Optimointia lukiolaiselle
Pro Gradu-tutkielma

Tero Koho
Matematiikan ja tilastotieteen laitos,
Turun Yliopisto

16. elokuuta 2013

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

KOHO, TERO: Optimointia lukiolaiselle
Pro Gradu -tutkielma. 98 s.
Matematiikka
Elokuu, 2013

Optimointi on sovelletun matematiikan osa-alue, jossa pyritään löytämään paras mahdollinen ratkaisu ongelmaan useimmiten etsimällä funktion minimiä tai maksimia. Myös löydettyjen arvojen ominaisuuksien kuten herkkyyden tutkiminen on osa optimointia. Optimoinnista on konkreettista hyötyä monenlaisissa tilanteissa, koska sillä voidaan esimerkiksi tuoda kustannussäästöjä.

Tutkielmassa käsitellään alueittain optimointia lukiolaisen lähtökohdista käsin pääpainon ollessa monipuolisuudessa ja sovellettavuudessa. Käsiteltäviä alueita ovat optimointiohjelmistojen käyttö, optimoinnnin perusteet, lineaariset ja epälineaariset optimointitehtävät sekä matemaattiset optimointimenetelmät. Käsittelyä havainnollistetaan esimerkein. Tutkielma perustuu kirjallisuuteen ja pääasiallisina lähteinä ovat toimineet Timo Leipälän monisteet Matemaattinen Optimointi I ja II.

Asiasanat: matemattiikka, optimointi, lukio

Sisältö

Luku 1

Johdanto

Tutkielmassa käsitellään optimoinnin perusteita ja optimointiin liittyviä ohjelmistoja. Käsiteltävät asiat perustuvat suurelta osin kirjallisuuteen, mutta ohjelmistojen osalta mukana on myös omia huomioita. Asiat on koostettu itsenäisesti ja niiden ulkoasu on pyritty saamaan helpostiomaksuttavaan muotoon. Kaikki esimerkit on kehitetty itse, vaikka niihin on saatukin runsaasti ideoita kirjallisuudesta. Pääasiallisina lähteinä ovat toimineet Leipälän monisteet Matemaattinen Optimointi I ja II [6] [7].

1.1 Esipuhe

Tutkielman alkuperäinen idea oli toimia optimoinnin kurssina lukiolaisille, mutta siitä tulikin jotain muuta. Jos olen onnistunut tavoitteessani, tutkielma kelpaa matemaattisen optimoinnin peruskirjaksi tai käsikirjaksi lähes jokaiselle optimoinnista tai matematiikasta kiinnostuneelle, jonka omat opinnot eivät käsittele asiaa liian syvällisesti. Joitakin mieleen tulevia ammattiryhmiä ovat liikemiehet, projektien suunnittelijat, koneiden käyttäjät ja insinöörit. Tutkielman tarkoitus on olla matalan vaikeustason laaja katsaus optimointiin, eli oikeastaan pintaraapaisu.

Vaikka vaikeustason on tarkoitus olla matala, matematiikassa rakennetaan aina jo olemassa olevan tiedon varaan[1]. Koska johonkin on vedettävä lähtötason raja, se on vedetty noin lukion lyhyen matematiikan opintojen hyväksyttyyn suorittamiseen. Tämä näkyy lähinnä käsiteltäessä derivaatan käsitettä ja mahdollisesti joissakin matemaattisissa termeissä. Suuri osa tutkielmasta pitäisi kuitenkin olla ymmärrettävissä myös ilman lukion matematiikan taitoja. Tutkielmassa on kuitenkin otettu käyttöön matriisit, joita ei juurikaan käsitellä lukiossa. Niiden osuus on pieni, joten niiden ymmärtämistä ei vaadita kokonaiskuvan saamiseksi. Toisaalta jos lukija ei tunne matriiseja entuudestaan, olkoon tämä haaste hänelle; haasteet auttavat ih-

[1] Paitsi ehkä aksiomatiikassa, jossa pyritään tutkimaan matematiikkaa mahdollisimman pienillä olettamuksilla.

mistä oppimaan ja kasvamaan eikä kummastakaan niistä ole haittaa. Jotta tutkielman asioihin perehtyminen olisi tehokasta, on suositeltavaa kokeilla esimerkkien ratkaisemista itse, varsinkin mikäli lukija haluaa käyttää ratkaisussa eri ohjelmaa kuin tutkielman ratkaisussa on käytetty.

Johdannossa käsitellään lähinnä optimointiin soveltuvia ohjelmistoja, joiden tuntemisesta on hyötyä muissa luvuissa. Luvussa kaksi käsitellään optimonnin yleisiä periaatteita, kuten tehtävien esitystapaa ja termistöä. Luvuissa kolme ja neljä käsitellään erilaisia optimointitehtäviä ja niiden ratkaisemista eri ohjelmistoilla. Luvussa viisi käsitellään eri tilanteisiin sopivia optimointimenetelmiä. Luvussa kuusi käsitellään lyhyesti joitakin optimointiin liittyviä asioita, jotka eivät sopineet muihin lukuihin, mutta joiden poisjättäminen olisi ollut epäkunnioittavaa sekä optimointia että lukijaa kohtaan. Lisälukemista aiheesta löytyy esimerkiksi alla olevista osoitteista. Kyseessä on Marko Mäkelän kirjoittama matemaattisen optimoinnin luentomoniste.

`http://www.math.utu.fi/opiskelu/opetusohjelma/kurssit`
`/aineopinnot/smat5108/MonisteMatOpt1.pdf`
`http://www.math.utu.fi/opiskelu/opetusohjelma/kurssit`
`/aineopinnot/smat5109/opt2.pdf` .

Vielä ennen siirtymistä itse asiaan on aiheellista kiittää eräitä tämän tutkielman valmistumista edesauttaneita henkilöitä. Anne Leskelälle kiitos kielen tarkistuksesta ja lukijan näkökulman tuomisesta. Kiitos Marko Mäkelälle ja Matti Vuoriselle kärsivällisyydestä ja ohjauksesta. Kyösti ja Katariina Tarvaiselle kiitos tuesta ja avusta.

1.2 Ohjelmistoista

Optimointi on hyvin pitkälle tietokoneilla suoritettavaa toimintaa, ja siksi siihen kykenevistä ohjelmistoista kannattaa olla tietoinen. Matemaattisilla ohjelmistoilla on ikävä taipumus olla hinnakkaita, eikä yksittäistä henkilöä luultavasti ole helppo motivoida ohjelmiston hankkimiseen. Toisaalta rahalle saa usein vastinetta, ja ohjelmistoista ainakin Matlab, Mathematica ja Maple pystyvät myös optimointiin. Erityisesti optimointiin on myös kehitetty useita kaupallisia ohjelmistoja, kuten LINDO, Lamps, LOQO ja HOPDM. [9] Jos opiskelee tai aikoo opiskella matematiikkaa ja optimointia, törmää todennäköisesti johonkin näistä tai vastaavaan ohjelmistoon. Jatkossa esitellään joitakin optimointiin soveltuvia ohjelmia tai ohjelmistoja ja niiden esittelyssä voidaan mainita eräitä optimointiin liittyviä termejä. Kyseiset termit on käsitelty niille varatuissa osioissa eikä niiden merkitystä tässä vaiheessa avata.

On kuitenkin olemassa ainakin kaksi ilmaista matematiikkaohjelmaa. Ohjelmistot ovat nimeltään SAGE ja SciLab ja niitä voisi verrata OpenOfficeen: ilmaiseksi saatavilla oleva työkalupaketti, joka riittää erinomaisesti satunnaiselle käyttäjälle. Lukijan ei odoteta investoivan optimointiohjelmistoon tässä

vaiheessa, joten suosittelen SAGEa osoitteesta

`http://www.sagemath.org` ,

tai muita myöhemmin mainittavia ohjelmistoja. SAGEn käytöstä on myöhemmin vielä lyhyt ohje ja Scilabin käyttöön löytyy ohjeita muun muassa osoitteesta

`https://sites.google.com/site/laskenta/scilab` .

Toisaalta myös Microsoft Officen Excel -ohjelmalla voi ratkaista optimointitehtäviä tiettyyn rajaan asti. Tätä menetelmää on selvitetty Taanilan monisteessa, joka on saatavissa osoitteesta

`http://myy.haaga-helia.fi/~taaak/m/optim.pdf`.[12]

Myös Tarvainen on kirjoittanut aiheesta ja viittaa monisteessaan Pulkkisen ja Holopaisen kirjaan Talous- ja rahoitusmatematiikka, WSOY, 1999. [13]

Varsinaiseen optimointiin kykenevien ohjelmien lisäksi on syytä mainita GeoGebra, joka löytyy esimerkiksi osoitteesta

`http://www.geogebra.org/cms/`.

Kyseessä on matemaattinen piirto-ohjelma, jota on hyvin kevyt käyttää. Ohjelma on selkeyteensä ja yksinkertaisuuteensa nähden yllättävän monipuolinen ja mahdollistaa havainnollistavien kuvien piirtämisen yksittäisistä suorista aina planeettojen liikkeitä kuvaaviin värikkäisiin ja tarkkoihin animaatioihin.[2] Optimoinnin suhteen GeoGebra on kätevä graafisen ratkaisun apuvälineenä: rajoittavat epäyhtälöt saadaan selkeästi näkyviin ja kohdefunktion arvon kehitystä voidaan seurata asettamalla se muuttuvaksi parametriksi. Kyseessä on siis ohjelma, joka sopii opettajalle havainnollistusvälineeksi ja oppilaalle ratkaisujen kokeilualustaksi. GeoGebraa on käytetty myös joidenkin tässä työssä esiintyvien kuvien piirtämiseen.

1.2.1 Exelin ja OpenOfficen käyttö optimoinnissa

Microsof Exelin käytöstä optimoinnissa on hyvä ohje yllä mainitussa Taanilan monisteessa, joten toisen ohjeen käyminen tässä ei liene tarpeen. On syytä huomata, että Excelin käyttämä ratkaisin (engl. solver) on Excelin ulkopuolelta tuleva apuohjelma, eivätkä ne ole käytössä esimerkiksi Microsoft Excel Starter -versiossa. Kyseinen ratkaisin on melko monipuolinen asiantuntijaohjelma, joka käyttää eri optimointimenetelmiä tehtävästä riippuen. Kuitenkin saadun ratkaisun järkevyys kannattaa aina tarkistaa, koska ainakin vanhemmissa Exelin versioissa käytetty GRG2-ohjelma saattoi epälineaarisessa optimoinninssa jäädä jumiin paikalliseen ääriarvokohtaan.[3] [13] Eräs Exelillä optimoinnin erityispiirre on vastauksen lisäksi tulevat raportit, joista erityisesti herkkyyteen liittyvää raporttia kannattaa tosielämän tilanteissa hyödyntää. [12]

Hieman yllättävästi myös ilmainen OpenOffice Calc pystyy optimointiin samaan tapaan kuin Excelkin. OpenOfficen mukana tuleva versio ratkaisi-

[2] http://www.geogebra.org/en/upload/files/english/mojca/SolarSystem.html

[3] Käytettyihin termeihin palataan myöhemmin tutkielmassa.

mesta on kuitenkin rajoittunut vain lineaarisiin yhtälöihin, vaikka pystyykin käsittelemään kokonaislukumuuttujia. Ratkaisin toimii melko samalla tavalla kuin Excelissä, eli luodaan taulukko, jossa on kaavat tarvittaville rajoitteille, jätetään päätösmuuttujien solut tyhjiksi, valitaan "Työkalut - Ratkaisin", määritetään optimoitava solu, optimoinnin tyyppi (voidaan myös asettaa tavoitearvo), muuttujien solut, reunaehdot ja lisäasetukset. Tarvittavien toimenpiteiden lista on pitkähkö, mutta ratkaisinta on selkeä käyttää ja jos sen jokaisen rivin lukee, myös kaikki tarvittavat tiedot tulee melko varmasti syötettyä. On myös olemassa ilmainen beta-versio epälineaaristen ongelmien ratkaisimesta osoitteessa
`http://extensions.services.openoffice.org/en/project/NLPSolver` .
Kyseinen ratkaisin käyttää evoluutioalgoritmeja, kuten geneettisiä algoritmeja. Se vaatii asennettaessa, että Java-ajoympäristö on käytössä, mutta ei erikseen kerro tätä. Siispä mikäli asennuksessa on ongelmia, kannattaa tarkistaa Java-ajoympäristö valikosta Työkalut → Asetukset → OpenOffice.org → Java. Toimivuus on testattu versiolla OpenOffice.org 3.4.1 ja Java-ympäristöllä Sun Microsystems Inc. 1.6.0_37.

1.2.2 GeoGebran käyttö optimoinnissa

Paras tapa oppia käyttämään GeoGebraa lienee leikkiä sillä: piirtää suoria ja muita funktioita käsin ja yhtälöin, ratkaista yhtälöpareja ja funktioiden nollakohtia graafisesti, piirtää funktio derivaattoineen samaan kuvaan ja niin edelleen. Tämä voi auttaa myös mahdollisten matematiikantehtävien ratkaisemisessa ja ymmärtämisessä. GeoGebra on hyvin vakaa eikä mene helposti rikki. Tästä huolimatta alla on lyhyt ohjeistus GeoGebran käyttöön. Suuri osa ohjelman käyttämisestä jätetään kuitenkin lukijan oman tutkimuksen varaan. Tämän ei pitäisi olla vaikeaa, koska GeoGebra on saatavilla myös suomeksi ja internetissä on laaja ohjeistus osoitteessa
`http://wiki.geogebra.org/en/Manual%3AMain_Page?note=fi` ,
johon pääsee myös GeoGebran 'Opastus'-valikon kautta.

Perusnäkymä GeoGebrassa koostuu piirtoalueesta, algebraikkunasta, syöttökentästä ja yläreunan työkaluista. Yksinkertaisin näistä lienee syöttökenttä, johon voi syöttää muiden muassa yhden muuttujan funktioita ja kartioleikkauksia. Erilaisten kuvioiden määrä on suuri, joten niitä ei käydä tässä. Syötetyt kuviot näkyvät piirtoalueella ja algebrakentässä. Algebrakentässä näkyviä lausekkeita voi muuttaa tuplaklikkaamalla niitä ja niillä on myös suuri määrä esittämistä koskevia ominaisuuksia, joihin pääsee käsiksi hiiren oikealla näppäimellä. Työkalurivi sisältää valikoita erilaisista melko usein tarvittavista ominaisuuksista ja kuvioista. Yleisimpiä niistä lienevät 'siirrä'- ja 'siirrä piirtoaluetta'-työkalut.

Koska GeoGebra on kaksiulotteinen piirto-ohjelma, se soveltuu optimointitehtävien ratkaisemiseen korkeintaan kahden (riippumattoman) muuttujan tehtävissä. Luonnollisesti saadut ratkaisut on saatu graafisella ratkaisemisel-

la tai mahdollisesti raa'an voiman menetelmän erikoistapauksena. On myös yleistä, että ratkaisuna saadaan vain likiarvo. Kaikkein parhaiten GeoGebra soveltuu yksiulotteiseen optimointiin, koska silloin voidaan yhdellä komennolla etsiä funktion suurin tai pienin arvo. Lineaarinen optimointi onnistuu myös melko hyvin, koska silloin kärkipisteet ovat selkeitä ja ne voidaan laskea tarkasti. Epälineaarinenkin optimointi onnistuu, mutta tarkkuudesta voi tulla ongelma. GeoGebran optimointikäyttöä on käsitelty enemmän esimerkissä 15 sivulla 53.

1.2.3 SAGEn käyttö optimoinnissa

Uusien ohjelmistojen käyttöönotto on aina hieman haasteellista ja ohjelmiin totuttelu vie aikaa. Niinpä on hyvä tietää, että windows-ympäristössä SAGEn asentaminen vie kokemattomalta hetken jos toisenkin. Ohjelmisto nimittäin toimii virtuaalikoneena. Saatavilla on luultavasti useita käypiä vaihtoehtoja, mutta tätä työtä tehdessä on käytetty Oracle VM VirtualBox -ohjelmaa, jonka voi ladata ilmaiseksi osoitteesta

`https://www.virtualbox.org/wiki/Downloads` .

SAGEn ja VirtualBoxin sivuilta ja ohjekirjoista löytyvät tarvittavat tiedot asennukseen. Ne kannattaa lukea tarkkaan. Suosittelen myös käyttämään SAGEa internetselaimen välityksellä, koska ääkköset ovat erikoismerkkejä ja voivat muuten aiheuttaa ongelmia ohjelmalle. Varsinaisesta SAGEn käytöstä on myös suomenkielinen opas osoitteessa

`http://users.utu.fi/lhjruo/sage/` .

Kyseessä on Lauri Ruotsalaisen pro gradu -tutkielma ja siihen liittyvä pikaopas, joten informaation puutteesta tuskin tulee ongelmaa. Jos kuitenkin tarvitaan lisätietoa, internetistä löytyy sitä runsaasti. SAGE myös muistuttaa melko paljon Mathematicaa, joten jos kyseinen ohjelmisto on tuttu, SAGEn käytön opettelu on melko nopeaa. Tässä tutkielmassa osa kuvista on tehty SAGEn avulla.

Koska SAGE on laaja ohjelmisto ja nyt keskitymme optimointiin, keskitytään seuraavassa lähinnä ohjelmiston optimointiin liittyvistä toiminnoista. Aloitetaan kuitenkin muutamalla ohjelmiston käytön kannalta tärkeillä huomioilla.

- SAGEn käynnistämiseksi Windows-ympäristössä käynnistä ensin VirtualBox tai vastaava, lataa ohjelmaan SAGE ja käynnistä se.
- SAGEa kannattaa käyttää internet- selaimen kautta, koska muuten voi esiintyä ongelmia ääkkösten ja muiden erikoismerkkien kanssa. Tämä onnistuu käynnistämällä SAGE, käynnistämällä selain ja menemällä osoitteeseen

  `http://localhost:8000/home/admin/` .

- SAGEn dokumentit ovat ”worksheet”-nimisiä ja ne koostuvat soluista. Solut ovat yhden tai useamman rivin kattavia komentosarjoja, jotka voidaan suorittaa kerralla. Solu voi sisältää kaikkea yksinkertaisista ’1+2’-laskuista differentiaalilaskentaan ja monimutkaisiin kuvionpiirtämiskäskyihin. Solun käskyt voidaan suorittaa näppäinyhdistelmällä ’SHIFT+ENTER’.

- Desimaalipilkun tilalla käytetään pistettä. Kertomerkit on kirjoitettava näkyviin ja siinä käytetään asteriskia, eli tähteä *.

- Eräs kätevimmistä komennoista SAGEssa on ’?’, eli kysymysmerkki. Jos et ole varma jostakin komennosta, kirjoita komento ja sen perään kysymysmerkki; esimerkiksi ”plot?”. Tällä tavalla saat tarkan kuvauksen koko komennosta kaikkine muuttujineen. Suuri osa tiedosta on todennäköisesti turhaa, mutta myös tärkeää tietoa on. Ohje sisältää myös esimerkkejä komennon käytöstä. Huono puoli ohjeessa on, että hyödyllisen ja hyödyttömän tiedon erottaminen voi vaatia harjaantumista tai jonkinlaista käsitystä ohjelmoinnista.

Koska SAGE on matemaattinen ohjelmisto, siinä on optimoinnille käteviä työkaluja, kuten derivaatan laskeva komento. Hieman soveltaen tätä voidaan käyttää maksimin tai minimin löytämisessä esimerkiksi pykälässä 2.2.5 kuvatun yksiulotteisen funktion tapauksessa.

SAGE sisältää myös erityisesti optimointiin tarkoitettuja työkaluja, kuten konveksin optimoinnin (termiä selittää ehkä parhaiten kuva 5.8). Ikävä kyllä SAGEn konveksi optimointi ymmärtää tehtävän vain matriisimuodossa, kuten esimerkissä 12 sivulta 41 lähtien. Kyseisessä esimerkissä optimointi on käyty läpi vaihe vaiheelta, joten sen seuraaminen on luultavasti paras ohje aloittelevalle käyttäjälle. On myös suositeltavaa, että aloitteleva käyttäjä ”leikkii” ohjelmalla, eli kokeilee itseään kiinnostavia asioita samalla kun lukee ohjeita. Tämä luultavasti nopeuttaa oppimista huomattavasti.

1.2.4 AMPL

On saatavilla myös ilmaisia versioita nimenomaan optimointiin tarkoitetuista ohjelmistoista ja yksi niistä on AMPL, jonka opiskelijaversio on ladattavissa tai käytettävissä internetin välityksellä ilmaiseksi. Tarvittavaa tietoa löytyy osoitteesta

`http://www.ampl.com/DOWNLOADS/index.html` ,

missä on myös linkit ohjelmiston ohjekirjan ensimmäiseen lukuun sekä ohjelmiston ladattavaan ja nettiversioon. Ladatun version käyttäminen ohjekirjan kuvaamalla tavalla on melko kömpelöä, joten on suositeltavaa käyttää ohjelmaa osoitteessa

`http://ampl.com/cgi-bin/ampl/amplcgi` .

Tämä vaatii ohjelman lisenssisopimuksen hyväksymisen.

Internetissä käytettävässä versiossa ei ole tallennustoimintoa, joten käyttäjä joutuu itse tallentamaan työnsä oman harkintansa mukaan. Tämä ei kuitenkaan ole iso ongelma, sillä optimointitehtävät ovat muodossa, jota yksinkertaisimmatkin tekstinkäsittelyohjelmat ymmärtävät. On suositeltavaa käyttää melko yksinkertaista tekstinkäsittelyohjelmaa AMPL:n kanssa, koska kehittyneempien ohjelmien muotoilut voivat häiritä optimointiohjelmaa. Esimerkki AMPL:en käytöstä sisältyy esimerkkiin 13 alkaen sivulta 44. Esimerkissä käydään läpi tärkeimmät komennot ja toimintaperiaatteet. Kaiken kaikkiaan AMPL:ssä on melko korkean tason tekstipohjainen käyttöliittymä, joten sen käytön oppii melko nopeasti. Lisää helppoutta tuo se, että tehtävän muoto noudattaa melko tarkasti yleisen optimointitehtävän muotoa, jota käytetään myös tässä teoksessa.

1.2.5 Muista optimointiohjelmistoista

Yllä mainittujen optimointiin pystyvien ohjelmistojen listaus on vain pieni pintaraapaisu aiheeseen ja ainakin osittain vastaavia ohjelmistoja on suuri määrä. Niiden kaikkien käsitteleminen edes pintapuolisesti vaatisi oman kirjansa ja niinpä niitä ei nyt tarkastellakaan. Kuvaavaa kuitenkin on, että Pursiheimo [9] mainitsee 20 optimointiohjelmistoa, joista yhdelle viisi eri versiota ja wikipedia[4] mainitsee yli 60 optimointiohjelmistoa. Osoitteessa

`http://plato.asu.edu/sub/pns.html`

on myös englanninkielinen sivusto, jossa on tarjolla puumainen opas parhaan optimointiohjelmiston valintaan sekä muuta tietoa optimoinnista. Osoitteesta

`http://www.neos-server.org/neos/`

taas löytyy muun muassa useita eri ratkaisimia monenlaisille optimointitehtäville. Sivusto on englanniksi ja ratkaisinten oikea käyttö vaatii hieman perehtymistä ratkaisimen syötteisiin ja toimintaan. Tästä syystää ratkaisinten yhteyteen on liitetty niiden käyttöohjeet. Jotkin sivuston toiminnot vaativat käyttäjätunnuksen ja ratkaisinten käyttö vaatii käyttöehtojen hyväksymisen.

