Hissiryhmän kutsuallokoinnin mallintaminen kokonaislukutehtäväksi

Heikki Uljas
Mark Mehtonen
Teemu Ojalehto

Juha Salomaa

Sisällysluettelo

1. Johdanto

Modernin henkilöhissin tarinan voidaan katsoa alkavan vuodesta 1854, kun Elez Graves Otis onnistui suunnittelemaan maailman ensimmäisen matkustajaturvallisen hissin. Tämä perustui Otiksen keksimään salpatarraimeen (instantaneous safety gear), joka köyden katketessa estää hissin putoamisen kuilun pohjalle.[1] Tämän hissimatkan turvallisuuden takaavan keksinnön myötä sai alkunsa hissien laajamittainen käyttö ihmisten kuljetukseen.

Jo vuosikymmenien ajan kaupunkien ja talojen kasvaessa hissien tarpeellisuus on jatkuvasti lisääntynyt. Esimerkiksi Suomessa on nykyisin lailla määrätty hissin rakentamispakko kaikkiin, vähintään neljä kerrosta korkeisiin uusiin asuinrakennuksiin. Mitä suurempi ja korkeampi rakennus, sitä enemmän hissejä tarvitaan. Isoissa rakennuksissa suurien ihmisvirtojen kuljetuksiin tarvitaan samoille kerrosväleille useampia hissejä. Tällaista useamman hissin kokonaisuutta kutsutaan hissiryhmäksi, kun hissit ovat yhteisessä kutsuohjauksessa.

Hissiryhmän ohjaus on optimointiongelma, jossa tulee ratkaista mikä hissi palvelee kutakin kutsua. Optimoitava eli tässä tapauksessa minimoitava kohdefunktio voi olla esimerkiksi henkilön odotusaika kutsunapin painalluksesta hissin ovien aukeamiseen oikeassa kerroksessa. Kerroskutsujen noutoon hissiryhmän ohjauksessa käytetään nykyisin varsin yleisesti geneettisiin algoritmeihin perustuvia menetelmiä, koska ne tuottavat nopeasti riittävän hyviä ratkaisuja.

Geneettisten ohjausalgoritmien tuottamien ratkaisujen optimaalisuudesta on hankala antaa tarkkaa arviota, sillä algoritmit eivät välttämättä tuota globaalia optimia. Tämän työn motivaationa onkin luoda hissiryhmän hissien kutsuallokaatio-ongelmasta tehtävää kuvaava lineaarinen kokonaislukumalli. Muodostetulle lineaariselle mallille on puolestaan löydettävissä ongelman, valitun optimointikriteerin mielessä ratkaiseva, absoluuttisesti paras ratkaisu eli tehtävän globaali optimi. Tätä kokonaislukumallin tuottamaa parasta mahdollista ratkaisua voidaan sitten käyttää vertailukohtana geneettisen algoritmin luoman ratkaisun optimaalisuuden arvioimiseen.

Ongelman käsittely on tässä raportissa pääpiirteissään seuraavaa. Aluksi olemme määritelleet tehtävän, siihen liittyvät tavoitteet ja tämän projektimuotoisen työn aikataulun. Sen jälkeen esitellään ja käydään läpi hisseihin liittyvää toimintaympäristöä, käsitteitä ja erilaisia hissityyppejä. Tämä kattaa kappaleet kaksi ja kolme. Kappaleessa neljä olemme käyneet läpi lineaarisen kokonaislukutehtävän teoreettista perustaa ja mahdollisia erilaisia vaihtoehtoisia lähestymistapoja kokonaislukuoptimointi ongelmiin.

Kappaleessa viisi siirrymme varsinaisesti ensi kertaa erityisesti tämän projektin ongelmaan, ja teorian soveltamiseen tehtävän erikoistapaukseen. Tässä käsittelemme ongelman matemaattisen mallin kehittämisen eri vaiheet, malliversiot ja selostamme mallien rakenteen ja komponentit yksityiskohtaisesti. Kappaleessa kuusi selostamme mallin mukaisen ongelman ratkaisemisen vaiheet ja esittelemme ratkaisemisessa tarvittavat työkalut ja metodit. Näitä ovat mm. itse mallin ratkaisemisessa tarvittava kokonaislukuoptimointi ongelman ratkaisuohjelmisto, ja itse ohjelmoimamme ”generaattori”. Generaattori on ohjelma, joka muokkaa matemaattisen mallin sääntelemän ongelman ratkaisijan ymmärtämään muotoon. Seuraavaksi, kappaleessa seitsemän, käsittelemme mallin antamat tulokset. Tässä arvioimme valitun mallin hyvyyttä, tulosten optimaalisuutta, ja teemme hieman vertailua käytössä oleviin muihin algoritmeihin.

---

[1] Otis Oy. Hissisanasto. http://www.otis.com/glossary/0,1374,CLI7_RES1,00.html

Tämä projektityö on tehty Teknillisen Korkeakoulun kurssille Mat-2.177 Operaatiotutkimuksen projektityöseminaari. Olennainen osa kurssia, ja kaikkia huolellisesti tehtyjä projekteja, on loppuarvio. Viimeisessä kahdessa kappaleessa on tämän projektin analysointi. Kappaleessa kahdeksan arvioimme projektin onnistuneisuutta usean eri kriteerin kannalta, näitä ovat mm. aikataulu, riskienhallinta, tuloksellisuus, ja ryhmätyö. Lisäksi käymme läpi, mitä projektissa olisi voitu tehdä paremmin ja miten ongelmaa olisi tulevaisuudessa järkevää lähestyä. Viimeisenä on yhteenveto.

Raportissa on varsin ekstensiiviset liitteet. Liitesivujen suuri lukumäärä johtuu osittain siitä että tehtävän kokonaislukuongelma muodostuu rajoitteineen pienessäkin tehtävässä jo hyvin suureksi. Liitteinä on mm. esimerkkiongelman muotoilu mallin mukaan, generaattoriohjelman lähdekoodia osin, ja mallin antamia tuloksia. Niitä on hyvä tutkia, jotta ymmärtää raportin kuvaamat ongelman osat.

Kokonaislukumuuttujia sisältävä optimointiongelma on mielenkiintoinen ja haastava. Kombinatorisen luonteensa vuoksi kokonaislukuongelman ratkaiseminen eroaa oleellisesti jatkuvista ongelmista, eikä ratkaiseminen ole yhtä korkealle kehittynyttä kuin jatkuvien ongelmien ratkaiseminen. Useat käytännön ongelmat sisältävät kuitenkin juuri diskreettejä ominaisuuksia, joita ei useinkaan voi aivan tarkasti mallintaa jatkuvin muuttujin. Lisäksi jatkuvan mallin antaman ratkaisun ”pyöristäminen” ei tarjoa ratkaisua – jatkuvan mallin kokonaisluvuiksi pyöristetty ratkaisu, ei yleisesti ole sama kuin kokonaislukumallin optimi. Haastetta ongelmassamme siis riittää. Myös yleisesti voidaan todeta että kokonaislukuoptimoinnissa riittää pähkinöitä rikottaviksi, paitsi käytännön ongelmissa, niin myös jopa teoreettiseen menetelmäkehitykseen asti. Tässä työssä tavoitteena on kuitenkin soveltaa olemassa olevia metodeja uuteen käytännön ongelmaan, ei kehittää uutta teoriaa. Toivottavasti kiinnostuksesi aiheeseen on herännyt, sillä ongelmat tällä pellolla ovat usein yllättävän lähellä jokapäiväistä ihmiselämää.

2. Tehtävän määrittely

2.1 Tavoite

Kone Oyj:n antama lähtökohtainen tehtävä oli muodostaa optimoitava kokonaislukumalli hissiryhmän ohjauksesta kerroskutsuihin. Ongelma jaettiin ryhmän ja ohjaajan toimesta useaksi konkreettisemmaksi eri tavoitteeksi. Ensimmäisenä tavoitteena on matemaattisesti mallintaa hissiryhmän ohjaus kerroskutsujen palvelussa. Mallinnus tulee toteuttaa lineaarisena kokonaislukumallina. Mallinnukseen sisältyy tehtävän optimointikriteerin valitseminen.

Tehtävän toisena tavoitteena on ratkaista lineaarisen kokonaislukumallin mukainen hissiryhmän optimaalinen ohjaus, ryhmän valitseman optimointikriteerin kannalta. Valitun kriteerin mielessä optimaalinen ohjaus tulisi löytää kaikissa mahdollisissa hissiryhmän kerroskutsu -tilanteissa. Ohjauksen tulisi löytää tehtävän globaali optimi.

Kolmantena tavoitteena on verrata saatua lineaarisen kokonaislukumallin antamaa ratkaisua geneettisten algoritmien samassa tilanteessa antamiin ratkaisuihin. Näin voidaan analysoida geneettisten algoritmien antamien ratkaisuiden hyvyyttä ja niiden optimaalisuutta.

2.2 Aikataulu

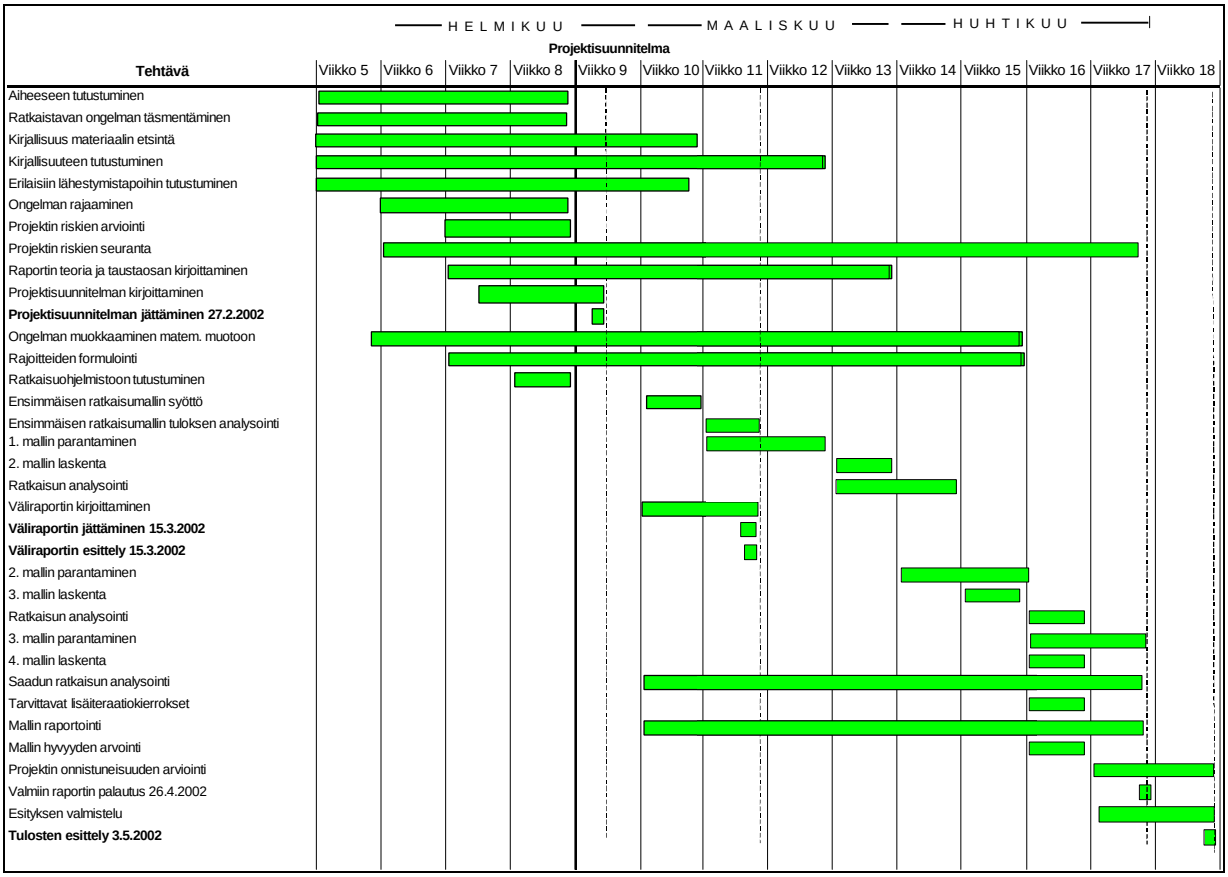
Tehtävästä koottiin heti ensimmäisellä viikolla yksityiskohtainen, kaikkiaan 35 kohdaksi, eritelty viikoittainen tehtäväkohtainen projektiaikataulu. Projektin aikajänne kesti kaikkiaan 13 viikkoa, jonka jälkeen tehtävän tuli olla valmis ja raportin kirjoitettu. Valmiista raportista pidettiin kurssille esitelmä palautusta seuraavalla viikolla.

Aikataulun suunnittelussa pyrimme projektin etupainotteisuuteen, ja siihen että aikataulussa olisi varattuna riittävästi aikaa iteraatiokierroksille mallin kehittämisessä. Annoimme mallin jalostamiselle paljon painoa myös siksi että kukaan ryhmän jäsenistä ei varsinaisesti aiemmin ollut mukana tämän suuruisten optimointiongelmien muodostamisessa ja ratkaisemisessa. Pyrimme aikataulussa myös siihen että tehtävien erittely olisi riittävän tarkka. Tämä siksi jotta projektin mahdollisimman useat osa-alueet tulisivat heti alkuvaiheessa esille konkreettisina tehtävinä, jotka projektin kuluessa tulee ajallaan huomioida. Aikataulu tehtävineen jakautuu tavallaan kolmeen pääosioon: esiselvitys- ja teoriaosaan, ongelman muodostamiseen matemaattiseen muotoon, ja mallin mukaan muodostetun ongelman ratkaisemiseen.

Kurssi ja projekti alkoi 18.1.2002, jolloin muodostettiin ryhmät ja jaettiin tehtävät. Projektin kolme tärkeää päivämäärää olivat 27.2.2002 jolloin oli jätettävä projektisuunnitelma, 15.3.2002 projektin väliraportin jättäminen ja esittely, sekä 26.4.2002 jolloin valmis projektiraportti oli jätettävä. Lisäksi valmiista projektista pidettiin lyhyt esitelmä kurssilla 3.5.2002.

Ongelman teoriaan ja hissialaan perehtyminen ajoitettiin tehtäväksi ennen projektisuunnitelman jättämistä. Tässä onnistuttiin varsin hyvin sillä ahkerasta hakemisesta huolimatta hissiryhmän ohjaamiseen emme löytäneet yhtään aiemmin tehtyä kokonaislukumallia. Yksittäiselle hissille tehtyjä kokonaislukumalleja sen sijaan löysimme kyllä. Ne eivät kuitenkaan tässä tapauksessa juuri auta, vaan ongelma oli suunniteltava kokonaan uusiksi alusta lähtien. Tässä ryhmälle projektiohjaajan asiantuntemuksesta oli suuri apu. Kokonaisluku optimoinnin yleinen teoria oli

projektiryhmälle periaatteessa tuttua korkeakoululla käytyjen lineaarista optimointia käsittelevien kurssien perusteella (mm. Mat-2.140 Lineaarinen ohjelmointi).



Kuva 1: Projektiaikataulu

Toisessa vaiheessa tapahtui pääosin ongelman muodostaminen matemaattiseen muotoon. Tässä ensiarvoisen olennaista oli mallin rajoitteiden muodostaminen ja kaikkien tarvittavien rajoitteiden tunnistaminen. Tässä osuudessa etsimme myös mallin ratkaisussa tarvittavia työkaluja. Toiseen vaiheeseen laskimme kuuluvaksi myös mallin ensimmäisen parannuskierroksen. Toisen vaiheen ajoitimme päättyväksi väliraportin esittelyyn. Tämä siksi että väliraportin esittelyssä ryhmällä olisi jo ensimmäisiä konkreettisia kokemuksia mallista ja sen tekemisestä, sekä ensimmäisiä mallin antamia tuloksia.

Projektin kolmas pääosuus kesti väliraportoinnista projektin loppumiseen. Tähän aikajaksoon kuuluivat mallin parantamisen seuraavat iteraatiokierrokset ja mallin optimointitulosten laskeminen. Tässä vaiheessa arvioitiin myös mallin hyvyyttä ja sen antamia tuloksia. Raportin kirjoittamisesta syntyi leijonanosa tässä viimeisessä kolmanneksessa.

