Mat-2.4177 Operaatiotutkimuksen projektityöseminaari

Vapaapäivien optimointi
Loppuraportti, 14.5. 2014

Asiakas:
Computational Intelligence Oy

Projektiryhmä:
Teemu Kinnunen (projektipäällikkö)
Ilari Vähä-Pietilä
Lotta Martikainen
Santtu Klemettilä

Sisällysluettelo

1 Johdanto

Tämä projektityö tehtiin kevään 2014 aikana Mat-2.4177 Operaatiotutkimuksen projektityöseminaari L -kurssilla Aalto-yliopistossa. Projektityön tavoitteena oli löytää tarkoituksenmukainen tapa vapaapäivien optimointiin annettujen kovien ja pehmeiden rajoitteiden vallitessa. Toteutuksesta vastasi neljän hengen opiskelijaryhmä.

Projektityön aiheen asettajana toimi CI Computational Intelligence Oy, joka on yksi Suomen johtavista laskennallisen älykkyyden ja optimoinnin asiantuntijayrityksistä. Yritys on kehittänyt ratkaisuja muun muassa työvuorojen optimoinnin saralla ja toivoi projektityön tuovan yrityksen ratkaisujen kehitystyöhön tuoreita näkökulmia ja oivalluksia. Yrityksen puolesta projektiryhmän yhteyshenkilönä toimi toimitusjohtaja Cimmo Nurmi.

Projektityön tuotoksena syntyi CPLEXissä toteutettu optimointityökalu, jonka avulla voidaan ratkaista vapaapäivien ajoittuminen valitulle työntekijäjoukolle. Ratkaisu täyttää sille asetetut kovat rajoitteet kuten päivittäisten työntekijöiden minimimäärän ja työjakson maksimipituuden. Ratkaisu myös minimoi pehmeistä rajoitteista kuten yksittäisistä työ- ja vapaapäivistä aiheutuvat sakot. Itse ratkaisun lisäksi testasimme mallia erilaisilla rajoitteiden arvoilla.

Tämän loppuraportin luvussa 2 käydään tarkemmin läpi työn tausta ja tavoitteet. Luku 3 tiivistää projektin alkuvaiheessa toteutetun kirjallisuuskatsauksen löydökset. Luvussa 4 käydään läpi tutkimusongelman formulointi, ja luvussa 5 esitellään kehitetty ratkaisumenetelmä. Projektityön tuloksia analysoidaan luvussa 6 ja johtopäätelmät esitetään luvussa 7. Loppuraportti päättyy ryhmän itsearviointiin luvussa 8.

2 Tausta ja tavoitteet

Työvuorojen optimoinnin menetelmät ovat kehittyneet merkittävästi viimeisten vuosikymmenten aikana. Kenties merkittävimmät syyt tähän ovat tietokoneiden laskentatehon kasvaminen sekä matemaattisten menetelmien huomattava kehittyminen, minkä ansiosta tietokoneet pystyvät nykyään ratkaisemaan laajojakin tehtäviä kohtuullisessa ajassa. Toisaalta työvuorosuunnittelun vaatimukset ja rajoitteet monimutkaistuvat jatkuvasti, jolloin manuaalinen suunnittelutyö vie yhä enemmän aikaa. (Kyngäs, 2011) Työvoiman hallinnan optimointi mahdollistaa yrityksissä keskimäärin noin 10 prosentin tuottavuuden lisääntymisen. Optimoitujen ratkaisujen on todettu myös lisäävän työtyytyväisyyttä ja vähentävän näin sairauspoissaoloja. (Computational Intelligence, 2014)

Tutkimusta on tehty vastaamaan erityisesti bussi-, juna- ja lentoliikenteen, sairaaloiden, tehtaiden ja terveydenhuollon tarpeisiin. Menetelminä käytetään muun muassa matemaattisia optimointialgoritmeja, heuristisia menetelmiä, geneettisiä algoritmeja, monitavoiteoptimointia,

rajoiteoptimointia ja erilaisia hybridialgoritmeja. Valittava menetelmä riippuu luonnollisesti pitkälti ratkaistavasta ongelmasta. (Ernst et al, 2004; Naveh et al, 2007; Tower, 2007; Vander Berghe, 2002)
CI Computational Intelligence esitti projektityöhön optimoinnin kohteeksi aluksi 30 työntekijän vapaapäivät neljälle viikolle. Alla on kuvattu annetut kovat ja pehmeät rajoitteet sekä näihin liittyvät CI Computational Intelligencelta saadun alkuoletukset.

Toteutettavat kovat rajoitteet ovat:

- Työntekijöitä pitää olla töissä jokaisena päivänä vähintään määritellyn päiväkohtaisen vähimmäismäärän verran. Alkuoletuksena ma-to=22, pe=25, la=15 ja su=12.
- Ylimääräisten työntekijöiden määrä per päivä pitää jakaa tasaisesti. Aluksi tämä tarkoittaa käytännössä sitä, että 10 päivänä on yksi ylimääräinen työntekijä ja 18 päivänä ei yhtään.
- Työntekijät eivät saa työskennellä yli päivää yhtäjaksoisesti, alkuoletuksena 6 päivää
- Jotkut työntekijät eivät työskentele viikonloppuisin, alkuoletuksena 6 työntekijää

Tavoiteltavat pehmeät rajoitteet ovat:

- Yksittäisiä vapaapäiviä pitää välttää (1 sakkopiste/rikkomus)
- Yksittäisiä työpäiviä pitää välttää (2 sakkopistettä/rikkomus)
- Liian pitkiä vapaita pitää välttää (3 sakkopistettä jokaisesta vapaapäivän jälkeisestä vapaapäivästä, alkuoletuksena).
- Vapaiden viikonloppujen määrää tulee tasapainottaa, tavoitteena 1–2 vapaata viikonloppua jokaiselle työntekijälle (1 sakkopiste, jos 0 tai 3 vapaata viikonloppua, ja 2 sakkopistettä, jos 4 vapaata viikonloppua).

Projektin tavoitteena oli siis kehittää menetelmä, jolla saavutetaan vapaapäivien määrittämiseen käypä ja hyvä ratkaisu kohtuullisella laskentamäärällä. Pehmeiden rajoitteiden sakot ovat ennalta määrättyjä, joten minkäänlaista arvofunktioanalyysiä ratkaisussa ei tarvitse suorittaa. Itse menetelmän lisäksi tarkoituksena on myös analysoida mallin muokattavuutta ja toimivuutta erilaisilla rajoitteiden arvoilla.

Yllä kuvattujen rajoitteiden lisäksi ongelmaa laajennettiin kevään aikana vaiheittain. Lisätavoitteiksi määritettiin työntekijämäärän muuttaminen erikokoiseksi sekä työntekijöiden osaamisryhmien huomioiminen. Tämän lisäksi malliin lisättiin seuraavat rajoitteet

- Työntekijät joiden pitää olla samoissa työvuoroissa esimerkiksi yhteiskuljetusten vuoksi
- Yksittäisten vapaapäivien mahdollisimman tasainen jakautuminen työntekijöille
- Työntekijöiden vapaapäivätoiveiden huomioiminen
- Vaadittujen päivittäisten osaamisryhmien täyttyminen töissä olevilla työntekijöillä
- Yövuoroihin liittyvät rajoitteet

Lisäksi on huomioitava, että reaalimaailmassa tapahtuu muutoksia, jotka voivat vaatia muuttamaan aikatauluja lyhyelläkin varoitusajalla. Näin tapahtuu esimerkiksi työntekijän sairastuessa, laitteen rikkoutuessa tai työntekijän myöhästyessä. Tämän vuoksi aikatauluja optimoitaessa on otettava huomioon myös epävarmuustekijät, koska muuten alun perin kelpo suunnitelma voi muuttua epäsopivaksi pienen muutoksen osuessa kohdalle. Tavoitteena on, että osana ongelman ratkaisua vastaamme myös haasteisiin koskien epävarmuustekijöiltä suojautumista (robustness) sekä häiriöstä palautumista.

3 Kirjallisuuskatsaus

Kirjallisuuskatsausta varten käytiin läpi työvuorojen optimointiin sekä käytettyihin menetelmiin liittyviä julkaisuja ja tutkimuksia. Huomattiin, että tutkimusta on tehty vastaamaan erityisesti bussi-, juna- ja lentoliikenteen, sairaaloiden, tehtaiden ja terveydenhuollon tarpeita. Muita sovelluskohteita ovat esimerkiksi puhelinkeskukset ja turvallisuuspalveut.

Ensimmäisessä alaluvussa 3.1. kerrotaan, miten työvuorojen optimointiongelmia voidaan jaotella. Alaluvussa 3.2. esitellään sairaanhoitajien työvuorojen optimoinnissa tyypillisesti huomioitavia asioita. Alaluvussa 3.3 arvioidaan, miten epävarmuustekijät vaikuttavat optimoituihin ratkaisuihin ja pohditaan, miten epävarmuustekijöiltä voidaan suojautua. Alaluvussa 3.4. käydään lyhyesti läpi muutama esimerkkitapaus, joissa rajoiteohjelmointia hyödynnetään optimointiongelmien ratkomisessa. Rajoiteohjelmoinnista lukeminen auttoi ryhmäämme ymmärtämään paremmin työvuorojen ohjelmointiratkaisuihin vaikuttavia rajoitteita, vaikkei oma ratkaisumme perustu kyseiseen menetelmään.

Lähestyimme työvuorojen optimoinnin monimuotoista kenttää tarkastelemalla muutamia erityyppisiä esimerkkiongelmia ja – ratkaisuja. Luvussa 3.5. esitellään ANROM -tapaus, joka on hyvä esimerkki työvuorojen optimointitehtävästä sairaalamaailmassa. Luvussa 3.6. esitellään lyhyesti muutama muu mielenkiintoiseksi havaittu esimerkkitapaus.

3.1 Työvuorojen optimointiongelmien tyypit

Ernst et al. (2004 (1)) ovat koonneet yli 700 työvoiman optimointiin liittyvän artikkelin pohjalta yleiskatsauksen tutkimusalaan. Aiheeseen liittyvä tutkimus toimialoittain voidaan jaotella karkeasti alla olevan listauksen mukaisesti (kuva 1).

Categorisation of papers by application.

Application	Papers	Application	Papers
Buses	129	Civic Services and Utilities	22
Nurse Scheduling	103	Venue Management	19
Airlines	99	Protection and Emergency Services	16
Railways	37	Other Applications	14
Call Centres	37	Transportation Systems	12
General	33	Hospitality and Tourism	7
Manufacturing	29	Financial Services	6
Mass Transit	28	Sales	3
Health Care Systems	23		

Kuva 1: Tutkimusten jaottelu toimialan mukaan (Ernst et al. 2004 (1))

Työvuorojen optimointiongelmia on puolestaan kategorisoitu kuvan 2 listauksen mukaisesti.

Categorisation of papers by rostering classification.

Classification	Papers	Classification	Papers
Crew Scheduling	219	Task Based Demand	47
Tour Scheduling	185	Demand Modelling	40
Flexible Demand	107	Task Assignment	32
Workforce Planning	99	Shift Assignment	24
Crew Rostering	76	Disruption Management	16
Shift Scheduling	64	Other Classifications	12
Cyclic Roster	62	Stint Based Roster	9
Days Off Scheduling	61	Roster Assignment	6
Shift Demand	55		

Kuva 2: Tutkimusten jaottelu optimointiongelmien mukaan (Ernst et al 2004 (1))

Ernst et al. (2004 (2)) erittelevät myös työvuoro-optimoinnin tehtävätyyppejä. Julkaisussa määritellään työvuoro-optimoinnin eri osa-alueita, joita tulee miettiä alettaessa optimoimaan mitä tahansa työvuorojärjestelmää. Näitä osa-alueita ovat esimerkiksi kysynnän, vapaapäivien, itse työaikojen, tuotantolinjojen, työtehtävien sekä työntekijöiden optimointimoduulit. Yhteenvetona voidaan mainita kolme kohtaa jotka erottavat eri optimointitehtävät toisistaan:

1. Kuinka vapaapäivien ajoitus, tuotantolinjan rakentaminen sekä työtehtävät on integroitu systeemiin.
2. Jotkin tehtävät eivät vaadi esimerkiksi ollenkaan työtehtävien optimointia, kun taas toiset vaativat esimerkiksi kysynnän mallintamista.
3. Kysynnän sekä tuotantolinjan osien erilaisuus aiheuttaa eroja. Esimerkiksi lentoliikenteen ja puhelinkeskuksen työvuorojen optimoinnissa työtehtävillä on huomattavia eroja.