Ohjelmistoa valitessa kannattaa huomioida sen käyttösopimus, ja siksi on suositeltavaa aluksi käyttää GPL eli GNU General Public License lisenssillä varustettuja ohjelmia, koska niiden levittäminen on melko vapaata. Muilla lisensseillä varustettuja ohjelmia käytettäessä on suositeltavaa lukea lisenssi, vaikka se usein on työlästä ja hyväksytään lukematta. Optimointia opiskeltaessa tai tehtäessä yrityksen laskuun on todennäköistä, että törmätään ainakin yhteen seuraavista ohjelmistoista. Ne esitellään lyhyesti niiden yleisyyden vuoksi, vaikka ne onkin aiemmin jo mainittu.

Mathematica on hyvin laaja ja tehokas matematiikkaohjelmisto, joka sisältää myös optimointirutiineja. Sitä ei kuitenkaan ole tarkoitettu laajamittaiseen optimointiin.

[4] http://en.wikipedia.org/wiki/List_of_optimization_software

Matlab on matematiikkaohjelmisto, joka on tarkoitettu ensisijaisesti vektorien ja matriisien käsittelyyn. Se pystyy kuitenkin myös optimointiin ja sisältää valmiita optimointimalleja. Laaja suomenkielinen käyttöopas löytyy osoitteesta
`http://math.aalto.fi/~apiola/matlab/opas/lyhyt/perusteet.html`

LINDO on lineaariseen optimointiin kehitetty ohjelmisto. Sitä on saatavilla useina versioina, jotka voivat käsitellä jopa 100 000 muuttujan optimointitehtäviä.

Luku 2

Yleistä optimoinnista

Optimointi on matematiikan osa-alue jossa pyritään selvittämään, mikä olisi paras mahdollinen päätös jossakin tilanteessa. Joskus tosin on järkevää vain yrittää löytää ”riittävän hyvä” ratkaisu, jos parhaan ratkaisun etsiminen on erityisen vaikeaa. Optimointia voidaan periaatteessa soveltaa mihin tahansa tilanteeseen, jossa on tehtävä päätös. On sitten aivan eri asia, onko soveltaminen järkevää, koska ongelma on analysoitava, jotta siihen voidaan soveltaa optimoinnin menetelmiä. Esimerkiksi järkevän sijoituskohteen löytäminen lottovoitolle voi hyvinkin olla optimoinnin soveltamisen arvoista. Toisaalta ostettavan tikkarin maun valintaan optimointia tuskin kannattaa soveltaa.

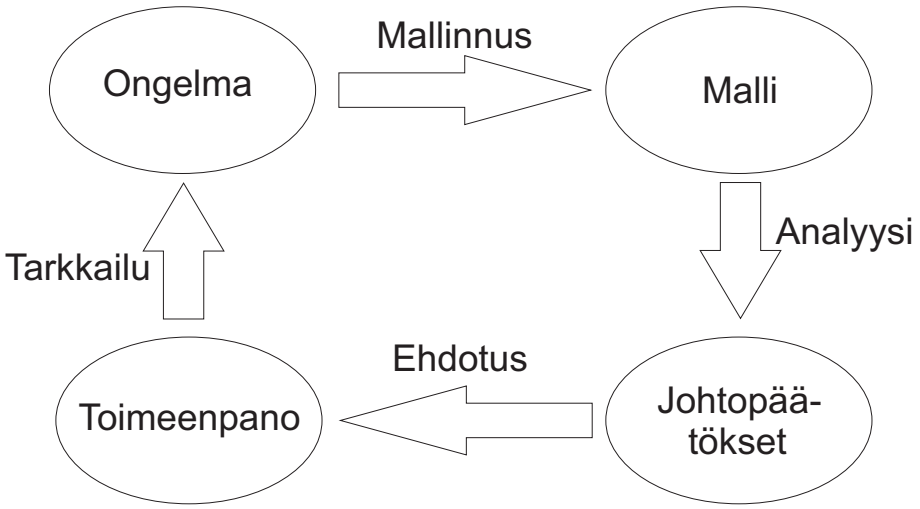
Tässä luvussa tutustutaan optimoinnin perusteisiin ja yleisiin lainalaisuuksiin. Jos tämä tutkielma on pintaraapaisu optimointiin, tämä luku on pintaraapaisu pintaraapaisuun ja sisältää aivan helpoimpia perusteita optimoinnista. Tämä ei tietenkään tarkoita sitä, että perusteet eivät olisi tärkeitä tai että niiden jälkeen esiteltävä asia olisi erityisen vaikeaa.

2.1 Operaatioanalyysistä

Optimoinnissa on käytännössä aina kyse siitä, että halutaan valita paras ratkaisu kyseessä olevaan päätöksenteko-ongelmaan. Jotta optimointia voitaisiin soveltaa todelliseen ongelmaan tai toimintaan, on se ensin muutettava enemmän tai vähemmän matemaattiseen muotoon. Tätä kutsutaan mallintamiseksi, ja eräs mallinnustyökalu on operaatioanalyysi.

Operaatioanalyysi on oikeastaan jatkuva prosessi ja ikuinen silmukka, koska muuttuvassa maailmassa mikään ratkaisu ei ole ikuisesti paras mahdollinen. Analyysiä ei tietenkään tarvitse jatkaa ikuisesti, jos tyydytään ratkaisuun, joka on jollakin hetkellä paras tai riittävän hyvä. Seuraava kuva havainnollistaa operaatioanalyysin toimintaa.

Operaatioanalyysissä lähdetään liikkeelle ongelmasta ja pyritään rakentamaan siitä matemaattinen malli. Näitä malleja käsitellään perusteellisem-



Kuva 2.1: Operaatioanalyysin kulku Leipälää mukaillen [6].

min seuraavassa luvussa. Kun malli on valmis, seuraa sen analysointi. Tässä vaiheessa pyritään selvittämään, mitä johtopäätöksiä mallista voidaan tehdä, kuten mahdolliset optimaaliset ratkaisut. Nyt on kuitenkin käytetty vain todellisuuden mallia, joten johtopäätökset voivat olla hyvinkin kaukana todellisuudesta ja siten huonoja tai käyttökelvottomia. Jos johtopäätökset kuitenkin näyttävät järkeviltä, voidaan niistä laatia ehdotus päätöksentekijälle, joka saattaa ottaa ehdotuksen käyttöön. Toimeenpanon jälkeen järjestelmää tarkkaillaan ja arvioidaan ratkaisun toimivuutta. Mikäli siihen ei olla tyytyväisiä, voidaan aloittaa operaatioanalyysissä uusi kierros ja parantaa mallia.

2.2 Optimointimallit

Tässä osiossa käsitellään edellä mainittuja malleja ja niiden tyyppejä. Käsitellään myös kertauksenomaisesti hieman graafista ratkaisua ja kärkipisteitä.

2.2.1 Tehtävän määrittely

Operaatioanalyysin mallinnusvaiheessa rakennetaan matemaattista mallia päätöksenteko-ongelmasta. Tätä mallia rakennettaessa on otettava huomioon muiden muassa seuraavat asiat:

1. Päätöksentekijän tekemät päätökset ja niitä rajoittavat ehdot.
2. Kriteerit, joilla voidaan arvioida tehtyjen päätösten paremmuutta.
3. Mallin tyyppi ja kohdefunktio.
4. Muuttujien, parametrien ja vakioiden määrittely.

Päätöksentekijän päätöksiä ja niiden rajoitteita voidaan käsitellä esimerkin avulla. Tarkastellaan kioskia, jonka pitäjä on päätöksentekijä. Hänen on tilattava myytävät tavarat ja hinnoiteltava ne sopivasti. Tässä tilattavan tavaran määrä tai myyntihinta ovat hänen päätöksiään. Edellä mainitun tavaratilauksen ehtoja voivat olla esimerkiksi tavaran saatavuuden rajoittuneisuus tai kioskin varastotilan koko. Hinnoittelua taas rajoittaa muun muassa asiakkaiden ostovoima.

Jos päätöksiä ei voida verrata toisiinsa, ei voida tietää miten hyviä ne ovat. Edellisen kappaleen kioskiesimerkissä tavaran tilauksista tehtyjä päätöksiä voidaan verrata esimerkiksi varaston riittävyyttä tarkkailemalla: liian iso varasto aiheuttaa kuluja, mutta jos tavara on loppu, ei saada tuloja ja asiakkaat ovat tyytymättömiä. Tavaroiden hinnoittelussa kriteerinä voidaan käyttää myynnistä saatua tuottoa tiettynä aikana.

Mallin tyypillä tarkoitetaan tässä kohtaa sitä, yritetäänkö löytää maksimi vai minimi. Esimerkiksi tuottoja yleensä maksimoidaan ja kuluja minimoidaan. Maksimoitavat ja minimoitavat arvot seuraavat mallissa jonkinlaista kaavaa, ja tätä kutsutaan kohdefunktioksi. Maksimointi ja minimointi liittyvät nimenomaan kohdefunktion arvoon. Jos mallista halutaan minimoida tai maksimoida arvoa, joka ei ole kohdefunktion arvo, jossakin vaiheessa on tapahtunut virhe, ja vähintäänkin kohdefunktio on mietittävä uudelleen.

Mallissa erilaiset asiat saavat lukuarvoja, ja nämä asiat voidaan luokitella muuttujiin, parametreihin ja vakioihin. Muuttujien arvoista pyritään löytämään kohdefunktion kannalta optimaaliset, parametreille voidaan antaa erilaisia vakioarvoja, jotta saadaan tietoa mallin toimivuudessa erilaisissa oloissa, ja vakioilla on aina sama vakioarvo. Erityisesti muuttujia, jotka kuvaavat päätöksentekoa kutsutaan päätösmuuttujiksi, eikä malleissa tavallisesti esiinnykään muita muuttujia, vaan ne ovat parametreja. Tähän sääntöön on kuitenkin useita poikkeuksia, kuten esimerkissä 11 sivulla 39 ja stokastinen optimointi, joka mainitaan simulointi–osiossa sivulla 25.

Esimerkki 1. Maanviljelijä haluaa aidata mahdollisimman suuren alueen. Alueen on oltava suorakulmion muotoinen ja sen aitaamiseen on käytettävissä 300 metriä aitaa. Muodostetaan nyt optimointitehtävä tilanteesta.

Merkitään aitauksen pinta-alaa A:lla ja se tulee maksimoida. Kun aitauksen leveys on x ja pituus y metriä, selvästi $A = x \cdot y$. Käytettävissä on 300 m aitaa, joten $2x + 2y \leq 300$. Tämä saadaan muotoon $x \leq 150 - y$ On myös selvää, että $x > 0$ ja $y > 0$, jotta aitauksella yleensä olisi pinta-ala. Näin saadaan optimointitehtävä

$$\begin{array}{ll} \max & A = x \cdot y \\ s.t. & x \leq 150 - y \\ & x > 0 \\ & y > 0 \end{array} .$$

Esimerkin tehtävän ensimmäinen rivi kertoo tehtävän tyypin ja kohdefunktion. Lyhenteen ”s.t.” (subject to) jälkeen on kerrottu tehtävän rajoit-

teet, joiden puitteissa ratkaisun tulee olla. Koska tehtävässä on kaksi muuttujaa, kyseessä on kaksiulotteinen optimointitehtävä ja siis kaksiulotteinen kohdefunktio. Tämä ei kuitenkaan ole mikään ongelma, sillä tällaisia tehtäviä on voitu ratkaista jo pitkän matematiikan geometrian kurssilla. [4]

2.2.2 Realistiset mallit

Edellisessä esimerkissä oli kaksi muuttujaa ja kolme rajoitusta. Todellisuudessa ollaan harvoin näin onnekkaita, vaan muuttujia ja rajoituksia on huomattavasti enemmän. Alla on esimerkki, joka on laajahko mutta selvästi käsitettävissä.

Esimerkki 2. Öljynjalostamossa valmistetaan raakaöljystä bensiiniä, lentobensiiniä ja voiteluöljyä. Raakaöljyä saadaan Lähi-idästä ja Etelä-Amerikasta. Jokaisesta tynnyristä Lähi-idän mustaa kultaa saadaan $0,4$ tynnyriä bensiiniä, $0,28$ tynnyriä lentobensiiniä ja $0,2$ tynnyriä voiteluöljyä. Eteläamerikkalaiselle raakaöljylle vastaavat luvut ovat $0,2$ tynnyriä bensiiniä, $0,4$ tynnyriä lentobensiiniä ja $0,3$ tynnyriä voiteluöljyä. Lähi-idästä on saatavilla 9000 tynnyriä öljyä päivässä 25\$ tynnyrihintaan ja Etelä-Amerikasta taas 8000 tynnyriä 20\$ kappalehintaan. Jalostamon täytyy tuottaa päivässä 2100 tynnyriä bensiiniä, 2400 tynnyriä lentobensiiniä ja 500 tynnyriä voiteluöljyä. Miten tämä voidaan tehdä mahdollisimman pienillä raaka-ainekustannuksilla?

Valitaan aluksi päätösmuuttujiksi Lähi-idästä ja Etelä-Amerikasta hankittavan ja jalostettavan öljyn määrä tuhansina tynnyrinä:

1. x_1:Lähi-idästä tuodun öljyn määrä tuhansina tynnyreinä.

2. x_2:Etelä-Amerikasta tuodun öljyn määrä tuhansina tynnyreinä.

Alaindeksi vain erottaa muuttujat toisistaan, ja se luetaan numerona muuttujan perään (esimerkiksi "x yksi"). On tärkeää, että päätösmuuttujista esitetään selkeästi merkitys ja laatu. Muut tarvittavat suureet ovat parametreja, jotka tällä kertaa ovat vakioita. Päätösmuuttujia voitaisiin myös merkitä eri kirjaimilla, kuten x ja y tai lähes millä tahansa muulla tavalla, kunhan merkinnät ovat selkeitä.[1]

Seuraavaksi mietitään rajoituksia, ja miettiminen on hyvä aloittaa ilmeisimmistä. Jalostamo ei voi tuoda negatiivista määrää öljyä, joten saadaan ehdot

$$x_1 \geq 0, \quad x_2 \geq 0.$$

Tällaisia rajoituksia kutsutaan ei-negatiivisuusrajoituksiksi, ja ne ovat hyvin tavallisia, mutta joskus päätösmuuttujat voivat saada vain kokonaislukuarvoja tai vain arvoja 1 ja 0. Tällaisia tapauksia käsitellään epälineaarisen optimoinnin luvussa.

[1]Päätösmuuttujien merkintään kävisivät jopa kirjainyhdistelmät tai erikoismerkit kuten ♠ ja ★. Niitä ei kuitenkaan yleensä käytetä.

Muita rajoituksia tuovat tuotantovaatimukset ja saatavuuden rajallisuus. Tuotantovaatimuksista aiheutuvat seuraavat kolme rajoitusta:

$$\begin{aligned} \text{bensiini:} &\quad 0,4x_1 + 0,2x_2 \geq 2{,}1 \\ \text{lentobensiini:} &\quad 0,28x_1 + 0,4x_2 \geq 2,4 \\ \text{voiteluöljy:} &\quad 0,2x_1 + 0,3x_2 \geq 0,5. \end{aligned}$$

Raaka-aineiden saatavuus tuottaa vielä kaksi rajoitetta:

$$\begin{aligned} \text{Lähi-itä:} &\quad x_1 \leq 9 \\ \text{Etelä–Amerikka:} &\quad x_2 \leq 8. \end{aligned}$$

Vain kohdefunktio ja tehtävän tyyppi ovat enää määrittelemättä. Koska pyritään pieniin raaka-ainekustannuksiin, minimoidaan funktiota $25x_1 + 20x_2$, joka kertoo raaka-ainekustannukset tuhansina dollareina. Ongelmasta on siis muodostettu optimointitehtävä

$$\begin{aligned} \min \quad & 25x_1 + 20x_2 \\ s.t. \quad & 0,4x_1 + 0,2x_2 \geq 2,1 \\ & 0,28x_1 + 0,4x_2 \geq 2,4 \\ & 0,2x_1 + 0,3x_2 \geq 0,5 \quad . \\ & x_1 \leq 9 \\ & x_2 \leq 8 \\ & x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

Edellisestä esimerkistä saa hieman viitteitä todellisten ongelmien monimutkaisuudesta, mutta kehitetäänpä sitä vielä lähemmäs todellisuutta lisäämällä monimutkaisuutta. Tarkastellaan öljy-yhtiötä, jolla on useita jalostamoja, joista yhden toimintaa on optimoitu yllä. Öljyä saadaan toimitettua jalostamoihin useista eri lähteistä ympäri maailmaa. Jokaisen lähteen öljystä voidaan valmistaa eri määrät eri tuotteita ja myöskin eri lähteiltä eri jalostamoille erisuuruiset rahtikustannukset. Jo 5 jalostamolla ja 25 lähteellä pelkkiä kuljetuskustannuksia on 125 erilaista. Myöskään kaikki jalostamot eivät ole samanlaisia, vaan joissakin niistä jonkin tuotteen valmistus on tehokkaampaa kuin toisissa eikä kaikkia tuotteita pystytä edes valmistamaan jokaisessa jalostamossa. Myöskään öljyn markkinahinta tai saatavuus eivät ole vakioita, vaan muuttuvat ajan kuluessa, joten optimointiin kannattaa liittää herkkyysanalyysi, josta kerrotaan hieman sivuilla 28 ja 92. Erittäin suurissa realistisissa malleissa voi olla miljoonia muuttujia ja satojatuhansia rajoituksia.

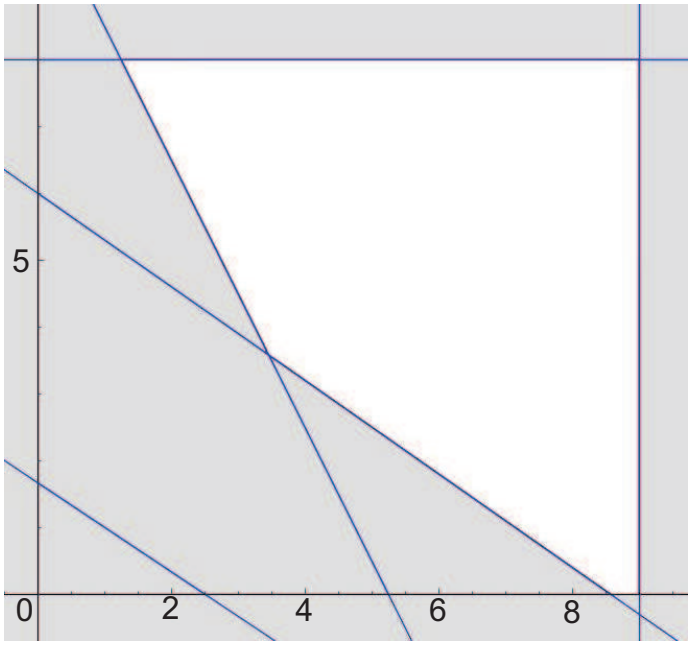
2.2.3 Graafinen ratkaisu ja kärkipisteet

Eräs havainnollisimmista ratkaisumenetelmistä on graafinen ratkaisu, koska silloin abstraktista asiasta saadaan konkreettinen kuva. Se saa myös jotkin

tehtävät muuttumaan triviaaleiksi, vaikka ratkaisijan matemaattiset taidot olisivat vaatimattomat. Esimerkiksi tehtävän $\min x^2$ ratkaisu graafisesti on piirtää funktion kuvaaja ja katsoa, milloin se on alimmillaan. Kaksiulotteisessa tapauksessa tarkasteltavan funktion kuvaaja voidaan esittää kolmiulotteisena pintana. Tällainen pinta voidaan mallintaa tietokoneella tai se voidaan esittää kartan korkeuskäyrien tapaan, jolloin kaikilla korkeuskäyrän pisteillä funktio saa saman arvon.

Jos graafista ratkaisua laaditaan käsin, ensimmäiseksi kannattaa laskea rajoittavien suorien (tai käyrien) leikkauspisteet, jotta akseleiden asteikot voidaan valita järkevästi. Sen jälkeen kannattaa piirtää koordinaatisto ja rajoittavat suorat (tai käyrät) ja selvittää, kummalla puolella kutakin käyrää on sallittu alue, eli kummalla puolella käyrää sen kuvaama rajoite on voimassa. [5]

Esimerkki 3. Ratkaistaan esimerkin 2 optimointitehtävä graafisesti piirtämällä x_1x_2-koordinaatistoon rajoituksia kuvaavat suorat. Näin on tehty kuvassa 2.2.

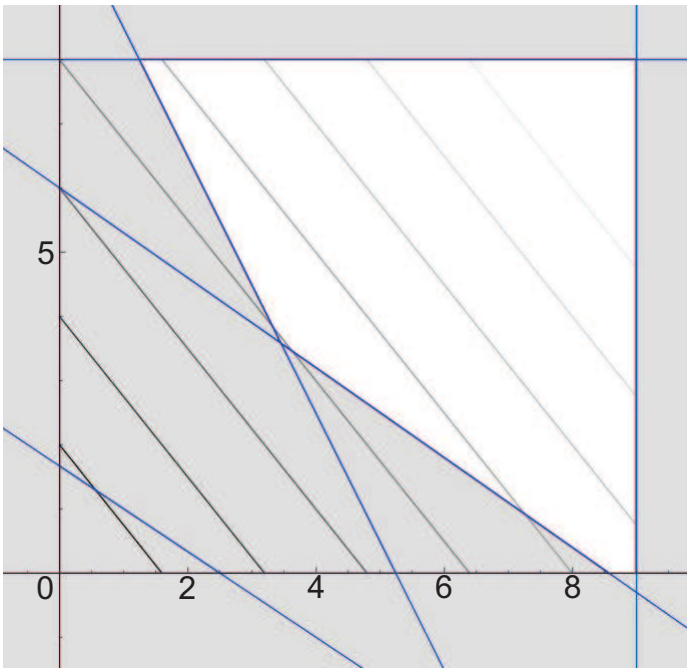


Kuva 2.2: Öljynjalostamoesimerkin rajoitukset ja sallittu alue.

Nyt kuvassa valkoisena näkyvällä alueella kaikki rajoitukset ovat voimassa, eli sen alueen pisteet ovat *sallittuja* pisteitä[2]. Piirretään seuraavaksi kuvaan kustannuksia kuvaavia tasa-arvokäyriä, kuten kuvassa 2.3 on tehty.

[2]Joskus sallitusta pisteestä käytetään myös nimitystä *käypä* piste.

Tällaisen käyrän pisteitä tarkasteltaessa kustannukset ovat siis samat kaikissa pisteissä.