3. Toimintaympäristö ja käsitteitä

Aloitetaan ja määritetään peruskysymys: mikä on hissi? Jokaisella meistä on varmaan omakohtaisia kokemuksia hisseistä, ja käsitys siitä mikä hissi on. Itse kuvaillen se voisi olla vaikkapa rakennuksessa ihmisten ja tavaroiden kerrosten väliseen siirtelyyn tarkoitettu laite. Etsivä löytää ja vastaus löytyy tietenkin, mistäpä muualtakaan, kuin Euroopan Unionin yhteisistä direktiiveistä, jossa hissikin on määritelty. Unionin kapean soveltamisalan mukaan määriteltynä, hissi on seuraavaa:

"Hissillä tarkoitetaan kiinteästi asennettua laitetta, joka liikkuu eri tasojen välillä ja on suunniteltu kuljettamaan henkilöitä, tavaroita tai molempia."

"Soveltamisen ulkopuolelle jäävät köysiradat, sotilas- ja poliisikäyttöön suunnitellut hissit, kaivoskuilujen nostolaitteet, näyttämönostimet, kuljetusvälineissä käytetyt hissit, hammasjunat, rakennushissit ja koneen osana olevat hissit, jotka on tarkoitettu yksinomaan työskentelypaikalle kulkemiseen."

Edellä mainittu on Hissidirektiivi 95/16/EY, ja se sisältää hissien suunnittelua ja rakentamista koskevat turvallisuusohjeet, joita Unionissa kutsutaan standardeiksi.[2] Hissidirektiivi sisältää 19 standardia, niistä viisi on päätetty. Hissidirektiivistä on Suomessa tehty kolme yhdenmukaistettua SFS standardia: standardi sähkökäyttöisille hisseille, standardi hydraulikäyttöisille hisseille ja standardi hissien sähkömagneettisesta yhteensopivuudesta.[3]

Projektimme "toiminta-alue" koskee juuri samoja hissejä joita direktiivissäkin tarkoitetaan. Tosin hissiryhmän ohjausta optimoiva kokonaislukumalli voi sinänsä soveltua muillekin samanlaisia optimointiongelmia kohtaaville laitteille, mutta muissa laitteissa ei juuri taida optimoitavia hissiryhmiä esiintyä.

3.1 Erilaisia hissityyppejä

Edellisen rajauksen mukaan määriteltyjäkin hissejä on useita eri tyyppisiä. Hissihän on periaatteessa uniikkituote sillä hissiä lähes aina modifioidaan rakennuksen erityistarpeisiin sopivaksi. Seuraavassa esitellään erilaisia hissityyppejä.

Yleisintä hissityyppiä, tavallista henkilöhissiä, on useita erilaisia. Hissin teknisellä puolella hissikoneisto voi olla joko köysihissi tai hydraulihissi. Korkeat hissit ovat aina köysihissejä. Koneistoista on kehitetty nk. konehuoneettomia hissimalleja, joissa hissin tekniikka on sijoitettu hissikuiluun, eikä erillistä konehuoneetta tarvita. Tavallisia henkilöhissejä on useita eri korikokoja, aina kahdesta kymmenien ihmisten kapasiteettiin asti. Tavallinen hissi voi myös olla ns. kaksikorihissi jolloin hissikoreja on kaksi päällekkäin. Kaksikorihissejä on yleensä vain hyvin suurissa rakennuksissa. Myös kolmikorisia hissejä on mietitty.[4] Ratkaisun etuna hissikuilun tehokkaampi tilankäyttö. Tavallisten henkilöhissien tyyppejä ovat lisäksi pikahissi, potilashissi, maisemahissi ja

---

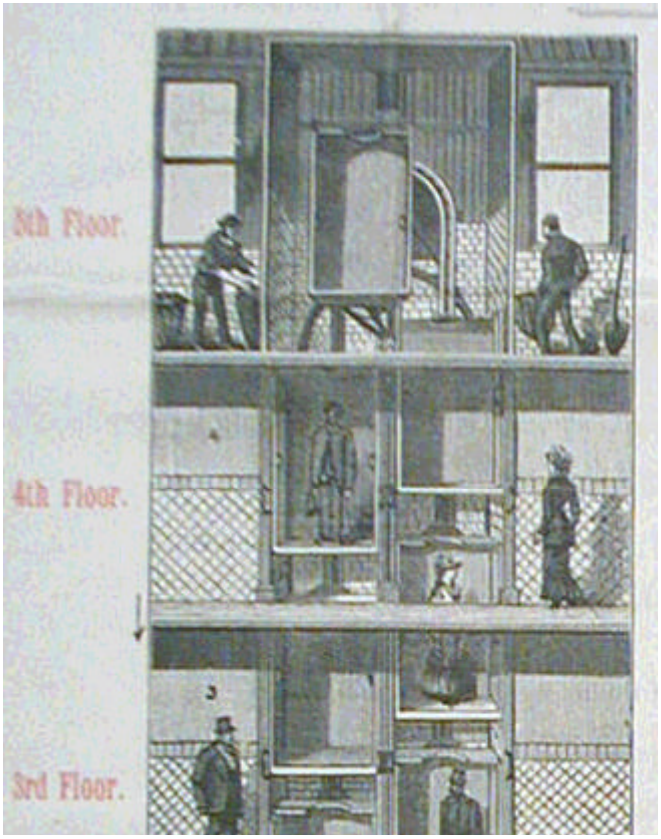
[2] Suomen Standardoimisliitto SFS ry. Avain Euroopan markkinoille - EU:n uudet tekniset määräykset ja standardit. http://www.sfs.fi/standard/avain.html

[3] Suomen Standardoimisliitto SFS ry. Palvelut, julkaisut: Hissidirektiivi. http://www.sfs.fi/standard/hissi.html

[4] Siikonen, Marja-Liisa. Liikennesuunnittelupäällikkö, Kone Oyj. Tutustumiskäynti 19.4.2002, Helsinki.

tavarahenkilöhissi.[5] Yhteiskutsussa olevia hissiryhmiä muodostetaan juuri tavallisista henkilöhisseistä.

Paternoster –hissi toimii täysin eri tekniikalla kuin tavallinen henkilöhissi. Paternoster hissi toimii siten että ketjussa kiinni olevat hissikorit kiertävät kahdessa hissikuilussa ympyrää. Hissikori tulee toista kuilua alas, kulkee alhaalla hetken "tasamaalla" vaakasuoraan, siirtyessään nousevan hissikuilun kohdalle, ja lähtee sitten nousemaan ylöspäin toisessa hissikuilussa. Ylätasanteella kori jälleen siirtyy hetken vaakasuoraan, laskevan kuilun kohdalle ja lähtee laskeutumaan alas. Hissin kyydissä voi olla myös käännöksen ajan ja riippuu rakennuksesta mitä käännöksen aikana näkee.



Kuva 2: Paternoster hissi

Hissi toimii jatkuvasti vakionopeudella, joten hissin palvelun odotusaika on hyvin pieni. Hissin kulkunopeus on kuitenkin alhainen, joten se ei sovellu korkean rakennuksen ainoaksi hissiksi. Paternoster hissi on äärimmäisen taloudellinen tapa kuljettaa ihmisiä sillä toinen puoli on aina tehokas vastapaino, ja se on myös varsin tehokas ruuhka-aikoina. Haittana on suuri tilantarve, mutta toisaalta vaadittava tila on tehokkaassa käytössä. Hissityyppi voi myös olla vaarallinenkin, kiilautumisvaara on ilmeinen jos hissin yrittää viime hetkellä tai jää osin korin ulkopuolelle[6]. Todennäköisesti suurimmaksi osaksi potentiaalisen vaarallisuutensa takia hissityyppi onkin jo enemmänkin historiaa ja uusia ei todennäköisesti juurikaan enää rakenneta. Suomessa uusia paternoster-hissejä ei saa enää ottaa käyttöön, mutta entisiä voi edelleen ylläpitää. Näillä hisseillä ei esiinny samanlaisia hissinohjauksen optimointiongelmia kuin tavanomaisilla kuiluhisseillä.

---

[5] Otis Oy. Hissituotteet. http://www.otis.com/products/listing/0,1357,CLI7_PRT100_RES1,00.html

[6] Helsingin Sanomat. Uutisarkisto, Torsti tietää 21.2.1999. http://www.helsinginsanomat.fi/uutisarkisto/19990222/erik/990221er10.html

Minihissi, kaitahissi. Nämä hissit ovat vanhoihin rakennuksiin tarkoitettuja jälkiasennus hissejä, jotka vievät hyvin vähän tilaa. Ne ovat usein hydraulikäyttöisiä ja sopivat ahtaisiinkin tiloihin. Asennukseen tarvittava kuilun minimileveys on vain metri. Näitä hissityyppejä ei asenneta hissiryhminä.

3.1.1 Käsitteitä

Jotta lukijalle kehkeytyisi parempi käsitys käytettävistä hisseihin liittyvistä termeistä, käydään ne läpi tässä osiossa.

Ajoaika:	Aika kahden pysähdyksen välillä
Hissiryhmä:	Useasta hissistä muodostettu ryhmä, jonka tarkoitus on palvella itsenäisiä hissejä tehokkaammin. Ryhmää karakterisoi yhteinen ohjaus.
Kerroskutsu:	Tietystä kerroksesta annettu kutsu.
Kiertoaika:	Aika, joka yhdeltä hissiltä kuluu sen lähdettyä ylöspäin aina takaisin aulaan palaamiseen.[7]
Korikutsu:	Hissin sisältä annettu kutsu, eli käytännössä ihmisten napin painamiset hississä.
Kuljetuskapasiteetti:	Matkustajamäärä, jonka hissiryhmä pystyy kuljettamaan sisääntuloruuhkan vallitessa viiden minuutin ajan.
Kutsuaika:	Aika kutsunapin painamisesta oikeansuuntaisen hissin kerrokseen saapumiseen.
Matkustusaika:	Hissijärjestelmässä vietetty kokonaisaika.[8]
Odotusaika:	Aika henkilön kerrokseen saapumisesta hissiin astumiseen
Palveluaika:	Kiertoaika jaettuna hissien lukumäärällä.[9]
Pysähdysaika:	Hissin kiihdytykseen ja hidastamiseen kulunut aika lisättynä oviaikoihin.
Ryhmäohjaus:	Jakaa kerroskutsut hisseille ja antaa muita palvelun parantamiseen tähtääviä ajokomentoja.[10]

[7] Rintala, Toni. Matkustajien kohdekerrosten ennustaminen Bayes-menetelmällä liikennetilastoja käyttäen. Pro gradu -tutkielma, Helsingin Yliopisto, Tietojenkäsittelytieteen laitos 2000.

[8] Rintala, Toni. Matkustajien kohdekerrosten ennustaminen Bayes-menetelmällä liikennetilastoja käyttäen. Sivu 3. Pro gradu -tutkielma, Helsingin Yliopisto, Tietojenkäsittelytieteen laitos 2000.

[9] Siikonen, Marja-Liisa. Hissiliikenteen ja ohjauksen mallintaminen tietokoneella. Sivu 7. Lisensiaattityö, Teknillinen Korkeakoulu, Tietotekniikan osasto, Matematiikan ja systeemianalyysin laitos 1989.

[10] Rintala, Toni. Matkustajien kohdekerrosten ennustaminen Bayes-menetelmällä liikennetilastoja käyttäen. Sivu 1. Pro gradu -tutkielma, Helsingin Yliopisto, Tietojenkäsittelytieteen laitos 2000.

4. Kutsuallokoinnin optimointi

4.1 Kokonaislukutehtävä, mahdollisia eri lähestymistapoja

Kokonaislukuoptimoinnissa optimointitehtävän muuttujat saavat kokonaislukuarvoja. Osa muuttujista voi myös olla jatkuvia ja osa kokonaislukumuuttujia, silloin optimointitehtävästä käytetään nimitystä sekalukuoptimointi. Kokonaislukuoptimointi eroaa normaalista optimointitehtävästä ainoastaan muuttujiensa suhteen.

Kokonaislukutehtävät ovat yleensä paljon työläämpiä ratkoa kuin vastaavat optimointitehtävät jatkuvilla muuttujilla. Tämä johtuu kokonaislukutehtävien kombinatorisesta luonteesta ja epäjatkuvuudesta. Kokonaislukutehtäville ei ole löydetty yleispätevää ratkaisualgoritmia, joka ratkaisisi tehokkaasti kokonaislukutehtäviä. Yleensä kokonaislukutehtävän ratkaisuaika on pidempi kuin samankokoisen tehtävän, jossa käytetään jatkuvia muuttujia.

Useat kokonaislukutehtävien ratkaisualgoritmit ratkaisevat ensiksi tehtävän jatkuvilla muuttujilla, jonka jälkeen ratkaisusta generoidaan käypiä kokonaisluku ratkaisuja, joiden optimaalisuutta tarkastellaan. Kokonaislukutehtävien ratkaisualgoritmit voidaan jakaa kahteen pääluokkaan: leikkaustasomenetelmiin (cutting plane methods) ja luettelointimenetelmiin (enumerative methods).[11]

Leikkaustasometodien ideana on johtaa uusia epäyhtälörajoituksia kokonaislukurajoituksista. Epäyhtälöt rajaavat käyvän joukon yhä pienemmälle alueelle ja lopulta löydetään optimaalinen kokonaislukuratkaisu.

Luettelointimenetelmissä lähdetään ratkaisua hakemaan jatkuvan tehtävän ratkaisusta, joista johdetaan sitten käypiä kokonaislukuratkaisuja. Sen jälkeen pidetään kirjaa ratkaisukandidaateista ja eliminoidaan ratkaisuja, jotka eivät voi olla optimiratkaisuja. *Branch and Bound*- algoritmi on tunnetuin ja käytetyin luettelointimenetelmä, joka on myös varsin tehokas.

Kokonaislukutehtävät ovat muuttujiensa puolesta kombinatorisia, niiden ratkaisu löytyy äärellisestä tai numeroituvasti äärellisestä ratkaisujoukosta. Koska muuttujat eivät ole jatkuvia ei niihin voida käyttää perinteisiä optimointimenetelmiä. Käytetyt menetelmät perustuvat siihen, että ensin ratkaisu etsitään reaaliarvoisille muuttujille ja seuraavaksi etsitään tätä ratkaisua lähellä oleva kokonaisluku ratkaisu. Tässä toisessa vaiheessa joudutaan kuitenkin vertailemaan eri vaihtoehtoja toisiinsa. Kombinatoriset ongelmat ovatkin NP – täydellisiä eli niille ei ole olemassa algoritmia, jonka suoritusaika kasvaisi polynomisesti tehtävän koon kasvaessa.

4.1.1 Kauppamatkustajan ongelma

Kauppamatkustajan ongelma (Traveling salesman problem, TSP) on klassinen esimerkki ongelmasta, jolle ei kombinatorisen luonteensa vuoksi ole olemassa käytännöllistä ratkaisumenetelmää. Kauppamatkustajan tehtävänä on kiertää n+1 kaupungin reitti ja palata takaisin lähtöpisteeseensä. Kussakin kaupungissa kauppamatkustajan tulee käydä täsmälleen kerran. Kaupunkien väliset etäisyydet oletetaan tiedetyiksi.

[11] Taha, H. A. (1996), Operations Research - An Introduction, Prentice Hall. (6. painos).

Kun ongelma formuloidaan kokonaislukutehtäväksi saadaan päätösmuuttujiksi binäärimuuttujat x_{ij}, joiden arvo kertoo kulkeeko kauppamatkustajan reitti kaupungista i kaupunkiin j. Lisäksi tarvitaan rajoitteita jotka takaavat että kuljettu reitti on yhtenäinen, ja että kussakin kaupungissa käydään täsmälleen kerran. Yleisessä tapauksessa erilaisia reittejä on (n+1)! kappaletta. Kauppamatkustajan ongelmalle ei ole olemassa ratkaisualgoritmia joka ratkaisisi sen polynomisessa ajassa.

Hissiryhmän kutsunallokointi voidaan myös formuloida kauppamatkustajan ongelmasta. Tällöin päätösmuuttujat x_{ijk} kertoo että hissi k kulkee kutsusta i kutsuun j. Lisäksi tarvitaan ehdot jotka takaavat että hissien kulkemat reitit ovat yhtenäisiä ja, että kukin kutsu palvellaan täsmälleen yhdellä hissillä täsmälleen kerran.

4.1.2 PDP

Asiakkaiden noutamisen ja toimittamisen ongelmassa (Pickup and delivery problem, PDP) kulkuneuvon on täytettävä joukko toimituspyyntöjä, joissa toimitus noudetaan jostain pisteestä ja toimitetaan toiseen pisteeseen.[12] Kauppamatkustajan ongelmaan verrattuna PDP:ssä on siis rajoite, joka sanoo että asiakkaan noutopisteen on oltava ennen toimituspistettä.