Tehtävien ratkaisemisessa käytetään usein hyödyksi muun muassa matemaattisia optimointialgoritmeja, heuristisia menetelmiä, geneettisiä algoritmeja, monitavoiteoptimointia, rajoiteoptimointia ja erilaisia hybridialgoritmeja. Matemaattisen ohjelmoinnin lähestymistavat eivät tähän mennessä ole olleet dominoivia sairaanhoitajien työvuorojen optimoinnissa. Sen sijaan monia heuristisia lähestymistapoja on kehitetty, jotta monimutkaiset ihmisten preferensseistä muodostuvat optimointihaasteet on saatu ratkaistua. Valittava menetelmä riippuu luonnollisesti pitkälti ratkaistavasta ongelmasta. (Ernst et al,

2004; Naveh et al, 2007; Tower, 2007; Vander Berghe, 2002) Alla oleva listaus (kuva 3) koostaa työvuorojen optimoimiseen liittyvän tutkimuksen käyttämät ratkaisumenetelmät ja antaa osviittaa menetelmien yleisyydestä.

Categorisation of papers by solution method.

Method	Papers	Method	Papers
Branch-and-Bound	14	Lagrangean Relaxation	32
Branch-and-Cut	9	Linear Programming	35
Branch-and-Price	30	Matching	36
Column Generation	48	Mathematical Programming	27
Constraint Logic Programming	46	Network Flow	38
Constructive Heuristic	133	Other Meta-Heuristic	11
Dynamic Programming	17	Other Methods	35
Enumeration	13	Queueing Theory	32
Evolution	4	Set Covering	58
Expert Systems	15	Set Partitioning	72
Genetic Algorithms	28	Simple Local Search	39
Goal Programming	19	Simulated Annealing	20
Integer Programming	139	Simulation	31
Iterated Randomised Construction	5	Tabu Search	16

Kuva 3: Ratkaisumenetelmät työvuorojen optimoinnissa (Ernst et al 2004 (1))

3.2 Sairaanhoitajien työvuorojen optimointi

Cheang et al. (2002) koostaa kirjallisuuskatsauksen keinoin sairaanhoitajien työn suunnittelun optimoimisen malleja ja menetelmiä. Mallintaminen perustuu usein työvuorojen ja vapaiden vuorojen erotteluun. Toisaalta työvuorot voidaan myös jakaa aamu-, iltapäivä ja iltavuoroihin. Joissain mallinnuksissa vapaapäivää on merkattu numerolla 0, ja vastaavasti kaikille mahdollisille työvuoroille asetetaan oma numeronsa. Osa mallinnuksista toimii binäärioptimoinnin logiikalla, jossa 1 vastaa tietyn vuoron tekemistä ja 0 muita vaihtoehtoja.

Optimointiratkaisut voivat johtaa optimaaliseen tulokseen, mutta monessa tapauksessa on todettu haastavaksi toteuttaa kaikki rajoitteet tyydyttävästi. Koska sairaalat ovat dynaamisia ympäristöjä, yksinkertaisemmat tai joustaviin algoritmeihin perustuvat ratkaisut on todettu arvokkaammiksi. Hyvä menetelmä riippuu ratkaistavasta ongelmasta, mutta yleisesti metaheuristiset ratkaisut ovat olleet suhteellisen helposti sovitettavissa sairaanhoitajien työvuorojen järjestelyyn liittyviin ongelmiin. Myös erilaiset yhdistelmät systemaattisista puurakenteista ja rajoiteoptimoinnista ovat olleet suhteellisen menestyksellisiä strategioita. Myös hybridialgoritmeista, jotka yhdistelevät perinteisen optimoinnin metodeja tekoälyn tekniikoihin, on saatu lupaavia tuloksia.

3.3 Epävarmuustekijöiltä suojautuminen

Epävarmuustekijöiltä suojautuminen (robustness) ei ole kovin selvästi määriteltävä käsite, minkä vuoksi yksikäsitteistä optimia on jossain määrin mahdoton löytää. Eräs lähestymistapa (Lan, 2003) on etsiä käypä ratkaisu, joka maksimoi resurssien odotteluajan, jolloin myöhästymisten mahdollisuus pienenee. Resurssien odotteluajan kasvamisesta tuleva varmuus ei toisaalta parane lineaarisesti odotteluajan kasvaessa, eikä kaikki odottelu ole samanarvoista. Eri prosessien kestoissa on myös eroavaisuuksia

varianssissa, minkä vuoksi stokastinen tarkastelu olisi tarpeen, mutta aikataulutusongelmien yleisen suuruuden vuoksi se nähdään usein raskaana suhteessa hyötyyn.

Häiriöistä palautumista tarkastellessa optimin määrittely on myös hankala tehtävä. Kohl et al. (2007) esittää, että tavoitteiksi voidaan asettaa asiakkaiden odotusten täyttäminen (lentoliikenne-esimerkissä toimittamalla heidät ja heidän matkatavaransa mahdollisimman ajoissa ja sovitulla tavalla määränpäähänsä), kustannusten minimointi sekä aikatauluun palaaminen mahdollisimman nopeasti. Asiakaslupauksen täyttäminen ja kustannukset voidaan selvähkösti, vaikkei kyllä helposti, suhteuttaa toisiinsa, mutta aikatauluun palaamisen venymisen kustannuksia on vaikeampi arvioida. Palautumisaika ei suoranaisesti johdu kustannuksiksi, mutta arvoa voidaan antaa häiriönhallinnan helpottumiselle ja aikataulunmukaisuudelle, ja liian pitkä aika voi olla jopa mahdotonta toteuttaa johtuen esimerkiksi erilaisista sopimuksista, jotka asettavat vaatimuksia aikataululle.

Tarkastelemassamme tilanteessa, työvuorojen optimoinnissa, häiriöistä palautuminen ilmentyy sairaus- ja muiden poissaolojen paikkaamisena. Sairaustapauksien heikon ennustettavuuden ja työvuorojen epäketjuuntuvuuden vuoksi häiriöistä palautumisen tarkastelu ei juuri muotoudu järkeväksi ongelma-alueeksi. Eräs tarkastelumahdollisuus on korvaavan työvoiman kutsumisen mahdollistaminen jakamalla työntekijöiden vapaapäivät siten, että jokaiselle päivälle on tarvittaessa henkilöitä kutsuttavana siten, etteivät kovat rajoitteet rikkoonnu.

Erilaisilla robustisuusmäärittelyillä ja –tavoitteilla päästään toki erilaisiin tuloksiin. Zuo et al. (2009) esittelee menetelmän, jolla on saatu suotuisia tuloksia. Suotuisten häiriöiden sattuessa riskejä ei-huomioivan mallin ratkaisu on tietysti parempi, mutta keskimääräiset kustannukset ovat heidän robustisuuteen pyrkivällä menetelmällään yleensä paremmat erityisesti, kun epävarmuuden aste nousee korkeammaksi.

Tower (2007) käsittelee terveydenhuollon työaikaoptimointiongelmien ratkaisumenetelmien robustiutta käsittelevässä julkaisussaan erilaisia malleja ja tarkastelee niiden toimintaa erityyppisissä tehtävissä. Tehtäville on käytetty useita eri ratkaisumenetelmiä, kuten matemaattisia optimointialgoritmeja, heuristisia menetelmiä ja geneettisiä algoritmeja. Tutkimuksen päätarkoituksena on esittää uusia malleja, joilla saadaan entistä parempia työvoiman optimointituloksia.

Työvuorojen optimoinnissa otetaan huomioon useita muita tekijöitä pelkän työvoimakustannuksen lisäksi. Näitä ovat esimerkiksi yksittäisen hoitajan työmäärä, peräkkäiset samat työvuorot, peräkkäiset työvuorot ja työpäivät sekä hoitajan taidot ja toiveet. Tutkiessaan eri malleja ja niiden robustiutta Tower huomasi, että suurimmat kustannukset aiheutuivat häiriöistä työvoiman saamisessa. Vaikkei näitä häiriöitä voida eliminoida, työvuoroja voidaan optimoida toimimaan paremmin häiriötapausten sattuessa. Tällöin esimerkiksi voidaan laittaa töihin ylimääräisiä työntekijöitä tai asettaa työntekijöitä hälytysvalmiuteen.

Puhuttaessa aikatauluista on usein otettava huomioon niissä tapahtuvat mahdolliset muutokset: työntekijät voivat sairastua, laitteet voivat rikkoutua, joku tai jokin voi olla myöhässä tai jotain muuta odottamatonta voi sattua. Tämän vuoksi aikatauluja optimoitaessa on otettava huomioon myös

epävarmuustekijät, koska ilman tätä näkökulmaa alun perin täydellinen suunnitelma voi muuttua erittäinkin huonoksi pienen muutoksen osuessa kohdalle.

Epävarmuustekijöitä tarkastellaan kirjallisuudessa hyvin usein soveltaen optimointia lentoliikenteen suunnitteluun, koska lentoliikenteessä suuret kustannukset allokoituvat selvästi erotettaviin yksiköihin, joiden liikuttaminen ja käyttö on kuitenkin toisaalta yksinkertaista. Tämän vuoksi monelle seikalle tässä lähestymistavassa ei suoranaisesti löydy selviä analogioita työvuorojen suunnittelussa.

Eggenberg (2009) jakaa epävarmuustekijöiden käsittelyn epävarmuuksilta suojautumiseen ja häiriöistä palautumiseen. Suojautumisella tarkoitetaan aikataulun suunnittelua siten, että pienet muutokset, esimerkiksi joidenkin prosessien kestossa, eivät vaikuta muuhun aikatauluun tai muihin resursseihin. Muutoksista palautuminen taasen tarkoittaa häiriötilasta suunniteltuun aikatauluun palaamista mahdollisimman pienin kustannuksin ja suunnitelmapoikkeamin sekä mahdollisimman lyhyessä ajassa.

3.4 Optimointiongelmien ratkominen rajoiteohjelmointia hyödyntäen

Naveh ja muut (2007) kehittivät rajoiteohjelmointia (constraint programming) hyödyntävän menetelmän edesauttamaan tehokasta ja laadukasta työvoiman optimointia. Menetelmässä rajoitteena ovat esimerkiksi työntekijän rooli, taitotaso, maantieteellinen sijainti, kielitaito ja mahdollisuus lisäkouluttamiseen. Rajoitteita voidaan asettaa sekä yksilö, että tiimitasolla. Teknologia kehitettiin IBM Global Servicelle, jossa pitää ohjata IBM:n konsultit ja palvelupuolen ihmiset tiimeihin ja asiakasprojekteihin eri puolilla maailmaa.

Rajoiteohjelmoinnissa työstetään abstrakteja ongelmia, jossa on monia toisiinsa kytkeytyneitä muuttujia. CSP-ohjelmointi (constraint satisfaction problem) etsii ratkaisuja, jossa rajoitteet täyttyvät. CSP on tyydyttävä, jos ainakin yksi ratkaisu on löydettävissä. Algoritmien suhteen on käytetty maintain-arc-consistency (MAC) – algoritmia, jossa iteratiivisesti pienennetään verkon kokoa ja asetetaan muuttujalle tämä pienemmän verkon arvo. Rajoitteiden esittämiseen ehdotetaan esimerkiksi Numericaa ja Optimization Programming – kieltä.

Rajoiteohjelmoinnissa määritetään erilaisia rajoitteita mahdollisten ratkaisujen joukolle, ja näitä etsitään laskennallisin menetelmin. Esimerkiksi Topaloglu ja Ozkarahan (2011) ovat kehittäneet rajoiteohjelmointiin perustuvan ratkaisun terveydenhuollon työntekijöiden aikataulujen optimoimiseen. Malli huomioi päivystysvuorot, vapaapäivät, lepoajat sekä työvuorot. Ratkaisu perustuu kahteen osaan. Ensinnäkin sarakkeiden generointiin perustuvan mallin (column generation model) avulla etsitään sellaisia ratkaisuja, joissa palvelutaso päivisin ja öisin täyttäisi mahdollisimman tarkoituksenmukaisesti tavoitetason. Rajoiteohjelmoinnin avulla etsitään puolestaan joukko toteutettavissa olevia aikatauluja.

Rajoitteiden avulla määritetään esimerkiksi viikoittainen maksimityöaika, peräkkäisten työpäivien maksimimäärä vapaavuorojen minimimäärä, päivystysten maksimimäärä ja -aika. Rajoitteet ovat pääosin kovia lukuun ottamatta työntekijöiden tavoitetason toteutumista kussakin vuorossa.

Bourdais et al. (2003) ovat kehittäneet HIBISCUS -nimisen tavan formuloida ja ratkaista terveydenhuollon työvuorojen optimointitehtäviä rajoiteohjelmoinnin ja heuristisen etsintästrategian avulla. Mallissa

käytetään pääosin kovia rajoitteita takaamaan ratkaisun laatu ja reiluus eri työntekijöitä kohtaan. Lisäksi hakemiseen käytetty heuristiikka mahdollistaa toiveisiin liittyvien kannustimien lisäämisen malliin.

Rajoitteita määritetään tässä ratkaisussa esimerkiksi erilaisten summa-, jakauma- ja polkusääntöjen perusteella. Näitä sovelletaan esimerkiksi kunkin ajanhetken työntekijöiden määrän asettamiseen, viikonloppuvuorojen tasaiseen jakautumiseen ja ylipäätään ratkaisujen järkevöittämiseen esimerkiksi peräkkäisten vuorojen suhteen.