Kuva 2.3: Öljynjalostamoesimerkin rajoitukset ja sallittu alue. Tasa-arvokäyrät piirretty osaan ensimmäistä neljännestä

On selvää, että mitä enemmän öljyä ostetaan, sitä suuremmat ovat kustannukset. Niinpä tasa-arvokäyrät osoittavat sitä pienempiä kustannuksia, mitä lähempänä ne ovat origoa. Niinpä minimi ja samalla optimi saavutetaan viisikulmion kärkipisteessä $\left(3\frac{6}{13}, 3\frac{15}{26}\right)$. Kustannukset näillä arvoilla ovat

$$25 \cdot 3\frac{6}{13} + 20 \cdot 3\frac{15}{26} = 158\frac{1}{13} \approx 158,077$$

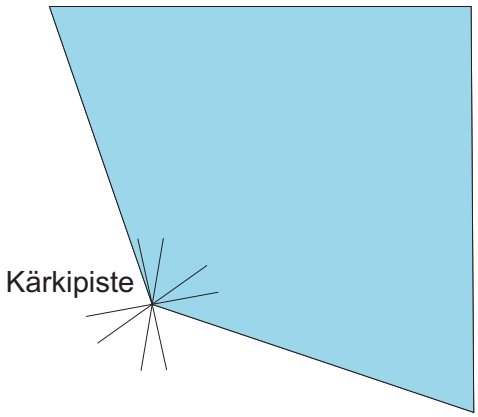
ja yksikkönä on tuhatta dollaria päivässä.

Edellisessä esimerkissä saadaan selkeästi rajattu alue, jolta parasta arvoa voidaan hakea. On kuitenkin mahdollista, joskin poikkeuksellista, että rajoitteet ovat niin tiukat, ettei mikään piste toteuta niitä. Tällöin optimointitehtävällä ei ole ratkaisua. Esimerkissä myös havaitaan, että optimaalinen arvo saavutettiin sallittujen arvojen joukkoa rajaavalla monikulmiolla ja tarkemmin sanottuna yhdessä sen kärjistä. Tästä päästään seuraaviin käsitteisiin, jotka pätevät lineaariseen optimointiin.

Reunapiste on sallittu piste, jossa ainakin yksi epäyhtälörajoitus toteutuu yhtäsuuruutena.

Sisäpiste on sallittu piste, jossa kaikki epäyhtälörajoitteet toteutuvat epäyhtälöinä, eivät yhtälöinä.

Kärkipiste on sallittu piste, jos sitä ei voi esittää kahden sallitun pisteen aitona konveksina yhdelmänä. Tämä tarkoittaa sitä, että jos kärkipiste otetaan minkä tahansa janan keskipisteeksi, janan toisessa päässä oleva piste ei ole sallittu piste. Tätä on havainnollistettu kuvassa 2.4. Joskus kärkipisteitä kutsutaan myös kulmapisteiksi.



Kuva 2.4: Havainnollistus kärkipisteen määritelmästä. Erään kärkipisteen kautta on piirretty muutamia janoja, joiden toisessa päässä on sallittu piste.

Kärkipisteet ovat tärkeitä, koska "jos lineaarisella optimointitehtävällä on yksikäsitteinen ratkaisu, saavutetaan se kärkipisteessä."[6] Tästä seuraa myös, että lineaarinen optimointitehtävä voidaan ratkaista tutkimalla funktion arvot kaikissa kärkipisteissä, ja valitsemalla niistä paras, ja tätä on saatettu joillakin matematiikan kursseilla käyttääkin.

Suurissa ja moniulotteisissa ongelmissa graafinen ratkaisu on melko käyttökelvoton piirtämisongelmien vuoksi. On kuitenkin syytä huomata, että esimerkiksi kärkipisteen määrittelyssä ei olla riippuvaisia graafisesta esityksestä, vaan niitä voidaan laskea ilman kuvaa. Jätämme kuitenkin nämä tarkastelut myöhemmäksi.

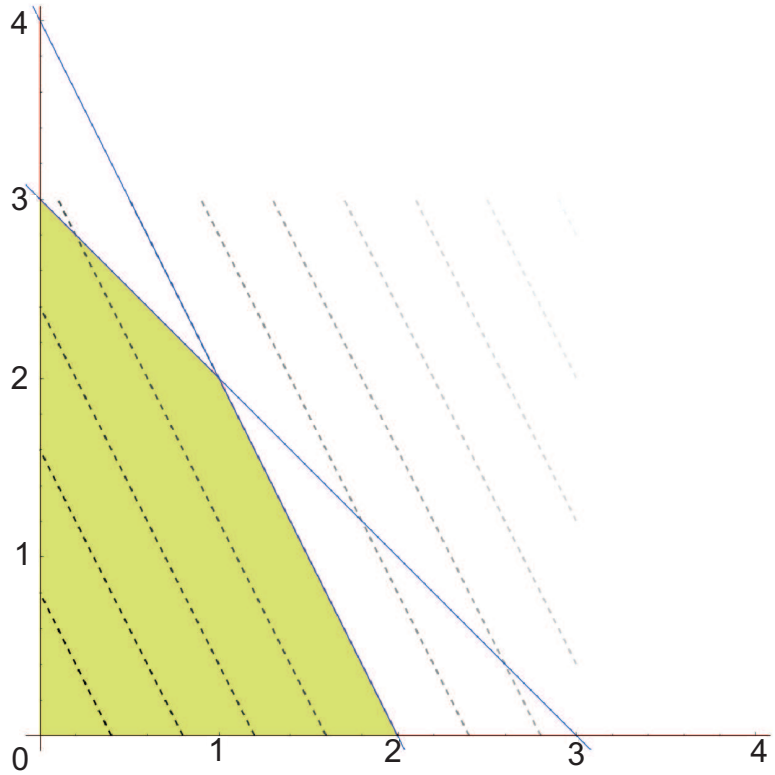
2.2.4 Ratkaisujen lukumäärästä

Tähän mennessä on käsitelty vain esimerkkejä, joiden ratkaisuna on ollut yksikäsitteinen optimi yksikäsitteisillä muuttujien arvoilla. Näin ei kuitenkaan aina ole, kuten seuraavista esimerkeissä nähdään. Tehtävät on ratkaistu graafisesti ja niihin on merkitty optimipisteiden sijainnit.

Esimerkki 4.

$$\begin{array}{ll} \max & 2x_1 + x_2 \\ s.t. & 2x_1 + x_2 \leq 4 \\ & x_1 + x_2 \leq 3 \\ & x_1 \geq 0, x_2 \geq 0 \end{array} .$$

Nyt kaikki reunapisteet suoralla $x_2 = -2x_1 + 4$ ovat optimaalisia, kuten kuvasta 2.5 sivulla 20 nähdään. Tämä käy ilmi myös kuvasta tutkimalla katkoviivalla piirrettyjä tasa-arvokäyriä. Tässäkin tapauksessa optimi löytyy kärkipisteestä, mutta se löytyy myös joistakin muista pisteistä.



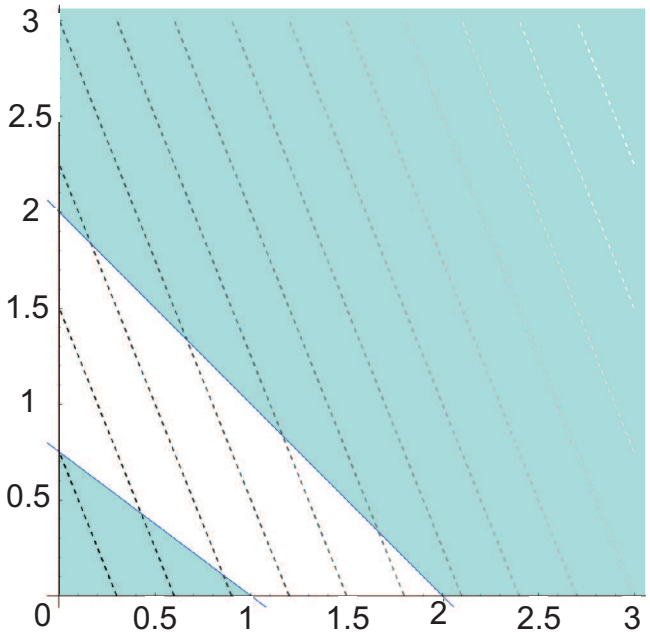
Kuva 2.5: Kuva esimerkin 4 tilanteesta.

Esimerkki 5.

$$\begin{array}{ll} \max & 5x_1 + 2x_2 \\ s.t. & x_1 + x_2 \geq 2 \\ & 3x_1 + 4x_2 \leq 3 \\ & x_1 \geq 0, x_2 \geq 0 \end{array} .$$

Alueet, joilla toinen epäyhtälörajoite ja molemmat ei-negatiivisuusrajoitteet ovat voimassa on merkitty sinertävällä kuvaan 2.6 sivulla 21. Kuitenkaan ei

ole lainkaan alueita, joilla kaikki rajoitteet olisivat voimassa. Tässä tapauksessa ei siis ole yhtään sallittua pistettä, joten optimiratkaisuakaan ei ole.

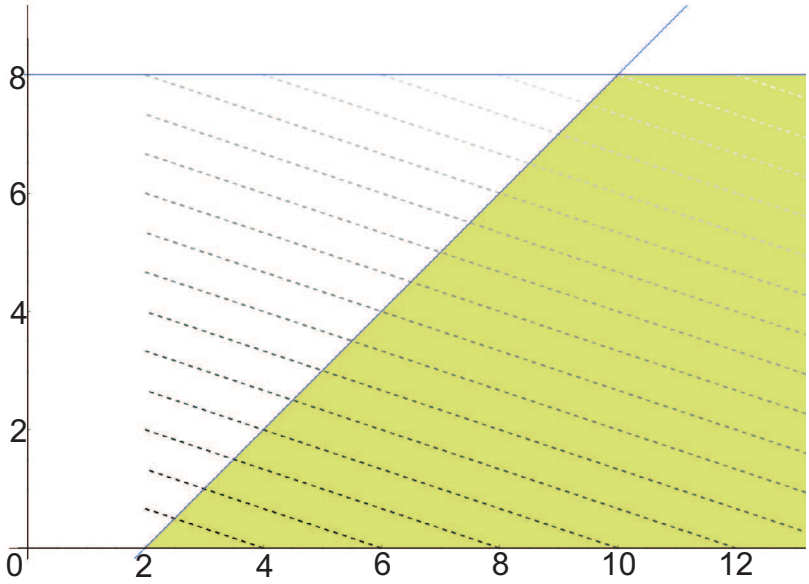


Kuva 2.6: Kuva esimerkin 5 tilanteesta.

Esimerkki 6.

$$\begin{array}{ll} \max & x_1 + 3x_2 \\ s.t. & x_1 - x_2 \geq 2 \\ & x_2 \leq 8 \\ & x_1 \geq 0, x_2 \geq 0 \end{array} .$$

Kuvaan 2.7 sivulla 22 on jälleen kerran piirretty tasa-arvokäyrät katkoviivalla ja sallittu alue vihreällä. Sallittu alue kuitenkin jatkuu vielä kuvan oikean laidan ulkopuolelle ja aina äärettömyyteen saakka. Optimiratkaisua ei siis pystytä määrittämään, koska muuttujien arvot sallivat kohdefunktion arvojen loputtoman kasvun. Tällainen tilanne on erittäin harvinainen todellisessa maailmassa, joten jos malli antaa tällaisen tuloksen, mallissa on todennäköisesti jotakin vikaa.



Kuva 2.7: Kuva esimerkin 6 tilanteesta.

2.2.5 Tehtävien luokittelusta

Asioiden luokittelu selkeyttää usein niistä puhumista, joten myös optimointitehtäviä on luokiteltu eri luokkiin. Selkeyden lisäksi luokittelu on tärkeää, koska tehtäviä ratkaistaan erilaisilla algoritmeilla ja tietokoneohjelmilla, jotka on voitu kehittää ratkaisemaan vain tietyntyyppisiä ongelmia. Esimerkiksi kärkipisteiden vertailu soveltuu vain lineaariseen optimointiin.

Tehtävien ulottuvuudet

Ehkä selkein luokitteluperuste on tehtävissä käytettävien ulottuvuuksien määrä. *Yksiulotteisissa* tehtävissä optimoitava funktio on muotoa $f(x) = \ldots$ ja tällaisia funktioita on käsitelty jo yläkoulusta lähtien ja ne voidaan esittää x, y-koordinaatistossa. Niiden optimointi on usein helppoa, koska optimi löytyy joko sallitun välin päätepisteistä, tai funktion derivaatan nollakohdista. Jos kyseessä on derivoituva lineaarinen funktio, optimi löytyy aina sallitun välin jommastakummasta päätepisteestä. On tilanteita, joissa optimia ei löydy, kuten rajoittamattoman lineaarisen funktion tapauksessa. Jopa jotkin laskimet pystyvät yksiulotteiseen optimointiin kertomalla funktion suurimman tai pienimmän arvon halutulla välillä. Näin voi tehdä esimerkiksi TI-Nspire -laskimilla käyttäen toimintoja fMin ja fMax, jotka tosin etsivät vain lokaalin ääriarvon. [15] Yksiulotteisia tehtäviä käsitellään enemmän kappaleessa 5.2 sivulla 84, mutta sellainen on jo muun muassa esimerkissä 7 sivulla 26.

Kaksiulotteinen optimointi koskee funktioita, joiden muoto on $f(x, y) = \ldots$ ja ne voidaan esittää x, y, z-koordinaatistossa. Tällöin koordinaatisto on kolmiulotteinen ja z-koordinaatilla ilmaistaan funktion saamia arvoja. Tehtävä voidaan esittää graafisesti myös x, y-koordinaatistossa piirtämällä funktion saamat arvot koordinaatistoon tasa-arvokäyrinä korkeuskäyrien tapaan. Tällöin tehtävän voi hahmottaa karttana, josta pitää löytää korkein nyppylä tai syvin kuoppa. Tällaisia optimointitehtäviä on ratkaistu ainakin joillain pitkän matematiikan kursseilla. Kaksiulotteisiin tehtäviin on jo törmätty muun muassa sivun 15 esimerkissä 2.

Moniulotteisessa tehtävässä on kaksi tai useampia ulottuvuuksia, jolloin optimoitavat funktiot voidaan kirjoittaa muotoon $f(x_1, x_2, \ldots, x_n) = \ldots$, missä n on ulottuvuuksien määrä. Kolmiulotteisen tehtävän voi hahmottaa animaationa maaston muotojen muuttumisesta, jolloin pitäisi löytää korkein tai matalin kohta maastossa tietyllä aikavälillä. Se on kuin sarja kaksiulotteisia tehtäviä. Neliulotteisen tehtävän voi yrittää hahmottaa sarjana kolmiulotteisia tehtäviä: useana samaan aikaan pyörivänä animaationa. Näin monen ulottuvuuden tehtäviä on hankala havainnollistaa, vielä useammista ulottuvuuksista puhumattakaan. Onneksi niiden hahmottamista ei vaadita niiden ratkaisemiseksi, vaan kolmannen ulottuvuuden jälkeen ratkaisujen matematiikkaan ei tule mitään järisyttävän uutta, vaan samankaltaiset laskut toimivat. On syytä huomata, ettei ulottuvuuksien määrällä ole ylärajaa. Kolmiulotteiseen tehtävään törmätään ensimmäistä kertaa esimerkissä 10 sivulla 37. Tämän jälkeen vastaan tulevat melko pian neljä- ja kuusiulotteinen tehtävä.

Ulottuvuuksien käsittämisen kannalta siis yksiulotteiset tapaukset ovat helppoja ja tuttuja, kaksiulotteiset ovat hieman oudompia ja kolmiulotteiset ovat melko vaikeita. Matemaattisesti suurin hyppy tapahtuu siirryttäessä yksiulotteisesta kaksiulotteiseen, mutta käsityskyvyn kannalta suurin siirtymä lienee tultaessa kolmi- tai neliulotteisiin tehtäviin. Matemaattinen siirtymä yksiulotteisesta moniulotteiseen on haastava, koska esimerkiksi derivaattaa ei voida laskea, vaan sen tilalle tulee gradientti, jota käsitellään myöhemmin sivulla 79. On kuitenkin tärkeää huomata, että optimoitava funktio f saa aina reaalilukuarvon, jotta funktion arvojen paremmuutta voidaan verrata.[3] Matemaattisesti voidaan siis sanoa, että f on kuvaus $\mathbb{R}^n \mapsto \mathbb{R}$, jossa n kertoo tehtävän ulottuvuuksien määrän.

Tehtävien tyypit

Optimointitehtävien luokittelussa tärkeimmät luokat ovat *lineaarinen* ja sitä vastaa *epälineaarinen* optimointi. Lineaarinen optimointi on epälineaarista yksinkertaisempaa ja sille ovat aina voimassa seuraavat viisi ehtoa.

1. Verrannollisuus: "kunkin päätösmuuttujan vaikutus kohdefunktioon ja

[3]Poikkeuksena tähän on monitavoiteoptimointi, jota käsitellään osiossa 4.2 sivulla 56.

rajoituksiin on suoraan verrannollinen päätösmuuttujan arvoon".

2. Additiivisuus: "kunkin päätösmuuttujan vaikutus kohdefunktioon ja rajoituksiin on riippumaton muiden päätösmuuttujien arvoista".

3. Jaollisuus: "kukin päätösmuuttuja on jatkuva eli se voi saada mielivaltaisia reaalilukuarvoja".

4. Deterministisyys: "tehtävän parametrit ovat vakioita eikä[4] satunnaismuuttujia".

5. Optimointitavoitteita on vain yksi. [6]

Jos yksikin yllä mainituista ehdoista jää täyttymättä, kyseessä on jokin epälineaarisen optimoinnin alaluokka. Jos toinen tai kumpikaan kahdesta ensimmäisestä ehdosta ei päde, tehtävässä on jokin epälineaarinen funktio. Jos kolmas ehto rikotaan, kyseessä on *diskreetti optimointi*, joka sisältää myös binääriset tapaukset, joissa jokin tai jotkin päätösmuuttujat voivat saada vain arvoja 1 tai 0. Jos neljäs ehto ei ole voimassa, puhutaan *stokastisesta optimoinnista* ja siihen liittyvät tilastotiede ja todennäköisyyslaskenta. Jos optimointitavoitteita on useita, on kyse *monitavoiteoptimoinnista*. Näitä tyyppejä käsitellään niille varatussa luvussa tarkemmin.

Lisäksi optimointitehtävät voidaan jakaa *rajoitteellisiin* ja *rajoitteettomiin optimointitehtäviin* sen mukaan, onko niissä yhtälö-, tai epäyhtälörajoitteita. Jos rajoitteita ei ole, kyseessä on rajoitteeton tehtävä. Jos taas tehtävässä on edes yksi rajoitus, se on rajoitteellinen tehtävä. Kaikki tähän mennessä esiintyneet tehtävät ovat olleet rajoitteellisia, mutta esimerkissä 7 käsitellään rajoitteetonta tehtävää. Näiden lisäksi on olemassa myös muita luokitteluperusteita, ja jokainen voi kehittää niitä lisää, mutta niitä ei tässä käsitellä erikseen.

[4]Kirjoitusvirhe lähdeteoksessa; pitäisi olla 'eivätkä'.

2.3 Simulointi

Sana *simulointi* tarkoittaa todellisuuden jäljittelyä. Käsite sinänsä on hyvin laaja ja sisältää menetelmiä mielikuvituksesta ja lasten leikeistä aina ilmaston muutoksen ennustamiseen supertietokoneilla. Optimoinnissa simulointi tarkoittaa lähinnä todellisuutta jäljittelevän mallin käyttöä jonkin systeemin tutkimisessa, ja apuna käytetään usein tietokonetta. Öljynjalostamoesimerkkiin simulointia voisi soveltaa antamalla raakaöljyjen hintojen ja saatavuuden vaihdella jollakin välillä tiettyjen sääntöjen puitteissa. Tällöin voitaisiin tarkastella jalostamon toimintaa aiemmin selvitetyllä optimaalisella tavalla. Tämä voi paljastaa ongelmia optimiratkaisun soveltamisessa, kuten suurten hinnanmuutosten vaikutuksen kustannuksiin.

2.3.1 Deterministiset ja stokastiset mallit simuloinnissa

Deterministinen tarkoittaa asiaa, joka noudattaa aina syy-seuraus -suhteita. Esimerkiksi kun kivi pudotetaan, se putoaa alaspäin tietyllä kiihtyvyydellä. Kiven pudottaminen on syy ja itse putoaminen sen seuraus. Öljynjalostamoesimerkki on deterministinen, koska siinä ei ole mukana satunnaisuutta: tuotantoprosessissa ei ole häiriöitä, hinnat ja saatavuus eivät muutu ja niin edelleen.[5]

Stokastiset asiat sisältävät satunnaisuutta ja ovat siten aivan erilaisia kuin deterministiset asiat. Wikipedia ilmaisee asian seuraavasti: "Stokastisilla prosesseilla tarkoitetaan ajassa sattumanvaraisesti eteneviä todellisuuden prosesseja kuvaavia matemaattisia prosesseja."[36] Täten stokastiset mallit sisältävät jonkin tai joitakin satunnaismuuttujia, joiden arvo muuttuu simuloinnin aikana. Öljynjalostamoesimerkissä näitä muuttujia voisivat olla raakaöljyjen saatavuus, joka taas heijastuisi hintaan. Tällaiset mallit mahdollistavat optimiratkaisun tutkimuksen esimerkiksi tilanteessa, jossa raaka-aineen saatavuus romahtaa.

Stokastisissa malleissa satunnaismuuttujat voivat noudattaa haluttua jakaumaa, kuten normaalijakaumaa. Tämä on järkevää, koska pienet heilahtelut tuotannossa ovat arkipäivää, mutta tuotannon tuplaaminen tai sen täydellinen pysähtyminen ovat hyvin harvinaisia. Kussakin tilanteessa muuttujille valittavaa jakaumaa kannattaa harkita huolella.

Kun malli on valmis, voidaan haluttua ratkaisua testata sillä melko nopeasti vuosien tai jopa satojen vuosien toiminnan edestä. Ikävä kyllä usein voidaan testata vain yhtä ratkaisua kerrallaan, ja toisen ratkaisun testaaminen vaatii uuden liudan simulaatioita. Joka tapauksessa testaamalla voidaan välttyä ikäviltä yllätyksiltä mallin toimivuudessa ja näin voidaan säästää runsaasti aikaa ja rahaa.

[5]Filosofiassa käydään jonkin verran keskustelua siitä, onko koko maailmankaikkeus deterministinen, jolloin esimerkiksi satunnaisuus ja vapaa tahto olisivat vain näennäisiä.