[12] Sorsa, Janne. Kaksikoristen hissien optimaalinen ryhmäohjaus. Diplomityö (valmisteilla), Teknillinen Korkeakoulu 2002.

5. Malli

Valitsimme malliimme kohdefunktioksi eli optimoitavaksi tekijäksi kaikkien kutsujen odotusaikojen yhteisen summan minimoinnin. Tämä valita tehtiin siksi koska se oli mielestämme järkevin tekijä hissien optimaalisen kulun mallintamiseen, kun pyrkimyksenä on minimoida henkilön odotusaikaa kerroksessa kutsunapin painamisesta hissin saapumiseen ja hissin ovien aukeamiseen.

Tässä vaiheessa rajoitimme, ryhmän ja ohjaajan yhteisellä päätöksellä, tehtävän ainoastaan kutsuaikojen summan minimointiin. Kehittämässämme mallissa ei siis lainkaan huomioida kutsun antaman henkilön haluamaa lopullista päämääräkerrosta, eli kerrosta jonka henkilö antaa astuttuaan hissiin ns. korikutsuna. Mallissa hissit minimoivat siis vain kerroskutsujen noutoon liittyvän ajan summaa, huomioimatta minne ihmiset mahdollisesti haluaisivat matkustaa. Tässä suhteessa, malli tässä muodossaan, on jossain määrin epärealistinen todelliseen käyttötilanteeseen verrattuna.

5.1 Ensimmäinen versio

Mallin ensimmäisessä versiossa kohdefunktiona oli kutsujen ajoaika ja muuttujana y_{ik} joka kertoo että hissi k palvelee kutsun i. Mallista yleensä voidaan sanoa, että se on kovin monimutkainen eikä kovinkaan selkeä.

$$\min_{x^0,x,y} \sum_{k\in E}\sum_{j\in C} w_j \left(\sum_{i\in C} x^0_{ik} t^0_{ik} + \sum_{i\in C \wedge i\neq j} d_{ijk} y_{ik} \left(t_i + \sum_{m\in C \wedge m\neq i} x_{mik} t_{mik}\right)\right) \qquad \textbf{5.1}$$

Kaavassa 5.1 on esitetty kohde funktio. Tässä w_i on kutsun painokerroin jolla on tarkoitus priorisoida korikutsut, d_{ijk} kertoo palveleeko hissi k kutsun i kutsua j ennen, x0ik kertoo palveleeko hissi k ensimmäiseksi kutsun i, xijk kertoo palveleeko hissi k kutsun i jälkeen seuraavaksi kutsun j, ti on kutsun i palveluun kuluva aika ja tijk ajoaika kutsusta i kutsuun j hissillä k.

$$x_{ijk} \Leftrightarrow y_{ik} \wedge y_{jk} \wedge d_{ijk} \wedge \forall m \in C(y_{mk} \Rightarrow d_{mik} \vee d_{jmk}), \forall i \in C \forall j \in C \forall k \in E \qquad \textbf{5.2}$$

$$x^0_{ik} \Leftrightarrow y_{ik} \wedge \forall m \in C(y_{mk} \Rightarrow d_{imk}), \forall i \in C \forall k \in E \qquad \textbf{5.3}$$

$$\forall i \in C \exists k \in E y_{ik} \qquad \textbf{5.4}$$

Kaavoissa 5.2-5.4 on esitetty rajoitteet joissa kahdessa ensimmäisessä määritellään x muuttujat ja viimeisessä määrätään että kaikki kutsut palvellaan.

$\forall i \in C \forall j \in C \forall k \in E$:

$$x_{ijk} \leq y_{ik} y_{jk} d_{ijk} (1 - y_{mk} + d_{mik} + d_{jmk}), \forall m \in C \qquad \textbf{5.5}$$

$$x_{ijk} \geq y_{ik} y_{jk} d_{ijk} - \sum_{m\in C} y_{mk} (1 - d_{mik})(1 - d_{jmk}) \qquad \textbf{5.6}$$

$$x_{ijk} \in \{0,1\} \qquad \textbf{5.7}$$

$\forall i \in C \forall k \in E$:

$$x^0_{ik} \leq y_{ik} (1 - y_{mk} + d_{imk}), \forall m \in C \qquad \textbf{5.8}$$

$$x^0_{ik} \geq y_{ik} - \sum_{m\in C} y_{mk} (1 - d_{imk}) \qquad \textbf{5.9}$$

$$x_{ik}^{0} \in \{0,1\} \quad \textbf{5.10}$$

$$\sum_{k \in E} y_{ik} = 1, \forall i \in C \quad \textbf{5.11}$$

$$y_{ik} \in \{0,1\}, \forall i \in C \forall k \in E \quad \textbf{5.12}$$

Rajoitteiden 5.5-5.12 kaikkien merkityksestä ei ole selvyyttä. Rajoitteet 5.7, 5.10 ja 5.12 kertovat binäärimuuttujat. Rajoite 5.11 sanoo jokaista kutsua palveltavan vain yhdellä hissillä.

Tässä mallissa on muutamia puutteita, kuten esimerkiksi se että rajoitteiden 5.5 ja 5.6 takia se ei ole lineaarinen. Vakavin puute on kuitenkin se että kohdefunktio ja rajoitteet ovat kovin monimutkaisia. Tämän voidaan katsoa johtuvan huonosta muuttujan valinnasta. Muuttujaksi on valittu mikä hissi palvelee kutsun vaikka paljon tärkeämpää olisi tietää missä järjestyksessä kutsut palvellaan. Kuten huomaamme seuraavassa versiossa, paremmalla muuttujien valinnalla mallista saadaan huomattavasti selkeämpi.

5.2 Toinen versio

Toisessa versiossa on lähestymistapa ongelmaan muutettu täysin edelliseen verrattuna. Muuttujina on z_{ijk}, joka ilmaisee ajetaanko kutsusta i kutsuun j hissillä k sekä r_k joka kertoo hissin lähtösuunnan (r=0 alas, r=1 ylös). Mikäli ongelma ajateltaisiin verkkona on nyt muuttujana on verkon kaari solmun sijaan. Vaikka tällä tapaa muuttujia onkin enemmän, on ongelma näin yksinkertaisemmin esitettävissä.

$$\sum_{i=0}^{N} \sum_{k=1}^{M} z_{ijk} = 1, \quad j = 1,\ldots,N \quad \textbf{5.13}$$

$$\sum_{j=1}^{N+1} \sum_{k=1}^{M} z_{ijk} = 1, \quad i = 1,\ldots,N \quad \textbf{5.14}$$

$$\sum_{i=0}^{N} z_{i,N+1,k} = 1, \quad k = 1,\ldots,M \quad \textbf{5.15}$$

$$\sum_{j=1}^{N+1} z_{0jk} = 1, \quad k = 1,\ldots,M \quad \textbf{5.16}$$

Kaavassa 5.13-5.16 on hissien reittien yksikäsitteisyyden takaavat rajoitteet. Kaavassa 5.13 oleva rajoite määrää, että täsmälleen yksi hissi täsmälleen yhdestä toisesta kutsusta saapuu kutsuun i. Kaavassa 5.14 oleva rajoite määrää, että täsmälleen yksi hissi lähtee kutsusta i täsmälleen yhteen kutsuun. Seuraava kaavassa 5.15 oleva rajoite kertoo, että jokainen hissi saapuu täsmälleen yhdestä kutsusta päätepisteeseensä ja kaavassa 5.16 oleva rajoite kertoo, että jokainen hissi lähtee lähtöpisteestään täsmälleen yhdestä pisteestä.

$$z_{ijk} \le d_{ijk}^{0} + r_k, \quad i = 1,\ldots,N, j = 1,\ldots,N, k = 1,\ldots,M \quad \textbf{5.17}$$

$$z_{ijk} \le d_{ijk}^{1} + 1 - r_k, \quad i = 1,\ldots,N, j = 1,\ldots,N, k = 1,\ldots,M \quad \textbf{5.18}$$

Kaavassa 5.17 oleva rajoite määrää kutsujen palvelu järjestyksen d^{0}_{ijk};n mukaiseksi, kun hissi lähtee alaspäin ($r_k = 0$). Muuttuja d^{0}_{ijk} ilmoittaa onko kutsu i

kiertojärjestyksessä ennen kutsua j hissin k lähtiessä alaspäin. Vastaavasti rajoite 5.18 määrä kutsujen järjestyksen d^1_{ijk}:n mukaiseksi, kun hissi lähtee ylöspäin ($r_k = 1$). Muuttuja d^1_{ijk} ilmoittaa onko kutsu i kiertojärjestyksessä ennen kutsua j hissin k lähtiessä ylöspäin. Olennaisesta on, että vain toinen näistä rajoitteista on voimassa, riippuen hissin lähtösuunnasta.

$$z_{ijk} \in \{0,1\},\ i = 0,...,N, j = 1,...,N+1, k = 1,...,M \quad \textbf{5.19}$$

$$r_k \in \{0,1\},\ k = 1,...,M \quad \textbf{5.20}$$

Kaavoissa 5.19 ja 5.20 on määrätty kaikkien z_{ijk} ja r_k olevan binäärimuuttujia.

Tässä versiossa mallia oli vain hissin reittien mallinnukseen liittyvät rajoitteet. Kohdefunktio joka puuttui on lisätty malliin sen seuraavassa versiossa.

5.3 Kolmas versio

Kolmas versio mallista on seuraavanlainen. Siinä kohdefunktiona on kutsujen ajoaika, joka kutsukohtaisesti on kirjoitettu auki rajoitteena. Muuttujina on edellisen mallin mukaan z_{ijk}, joka ilmaisee ajetaanko kutsusta i kutsuun j hissillä k sekä r_k joka kertoo hissin lähtösuunnan (r=0 alas, r=1 ylös).

$$\min F = \sum_{i=0}^{N+1} T_i \quad \textbf{5.21}$$

$$\{T_i\}_{i=1}^{n} = \sum_{k=1}^{M} \sum_{j=0}^{N+1} z_{jik} \left(t_{jik} + t_{jk} + T_j \right) \quad \textbf{5.22}$$

Kaavassa 5.21 on kohdefunktio jossa T_i on ajoaika jokaiseen kutsuun sitä palvelevalla hissillä. Kaavassa 5.22 on kutsun i ajoaika T_i esitetty kutsun j ajoajan T_j, kutsusta j kutsuun i kulkemiseen hissillä k kuluvan ajan t_{jik} ja hissin k kutsuun j pysähtymiseen kuluvan ajan t_{jk} avulla. Ajoaikaan T_i siis ei kuulu kutsuun i pysähtymiseen kuluva aika.

$$\sum_{i=0}^{N} \sum_{k=1}^{M} z_{ijk} = 1,\ \ j = 1,...,N \quad \textbf{5.23}$$

$$\sum_{j=1}^{N+1} \sum_{k=1}^{M} z_{ijk} = 1,\ i = 1,...,N \quad \textbf{5.24}$$

$$\sum_{i=0}^{N} z_{i,N+1,k} = 1,\ k = 1,...,M \quad \textbf{5.25}$$

$$\sum_{j=1}^{N+1} z_{0jk} = 1,\ k = 1,...,M \quad \textbf{5.26}$$

$$z_{ijk} \le d^0_{ijk} + r_k,\ i = 1,...,N, j = 1,...,N, k = 1,...,M \quad \textbf{5.27}$$

$$z_{ijk} \le d^1_{ijk} + 1 - r_k,\ i = 1,...,N, j = 1,...,N, k = 1,...,M \quad \textbf{5.28}$$

$$z_{ijk} \in \{0,1\},\ i = 0,...,N, j = 1,...,N+1, k = 1,...,M \quad \textbf{5.29}$$

$$r_k \in \{0,1\},\ k = 1,...,M \quad \textbf{5.30}$$

Muut malliin liittyvät rajoitteet ovat samat kuin mallin edellisessäkin versiossa. Edellä kuvattu malli ei kuitenkaan ole täysin toimiva, ensinnäkään se ei ole lineaarinen kohdan 5.22 takia ja lisäksi siitä puuttuu jotain muuttujiin ja reitin yhtenäisyyteen liittyviä rajoitteita. Nämä puutteet on korjattu mallin seuraavassa versiossa.

5.4 Neljäs versio

Neljäs ja optimoinnissa käytetty malli on kuten edellinen malli, mutta muutamin tarkennuksin. Suurin muutos on rajoitteen 5.22 linearisointi apumuuttujan y_{ijk} avulla.

$$\min F = \sum_{i=1}^{N} T_i \qquad \textbf{5.31}$$

$$\{T_i\}_{i=1}^{n} = \sum_{k=1}^{M}\left(z_{0ik}\left(t_{0ik} + t_{0k}\right) + \sum_{j=1}^{N}\left(z_{jik}\left(t_{jik} + t_{jk}\right) + y_{jik}\right)\right) \qquad \textbf{5.32}$$

$$y_{ijk} + T^{MAX}(1 - z_{ijk}) \geq T_i, \quad i = 1,...,N,\ j = 1,...,N, k = 1,...,M \qquad \textbf{5.33}$$

Kaavoissa 5.31-5.33 on esitetty kohdefunktio ja tähän liittyvät rajoitteet. Olennainen ero edelliseen malliin on kaavassa 5.22 olevan tulon z_{jik}*T_j linearisointi muuttujan y_{ijk} avulla. Kaavassa 5.33 on mukana apumuuttuja T^{MAX}, joka on jokin suuri luku joka on suurempi kuin mikään T_i. Idea rajoitteessa on, että kun z_{ijk} on nolla, se ei rajoita mitään. Jos z_{ijk} on 1 on y_{ijk} suurempi tai yhtä suuri kuin T_i. Optimiratkaisussa siis y_{ijk} saa arvon 0 kun z_{ijk} on 0 ja arvon T_i kun z_{ijk} on 1.

$$\sum_{i=0}^{N}\sum_{k=1}^{M} z_{ijk} = 1, \quad j = 1,\ldots,N \qquad \textbf{5.34}$$

$$\sum_{j=1}^{N+1}\sum_{k=1}^{M} z_{ijk} = 1, \ i = 1,...,N \qquad \textbf{5.35}$$

$$\sum_{i=0}^{N} z_{i,N+1,k} = 1, \ k = 1,...,M \qquad \textbf{5.36}$$

$$\sum_{j=1}^{N+1} z_{0jk} = 1, \ k = 1,...,M \qquad \textbf{5.37}$$

$$\sum_{i=0}^{N} z_{ijk} = \sum_{i=1}^{N+1} z_{jik}, \quad j = 1,...,N, k = 1,...,M \qquad \textbf{5.38}$$

Kaavoissa 5.34-5.38 on kuten edellisessäkin versiossa reitin yksikäsitteisyyden takaavat rajoitteet, mutta mukana on myös reitin yhtenäisyyden takaava rajoite 5.38. Tämä määrää sen, että sama hissi joka tulee kutsuun j myös lähtee siitä.

$$z_{ijk} \leq d_{ijk}^{0} + r_k, \ i = 1,...,N, j = 1,...,N, k = 1,...,M \qquad \textbf{5.39}$$

$$z_{ijk} \leq d_{ijk}^{1} + 1 - r_k, \ i = 1,...,N, j = 1,...,N, k = 1,...,M \qquad \textbf{5.40}$$

Kutsujen palvelujärjestyksen määräävät rajoitteet ovat samat kuin edellisessäkin mallissa.

$$z_{ijk} \in \{0,1\}, \ i = 0,...,N, \ j = 1,...,N+1, k = 1,...,M \qquad \textbf{5.41}$$

$$r_k \in \{0,1\},\ k = 1,...,M \qquad \textbf{5.42}$$

$$T_i \geq 0,\ i = 1,...,N \qquad \textbf{5.43}$$

$$y_{ijk} \geq 0,\ i = 1,...,N,\ j = 1,...,N,\ k = 1,...,M \qquad \textbf{5.44}$$

Rajoitteisiin kaavoissa 5.41-5.44 on lisätty rajoitteet apumuuttujille T_i ja y_{ijk}.

