apu2);
257     A = [A;(apu3(:))'];
258     b = [b;1];
259 end
260
261 % Vaaditaan seuraavaksi, että saman kurssin luennot ovat
262 % eri päivinä. Apumuuttujaan asetetaan kerroin 1 kyseisiä
263 % tapahtumia vastaaville käsiteltävän päivän tunneille,
264 % kaikille osallistuville ryhmille.
265
266 for ll=1:5
267     % OHJ luennot (1-2):
268     apu = xa;
269     apu(1:2,(1:4)+(ll-1)*4,1:nk)=ones(2,4,nk);
270     A = [A;(apu(:))'];
271     % Rajoitteena tapahtumiin osallistuvien ryhmien määrä:
272     b = [b;5];
273     % DIF luennot (14-15):
274     apu = xa;
275     apu(14:15,(1:4)+(ll-1)*4,1:2)=ones(2,4,2);
276     A = [A;(apu(:))'];
277     b = [b;2];
278     % FYKE luennot (26-27):
279     apu = xa;
280     apu(26:27,(1:4)+(ll-1)*4,5)=ones(2,4,1);
281     A = [A;(apu(:))'];
282     b = [b;1];
```

```
283      % LAFY luennot (21-22):
284      apu = xa;
285      apu(21:22,(1:4)+(ll-1)*4,3:4)=ones(2,4,2);
286      A = [A;(apu(:))'];
287      b = [b;2];
288      % LAB luennot (29-30):
289      apu = xa;
290      apu(29:30,(1:4)+(ll-1)*4,5)=ones(2,4,1);
291      A = [A;(apu(:))'];
292      b = [b;1];
293      % BIO luennot (31-32):
294      apu = xa;
295      apu(31:32,(1:4)+(ll-1)*4,5)=ones(2,4,1);
296      A = [A;(apu(:))'];
297      b = [b;1];
298  end
299
300  % Vaaditaan oppimistapahtumat järjestykseen L1,L2,H. Muodostetaan
301  % parit [a b], jotka kertovat, että tapahtuma a järjestetään ennen
302  % tapahtumaa b:
303
304  parit = [1 2;2 3;2 4;2 5;2 6;2 7;8 9;8 10;8 11;8 12;8 13;14 15;...
305      15 16;15 17;18 19;18 20;21 22;22 23;22 24;26 27;27 28;...
306      29 30;31 32;32 33];
307
308  % Apumuuttujiin asetetaan kertoimet rajoitteen (3.15) mukaisesti.
309  for i2 = 1:size(parit,1)
310      for kk=1:nk
311       apu = xa;
312       apu(parit(i2,1),1:nj,kk)=(-nj):-1;
313       apu(parit(i2,2),1:nj,kk)=nj:-1:1;
314           A = [A;(apu(:))'];
315      b = [b;0];
316      end
317  end
318
319  % OHJ tapahtumia (1-7) ei voi olla maanantaisin eikä perjantaisin,
320  % eli tunneilla 1-4 ja 17-20. Käytetään perustana rajoitetta (3.9),
321  % mutta rajoitteena on ryhmien sijaan opettajan vapaanaolo.
322  % Apumuuttujaan asetetaan kaikille ryhmille kerroin 1 kyseisten
323  % tapahtumien, maanantain ja perjantain tuntien kohdalle:
324   apu = ...
         [ones(7,4,nk),zeros(7,nj-8,nk),ones(7,4,nk);zeros(ni-7,nj,nk)];
325   A = [A;(apu(:))'];
```

```
 b = [b;0];

%% HYPPYTUNTIEN TUTKIMINEN

% Tutkitaan ensin yksittäiset, eli yhden oppimistapahtuman pituiset
% hyppytunnit.

% Apumuuttuja:
xa2 = zeros(nj,nk);

% Muuttujan y_{jk} koko:
n2 = length(xa2(:));
% Päivitetään epäyhtälörajoitteet:
A = [A,zeros(size(A,1),n2)];
% Päivitetään yhtälörajoitteet:
Aeq = [Aeq,zeros(size(Aeq,1),n2)];
% Muuttujan ylä- ja alarajat (muuttuja binäärinen):
lb = zeros(n+n2,1);
ub = ones(n+n2,1);
% Kokonaislukurajoitteen sisältävät alkiot:
intcon = (1:(n+n2))';

% Päivän ensimmäistä eikä viimeistä tuntia lasketa hyppytunniksi.
% Apumuuttujaan asetetaan kerroin 1 kaikille päivien ensimmäisille
% ja viimeisille tunneille:
apu2 = xa2;
apu2([1:4:end 4:4:end],:)=1;
Aeq = [Aeq;[zeros(1,n) apu2(:)']];
beq = [beq;0];