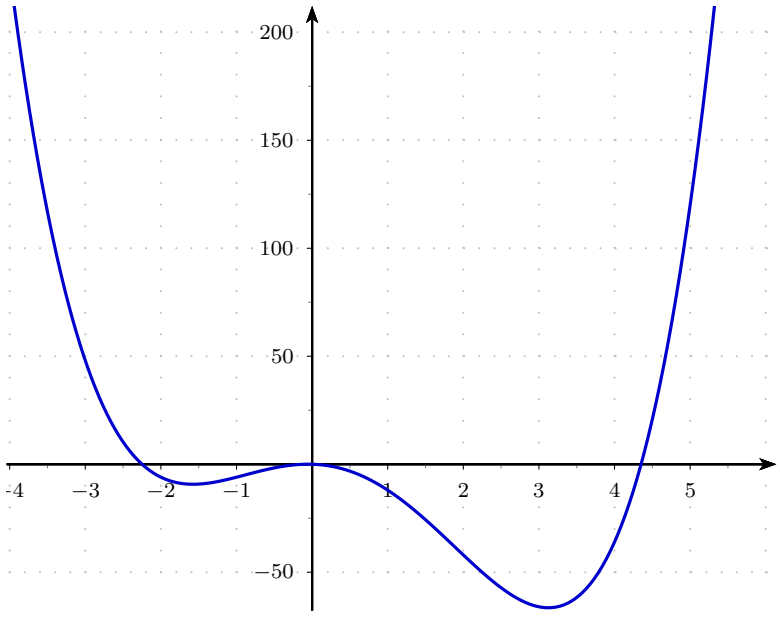
Mallin valintaa on syytä harkita tarkkaan, koska stokastisten mallien analysointi on yleisesti paljon hankalampaa kuin determinististen mallien. Niinpä usein käytetään determinististä mallia, jossa satunnaismuuttujat on korvattu parametreilla. Parametritkin saattavat simulaation aikana muuttua, jos niiden arvo noudattaa jonkin funktion arvoa, mutta arvot eivät kuitenkaan ole satunnaisia. Tällaisillakin malleilla on saatu hyviä tuloksia.

2.3.2 Lokaalit ja globaalit optimit

Yleensä optimointitehtäviä ratkaistaan tietokoneella, mikä asettaa ratkaisemiselle joitakin rajoituksia, joita asiaan perehtymätön ei välttämättä tule ajatelleeksi. Tietokoneohjelmat joutuvat toimimaan ennalta määrätyllä tavalla, joka on itse ohjelmakoodissa. Tämä voi joskus sisältää satunnaisuutta, ja on olemassa jopa oppivia ohjelmia, mutta siinä missä ihminen voi katsoa paraabelin kuvaajaa ja todeta heti, missä sen huippu on, ohjelma joutuu usein käymään systemaattisesti läpi paraabelin pisteitä ja tunnistamaan niistä huipun. Muitakin ratkaisumalleja on, mutta edellä oleva esimerkki havainnollistaa ohjelmien tai algoritmien toimintaa: ne eivät näe kokonaiskuvaa.

Koska tietokoneohjelmat usein tarkastelevat vain pientä osaa pisteistä, jotka voisivat olla optimaalisia, niiden on tärkeää tunnistaa oikein kunkin pisteen optimaalisuus. Tätä valottaa seuraava esimerkki.

Esimerkki 7. Käsitellään funktiota $f(x) = x^4 - 2x^3 - 10x^2 - x$, jonka kuvaaja on alla kuvassa 2.8.



Kuva 2.8: Funktion $f(x) = x^4 - 2x^3 - 10x^2 - x$ kuvaaja.

Jos nyt etsitään minimiä, kuvasta nähdään helposti funktion saavan pienimmän arvonsa kun $x \approx 3,13$. Tämä onkin minimi, ja sitä voidaan kutsua *globaaliksi* tai globaaliseksi minimiksi, koska *funktio ei saa missään muussa pisteessä pienempää arvoa.*

Joskus minimin etsimiseen voidaan käyttää yksinkertaista tietokoneohjelmaa, joka aloittaa jostakin funktion pisteestä ja etenee funktion kuvaajalla aina alaspäin, kuin vierivä kivi rinteellä. Kun ohjelma havaitsee pisteen, josta se ei pääse enää alemmas, se voi ilmoittaa tämän pisteen olevan minimi. Tätä menetelmää kutsutaan *nopeimman laskeutumisen mentelmäksi* ja siihen perehdytään osiossa 5.1.4 hieman syvällisemmin. Nyt riittää todeta, että jos kyseinen ohjelma aloittaa minimin etsimisen esimerkiksi kohdasta $x = -2$, se päätyy kohtaan $x \approx -1,57$ ja ilmoittaa tämän olevan minimi. Tällaista pistettä kutsutaan *lokaaliksi* tai lokaaliseksi minimiksi, koska *se on jossakin pienessä ympäristössä paras tai ei ainakaan huonompi kuin mikään ympäristön muista pisteistä.*

Äskeisessä esimerkissä määriteltiin siis lokaalinen ja globaalinen minimi. Samanlaiset ehdot voidaan määrittää myös maksimeille ja esimerkin tapauksessa kohdassa $x \approx 0,05$ onkin lokaalinen maksimi. Kaikkia maksimeita ja minimeitä kutsutaan ääriarvoiksi. Lisäksi määrittelyistä seuraa, että globaalinen ääriarvo on samalla lokaalinen ääriarvo, jossa tutkittava ympäristö voidaan valita miten suureksi tahansa. Näin ollen globaalisen minimin (tai maksimin) voi määritellä myös pienimmäksi (tai suurimmaksi) lokaaliseksi ääriarvoksi, kun tutkittava alue on suljettu ja rajoitettu. Tutkittavan funktiosta riippuen edellinen voi pitää paikkansa myös rajoittamattomassa ympäristössä, kuten esimerkin 7 globaalisen minimin tapauksessa.

Yleisesti voidaan sanoa, että globaalin ääriarvon löytäminen on hankalampaa kuin lokaalin ääriarvon, koska lokaalin ääriarvon löytämiseksi tarvitaan vain jokin lokaali ääriarvo, joita voi olla useita. Globaaliksi ääriarvoksi taas kelpaa vain yksi lokaaleista ääriarvoista: pienin minimi tai suurin maksimi. Jos lokaaleja ääriarvoja on useita tai tehtävä on muuten hankala, tämä voi olla suuri haaste optimointimenetelmille.

Optimaalista ratkaisua ei välttämättä ole olemassa, jos tutkittava väli tai alue on avoin. Yllä olevan esimerkin 7 tapauksessa funktiolle ei löydetä globaalista maksimia, koska funktio kasvaa rajatta positiiviseen ja negatiiviseen x-akselin suuntaan. Käsittelyssä on nyt siinä mielessä erikoinen tapaus, että funktiolla on lokaalinen maksimi, mutta ei globaalista. Tästä seuraa edellisen kappaleen maininta, että globaalinen maksimi on suurin lokaalisista maksimeista, kun tutkittava alue on rajoitettu. Mainintа suljetusta alueesta johtuu siitä, että jos esimerkissä etsittäisiin maksimia avoimelta väliltä $x \in (2, 5)$, välin päätepisteet eivät kuulu tutkittavaan väliin. On selvää, että maksimi saavutettaisiin kohdassa $x = 5$, mutta tämä piste ei enää kuulu avoimelle välille.

Hieman erikoisempi ääriarvojen määrittelyistä seuraava asia on, että jos

funktion ääriarvo sijaitsee kohdassa, jossa kuvaaja on jollakin alueella vaakasuora viiva, tältä viivalta voidaan valita mikä tahansa kohta ja kutsua sitä kyseiseksi ääriarvoksi. Tällainen ääriarvo voi joissakin tapauksissa olla jopa globaalinen. Joskus sama piste voi jopa olla sekä lokaali minimi että globaali maksimi. Optimoinnin kannalta on hyvä huomata, että jotta jokin ääriarvopiste olisi optimaalinen ratkaisu, sen täytyy olla sallittu piste. Eli jos löydetään ääriarvo, joka ei toteuta tehtävässä annettuja rajoitteita, kyseinen piste ei ole optimaalinen.

Lokaaliset minimit ja maksimit ovat optimoinnissa ongelmallisia, koska tietokoneohjelmat pystyvät harvoin erottamaan niitä globaalisista minimeistä ja maksimeista, jotka siis olisivat tehtävän optimaalisia ratkaisuja. Tästä syystä onkin kehitetty useita menetelmiä lokaalisten optimien välttämiseksi, kuten useita ohjelman suorituksia lähtien eri pisteistä tai satunnaisia hyppyjä pois hyvästä suunnasta, jotta löytyisi vielä parempi.

Lienee vielä syytä mainita, että jotkut lukijat saattavat muistaa edellisen esimerkin kaltaisten tehtävien ratkaisemista lukion pitkästä matematiikasta. Kyseisillä kursseilla tämänkaltaisia tehtäviä on voitu ratkaista laskemalla funktion derivaatan nollakohdat, laskea funktion arvo niissä ja valita arvoista paras. Rajoitetun välin tapauksessa myös funktion arvot välin päätepisteissä on otettu tarkasteluun mukaan. Tämä tapa toimii mukavasti, kun funktio on helposti derivoitavissa ja derivaatalla on rajoitetusti nollakohtia. Ikävä kyllä useamman muuttujan funktioiden tapauksessa menetelmän teho heikkenee verrattuna vaadittuun työmäärään samoin kuin hyvin monimutkaisten funktioiden tapauksessa. Yhden muuttujan polynomifunktion tapauksessa menetelmä on kuitenkin melko tehokas ja usein riittävän yksinkertainen käsin laskettavaksi.

2.3.3 Herkkyysanalyysistä

Eräs tärkeä näkökulma optimointiin on saadun ratkaisun käyttökelpoisuus vaihtelevissa oloissa. Realististen mallien yhteydessä mainittiin todellisessa maailmassa ilmenevä monimutkaisuus ja satunnaisuus, jotka kuvaavat hyvin vaihtelevia olosuhteita. Esimerkiksi kaupan toimintaa optimoitaessa on otettava huomioon kysynnän vaihtelut, joita voidaan ennakoida arvioimalla keskikulutusta. Ongelmia syntyy, kun arvio menee suuntaan tai toiseen pieleen. Jos kysyntä on ennakoitua suurempaa, tavara voi loppua ja korkeampi hinta voisi tuoda enemmän tuloja. Jos kysyntää taas ei ole arvioidulla tavalla, varastointikustannukset voivat nousta, tuotteet saattavat pilaantua ja alhaisempi hinta voisi tulla kysymykseen. Tällaisiin tilanteisiin herkkyysanalyysi pyrkii omalta osaltaan vastaamaan.

Herkkyysanalyysissä pyritään selvittämään, miten hyvä löydetty optimiratkaisu on, jos tehtävän rajoitteet tai osa funktiosta muuttuvat. Tällaisia muutoksia voivat olla esimerkiksi edellä mainitun kaupan kysynnän muutokset tai tavaran saatavuuden muutokset. Optimointitehtävissä tällaisia asioi-

ta kuvataan usein yksittäisillä merkeillä ja ne on oletettu tunnetuiksi. Nämä tunnetut asiat ovat edellä mainittuja parametreja ja niiden arvojen vaihtelun vaikutuksen tutkimista kutsutaan herkkyysanalyysiksi. Aina kun optimointimalli kehitetään jollekin operaatiolle, prosessille tai muulle systeemille, herkkyysanalyysi tulisi liittää siihen. Seuraavan esimerkin tarkoitus on havainnollistaa herkkyysanalyysin tärkeyttä.

Esimerkki 8. Työntekijä kulkee työmatkansa omalla autollaan ja hänellä on valittavanaan kaksi reittiä. Toinen reiteistä on valtaväylä, jonka ajamiseen kuluu a minuuttia ja toinen kulkee pienempiä teitä, jolloin reittiin kuluu aikaa b minuuttia. Tässä tapauksessa $b > a$ ja tavoitteena on minimoida matka-aika. Valtaväylä on selvästi nopeampi, mutta siellä on usein ruuhkia ja tietöitä, jotka hidastavat matkantekoa c minuuttia. Muuttujan x_1 arvo 1 tarkoittaa, että valitaan valtatie ja x_2:n arvo 1, että mennään sivuteitä. Nyt voidaan muotoilla minimointitehtävä

$$\begin{array}{ll} min & (a+c)x_1 + bx_2 \\ s.t. & b > a \\ & a > 0 \\ & b > 0 \\ & c > 0 \\ & x_1,\ x_2 \in \{0,1\} \end{array} .$$

Tehtävässä a, b ja c ovat parametreja ja niitä muuttelemalla saadaan erilaisia tehtäviä. Mielenkiinnon vuoksi oletetaan että $a = 20$ min ja $b = 25$ min. Nyt on jo intuitiivisesti selvää, että valtaväylää kannattaa käyttää vain, jos ruuhkat ja tietyöt hidastavat matkaa alle 5 minuuttia. Jos taas muutetaan arvoa $b = 22$ min, valtaväylä on nopeampi vain kun $c \leq 2$. Jälkimmäinen tapaus on siis herkempi.

2.3.4 Algoritmeista

Aiemmin on jo viitattu tietokoneiden ja ohjelmien käyttöön optimoinnissa ja nyt käsittelemme hieman ohjelmoinnin perusteita. Näiden perusteiden ymmärtämisestä on hyötyä, kun tutustumme erilaisiin optimointimenetelmiin.

Aivan ensimmäisenä tutustutaan algoritmin käsitteeseen. Algoritmi tarkoittaa oikeastaan mitä tahansa ohjetta jonkin toiminnon suorittamiseen. Se voidaan ymmärtää myös suoritettavan toiminnan kuvauksena. Esimerkiksi keittokirjassa oleva resepti on algoritmi jonkin ruuan valmistamiseksi. Samoin ajo-ohjeet ovat algoritmi, joita noudattamalla pääsee paikasta toiseen. Optimointialgoritmit ovat siis toimintaohjeita, jotta löydetään optimi.

Keittokirja, ajo-ohjeet ja optimointialgoritmit eroavat selvästi toisistaan, koska ne on usein kirjoitettu kukin omalla tyylillään. Reseptit alkavat raaka-aineiden listauksella ja jatkuvat toiminnan kuvauksella, ajo-ohjeet sisältävät etäisyyksiä, suuntia ja teiden nimiä kun taas optimointialgoritmit sisältävät

matemaattista tekstiä tai ohjelmakoodia. Onkin selvää, että kukin algoritmi on suoritettava siinä ympäristössä, johon se on suunniteltu ja jossa kaikki suorituksessa tarvittavat resurssit ovat saatavilla. Ruokaa ei yleensä kannata alkaa valmistaa vessassa, jos on tarkoitus käyttää hellaa tai monen sadan kilometrin ajo-ohjeita ei kannata seurata leluautolla. Samoin ei optimointialgoritmikaan toimi, jos jokin sen vaatimista tiedoista eli syötteistä ei ole saatavilla tai se on väärällä tavalla kerrottu. Tällaisia syötteille asetettuja ehtoja kutsutaan algoritmin alkuehdoiksi.

Tietokoneohjelmilla on oma kielensä, joka on usein hyvin pikkutarkkaa ilmaisua. Yksittäisen merkin puuttuminen voi muuttaa algoritmin suorituksen täysin tai hyvällä onnella estää koko algoritmin muodostamisen. Muodostamisen estyminen on hyvä, koska silloin ohjelmoija tietää jonkin olevan vialla jo ennen kuin algoritmi otetaan käyttöön. Historiasta tunnetaan esimerkiksi satelliitin kantoraketin räjähdys, jonka syynä oli ohjelmointivirhe. [22] Tällaisen ohjelmointikielen lukeminen on kuitenkin melko haastavaa kaikille muille kuin ohjelmoijille ja tietokoneille. Lisäksi kieliä on käytössä useita ja ne kaikki eroavat merkinnöiltään ja toimintaperiaatteiltaan toisistaan. Niinpä tässä kirjassa ei käytetäkään mitään ohjelmointikieltä kuvaamaan algoritmien suoritusta, vaan usein tyydytään käyttämään luonnollista kieltä, kuten sivulla 74 kuvassa 5.1.2. Joissakin tapauksissa voidaan käyttää niin kutsuttua pseudokoodia, joka on ohjelmointikielten ja luonnollisen kielen välimuoto. Tähän perehdymme, kun se on ajankohtaista.

Luku 3

Lineaarisesta optimoinnista

Tässä luvussa käsitellään erilaisia lineaarisia optimointitehtäviä ja niiden ratkaisemista. Lineaariselle optimointitehtävällehän olivat voimassa ehdot, jotka esiintyivät jo sivulla 23.

3.1 Lineaariset optimointitehtävät

Käsitellään ensiksi tehtävien yleistä esitysmuotoa ja sen jälkeen erilaisia optimointitehtäviä. Eri ongelmia käsittelevissä alaluvuissa kerrotaan ensin tehtävätyypin pääpiirteistä, jonka jälkeen seuraa esimerkki kyseisestä tehtävästä.

3.1.1 Tehtävien muodon muokkaus sekä yleisen tehtävän esitysmuoto

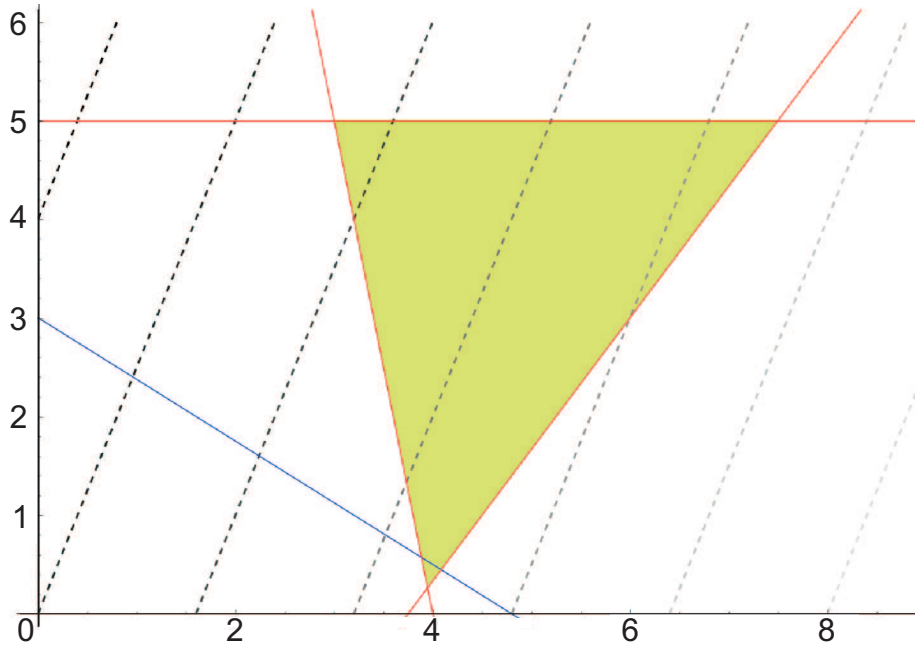
Tehtävän esitysmuodolla ei välttämättä tunnu olevan suurta merkitystä. Yksinkertaisissa tehtävissä tämä voi jopa pitää paikkansa, mutta suurten tehtävien hallinnassa yleisestä esitysmuodosta on hyötyä. Myös syötettäessä tehtävää ratkaisualgoritmille tehtävän muoto on tärkeä: algoritmi ei välttämättä hyväksy monenmuotoisia tehtäviä ja monimutkaisen tehtävän syöttämisessä voi helposti tulla inhimillinen näppäilyvirhe. Seuraavassa esimerkissä on esitetty sama optimointitehtävä kahdessa eri muodossa, jotka toivottavasti havainnollistavat muodon selkeyttä. Tehtävä on myös ratkaistu graafisesti, ja ratkaisu on sama molemmissa muodoissa. Huomaa rajoitteiden tyypit eri tehtävän muodoissa.

Esimerkki 9.

$$
\begin{aligned}
\max \quad & 2,5x_1 - x_2 \\
s.t. \quad & x_1 + 0,2x_2 \geq 4 \\
& 0,5x_1 + 0,8x_2 = 2,4 \\
& 0,4x_1 - 0,3x_2 \leq 1,5 \\
& x_1 \leq 10 \\
& x_2 \leq 5 \\
& x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

$$
\begin{array}{rl}
-\min & -2,5x_1 + x_2 \\
s.t. & -x_1 - 0,2x_2 \leq -4 \\
& 0,5x_1 + 0,8x_2 \leq 2,4 \\
& -0,5x_1 - 0,8x_2 \leq 2,4 \\
& 0,4x_1 - 0,3x_2 \leq 1,5 \\
& x_1 \leq 10 \\
& x_2 \leq 5 \\
& x_1 \geq 0, x_2 \geq 0
\end{array}
$$

Kaikki rajoitteet on saatu samantyyppisiksi epäyhtälöiksi ja tehtävän tyyppi vaihdettu minimoinniksi.



Kuva 3.1: Kuva esimerkin 9 tilanteesta.

Kuvassa 3.1 voimassaolevat epäyhtälörajoitteet on merkitty punaisilla suorilla ja niiden voimassaoloalue vihreällä. Yhtälörajoite on merkitty sinisellä suoralla, joten optimi löytyy tältä suoralta vihreän alueen sisältä tai rajalta. Koska kyseessä on lineaarinen tehtävä, tiedetään optimin löytyvän kärkipisteestä ja voidaan jättää vihreän alueen sisus tutkimatta. Optimi saavutetaan pisteessä $(\frac{192}{47}, \frac{21}{47})$.