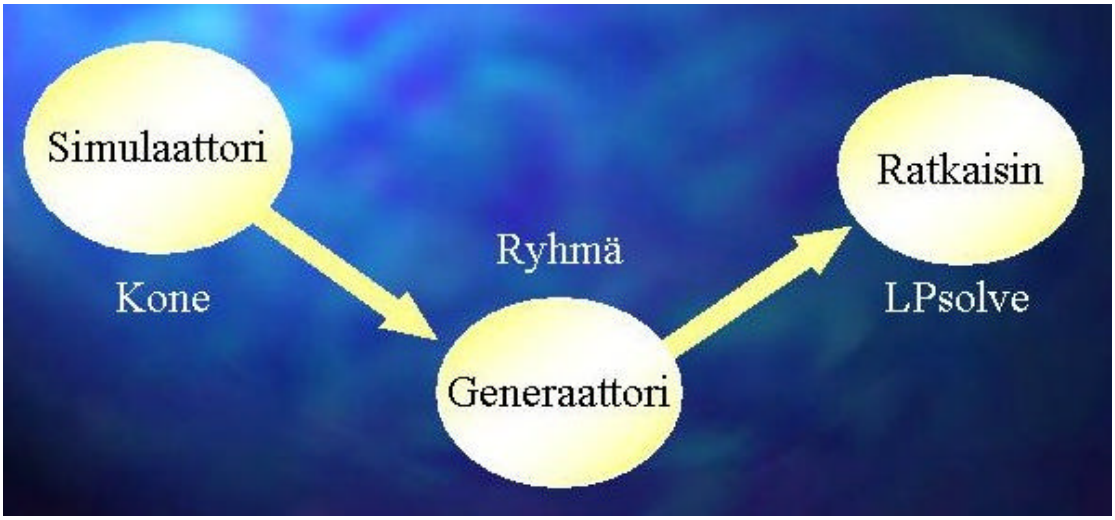
Edellä esitetty malli on lopullinen malli, jota on käytetty optimoinnissa. Mallissa on suuri määrä rajoitteita, noin luokkaa n^2m. Myös muuttujien määrä on luokkaa n^2m. Malli voi siis olla mahdoton ratkaista jo muutamalla kymmenellä kutsulla ja muutamalla hissillä.

Tämäkään malli ei ole täydellinen. Vakavin puute siinä lienee se, että korikutsuja ei ole suoranaisesti huomioitu mallissa. Periaatteessa nykyisessä mallissa ei erotella kori- eikä kerroskutsuja mitenkään. Mahdollisesti ne voitaisiin lisätä malliin pelkästään kiertosuunta rajoitteiden d^0_{ijk} ja d^1_{ijk} avulla. Näiden avulla siis vaadittaisiin, että kerroskutsu palvellaan ennen sitä vastaavaa korikutsua. Lisäksi vaadittaisiin jokin rajoite, joka määräisi saman hissin palvelemaan kerroskutsun ja siihen liittyvän korikutsun.

Nykyinen malli on jossain määrin TSP - ongelman mukainen. Pyritään muodostamaan reitti joka kulkee kaikkien pisteiden kautta. Olennainen ero TSP:hen lienee se, että tässä tapauksessa reittejä on useita (useampi hissi). Jos malliin lisättäisiin korikutsuihin liittyvät rajoitteet tulisi siitä PDP - ongelma.

6. Tekninen toteutus

Mallin ratkaiseminen koostui kolmesta eri päävaiheesta, alkutiladatan muodostamisesta, alkutiladatan määräämän ongelman muodostamisesta kokonaislukumallin muotoon, ja kokonaislukumallin optimin ratkaisemisesta. Alkutiladata on siis kussakin tilanteessa yhtä kuin ratkaistava kutsut-hissit allokaatio-ongelma. Alkutiladatan määrämuoto saatiin Kone Oyj:ltä, ja se on yhtiön käyttämän hissisimulaattorin tuloste. Se kertoo hissien ja kutsujen tilatiedot, sekä rakennuksen ja hissien ominaisuuksia. Tätä alkutiladataa muuttelimme sitten omassa simuloinnissamme tarpeen mukaan, mm. lisäämällä ja vähentämällä hissejä, kutsuja, kutsujen suuntia ja paikkoja jne.



Kuva 3: Rakennetun kokonaislukumallin ratkaisemisen eri vaiheet

Seuraavaksi alkutiladata tuli tulkita kokonaislukumallimme muotoon. Tähän käytettiin ryhmän itse ohjelmoimaa generaattoria. Ongelman kääntäminen kokonaislukumallin muotoon tehtiin ohjelmalla siksi että tehtävässämme kokonaislukumallista tulee hyvin nopeasti hyvin suuri, ja tällöin mallin kirjoittamisesta käsin tulee hyvin työlästä. Esimerkiksi ongelmassa jossa on pienin mahdollinen eli kahden hissin hissiryhmä, ja tälle hissiryhmälle kolme eri kerroskutsua, syntyy 34 eri muuttujaa. Kohdefunktiota tehtävässä on vain yksi rivi, mutta rajoitteita tälle hyvin yksinkertaiselle ongelmalle syntyy jo yhteensä viisi A4 sivua.

Paitsi että alkutiladata tuli saattaa kokonaislukumalliksi, tuli se myöskin esittää muodossa jota ratkaisuohjelmamme ymmärtäisi. Kaikeksi onneksi ratkaisuohjelmistomme otti sisäänlukutiedoksi normaalia lukukelpoista tekstimuotoista tietoa jota oli helppo myös itse suoraan tarkistaa ja tarkastella. Myös tässä, sisäänluettavan mallin oikeaan, ratkaisuohjelman lukemaan syntaksiin, saattamisessa, generaattori oli hyvin tärkeä.

Viimeinen vaihe on kokonaislukumallin optimin ratkaiseminen. Ratkaiseminen tapahtui kokonaislukumallien ratkaisemiseen suunnitellulla tietokoneohjelmistolla - nimeltään LPsolve.

6.1 Alkutiladata

Alkutiladatan määrämuoto nähdään taulukosta yksi. Alkutiladata määrittelee hissiryhmän tilan, kutsut ja suorituskykyarvot. Rakennukseen liittyvät alkutilatiedot ovat: rakennuksen kerrosten lukumäärä, rakennuksen kerrosten korkeusasema metreissä, ja rakennuksen hissiryhmässä olevien hissien lukumäärä. Tiedot näkyvät taulukosta yksi: riviltä kaksi kerrosten- ja hissien lukumäärä (otsikoilla #nF nE), ja kerrosten korkeusasemat riviltä seitsemän alkaen (otsikolla #h). Tässä taulukossa 1 olevassa esimerkissä hissejä on siis kaksi ja kerroksia 20.

Taulukko 1: Hissin alkutiladata hissisimulaatorin käyttämässä muodossa

```
#nF nE
20 2
#v a ? F?????
6 1.2 18 1 0 0 2 1 34
6 1.2 18 6 0 0 2 1 34
#h  u d??? ???
0.0  0 0 0 0 0 0 0 0 0
3.4  0 0 0 0 0 0 0 0 0
6.8  0 0 0 0 0 0 0 0 0
10.2 0 1 0 0 0 0 0 0 0
13.6 0 0 0 0 0 0 0 0 0
17.0 0 0 0 0 0 0 0 0 0
20.4 0 0 0 0 0 0 0 0 0
23.8 0 0 0 0 0 0 0 0 0
27.2 0 0 0 0 0 0 0 0 0
30.6 0 0 0 0 0 0 0 0 0
34.0 0 1 0 0 0 0 0 0 0
37.4 0 0 0 0 0 0 0 0 0
40.8 0 0 0 0 0 0 0 0 0
44.2 0 0 0 0 0 0 0 0 1
47.6 0 0 0 0 0 0 0 0 0
51.0 0 0 0 0 0 0 0 0 0
54.4 0 0 0 0 0 0 0 0 0
57.8 0 0 0 0 0 0 0 0 0
61.2 1 0 0 0 0 0 0 0 0
64.6 0 0 0 0 0 0 0 0 0
```

Hisseihin liittyvät tiedot ovat hissin maksiminopeus, hissin kiihdytys- ja jarrutuskiihtyvyys, hissikorien ko. hetken sijaintikerros, ovien sulkeutumisen ja ovien aukeamisen kesto sekunneissa, sekä ovien aukioloaika. Simulaattori muotoisessa datassa taulukossa 1 tiedot näkyvät riveillä neljä ja viisi molemmille hisseille. Vastaavassa järjestyksessä lukien tiedot ovat otsikoiden #v, a, F ja kolmen viimeisen kysymysmerkin alla. Tässä tapauksessa esimerkiksi molempien hissien maksiminopeus on siis 6m/s ja kiihtyvyys molempiin suuntiin 1,2 m/s. Vain nopeus, kiihtyvyys ja lähtökerros luetaan tästä, loput on ohjelmoitu generaattoriin parametreinä.

Kerroskutsuihin liittyviä alkutilatietoja ovat kutsujen sijaintikerrokset ja kutsujen suunnat. Nämä tiedot näkyvät simulaattorin kerrostiedoissa, eli taulukossa riviltä seitsemän alkaen. Jokainen rivi kertoo yhden kerroksen tiedot. Kerrostiedoissa, ensimmäisessä sarakkeessa, otsikolla #h, on kerroksen korkeusasema, seuraavaksi otsikolla u ylöspäinkutsut, ja sitten otsikolla d alaspäinkutsut. Jos kerroksessa on kutsu (tai molemmat yhtä aikaa) asianomaisessa kutsusarakkeessa on ykkönen, muuten arvo on nolla. Muut numerot ovat ongelmassamme tarpeettomia.

Esimerkin alkutiladatassa on siis kolme kutsua. Kutsut ovat: ylöspäinkutsu kerroksessa numero 18, ja alaspäinkutsut kerroksissa numero 3 ja 10. Tässä kerrosnumerointi menee siten että ensimmäinen eli maantasakerros, on kerros numero nolla.

6.2 Generaattori

Kuten jo mainittu koska kokonaislukumalli (kappaleessa “5.4”) tuottaa jo pienillä kutsujen ja hissien määrillä suuren määrän rajoitteita, päätimme rakentaa ongelman kokonaisluku malliksi kääntävän generaattorin. Generaattorin edut mallin käsin määrittelyyn verrattuna ovat tehtävän muodostamisen nopeus ja saadun mallin virheettömyys. Haitaksi projektissa voidaan lukea lähinnä generaattorin ohjelmoimiseen ja testaamiseen kuluva aika. Liitteessä 2 esitellään osa generaattorin lähdekoodista.

Tehtävä tulee myös saada käytetyn ratkaisuohjelmiston vaatimaan standardimuotoon. Tämä siksi jotta ratkaisuohjelmisto pystyisi suoraan lukemaan generaattorin kääntämän ongelman, omana syötteenänsä.

Generaattorin tehtävät voidaan jakaa seuraaviin vaiheisiin:

- Alkutilan lukeminen
  - Hissin paikat, nopeudet ja kiihtyvyydet
  - Kutsukerrokset
  - Ylöspäinkutsut
  - Alaspäinkutsut
  - Kerroskorkeudet
  - Malliin liittyvät vakiot T^{MAX}
- Mallin parametrien laskeminen alkutilojen perusteella
  - Kutsuvälien siirtymisajat t_{ijk}
  - Kiertojärjestyksen määrääminen d^0_{ijk} ja d^1_{ijk}
- Mallin muodostaminen ja kirjoittaminen
  - Kohdefunktio
  - Rajoitteet
  - Kokonaisluku muuttujat

Ohjelmoimamme generaattori lukee syötteenänsä taulukon yksi simulaattorimuotoisen alkutiladatan. Generaattori muodostaa annetusta lähtötilanteesta, johon kuuluvat hissit, kutsut, rakennus ja näihin liittyvät parametrit, rakennetun lineaarisen kokonaislukumallin määrittelemän tehtävän. Tämän datan se kääntää kokonaislukumallin vaatimaan muotoon. Liitteessä neljä näemme taulukossa yksi esitetyn esimerkkiongelman alkutiladatan generaattorin kääntämänä, rakentamamme kokonaislukumallin mukaisena esityksenä.

Taulukko 2: Generaattorin ilmoittama luettu alkutiladata

```
1:Elevator at 1 floor, max speed 6.0 and acceleration 1.2
2:Elevator at 6 floor, max speed 6.0 and acceleration 1.2
0: height 0.0
1: height 3.4
2: height 6.8
3: height 10.2 v
4: height 13.6
5: height 17.0
6: height 20.4
7: height 23.8
8: height 27.2
9: height 30.6
10: height 34.0 v
11: height 37.4
12: height 40.8
13: height 44.2
14: height 47.6
15: height 51.0
16: height 54.4
17: height 57.8
18: height 61.2  ^
19: height 64.6
1:F3 down
2:F10 down
3:F18 up
```

Generaattori ilmoittaa ensin lukemansa ongelman tiedot, ja tulostaa sitten kokonaislukumallin yhtälöt. Taulukossa kaksi nähdään generaattorin lukema esimerkkiongelman alkutila kokonaisuudessaan. Alkutila tulostuksesta nähdään luetut tiedot eli hissien lukumäärä, niiden ominaisuudet, kerrosten numerointi ja niiden korkeusasemat, sekä kutsujen sijainnit ja niiden suunnat. Kutsut ja niiden suunnat on esitetty paitsi kerroksissa merkeillä ^ (ylös) ja v (alas), mutta myös yhteenvetona kolmella viimeisellä rivillä. Alkutiedot ovat luonnollisesti kaikki samat kuin annetut tiedot, jotka ovat myös taulukossa yksi.

Työmäärältään suurin tehtävä generaattorin osalta oli mallin parametrien laskeminen alkutilojen perusteella. Erityisesti kiertojärjestyksen määrääminen osoittautui hankalaksi. Itse mallin muodostaminen oli jopa odotettua helpompaa.

Generaattorin toteutuskieleksi valittiin Java lähinnä koska ryhmässä oli kokemusta tämän kielen käytöstä. Väistämättömiä etuja ovat kuitenkin Java:n alustariippumattomuus, valmiit kirjastot, tehokkuus ja korkean tason kielen rakenne. Toisena vaihtoehtona olisi generaattorin toteuttaminen C/++:lla joka ei kuitenkaan olisi ollut yhtä joustava. Kaikin puolin Java:n valintaan generaattorin toteutuksessa ollaan tyytyväisiä, sillä se toimii mukisematta kaikissa Java ympäristöissä ja mallin generoimiseen kuluu aikaa vain joitain sekunteja.

6.3 Ratkaisuohjelmisto

Ratkaisuohjelmistona käytettiin LPsolve nimistä ohjelmistoa. LPsolve on tarkoitettu nimenomaan lineaariseen kokonaisluku ongelmien ratkaisemiseen. Ohjelma on ilmainen, internetistä ladattavissa ja vapaasti levitettävissä sekä muokattavissa. LPsolve pystyy tekijänsä mukaan ratkaisemaan kohtuullisen kokonaisia ongelmia, jossa on tuhansia rajoitteita. Erikoista kuitenkin oli että kattavasta etsimisestä huolimatta emme löytäneet ohjelmasta tarkempaa sanallista ja teknistä spesifikaatiota. Olisimme kaivanneet tietoja esimerkiksi rajoitteiden tai muuttujien lukumäärästä, jotka ohjelma pystyy enintään käsittelemään. Myöskään muuta yleistä kuvausta ohjelman toiminnasta emme löytäneet.

Ohjelma otti ongelman syötteekseen tiedostona, liitteen neljä esittämässä eli tekstimuodossa. Tekstimuotoinen syöte on erinomainen ominaisuus, sillä siten syöte on myös helposti ohjelman käyttäjän tarkasteltavissa ja analysoitavissa. Lineaarisen kokonaislukumallin ratkaisemisessa LPsolve käyttää Branch and Bound -tekniikkaa. Esimerkin mukaisen ongelman optimin ratkaisemisessa ohjelmalla meni hyvin lyhyt aika, ehkä noin kolme sekuntia.

Ohjelma antaa tulosteensa ratkaisusta liitteen viisi esittämässä muodossa. Liitteessä vastausdata on kokonaisena. LPsolven löytämän ratkaisun tulostus, eli tehtävän globaali optimi ja siihen vaadittavat hissien reitit, nähdään lyhennettynä mutta olennaisin ratkaisun kertovin osin, taulukossa kolme. Esimerkkiongelmamme odotusaikojen minimiksi saadaan 61,2 sekuntia, mikä kohdefunktion arvona nähdään. Ratkaisun vaatima hissien reitti on nähdään tutkimalla muuttujia z, joita tulkitaan z[kutsusta][kutsuun][hissillä]. Reitti toteutuu mikäli z:n arvo on yksi.

Taulukko 3: Esimerkkitehtävän optimi ja hissien reitit.