% Käydään läpi kaikki mahdolliset hyppytunnit:
for jj=[2:4:20 3:4:20]
    for kk=1:nk

        % Tämä ehto takaa, että muuttujaan y_{jk} tulee hyppytunnin
        % kohdalle 1. Käytetään apuna rajoitetta (3.18)
        apu = xa;
        apu(:,jj-1:jj+1,kk)=[ones(ni,1) -ones(ni,1) ones(ni,1)];
        apu2 = xa2;
        apu2(jj,kk) = -1;
        A = [A;[apu(:)' apu2(:)']];
        b = [b;1];

        % Tämä ehto takaa, että mikäli tunti ei ole hyppytunti,
```

```
370         % muuttujaan tulee arvo 0, (rajoite (3.19)):
371         apu = xa;
372         apu(:,jj-1:jj+1,kk)=[-ones(ni,1) ones(ni,1) -ones(ni,1)];
373         apu2 = xa2;
374         apu2(jj,kk) = 3;
375         A = [A;[apu(:)' apu2(:)']];
376         b = [b;1];
377     end
378 end
379 
380 % Tutkitaan sitten tuplahyppytunnit, eli kahden oppimistapahtuman
381 % pituiset hyppytunnit:
382 
383 % Apumuuttuja:
384 xa3 = zeros(nj,nk);
385 
386 % Muuttujan z_{jk} koko:
387 n3 = length(xa3(:));
388 % Päivitetään epäyhtälörajoitteet:
389 A = [A,zeros(size(A,1),n3)];
390 % Päivitetään yhtälörajoitteet:
391 Aeq = [Aeq,zeros(size(Aeq,1),n3)];
392 % Muuttujan ylä- ja alaraja (muuttuja binäärinen):
393 lb = zeros(n+n2+n3,1);
394 ub = ones(n+n2+n3,1);
395 % Kokonaislukurajoitteen sisältävät alkiot:
396 intcon = (1:(n+n2+n3))';
397 
398 % Päivän ensimmäinen eikä viimeinen tunti voi olla hyppytunti:
399 apu3 = xa3;
400 apu3([1:4:end 4:4:end],:)=1;
401 Aeq = [Aeq;[zeros(1,n+n2) apu3(:)']];
402 beq = [beq;0];
403 
404 % Käydään läpi mahdolliset tuplahyppytunnit. Koska käsitellään
405 % tuplahyppytunteja, käydään läpi ainoastaan jokaisen päivän
406 % 2. tunti, sillä sen käsittely sisältää myös sitä seuraavan
407 % tunnin:
408 for jj=[2:4:20]
409     for kk=1:nk
410 
411         %Tämä ehto takaa, että mikäli on tuplahyppytunti,
412         % muuttujaan z_{jk} tulee hyppytuntien kohdalle 1.
413         % Rajoitteen (3.20) mukaan:
```

```
414          apu = xa;
415          apu(:,jj-1:jj+2,kk)=[2*ones(ni,1) -2*ones(ni,1) ...
                 -2*ones(ni,1) 2*ones(ni,1)];
416          apu2 = xa2;
417          apu3 = xa3;
418          apu3(jj,kk) = -1;
419          apu3(jj+1,kk) = -1;
420          A = [A;[apu(:)' apu2(:)' apu3(:)']];
421          b = [b;2];
422
423          % Tämä ehto takaa, että mikäli tuplahyppytuntia ei ole,
424          % muuttujaan tulee molempien tuntien kohdalle 0
425          % (rajoite (3.21)):
426          apu = xa;
427          apu(:,jj-1:jj+2,kk)=[-2*ones(ni,1) 2*ones(ni,1) ...
                 2*ones(ni,1) -2*ones(ni,1)];
428          apu2 = xa2;
429          apu3 = xa3;
430          apu3(jj,kk) = 8;
431          A = [A;[apu(:)' apu2(:)' apu3(:)']];
432          b = [b;4];
433          apu3 = xa3;
434          apu3(jj+1,kk) = 8;
435          A = [A;[apu(:)' apu2(:)' apu3(:)']];
436          b = [b;4];
437      end
438  end
439
440  %% Ratkaiseminen:
441
442  % Kustannusfunktion kertoimet:
443  C_h=1; % Hyppytunnin kustannus on 1.
444
445  % Seuraavista valitaan tilanteen mukaan:
446      % Seuraavat kaksi muuttujan arvoa valitaan, mikäli
447      % optimoidaan vain hyppytunteja. Saadut lukujärjestykset on
448      % esitetty liitteissä B1 ja B2.
449  C=0;
450  C_t=0;
451
452      % Mikäli optimoidaan hyppytuntien lisäksi myös
453      % tilakustannuksia, muuttujien arvot valitaan seuraavista:
454  %C=1; % Tämä valitaan aina, kun optimoidaan sekä hyppytunteja
455        % että tilakustannuksia. Lisäksi valitaan toinen
```