Yleinen tehtävä

Tämä osio käsittelee matemaattisia merkintöjä, kuten lyhennysmerkintöjä, joten sen seuraaminen ei ole asian ymmärtämisen kannalta aivan välttämä-

töntä. Osio on kuitenkin hyödyllinen merkintöjen taustojen ymmärtämiseksi. Seuraavaksi käsitellään yleisen tehtävän merkintää ja lähdetään liikkeelle laajimmasta merkintätavasta. Tehtävässä on n muuttujan kohdefunktio, s "suurempi tai yhtäsuuri kuin" epäyhtälörajoitusta, t yhtälörajoitusta, u "pienempi tai yhtäsuuri kuin" epäyhtälörajoitusta sekä n ei-negatiivisuusrajoitusta. Tehtävä näyttää siis melkoisen hirviömäiseltä. Prosessia selventävä esimerkki on sivulla 37.

$$
\begin{aligned}
\max \quad & a_1x_1 + a_2x_2 + \ldots + a_nx_n \\
s.t. \quad & b_{11}x_1 + b_{21}x_2 + \ldots + b_{n1}x_n \geq h_{11} \\
& \vdots \qquad \text{(yhteensä } s \text{ kappaletta)} \\
& b_{1s}x_1 + b_{2s}x_2 + \ldots + b_{ns}x_n \geq h_{s1} \\
& c_{11}x_1 + c_{21}x_2 + \ldots + c_{n1}x_n = h_{12} \\
& \vdots \qquad \text{(yhteensä } t \text{ kappaletta)} \\
& c_{1t}x_1 + c_{2t}x_2 + \ldots + c_{nt}x_n = h_{t2} \\
& d_{11}x_1 + d_{21}x_2 + \ldots + d_{n1}x_n \leq h_{13} \\
& \vdots \qquad \text{(yhteensä } u \text{ kappaletta)} \\
& d_{1u}x_1 + d_{2u}x_2 + \ldots + d_{nu}x_n \leq h_{u3} \\
& x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0
\end{aligned}
$$

Huomaa että vain x:t ovat muuttujia ja muut ovat vakioita. Vakiot voivat myös olla positiivisia, negatiivisia tai nollia. Alaindeksejä on käytetty erottamaan eri vakiot ja muuttujat toisistaan. Summamerkinnällä saadaan tiiviimpi esitys samalle asialle, vaikka se viekin enemmän palstatilaa.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} a_i x_i \\
s.t. \quad & \sum_{i=1}^{n} b_{i1} x_i \geq h_{11} \\
& \vdots \qquad \text{(yhteensä } s \text{ kappaletta)} \\
& \sum_{i=1}^{n} b_{is} x_i \geq h_{s1} \\
& \sum_{i=1}^{n} c_{i1} x_i = h_{12} \\
& \vdots \qquad \text{(yhteensä } t \text{ kappaletta)} \\
& \sum_{i=1}^{n} c_{it} x_i = h_{t2} \\
& \sum_{i=1}^{n} d_{i1} x_i \leq h_{13}
\end{aligned}
$$

$$
\begin{aligned}
&\vdots \qquad \text{(yhteensä } u \text{ kappaletta)} \\
&\sum_{i=1}^{n} d_{iu} x_i \leq h_{u3} \\
&x_1 \geq 0, x_2 \geq 0, \ldots, x_n \geq 0
\end{aligned}
$$

Indeksoimalla rivit päästään jo selvästi lyhyempään esitykseen.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} a_i x_i \\
s.t. \quad & \sum_{i=1}^{n} b_{ij} x_i \geq h_{j1}, j = 1, \ldots, s \\
& \sum_{i=1}^{n} c_{ik} x_i = h_{k2}, k = 1, \ldots, t \\
& \sum_{i=1}^{n} d_{il} x_i \leq h_{l3}, l = 1, \ldots, u \\
& x_i \geq 0, i = 1, \ldots, n
\end{aligned}
$$

Muuttamalla kaikki tehtävän yhtälö- ja epäyhtälörajoitukset ”pienempi tai yhtäsuuri kuin” -muotoon saadaan tehtävän esitys vielä yksinkertaisemmaksi. Tällaista muotoa kutsutaan *kanoniseksi muodoksi*. Nyt jokainen yhtälörajoite muuttuu kahdeksi epäyhtälörajoitteeksi, joten rajoitteita on yhteensä $s + 2t + u = m$ kappaletta.

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} a_i x_i \\
s.t. \quad & \sum_{i=1}^{n} b_{ij} x_i \leq h_j, j = 1, \ldots, m \qquad (3.1) \\
& x_i \geq 0, i = 1, \ldots, n \qquad (3.2)
\end{aligned}
$$

Näiden lisäksi on vielä olemassa matriisimuoto tehtävästä. Matriisien opetteleminen tässä vaiheessa on kuitenkin liian työlästä, joten seuraava muoto on tarkoitettu vain niille, jotka ovat tutustuneet matriiseihin jo entuudestaan tai aikovat ottaa niistä pikaisesti selvää. Matriisien perusteet on esitelty havainnollisesti muiden muassa kirjan [1] liitteessä A3. Matriisit on merkitty lihavoiduilla kirjaimilla. Seuraavassa vektorien $\mathbf{a}$, $\mathbf{x}$ ja $\mathbf{0}$ dimensio

on n ja vektorin $\mathbf{h}$ dimensio on m, eli kolme ensimmäistä sisältävät n ja neljäs sisältää m komponenttia. Matriisi $\mathbf{B}$ on kokoa $m \times n$. Matriisimuotoa havainnollistetaan esimerkissä 11 sivulla 39.

$$\begin{aligned} \max \quad & \mathbf{a}^T\mathbf{x} \\ & \mathbf{Bx} \leq \mathbf{h} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{3.3}$$

Tehtävän muodon muokkaus

Nyt kun yleisen tehtävän merkinnät on käsitelty, voidaan siirtyä sen muodon muokkaukseen. Asiat on käsiteltävä tässä järjestyksessä, jotta olisi selvää, että muunnokset ovat päteviä kaikille tehtäville. Seuraavaksi käsitellään keinoja muokata tehtävä erilaiseen muotoon. On suositeltavaa tarkastella keinojen lukemisen aikana jotakin optimointitehtävää, kuten esimerkkejä 9 ja 10.

1. Maksimointi voidaan muuttaa minimoinniksi ja päinvastoin vaihtamalla kohdefunktion merkit seuraavasti

   $$\max \sum_{i=1}^{n} a_i x_i = -\min \sum_{i=1}^{n} (-a_i) x_i.$$

   Esimerkiksi funktion $f(x) = x^2 - 2x + 2$ minimi on sama kuin funktion $g(x) = -x^2 + 2x - 2$ maksimin käänteisluku, ja ne saavutetaan samassa kohdassa. Tätä on havainnollistettu kuvassa 3.2

2. Epäyhtälön merkki voidaan kääntää kertomalla se (-1):llä, joten kaikki epäyhtälöt voidaan muokata olemaan samaa tyyppiä kaavalla

   $$\sum_{i=1}^{n} b_i x_i \leq h \Leftrightarrow \sum_{i=1}^{n} (-b_i) x_i \geq -h.$$

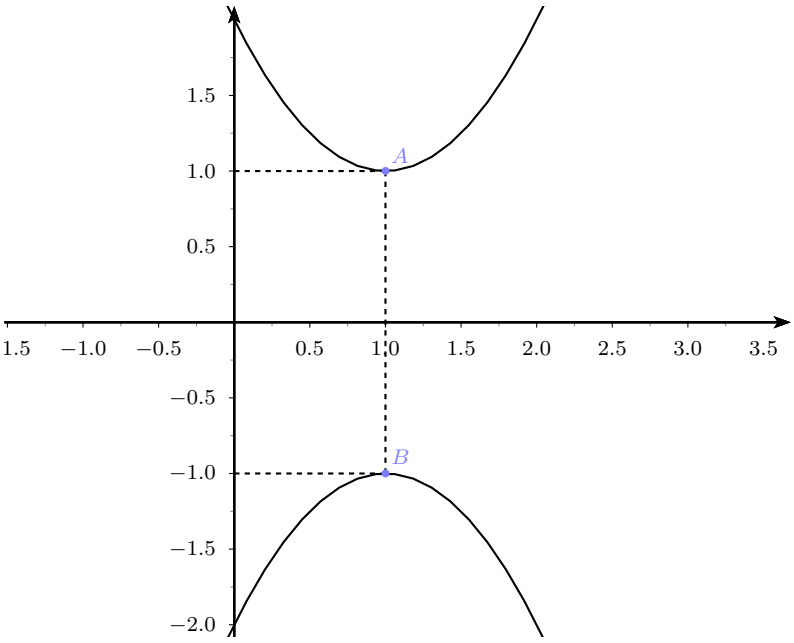
3. Muotoa $\sum_{i=1}^{n} b_{ij} x_i = h_j, j = 1, \ldots, m$ olevat yhtälöt voidaan muuttaa yhtäpitäviksi epäyhtälöiksi kahdella tavalla. Ensimmäinen tapa on muuttaa jokainen yhtälö kahdeksi seuraavanlaiseksi epäyhtälöksi

   $$\sum_{i=1}^{n} b_{ij} x_i \leq h_j, j = 1, \ldots, m$$

   $$\sum_{i=1}^{n} (-b_{ij}) x_i \leq -h_j, j = 1, \ldots, m.$$

   Toisessa tavassa otetaan käyttöön yksi lisärajoitus, jonka avulla yhtälöt voidaan muuttaa muotoon

   $$\sum_{i=1}^{n} b_i x_i \leq h,$$



Kuva 3.2: Havainnollistus säännöstä 1.

$$\sum_{i=1}^{n} r_i x_i \leq s,$$

missä $r_i = -\sum_{j=1}^{m} b_{ij}, j = 1, \ldots, n$ ja $s = -\sum_{j=1}^{m} h_j$. Tämä jälkimmäinen tapa on matemaattisesti elegantimpi, mutta ensimmäisen tavan käyttäminen on yksinkertaisempaa.

4. Nollan tai sitä pienempiä arvoja saavat muuttujat $x \leq 0$ voidaan muuttaa ei-negatiivisiksi uusilla muuttujilla $y = -x$.

5. Jos muuttujan merkki ei ole rajoitettu, voidaan se muuttaa ei-negatiiviseksi merkitsemällä $x = x' - x''$, $x' \geq 0$, $x'' \geq 0$. Vaikka uusien muuttujien arvot eivät ole yksikäsitteisiä, voidaan määrätä toinen niistä aina nollaksi, eli $x' = \max\{0, x\}$, $x'' = \max\{0, -x\}$.

6. Jos halutaan käyttää yhtälörajoitteita, epäyhtälö voidaan muuttaa yhtälöksi ottamalla käyttöön uusi apumuuttuja seuraavasti

$$\sum_{i=1}^{n} b_i x_i \leq h \Leftrightarrow \sum_{i=1}^{n} b_i x_i + s = h,\ s \geq 0$$

$$\sum_{i=1}^{n} b_i x_i \geq h \Leftrightarrow \sum_{i=1}^{n} b_i x_i - t = h,\ t \geq 0.$$

Uudet muuttujat ovat ei-negatiivisia ja niitä kutsutaan *ali-* (s, engl. slack) ja *ylijäämämuuttujiksi* (t, engl. surplus).

Lineaarisessa optimoinnissa käytetään kahta yleistä perusmuotoilua ongelmille, eli *kanonista* ja *standardimuotoa*. Kanoninen muoto on yleensä luonnollisempi, mutta standardimuoto on usein kätevämpi teoreettisissa tarkasteluissa. Alla on esitetty nämä muodot. Yllä esitettyjen keinojen avulla lineaarinen optimointitehtävä saadaan aina kanoniseen muotoon ja standardimuotoon.

1. Kanoninen muoto on sivulla 34 kohdissa 3.1 ja 3.3.

2. Standardimuodossa rajoitteet ovat epäyhtälöiden sijaan yhtälöitä. (Tätä muotoa saatetaan joskus kutsua myös argumenttimuodoksi, jolloin standardimuodon ja kanonisen muodon voi sekoittaa toisiinsa. [31])

$$\begin{array}{ll} \max & \sum_{i=1}^{n} a_i x_i \\ s.t. & \sum_{i=1}^{n} b_{ij} x_i = h_j, j = 1, \ldots, m \\ & x_i \geq 0, i = 1, \ldots, n \end{array}$$

Matriisimuodossa esitys on seuraava:

$$\begin{array}{ll} \max & \mathbf{a}^T \mathbf{x} \\ & \mathbf{B}\mathbf{x} = \mathbf{h} \\ & \mathbf{x} \geq \mathbf{0} \end{array} .$$

Alla on kaksi esimerkkiä tehtävän muodon muuttamisesta. Ensimmäinen on samankaltainen kuin esimerkki 9, mutta nyt muutos käydään tarkemmin läpi ja pyritään kanoniseen muotoon. Toisessa esimerkissä pyritään standardimuotoon ja myös matriisimuoto on esitetty, joten esimerkkiä voidaan pitää lisälukemisena.

Esimerkki 10. Alkuperäinen tehtävä on seuraava:

$$\begin{array}{ll} \min & x_1 + 2x_2 \\ s.t. & x_1 - 2x_2 \geq -3 \\ & 0,5x_1 + 2x_2 \geq -3,5 \\ & -x_1 + x_2 \geq -4 \\ & -x_1 - x_2 \geq -2 \\ & 2x_1 + x_2 \geq -3 \\ & x_2 \leq 0 \end{array} .$$

Pyritään kanoniseen muotoon, joten muutetaan ensin tehtävän tyyppi sekä kaikki epäyhtälörajoitteet oikeaan muotoon sivun 35 kohtien 1 ja 2 mukaisesti.

$$
\begin{array}{ll}
-\max & -x_1 - 2x_2 \\
s.t. & -x_1 + 2x_2 \leq 3 \\
& -0,5x_1 - 2x_2 \leq 3,5 \\
& x_1 - x_2 \leq 4 \\
& x_1 + x_2 \leq 2 \\
& -2x_1 - x_2 \leq 3 \\
& x_2 \leq 0
\end{array}
$$

Seuraavaksi käsitellään ei-negatiivisuusrajoitteita ja huomataan niiden puuttuvan. Otetaan siis käyttöön uusia muuttujia kohtien 4 ja 5 mukaisesti, jotta saadaan käyttöön ei-negatiivisia muuttujia.

$$
\begin{array}{ll}
-\max & -x_1 - 2x_2 \\
s.t. & -x_1 + 2x_2 \leq 3 \\
& -0,5x_1 - 2x_2 \leq 3,5 \\
& x_1 - x_2 \leq 4 \\
& x_1 + x_2 \leq 2 \\
& -2x_1 - x_2 \leq 3 \\
& x_3 = -x_2 (\Rightarrow x_3 \geq 0) \\
& x_1 = x_1' - x_1'' \\
& x_1' = \max\{0, x_1\} \geq 0 \\
& x_1'' = \max\{0, x_1\} \geq 0 \\
& x_2 \leq 0
\end{array}
$$

Tässä välivaiheessa kaikki rajoitteet on saatu samantyyppisiksi epäyhtälöiksi ja on luotu perusta muuttujien muuttamiselle. Nyt enää korvataan vanhat muuttujat uusilla.

$$
\begin{array}{ll}
-\max & -x_1' + x_1'' + 2x_3 \\
s.t. & -x_1' + x_1'' - 2x_3 \leq 3 \\
& -0,5x_1' + 0,5x_1'' + 2x_3 \leq 3,5 \\
& x_1' - x_1'' + x_3 \leq 4 \\
& x_1' - x_1'' - x_3 \leq 2 \\
& -2x_1' + 2x_1'' + x_3 \leq 3 \\
& x_3 \geq 0 \\
& x_1 = x_1' - x_1'' \\
& x_1' = \max\{0, x_1\} \geq 0 \\
& x_1'' = \max\{0, -x_1\} \geq 0 \\
& x_3 = -x_2 \\
& x_2 \leq 0
\end{array}
$$

Jotta tehtävästä tulisi mukavamman näköinen, jätetään muutoksen jälkeen negatiivinen muuttuja x_2 pois, koska se on tarpeeton. Lisäksi "$\ldots = \max \ldots \geq 0$" ovat epälineaarisia rajoitteita ja siten hyvin harmillisia. Usein

kyseiset rajoitteet kuitenkin toteutuvat automaattisesti ja siihen luottaen ne voidaan jättää pois. Tuloksia tulkittaessa on tietenkin aina syytä tarkistaa tuloksen järkevyys, jolloin selviää onko poisjätettyjä rajoitteita noudatettu: jos on, $x_1 = x_1'$ tai $x_1 = -x_1''$. Tehtävää ratkaistaessa voidaan myös jättää pois rajoite $x_1 = x_1' - x_1''$, koska x_1 ei enää esiinny missään muualla.

$$\begin{array}{ll} -\max & -x_1' + x_1'' + 2x_3 \\ s.t. & -x_1' + x_1'' - 2x_3 \leq 3 \\ & -0,5x_1' + 0,5x_1'' + 2x_3 \leq 3,5 \\ & x_1' - x_1'' + x_3 \leq 4 \\ & x_1' - x_1'' - x_3 \leq 2 \\ & -2x_1' + 2x_1'' + x_3 \leq 3 \\ & x_1' \geq 0 \\ & x_1'' \geq 0 \\ & x_3 \geq 0 \end{array}$$

Nyt kaikki päätösmuuttujat saavat vain ei-negatiivisia arvoja, vaikka muuttujia onkin nyt enemmän. Käsin laskevalle ihmiselle tämä ei ole suotuisaa kehitystä, mutta rajoittuneelle tietokoneohjelmalle on. Optimi saavutetaan pisteessä $x_1' = 0$, $x_1'' = 5/7$, $x_3 = 11/7$, eli alkuperäisiin muuttujiin palautettuna $x_1 = -5/7$, $x_2 = -11/7$. Tätä on selvennetty kuvassa 3.3.

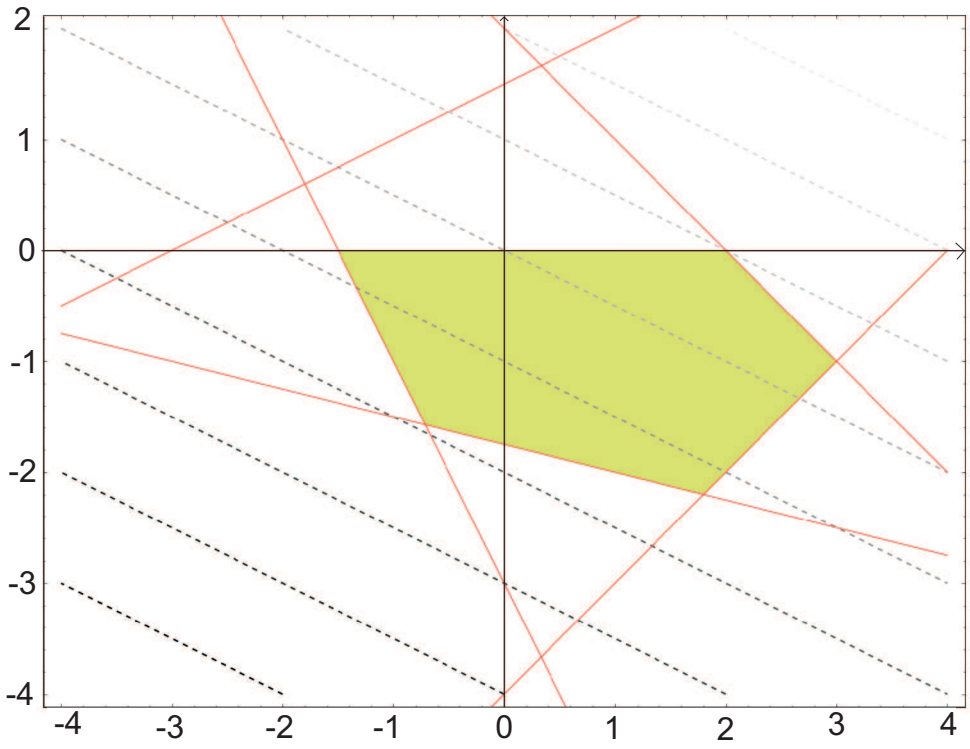
Esimerkki 11. Alkuperäinen tehtävä on seuraava:

$$\begin{array}{ll} \max & 3x_1 + 2x_2 \\ s.t. & x_1 + 3x_2 \leq 9 \\ & 2x_1 + x_2 \leq 8 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{array} .$$

Pyritään standardimuotoon muotoon, joten muutetaan epäyhtälörajoitteet yhtälörajoitteiksi sivun 36 kohdan 6 mukaisesti.

$$\begin{array}{ll} \max & 3x_1 + 2x_2 \\ s.t. & x_1 + 3x_2 + s_1 = 9 \\ & 2x_1 + x_2 + s_2 = 8 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & s_1 \geq 0 \\ & s_2 \geq 0 \end{array}$$

Näin on päästy standardimuotoon, joten voidaan jatkaa siirtymistä matriisimuotoon, jonka komponentit ovat seuraavat.



Kuva 3.3: Kuva esimerkin 10 tilanteesta.

$$\mathbf{a} = \begin{pmatrix} 3 \\ 2 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{a}^T = \begin{pmatrix} 3 & 2 & 0 & 0 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix} \quad \mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{h} = \begin{pmatrix} 9 \\ 8 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 1 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}$$

Näistä saadaan sivulla 37 kohdassa 2 esitetty matriisimuoto

$$\max \quad \begin{pmatrix} 3 & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 8 \end{pmatrix},$$
$$\begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

joka on yhtäpitävä alla olevan yksinkertaistetun muodon kanssa.

$$\max \quad \begin{pmatrix} 3 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 3 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} 9 \\ 8 \end{pmatrix}$$
$$\begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \end{pmatrix} \geq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Optimin löytäminen jätetään harjoitustehtäväksi.

3.1.2 Allokaatiotehtävät

Allokaatio tarkoittaa jakamista eri kohteisiin ja allokaatiotehtävissä pyritäänkin jakamaan rajalliset resurssit parhaalla mahdollisella tavalla. Koska resurssit ovat rajalliset, voidaan myös ajatella eri käyttökohteiden kilpailevan keskenään resurssien osuuksista. Erilaiset budjetit ovat usein tällaisia ongelmia.