Value of objective function: 61.20216179	
r[1]	0
r[2]	0
y[1][1][2]	0
y[1][1][1]	0
y[3][1][2]	0
y[2][1][1]	0
y[2][1][2]	0
y[3][1][1]	0
z[1][3][2]	0
z[0][3][1]	1
z[0][1][1]	0
z[0][1][2]	1
y[1][2][2]	10.831
y[1][2][1]	0
y[3][2][2]	0
y[2][2][1]	0
y[2][2][2]	0
y[3][2][1]	0
z[3][4][1]	1
z[2][4][2]	1
y[1][3][2]	0
y[1][3][1]	0
y[3][3][2]	0
y[2][3][1]	0
y[2][3][2]	0
y[3][3][1]	0
T[1]	10.831
T[2]	30.738
T[3]	19.633
z[1][2][2]	1

Malli käsittelee eri kutsuja niiden juoksevan tunnusnumeroinnin perusteella. Kutsut numeroidaan aina juoksevasti alhaalta ylöspäin eli tässä esimerkissä saadaan seuraava numerointi: kerroksessa numero kolme alaspäin oleva kutsu on kutsu numero yksi, kerroksen nro

10 alaspäin oleva kutsu on kutsu numero kaksi, ja kerroksen nro 18 ylöspäin oleva kutsu on kutsu numero kolme.

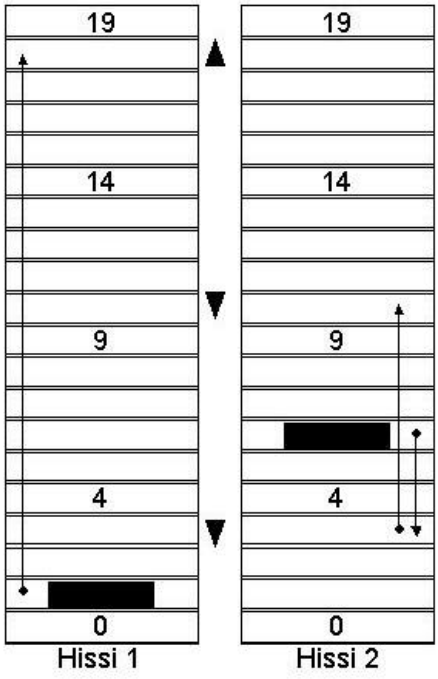
Optimaaliseen ratkaisuun johtava hissiryhmän ohjaus nähdään viereisestä taulukosta, joka on siis myös osa ratkaisutulosteesta. Optimaalisessa ratkaisussa hissi yksi menee suoraan kutsuun kolme, eli hissi lähtee kerroksesta nro 1 (yksi yli maantason), ja menee kerrokseen nro 18, ja jää sinne.

Hissi kaksi taas puolestaan lähtee kerroksesta nro 6, laskeutuu ensin kerrokseen nro 3 poimii matkustajan, ja menee sitten kerrokseen nro 10, ja jää sinne. Lisäksi taulukossa näkyy ylimääräiset ns. lopputilareitit, joista ei kuitenkaan aiheudu vaikutusta aikaan. Seuraavassa, kappaleessa 7 analysoidaan ratkaisua ja sen optimaalisuutta tarkemmin.

Käytetyn ratkaisuohjelman eli LPsolven löytymisen jälkeen emme ei juurikaan tutustunut muihin ohjelmistoihin. Tämä johtui siitä, että pikaisella katsauksella näytti siltä että joutuisimme rakentamaan joka ohjelmalle oman uuden generaattorin. Tämä siksi koska ohjelmien vaatima sisäänluettavan tiedon syntaksi ei eri ohjelmilla ollut sama. Ohjelmia ei myöskään löytynyt kovin monia vapaasti ladattavina.

7. Tulokset ja mallin hyvyyden arviointi

Edellä esitellyn esimerkki ongelman ratkaisu on esitetty havainnollisemmin kuvassa neljä. Hissien alkupaikat on merkitty kuvaan tummennetuilla suorakaiteilla, kutsukerrokset ja kutsujen suunnat kolmioilla. Ratkaistu hissien optimaalinen kulku on merkitty nuolien avulla.



Kuva 4: Mallin löytämä optimaalinen ratkaisu

Teimme vertailua siitä millaisia aikoja muut järkevän tuntuiset reitit voisivat tarjota. Taulukoissa 4 nähdään vertailu reiteistä, joissa kumpikin hissi palvelee ainakin yhden kutsun. Tässä vertaillaan hissien kulkemien kerrosvälien määrää toisiinsa.

Taulukko 4: Pareittainen reittivertailu

Hissi 1 reitti kutsuun	Hissi 2 reitti kutsuun	Hissi 1 ajamat kerrosvälit	Hissi 2 ajamat kerrosvälit
1	**3-2**	**2**	**12+9**
1	2-3	2	4+9
2	**1-3**	**9**	**3+15**
2	3-1	9	12+15
3	**1-2**	**17**	**3+7**
3	2-1	17	4+7
3-2	**1**	**17+9**	**3**
2-3	1	9+9	3
1-3	2	2+15	4
3-1	**2**	**17+15**	**4**
1-2	3	2+7	12
2-1	**3**	**9+7**	**12**

Kaksi ensimmäistä saraketta kertovat kunkin hissin reitin. Kaksi seuraavaa kertovat hissien kulkemien kerrosvälien lukumäärän. Värillä merkityt rivit ovat kiellettyjä

reittejä kiertosuuntarajoitteiden perusteella. Pareittaisista vertailuista paremmat on esitetty lihavoituina. Niiden vertailu on esitetty laskemalla kokonaisodotusaika taulukkoon 5. Ensimmäisessä sarakkeessa on hissin numero, toisessa sen reitti ja kolmannessa reittiin liittyvät kerrosvälit. Seuraavat kolme saraketta kertovat matkan, siihen käytetyn ajan sekä pysähdyksiin käytetyt ajat.

Taulukko 5: Reittiaikojen vertailu

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 3	17	57,8	14,63	5	19,63
2	0 -> 1	3	10,2	5,83	5	10,83
2	1 -> 2	7	23,8	8,91	11	30,74
						61,20

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 2	9	30,6	10,10	5	15,10
2	0 -> 1	3	10,2	5,83	5	10,83
2	1 -> 3	15	51	13,50	11	35,33
						61,26

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 1	2	6,8	4,76	5	9,76
2	0 -> 3	12	40,8	11,80	5	16,80
2	3 -> 2	9	30,6	10,10	11	37,90
						64,46

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 3	17	57,8	14,63	5	19,63
1	3 -> 2	9	30,6	10,10	11	40,73
2	0 -> 1	3	10,2	5,83	5	10,83
						71,20

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 2	9	30,6	10,10	11	21,10
1	2 -> 1	7	23,8	8,91	5	35,01
2	0 -> 3	12	40,8	11,80	5	16,80
						72,91

Hissi	Reitti	Kerrosvälejä	Matka	Liike	Pysähdys	Odotusaika
1	0 -> 3	17	57,8	14,63	5	19,63
1	3 -> 1	15	51	13,50	11	44,13
2	0 -> 2	4	13,6	6,73	5	11,73
						75,50

Viimeinen sarake on kunkin kutsun odotusaika ja odotusaikojen summa alla. Reitit on merkitty paremmuusjärjestykseen siten, että optimiratkaisu on ylinnä. Laskelmista on helppo havaita että annetulle esimerkkiongelmalle malli löysi globaalin optimin.

7.1 Ratkaisun optimaalisuus

Branch and bound algoritmiin liittyy muutama ongelma kun pohditaan sen antaman tuloksen optimaalisuutta. Algoritmi antaa vastaukseksi ainoastaan yhden ratkaisun, vaikka ratkaisuja olisi olemassa useita. Käyvässä alueessa ei välttämättä myöskään ole olemassa kokonaislukuratkaisuja.

Jos tehtävä kuitenkin on mielekkäästi rakennettu, antaa branch and bound aina optimaalisen ratkaisun. Syy tähän on tehtävän lineaarisuus ja siitä johtuva konveksisuus. Vaikka konveksi alue jaetaankin uusin rajoittein ovat nämäkin lineaarisia ja niiden muodostamat tehtävät konvekseja. Näin ollen kyseessä ei voi olla kuoppa ja ratkaisu on optimaalinen.

7.2 Ratkaisu vs. tosielämä

Esimerkkiongelman löydetty ratkaisu on tekemämme ongelman määrittelyn kannalta oikea ja realistinen. Ongelman rajauksen takia ei-virtuaalielämään verrattuna ratkaisu on kuitenkin huono. Ratkaisun huonous käytännön kannalta johtuu siitä että tehtävän laajuuden ja kurssin rajallisuuden takia ongelmaa oli rajattava. Rajaus tehtiin jättämällä korikutsut kokonaan ongelman ulkopuolelle.

On kuitenkin ilman korikutsujakin selvää, ja oletettava, että alaspäin tilatulla kerroskutsulla henkilö haluaa siirtyä rakennuksessa johonkin tilauskerrosta alempaan kerrokseen. Pitäen tämän mielessä ja katsoen esimerkki ongelmamme ratkaisua havaitaan että hissin kulku olisi tositilanteessa epäviisasta. Hissi kaksi nimittäin lähtee ensin alaspäin noutamaan alaspäintilauksen kerroksesta kolme, ja lähtee sitten ylöspäin kerrokseen 10 noutamaan toisen alaspäintilauksen. Tilaukset noutava hissi kulkee siis eri suuntaan kuin mikä on suunta minne tilaukset on tehty. Ratkaisu on kuitenkin mallin ja ongelman määrittelyn mukaan oikein.

7.3 Vertailu käytössä oleviin algoritmeihin

Vaikka alkuperäinen tavoite ole vertailla mallin antamaa ratkaisua olemassa olevien algoritmien tuottamiin ratkaisuihin, ei tähän tarjoutunut mahdollisuutta. Aineiston saaminen hissisimulaattorista ei ollutkaan aivan niin helppoa kuin mitä oli odotettu ja tästä syystä vertailusta jouduttiin luopumaan. Toisaalta mallin ratkaisun hankaluuksista johtuen aikataulukin oli sen verran tiukka, ettei vertailuja olisi voitu tehdä kovinkaan kattavasti.

7.4 Vaikeampien tehtävien ratkaiseminen

Malli siis löysi esimerkkiongelmamme optimin. Esimerkki oli kuitenkin hyvin yksinkertainen, hissiryhmä oli pienin mahdollinen ja kutsujakin hissiryhmälle oli vain kolme. Suurempien tehtävien kanssa esiintyy kuitenkin ongelmia. Tehtävät joissa hissiryhmän koko on edelleen kaksi hissiä, ratkeavat prosessillamme aina siihen asti kunnes kutsuja alkaa olla yli seitsemän kappaletta. Tämän jälkeen LPsolve alkaa ilmoittamaan tehtäville outoja ratkaisuja laskien hisseille mm. negatiivisia matkustusaikoja. Ongelma alkaa vaivaamaan myös heti jos hissiryhmän kokoa kasvatetaan suuremmaksi kuin kaksi hissiä. Esimerkiksi kolmen hissin hissiryhmän ratkaiseminen ei onnistu vaikka kutsuja hissiryhmälle olisi vain kolme.

Usko malliin ryhmässämme on kuitenkin kova, joten vikaa etsittiin ratkaisuohjelmistosta eli LPsolve:sta. LPsolven analyysi apuvälineillä apua ongelmaan ei kuitenkaan löytynyt, joten kahdeksan hissin hissiryhmän 30:n kutsun ongelmien ratkaiseminen jäi ryhmällemme haaveeksi.

Etsiessämme ja analysoidessamme syitä ongelmiin päädyimme siihen että vian täytyy olla ratkaisuohjelmistossa. Toista ratkaisuohjelmistoa emme kuitenkaan ehtineet testata, sillä generaattoria olisi pitänyt muuttaa jotta ongelma olisi saatu toisen ohjelman lukemaan muotoon. Lisäksi vaikutti siltä että ilmaisista kokonaislukuongelma ratkaisimista ei tuntunut löytyvän yhtään jo käyttämäämme ohjelmaa kehittyneempää

ratkaisinta. Pikemminkin päinvastoin. Kaupallisia ohjelmistoja emme päässeet testaamaan - niistä oletettavasti löytyisi parempia kokonaislukuongelma ratkaisimia. Mieleemme pälkähtikin että olisiko Systeemianalyysin laboratoriolla syytä olla tällainen ratkaisin.

8. Projektin arviointi

8.1 Aikataulu

Aikataulua rakennettaessa pyrittiin toteuttamaan seuraavat kriteerit:

- Realistisuus
- Kattavuus

Realistisuudella tarkoitetaan aikataulua, jota pystyttäisiin todennäköisesti seuraamaan, samalla kuitenkin aliarvioimatta etenemisnopeutta. Kattavuudella puolestaan tarkoitetaan jokaisen työtehtävän määrittämistä ja aikatauluttamista. Tämä tehtiin jottei työvaiheita unohtuisi ja jotta ryhmälle olisi selvää mitä olisi tehtävä projektin valmistumiseksi.

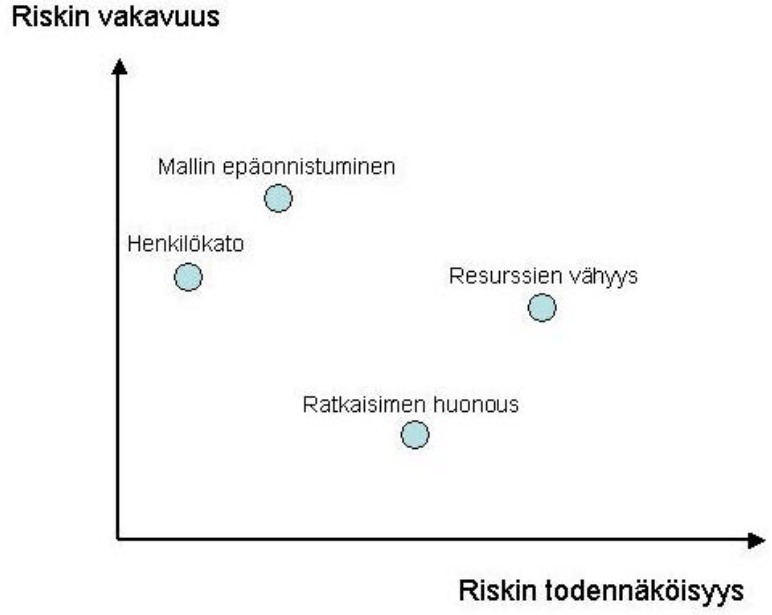
Aikataulun realistisuutta voidaan arvioida tarkastelemalla jälkiviisaudella tehtyjä kohtia ja niissä olevia aikaikkunoita. Suurin yksittäinen virhearviointi liittyi ratkaisuohjelmaan. Ohjelmaan tutustumisen arvioitiin kestävän viikon, kun se todellisuudessa vei useita viikkoja. Toisen ratkaisuohjelman tarvetta ei aikataulua tehdessä osattu odottaa, joten tämä jäi pois aikataulusta.

Onnistunut simulointi ja tulosten analysointi jäi käytännön ongelmista johtuen jälkeen aikataulusta. Käytännössä vertailu muihin tuloksiin jäi kokonaan projektin ulkopuolelle. Mallin rakentamiselle oltiin varattu juuri se aika mikä siihen kuluikin, joten tätä arviota voidaan pitää onnistuneena.

8.2 Riskinhallinta

8.2.1 Projektin alussa havaitut riskit

Projektin alussa oli tärkeää määritellä minkälaisia riskejä projektiin liittyisi. Havaittiin todennäköisimmiksi kuvassa 2 näkyvät riskit.



Kuva 5: Riskit

Ensimmäinen ja todennäköisin oli resurssien loppuminen kesken. Resursseilla tarkoitetaan aikaa ja henkistä pääomaa. Koska kyseessä oli ongelma josta ei tiedetty

juuri mitään ja josta ei taustamateriaalia löytynyt, oli olemassa mahdollisuus ettei ongelmaa saataisi ratkaistua annetussa ajassa. Ei myöskään voitu olla varmoja siitä, että kyvyillämme saataisiin tämänkaltainen ongelma ratkaistua. Riskin vakavuutta pohtiessamme totesimme sen olevan keskitasoa, sillä vaikka aika loppuisi olisi tekemästämme työstä kuitenkin todennäköisesti hyötyä asiaa jatkossa tutkiville. Näin ollen tämä riski sijoittuu keskitasolle pystyasteikolla ja oikealle vaaka-asteikolla kuvaajassa.