3.5 Esimerkkitapaus sairaalamaailmasta: Advanced Nurse Rostering Model

Vanden Berghe (2002) käsittelee väitöskirjassaan työvuorojen optimointia belgialaisissa sairaaloissa. Ongelma sisältää paljon rajoituksia ja on erittäin vaikea ratkaistavaksi. Ratkaisemista varten on kehitetty malli nimeltään ANROM (Advanced Nurse ROstering Model).
ANROM sisältää seuraavat kovat rajoitteet:

- Henkilölle ei voida määrätä kahta samanaikaista työvuoroa
- Henkilöä ei voida asettaa tekemään työvuoroa, johon hänen pätevyytensä ei riitä
- Minimi töissä olevien henkilöiden määrälle eri ajanhetkinä

Lisäksi ANROM sisältää 28 pehmeää rajoitetta, jotka mahdollistavat ratkaisujen säätämisen monipuolisesti. Esimerkkejä pehmeistä rajoitteista ovat:

- Minimiaika kahden työvuoron välillä
- Työvuorojen maksimimäärä suunnittelujaksossa
- Minimi- ja maksimimäärä peräkkäisiä työpäiviä
- Minimi- ja maksimimäärä peräkkäisiä vapaapäiviä
- Minimi- ja maksimimäärä työtunteja suunnittelujaksossa
- Viikonloppuvapaassa koko viikonlopun pitää olla vapaa
- Yövuoron jälkeen 2 vapaapäivää
- Vuorojen asettaminen peräkkäin tiettyjen syklisyyssääntöjen mukaan
- Laskurit, jotka tasapainottavat työmäärän suunnittelujaksoa pidemmällä aikavälillä
- Henkilökohtaiset toiveet: vapaapäivä, vapaa työvuoro, töissä tiettynä työvuorona
- Uusi työntekijä ei voi olla töissä ilman, että paikalla on kouluttaja

Väitöskirja sisältää laajan kirjallisuuskatsauksen hoitohenkilöstön työvuorojen optimoinnista. Käytettyjä menetelmiä ovat matemaattinen ohjelmointi, tekoälymenetelmät, heuristiikat ja metaheuristiikat. ANROM:ia verrataan lukuisiin kirjallisuudessa esiintyviin optimointimalleihin. ANROM osoittautuu kaikkiin aikaisempiin ratkaisimiin verrattuna joustavammaksi ja kattavammaksi työvuorojen optimointimalliksi. Lisäksi käsitelty Belgian sairaaloiden tapaus on laajempi ja monimutkaisempi kuin mikään aiemmin kirjallisuudessa ratkaistu.

Koska käsiteltävä optimointiongelma on liian vaikea ratkaistavaksi eksaktisti, ANROM perustuu metaheuristiikoiden ja niiden hybridien käyttöön. *Variable neighbourhood search* -menetelmässä (VNS)

työvuoroja tai useiden työvuorojen kokonaisuuksia vaihdetaan keskenään, kun lokaali optimi saavutetaan. Algoritmi käyttää vaihdoksia päästäkseen pois lokaalista optimista, ja sitä voidaan käyttää yhdessä metaheuristiikoiden kanssa ratkaisujen monipuolistamiseksi. Erilaisia vaihtotyyppejä ovat mm. yksittäinen työpäivä, viikonloppu, eniten rikottu pehmeä rajoitus (rikkomuksien vähentämiseen keskittyminen) ja sekoittaminen (useiden työpäivien vaihtaminen työntekijöiden kesken). Parhaat tulokset saadaan, kun erityyppisiä vaihtoja käytetään yhdessä.

Tabu-hakumenetelmä (tabu search, TS) on metaheuristiikka, joka ohjaa lokaalia hakua tarkastelemaan ratkaisuja lokaalin optimin ulkopuolelta. Menetelmä etsii kaikki ratkaisut lähialueelta ja siirtyy parhaimpaan. Samalla menetelmä listaa edellisten iteraatioiden ratkaisuja, mikä pakottaa etenemään uusille alueille ja estää menemisen vanhoihin ratkaisuihin. ANROM käyttää hajautusfunktioita (*hashing function)* listojen koon pienentämiseksi. Algoritmi pysähtyy, kun on tapahtunut tietty määrä iteraatioita ilman tuloksen parantumista. Ratkaisujen monipuolistamiseksi algoritmiin yhdistetään VNS-vaihtoja, jolloin kyseessä on hybridialgoritmi. Tämä pidentää laskenta-aikaa, mutta parannukset ratkaisujen laadussa ovat huomattavia. Kiireellisissä tapauksissa voidaan käyttää puhdasta TS-algoritmia, mutta jos suunnitteluaikaa on esimerkiksi yön yli, on hybridialgortimi parempi valinta. Jopa hybridialgoritmilla on kuitenkin jonkin verran vaikeuksia erittäin monimutkaisten tosielämän ongelmien kanssa, sillä ongelmaksi nousee robustiuden puute.

Memeettisillä algoritmeilla voidaan saavuttaa parempia tuloksia vaikeisiin ongelmiin. Menetelmät soveltavat jyrkimmän laskun heuristiikoita käyttäen runkonaan geneettisiä algoritmeja. Laskenta-aika on pidempi kuin TS-algoritmeilla, mutta samalla ratkaisuille saavutetaan parempi robustisuus erilaisilla parametreilla ja erilaisissa ongelmissa. Geneettiset algoritmit toimivat kuten biologinen prosessi. Hyviä ratkaisuja mutatoidaan keskenään sukupolvi toisensa jälkeen, ja uusista ratkaisuista parhaat toimivat seuraavan mutaatiokierroksen lähtökohtana. Memeettisissä algoritmeissa jokaista ratkaisua parannetaan lokaalisti ennen uuden mutaatiokierroksen aloittamista. ANROM voi käyttää lokaaliin parantamiseen esimerkiksi TS-algoritmia, jolloin käytettävällä memeettisellä TS-hybridialgoritmilla saadaan erittäin hyviä ratkaisuja (parempia, kuin kummallakaan menetelmällä erikseen). Lisäksi hybridialgoritmin laskenta-aika on samaa luokkaa kuin pelkkien memeettisten algoritmien.

Uutena tutkimussuuntana väitöskirjassa nostetaan esille monitavoiteoptimointi. Koska työvuorojen suunnitteluun liittyvät rajoitteet ja parametrit ovat keskenään täysin erilaisia, suunnitteluhenkilöstön on vaikea ilmaista halujaan niiden avulla. Monitavoitemenetelmän avulla suunnittelija voi määrittää kunkin rajoituksen tärkeyden omien preferenssiensä mukaan ja samalla voidaan käyttää kaikkia ANROM:in sisältämiä metaheuristiikoita. Monitavoiteoptimoinita varten kaikille rajoituksille pitää määrittää paras ja huonoin mahdollinen arvo. Menetelmän avulla saadaan käsiteltyä erilaisia ongelmia paremmin kuin yksittäiseen kustannusfunktioon pohjautuvassa lähestymistavassa. Väitöskirjassa ei ole vielä päästy soveltamaan monitavoiteoptimointia todellisiin tilanteisiin, mutta testit ovat antaneet lupaavaa näyttöä siitä, että menetelmän avulla aiemmin saavutettuja tuloksia voidaan parantaa edelleen.

3.6 Muutama esimerkkitapaus lyhyesti

3.6.1 Työvuorojen optimointi lentoliikenteessä

Kohl ja Karisch (2004) käsittelevät työvuoro-optimointia ja aikataulutusta lentoliikenteen näkökulmasta Carmen Crew Rostering -järjestelmällä. Lentoliikenteessä pienetkin prosentuaaliset parannukset kustannusfunktiossa voivat aiheuttaa jopa kymmenien miljoonien dollareiden säästöt vuosittain.

Lentohenkilökunnan suunnittelu ja optimointi jaetaan usein kahteen eri osioon optimoitaessa. Ensiksi optimoidaan anonyymit henkilökuntarotaatiot lennoille, joilla ei ole pidempiä välipysähdyksiä. Tämän jälkeen optimoidaan henkilöt rooleihin heidän erikoisosaamisensa ja muiden vaadittavien tehtävien mukaan. Molemmissa optimoinnin vaiheissa monimutkaiset rajoitukset mm. lainsäädännöstä ja sopimuksista tulee täyttyä.

Vaikka useille lentoyhtiöille optimointiongelman rajoitteet voivat olla erilaisia, ongelman pääpiirteet ovat kaikissa samankaltaisia. Päämääränä on optimoida työvuoroja siten, että kustannuksia minimoidaan samalla ottaen huomioon työntekijöiden preferenssejä mm. vapaapäivien suhteen. Tyypillinen tapa työvuorojen optimointiin on ensin generoida joukko käypiä työvuorolistoja, jonka jälkeen optimoidaan niistä parhaat vaihtoehdot.

3.6.2 Työvuorojen optimointi kuljetustehtävissä

Çezik & Günlük (2004) käsittelee kuljetusrajoitteiden uudelleenformulointia lineaarisen ohjelmoinnin malleissa. Formuloinnin avulla tehtävä saadaan yksinkertaisempaan muotoon, jolloin siinä on vähemmän muuttujia ja se voidaan ratkaista nopeammin. Työvuorojen optimoinnissa tarkoituksena on määrätä riittävä määrä työntekijöitä jokaista työvuoroa kohti siten, että lineaarinen kustannusfunktio minimoituu.

Uudelleenformuloinnissa oletetaan, että työvuorojen tyypit I vaadittujen työntekijöiden määrän d_j jokaiselle ajanjaksolle $j \in [1, P]$ on annettu. Merkitään x_i: llä jokaiseen työvuorotyyppiin määrättyjen työntekijöiden lukumäärää ja w_{ij}:llä tauolle ajanhetkellä j määrättyjen työntekijöiden määrää. Lisäksi y_j kuvaa kokonaismäärää kuinka monta työntekijää voi olla tauolla yhtäaikaisesti ajanhetkellä $j \in R_i$. Myös jokaiselle työvuorotyypille $i \in I$ on määritelty vuoron alkamisaika s_i, loppumisaika f_i sekä ikkuna mahdollisille tauoille $R_i = [a_i, b_i]$. Ongelma voidaan tällöin formuloida muodossa

$$\min \Sigma_{\mathrm{i}\in \mathrm{I}}\, c_i x_i$$

siten että,

$$\Sigma_{i:j\in[s_i,f_i]}\, x_i - y_j \geq d_j \;\forall\, j \in [1, P]$$

$$x_i = \Sigma_{j\in R_i} w_{ij} \;\forall\, i \in I\,,$$

$$y_j = \Sigma_{i:j\in R_i}\, w_{ij} \;\forall\, j \in J$$

$x, y, w \geq 0$ ja kokonaislukuja.

Tällöin rajoitusyhtälöistä ensimmäinen pitää huolen siitä että periodin j aikana on ainakin d_j työntekijää käytettävissä. Toinen rajoitusyhtälö pitää huolen että jokaiselle työntekijälle on taukomahdollisuus ja kolmas rajoitus laskee tauolla olevien työntekijöiden lukumäärää. Julkaisussa on myös esitetty pidemmälle vietyjä formulointeja työvuorojen optimointitehtävälle.

Julkaisussa on keskitytty enemmän kuljetustehtävien tutkimukseen. Tällä projekti-idealla voidaan, mikäli flow-muuttujia ei ole kohdefunktiossa tai muissa kuin kuljetusrajoitteissa, vähentää tehtävän muuttujia ja nopeuttaa laskentaa.

3.6.3 Työvuorojen optimointi dynaamisien tehtävälistojen tapauksessa

Borenstein et al. (2010) esittää British Telecomin työvoiman optimoinnin ratkaisun, jossa teknikoille, joilla on erit osaamiset, annetaan eri taitoja vaativia tehtäviä, jotka saapuvat dynaamisesti päivän aikana. Ratkaisu perustuu siihen, että toiminta-alue on jaettu pienempiin osiin, joista yhteen kukin teknikko päivän aluksi määrätään. Tätä menetelmää on käytetty, jotta ongelma on voitu jakaa pienempiin osiin.

Tutkimuksen empiirinen osio osoittaa, että suoriutuminen riippuu tehtävien etäisyydestä ja alueen koosta. Kaiken kaikkiaan ongelman jakaminen pienempiin osiin sekä tehtävien klusterointi paransi koko systeemin suoriutumista.

4 Tutkimusongelman formulointi

Formuloidaan käsiteltävä vapaapäivien optimointiongelma matemaattisesti. Tähän formulaatioon on lisätty myös kaikki alkuperäiseen ongelmaan tehdyt laajennukset, esimerkiksi yötyövuorot ja osaamis-ryhmät.