Esimerkki 12. Kaupungilla on käytössä 18200 metriä rantaviivaa, jota voidaan käyttää uimarantana, rakennusmaana, venesatamana ja raskaana satamana. Tarvitaan vähintään 300 m uimarantaa, 1000 m rakennusmaata, 500 m venesatamaa ja 2000 m raskasta satamaa. Lisäksi rantaa voidaan käyttää korkeintaan 2000 m uimarantaan, 14000 m rantatontteihin, 18200 m venesatamaan ja 14000 m raskaaseen satamaan. Kukin käyttötarkoitus tuottaa kaupungille rahaa kutakin käytettyä metriä kohti, ja tuotot ovat toisiinsa

suhteutettuina 0,5/m uimarannalta, 10/m rantatonteilta, 5/m venesatamalta ja 9/m raskaalta satamalta. Muodostetaan optimointitehtävä, jossa pyritään maksimoimaan tuotot. Seuraavassa x_1 kertoo uimarantaan käytettyjen metrien määrän ja x_2, x_3 sekä x_4 ovat vastaavat arvot rantatonteille, venesatamalle ja raskaalle satamalle. Koska on käytössä 4 päätösmuuttujaa, kuvan piirtäminen tilanteesta on liian vaikeaa, eikä graafista ratkaisua esitetä.

$$
\begin{array}{ll}
\max & 0,5x_1 + 10x_2 + 5x_3 + 9x_4 \\
s.t. & x_1 + x_2 + x_3 + x_4 \leq 18200 \\
 & x_1 \leq 2000 \\
 & x_2 \leq 14000 \\
 & x_3 \leq 18200 \\
 & x_4 \leq 14000 \\
 & x_1 \geq 300 \\
 & x_2 \geq 1000 \\
 & x_3 \geq 500 \\
 & x_4 \geq 2000
\end{array}
$$

Asiasta kiinnostuneille alla on sama matriisimuodossa. Neljä viimeistä rajoitusta on käännetty ympäri, jotta kaikki rajoitteet saataisiin samaan matriisiin. (SAGElla voi tehokkaasti ratkaista alla olevassa muodossa olevia tehtäviä.)

$$
\max \quad \begin{pmatrix} 0,5 & 10 & 5 & 9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}
$$

$$
\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 18200 \\ 2000 \\ 14000 \\ 18200 \\ 14000 \\ -300 \\ -1000 \\ -500 \\ -2000 \end{pmatrix}.
$$

Optimiratkaisuna pisteessä $(x_1, x_2, x_3, x_4) = (300, 14000, 500, 3400)$ saavutetaan arvo 173250. Loput esimerkistä keskittyy tehtävän ratkaisemiseen SAGE:lla, joten jos käytät jotakin muuta ohjelmaa tai ohjelmistoa, voit hypätä tämän kohdan yli. Tehtävän ratkaiseminen onnistuu kirjoittamalla alla olevat komennot ja suorittamalla solu.

```
import numpy
from cvxopt.base import spmatrix
from cvxopt.base import matrix as m
from cvxopt import umfpack
RealNumber=float
Integer=int
from cvxopt import solvers

c = m([-0.5, -10., -5., -9.])

G = m([[1., 1., 0., 0., 0., -1., 0., 0., 0. ], [1., 0., 1., 0., 0., 0., -1., 0., 0. ],
[1., 0., 0., 1., 0., 0., 0., -1., 0. ], [1., 0., 0., 0., 1., 0., 0., 0., -1. ]])

h = m([18200., 2000., 14000., 18200., 14000., -300., -1000., -500., -2000.])

sol = solvers.lp(c,G,h)
```

Seitsemän ensimmäistä komentoa hakevat tarvittavia tietoja ja luovat määrittelyjä. Kolme seuraavaa riviä (jotka alkavat 'c=', 'G=' ja 'h=') sisältävät kuvaukset tehtävän matriiseista ja viimeinen rivi käyttää itse "ratkaisinta", joka etsii optimaalisen ratkaisun. Solun suorituksen jälkeen ilmestyy seuraava teksti:

	pcost	*dcost*	*gap*	*pres*	*dres*	*k/t*
0 :	$-1.6248e+05$	$-4.1027e+05$	$3e+05$	$2e-01$	$6e-01$	$1e+00$
1 :	$-1.6443e+05$	$-2.0928e+05$	$4e+04$	$4e-02$	$1e-01$	$1e+03$
2 :	$-1.6910e+05$	$-1.7745e+05$	$8e+03$	$8e-03$	$2e-02$	$4e+02$
3 :	$-1.7315e+05$	$-1.7372e+05$	$5e+02$	$6e-04$	$1e-03$	$3e+01$
4 :	$-1.7325e+05$	$-1.7326e+05$	$6e+00$	$7e-06$	$2e-05$	$4e-01$
5 :	$-1.7325e+05$	$-1.7325e+05$	$6e-02$	$7e-08$	$2e-07$	$4e-03$
6 :	$-1.7325e+05$	$-1.7325e+05$	$6e-04$	$7e-10$	$2e-09$	$4e-05$

$Optimal solution found.$

Nyt itse optimi saadaan näkyviin komennolla `print sol['x']` ja suorittamalla sen sisältävä solu. Tällöin näytölle tulostuu optimin paikka

$$\begin{matrix}[3.00e+02]\\ [1.40e+04]\\ [5.00e+02]\\ [3.40e+03]\end{matrix}.$$

Tästä saadaan optimin arvo laskemalla SAGE:lla

$$0.5*3.00e+02+10*1.40e+04+5*5.00e+02+9*3.40e+03=173250.0.$$

3.1.3 Sekoitukset

Sekoitustehtävään olemmekin jo törmänneet esimerkissä 2 sivulla 15 . Tällaisissa tehtävissä sekoitetaan kahta tai useampaa eri resurssia, jotta päästäisiin parhaaseen lopputulokseen.

Esimerkki 13. Yritys haluaa valmistaa vähintään 2000 kg messinkiä, jossa on vähintään 60% kuparia, 38% sinkkiä ja korkeintaan 1% lyijyä. Käytettävissa ovat taulukon 3.1 mukaiset raaka-aineet. Millä raaka-aineiden yhdistelmällä niiden hankintakustannukset ovat mahdollisimman pienet?

Materiaali	Kuparia (%)	Sinkkiä (%)	Lyijyä (%)	Saatavuus (kg)	Hinta (€/kg)
Messinki 1	85	15	0	1000	4
Messinki 2	58	39	3	1100	3,2
Pronssi	85	0	0	300	3,6
Kupari	100	0	0	2000	6
Sinkki	0	100	0	2000	1,5
Lyijy	0	0	100	2000	1,8

Taulukko 3.1: Esimerkissä 13 käytössä olevat raaka-aineet.

Muodostetaan tapauksesta optimointitehtävä. Merkitään hankittavien materiaalien määriä seuraavasti: messinki 1:tä x_1 kg, messinki 2:ta x_2, pronssia x_3 kg, kuparia x_4 kg, sinkkiä x_5 kg ja lyijyä x_6 kg.

$$
\begin{aligned}
\min \quad & 4x_1 + 3,2x_2 + 3,6x_3 + 6x_4 + 1,5x_5 + 1,8x_6 \\
s.t. \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 2000 \\
& 0,85x_1 + 0,58x_2 + 0,85x_3 + x_4 \geq 0,6 \cdot 2000 \\
& 0,15x_1 + 0,39x_2 + x_5 \geq 0,38 \cdot 2000 \\
& 0,03x_2 + x_6 \leq 0,01 \cdot 2000 \\
& x_1 \leq 1000 \\
& x_2 \leq 1100 \\
& x_3 \leq 300 \\
& x_4 \leq 2000 \\
& x_5 \leq 2000 \\
& x_6 \leq 2000 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

Ensimmäinen rajoite tulee lopputuotteen määrästä, kolme seuraavaa eri metallien osuuksista ja loput materiaalien määristä. Sama on alla matriisimuodossa.

$$\max \quad \begin{pmatrix} 4 & 3,2 & 3,6 & 6 & 1,5 & 1,8 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix}$$

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ -0,85 & -0.58 & -0,85 & -1 & 0 & 0 \\ -0,15 & -0,39 & 0 & 0 & -1 & 0 \\ 0,03 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leq \begin{pmatrix} -2000 \\ 2000 \\ -1200 \\ -760 \\ 20 \\ 1000 \\ 1100 \\ 300 \\ 2000 \\ 2000 \\ 2000 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Optimaalinen ratkaisu on käyttää messinki 1:tä 1000 kg, messinki 2:ta n. 300,8 kg, pronssia n. 206,5 kg ja sinkkiä n. 492,7 kg. Tällöin kokonaiskustannus on noin 6445 €. Tämä on toistaiseksi ainoa tehtävä, jossa voidaan havaita eroja eri optimointialgoritmien tehokkuudessa. Edellä mainittu arvo on saatu AMPL:llä osoitteessa

`http://ampl.com/cgi-bin/ampl/amplcgi`

käyttämällä ”minos”-ratkaisumallia. Käyttämällä SAGEa optimiksi saatiin noin 6497 €. Erikoista menetelmien käyttämisessä on, että jos tehtävän yhtälörajoitteen muuttaa ’suurempi tai yhtäsuuri kuin’ -rajoitteeksi (jolloin sallitaan tuotettavaksi yli 2000 kg messinkiä), optimit olivat AMPL:llä n. 6423 €ja SAGElla n. 6443 €. Näihin eroihin palataan lyhyesti luvussa 6. Tässä tapauksessa erot eivät ole kovin merkittäviä, koska suurinkin niistä on 74€, eli noin prosentin luokkaa koko kustannuksesta. Jos kuitenkin yksikkönä olisi eurojen sijaan vaikkapa ”sataa tuhatta euroa”, näinkin pienet erot kannattaisi huomioida.

Tehtävä saadaan ratkaistua AMPL:lla syöttämällä alla olevat komennot. Huomaa, että jokaisen rivin on päätyttävä puolipisteeseen.

```
var X1;
var X2;
var X3;
var X4;
var X5;
var X6;
minimize Cost: 4*X1+3.2*X2+3.6*X3+6*X4+1.5*X5+1.8*X6;
subject to Amount: X1+X2+X3+X4+X5+X6=2000;
subject to Copper: 0.85*X1+0.58*X2+0.85*X3+1*X4>=1200;
subject to Zinch: 0.15*X1+0.39*X2+1*X5>=760;
subject to Lead: 0.03*X2+1*X6<=20;
subject to A_limit: 0 <= X1 <= 1000;
subject to B_limit: 0 <= X2 <= 1100;
subject to C_limit: 0 <= X3 <= 300;
subject to D_limit: 0 <= X4 <= 2000;
subject to E_limit: 0 <= X5 <= 2000;
subject to F_limit: 0 <= X6 <= 2000;
solve;
display X1, X2, X3, X4, X5, X6;
display 4*X1+3.2*X2+3.6*X3+6*X4+1.5*X5+1.8*X6;
```

Kuusi ensimmäistä riviä esittelevät muuttujat, joiden jälkeisellä rivillä kerrotaan optimoinnin tyyppi (minimize tai maximize), funktion nimi (tässä tapauksessa ”Cost”) ja funktio itsessään. Kymmenellä tämän jälkeen olevalla rivillä asetetaan jonkin rajoitteen ”subject to” -komennolla. Komennon jälkeen seuraa rajoitteen nimi, kaksoispiste ja itse rajoite. Kaikki nimet ovat vapaavalintaisia, mutta niiden kannattaa olla kuvaavia, jotta muistetaan, mihin asiaan ne liittyvät. Kolmanneksiviimeisellä rivillä on vielä komento käyttää ratkaisinta. Tämän jälkeisten rivien komennot tuovat näkyviin muuttujien arvot optimissa ja kohdefunktion arvon siinä. Kun yllä olevat komennot kopioi AMPL:n Internet-sovelluksen alempaan kenttään ja klikkaa ”Send”, tulostuu ikkunaan seuraava teksti.

MINOS 5.51: optimal solution found.
4 iterations, objective 6445.04065
X1 = 1000
X2 = 300.813
X3 = 206.504
X4 = 0
X5 = 492.683
X6 = 0

4*X1 + 3.2*X2 + 3.6*X3 + 6*X4 + 1.5*X5 + 1.8*X6 = 6445.04

Tästä nähdään jo aiemmin mainittu optimaalinen ratkaisu, josta ilmenevät metallien käyttösuhteet ja niiden hinta. Lisäksi nähdään ratkaisimen nimi ja kuinka monta iteraatiota tai "kierrosta" se suoritti. Nämä ovat tärkeitä tietoja, kun vertaillaan eri ratkaisimia keskenään. On myös mahdollista, että jotkin ratkaisimet eivät anna tiettyyn tehtävään tulosta lainkaan, kuten AMPL:n "tn" -ratkaisin, tai antavat virheellisen optimin, kuten AMPL:n "path" -ratkaisin ja SAGE.

3.1.4 Toiminnan suunnittelu

Usein elämässä törmää tilanteisiin, joissa pitäisi tehdä monia asioita ja niiden tekemiseen on useita mahdollisia paikkoja. Esimerkiksi remontteja hoitavalla yrityksellä voi olla useita kohteita työn alla ja jotkin kohteet saattavat vaatia erilaisia toimenpiteitä tai yrityksellä voi olla useita erilaisia tuotantolaitoksia, jotka voivat pystyä osittain erilaisten asioiden tuottamiseen. Toiminnan suunnittelun ytimenä on päättää, mitä, missä ja milloin tehdään.

Esimerkki 14. Lannoitetehtaassa valmistetaan kahdella tuotantolinjalla kolmea tuotetta. Tuotteiden minimikysynnät sekä eri tuotantolinjojen kapasiteetit ja kustannukset on esitetty taulukossa 3.2. Miten paljon kutakin tuotetta kannattaa valmistaa kullakin linjalla, jotta kustannukset olisivat mahdollisimman pienet? Kysynnän ja kapasiteetin yksikkönä on tonni ja kustannuksen yksikkönä kymmentätuhatta euroa tonnia kohden.

	Tuote 1	Tuote 2	Tuote 3	Yhteensä
Kysyntä (1000 kg)	2	5	3	-
Linjan 1 kapasiteetti	1	2	3	5
Linjan 2 kapasiteetti	2	5	1	6
Linjan 1 kustannus	1	3	4	-
Linjan 2 kustannus	2	2	1	-

Taulukko 3.2: Esimerkissä 14 olevan lannoitetehtaan tiedot.

Merkitään muuttujalla x_{ij} linjalla tuotettavien tuotteiden määrää, missä alaindeksi i tarkoittaa linjaa ja j tuotetta. Näin saadaan optimointitehtävä

$$\begin{aligned}
\min \quad & x_{11} + 3x_{12} + 4x_{13} + 2x_{21} + 2x_{22} + x_{23} \\
s.t. \quad & x_{11} + x_{21} \geq 2 \\
& x_{12} + x_{22} \geq 5 \\
& x_{13} + x_{23} \geq 3 \\
& x_{11} + x_{12} + x_{13} \leq 5 \\
& x_{21} + x_{22} + x_{23} \leq 6 \\
& x_{11} \leq 1 \\
& x_{12} \leq 2 \\
& x_{13} \leq 3 \\
& x_{21} \leq 2 \\
& x_{22} \leq 5 \\
& x_{23} \leq 1 \\
& x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \geq 0
\end{aligned} \quad .$$

Optimaalinen arvo on 23, joka saavutetaan taulukon 3.3 tilanteessa. On syytä huomata, että tehtävässä olisi sallittu myös muut kuin kokonaislukuarvot, koska yksikkönä oli 1000 kg.

	Tuote 1	Tuote 2	Tuote 3	Yhteensä
Kysyntä (1000 kg)	2	5	3	-
Linjan 1	1	1	2	4
Linjan 2	1	4	1	6

Taulukko 3.3: Esimerkin 14 optimi.

3.1.5 Muista lineaarisista ongelmista

Edellä on tutkittu melko pitkästikin erilaisia lineaarisia ongelmia. Näiden lisäksi on vielä muitakin, mutta niiden hienouksiin ei ole tarkoituksenmukaista perehtyä tässä tutkielmassa. Seuraavaksi tutustutaan vielä lyhyesti kolmeen ongelmatyyppiin, joista saa lisätietoa ainakin Leipälän monisteesta [6] sivuilta 36-43.

Dynaamiset ongelmat

Dynaamisuus on eräs tapa jakaa lineaariset ongelmat kahteen luokkaan: staattisiin ja dynaamisiin (engl.dynamic, time phased). Luokkien ero on siinä, että dynaamiset ongelmat ovat ajasta riippuvia ja niissä ongelmaa tarkastellaan eri aikoina. Tällaisia ongelmia ovat esimerkiksi kiinteäkorkoisiin lainoihin ja niiden takaisinmaksuun liittyvät ongelmat.

Dynaamisiin ongelmiin liittyy termi *suunnitteluperiodi* (engl. time horizon), jonka aikana ongelmaa tarkastellaan. Hyvän suunnitteluperiodin valinta on tärkeää dynaamisissa malleissa, koska se vaikuttaa tuloksiin suuresti. Esimerkiksi jos yritys haluaa maksimoida viikon päästä kassassa olevan rahasumman ja suunnitteluperiodi on tuo viikko, voi vaikuttaa järkevältä ottaa valtava laina. Tällöin kassaan saadaan paljon rahaa, mutta yritys jatkaa toimintaansa myös viikon jälkeen, jolloin laina alkaa kasvaa korkoa, eikä sen ottaminen enää ollutkaan hyvä idea.

Edellä kuvattuun ongelmaan on useita ratkaisuja, ja ehkä selkein niistä on käyttää *ääretöntä suunnitteluperiodia* eli *ääretöntä suunnittelukautta* (engl. infinite time horizon). Tässä menetelmässä mallista tehdään syklinen niin, että edellisen suunnitteluperiodin lopputila otetaan seuraavan kauden lähtökohdaksi. Näin mallilla voidaan kierrosten kokonaismäärää säätämällä tutkia päätösten vaikutusta hyvinkin pitkällä ajanjaksolla.

Työvuorosuunnittelu

Työvuorosuunnittelu (engl. shift scheduling, staff planning) on eräänlainen välimuoto staattisen ja dynaamisen mallin välillä: toisaalta ne voivat toimia aikariippuvasti mutta ne on myös helppo mieltää staattisiksi *peittoongelmiksi*, jossa rajoitetuilla resursseilla yritetään kattaa tai peittää vaadittu tai mahdollisimman suuri alue. Tavoitteena on suorittaa ennalta tiedetty määrä työtä työntekijöiden järkevällä sijoittelulla työvuoroihin. Vaaditut työn suoritukset ovat peittorajoituksia (engl. cover constraint), jotka täytyy täyttää tai "peittää" työntekijöiden työpanoksella kussakin työvuorossa. Kussakin vuorossa täytyy siis olla riittävän monta työntekijää kun yksittäisen työntekijän työpanos tiedetään. Pienet työvuorosuunnittelun ongelmat ovat usein myös diskreettejä. Esimerkiksi kolmen työntekijän jakamisessa kolmeen työvuoroon optimi voisi olla 2,56 työntekijää ensimmäisessä ja 0,44 kolmannessa vuorossa, mikä ei kuulosta kovin helpolta toteuttaa järkevästi käytännössä.

Työvuorosuunnitteluun liittyvissä ongelmissa on usein viisasta käyttää päätösmuuttujina tietyntyyppisten työntekijöiden määrää kussakin vuorossa tai kunakin tuntina. Voi olla selkeyttävää laatia taulukko kaikista mahdollisista työvuoroista. Tehtävän ehdot voivat tuoda tehtävään mukaan muita muuttujia, kuten rästissä olevan työn määrän. Jos työaika on liukuva ja jos on monia erilaisia työntekijöitä, tehtävistä tulee helposti hyvin suuria.

Linearisoituvat epälineaariset tehtävät; Minimax

Epälineaariset tehtävät ovat aina hankalampia kuin lineaariset: ne noudattavat harvempia rajoituksia ja ovat tietokoneilla ratkaistaessa suurempia ja vievät enemmän aikaa. Niinpä kannattaa pyrkiä pitämään ratkaistava ongelma lineaarisena niin pitkään kuin mahdollista.

Vaikka ongelma olisikin epälineaarinen, se voi silti joissakin tapauksissa olla linearisoituva. Tällaisia ovat esimerkiksi erilaiset *minimax*- ja *maximin*-ongelmat, joissa minimoidaan maksimia tai toisin päin. Näitä kutsutaan myös *pullonkaula* (engl. bottleneck) ongelmiksi.

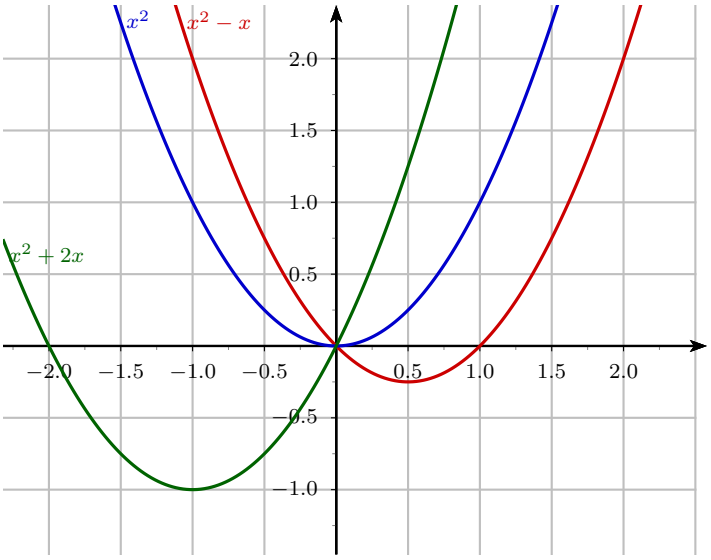
Minimin maksimointi voi kuulostaa ristiriitaiselta, joten annetaan siitä esimerkki: Millä paraabeleista $f(x) = x^2 + ax$ on suurin minimi, kun $a = -1,\ 0$ tai 2? Näissä tilanteissa funktion kuvaaja on esitetty kuvassa 3.4. Selvästi minimi on huipussa, joten saadaan arvot $-0,25,\ 0$ ja -1. Siispä suurin minimi saadaan kun $a = 0$ ja minimin arvo on tällöin myös 0. Tällaiset ongelmat linearisoituvat, kun tehtävän jälkimmäinen osa muutetaan uudeksi rajoitteeksi esimerkiksi funktion f suhteen, ja tehtävä muotoillaan tämän funktion maksimoinniksi tai minimoinniksi. Esimerkiksi tapauksessa

$$\max \min_{i=1,2,\ldots,11} (a_i \cdot x_i)$$

voidaan kirjoittaa

$$\begin{array}{ll} \max & f \\ s.t. & f \leq a_i x_i, \quad i = 1, 2, \ldots, 11. \end{array}$$

Toinen linearisoituva epälineaarinen tehtävätyyppi ovat *minimipoikkeamamallit*, joissa minimoidaan niiden nimen mukaan kahden lausekkeen poikkeamaa toisistaan. Käytännössä tämä tarkoittaa lausekkeiden erotuksen itseisarvon minimointia, eli $\min |p(x) - q(x)|$. Tällainen tehtävä saadaan yleensä linearisoitua jakamalla erotus positiiviseen ja negatiiviseen osaan muodossa $p(x) - q(x) = s^+ - s^-$, kun $s^+ \geq 0$, $s^- \geq 0$ ja $s^+ s^- = 0$, jolloin $|p(x) - q(x)| = s^+ + s^-$, joka on lineaarinen. Menetelmä toimii, kunhan $s^+ s^- = 0$ toteutuu, mutta niin käy usein automaattisesti. [6]



Kuva 3.4: Funktion $f(x) = x^2 + ax$ kuvaajat, kun $a = -1,\ 0$ ja 2.

Luku 4

Epälineaarisesta optimoinnista

Kuten aiemmin on todettu, kaikki tehtävät, jotka eivät ole lineaarisia, ovat epälineaarisia. Yleisesti ottaen epälineaariset tehtävät ovat matemaattisesti hankalampia kuin lineaariset. Uusia hankaluuksia ovat esimerkiksi:

- Optimointitehtävällä ei välttämättä ole yksikäsitteistä optimia vaan esimerkiksi useita globaaleja minimejä.
- Optimoitavalla funktiolla voi olla useita lokaaleja minimejä.
- Muuttujien vaihteluvälit eli skaalat voivat olla erilaisia. Esimerkiksi funktiossa $f(x, y) = x^2 + xy$ x voi kuulua välille $[1, 100]$ ja y välille $[0; 0,001]$ tai toisin päin.
- Optimoitava funktio voi olla hankalaa tai raskasta laskea. Sen arvo voi esimerkiksi olla peräisin suuren luokan simulaatiosta tai differentiaaliyhtälöiden ryhmästä.
- Funktion derivaattoja ei voida laskea analyyttisesti tai se on hyvin hidasta ja vaikeaa. On myös mahdollista että funktio on "musta laatikko", jonka rakennetta ei tunneta. Tällaisissa tapauksissa derivaattaan tai gradienttiin perustuvien menetelmien (kuten NLM:n) käyttö on mahdotonta tai hyvin vaikeaa.
- Stokastisen optimoinnin tapauksessa optimointitehtävä sisältää satunnaismuuttujan, jonka arvojen jakauma saatetaan tietää. Tällöin optimin löytyminen jollakin satunnaismuuttujan arvolla on toteutuskelpoinen vain jossakin tämän arvon lähettyvillä eli jossakin sen ympäristössä. Käyttökelpoisen ratkaisun löytäminen voi vaatia tilastotieteen menetelmiä.
- Hyvän alkuarvauksen löytäminen on vaikeaa. Tämä aiheuttaa haasteita, koska monet optimointialgoritmit aloittavat jostakin pisteestä ja etenevät siitä kohti optimia. [1]

Matemaattisista haasteista huolimatta epälineaaristen tehtävien ymmärtäminen ei useinkaan ole erityisen hankalaa.