Toinen selvä riski oli ratkaisuohjelmiston toimivuus. LPsolve todettiin heti projektin alussa olevan riskialtis tekijä. Koska kellään ei ollut kokemusta tästä internetistä löytyneestä ohjelmistosta, emme voineet olla varmoja sen toimivuudesta, soveltuvuudesta emmekä sen tehokkuudesta. Oikeassa ja merkitsevässä projektissa tuskin oltaisiin luotettu tämän kaltaiseen ratkaisuun sen epävarmuuden sekä käyttäjätuen ja valmistajan vastuun puutteen vuoksi. Toisaalta projektia voitaisiin viedä pitkälle ilman ohjelmaa jo mallintamisen avulla. Näin ollen riskin toteutuman vakavuus ei ole niin suuri, ja se sijoittuu alas pystyasteikolla.

Mallia rakennettaessa on aina olemassa riski, ettei se vastaa todellisuutta tai ratkaise optimaalisesti ongelmaa. Tätä voidaan pitää vakavimpana riskinä kaikista, sillä ilman validia mallia kaikki muu työ menee hukkaan. Toisaalta näimme mallin epäonnistumisen mahdollisuuden pienempänä kuin muut riskit; sen parissa työskenteli ryhmän lisäksi projektin ohjaaja.

Ryhmä koki alussa erääksi pieneksi mutta olennaiseksi riskiksi henkilökadon. Koska kyseessä on opiskeluprojekti ei ryhmän jäsenten sitoutumista voida millään tavalla taata. Näin ollen ryhmän jäsenille tärkeämpien asioiden ilmaantuessa voisi kyseinen ryhmätyö jäädä pienemmälle huomiolle. Tämän riskin vakavuutta on hankala määritellä, sillä yhden jäsenen poisjäänti tuskin kaataa projektia, mutta kahden voisi sen hyvinkin tehdä.

8.2.2 Riskien toteutuminen

Kun mallia alettiin ajaa todettiin LPsolven rakenteesta johtuva ongelma. Tietyillä malleilla ohjelma ei kyennyt tuottamaan ratkaisua numeeristen ongelmien vuoksi. Muodostuneen matriisiin koon vuoksi sen kääntäminen ei onnistunut. Tämän epäkohdan poistamiseksi aikamuuttujia skaalattiin sopiviksi sekä haettiin hyvää T^{MAX} arvoa. Valitettavasti tämäkään ei poistanut ongelmia. Seuraavaksi etsittiin toista ratkaisuohjelmaa, joka olisi kyennyt käsittelemään suuria lukulukuja.

Tehtäessä malleja huomattiin parhaimman versionkin sisältävän huomattavan määrän muuttujia ja rajoitteita (kokoluokkaa n^2m). Tämän vuoksi mallin ratkaiseminen vaatii suurta laskentatehoa ja on jo erittäin raskasta muutamalla kymmenellä kutsulla ja muutamalla hissillä.

Kuten aikataulua käsittelevästä kappaleesta huomataan pysyttiin suunnitelmissa kohtalaisen hyvin. Muutamassa kohdassa jäätiin hieman jälkeen, mutta keskittämällä työpanosta saatiin nämä kohdat kiinni.

8.2.3 Puutteet riskinhallinnassa

Riskinhallinnassa suurin puute oli ehkä aktiivinen vaihtoehtojen etsiminen ja näin ollen liian suuri luottamus löydettyihin ratkaisuihin. Tästä esimerkkinä ratkaisuohjelmiston käyttäminen kunnes huomattiin sen puutteellisuus. Vasta tässä vaiheessa siirryttiin etsimään uutta ohjelmaa. Toki tiedettiin Systeemianalyysin

laboratoriolla olevan kaupallisen ohjelman testiversio, mutta sen käyttöä ei erikseen mietitty muuten kuin vertailumielessä.

Malleja päivitettiin kiitettävään tahtiin projektin edetessä, mutta vaihtoehtoja ei tehty. Luotettiin siis siihen että mallin pohja tuottaisi lopulta ratkaistavissa olevan muodon. Vaihtoehtoisesti oltaisiin voitu luoda toinen malli ja rakentaa sitä taustalla varmistaakseen loppupeleissä toimivan mallin.

8.3 Tuloksellisuus

8.3.1 Tavoitteiden saavuttaminen

Ryhmän tavoitteena oli luoda hissiryhmästä kokonaislukumalli, jonka avulla pystytään vertailemaan erilaisia hissien ohjausalgoritmeja. Ryhmä sai hissiryhmän mallinnettua siten, että se voidaan ratkaista kokonaislukuongelmana muutamalla eri ohjelmistolla. Voidaan sanoa, että ryhmä saavutti osan asettamistaan tavoitteistaan. Ryhmä sai valmiiksi toimivan mallin hissiryhmästä, mutta kyseisen mallin ratkaiseminen valitulla ohjelmistolla ei ole vielä onnistunut täydellisesti. Ryhmä on paikantanut vian kyseiseen ratkaisuohjelmistoon. Toinen tavoitteemme oli saada ratkaistua hissiryhmän kokonaislukumalli kohtuullisen kokoiselle hissiryhmälle, mutta tässä tavoitteessaan ryhmä ei onnistunut.

Toisaalta tästä on helppo jatkaa jollakin toisella ratkaisuohjelmistolla, sillä tarvitaan vain joitakin muutoksia ongelmageneraattoriin, jotta hissiryhmän malli saadaan ratkaisuohjelmiston lukemaan muotoon.

8.3.2 Mitä jäi tekemättä

Kuten alkuun todettiin on aihetta tutkittu varsin vähän, ainakin tutkimusta on dokumentoitu erittäin vähän, joten on helppoa sanoa, että paljon olisi vielä ollut tehtävää. Projektin alussa suunnitteli tekevänsä useampia malleja hissiryhmästä ja kokeilevansa monia ratkaisuohjelmistoja. Kyseinen suunnitelma oli varsin epärealistinen sen suuren työmäärän vuoksi, johon ryhmä havahtui vasta projektin edetessä.

Ryhmä olisi voinut tehdä useita malleja hissiryhmistä, jotta oltaisiin voitu vertailla mitkä niistä parhaiten ratkeisivat ratkaisuohjelmistoilla. Toisen mallin tekeminenkin olisi ollut varsin työläs urakka, koska sille olisi joutunut suunnittelemaan ja ohjelmoimaan uuden tehtävägeneraattorin ongelman syöttämiseksi ratkaisuohjelmistoon.

Eri ratkaisuohjelmistojen kanssa on sama ongelma kuin mallien kanssa eli jokaista ohjelmistoa varten olisi pitänyt rakentaa oma ongelmageneraattori. Jos malleja olisi vaikka kaksi ja kolme ratkaisuohjelmistoja niin pahimmassa tapauksessa oltaisiin jouduttu ohjelmoimaan kuusi ongelmageneraattoria.

Myöskin hissisimulaattori eri ohjailualgoritmien vertailemiseen olisi voitu toteuttaa myös, mutta se jäi tekemättä suuren työmäärän ja tiukan aikataulun takia.

Vaikuttaa siltä, että tätä projektia pitäisi vielä jatkaa, jotta saataisiin kunnollisia tuloksia joiden avulla voidaan arvioida eri ohjausalgoritmien tehokkuuksia.

8.3.3 Mitä olisi voitu tehdä paremmin projektissa?

Parannettavaa löytyy aina, myös tästä projektista. Projektin aloituksen kannalta suurin parannettava asia olisi projektin laajuuden ja tavoitteiden tarkempi määrittely. Nyt ryhmäläisille kesti jonkin aikaan saada selville, mitä tällä projektilla halutaan selvittää.

8.4 Ryhmätyö

8.4.1 Ryhmätyön toimivuus

Ryhmätyö on mitä ilmeisemmin toiminut hyvin, mikäli ryhmä on saavuttanut itsellensä asettamansa tavoitteensa ja lisäksi ryhmän jäsenet ovat vielä projektin valmistumisen jälkeen kavereita keskenään. Projektipalaverit vietettiin hyvässä ilmapiirissä, jossa työniloa ja ryhmähenkeä oli aistittavissa ilmassa.

Ainoastaan tiukka aikataulu asetti muutaman kerran tummia pilviä ryhmätyön ylle, koska ryhmäläiset eivät aina ehtineet panostamaan omasta ajastaan palaverin onnistumiseen tarpeeksi. Toisin sanoen, joskus kiireen takia kaikkia sovittuja hommia ei saatu tehtyä kunnolla, mutta nekin valmistuivat viimeistään seuraavaan kertaan mennessä.

8.4.2 Työnjako

Projektin käynnistettyä ryhmä ei tehnyt mitään varsinaista työnjakoa, vaan yleensä työt tehtiin siten, että jokainen ryhmän jäsen tutustui yleisesti aiheeseen, jonka jälkeen aihe jaettiin pienempiin osiin ryhmäläisten kiinnostuksen mukaan. Ryhmän lähtökohtana oli tehdä ryhmätyötä koko projektin ajan, jotta asioista saataisiin mahdollisimman monipuolinen kuva ja asiat tulisivat mietittyä monelta eri kantilta. Projektin työtehtäviä ei siis jaettu ennakolta jäsenien vahvuuksien mukaan, vaan projektin työt pyrittiin jakamaan jokaisen tapaamiseen jälkeen jäsenien halujen mukaan.

Kuitenkin projektin kuluessa muodostui ryhmäläisten kesken selkeä työnjako seuraavien tehtävien kesken. Muut työtehtävät jakautuivat varsin tasaisesti ryhmäläisten kesken.

8.4.2.1 Mallin suunnittelu

Hissiryhmän mallin suunnittelusta vastasi pääasiassa ryhmän ohjaaja Henri Hakonen. Projektin alkuvaiheissa Henri toi näytille kehittelemänsä hissiryhmän mallin, jota hän ehdotti tässä projektissa käytettäväksi malliksi. Muille ryhmän jäsenille olivat hissit ja hissiryhmät olivat vielä kyseisessä projektin vaiheessa varsin tuntemattomia, joten Henrin mallia alettiin innolla tutkia. Lopulta Henrin mallista tuli pienin muutoksin hissiryhmän malli, koska se osoittautui toimivaksi malliksi. Ryhmän jäsenillä oli myös joitakin omia viritelmiä hissiryhmän malliksi, mutta ne eivät olleet yhtä hyviä kuin Henrin mallit.

8.4.2.2 Generaattorin suunnittelu ja toteutus

Generaattorin suunnitteli ja toteutti Heikki, jolla oli alusta alkaen selkeä kuva siitä minkälainen generaattori tarvittaisiin. Lisäksi Heikillä oli eniten taipumuksia ohjelmoinnin suuntaan, joten generaattori valmistui varsin nopeasti. Generaattori toimi alusta alkaen hyvin, pienin muutoksiin saatiin se toimimaan käyttötarpeensa mukaisesti.

8.4.3 Yhteistyö ohjaavien henkilöiden kanssa

Yhteistyö ohjaajan kanssa oli antoisaa ja tuloksellista. Henriltä saimme heti kättelyssä paksun nipun aiheeseen liittyviä diplomi- ja lisensiaattitöitä, jotka perehdyttivät ryhmäläisen hissien ihmeelliseen maailmaan. Lisäksi ohjaaja auttoi niin hissiryhmän mallintamisessa kuin ratkaisuohjelmistojen kanssa.

8.5 Kritiikkiä ja pohdintaa itse projektiin

Ryhmä havaitsi tehtävän olevan vielä kurssin ongelman-antovaiheessa, varsin epäselvä ja jäsentymätön. Tehtävän epäselvyys aiheutti projektin alussa huomattavasti hämmennystä, ja epätietoisuutta. Tämä vei osaltaan hieman aikaa varsinaiselta tehtävältä, eli ongelman ratkaisulta. Palautetta antaisimmekin siitä että tehtävän ei tulisi olla vielä kurssin jo alkaessa näin epäselvä ja laaja. Ryhmältä kului vielä projektissa huomattavasti aikaa ongelman rajaamiseen ja oikean yksilöinnin ymmärtämiseen.

Yleisesti projektin tehtävästä voi sanoa että se muodostui alunperin liian laajaksi, suurelta osin juuri tehtävänannon jäsentymättömyyden vuoksi. Ryhmä joutui kyselemään ohjaajalta selvennyksiä tehtävänkuvaan vielä projektiviikolla 10; kolme viikkoa ennen projektin raportoimista.

Tehtävää voidaan ryhmän käsityksen mukaan myös pitää myös varsin laajana työkurssille. Kuten ryhmän taustatieto- ja teoria kartoituksessa selvisi, kukaan ei ollut ilmeisestikään koskaan aiemmin rakentanut kokonaislukumallia hissiryhmälle. Ongelmaa voidaankin, etenkin työkurssin puitteissa, luonnehtia vähintäänkin haastavaksi, ettemme sanoisi jopa vaikeaksi, näin rajoitetulle ajalle.

Tapaaminen kurssin vetäjän kanssa olisi ryhmän mielestä voinut olla hyödyllistä ja jo puolen tunnin aikana olisi voinut saada hyviä vinkkejä ja ehdotuksia projektin läpivientiin. Tässä olisi taas ryhmä voinut olla aktiivinen, mutta ehdotus on, että tällainen tapaaminen järjestettäisiin tulevilla kursseilla puolessa välissä kurssia. Ryhmän näkemyksen mukaan olisi myöskin ollut hyödyllistä tutustua Koneella kutsuallokointia miettiviin tahoihin, kuten Siikoseen ja jopa tutustua työympäristöön ja työvälineisiin. Tämä olisi voinut antaa lisää näkemystä projektityöhön ja motivoida ryhmää entistä enemmän.

Näistä kritiikin sanoista huolimatta, saimme kurssin aikana ryhmän ohjaajalta korvaamattoman suurta apua projektin ongelmien ratkaisussa. Siitä, ja pyyteettömästä kärsivällisyydestä, hänelle suuri kiitos.

9. Yhteenveto

Hissiryhmän kutsuallokoinnin mallintaminen kokonaislukuoptimoinnin tehtävänä on haaste. Hyvän mallin rakentaminen vaatii onnistunutta lähestymistapaa, mikä puolestaan löydetään vasta yrityksen ja erehdyksen kautta. Iteraatiokierroksia voi tulla jonkin verran ennen kuin sopiva malli on koossa. Tämän kaltainen tehtävä vaatii myös tuntemusta jostain ohjelmointikielestä, sillä onnistuminen on mahdollista vain jatkuvien ohjelmointiponnistusten kautta. Rajoitusten ja ongelman syöttäminen käsin on käytännössä mahdotonta ilman sopivaa generaattoria. Ryhmän rakentama malli vaikuttaa validilta ja tulosten saamisen ongelmat näyttävät liittyvän ennemmin ratkaisimessa oleviin virheisiin. Tämä kiteyttää projektin kulun kohtalaisen hyvin. Vaikka malli oli koossa jonkin aikaa aiheutti LPsolven käyttö mutkia matkaan ja hidasti projektia.

Mallin luominen ja testien tekeminen on projektina luonteeltaan iteratiivista. Tuloksia saadaan ja malleja parannetaan yhä uudestaan. Tämä herättää kysymyksen siitä, onko kyseisen kaltainen tehtävä sopiva projektityöseminaarin kurssille, jossa aika on kohtalaisen rajoitettu ja ryhmä kokoontuu vain projektin merkeissä. Tosiasiassa luulisi kyseisen kaltaisen ongelman vaativan kokopäivätyötä ja pidempää ajanjaksoa, jolloin olisi aikaa kehittää todellisesti erilaisia malleja ja ajoja. Nyt saadut tulokset tuskin ovat itsessään kovin merkittäviä.

Projektityötä tarkasteltaessa voidaan todeta ryhmätyön toimineen hyvin annetuissa puitteissa. Ryhmän sisällä oli reipas yrittämisen henki ja kokoontumisia saatiin helposti aikaan. Alussa olleet epäselvyydet projektin todellisesta luonteesta ja tavoitteista hiukan hidastivat myös ryhmätyön käynnistymistä, mutta jatkuvalla työnteolla saatiin aikaiseksi kelpo työpanos. Suurimpana ongelmana nähtiin interaktiivisuuden puute ohjaajan kanssa. Tässä ryhmä ja varsinkin ryhmän vetäjä olisi voinut olla aktiivisempi. Ryhmälle ei muodostunut kunnollista käsitystä ongelmasta tarpeeksi nopeasti. Aikataulussa pysyttiin kuitenkin niiltä osin kuin projekti eteni.

Ohjaajan rooli projektissa oli ennemmin taustatekijä kuin ohjaaja. Ryhmä sai päivitettyjä versioita mallista ja www-osoitteen ratkaisimeen, mutta vinkkejä itse projektin tekemiseen oli ehkä voinut olla enemmän. Selviä ohjaajan kutsumia tapaamisia ei ollut vaan ryhmä teki itsenäisesti töitä ohjaajan satunnaisesti vieraillessa.