Indeksointi:

$$\text{Työntekijät } i = 1 \dots 30$$
$$\text{Työpäivät } j = 1 \dots 28$$
$$\text{Viikonloput } k = 1 \dots 4$$

Päätösmuuttujat:

$$x_{ij} = \text{Työntekijä } i \text{ on töissä päivänä } j$$
$$y_{ij} = \text{Työntekijä } i \text{ on töissä yönä } j$$
$$YVP_{ij} = \text{Työntekijällä } i \text{ on yksittäinen vapaapäivä } j$$
$$YTP_{ij} = \text{Työntekijällä } i \text{ on yksittäinen työpäivä } j$$
$$YKVP_{ij} = \text{Työntekijällä } i \text{ on kahden vapaan päivän jälkeen vapaapäivä päivänä } j$$
$$VKLT_{ik} = \text{Työntekijä } i \text{ on töissä viikonloppuna } k$$
$$VKLR_i = \text{Työntekijän } i \text{ viikonloppurangaistukset}$$
$$TVR_i = \text{Toiverangaistusten määrä työntekijällä } i$$

Vakiot:

pt_j = Päivän j työntekijätarve

yt_j = Yön j työntekijätarve

v_i = Saako työntekijä tehdä viikonlopputöitä (1 = kyllä, 0 = ei)

d_i = Työntekijän i vapaapäivien lukumäärä ajanjaksossa

max_z = peräkkäisten työpäivien maksimimäärä

max_y = peräkkäisten yövuorojen maksimimäärä

$p_{i,n}$ = haluaako i olla n: n kanssa aina töissä

$s_{i,t}$ = onko työntekijällä i erikoistaitoa t

$st_{j,t}$ = erikoistaidon t tarve päivälle j

$t_{i,j}$ = onko työntekijällä i vapaapäivätoive päivälle j

Rajoitusehdot:

Apumuuttuja z_{ij} sille, onko työntekijä vuorokauden *j* aikana töissä:

$$x_{ij} + y_{ij} \leq z_{ij}$$

Työntekijätarve on täytettävä

$$\sum_i x_{ij} \geq pt_j\,, \qquad \forall j$$

$$\sum_i y_{ij} \geq yt_j\,, \qquad \forall j$$

Peräkkäisten työpäivien lukumäärä rajoitetaan

$$\sum_{n=j}^{j+max_z} z_{in} \leq max_z, \qquad \forall i, j = 1 \dots 28 - max_z$$

Rajoitetaan peräkkäisten yövuorojen määrä

$$\sum_{n=j}^{j+max_y} y_{in} \leq max_y, \qquad \forall i, j = 1 \dots 28 - max_y$$

Pakotetaan peräkkäisten vapaapäivien sakkomuuttuja oikeaan arvoonsa

$$1 - z_{ij} - z_{i(j+1)} \leq z_{i(j+2)} + YKVP_{i(j+2)}, \qquad \forall i, j = 1 \dots 26$$

Pakotetaan yksittäisten vapaapäivien sakkomuuttuja oikeaan arvoonsa

$$z_{ij} + z_{i(j+2)} \leq z_{i(j+1)} + YVP_{i(j+1)}, \qquad \forall i, j = 1 \dots 26$$

Pakotetaan yksittäisten työpäivien sakkomuuttuja oikeaan arvoonsa

$$-z_{ij} - z_{i(j+2)} \leq -z_{i(j+1)} + YTP_{i(j+1)}, \qquad \forall i, j = 1 \dots 2$$

Pakotetaan viikonloppuja töissä – muuttuja oikeaan arvoonsa

$$VKLT_{ik} * 2 \geq z_{i,7*k-1} + z_{i,7*k}$$
$$VKLT_{ik} \leq z_{i,7*k-1} + z_{i,7*k}$$

Pakotetaan viikonloppurangaistus oikeaan arvoonsa

$$4 + \sum_k VKLT_{ik} + VKLR_i \geq 1, \qquad \forall i$$
$$4 + \sum_k VKLT_{ik} - VKLR_i \leq 2, \qquad \forall i$$

Vapaapäivien rajoittaminen

$$28 - \sum_j z_{ij} = d_i, \qquad \forall i$$

Viikonlopputöiden estot joillekin työntekijöille

$$\sum_k VKLT_{ik} \leq 4 * v_i, \qquad \forall i$$

Pakotetaan samoina aikoina työtä tekevät työntekijät samoihin vuoroihin

$$p_{in}x_{ij} = p_{in}x_{nj}, \qquad \forall i, \forall j, n = 1 \dots 30$$
$$p_{in}y_{ij} = p_{in}y_{nj}, \qquad \forall i, \forall j, n = 1 \dots 30$$

Pakotetaan erikoistaitojen tarpeiden täyttö

$$\sum_i (s_{i,t} * x_{i,j}) \geq st_{j,t}, \qquad \forall j, \forall t$$

Pakotetaan toiveiden rikkomisesta tuleva rankaisumuuttuja oikeaan arvoonsa

$$\sum_j (t_{i,j} * z_{i,j}) \leq TVR_i, \qquad \forall i$$

5 Kehitetty ratkaisumenetelmä

Edellisessä luvussa esitetty formulointi toimi pohjana ongelman implementoinnissa. Ongelman ratkaisemista varten valittiin alustaksi CPLEX, joka on hyvä väline MILP-tehtävien eksaktiin ratkaisemiseen. Ensin CPLEX:iin koodattiin alkuperäinen ongelma, joka on esitelty luvussa 2.

Mallia laajennettiin asteittain. Aiheenasettajalta pyydettiin aina lisäyksiä malliin, kun edelliset lisäykset oli saatu tehtyä. Lopullisessa mallissa työvuorot jaettiin päivä- ja yövuoroihin, ja työntekijöille määritettiin kolme erilaista osaamisryhmää. Lisäksi mallin työntekijöiden määrää on mahdollista muuttaa isommaksi helposti. Malli sisälsi seuraavat uudet rajoitteet verrattuna alkuperäiseen ongelmaan:

Kovia rajoitteita:

- Yövuoroa edeltävän ja yövuoron jälkeisen päivävuoron tulee olla vapaita (24 tunnin yhtäjaksoista työntekoa ei sallita).
- Yövuoroputken (1-3 yövuoroa) viimeisen yövuoron jälkeen pitää olla vähintään 1 vapaapäivä (kova rajoite).
- Jokaiselle työpäivälle asetetaan tarve yövuorossa olevien työntekijöiden määrälle, joka on viikonloppuisin pienempi kuin arkisin. Työntekijöiden tarpeeksi yövuoroissa asetettiin noin kolmasosa samojen päivien päivävuorojen tarpeista.
- Työntekijällä voi olla yövuoroja maksimissaan 3 peräkkäisenä päivänä.
- Kahdella työntekijällä on oltava samat työvuorot ja vapaapäivät.
- Jokaiselle päivälle vaaditut määrät kunkin osaamisryhmän työntekijöitä.

Pehmeitä rajoitteita:

- Yksittäisten vapaapäivien mahdollisimman tasainen jakaminen työntekijöille (maksimin ja minimin välinen ero maksimissaan 1). Sakot asetettiin siten, että sakon määrä on yksittäisten vapaapäivien maksimin ja minimin välinen erotus.
- Työntekijöiden vapaapäivätoiveiden huomioiminen, jokaisesta toiveen rikkomisesta 0,2 sakkoa.

Käytetty CPLEX-koodi on esitetty liitteessä 1.

6 Mallin testaus ja herkkyysanalyysi

6.1 Ongelman koon kasvattaminen

Alkuperäiselle tehtävälle saatiin ratkaisu, jossa kohdefunktion sakot aiheutuvat ainoastaan tehtävänannossa määritellyistä työntekijöiden pakotetuista vapaista viikonlopuista. Pehmeät rajoitteet

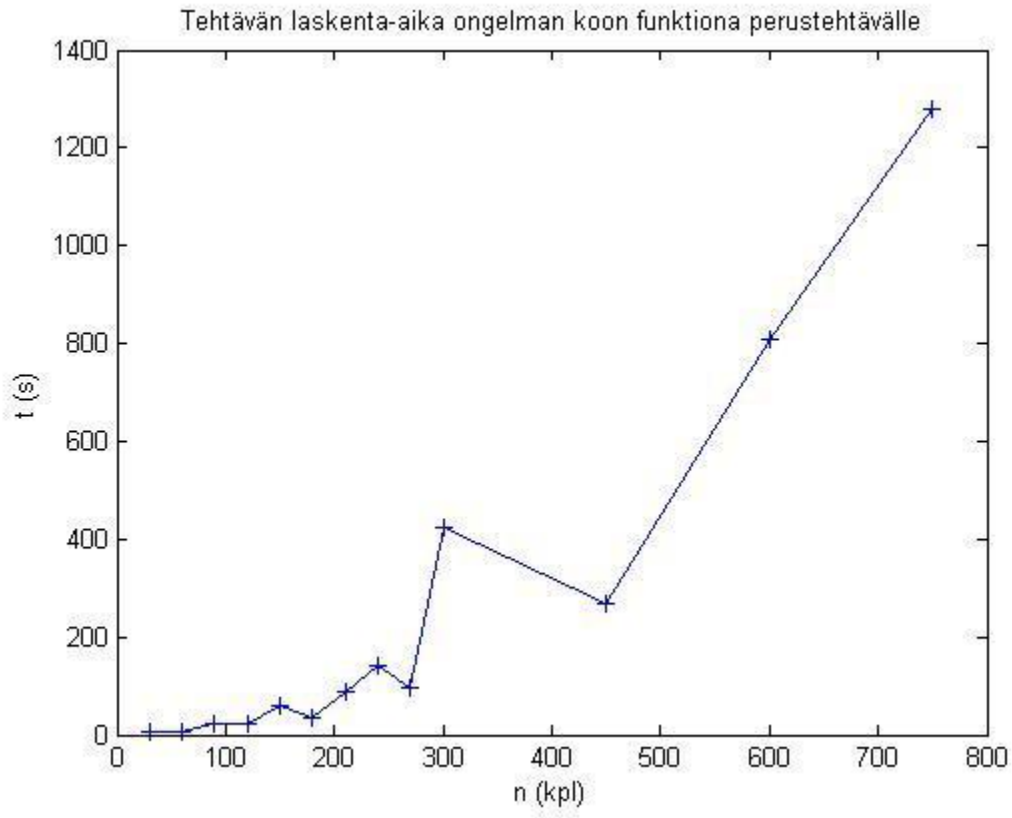
eivät rajoita ratkaisujoukkoa liikaa, joten on mahdollista löytää ratkaisu, joka ei aiheuta ylimääräisiä sakkoja pehmeistä rajoitteista.

Kohdefunktion arvoksi, joka on siis sakkojen kokonaismäärä, saadaan tällöin 12. Kasvattamalla ongelman kokoa alkuperäisen tehtävän monikertoina pysyvät ratkaisut myös alkuperäisen tehtävän monikertoina. Lisäksi ratkaisun sakot ovat alkuperäisen tehtävän sakkojen määrien monikertoja. Suurimmat erot erikokoisten tehtävien välillä ovat tehtävien laskenta-ajoissa. Ongelman kasvaessa tehtävän laskenta-aika kasvaa.

Tehtävän laskenta-ajat eivät kuitenkaan kasva lineaarisesti, vaan kuvaajan (kuva 4) perusteella voidaan sanoa laskenta-aikojen muistuttavan eksponenttijakauman muotoa. Joillain työntekijöiden lukumäärillä tehtävän laskenta-aika kuitenkin myös pienenee tehtävän koon kasvaessa. Tämä johtuu todennäköisesti siitä, että ylimääräisiin päiväkohtaisiin ylimääräisiin työntekijöihin liittyvä rajoite muuttuu löysemmäksi (30 työntekijän ongelmassa voi olla 0 tai 1 ylimääräistä per päivä, kun taas 60 työntekijän ongelmassa voi olla 0, 1 tai 2 ylimääräistä työntekijää per päivä).

Ryhmän käytössä olevilla laitteistoilla ei voitu ratkaista suurempia kuin 750 työntekijän ongelmia. Tämän jälkeen tehtävän laskenta-ajat kasvoivat yli 25 minuuttiin ja käytettävistä laitteistoista loppui muisti kesken.

Kuvassa 4 on esitetty alkuperäisen ongelman laskenta-ajat ongelman koon funktiona.



Kuva 4: Laskenta-aika eri työntekijämäärillä

Nämä testit tehtiin liittyen alkuperäiseen ongelmaan. Jatkossa ongelman koko on 30 työntekijää ja lähtökohtana eri rajoitusten muuttamisen testaamiselle käytetään seuraavassa aliluvussa esiteltävää vertailuratkaisua.