4.1 Epälineaarisen funktion sisältävät tehtävät

Jos optimoitava funktio on epälineaarinen, toinen tai kumpikaan ehdoista

- Verrannollisuus: kunkin päätösmuuttujan vaikutus kohdefunktioon ja rajoituksiin on suoraan verrannollinen päätösmuuttujan arvoon.
- Additiivisuus: kunkin päätösmuuttujan vaikutus kohdefunktioon ja rajoituksiin on riippumaton muiden päätösmuuttujien arvoista.

ei ole voimassa. Esimerkiksi funktio $f(x) = x^2$ riittää rikkomaan verrannollisuusehdon ja funktio $g(x, y) = x \cdot y$ rikkoo additiivisuusehdon. Matemaattisesti tällaisten funktioiden optimointi on hankalampaa kuin lineaaristen, koska optimi voi olla myös muualla kuin kärkipisteissä. Esimerkiksi minimoitaessa funktiota $f(x) = x^2$ välillä $(-2, 1)$, kärkipisteet vastaavat välin päätepisteitä, mutta minimi saavutetaan kun $x = 0$.

Esimerkki 15. Metallipajassa halutaan valmistaa metallilevystä 1,3 litran vetoisia pyöreitä säilykepurkkeja käyttäen mahdollisimman vähän metallia. Purkeissa täytyy olla pohja ja kansi. Metallin leikkaamisesta ei synny hukkapaloja. Mitkä ovat parhaan mahdollisen säilykepurkin mitat millimetrin tarkkuudella?

Tehtävästä saadaan optimointitehtävä, jossa r kuvaa pohjan sädettä ja y lieriön korkeutta senttimetreissä

$$\begin{array}{ll} \min & 2 \cdot \pi r^2 + 2 \cdot \pi r y = 2\pi r(r + y) \\ s.t. & 2\pi r^2 y = 1300 \\ & r, y \geq 0 \end{array} .$$

Ratkaistaan tehtävä GeoGebralla. Koska GeoGebra ymmärtää syötetyt funktiot vain muodossa $f(x) = \ldots$, muutetaan tehtävässä oleva yhtälörajoite tähän muotoon.

$$2\pi r^2 y = 1300$$

$$y = \frac{1300}{2\pi r^2}$$

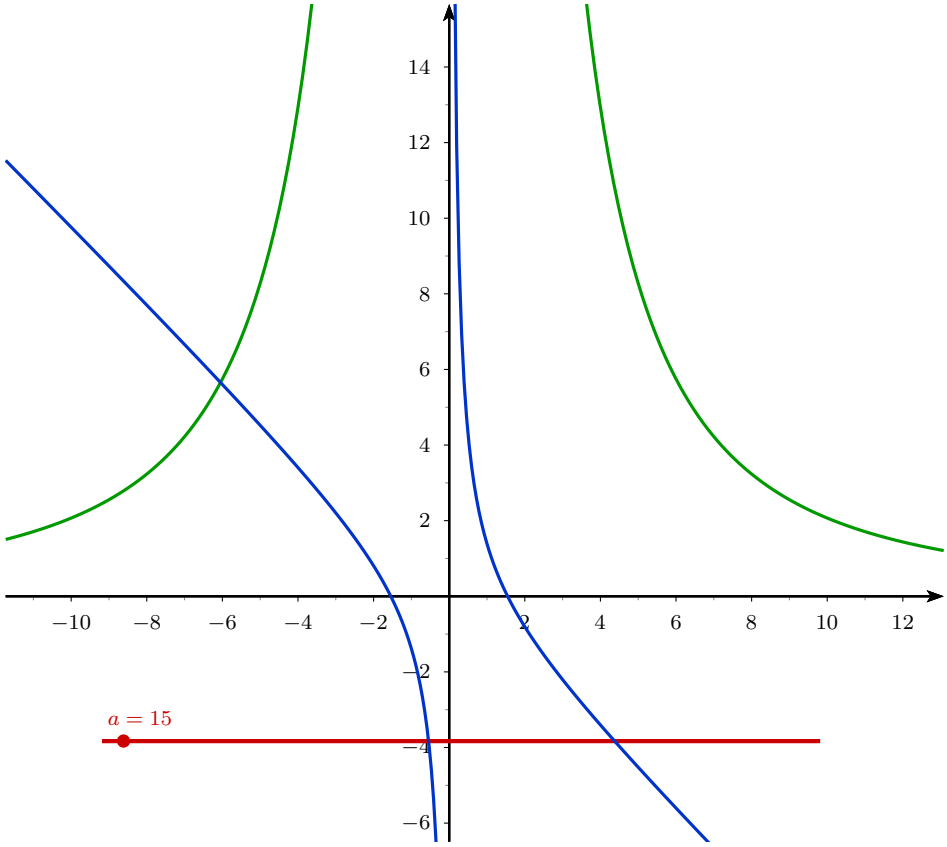
Samaan muotoon muutetaan myös minimoitava funktio, mutta annetaan minimille ennen sitä parametrin a arvo, jolloin saadaan

$$2 \cdot \pi r^2 + 2 \cdot \pi r y = a$$

$$2 \cdot \pi r y = a - 2 \cdot \pi r^2$$

$$y = \frac{a - 2 \cdot \pi r^2}{2 \cdot \pi r}.$$

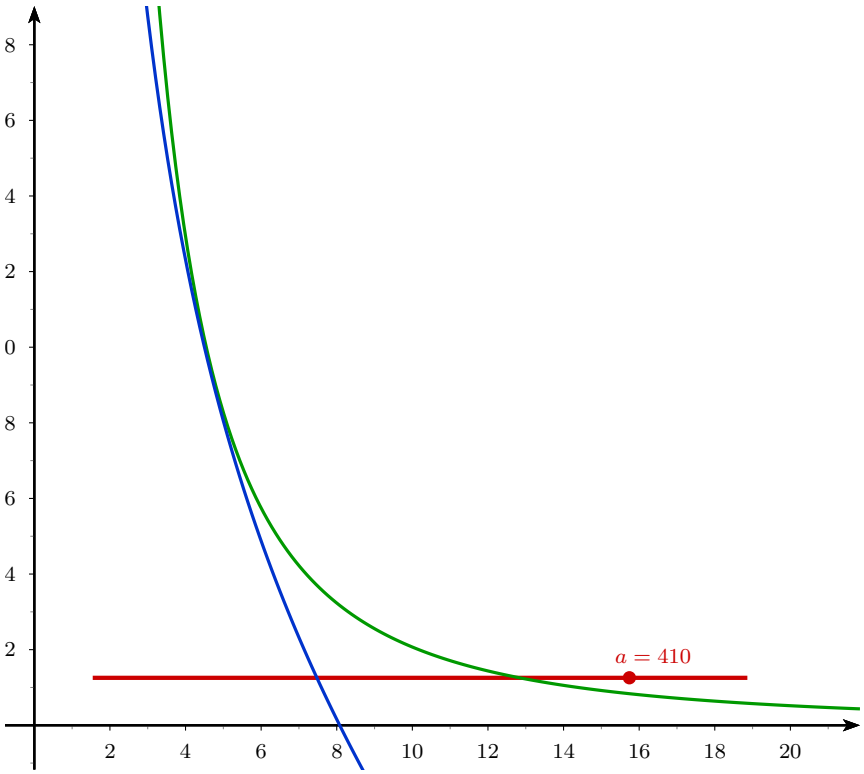
Nyt syötetään tehtävä GeoGebraan luomalla ensin ”liuku” nimeltä a ja syöttämällä sen jälkeen yllä saadut yhtälöt ohjelman syöttökenttään. Tässä ratkaisussa liu'un arvot kuuluvat välille [0,500] ja ne ovat 0,05 välein. Lisäksi liu'un ominaisuuksista on asetettu sen väri punaiseksi, paksuus 4:ksi, pituus 500 pikseliksi ja se on siirretty piirtoalueen alalaitaan. Syötettäessä yhtälöitä vaihdetaan muuttujan r nimi ja kirjoitetaan sen sijaan x, koska GeoGebra ei hyväksy r:ää. Tässä ratkaisussa syötettiin ensin rajoite ($f(x)$ vihreällä) ja sen jälkeen kohdefunktio ($g(x)$ sinisellä). Tuloksena saatiin kuvan 4.1 kaltainen tilanne.



Kuva 4.1: Esimerkin 15 alkutilanne.

Seuraava vaihe ratkaisussa on siirtää liukua niin, että $f(x)$ ja $g(x)$ koskettavat toisiaan koordinaatiston ensimmäisessä neljänneksessä, missä x ja y ovat positiivisia. Kuvassa 4.2 on kuva tilanteesta, jossa aletaan olla lähellä oikeaa arvoa.

Muuttamalla liu'un väliä ja askelpituutta väliksi [413,417] askelpituudel-



Kuva 4.2: Esimerkin 15 etsintävaihe.

la 0.01 ja tarkentamalla kohtaan, jossa käyrät näyttävät yhtyvän, huomataan että minimi on noin 415,4 cm^2. Käyttämällä kahden objektin leikkauspisteen merkitsemistä, saadaan likimain pisteet (4,70759; 9,3361), (4,68118; 9,44175), kun minimi olisi 415,39 cm^2. Voidaan havaita, että x:n arvo on välillä [4,68; 4,71], joten $r = x \approx 4,7\text{cm}$. Samoin voidaan päätellä, että $y \approx 9,4\text{cm}$, vaikka tämä arvo hipookin riittävän tarkkuuden rajaa.

Esimerkki oli melko työläs, koska käyrät lähestyivät toisiaan hitaasti, eikä yksittäisen leikkauspisteen löytäminen ollut helppoa. Suoritus on kuitenkin hyvä ohjelmalta, joka ei ole edes suunniteltu optimointia varten. Mielenkiinnon vuoksi todettakoon, että OpenOfficen NLP-ratkaisin[1] löysi saman optimin 10,07 sekunnissa 198 laskentakierroksen jälkeen, mutta vasta kun yhtälörajoite oli muutettu epäyhtälöksi $2\pi r^2 y \geq 1300$, eli sitä oli *relaksoitu*. Relaksoimiseen palataan osiossa 6.1 sivulla 92. Ilman tätä muutosta laskenta pääsi 4000 kierroksen jälkeen vasta minimin arvoon, joka oli yli 600. Ero johtunee siitä, että käytetty algoritmi on evoluutiota jäljittelevä ja toimii paremmin, kun saa ”liikkumatilaa”. Yhtälörajoite kahlitsee arvojen muutokset

[1]Käytössä oli Solver for Nonlinear Programming 0.9; Sun Microsystems; DEPS Evolutionary algorithm.

yksittäiselle suoralle, kun taas epäyhtälö antaa satunnaisille muutoksille ja parantaville suunnille enemmän vaikutusta. Osiossa 5.3.2 todetaankin, etteivät geneettiset algoritmit toimi erityisen hyvin rajoitetuissa tilanteissa.

4.2 Monitavoiteoptimointi

Monitavoiteoptimoinnissa on useita optimointitavoitteita. Tähän asti käsiteltyihin ongelmiin nähden monitavoiteongelmat ovat joka suhteessa hankalampia, koska niihin harvemmin on yhtä oikeaa ratkaisua. Usein optimoinnin kohteena olevat asiat ovat jopa eri suureita tai niille on hankalaa edes määrittää suuretta.

Tutustutaan ensin tehokkaan pisteen käsitteeseen. *Tehokas* eli *Pareto optimaalinen* piste on sellainen sallittu piste, joka on kaikkien kriteerien suhteen ainakin yhtä hyvä kuin mikä tahansa muu piste. Voidaan myös sanoa, että tällainen piste on dominoimaton, eli mikään piste ei dominoi sitä. Tehokas piste ei ole yksikäsitteinen optimi, jos on olemassa pisteitä, jotka tuottavat jonkin kriteerin suhteen paremman arvon, vaikka muiden kriteerien arvot olisivat heikompia. Tällainen tapaus on esimerkissä 16. Tehokkaita pisteitä käytetään osana muita monitavoiteoptimoinnin ratkaisutyylejä.

Monitavoiteoptimoinnista on syytä tietää kolme perustavaa ratkaisutyyliä.

Leksikografisessa optimoinnissa eri tavoitteita painotetaan niin, että jotkin niistä ovat toisia tärkeämpiä. Tehtävä ratkaistaan ensin tärkeimmän tavoitteen suhteen ja kiinnitetään tavoitteeseen liittyvien muuttujien arvot. Sen jälkeen tehtävä ratkaistaan toiseksi tärkeimmän tavoitteen mukaan kiinnitetyillä arvoilla ja kiinnitetään tähänkin tavoitteeseen liittyvät arvot. Näin jatketaan kunnes kaikki tavoitteet on käyty läpi tai kaikkien muuttujien arvot on kiinnitetty.

Tavoiteoptimointi on suosituimpia monitavoiteoptimointitehtävien ratkaisumenetelmiä. Siinä jokaiselle tavoitteelle määritetään raja, johon pyritään, mutta joka voidaan tarvittaessa ylittää. Saatujen ratkaisujen paremmuutta arvioidaan sillä, miten kaukana tavoitteet ovat niille asetetuista rajoista. Tavoitteille voi tässä menetelmässä myös antaa erilaiset painotukset niin, että jonkin tavoitteen toteutuminen on tärkeämpää kuin toisten.

Painokerroinmenetelmä on myös erittäin suosittu toimintatapa monitavoiteoptimoinnissa. Siinä pyritään muotoilemaan kaikki tavoitteet ja muuttujat siten, että sopivasti painottaen niitä voidaan verrata keskenään. Näin saadaan yksitavoitteinen optimointitehtävä. Menetelmän ongelmana on sopivien painotuskertoimien löytäminen ja se, että tavoitteiden arvojen vertaaminen keskenään voi olla hyvin hankalaa tai mielipidekysymys.

Esimerkki 16. Marsiin suunnitellaan avaruuslentoa ja sen aikana tärkeysjärjestys on 1) eloonjääminen ja terveys, 2) psyykkinen terveys, 3) varusteiden kunto ja 4) mukavuus. Alukseen voidaan ottaa kahdeksaa erilaista hyödykettä, joilla on vaikutuksensa edellä mainittuihin asioihin taulukon 4.1 mukaisesti. Hyödykkeiden yhteispaino saa olla korkeintaan 2000 kg. Mitä kannattaa ottaa mukaan ja miten paljon?

			Vaikutus/kg			
Hyödyke	Määrä	Enimmäismäärä	1	2	3	4
1	x_1	400	4	3	0	1
2	x_2	150	5	2	0	0
3	x_3	100	0	1	1	3
4	x_4	500	0	0	4	1
5	x_5	400	3	1	1	2
6	x_6	300	0	5	4	4
7	x_7	250	0	0	1	3
8	x_8	100	0	0	0	4

Taulukko 4.1: Esimerkin 16 avaruuslennolle otettavien hyödykkeiden tietoja.

Kun ratkaistaan tämä tehtävä leksikografisesti, saadaan neljä optimointitehtävää, jotka ratkaistaan peräkkäin. Näistä ensimmäinen on

$$
\begin{aligned}
\max \quad & 4x_1 + 5x_2 + 3x_5 \\
s.t. \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 2000 \\
& x_1 \leq 400 \\
& x_2 \leq 150 \\
& x_3 \leq 100 \\
& x_4 \leq 500 \\
& x_5 \leq 400 \\
& x_6 \leq 300 \\
& x_7 \leq 250 \\
& x_8 \leq 100 \\
& x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \geq 0
\end{aligned}
\quad .
$$

Ratkaisuksi saadaan arvot $x_1 = 400, x_2 = 150, x_5 = 400$. Kiinnitetään nyt nämä arvot ja siirrytään seuraavaan optimointitehtävään. Kohdefunktiosta voidaan jättää vakiot pois, koska ne eivät vaikuta maksimin sijaintiin.

$$
\begin{array}{ll}
\max & x_3 + 5x_6 \\
s.t. & 400 + 150 + x_3 + x_4 + 400 + x_6 + x_7 + x_8 \leq 2000 \\
& x_3 \leq 100 \\
& x_4 \leq 500 \\
& x_6 \leq 300 \\
& x_7 \leq 250 \\
& x_8 \leq 100 \\
& x_3, x_4, x_6, x_7, x_8 \geq 0
\end{array}
$$

Optimi on kohdassa $x_3 = 100, x_6 = 300$ ja tässä vaiheessa hyödykkeitä on valittu mukaan 1350kg. Voidaan kiinnittää saadut arvot ja siirtyä seuraavaan vaiheeseen.

$$
\begin{array}{ll}
\max & 4x_4 + x_7 \\
s.t. & 1350 + x_4 + x_7 + x_8 \leq 2000 \\
& x_4 \leq 500 \\
& x_7 \leq 250 \\
& x_8 \leq 100 \\
& x_4, x_7, x_8 \geq 0
\end{array}
$$

Maksimi on kohdassa $x_4 = 500, x_7 = 150$, jolloin koko kapasiteetti on käytetty. Hyödykettä 8 ei siis tule mukaan, eikä neljättä optimointitehtävää tarvitse ratkaista. Tällöin kriteerien arvot ovat 3550, 3500, 3850 ja 3650, kun suurimmat mahdolliset arvot ovat 3550, 3500, 3950 ja 4350. Optimipiste siis on $x_1 = 400, x_2 = 150, x_3 = 100, x_4 = 500, x_5 = 400, x_6 = 300, x_7 = 150, x_8 = 0$ ja kyseessä on tehokas piste, mutta ei yksikäsitteinen optimi, koska esimerkiksi $x_1 = 400, x_2 = 50, x_3 = 100, x_4 = 500, x_5 = 400, x_6 = 300, x_7 = 200, x_8 = 50$ tuottaa paremman mukavuuskriteerin arvon, mutta silloin tärkeämpien kriteerien arvot eivät enää ole yhtä hyviä.

4.3 Diskreetit optimointitehtävät

Diskreetti tarkoittaa epäjatkuvaa ja niinpä diskreeteissä optimointitehtävissä kaikki päätösmuuttujat saavat arvoja epäjatkuvasta joukosta. Esimerkiksi reaalilukujen joukko on jatkuva, mutta kokonaislukujen tai murtolukujen joukot eivät ole. Näin siis mikä tahansa kappaletavaraa käsittelevä optimointitehtävä, joka ei salli yksittäisten kappaleiden pilkkomista, on diskreetti. Hyvänä esimerkkinä voisi toimia tehtävä, jossa käsitellään ihmisten määrää ja lineaarisesti ratkaistuna saadaan tulokseksi 9,65. Tällainen tulos ei ole järkevä, koska 0,65 ihmistä on usein mahdoton mitata[2]. Niinpä tehtävä kannattaa ratkaista kokonaislukuoptimoinnilla, joka on osa diskreettiä

[2]John James Rambolta se saattaa onnistua, mutta ei puhuta siitä.

optimointia. Useimmat diskreetit optimointitehtävät sisältävätkin joko kokonaislukumuuttujia tai binäärisiä muuttujia. Jos osa päätösmuuttujista on diskrettejä ja osa jatkuvia, puhutaan sekalukuoptimoinnista tai lyhyemmin sekaoptimoinnista.

Binääriset tai diskreetit tehtävät saattavat aluksi kuulostaa helpoilta, koska niissä muuttujien ja mahdollisesti funktion arvojen joukot ovat rajatumpia. Tämä pitää kuitenkin paikkansa lähinnä vain raa'an voiman tai täydellisen luetteloinnin menetelmiä käytettäessä, jotka ovat yksinkertaisia ja usein tehottomia menetelmiä. Ongelman aiheuttaa se, ettei kohdefunktio ole jatkuva välttämättä yhdessäkään pisteessä, joten useiden menetelmien käyttämiä derivaattoja ei voida laskea. Lisäksi funktion käyttäytymistä voi olla hankala ennakoida. Esimerkiksi funktio $|\sin(x)|$ saa taulukkossa 4.2 esitetyt arvot, kun $x = 1, 2, \ldots, 10$.

x	$\sin(x)$	x	$\sin(x)$
1	0,84147...	6	0,27941...
2	0,90929...	7	0,65698...
3	0,14112...	8	0,98935...
4	0,75680...	9	0,41211...
5	0,95892...	10	0,54402...

Taulukko 4.2: Eräitä funktion $|\sin(x)|$ arvoja.

On siis hankalaa arvata, saadaanko seuraavassa pisteessä parempi vai huonompi arvo. Tietenkin funktion kulun tuntemista voi käyttää apuna tässä, ja säännöllisen trigonometrisen funktion tapauksessa se on helppoa. Epäsäännöllisten funktioiden tapauksessa diskreetti optimointi voi kuitenkin olla hyvin hankalaa.

4.3.1 Selkäreppu- ja budjetointiongelmat

Selkäreppuongelmat (engl. knapsack) ovat saaneet nimensä tehtävästä, jossa retkeilijän tai kullankaivajan täytyy valita kaikkein arvokkaimmat tavarat mukaan otettaviksi. Tällöin joko repun tilavuus tai paino tulee rajoitteeksi. Yleisemmin pitää siis valita paras mahdollinen joukko kohteita, ominaisuuksia, projekteja tai investointeja ylittämättä resurssirajoitusta ja kelpuuttamatta osittaisia ratkaisuja. Nykyoptimoinnissa selkäreppuongelmilla tarkoitetaan lineaarista kokonaislukuoptimointitehtävää, jossa on vain yksi rajoitus.

Esimerkki 17. Tehtaan kokoonpanolinjan nopeutta halutaan tehostaa ja on esitetty seitsemää erilaista muutosta, joiden vaikutukset on esitetty taulukossa 4.3.

Muutos j	1	2	3	4	5	6	7
Kustannus (1000 eur.)	4,4	9,4	12,0	54,2	19,2	8,7	35,2
Nopeuden lisäys (1000 kpl/h)	3	7	11	78	43	15	30

Taulukko 4.3: Esimerkissä 17 olevan tehtaan mahdolliset muutokset.

1. Mikä on maksimaalinen nopeuden parannus, kun käytössä on 55000 euroa?
2. Miten saadaan mahdollisimman halvalla vähintään 50000 tuotteen lisäys tunnissa?