Kun projektia tarkastelee asiakkaan kannalta, herää kysymys, oliko ryhmän tekemästä työpanoksesta todellista hyötyä. Koska yleistä ratkaisua ei onnistuttu tuottamaan, on jatkossa vaikea käyttää hyväksi ryhmän tuottamaa generaattoria ja käytettyä ratkaisinta vertailujen tekemiseen. Suurin hyöty lienee teoreettisessa mielessä ohjaajan tekemän mallin sparraaminen ja käytännössä testilaboratoriona toimiminen. Suuria oivalluksia ei syntynyt ja käytännön hyöty syntynee ainoastaan, jos projekti jatkuu ja tehtyä työtä viedään eteenpäin asiakkaan toimesta.

10. Lähteet

Helsingin Sanomat. Uutisarkisto, Torsti tietää 21.2.1999.
http://www.helsinginsanomat.fi/uutisarkisto/19990222/erik/990221er10.html

Otis Oy. Hissisanasto.
http://www.otis.com/glossary/0,1374,CLI7_RES1,00.html

Otis Oy. Hissituotteet.
http://www.otis.com/products/listing/0,1357,CLI7_PRT100_RES1,00.html

Rakennuslehti. Simuloimalla tehoa hisseihin. Rakennuslehti 12-13/2002, 28.3.2002.

Rintala, Toni. Matkustajien kohdekerrosten ennustaminen Bayes-menetelmällä liikennetilastoja käyttäen. Pro gradu -tutkielma, Helsingin Yliopisto, Tietojenkäsittelytieteen laitos 2000.

Siikonen, Marja-Liisa. Hissiliikenteen ja ohjauksen mallintaminen tietokoneella. Lisensiaattityö, Teknillinen Korkeakoulu, Tietotekniikan osasto, Matematiikan ja systeemianalyysin laitos 1989.

Siikonen, Marja-Liisa. Liikennesuunnittelupäällikkö, Kone Oyj. Tutustumiskäynti 19.4.2002, Helsinki.

Sorsa, Janne. Kaksikoristen hissien optimaalinen ryhmäohjaus. Diplomityö (valmisteilla), Teknillinen Korkeakoulu 2002.

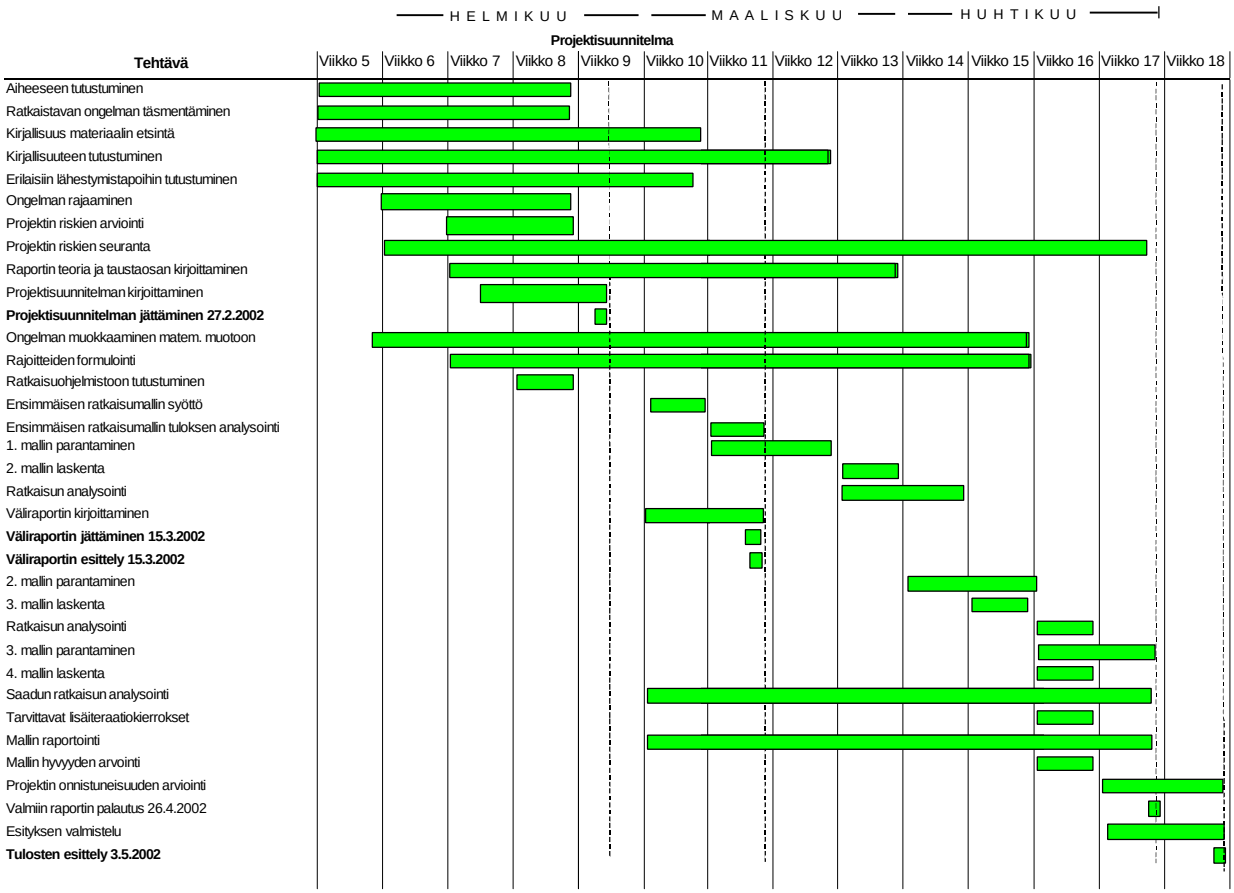
Suomen Standardoimisliitto SFS ry. Avain Euroopan markkinoille - EU:n uudet tekniset määräykset ja standardit. http://www.sfs.fi/standard/avain.html

Suomen Standardoimisliitto SFS ry. Palvelut, julkaisut: Hissidirektiivi.
http://www.sfs.fi/standard/hissi.html

Taha, Hamdy. A. Operations Research - An Introduction. PrenticeHall 1996. (6.painos).

11. Liitteet

Liite 1. Projektiaikataulu



Liite 2. Generaattorin lähdekoodi

```
package elevator;

import Java.util.*;
import Java.text.*;

public class ElevatorProblemFormat {

  public ElevatorProblemFormat() {
    super();
  }

  public String getTargetFunction() {
    int n = CallData.numberOfCalls();
    StringBuffer buffer = new StringBuffer();
    buffer.append("min:");
    for (int i = 1; i <= n; i++) {
      buffer.append("T[");
      buffer.append(i);
      buffer.append("]");
      if (i < n) {
        buffer.append(" + ");
      }
    }
    buffer.append(";");
    return buffer.toString();
  }

  public String getConstraints() {
    int n = CallData.numberOfCalls();
    int m = ElevatorData.numberOfElevators();
    StringBuffer buffer = new StringBuffer();
    // Elevator drive time to the call based on the
    // drive time to the previous call
    for (int i = 1; i <= n; i++) {
      for (int k = 1; k <= m; k++) {
        buffer.append(t(0,i,k) + t (0,k));
        buffer.append(" * ");
        buffer.append(z(0,i,k));
        buffer.append(" + ");
        for (int j = 1; j <= n; j++) {
          buffer.append(t(j,i,k) + t(j,k));
          buffer.append(" * ");
          buffer.append(z(j,i,k));
          buffer.append(" + ");
          buffer.append(y(j,i,k));
          if (j < n) {
            buffer.append(" + ");
          }
        }
        if (k < m) {
          buffer.append(" + ");
        }
      }
      buffer.append(";\n");
    }
    // Previous drive time
    for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
        for (int k = 1; k <= m; k++) {
          buffer.append(y(i,j,k));
          buffer.append(" + ");
          buffer.append(T());
          buffer.append(" - ");
```

```
            buffer.append(T());
            buffer.append(" * ");
        buffer.append(z(i,j,k));
        buffer.append(")");
        buffer.append(" >= ");
        buffer.append(T(i));
        buffer.append(";\n");
      }
    }
  }
  // before call exactly one call
  for (int j = 1; j <= n; j++) {
    for (int i = 0; i <= n; i++) {
      for (int k = 1; k <= m; k++) {
        buffer.append(z(i,j,k));
        if (k != m || i != n) {
          buffer.append(" + ");
        }
      }
    }
    buffer.append(" = 1");
    buffer.append(";\n");
  }
  // after call exactly one call
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n+1; j++) {
      for (int k = 1; k <= m; k++) {
        buffer.append(z(i,j,k));
        if (k != m || j != n+1) {
          buffer.append(" + ");
        }
      }
    }
    buffer.append(" = 1");
    buffer.append(";\n");
  }
  // before end point exactly one call for each elevator
  for (int k = 1; k <= m; k++) {
    for (int i = 0; i <= n; i++) {
      buffer.append(z(i,n+1,k));
      if (i < n) {
        buffer.append(" + ");
      }
    }
    buffer.append(" = 1");
    buffer.append(";\n");
  }
  // after start point exactly one call for each elevator
  for (int k = 1; k <= m; k++) {
    for (int j = 1; j <= n+1; j++) {
      buffer.append(z(0,j,k));
      if (j < n+1) {
        buffer.append(" + ");
      }
    }
    buffer.append(" = 1");
    buffer.append(";\n");
  }
  // elevator leaves the call as many times it has arrived in it
  for (int j = 1; j <= n; j++) {
    for (int k = 1; k <= m; k++) {
      for (int i = 0; i <= n; i++) {
        buffer.append(z(i,j,k));
        if (i < n) {
```

```
              buffer.append(" + ");
            }
        }
        buffer.append(" = ");
        for (int i = 1; i <= n+1; i++) {
          buffer.append(z(j,i,k));
          if (i < n+1) {
            buffer.append(" + ");
          }
        }
        buffer.append(";\n");
      }
    }
    // Calls are served in the order defined by the d0 when going to
the
    // direction 0
    for (int i = 0; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
        for (int k = 1; k <= m; k++) {
          buffer.append(z(i,j,k));
          buffer.append(" <= ");
          buffer.append(d0(i,j,k));
          buffer.append(" + ");
          buffer.append(r(k));
          buffer.append(";\n");
        }
      }
    }
    // Calls are served in the order defined by the d1 when going to
the
    // direction 1
    for (int i = 0; i <= n; i++) {
      for (int j = 1; j <= n; j++) {
        for (int k = 1; k <= m; k++) {
          buffer.append(z(i,j,k));
          buffer.append(" <= ");
          buffer.append(d1(i,j,k));
          buffer.append(" + 1 - ");
          buffer.append(r(k));
          buffer.append(";\n");
        }
      }
    }
    // z = {0,1}
    for (int i = 0; i <= n+1; i++) {
      for (int j = 0; j <= n+1; j++) {
        for (int k = 1; k <= m; k++) {
          if (i == j) {
            buffer.append(z(i,j,k));
            buffer.append(" = 0;\n");
          }
          else {
            buffer.append(z(i,j,k));
            buffer.append(" <= 1");
            buffer.append(";\n");
            buffer.append(z(i,j,k));
            buffer.append(" >= 0");
            buffer.append(";\n");
          }
        }
      }
    }
    // r = {0,1}
    for (int k = 1; k <= m; k++) {
```

```
        buffer.append(r(k));
        buffer.append(" <= 1");
      buffer.append(";\n");
      buffer.append(r(k));
      buffer.append(" >= 0");
      buffer.append(";\n");
    }
    // T >= 0
    for (int i = 1; i <= n; i++) {
      buffer.append(T(i));
      buffer.append(" >= 0");
      buffer.append(";\n");
    }
    // y >= 0
    for (int i = 0; i <= n+1; i++) {
      for (int j = 0; j <= n+1; j++) {
        for (int k = 1; k <= m; k++) {
          buffer.append(y(i,j,k));
          buffer.append(" >= 0");
          buffer.append(";\n");
        }
      }
    }
    return buffer.toString();
  }

  public String getIntegerVariables() {
    int n = CallData.numberOfCalls();
    int m = ElevatorData.numberOfElevators();
    StringBuffer buffer = new StringBuffer();
    buffer.append("int ");
    for (int i = 0; i <= n+1; i++) {
      for (int j = 0; j <= n+1; j++) {
        for (int k = 1; k <= m; k++) {
          buffer.append(z(i,j,k));
          buffer.append(" , ");
        }
      }
    }
    for (int k = 1; k <= m; k++) {
      buffer.append(r(k));
      if (k < m) {
        buffer.append(" , ");
      }
    }
    buffer.append(";");
    return buffer.toString();
  }

  protected String z(int i, int j, int k) {
    StringBuffer buffer = new StringBuffer();
    buffer.append("z[");
    buffer.append(i);
    buffer.append("][");
    buffer.append(j);
    buffer.append("][");
    buffer.append(k);
    buffer.append("]");
    return buffer.toString();
  }

  protected String T(int i) {
    StringBuffer buffer = new StringBuffer();
    buffer.append("T[");
```

```
      buffer.append(i);
      buffer.append("]");
  return buffer.toString();
}

protected String T() {
  return "T";
}

protected float t(int i, int j, int k) {

  float doorTime = 2.0f + 3.0f;
  if (i == j) {
    return 0.0f;
  }
  else if (i == 0) {
    CallData to = CallData.get(j);
    ElevatorData who = ElevatorData.get(k);
    float s = Math.abs(to.getHeight() - who.getHeight());
    float a = who.getAcceleration();
    float v = who.getAcceleration();
    float t1 = v/a;
    float v1 = v/2;
    float s1 = t1*v1;
    if (s - 2*s1 > 0) {
      float s2 = s - 2*s1;
      float t2 = s2/v;
      return 2*t1 + t2 + doorTime;
    }
    else {
      float s2 = s / 2;
      float t2 = (float) Math.sqrt(2 * s2 / a);
      return 2*t2 + doorTime;
    }
  }
  else if (j == 0) {
    CallData from = CallData.get(i);
    ElevatorData who = ElevatorData.get(k);
    float s = Math.abs(from.getHeight() - who.getHeight());
    float a = who.getAcceleration();
    float v = who.getAcceleration();
    float t1 = v/a;
    float v1 = v/2;
    float s1 = t1*v1;
    if (s - 2*s1 > 0) {
      float s2 = s - 2*s1;
      float t2 = s2/v;
      return 2*t1 + t2 + doorTime;
    }
    else {
      float s2 = s / 2;
      float t2 = (float) Math.sqrt(2 * s2 / a);
      return 2*t2 + doorTime;
    }
  }
  else {
    CallData from = CallData.get(i);
    CallData to = CallData.get(j);
    ElevatorData who = ElevatorData.get(k);
    float s = Math.abs(from.getHeight() - to.getHeight());
    float a = who.getAcceleration();
    float v = who.getAcceleration();
    float t1 = v/a;
    float v1 = v/2;
```

```
        float s1 = t1*v1;
        if (s - 2*s1 > 0) {
      float s2 = s - 2*s1;
      float t2 = s2/v;
      return 2*t1 + t2 + doorTime;
    }
    else {
      float s2 = s / 2;
      float t2 = (float) Math.sqrt(2 * s2 / a);
      return 2*t2 + doorTime;
    }
  }
}

protected float t(int i, int k) {
  if (i == 0) {
    return 0.0f;
  }
  else {
    return 1.0f;
  }
}

protected int d0(int i, int j, int k) {
  if (i == 0) {
    return 1;
  }
  if (j == 0) {
    return 0;
  }
  CallData from = CallData.get(i);
  CallData to = CallData.get(j);
  ElevatorData who = ElevatorData.get(k);
  int floor = who.getFloor();
  if (from.isBefore(to, floor, 0)) {
    return 1;
  }
  else {
    return 0;
  }
}

protected int d1(int i, int j, int k) {
  if (i == 0) {
    return 1;
  }
  if (j == 0) {
    return 0;
  }
  CallData from = CallData.get(i);
  CallData to = CallData.get(j);
  ElevatorData who = ElevatorData.get(k);
  int floor = who.getFloor();
  if (from.isBefore(to, floor, 1)) {
    return 1;
  }
  else {
    return 0;
  }
}

protected String r(int k) {
  StringBuffer buffer = new StringBuffer();
  buffer.append("r[");
```

```
        buffer.append(k);
        buffer.append("]");
      return buffer.toString();
    }

    protected String y(int i, int j, int k) {
      StringBuffer buffer = new StringBuffer();
      buffer.append("y[");
      buffer.append(i);
      buffer.append("][");
      buffer.append(j);
      buffer.append("][");
      buffer.append(k);
      buffer.append("]");
      return buffer.toString();
    }

}
```