6.2 Vertailuratkaisu

Mallia laajennettaessa huomattiin, että laskenta-ajat alkoivat kasvaa erittäin paljon, joten kaikki lopullisella mallilla tehdyt testit päädyttiin tekemään 30 työntekijän tapauksella. Eri rajoitteita koskevien herkkyysanalyysien tekemistä varten päädyttiin luomaan vertailuratkaisu, joka toimisi lähtökohtana kaikissa testeissä. Vaikka tulevissa tapauksissa optimoitiin vain 30 työntekijän ongelmaa, vertailuratkaisua piti relaksoida aika paljon, jotta sen laskenta-aika pysyisi järkevänä ja laskenta-aikojen kasvamiselle rajoitteita muutettaessa jäisi varaa. Esimerkiksi päivittäisiä vaadittavien työntekijöiden määriä osaamisryhmää kohden piti alentaa melko alhaisiksi, ja lisäksi jokaiselle päivälle sallittiin yksi ylimääräinen työntekijä sekä päivä- että yövuoroihin (alun perin sallittua oli vain 1 ylimääräinen työntekijä per työpäivä). Toiverangaistukset jouduttiin muuttamaan sellaiseen muotoon, että ne eivät skaalaudu yli 30 työntekijän ongelmiin, mutta jatkossa käsitellään muutenkin vain 30 työntekijän ongelmaa. Lisäksi toiveiden määrä jouduttiin rajoittamaan kahteen normaalien työntekijöiden osalta ja nollaan niiden työntekijöiden osalta, jotka eivät saa tehdä töitä viikonloppuisin. Liitteessä 1 esitetty koodi on käytetty vertailuratkaisu.

worker	ma	ti	ke	to	pe	la	su	ma	ti	ke	to	pe	la	su	ma	ti	ke	to	pe	la	su	ma	ti	ke	to	pe	la	su	days off
100	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	8
101	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	8
102	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	8
103	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	8
104	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	8
105	1	0	0	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	8
106	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	8
107	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	8
108	1	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	8
109	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	8
110	1	0	0	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	10
111	1	0	0	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	10
112	1	0	0	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	10
113	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	10
114	1	1	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	0	0	1	1	0	0	10
115	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	10
116	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	0	0	10
117	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	0	0	10
118	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	10
119	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	10
120	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	9
121	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	9
122	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	9
123	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	9
124	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	0	9
125	1	1	1	0	0	1	1	0	0	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	9
126	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	9
127	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1	9
128	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	9
129	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	0	0	9

Kuva 5: Vertailuratkaisu

Vertailuratkaisussa tulokseksi saatiin 12,4 sakkoa. Näistä 12 johtuu työntekijöistä, joilla kaikki viikonloput ovat vapaita. Nämä 12 sakkopistettä tulevat tehdyillä oletuksilla joka tapauksessa, joten 12 on pienin mahdollinen määrä sakkoja, vaikka rajoitteita muutettaisiinkin. Loput 0,4 sakkoa johtuvat kahden vapaapäivätoiveen rikkomisesta. Laskenta-ajaksi saatiin 3 min 10 s, joka on riittävän lyhyt rajoitteiden tiukentamisen ja tästä johtuvan laskenta-ajan kasvun mahdollistamiseksi. Kuvassa 5 on esitetty Excelin avulla tehty kuvaaja vertailuratkaisun lopputuloksesta. Keltaisella merkityt päivät sisältävät päivävuoron ja sinisellä merkityt päivät sisältävät yövuoron.

6.3 Yksittäiset vapaapäivät

Vertailuratkaisussa yksittäisiä vapaapäiviä koskevan pehmeän rajoitteen painokertoimeksi on asetettu 1. Tehtävälle löytyi ratkaisu kaikilla testatuilla painokertoimilla. Laskenta-ajat vaihtelevat 4 sekunnin ja 7 minuutin 39 sekunnin välillä. Tulokset on esitetty taulukossa 1.

Yksittäisten vapaapäivien määrä pysyy nollassa, kun painokerroin on suurempi kuin noin 5*10^-5. Yksittäisten vapaapäivien määrä kasvaa, kun painokerrointa aletaan pienentää tästä. Samalla myös kokonaissakon määrä kasvaa hieman. Kun painokertoimen arvo asetetaan nollaan, yksittäisiä vapaapäiviä tulee 570 kappaletta. Sakkojen määrä on tällöin sama kuin vertailuratkaisussa eli 12,4.

Taulukko 1: Yksittäisten vapaapäivien painokertoimen vaikutus

Painokerroin	Laskenta-aika	Sakot	Yksittäiset vapaapäivät
1	3:10	12,4	0
0,2	3:56	12,4	0
0,05	7:39	12,4	0
5*10^-3	1:19	12,4	0
5*10^-4	0:56	12,4	0
5*10^-5	0:53	12,4	0
1*10^-5	0:10	12,4006	60
1*10^-6	0:21	12,40009	90
1*10^-7	0:13	12,400006	60
1*10^-8	0:09	12,4000012	120
1*10^-9	0:06	12,40000036	360
0	0:04	12,4	570

6.4 Yksittäiset työpäivät

Yksittäisten työpäivien painokerrointa muutettaessa hyvinkin pienillä painokertoimen arvoilla löytyy ratkaisu, jossa yksittäisten työpäivien lukumäärä on nolla. Suuremmilla kertoimen arvoilla kuin 0,05 ratkaisun yksittäisten työpäivien määrä pysyy nollassa ja sakon suuruus vertailuratkaisun mukaisena. Laskenta-ajat eri parametrien arvoilla vaihtelevat 2 ja 3 minuutin välillä.

Kerrointa edelleen pienennettäessä yksittäisten työpäivien merkitys vähenee, jolloin ratkaisuun tulee yksittäisiä työpäiviä. Nollaa suuremmilla kertoimilla yksittäisiä työpäiviä on 2 kappaletta työpäiviä.

Kerrointa pienennettäessä 0,00001 pienemmäksi ratkaisun sakoksi saadaan 12,2 (eli ratkaisussa rikotaan yksi työvuorotoive). Laskenta-aika oli tällöin 2 minuuttia 3 sekuntia. Poistamalla muuttujan vaikutus sakkofunktiosta kokonaan tulee yksittäisten työpäivien lukumääräksi 26. Tällöin tehtävän laskenta-aika tosin kasvaa noin 6 minuuttiin 30 sekuntiin. Tulokset on esitetty taulukossa 2.

Raja painokertoimelle, jolla yksittäisiä työpäiviä tulee ratkaisuun, määräytyy toiverajoitteen painokertoimen mukaisesti. Jos työvuorotoiveita ei rikota, optimiratkaisuun on järkevämpää ottaa yksittäisiä työpäiviä, mikäli kokonaissakko on tällöin pienempi kuin työvuorotoiveita rikottaessa.

Taulukko 2: Yksittäisten työpäivien painokertoimen vaikutus

Painokerroin	Laskenta-aika	Sakot	Yksittäiset työpäivät
2	3:15	12.4	0
0.5	2:28	12.4	0
0.1	3:52	12.4	0
0.05	2:57	12.3	2
0.01	3:14	12.22	2
0.005	3:10	12.21	2
0.001	4:05	12.203	3
0.0005	8:24	12.201	2
0.0001	2:49	12.2002	2
0.00005	5:54	12.2001	2
0.00001	2:26	12.2	2
0	6:16	12.2	26

6.5 Yli 2 vapaata putkeen

Kahden vapaapäivän ylittävien vapaapäivien painokertoimen alkuarvo oli 3. Esimerkiksi, mikäli vapaapäiviä on kolme putkeen, tulee 3 sakkoa, ja mikäli neljä, tulee sakkoa 6 pistettä. Painokerrointa pienentämällä nähdään, missä vaiheessa vertailuratkaisun vapaapäivätoiveisiin liittyvät sakot muuttuvat yli 3 perättäiseen vapaapäivään liittyviksi sakoiksi. Tulokset on esitetty taulukossa 3.

Taulukko 3: Kahden vapaapäivän ylittävien vapaiden painokertoimen muuttamisen vaikutus

Painokerroin	Laskenta-aika	Sakot
3	3:10	12,4
1	1:30	12,4
0,4	4:56	12,4
0,2	7:55	12,4
0,1	8:06	12,4
0,099	4:50	12,396
0,05	7:25	12,2
0,01	6:58	12,04
0	6:14	12

Painokertoimella 0,1 sakot tulevat vielä vapaapäivätoiveiden rikkomisesta, kun taas tästä pienemmäksi mentäessä vapaapäivätoiveet eivät enää rikkoudu ja sakot alkavat johtua yli kahden vapaapäivän putkista. Itse asiassa ratkaisuissa esiintyy 0,1:stä pienemmillä (mutta >0) painokertoimilla 2 neljän vapaapäivän putkea, jotka aiheuttavat sakoissa 12:n yli menevän osuuden. Testatuissa tapauksissa laskenta-ajat pysyvät enintään 8 minuutissa.

Kun painokerroin tiputetaan 0:aan, eli rajoite otettiin pois käytöstä, syntyy ratkaisu, jossa on 12 kolmen vapaapäivän putkea ja 4 neljän vapaapäivän putkea. Sakot tippuvat pienimpään mahdolliseen arvoon, eli 12:een.

6.6 Viikonloppuvapaiden määrä

Vertailuratkaisussa viikonloppuvapaiden määrän poikkeamista tuli 12 sakkopistettä pakotettujen viikonloppuvapaiden työntekijöiden vuoksi. Muuttamalla painokerrointa sakkotermille viikonloppuvapaiden määrän poikkeamiselle yhdestä tai kahdesta, ei tulos muutu, kun käytetään nollaa suurempia painokertoimia. Kertoimen asettaminen nollaksi muuttaa ratkaisua, koska viikonloppuvapaiden määrää ei enää yritetä säätää millään tavalla.

6.7 Yhteiset vapaapäivät

Jotta voidaan asettaa kahden työntekijän työvuorot keskenään samanaikaisiksi, tulee näiden kahden työntekijän kuulua samaan perusjoukkoon, jolla on sama määrä vapaapäiviä kuukaudessa. Tällöin siis esimerkiksi tulee molemmilla työntekijöillä olla kahdeksan vapaapäivää kuukauden aikana. Tämä tilanne tulee ajankohtaiseksi, mikäli työntekijät tulevat töihin esimerkiksi samalla kuljetuksella pitkän matkan päästä.

Yhteisten työvuorojen rajoite voidaan asiakkaan toiveiden mukaan muuttaa joko pehmeäksi rajoitteeksi, jolloin ei ole välttämätöntä toteuttaa työntekijöiden toivetta samoista työvuoroista, tai kovaksi rajoitteeksi, jolloin työvuorojen on pakko olla samat. Muuttamalla rajoite pehmeäksi voidaan muuttaa yhteisien työvuorojen rikkomisen painoarvoa, jolloin työvuorojen ei täydy olla täysin samat.

Vertailuratkaisun sakot muodostuivat pakotetuista viikonlopuista sekä toteuttamattomista toiveista. Vertailuratkaisussa käytettiin kovaa rajoitetta yhteisille työvuoroille, ja yhteiset työvuorot laitettiin kahdelle työntekijälle. Testataan nyt pehmeää rajoitetta. Painokertoimen arvoilla, jotka ovat suurempia kuin 0,01 (eli 0,01 sakkoa per työvuoro, jossa kyseiset työntekijät eivät ole töissä yhtä aikaa) ovat kahden työntekijän työvuorot täysin samat, jolloin siis poikkeamia kahden työntekijän työvuoroissa ei ole ollenkaan. Pienentämällä edelleen painokertoimen arvoa tulee ratkaisun kannalta edullisemmaksi rikkoa yhteisten työvuorojen rajoitetta työntekijöiden toiveiden sijaan, jolloin optimiratkaisussa on riittävästi poikkeamia, jotta saadaan työntekijöiden työvuorotoiveet täytettyä. Painokertoimen arvo, jolla kyseinen rajoite ei enää aiheuta sakkoja, on 0,01. Tätä pienemmillä painokertoimen arvoilla on edullista rikkoa ennemmin yhteisten työvuorojen rajoitetta kuin työvuorotoiveita.

Laskenta-aika tehtävälle on 0,01:tä suuremmilla painokertoimen arvoilla noin 6 minuuttia ja 0,01 pienemmillä painokertoimen arvoilla alle 3 minuuttia. Pienemmillä painokertoimen arvoilla tehtävän laskenta-aika pienenee. Riittävän pienillä painokertoimen arvoilla laskenta-aika pienenee noin puoleen, ja toiverajoituksen sijasta rikotaan yhteisten työvuorojen rajoitetta. Laskenta-aika pienenee luultavasti siksi, että yhteisten työvuorojen rajoitetta rikottaessa on helpompi löytää ratkaisu, joka rikkoo rajoitetta mahdollisimman vähän, toisin kuin toiverajoituksen tapauksessa. Tulokset on esitetty taulukossa 4.