```
456         % seuraavista:
457 %C_t=1.2; % Valitaan, kun tilakustannus on 1,2. Saadut
458           % lukujärjestykset ovat liitteessä B3.
459 %C_t=2;   % Valitaan, kun tilakustannus on 2. Saadut
460           % lukujärjestykset ovat liitteessä B4.
461
462 % Kustannusfunktio:
463 f = [C*kron(ones(5*nk,1),kron([0;1;1;0],(C_t)./beq(1:ni))); ...
       C_h*ones(n2+n3,1)];
464
465 % Ratkaisijan optiot:
466 options = optimoptions('intlinprog');
467 options = optimoptions(options,'LPMaxIter',5e7)
468 options = optimoptions(options,'MaxTime',7200)
469
470 % Ratkaistaan tehtävä, ratkaisu X saadaan pystyvektorina:
471 X = intlinprog(f,intcon,A,b,Aeq,beq,lb,ub,options);
472 kustannukset=f'*X
473
474 %% Analysoidaan ratkaisua:
475
476 % Järjestetään saatu ratkaisu vektorista 3-ulotteiseksi taulukoksi:
477 x=reshape(X(1:n),ni,nj,nk);
478
479 % Muodostetaan lukujärjestykset:
480 for kk=1:nk
481 lukkari = x(:,:,kk);
482 LUKKARI = cell(4,5);
483 for ii=1:ni
484 LUKKARI(find(lukkari(ii,:)))=Oppimistapahtumat(ii);
485 end
486 disp(['Ryhmän ' num2str(kk) ' lukujärjestys on:'])
487 LUKKARI
488 end
```

B. ESIMERKKITAPAUSTEN TULOKSET

B.1 Hyppytuntien optimointi, esimerkkiratkaisu 1

Taulukko B.1 *Ryhmän 1 lukujärjestys 1, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10				TETA H5	LATEX H1
10-12	LATEX L1		OHJ L2		DIF L2
12-14		OHJ L1	DIF L1		DIF H2
14-16		TETA L1	OHJ H1		

Taulukko B.2 *Ryhmän 2 lukujärjestys 1, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10				OHJ H4	
10-12	LATEX L1		OHJ L2		DIF L2
12-14		OHJ L1	DIF L1		TETA H4
14-16		TETA L1	LATEX H2		DIF H1

Taulukko B.3 *Ryhmän 3 lukujarjestys 1, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10	LAFY L1		FYS L1	LAFY L2	
10-12			OHJ L2	OHJ H2	
12-14		OHJ L1	TETA H1	LAFY H2	
14-16		TETA L1			

Taulukko B.4 *Ryhmän 4 lukujärjestys 1, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10	LAFY L1		FYS L1	LAFY L2	
10-12			OHJ L2	LAFY H1	
12-14		OHJ L1		TETA H2	
14-16		TETA L1		OHJ H3	

Taulukko B.5 *Ryhmän 5 lukujarjestys 1, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10		FYKE L1	TETA H3		BIO L2
10-12		LAB L2	OHJ L2		BIO H1
12-14		OHJ L1	OHJ H5	FYKE L2	
14-16	LAB L1	TETA L1	BIO L1	FYKE H1	

B.2 Hyppytuntien optimointi, esimerkkiratkaisu 2

Taulukko B.6 *Ryhmän 1 lukujärjestys 2, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10	DIF L1	TETA L1	OHJ L2		
10-12		OHJ L1		OHJ H2	DIF H1
12-14		LATEX L1		TETA H1	
14-16		LATEX H1		DIF L2	

Taulukko B.7 *Ryhmän 2 lukujärjestys 2, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10	DIF L1	TETA L1	OHJ L2		
10-12		OHJ L1	LATEX H2		
12-14		LATEX L1	OHJ H1		DIF H2
14-16			TETA H3	DIF L2	

Taulukko B.8 *Ryhmän 3 lukujärjestys 2, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10		TETA L1	OHJ L2		