Liite 3. Koneen hissisimulaattorin antama alkutiladata

```
#nF nE
20 2
#v  a  ?  F ? ? ? ? ?
6 1.2 18 1 0 0 2 1 34
6 1.2 18 6 0 0 2 1 34
#h    u d ? ? ?  ? ? ?
0.0   0 0 0 0 0 0 0 0
3.4   0 0 0 0 0 0 0 0
6.8   0 0 0 0 0 0 0 0
10.2 0 1 0 0 0 0 0 0
13.6 0 0 0 0 0 0 0 0
17.0 0 0 0 0 0 0 0 0
20.4 0 0 0 0 0 0 0 0
23.8 0 0 0 0 0 0 0 0
27.2 0 0 0 0 0 0 0 0
30.6 0 0 0 0 0 0 0 0
34.0 0 1 0 0 0 0 0 0
37.4 0 0 0 0 0 0 0 0
40.8 0 0 0 0 0 0 0 0
44.2 0 0 0 0 0 0 0 1
47.6 0 0 0 0 0 0 0 0
51.0 0 0 0 0 0 0 0 0
54.4 0 0 0 0 0 0 0 0
57.8 0 0 0 0 0 0 0 0
61.2 1 0 0 0 0 0 0 0
64.6 0 0 0 0 0 0 0 0
```

Liite 4. Generaattorin tulostus

Lainausmerkeissä olevat kommentit jälkikäteen lisättyjä

```
“Kohdefunktio”
min:T[1] + T[2] + T[3];

“Kutsun yksi odotusaika”
T[1] = 9.760952 * z[0][1][1] + 6.0 * z[1][1][1] + y[1][1][1] +
19.906925 * z[2][1][1] + y[2][1][1] + 24.5 * z[3][1][1] + y[3][1][1]
+ 10.830952 * z[0][1][2] + 6.0 * z[1][1][2] + y[1][1][2] + 19.906925
* z[2][1][2] + y[2][1][2] + 24.5 * z[3][1][2] + y[3][1][2];
“Kutsu kaksi odotusaika”
T[2] = 15.1 * z[0][2][1] + 19.906925 * z[1][2][1] + y[1][2][1] + 6.0
* z[2][2][1] + y[2][2][1] + 20.521904 * z[3][2][1] + y[3][2][1] +
11.733004 * z[0][2][2] + 19.906925 * z[1][2][2] + y[1][2][2] + 6.0 *
z[2][2][2] + y[2][2][2] + 20.521904 * z[3][2][2] + y[3][2][2];
“Kutsun kolme odotusaika”
T[3] = 19.633333 * z[0][3][1] + 24.5 * z[1][3][1] + y[1][3][1] +
20.521904 * z[2][3][1] + y[2][3][1] + 6.0 * z[3][3][1] + y[3][3][1] +
16.8 * z[0][3][2] + 24.5 * z[1][3][2] + y[1][3][2] + 20.521904 *
z[2][3][2] + y[2][3][2] + 6.0 * z[3][3][2] + y[3][3][2];

“Rajoitteet:”
“Aikaisemmat ajoajat”
y[1][1][1] + 5000 - 5000 * z[1][1][1] >= T[1];
y[1][1][2] + 5000 - 5000 * z[1][1][2] >= T[1];
y[1][2][1] + 5000 - 5000 * z[1][2][1] >= T[1];
y[1][2][2] + 5000 - 5000 * z[1][2][2] >= T[1];
y[1][3][1] + 5000 - 5000 * z[1][3][1] >= T[1];
y[1][3][2] + 5000 - 5000 * z[1][3][2] >= T[1];
y[2][1][1] + 5000 - 5000 * z[2][1][1] >= T[2];
y[2][1][2] + 5000 - 5000 * z[2][1][2] >= T[2];
y[2][2][1] + 5000 - 5000 * z[2][2][1] >= T[2];
y[2][2][2] + 5000 - 5000 * z[2][2][2] >= T[2];
y[2][3][1] + 5000 - 5000 * z[2][3][1] >= T[2];
y[2][3][2] + 5000 - 5000 * z[2][3][2] >= T[2];
y[3][1][1] + 5000 - 5000 * z[3][1][1] >= T[3];
y[3][1][2] + 5000 - 5000 * z[3][1][2] >= T[3];
y[3][2][1] + 5000 - 5000 * z[3][2][1] >= T[3];
y[3][2][2] + 5000 - 5000 * z[3][2][2] >= T[3];
y[3][3][1] + 5000 - 5000 * z[3][3][1] >= T[3];
y[3][3][2] + 5000 - 5000 * z[3][3][2] >= T[3];
“Ennen ajoa on yksi kutsu”
z[0][1][1] + z[0][1][2] + z[1][1][1] + z[1][1][2] + z[2][1][1] +
z[2][1][2] + z[3][1][1] + z[3][1][2] = 1;
z[0][2][1] + z[0][2][2] + z[1][2][1] + z[1][2][2] + z[2][2][1] +
z[2][2][2] + z[3][2][1] + z[3][2][2] = 1;
z[0][3][1] + z[0][3][2] + z[1][3][1] + z[1][3][2] + z[2][3][1] +
z[2][3][2] + z[3][3][1] + z[3][3][2] = 1;
“Ajon jälkeen on yksi kutsu”
z[1][1][1] + z[1][1][2] + z[1][2][1] + z[1][2][2] + z[1][3][1] +
z[1][3][2] + z[1][4][1] + z[1][4][2] = 1;
z[2][1][1] + z[2][1][2] + z[2][2][1] + z[2][2][2] + z[2][3][1] +
z[2][3][2] + z[2][4][1] + z[2][4][2] = 1;
z[3][1][1] + z[3][1][2] + z[3][2][1] + z[3][2][2] + z[3][3][1] +
z[3][3][2] + z[3][4][1] + z[3][4][2] = 1;
“Ennen loppua kaikilla on yksi kutsu”
z[0][4][1] + z[1][4][1] + z[2][4][1] + z[3][4][1] = 1;
z[0][4][2] + z[1][4][2] + z[2][4][2] + z[3][4][2] = 1;
“Alun jälkeen jokaisella on yksi kutsu”
z[0][1][1] + z[0][2][1] + z[0][3][1] + z[0][4][1] = 1;
z[0][1][2] + z[0][2][2] + z[0][3][2] + z[0][4][2] = 1;
```

```
    “Lähtöjä on oltava yhtä monta kuin saapumisia”
    z[0][1][1] + z[1][1][1] + z[2][1][1] + z[3][1][1] = z[1][1][1] +
z[1][2][1] + z[1][3][1] + z[1][4][1];
z[0][1][2] + z[1][1][2] + z[2][1][2] + z[3][1][2] = z[1][1][2] +
z[1][2][2] + z[1][3][2] + z[1][4][2];
z[0][2][1] + z[1][2][1] + z[2][2][1] + z[3][2][1] = z[2][1][1] +
z[2][2][1] + z[2][3][1] + z[2][4][1];
z[0][2][2] + z[1][2][2] + z[2][2][2] + z[3][2][2] = z[2][1][2] +
z[2][2][2] + z[2][3][2] + z[2][4][2];
z[0][3][1] + z[1][3][1] + z[2][3][1] + z[3][3][1] = z[3][1][1] +
z[3][2][1] + z[3][3][1] + z[3][4][1];
z[0][3][2] + z[1][3][2] + z[2][3][2] + z[3][3][2] = z[3][1][2] +
z[3][2][2] + z[3][3][2] + z[3][4][2];
“Suunta alas”
z[0][1][1] <= 1 + r[1];
z[0][1][2] <= 1 + r[2];
z[0][2][1] <= 1 + r[1];
z[0][2][2] <= 1 + r[2];
z[0][3][1] <= 1 + r[1];
z[0][3][2] <= 1 + r[2];
z[0][4][1] <= 1 + r[1];
z[0][4][2] <= 1 + r[2];
z[1][1][1] <= 0 + r[1];
z[1][1][2] <= 0 + r[2];
z[1][2][1] <= 0 + r[1];
z[1][2][2] <= 1 + r[2];
z[1][3][1] <= 0 + r[1];
z[1][3][2] <= 1 + r[2];
z[1][4][1] <= 1 + r[1];
z[1][4][2] <= 1 + r[2];
z[2][1][1] <= 1 + r[1];
z[2][1][2] <= 0 + r[2];
z[2][2][1] <= 0 + r[1];
z[2][2][2] <= 0 + r[2];
z[2][3][1] <= 0 + r[1];
z[2][3][2] <= 0 + r[2];
z[2][4][1] <= 1 + r[1];
z[2][4][2] <= 1 + r[2];
z[3][1][1] <= 1 + r[1];
z[3][1][2] <= 0 + r[2];
z[3][2][1] <= 1 + r[1];
z[3][2][2] <= 1 + r[2];
z[3][3][1] <= 0 + r[1];
z[3][3][2] <= 0 + r[2];
z[3][4][1] <= 1 + r[1];
z[3][4][2] <= 1 + r[2];
“Suunta ylös”
z[0][1][1] <= 1 + 1 - r[1];
z[0][1][2] <= 1 + 1 - r[2];
z[0][2][1] <= 1 + 1 - r[1];
z[0][2][2] <= 1 + 1 - r[2];
z[0][3][1] <= 1 + 1 - r[1];
z[0][3][2] <= 1 + 1 - r[2];
z[0][4][1] <= 1 + 1 - r[1];
z[0][4][2] <= 1 + 1 - r[2];
z[1][1][1] <= 0 + 1 - r[1];
z[1][1][2] <= 0 + 1 - r[2];
z[1][2][1] <= 0 + 1 - r[1];
z[1][2][2] <= 0 + 1 - r[2];
z[1][3][1] <= 0 + 1 - r[1];
z[1][3][2] <= 0 + 1 - r[2];
z[1][4][1] <= 1 + 1 - r[1];
z[1][4][2] <= 1 + 1 - r[2];
z[2][1][1] <= 1 + 1 - r[1];
```

```
    z[2][1][2] <= 1 + 1 - r[2];
    z[2][2][1] <= 0 + 1 - r[1];
z[2][2][2] <= 0 + 1 - r[2];
z[2][3][1] <= 0 + 1 - r[1];
z[2][3][2] <= 0 + 1 - r[2];
z[2][4][1] <= 1 + 1 - r[1];
z[2][4][2] <= 1 + 1 - r[2];
z[3][1][1] <= 1 + 1 - r[1];
z[3][1][2] <= 1 + 1 - r[2];
z[3][2][1] <= 1 + 1 - r[1];
z[3][2][2] <= 1 + 1 - r[2];
z[3][3][1] <= 0 + 1 - r[1];
z[3][3][2] <= 0 + 1 - r[2];
z[3][4][1] <= 1 + 1 - r[1];
z[3][4][2] <= 1 + 1 - r[2];
“Muuttujien kokonaislukuehdot”
z[0][1][1] <= 1;
z[0][1][1] >= 0;
z[0][1][2] <= 1;
z[0][1][2] >= 0;
z[0][2][1] <= 1;
z[0][2][1] >= 0;
z[0][2][2] <= 1;
z[0][2][2] >= 0;
z[0][3][1] <= 1;
z[0][3][1] >= 0;
z[0][3][2] <= 1;
z[0][3][2] >= 0;
z[0][4][1] <= 1;
z[0][4][1] >= 0;
z[0][4][2] <= 1;
z[0][4][2] >= 0;
z[1][1][1] = 0;
z[1][1][2] = 0;
z[1][2][1] <= 1;
z[1][2][1] >= 0;
z[1][2][2] <= 1;
z[1][2][2] >= 0;
z[1][3][1] <= 1;
z[1][3][1] >= 0;
z[1][3][2] <= 1;
z[1][3][2] >= 0;
z[1][4][1] <= 1;
z[1][4][1] >= 0;
z[1][4][2] <= 1;
z[1][4][2] >= 0;
z[2][1][1] <= 1;
z[2][1][1] >= 0;
z[2][1][2] <= 1;
z[2][1][2] >= 0;
z[2][2][1] = 0;
z[2][2][2] = 0;
z[2][3][1] <= 1;
z[2][3][1] >= 0;
z[2][3][2] <= 1;
z[2][3][2] >= 0;
z[2][4][1] <= 1;
z[2][4][1] >= 0;
z[2][4][2] <= 1;
z[2][4][2] >= 0;
z[3][1][1] <= 1;
z[3][1][1] >= 0;
z[3][1][2] <= 1;
z[3][1][2] >= 0;
```

```
    z[3][2][1] <= 1;
    z[3][2][1] >= 0;
z[3][2][2] <= 1;
z[3][2][2] >= 0;
z[3][3][1] = 0;
z[3][3][2] = 0;
z[3][4][1] <= 1;
z[3][4][1] >= 0;
z[3][4][2] <= 1;
z[3][4][2] >= 0;
r[1] <= 1;
r[1] >= 0;
r[2] <= 1;
r[2] >= 0;
T[1] >= 0;
T[2] >= 0;
T[3] >= 0;
y[1][1][1] >= 0;
y[1][1][2] >= 0;
y[1][2][1] >= 0;
y[1][2][2] >= 0;
y[1][3][1] >= 0;
y[1][3][2] >= 0;
y[2][1][1] >= 0;
y[2][1][2] >= 0;
y[2][2][1] >= 0;
y[2][2][2] >= 0;
y[2][3][1] >= 0;
y[2][3][2] >= 0;
y[3][1][1] >= 0;
y[3][1][2] >= 0;
y[3][2][1] >= 0;
y[3][2][2] >= 0;
y[3][3][1] >= 0;
y[3][3][2] >= 0;

“Käytetyt muuttujat”
int
z[0][1][1] , z[0][1][2] , z[0][2][1] , z[0][2][2] , z[0][3][1] ,
z[0][3][2] , z[0][4][1] , z[0][4][2] , z[1][1][1] , z[1][1][2] ,
z[1][2][1] , z[1][2][2] , z[1][3][1] , z[1][3][2] , z[1][4][1] ,
z[1][4][2] , z[2][1][1] , z[2][1][2] , z[2][2][1] , z[2][2][2] ,
z[2][3][1] , z[2][3][2] , z[2][4][1] , z[2][4][2] , z[3][1][1] ,
z[3][1][2] , z[3][2][1] , z[3][2][2] , z[3][3][1] , z[3][3][2] ,
z[3][4][1] , z[3][4][2] , r[1] , r[2];
```

Liite 5. Generaattorin alkutilatulostus

1:Elevator at 1 floor, max speed 6.0 and acceleration 1.2
2:Elevator at 6 floor, max speed 6.0 and acceleration 1.2
0: height 0.0
1: height 3.4
2: height 6.8
3: height 10.2 v
4: height 13.6
5: height 17.0
6: height 20.4
7: height 23.8
8: height 27.2
9: height 30.6
10: height 34.0 v
11: height 37.4
12: height 40.8
13: height 44.2
14: height 47.6
15: height 51.0
16: height 54.4
17: height 57.8
18: height 61.2 ^
19: height 64.6
1:F3 down
2:F10 down
3:F18 up

Liite 6. LPsolven ratkaisutulostus

Value of objective function:	61.20216179
r[1]	0
r[2]	0
y[1][1][2]	0
y[1][1][1]	0
y[3][1][2]	0
y[2][1][1]	0
y[2][1][2]	0
y[3][1][1]	0
z[1][3][2]	0
z[0][3][1]	1
z[0][3][2]	0
z[1][3][1]	0
z[3][3][2]	0
z[2][3][1]	0
z[2][3][2]	0
z[3][3][1]	0
z[1][1][2]	0
z[0][1][1]	0
z[0][1][2]	1
z[1][1][1]	0
z[3][1][2]	0
z[2][1][1]	0
z[2][1][2]	0
z[3][1][1]	0
y[1][2][2]	10.831
y[1][2][1]	0
y[3][2][2]	0
y[2][2][1]	0
y[2][2][2]	0
y[3][2][1]	0
z[0][4][1]	0
z[1][4][2]	0
z[0][4][2]	0
z[1][4][1]	0
z[3][4][2]	0
z[2][4][1]	0
z[3][4][1]	1
z[2][4][2]	1
y[1][3][2]	0
y[1][3][1]	0
y[3][3][2]	0
y[2][3][1]	0
y[2][3][2]	0
y[3][3][1]	0
T[1]	10.831
T[2]	30.738

T[3]	19.633
z[1][2][2]	1
z[0][2][1]	0
z[0][2][2]	0
z[1][2][1]	0
z[3][2][2]	0
z[2][2][1]	0
z[2][2][2]	0
z[3][2][1]	0