Taulukko 4: Yhteisten työvuorojen painokertoimen vaikutus

Painokerroin	Laskenta-aika	Sakot	Poikkeamat
3	5:46	12.4	0
0.1	6:33	12.4	0
0.01	3:14	12.04	4
0.001	3:10	12.004	4
0.0001	1:35	12.0004	4
0.00001	2:54	12	4
0	1:36	12	12

Lisättäessä yhteiset työvuorot käsittämään useampaa kuin kahta työntekijää, tehtävän laskenta-aika kasvaa. Kolmella työntekijällä tehtävän ratkaisemiseen kuluu 7 minuuttia 3 sekuntia, ja tällöin sakko pysyy samana kuin kahden työntekijän tapauksessa. Neljällä työntekijällä ei työntekijöiden toiveita voida toteuttaa yhtä hyvin kuin aiemmin, jolloin sakoksi saatiin 12,8 optimiratkaisussa (4 toiverikkomusta). Laskenta-aika pysyy lähellä 7 minuuttia. Kun samojen työvuorojen työntekijöiden määrä kasvatetaan viiteen, laskenta-aika kasvaa noin 25 minuuttiin. Tällöin ratkaisun sakko on 13,2 (6 toiverikkomusta), joka on kohtalaisen hyvä huomioiden, että viidellä työntekijällä tulee olla samat vuorot. Tulokset on esitetty taulukossa 5.

Taulukko 5: Yhteisiä työvuorja tarvitsevien hemkilöiden määrän muutttamisen vaikutus

Lukumäärä	Laskenta-aika	Sakot
2	3:06	12.4
3	7:29	12.4
4	6:45	12.8
5	25:11	13.2

6.8 Yksittäisten vapaapäivien mahdollisimman tasainen jakaminen työntekijöille

Vertailuratkaisussa ei ole yhtään yksittäisiä vapaapäiviä, ja siten niiden jakautumisesta ei seuraa sakkoja. Kuten yksittäisten vapaapäivien herkkyysanalyysissa huomattiin, yksittäisiä vapaapäiviä ei tule niin kauan kuin yksittäisten vapaapäivien painokerroin on suurempi kuin noin 5*10^-5. Siten näissä tapauksissa ei myöskään tule rangaistuksia yksittäisten vapaapäivien epätasaisesta jakautumisesta. Pienemmillä painokertoimilla esiintyvät vapaapäivät jakautuvat tasaisesti, eikä näistä seuraa sakkoja.

Kun muutetaan yksittäisten vapaapäivien mahdollisimman tasaisen jakautumisen painokerrointa, sakkojen määrä pysyy edellä mainituista syistä 12,4:ssä. Painokertoimen muuttaminen vaikuttaa laskenta-aikaan siten, että laskenta-aika vaihtelee 40 sekunnin ja 4 minuutin 32 sekunnin välillä (ks. taulukko 6).

Taulukko 6: Yksittäisten vapaapäivien eron painokertoimen muuttamisen vaikutus

Painokerroin	Laskenta-aika (min)	Sakot	Yksittäiset VPt	YksittäisetVPtERORangaistus
1	3:06	12,4	0	0
0,2	4:16	12,4	0	0
0,05	1:48	12,4	0	0
0,005	4:32	12,4	0	0
0,0005	1:34	12,4	0	0
0	0:40	12,4	0	0

Jos yksittäisten vapaapäivien painokerroin asetetaan nollaksi ja muutetaan tasaisen jakautumisen painokerrointa, saadaan alla olevan taulukon mukaiset tulokset. Huomataan, että muutamilla hyvin pienillä painokertoimilla yksittäisten vapaapäivien epätasaisesta jakautumisesta seuraa rangaistuksia. Tulokset on esitetty taulukossa 7.

Taulukko 7: Testejä, kun yksittäiseten vapaapäivien painokerroin on asetettu 0:aan

Painokerroin	Laskenta-aika	Sakot	Yksittäiset vapaapäivät	YksittäisetVPtER Orangaistus
1	0:04	12,4	570	0
0,05	0:03	12,4	540	0
5*10^-4	0:05	12,4	570	0
5*10^-6	0:04	12,4	510	0
5*10^-8	0:04	12,40000005	521	1
5*10^-10	0:04	12,4	510	0
5*10^-12	0:04	12,40000000007	696	14
5*10^-14	0:03	12,4000000000007	696	14
5*10^-20	0:03	12,4	780	0
0	0:03	12,4	780	0

6.9 Vapaapäivätoiveet

Vertailuratkaisuissa työntekijöillä, joilla viikonlopputyöt eivät ole sallittuja, ei ole toiveita, ja muilla on 2 toivetta. Painokerroin toiveiden rikkomiselle on 0,2 per rikottu toive. Vertailuratkaisussa käytetyt toiveet on esitetty liitteessä 2 (tapaus 2 vapaapäivää). Testataan ensin painokertoimen muuttamista. Tulokset on esitetty taulukossa 8.

Taulukko 8: Vapaapäivätoiveiden painokertoimen vaikutus

Painokerroin	Laskenta-aika	Sakot
0	0:44	12
0,01	4:56	12,02
0,2	3:10	12,4
1	7:55	14
2	8:06	16
3	4:50	20
4	4:50	20
4,01	7:25	20,01
5	6:58	21
6	6:14	22
8	3:25	24
8,01	3:42	24
40	1:01	24

Kun painokerroin on välillä]0,4], sakot tulevat kahdesta toiveen rikkomisesta. Painokerrointa nostettaessa sakot tulevat 2 yksittäisestä työpäivästä ja 1 rikotusta toiveesta (välillä]4,8]). Tätä suuremmilla painokertoimilla sakot tulevat kahdesta 4 vapaapäivän putkesta, vapaapäivätoiveita ei enää rikota yhtään. Laskenta-ajat pysyivät näissä testeissä maksimissaan 8 minuutissa. Kun painokerroin tiputetaan 0:aan, eli rajoitus otetaan pois päältä, laskenta-aika tippuu 44 sekuntiin ja sakot tippuvat pienimpään mahdolliseen arvoonsa.

Seuraavaksi muutetaan toiveiden määrää. Testitapauksina käytetään 1, 2 ja 3 toiveen tapauksia sekä kahta tapausta, joissa osalla työntekijöistä on 2 ja osalla 3 toivetta. Pakotettujen viikonloppuvapaiden työntekijöille ei anneta yhtään toivetta. Käytetyt toivepäivät ja tulokset on esitetty liitteessä 2.

Vertailuratkaisu on 2 toiveen tapaus. Kun toiveiden määrää pudotetaan yhteen, sakot pienenevät 12,2:een, eli tulee yhteensä 1 toiverikkomus. Laskenta-aika pysyy lähes kolmessa minuutissa. Kun toiveiden määrä nostetaan kahdesta, laskenta-ajat kasvavat huomattavasti. Kolmella toiveella laskenta-aika on yli 3 tuntia ja optimia ei löydy vieläkään (out of memory -virhe), alarajaksi sakkojen määrälle saadaan 13,4 eli 7 rikkomusta. Tapauksissa, joissa on sekä 2 että 3 toiveen työntekijöitä, laskenta-ajat ovat tunnin suuruusluokkaa ennen out of memory -virhettä. Näissäkin tapauksissa päästään 0,2 sakon päähän alarajasta.

6.10 Osaamisryhmät

Vertailuratkaisussa oli kolme osaamisryhmää, joista kaksi sellaisia, joihin kuului useita työntekijöitä, ja kolmas (”esimiehet”) sellainen, johon kuului vain 5 työntekijää. Kun osaamisryhmän 3 tarve kasvatetaan kolmeen per päivä, kasvaa ratkaisuaika arvoon 9 min 39 s. Kun erikoistarpeet 2 ja 3 poistetaan kokonaan, tulee ratkaisuajaksi 5 min 16 s. Täysin ilman erikoistarpeita laskee ratkaisuaika arvoon 3 min 11 s. Kun tarpeet kasvatetaan suuriksi, mutta mahdollisiksi, muisti loppuu 42 min 13 s jälkeen ja ratkaisu jää 17% päähän sen hetkisestä alarajasta.

6.11 Yövuorot

Jos vertailuratkaisun yötyötarvetta vähennetään ja päivätyötarvetta vastaavasti nostetaan, sakkojen määrä pysyy 12,4:ssä ja laskenta-aika putoaa hieman yli puoleen minuuttiin. Yötyötarvetta voidaan nostaa kahdella henkilöllä per vuoro ilman, että sakkojen määrä muuttuu. Tulokset on esitetty taulukossa 9. Jos yötyötarvetta nostetaan enemmän kuin kahdella henkilöllä per vuoro, CPLEXin muisti loppuu kesken. Tässä vaiheessa sakkoja on kertynyt vertailuratkaisua enemmän. Lisäsakot muodostuvat yksittäisistä vapaapäivistä, viikonloppurangaistuksista ja yksittäisten vapaapäivien epätasaisesta jakaantumisesta.

Taulukko 9: Yövuorotarpeen muuttamisen vaikutus

Yötyötarpeen muutos	Laskenta-aika	Sakot
kaikki päivätyössä	0:34	12,4
-5	0:31	12,4
0	3:06	12,4
+1	5:36	12,4
+2	3:06	12,4
+3	out of memory 19:27	17,4
+5	out of memory 34:20	25,2
+10	out of memory 19:10	17,4
kaikki yötöissä	no value	no value

Yötyöjakson maksimipituus on asetettu vertailuratkaisuksi kolmeksi yövuoroksi. Kun maksimipituus lasketaan kahteen yövuoroon, CPLEXin muisti loppuu 18-21 minuutin ajon jälkeen. Tällöin sakkojen määrä on 14,4. Lisäsakot vertailuratkaisun 12,4 sakkoon tulevat yhdestä yksittäisestä vapaapäivästä ja tästä seuraavasta yksittäisten vapaapäivien mahdollisimman tasaisesta jakautumisesta. Yötyöjakson maksimipituuden nostaminen ei vaikuta sakkojen määrään, mutta laskee laskenta-ajan noin viiteen sekuntiin (ks. taulukko 10).

Taulukko 10: Peräkkäisten yövuorojen maksimimäärän muuttamisen vaikutus

Yötyöjakson maksimipituus	Laskenta-aika	Sakot
6	0:04	12,4
4	0:05	12,4
3	3:06	12,4
2	out of memory 18-21min jälkeen	14,4

Vertailuratkaisussa pehmeä rajoite siitä, että yövuoroputken jälkeen työntekijällä on oltava 2 vapaapäivää, on poistettu kohdefunktiosta. Rajoitteen mukaanottaminen ei vaikuta sakkojen määrään, mutta nostaa laskenta-ajan noin 9 minuuttiin 15 sekuntiin. Tulokset on esitetty taulukossa 11.

Taulukko 11: Rajoitteen "on oltava 2 vapaapäivää yötyöputken jälkeen" muuttamisen vaikutus

Yötyörangaistukset painokerroin	Laskenta-aika	Sakot
0	3:06	12,4
0,000001	9:17	12,4
0,005	9:17	12,4
0,5	9:18	12,4
5	9:13	12,4
50	9:23	12,4
5000	9:12	12,4

7 Yhteenveto ja pohdintaa

Projektityön päämielenkiinnonkohteena olleesta työvuorosuunnitteluongelman esimerkkitapauksesta saatiin kehitettyä optimointimalli, joka myös implementoitiin CPLEX-ohjelmistolla. Malli on toimiva ja implementaatio toteutetuilta osiltaan tarpeenmukaisesti säädeltävissäkin, mutta alkuperäistä 30 työntekijän tapausta suuremmissa ja monimutkaisemmissa ongelmissa malli jää selvästi tehottomaksi jääden työn suorittamiseen käytetyllä laitteistolla jopa ratkaisematta.

Työssä tutkittiin myös mallin hyötyfunktion kertoimien ja rajoitusehtojen vaikutusta ratkaisuun. Alkuperäinen ongelma saatiin ratkaistua ilman ylimääräisiä sakkopisteitä, joten herkkyysanalyysiä varten määriteltiin haastavampi ongelma. Herkkyysanalyysissä huomattiin muutoksia paitsi ratkaisuissa myös ratkaisuajoissa, jotka kasvoivat joitain rajoituksia tiukennettaessa jopa yli kymmenkertaisiksi. Reaalitilanteessa rajoitusten ehdottomuudet ja hyötyfunktion kertoimet olisivat tietysti jossain määrin asiakaskohtaisia, koska asiakkaat kokevat työvuorosuunnitelman eri hyvyystekijöiden vaikuttavuussuhteet eri tavoin.

Projektityön tavoitteena oli myös pohtia työvuorosuunnitelman epävarmuustekijöiden huomioonottamista. Työvuorosuunnittelussa on kyse ihmisistä, joten suunnitellessa on otettava huomioon häiriöiden mahdollisuus. Vaikka yleensä aikatauluoptimoinnissa on monia keinoja epävarmuustekijöiden huomioimiseen, suoraan malliin liitettäviä mahdollisuuksia ei todettu työvuorosuunnittelun tapauksessa olevan.