10-12		OHJ L1	LAFY L2		FYS L1
12-14			TETA H2		
14-16	LAFY L1		OHJ H5	LAFY H2	

Taulukko B.9 *Ryhmän 4 lukujärjestys 2, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10		TETA L1	OHJ L2		
10-12		OHJ L1	LAFY L2	LAFY H1	FYS L1
12-14				OHJ H3	
14-16	LAFY L1			TETA H4	

Taulukko B.10 *Ryhmän 5 lukujärjestys 2, kun minimoidaan hyppytuntien määrää.*

	ma	ti	ke	to	pe
8-10		TETA L1	OHJ L2	FYKE L2	BIO L2
10-12	FYKE L1	OHJ L1	OHJ H4		TETA H5
12-14	LAB L1		BIO L1		BIO H1
14-16			LAB L2		FYKE H1

B.3 Hyppytuntien ja tilakustannusten optimointi kun $C_t = 1{,}2$

Taulukko B.11 *Ryhmän 1 lukujärjestys, kun tilaskustannusen kerroin on 1,2.*

	ma	ti	ke	to	pe
8-10		LATEX H2		DIF L2	
10-12		OHJ L1	OHJ L2		
12-14	LATEX L1		TETA L1	DIF H1	
14-16			DIF L1	OHJ H2	TETA H1

Taulukko B.12 *Ryhmän 2 lukujärjestys, kun tilaskustannusen kerroin on 1,2.*

	ma	ti	ke	to	pe
8-10				DIF L2	DIF H2
10-12		OHJ L1	OHJ L2	OHJ H1	TETA H5
12-14	LATEX L1		TETA L1		
14-16	LATEX H1		DIF L1		

Taulukko B.13 *Ryhmän 3 lukujärjestys, kun tilaskustannusen kerroin on 1,2.*

	ma	ti	ke	to	pe
8-10			LAFY H2	TETA H2	
10-12		OHJ L1	OHJ L2		
12-14		LAFY L2	TETA L1		
14-16	LAFY L1		OHJ H5		FYS L1

Taulukko B.14 *Ryhmän 4 lukujärjestys, kun tilaskustannusen kerroin on 1,2.*

	ma	ti	ke	to	pe
8-10					
10-12		OHJ L1	OHJ L2		
12-14		LAFY L2	TETA L1	OHJ H4	
14-16	LAFY L1	LAFY H1		TETA H3	FYS L1

Taulukko B.15 *Ryhmän 5 lukujärjestys, kun tilaskustannusen kerroin on 1,2.*

	ma	ti	ke	to	pe
8-10	LAB L1	LAB L2	BIO L1	OHJ H3	BIO L2
10-12		OHJ L1	OHJ L2	FYKE L1	BIO H1
12-14			TETA L1		FYKE L2
14-16			TETA H4		FYKE H1

B.4 Hyppytuntien ja tilakustannusten optimointi kun $C_t = 2$

Taulukko B.16 *Ryhmän 1 lukujärjestys, kun tilaskustannusen kerroin on 2.*

	ma	ti	ke	to	pe
8-10	DIF L1			LATEX H2	DIF H2
10-12	LATEX L1	OHJ L1	OHJ L2		
12-14			OHJ H5	TETA L1	
14-16				DIF L2	TETA H2

Taulukko B.17 *Ryhmän 2 lukujärjestys, kun tilaskustannusen kerroin on 2.*

	ma	ti	ke	to	pe
8-10	DIF L1		LATEX H1		
10-12	LATEX L1	OHJ L1	OHJ L2	OHJ H1	
12-14				TETA L1	TETA H5
14-16				DIF L2	DIF H1

Taulukko B.18 *Ryhmän 3 lukujärjestys, kun tilaskustannusen kerroin on 2.*

	ma	ti	ke	to	pe
8-10				OHJ H4	
10-12		OHJ L1	OHJ L2	FYS L1	TETA H1
12-14	LAFY L1	LAFY L2		TETA L1	
14-16		LAFY H2			

Taulukko B.19 *Ryhmän 4 lukujärjestys, kun tilaskustannusen kerroin on 2.*

	ma	ti	ke	to	pe
8-10			LAFY H1		
10-12		OHJ L1	OHJ L2	FYS L1	
12-14	LAFY L1	LAFY L2		TETA L1	
14-16			OHJ H3	TETA H4	

Taulukko B.20 *Ryhmän 5 lukujärjestys, kun tilaskustannusen kerroin on 2.*

	ma	ti	ke	to	pe
8-10		BIO L2	FYKE L1		TETA H3
10-12		OHJ L1	OHJ L2		FYKE L2
12-14		BIO H1	LAB L2	TETA L1	FYKE H1
14-16	BIO L1	LAB L1		OHJ H2	