Epävarmuustekijöihin voidaan vaikuttaa heikosti työvuorosuunnittelussa, mutta niiden aiheuttamia häiriöitä voidaan hallita monin keinoin. Työvoimavajetta voidaan korjata esimerkiksi varamiespalvelun kautta ja töihin kutsuttavalla lisätyövoiman reservillä. Reservi koostuu useilla aloilla pitkälti opiskelijoista ja eläkeläisistä, joille vajavainen työllistyminen on jopa tervetullutta. Suuremmissa yksiköissä, kuten sairaaloissa, voidaan hyödyntää mahdollista toisten osayksiköiden työvoiman hetkellistä ylimäärää. Lisäksi, mikäli työntekijöiden keskimääräisestä sairauspoissaolojen määrästä on jonkinlaista tietoa, voidaan tämä ottaa huomioon jo suunnitteluvaiheessa asettamalla työvuoroon ylimääräisiä henkilöitä oletetun poissaolomäärän verran. Yrityksen on tähän liittyen määritettävä riskitaso, jolla se on valmis toimimaan työvoiman saatavuuteen liittyvien epävarmuustekijöiden vallitessa.

Hyvän työvuorosuunnitelman perusedellytyksenä on kuitenkin myös työvoimatarpeen oikea määrittäminen, mikä asettaa useilla aloilla sairastapausten ja muiden työvoimahäiriöiden huomioonottamisen mielekkyyden jopa olemattomaksi, itse tarpeen ennustettavuuden ollessa huomattavasti merkitsevämpi vaikeustekijä. Esimerkiksi kaupanalalla asiakasmäärät ovat työntekijöiden töihin saapumista huomattavasti ailahtelevaisempi työvuorosuunnittelussa huomiota vaativa tekijä.

8 Itsearviointi

Työn tekeminen sujui suurelta osin alkuperäisen projektisuunnitelman mukaisesti. Asiakkaan tapa antaa tehtävään vähitellen tarkennuksia ja lisäyksiä varmisti, että projekti pysyi hallinnassa eikä kerralla tullut liian suuria haasteita ratkaistavaksi. Asiakkaan antamat lisäykset toteutettiin malliin asteittain, ja kun edelliset oli saatu tehtyä, asiakkaan kanssa sovittiin uusista lisäyksistä. Tällä tavalla projektin loppuvaiheen työmäärä pystyttiin pitämään kohtuullisena ja aikataulussa pysyttiin hyvin.

Projektin todellinen työmäärä pysyi melko hyvin alkuperäisen 5 opintopisteen työmäärän suuruusluokassa. Ryhmän tapa tavata viikoittain ja jakaa seuraavalle viikolle itsenäisesti suoritettavat tehtävän osiot teki työskentelystä tehokasta. Kukin ryhmästä pystyi tekemään työtä, kun siihen itsellään oli aikaa ja resursseja. Ryhmän kokoontumisissa käytiin läpi, kuinka tehtävät ovat edistyneet, ja projektipäällikkö jakoi seuraavaan tapaamiseen mennessä suoritettavat tehtävät. Pääosin annetut tehtävät olivat aina suoritettu määräaikaan mennessä, jolloin aikataulusta myöhästymisen ongelmaa ei tullut.

Ryhmä teki Facebookiin ryhmän, jonka avulla viestimisen viive jäi todella pieneksi ja kommunikaatio toimi hyvin. Viestintä asiakkaan kanssa hoidettiin pääosin sähköpostilla. Viestinnässä oli ajoittain viiveitä, koska asiakas oli kiireinen. Tämä aiheutti työn suorittamiseen pieniä katkoja, mutta toisaalta taas vähensi työn kuormittavuutta. Asiakkaan kanssa pidettiin kaksi Skype-palaveria, joissa kysyttiin tarkentavia ohjeita työn suorittamista varten.

Projektin työmäärä jakautui hyvin ryhmän eri jäsenten kesken. Kukin suoritti tehtäviä, joissa tunsi parhaiten auttavansa ryhmää projektin suorittamisessa. Projektipäällikön kokemattomuus ei projektin alkuvaiheiden pienten epävarmuuksien jälkeen vaikuttanut työn laatuun laisinkaan. Ryhmän synergia parani työn edetessä jatkuvasti, ja työnjako sujui projektin loppuvaiheessa jo luonnollisesti. Lisäksi projektipäällikön ottama suurempi vastuu tehtävien jakamisessa loi luottamusta ryhmän sisällä projektin onnistumiseen ja sujuvuuteen.

Projektin alkuvaiheessa työmäärä ei jakautunut aivan yhtä tasapuolisesti kaikkien ryhmän jäsenten kesken, sillä Santun aikaisempi kokemus samantyyppisten ongelmien parissa lisäsi hänen työmääräänsä alkuperäisen ongelman implementoinnissa CPLEX:in. Toisaalta Santtu tuli mukaan projektiin hieman myöhässä, mikä tasapainotti työmäärää. Alkuperäisen ongelman implementoinnin jälkeen myös muut ryhmän jäsenet täydensivät mallin koodia, jakaen vastuun tasaisemmin ryhmän kesken.

Kommunikaatiossa asiakkaan suuntaan olisi voinut yrittää pitää tiiviimpää yhteyttä, jotta työn tekemisessä tulleet tauot olisivat olleet pienempiä. Asiakas olisi myös voinut miettiä valmiiksi suurempia

kokonaisuuksia kerralla lisättäväksi, jolloin ryhmä olisi pystynyt kehittämään ratkaisuaan ilman katkoksia. Toisaalta tämä olisi voinut nostaa projektin työmäärän liian suureksi opintopisteisiin nähden. Projektia suunniteltaessa olisi voinut tarkemmin miettiä, mitkä osa-alueet työstä ovat järkeviä toteuttaa ja mitkä eivät. Tavoitteet muuttuivat kohtalaisen paljon työn aikana. Alkuperäisen ratkaisumenetelmien tutkimisen sijaan työn pääpaino siirtyi herkkyysanalyysiin.

Työn tavoitteet alkuperäiset tavoitteet saavutettiin kohtuullisen hyvin. Tämän lisäksi keskityttiin herkkyysanalyysiin ja parametrien vaikutuksien testaamiseen. Työn tekemisen aikana vähitellen lisätyt tavoitteet hankaloittivat työn edistymisen arviointia, joten lopullista tuotetta oli vaikea arvioida etukäteen.

Kurssi on loistava tapa päästä soveltamaan opiskelujen aikana opittuja taitoja oikeiden ongelmien ratkaisussa. Kurssilla oli monia mielenkiintoisia ja erilaisia aiheita, jotka takasivat, että jokaiselle opiskelijalle löytyi mielenkiintoinen aihe. Erityisen mielenkiintoisen kurssista teki ekskursiot aiheiden asettavien yritysten tiloihin, jolloin ryhmät pääsivät pitämään oman esityksensä vieraalle yleisölle ja harjoittelemaan esiintymistä.

Kurssin opetuksen rakenne antoi paljon tilaa opiskelijoiden omille suorittamistavoille ja aikatauluille. Jokainen ryhmä oli sopivassa määrin vastuussa omasta edistymisestään ja työn laadusta.

Lähteet

[1] Borenstein, Y., Shah, N., Tsang, E., Dorne, R. Alsheddy, A. & Voudouris, C.. 2010. *On the partitioning of dynamic workforce scheduling problems.* Journal of Scheduling, Vol. 13, Issue 4: 411-425.

[2] Bourdais S., Galinier P., & Pesant G.. 2003. *A Constraint Programming Application to Staff Scheduling in Health Care*. Lecture Notes in Computer Science, Vol. 2833: 153-167.

[3] Çezik, T., & Günlük, O.. 2004. *Reformulating Linear Programs with Transportation Constraints—With Applications to Workforce Scheduling.* Naval Research Logistics Vol. 51, Issue 2: 275–296.

[4] Cheang, B., Li, H., Lim, A. & Rodrigues, B. 2002.. *Nurse Rostering Problems–a Bibliographic Survey.* European Journal of Operational Research, Vol. 151, Issue 3: 447–460.

[5] CI Computational Intelligence. 2014. *Työvoiman hallinnan optimointi (white paper).* Saatavissa: http://www.computationalintelligence.fi/TyovoimanHallinnanOptimointi.pdf (sähköinen) Viitattu 25.2.2014.

[6] Eggenberg, N.. 2009. *Combining Robustness and Recovery for Airline Schedules.* EPFL.

[7] Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B. & Sier, D.. 2004(1). *An Annotated Bibliography of Personnel Scheduling and Rostering.* Annals of Operations Research Vol. 127, Issue 1-4: 21-144.

[8] Ernst, A.T., Jiang, H., Krishnamoorthy, M. & Sier, D.. 2004(2). *Staff Scheduling and Rostering: A Review of Applications, Methods and Models.* European Journal of Operational Research Vol. 153, Issue 1:3–27.

[9] Kohl, N., & Karisch, S.E.. (2004) *Airline Crew Rostering: Problem Types, Modeling, and Optimization.* Annals of Operations Research March 2004, Vol. 127, Issue 1-4: 223-257.

[10] Kohl, N., Larsen, A., Larsen, J., Ross, A. & Tiourine, S.. 2007. *Airline Disruption Management - Perspectives, Experiences and Outlook.* Journal of Air Transport Management, Vol. 13, Issue 3: 149-162.

[11] Kyngäs, J.. 2011. *Solving challenging real-world scheduling problems.* Väitöskirja. Turku Centre for Computer Science. Saatavissa: http://www.doria.fi/bitstream/handle/10024/72127/D140%20doria.pdf?sequence=1 (sähköinen). Viitattu 25.2.2014.

[12] Lan, S.. 2003. *Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Achieve Minimum Passenger Disruptions.* Transportation Science, Vol. 40, Issue 1: 15-28

[13] Naveh, Y., Richter, Y., Altshuler, Y., Gresh, D.L. & Connors, D.P.. 2007. *Workforce Optimization: Identification and Assignment of Professional Workers Using Constraint Programming.* IBM Journal of Research and Development - Business optimization, Vol. 51, Issue 3: 263-279.

[14] Topaloglua, S. & OzkarahanbI, I.. 2010. *A Constraint Programming-based Solution Approach for Medical Resident Scheduling Problems.* Computers & Operations Research, Vol. 38, Issue 1: 246–255.

[15] Tower, P.K.. 2007. *An Analysis of Robust Workforce Scheduling Models for a Nurse Rostering Problem.* Thesis, Department of Air Force, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.

[16] Vanden Berghe, G. 2002.. *An Advanced Model and Novel Meta-heuristic Solution Methods to Personnel Scheduling in Healthcare.* PhD Thesis, School of Computer Science and Information Technology, University of Nottingham.

[17] Zuo, X., Mo, H., Wu, J.. 2009. *A robust scheduling method based on a multi-objective immune algorithm.* Information Sciences, Vol. 179, Issue. 19: 3359-3369.

Liite 1: CPLEX-koodi

```
/*********************************************
* OPL 12.4 Model
* Author: Santtu Klemettilä, Teemu Kinnunen, Ilari Vähä-Pietilä, Lotta Martikainen
* Creation Date: 14.05.2014 at 17.15.50
*********************************************/
// Basic definitions
int A=1; //Multiplier to amount of workers, 1=30 workers, 2=60 and so on.
range Workers = 1..A*30; // Number of workers
range Days = 1..28; // Number of days
range Weekends = 1..4; // Number of weekends
//Number of workers needed in day shift per day
int Tarve[Days] = [A*16,A*16,A*16,A*16,A*18,A*11,A*9,A*16,A*16,A*16,A*16,A*18,
A*11,A*9,A*16,A*16,A*16,A*16,A*18,A*11,A*9,A*16,A*16,A*16,A*16,A*18,A*11,A*9];
//Number of workers needed in night shift per day
int Yotarve[Days] = [A*6,A*6,A*6,A*6,A*6,A*3,A*3,A*6,A*6,A*6,A*6,A*6,A*3,A*3,
A*6,A*6,A*6,A*6,A*7,A*4,A*3,A*6,A*6,A*6,A*6,A*6,A*3,A*3];

//Definition of knowledge groups to workers
int Erikois1[Workers] = [1,1,1,0,0,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,1];
int Erikois2[Workers] = [1,1,0,1,0,0,0,0,1,0,0,1,0,1,0,1,1,0,1,0,1,1,1,1,1,0,0,1,0,0];
int Erikois3[Workers] = [0,0,0,0,0,0,0,0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
//Number of workers of different knowledge groups needed per day
int Erikois1Tarve[Days] = [4,4,4,4,4,2,2,4,4,4,4,4,2,2,4,4,4,4,4,2,2,4,4,4,4,4,2,2];
int Erikois2Tarve[Days] = [7,7,7,7,7,5,5,7,7,7,7,7,5,5,7,7,7,7,7,5,5,7,7,7,7,7,5,5];
int Erikois3Tarve[Days] = [2,2,2,2,2,1,1,2,2,2,2,2,1,1,2,2,2,2,2,1,1,2,2,2,2,2,1,1];

// Decision variables
dvar boolean paivatyo[Workers,Days]; //Does the day have a day shift?
dvar boolean yotyo[Workers,Days]; //Does the day have a night shift?
//single day off
dvar boolean yksittainenVP[Workers,Days];
//single working day
dvar boolean yksittainenTP[Workers,Days];
//over 2 days off in a row
dvar boolean ylitoinenVP[Workers,Days];
```

```
//working weekend
dvar boolean vklopputoissa[Workers,Weekends];
//penalty from too small/large amount of free weekends
dvar int vkloppurangaistus[Workers];
//penalty from breeaking common working days
dvar boolean parirangaistus[Days];
//penalty from breaking day off requests
dvar int toiverangaistus[Workers];
//penalty from under two days off after night shifts, not in use in the final version
//dvar boolean yotyorangaistus[Workers, Days];

// **********
// Expressions involving decision variables
// **********

//variable indicating working days
dexpr float TP[i in Workers, j in Days] = paivatyo[i,j] + yotyo[i,j];

// Number of workers in day j in different shifts and from different knowledge groups
dexpr float NumWorkersPaiva[j in Days]=sum(i in Workers) paivatyo[i,j];
dexpr float NumWorkersYo[j in Days]=sum(i in Workers) yotyo[i,j];
dexpr float NumWorkersErikois1[j in Days]=sum(i in Workers) Erikois1[i]*paivatyo[i,j];
dexpr float NumWorkersErikois2[j in Days]=sum(i in Workers) Erikois2[i]*paivatyo[i,j];
dexpr float NumWorkersErikois3[j in Days]=sum(i in Workers) Erikois3[i]*paivatyo[i,j];

//surplus workers in day and night shifts
dexpr float PaivaSurplus[j in Days]= NumWorkersPaiva[j] - Tarve[j];
dexpr float YoSurplus[j in Days]= NumWorkersYo[j] - Yotarve[j];

//variable indicating day offs
dexpr float VP[i in Workers, j in Days] = 1 - TP[i,j];

//variable which indicates amount of working weekends
dexpr float VkloppujaToissa[i in Workers] = sum(j in Weekends) vklopputoissa[i,j];

//Counters
dexpr float yksittaisetVPtapu[j in Days] = sum(i in Workers) yksittainenVP[i,j];
dexpr float yksittaisetVPt = sum(j in Days) yksittaisetVPtapu[j];
dexpr float yksittaisetVPtERO[i in Workers] = sum(j in Days) yksittainenVP[i,j];
```

```
dexpr float yksittaisetVPtEROmax = max(i in Workers) yksittaisetVPtERO[i];
dexpr float yksittaisetVPtEROmin = min(i in Workers) yksittaisetVPtERO[i];
//penalty from the difference in amount of single days off
dexpr float yksittaisetVPtEROrangaistus = yksittaisetVPtEROmax - yksittaisetVPtEROmin;
dexpr float yksittaisetTPtapu[j in Days] = sum(i in Workers) yksittainenTP[i,j];
dexpr float yksittaisetTPt = sum(j in Days) yksittaisetTPtapu[j];
dexpr float ylitoisetapu[j in Days] = sum(i in Workers) ylitoinenVP[i,j];
dexpr float ylitoiset = sum(j in Days) ylitoisetapu[j];
dexpr float vkloppurangaistukset = sum(i in Workers) vkloppurangaistus[i];
dexpr float daysoffcount[i in Workers] = sum(j in Days) VP[i,j];
dexpr float parirangaistukset = sum(j in Days) parirangaistus[j];
dexpr float toiverangaistukset = sum(i in Workers) toiverangaistus[i];
//dexpr float yotyorangaistuksetapu[j in Days] = sum(i in Workers) yotyorangaistus[i, j];
//dexpr float yotyorangaistukset = sum(j in Days) yotyorangaistuksetapu[j];

//calculating total amount off penalties
dexpr float TotalScore = yksittaisetVPt + yksittaisetVPtEROrangaistus + 2*yksittaisetTPt +
3*ylitoiset +
vkloppurangaistukset + 3*parirangaistukset + 0.2*toiverangaistukset; // + 0.5 *
yotyorangaistukset; // + 0.2*toiverangaistukset;

// ************************
// Objective function
// ************************

//Maximize objective
maximize(-TotalScore);

// ************************
// Constraints
// ************************

subject to{
forall(i in Workers, j in Days){
TP[i,j] <= 1;
}

forall(i in Workers, j in 1..27){
```

```
yotyo[i,j] + paivatyo[i,j+1] <= 1;
}

forall(j in Days){
        //amount of workers needed
        //NumWorkers[j]>=Tarve[j];
        //Surplus
PaivaSurplus[j] <= A;
PaivaSurplus[j] >= 0;
YoSurplus[j] <= A;
YoSurplus[j] >= 0;
//amount of workers from different knowledge groups per day must be equal or greater than the need
NumWorkersErikois1[j] >= Erikois1Tarve[j];
NumWorkersErikois2[j] >= Erikois2Tarve[j];
NumWorkersErikois3[j] >= Erikois3Tarve[j];
}

forall(i in Workers){
//if under 1 or over 2 free weekends, penalty is given
4-VkloppujaToissa[i] + vkloppurangaistus[i] >=    1;
4-VkloppujaToissa[i] - vkloppurangaistus[i] <= 2;
}

forall(i in Workers, j in Weekends){
   //calculating amount of working weekends
   vklopputoissa[i,j] * 2 >= paivatyo[i,(j-1)*7+6] + yotyo[i,(j-1)*7+6] + paivatyo[i,(j-1)*7+7] + yotyo[i,(j-1)*7+7];
   vklopputoissa[i,j] <= paivatyo[i,(j-1)*7+6] + yotyo[i,(j-1)*7+6] + paivatyo[i,(j-1)*7+7] + yotyo[i,(j-1)*7+7];
}

forall(i in (A*10+1)..(A*10+A*3), j in 1..4){
   //some workers are not allowed to work on weekends
   vklopputoissa[i,j] == 0;
}

forall(i in (A*20+1)..(A*20+A*3), j in 1..4){
```

```
   //some workers are not allowed to work on weekends
   vklopputoissa[i,j] == 0;
}

forall(i in 1..(A*10)){
   //hard constraint to total amount of days off per worker
   daysoffcount[i] == 8;
}

forall(i in (A*10+1)..(A*20)){
   //hard constraint to total amount of days off per worker
   daysoffcount[i] == 10;
}

forall(i in (A*20+1)..(A*30)){
   //hard constraint to total amount of days off per worker
   daysoffcount[i] == 9;
}

forall(i in Workers, j in 1..26){
//penalties from single days off
TP[i,j] + TP[i,j+2] - 1 <= TP[i,j+1] + yksittainenVP[i,j+1];
//penalties from single working days
VP[i,j] + VP[i,j+2] - 1 <= VP[i,j+1] + yksittainenTP[i,j+1];
}
forall(i in Workers, j in 3..28){
//penalties from additional days off after 2 days off in a row
VP[i,j-2] + VP[i,j-1] - 1 <= TP[i,j] + ylitoinenVP[i,j];
}
forall(i in Workers, j in 1..22){
//hard constraint: maximum 6 working days in a row
TP[i,j] + TP[i,j+1] + TP[i,j+2] + TP[i,j+3] + TP[i,j+4] + TP[i,j+5] + TP[i,j+6] <= 6;
}
forall(i in Workers, j in 1..25){
//hard constraint: maximum 3 night shifts in a row
yotyo[i,j] + yotyo[i,j+1] + yotyo[i,j+2] + yotyo[i,j+3] <= 3;
}
```

```
forall(i in 19..19, j in Days){
//hard constraint for common working days
paivatyo[i,j] == paivatyo[i+1,j];
yotyo[i,j] == yotyo[i+1,j];

//soft constraint for common working days 2 workers
//paivatyo[i,j]<=paivatyo[i+1,j]+parirangaistus[j];
//paivatyo[i+1,j]<=paivatyo[i,j]+parirangaistus[j];
//yotyo[i,j]<=yotyo[i+1,j]+parirangaistus[j];
//yotyo[i+1,j]<=yotyo[i,j]+parirangaistus[j];
}

forall(i in Workers, j in 0..0){
//Day off requests, if day x is requested as day off, insert TP[i, j+x]. Soft constraint.
if(i==A){TP[i,j+6]    + TP[i,j+7]    <=toiverangaistus[i];}
else if (i==2*A)    {TP[i,j+1]    + TP[i,j+2]    <=toiverangaistus[i];}
else if (i==3*A)    {TP[i,j+3]    + TP[i,j+4]    <=toiverangaistus[i];}
else if (i==4*A)    {TP[i,j+5]    + TP[i,j+6]    <=toiverangaistus[i];}
else if (i==5*A)    {TP[i,j+7]    + TP[i,j+8]    <=toiverangaistus[i];}
else if (i==6*A)    {TP[i,j+9]    + TP[i,j+10] <=toiverangaistus[i];}
else if (i==7*A)    {TP[i,j+11] + TP[i,j+12] <=toiverangaistus[i];}
else if (i==8*A)    {TP[i,j+13] + TP[i,j+14] <=toiverangaistus[i];}
else if (i==9*A)    {TP[i,j+15] + TP[i,j+16] <=toiverangaistus[i];}
else if (i==10*A) {TP[i,j+17] + TP[i,j+18] <=toiverangaistus[i];}
else if (i==11*A) {0 <=toiverangaistus[i];}
else if (i==12*A) {0 <=toiverangaistus[i];}
else if (i==13*A) {0 <=toiverangaistus[i];}
else if (i==14*A) {TP[i,j+25] + TP[i,j+26] <=toiverangaistus[i];}
else if (i==15*A) {TP[i,j+27] + TP[i,j+28] <=toiverangaistus[i];}
else if (i==16*A) {TP[i,j+1]    + TP[i,j+2]    <=toiverangaistus[i];}
else if (i==17*A) {TP[i,j+3]    + TP[i,j+4]    <=toiverangaistus[i];}
else if (i==18*A) {TP[i,j+5]    + TP[i,j+6]    <=toiverangaistus[i];}
else if (i==19*A) {TP[i,j+7]    + TP[i,j+8]    <=toiverangaistus[i];}
else if (i==20*A) {TP[i,j+9]    + TP[i,j+10] <=toiverangaistus[i];}
else if (i==21*A) {0 <=toiverangaistus[i];}
else if (i==22*A) {0 <=toiverangaistus[i];}
else if (i==23*A) {0 <=toiverangaistus[i];}
else if (i==24*A) {TP[i,j+17] + TP[i,j+18] <=toiverangaistus[i];}
```

```
else if (i==25*A) {TP[i,j+19] + TP[i,j+20] <=toiverangaistus[i];}
else if (i==26*A) {TP[i,j+21] + TP[i,j+22] <=toiverangaistus[i];}
else if (i==27*A) {TP[i,j+23] + TP[i,j+24] <=toiverangaistus[i];}
else if (i==28*A) {TP[i,j+25] + TP[i,j+26] <=toiverangaistus[i];}
else if (i==29*A) {TP[i,j+27] + TP[i,j+28] <=toiverangaistus[i];}
else if (i==30*A) {TP[i,j+27] + TP[i,j+28] <=toiverangaistus[i];}
}

}
```

Liite 2: Taulukko vapaapäivätoiveista

	2 toivetta		1 toive	3 toivetta			2/3 toivetta			2/3 toivetta		
laskenta-aika	0:03:10		0:02:53	3:02:00			0:43:04			>1:15:00		
sakot	12,4		12,2	13,6			13,0			12,8		
lower bound	12,4		12,2	13,4			12,8			12,6		
työntekijä\toive nro.	**1**	**2**	**1**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
1	6	7	6	6	7	23	6	7	-	6	7	-
2	1	2	1	1	2	18	1	2	-	1	2	-
3	3	4	3	3	4	20	3	4	-	3	4	-
4	5	6	5	5	6	22	5	6	-	5	6	-
5	7	8	7	7	8	24	7	8	-	7	8	-
6	9	10	9	9	10	26	9	10	-	9	10	-
7	11	12	11	11	12	28	11	12	-	11	12	-
8	13	14	13	13	14	2	13	14	-	13	14	-
9	15	16	15	15	16	4	15	16	-	15	16	-
10	17	18	17	17	18	6	17	18	-	17	18	-
14	25	26	25	25	26	8	25	26	8	25	26	-
15	27	28	27	27	28	10	27	28	10	27	28	-
16	1	2	1	1	2	12	1	2	12	1	2	-
17	3	4	3	3	4	14	3	4	14	3	4	-
18	5	6	5	5	6	16	5	6	16	5	6	-
19	7	8	7	7	8	18	7	8	18	7	8	-
20	9	10	9	9	10	20	9	10	20	9	10	-
24	17	18	17	17	18	1	17	18	1	17	18	1
25	19	20	19	19	20	3	19	20	3	19	20	3
26	21	22	21	21	22	5	21	22	5	21	22	5
27	23	24	23	23	24	7	23	24	7	23	24	7
28	25	26	25	25	26	9	25	26	9	25	26	9
29	27	28	27	27	28	11	27	28	11	27	28	11
30	27	28	27	27	28	13	27	28	13	27	28	13