Valitaan ensin päätösmuuttujiksi

$$x_j = \begin{cases} 1, & \text{jos muutos } j \text{ toteutetaan} \\ 0, & \text{muuten.} \end{cases}$$

1. Nyt saadaan optimointitehtävä

$$\begin{array}{ll} \max & 3x_1 + 7x_2 + 11x_3 + 78x_4 + 43x_5 + 15x_6 + 30x_7 \\ s.t. & 4,4x_1 + 9,4x_2 + 12x_3 + 54,2x_4 + 19,2x_5 + 8,7x_6 + 35,2x_7 \leq 55 \\ & x_1, \ldots, x_7 = 0 \vee 1 \end{array}$$

2. Nyt tehtävä on tavallaan käänteinen edelliseen nähden ja saadaan

$$\begin{array}{ll} \min & 4,4x_1 + 9,4x_2 + 12x_3 + 54,2x_4 + 19,2x_5 + 8,7x_6 + 35,2x_7 \\ s.t. & 3x_1 + 7x_2 + 11x_3 + 78x_4 + 43x_5 + 15x_6 + 30x_7 \geq 50 \\ & x_1, \ldots, x_7 = 0 \vee 1 \end{array}$$

Esimerkiksi OpenOffice on kykenevä ratkaisemaan tällaiset tehtävät, mutta niiden ratkaiseminen on helppoa myös käsin. Seuraavaksi ratkaistaan edelisistä tehtävistä ensimmäinen kynällä ja paperilla.

Lasketaan jokaiselle muutokselle hyötysuhde $h_j = \frac{c_j}{a_j}$, missä c_j on nopeuden lisäys ja a_j hinta. Näin saadaan taulukon 4.4 arvot.

Muutos j	1	2	3	4	5	6	7
Hyötysuhde	0,6...	0,7...	0,9...	1,4...	2,2...	1,7...	0,8...

Taulukko 4.4: Esimerkissä 17 olevien mahdollisten muutosten hyötysuhteet.

Nyt valitaan niin monta muutosta kuin mahdollista, aloittaen siitä, jonka hyötysuhde on paras ja saadaan muutokset 5, 6, 3, 2 ja 1, jolloin kokonaiskustannus on 53 700 euroa ja nopeuden lisäys on 79 000 kappaletta tunnissa. Tällaista menettelyä voidaan kutsua "ahneeksi", koska se ottaa mukaan aina sen vaihtoehdon, joka näyttää parhaalta ajattelematta kokonaisuutta.

Tämä menetelmä voi kohdata ongelmia, jos "muutosten" joukossa on jokin hyvin lähellä sallittua hintaa oleva työ, jonka hyötysuhde on melko hyvä, mutta ei paras. Tällainen tilanne saataisiin, jos esimerkiksi muutos nro. 4 toisi nopeuden lisäystä 90 000 kappaletta eikä 78 000. Olkoon tämä osoitus päätöksentekijän ja tulosten järkevän tulkinnan tärkeydestä.

Jälkimmäisen tehtävän ratkaisuksi saadaan muutokset 5 ja 6, jolloin nopeus lisääntyy 58 000 kappaleella tunnissa ja kustannus on 27 900 euroa.

Pääoman budjetoinnissa on kyse moniulotteisesta selkäreppuongelmasta. Tämä on tilanne, jos on useita rajoituksia tai useita suunnittelukausia, kuten vaikkapa esineen paino ja tilavuus reppua pakattaessa. Näiden lisäksi voi olla vielä joitakin *toisensa poissulkevuus*ehtoja, tai jokin projekti tai tavara on otettava mukaan ennen jotakin toista, eli niillä on *edeltäjärelaatio*. Jos projektit l ja k ovat toisensa poissulkevat, voidaan tämä esittää rajoituksella $x_k + x_l \leq 1$. Jos taas projektin n edellytyksenä on projekti m, saadaan tästä rajoitus $x_n \leq x_m$.

Kokkareiset ongelmat

Kokkareisissa (engl. lumpy) ongelmissa on mukana rajoituksia, jotka ovat joko-tai -tyyppisiä tai kohdefunktioon liittyy kiinteä aloituskustannus. Esimerkiksi uuden tuotantolinjan ylösajoon kuluu aikaa ja rahaa jo ennen kuin se tuottaa yhtään mitään. Kokkareiset ongelmat muistuttavat siis hieman budjetointiongelmia.

Kokkareen muodostaa esimerkiksi muuttuja, josta on valittava kaikki tai ei mitään, eli

$$x_j = \begin{cases} u_j, & \text{jos } x_j \text{ valitaan} \\ 0, & \text{muuten.} \end{cases}$$

Tällaisessa tapauksessa voidaan ottaa käyttöön binäärinen muuttuja $y_j = 0 \vee 1$ ja kirjoittaa $x_j = u_j y_j$.

Toisenlaisen kokkareen muodostaa muuttuja, jolla on kiinteä aloituskustannus, eli muuttujasta x_j aiheutuu kustannus

$$f(x_j) = \begin{cases} a_j + c_j x_j, & \text{jos } x_j > 0 \\ 0, & \text{jos } x_j = 0. \end{cases}$$

Tässä a_j on kiinteä aloituskustannus ja c_j jokin vakio. Tällainen kokkare voidaan mallintaa ottamalla käyttöön uudet päätösmuuttujat

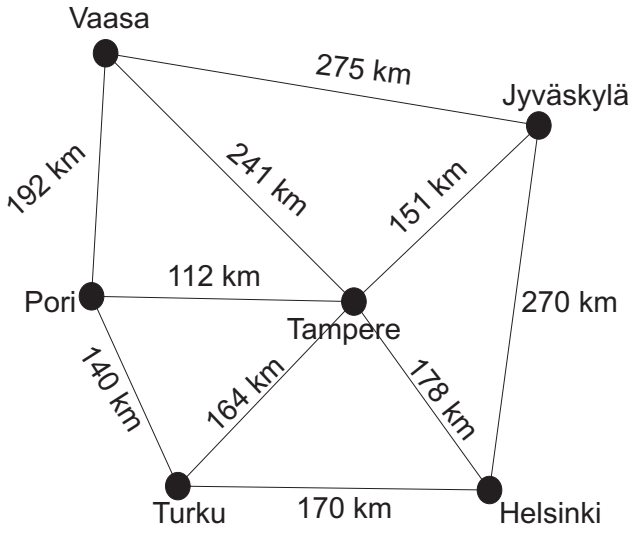
$$y_j = \begin{cases} 1, & \text{kun } x_j > 0 \\ 0, & x_j = 0. \end{cases}$$

Näiden avulla voidaan kohdefunktiota muokata niin, että $f(x_j) = a_j y_j + c_j x_j$. Täytyy myös kytkeä muuttujat x_j ja y_j toisiinsa lisäämällä tehtävään rajoituksia $x_j \leq u_j y_j$, missä u_j on niin suuri parametri, ettei x_j voi ylittää sitä. Näin x_j ei voi olla positiivinen kun $y_j = 0$.

4.3.2 Verkko-ongelmat

Kaikista optimointiongelmista verkko-ongelmat saattavat olla helpoimpia ymmärtää intuitiivisesti. Havainnollistetaan tätä hieman: Lukijaa pyydetään ajattelemaan jotakin usein kuljettua matkaa, kuten koulu- tai työmatkaa. Mikä on paras reitti kyseiselle matkalle? Tämä on eräänlainen verkko-ongelma, jossa tiet, kadut tai huoneet muodostavat verkon, jonka risteyksien välillä kuljetaan. Jokaisen risteysparin väliselle reitille pystytään määrittämään matka ja siihen kulunut aika. Paras reitti on usein lyhin tai nopein reitti, jolloin ongelmaa sanotaan *lyhimmän tien ongelmaksi*. Koska kyseessä on helposti ymmärrettävä ongelma, käsitellään sitä esimerkin avulla.

Esimerkki 18. Alla olevassa kuvassa on esitetty joidenkin Suomen kaupunkien summittaiset sijainnit ja etäisyydet toisistaan nopeinta reittiä pitkin. Kuvaan on myös merkitty kaupunkien väliseen matkaan kuluva aika nopeinta reittiä pitkin. [19] Lähettiyritys toimii Helsingistä ja toimittaa lähetyksiä sieltä Turkuun, Poriin, Tampereelle, Jyväskylään ja Vaasaan.



Kuva 4.3: Joitakin Suomen kaupunkeja ja niiden välisiä etäisyyksiä. [19]

Mikä on lyhin reitti lähetykselle Vaasaan? Entä Poriin?

Koska kyseessä on pienehkö kartta todellisesta maailmasta, on houkuttelevaa käsitellä ensimmäistä kohtaa kuten mitä tahansa karttaa, ja tämä olisikin toimiva lähtökohta. Matemaattisen optimoinnin kannalta on kuitenkin mielekkäämpää tutkia sitä hieman eri tavalla. Tarkastellaan seuraavaksi hieman verkkojen teoriaa ja palataan sen jälkeen ongelman pariin esimerkissä 19.

Verkko-ongelmien käsittely ei ole vain optimoinnin alla, vaan verkkoja käsitellään myös tietojenkäsittelytieteessä ja on olemassa graafiteoriaksi

kutsuttu matematiikan ala, joka keskittyy pelkästään verkkojen eli graafien tutkimukseen [3]. Koska verkot ovat näin suosittuja, myös termistö saattaa hieman vaihdella lähteestä riippuen. Perustermit ovat kuitenkin usein samat. Ne käydään läpi seuraavaksi.

Verkko eli graafi (engl. graph) koostuu *solmuista* ja niitä yhdistävistä *kaarista*. Kuva 4.3 on esimerkki graafista.

Solmu (engl. node, vertex, point) on graafin piste. Kuvassa 4.3 kaupungit ovat graafin solmuja.

Kaari (engl. edge, line, link) on reitti tai linkki kahden solmun välillä. Kuvassa 4.3 tiet ovat graafin kaaria. On mahdollista, että kaari tekee silmukan niin, että alku- ja lähtösolmu ovat sama solmu, mutta sellaisiin tapauksiin tuskin optimoinnissa törmätään.

Kaaren paino (engl. weight) on luku, joka kuvaa kustannusta tai tilan muutosta kuljettaessa kaarta pitkin solmusta toiseen. Jos kaarten painoja ei ole merkitty, jokaisen kaaren paino on yksi. Kuvassa 4.3 jokaisella kaarella eli tiellä on paino, joka on merkitty kilometreinä.

Solmun naapureita (engl. neighbour) ovat ne solmut, joihin pääsee kulkematta toisten solmujen kautta. Kuvassa 4.3 Turun naapureita ovat Helsinki, Tampere ja Pori.

Suunnattuja kaaria voi kulkea vain yhteen suuntaan, ja niitä kutsutaan usein *nuoliksi*, koska ne piirretään nuolina. Jos graafissa on suunnattuja kaaria, se on *suunnattu graafi* (engl. directed graph). Kuvassa 4.3 ei ole suunnattuja kaaria, mutta niitä voisi olla, jos kartassa olisi yksisuuntaisia teitä.

Polku (engl. path) on reitti tai kaarten jono polun ensimmäisestä solmusta viimeiseen. Kuvassa 4.3 eräs polku on Pori-Vaasa-Jyväskylä-Helsinki.

Sykli (engl. cycle) on polku, jonka alku- ja loppusolmu ovat sama solmu. Kuvassa 4.3 eräs sykli on polku Turku-Pori-Vaasa-Tampere-Turku.

Puu (engl. tree) on verkko, jossa ei ole syklejä. Yleisesti ottaen syklittömät verkot ovat helpompia käsitellä kuin syklilliset.

Koska termit yksinään eivät ratkaise mitään, tarkastellaan seuraavaksi algoritmia lyhimmän tien etsimiseen. Kyseessä on Djikstran algoritmi [7], ja se toimii, kun verkossa ei ole lainkaan negatiivisia painoja. Algoritmi kertoo etäisyydet yhdestä solmusta kaikkiin muihin. Vaihtoehtoisia algoritmeja olisivat Bellman-Ford, Floyd-Warshall, Prim tai Kruskal. Algoritmit on usein nimetty niiden keksijän mukaan.

1. (Alustus) Alkusolmuna on s, Kreikkalainen kirjain 'nyy' ν kuvaa etäisyyttä ja i graafin solmujen indeksinumeroa. Asetetaan

$$\nu[i] \leftarrow \begin{cases} 0, & \text{jos } i = s \\ \infty, & \text{muulloin.} \end{cases}$$

   Merkitään kaikkien solmujen etäisyydet väliaikaisiksi ja kiinnitetään seuraavaksi solmun $p \leftarrow s$ etäisyys.

2. (Prosessointi) Merkitään solmu p kiinnitetyksi ja jokaiselle siitä kiinnittämättömään solmuun i lähtevälle nuolelle (p, i) päivitetään etäisyydet

$$\nu[i] \leftarrow \min\{\nu[i], \nu[p] + c_{pi}\},$$

   missä c_{pi} tarkoittaa solmusta p solmuun i menevän nuolen painoa. Jos $\nu[i]$ muuttuu, asetetaan i:lle edeltäjä $d[i] \leftarrow p$.

3. (Lopetus) Jos väliaikaisia solmuja ei enää ole, lopetetaan. Arvot $\nu[i]$ ilmoittavat lyhimmät etäisyydet solmuun i.

4. (Seuraava kiinnitetty solmu) Kiinnitetään seuraavaksi se solmu p, jonka väliaikainen etäisyys on pienin, toisin sanoen

$$\nu[p] = \min\{\nu[i] \mid i \text{ väliaikainen}\}.$$

   Tämän jälkeen palataan vaiheeseen 1.

Esimerkki 19. Palataan esimerkin 18 käsittelyyn. Sovelletaan Djikstran algoritmia selvitettäessä lyhimmät reitit Helsingistä kaikkiin muihin kaupunkeihin. Kannattaa kuitenkin laskea etäisyydet vain kaupunkeihin, joihin on toimituksia. Indeksoidaan kaupungit seuraavasti: 1=Helsinki, 2=Turku, 3=Pori, 4=Tampere, 5=Jyväskylä ja 6=Vaasa. Algoritmin toiminta on esitetty taulukoissa 4.5 ja 4.6.

solmu	Kokonaismatkat					
p	$\nu[1]$	$\nu[2]$	$\nu[3]$	$\nu[4]$	$\nu[5]$	$\nu[6]$
-	0	∞	∞	∞	∞	∞
1	0	170	∞	178	270	∞
2	0	170	310	178	270	∞
4	0	170	290	178	270	419
5	0	170	290	178	270	419
3	0	170	290	178	270	419
6	0	170	290	178	270	419

Taulukko 4.5: Esimerkissä 19 käytetyn algoritmin löytämät kokonaisetäisyydet.

solmu	Solmujen edeltäjät					
p	$d[1]$	$d[2]$	$d[3]$	$d[4]$	$d[5]$	$d[6]$
1	-	1	-	1	1	-
2	-	1	2	1	1	-
4	-	1	4	1	1	4
5	-	1	4	1	1	4
3	-	1	4	1	1	4
6	-	1	4	1	1	4

Taulukko 4.6: Esimerkissä 19 käytetyn algoritmin löytämät edeltäjät.

Taulukot saattavat näyttää hieman sekavilta, joten valotetaan niiden sisältöä hieman. Kokonaisetäisyyttä etsivä taulukko 4.5 esittää tilanteen solmu solmulta korvaten aiemman etäisyyden uudella, jos on löydetty lyhyempi reitti. Edeltäjien taulukko 4.6 puolestaan kertoo, minkä solmujen kautta lyhin löydetty reitti kuhunkin solmuun kulkee. Kunkin sarakkeen arvo kertoo, missä solmussa on käytävä ennen sarakkeen kuvaamaa solmua, jotta reitti olisi lyhin mahdollinen. Siispä lyhin reitti on oikeastaan ilmoitettu taulukossa takaperin.

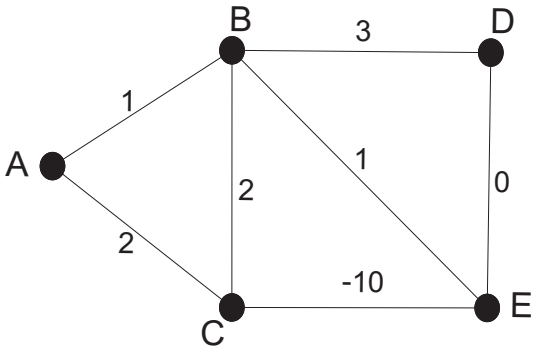
Siispä lyhimmät reitit, joita kysyttiin ovat Helsinki→Tampere→Vaasa 419 km ja Helsinki→Tampere→Pori 290 km.

Joskus graafeissa esiintyy ominaisuuksia, jotka joko hankaloittavat tai helpottavat niiden käsittelyä. Jos graafissa on suunnattuja kaaria tai nuolia, eli kyseessä on suunnattu graafi, voi syntyä tilanteita, joissa jostakin solmusta ei päästä enää pois tai solmuun saapuu useita nuolia, mutta siitä lähtee vain yksi. Tällaiset tilanteet voivat tehdä joistakin algoritmeista käyttökelvottomia.

Jos graafissa esiintyy negatiivisia painoja, optimoinnissa voi ilmetä yllättäviä ongelmia. Tarkastellaan alla olevaa graafia ja pyritään löytämään lyhin tie solmusta A solmuun E. Melko nopeasti huomataan, että lyhin reitti positiivisilla painoilla on A-B-E, jonka pituus painojen mukaan laskettuna on 2. On kuitenkin olemassa ääretön määrä parempia reittejä, kuten A-B-C-E, jonka pituus on -7, sekä A-B-E-C-B-E painolla -15. Koska graafissa on olemassa negatiivinen sykli, sitä kannattaa kulkea hamaan ikuisuuteen asti. Tällaiset tapaukset ovat todellisuudessa niin harvinaisia, että malli on lähes varmasti väärin rakennettu. Siispä minimoitavissa verkko-ongelmissa kannattaa välttää negatiivisia painoja tai ainakin negatiivisia syklejä.

Kauppamatkustajaongelma

Kauppamatkustajaongelma (engl. traveling salesman problem) on luultavasti kuuluisin verkko-ongelma. Siinä kauppamatkustaja haluaa käydä kaikissa



Kuva 4.4: Graafi, jossa yhdellä kaarella on negatiivinen paino.

jonkin alueen kaupungeissa tasan kerran ja palata lähtöpaikkaansa mahdollisimman lyhyttä tietä pitkin. Kaikista kaupungeista on yhteys kaikkiin muihin ja nämä etäisyydet tiedetään. Tällaisesta tiedosta on hyötyä esimerkiksi logistiikassa. Tämän ongelman muunnelma on esimerkiksi kysymys: Missä järjestyksessä piirilevyyn kannattaa porata tietty määrä reikiä tiettyihin kohtiin, jotta poraa tarvitsisi liikuttaa mahdollisimman vähän? Tämä saattaa kuulostaa mitättömältä, mutta elektroniikan liukuhihnatyössä poraaminen saattaa joskus olla hitain vaihe ja näin hidastaa tuotantoa.

Ongelma on hyvin vaikea ratkaistavaksi tietokoneella ja se onkin niin kutsuttu NP -täydellinen ongelma. Vaikeus piilee siinä, että jos kaupunkeja on n kappaletta, jokaisesta niistä voidaan päästä $n-1$ kaupunkiin, ja erilaisia reittejä on $(n-1)!$ kappaletta. Siispä viidelle kaupungille reittejä on $4! = 24$, kymmenelle 362880 ja 15:lle jo yli 87 miljardia. Ei siis liene yllätys, että on kehitetty useita menetelmiä, jotka etsivät parasta reittiä käymättä kaikkia vaihtoehtoja läpi. Vaikeuksista huolimatta työn alla on ratkaisu jopa koko maailman kauppamatkustajaongelmaan. Lisätietoja aiheesta saa muun muassa alla olevista osoitteista.

- Kauppamatkustajaongelma suomeksi (suppea):
  `http://fi.wikipedia.org/wiki/Kauppamatkustajan_ongelma`
- Kauppamatkustajaongelma englanniksi (laaja):
  `http://en.wikipedia.org/wiki/Travelling_salesman_problem`
- Koko maailman kauppamatkustajaongelma:
  `http://www.tsp.gatech.edu/world/`

Pisimmän tien ongelmat

Kuulostaa arveluttavalta etsiä verkosta pisintä tietä, jos verkko kuvaa karttaa. Niin se onkin. Verkko voi kuitenkin kuvata paljon muutakin kuin karttaa: se voi kuvata lähes mitä tahansa tilannetta, jossa siirrytään vaiheesta

toiseen, kuten reseptiä, suurta rakennusprojektia tai matematiikan opiskelua. Tällöin on usein kannattavaa merkitä projektin vaiheet solmuiksi ja niiden välillä siirtyminen nuoliksi niin, että nuolet kulkevat projektin alkuvaiheista loppua kohti. Nuolien painolla voidaan kuvata tarvittua aikaa. Seuraava esimerkki havainnollistaa tätä.

Yllä kuvatussa tilanteessa koko projektin kesto saadaan etsimällä pisin polku projektin alusta loppuun. Tätä polkua kutsutaan *kriittiseksi poluksi*, koska jos polun vaiheissa tapahtuu viivästys, koko projekti viivästyy. Kriittiselle polulle kuulumattomat vaiheet voivat jonkin verran myöhästyä, eli niillä on *pelivaraa*.

Esimerkki 20. Jauhelihapizzan valmistuksessa vaaditaan taulukossa 4.7 esitetyt vaiheet.

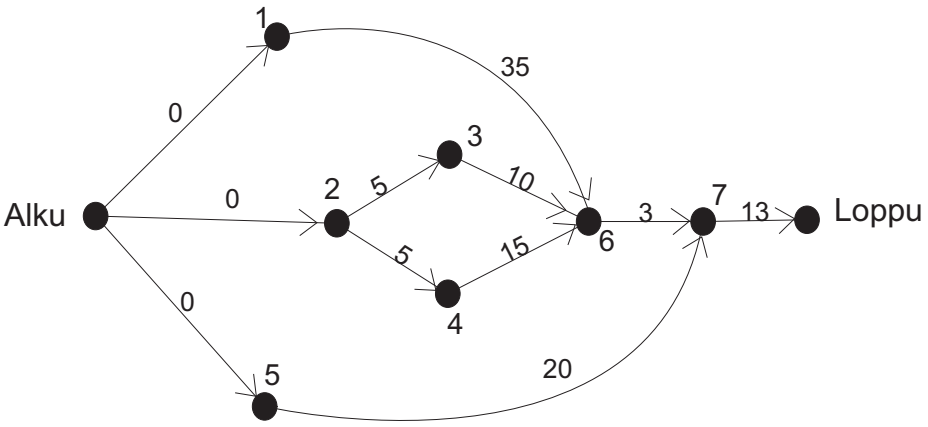
Numero	Työ	Kesto (min)	Edeltäjä
1	Pohjan valmistus	35	-
2	Sipulin ja yrttien hienontaminen	5	-
3	Tomaattikastikkeen valmistus	10	2
4	Jauhelihan valmistus	15	2
5	Uunin lämmittäminen	20	-
6	Täytteiden levittäminen	3	1, 3, 4
7	Paistaminen	13	5, 6

Taulukko 4.7: Jauhelihapizzan valmistusvaiheet esimerkissä 20.

Taulukosta 4.7 voidaan muodostaan verkko, jossa on alku- ja loppusolmun lisäksi 7 vaiheita kuvaavaa solmua. Jos solmulla ei ole edeltäjää, vedetään siihen nuoli alkusolmusta painolla 0. Jos taas solmu ei ole minkään solmun edeltäjä (eli sillä ei ole *seuraajaa*), vedetään siitä nuoli loppusolmuun painolla 0. Näin saadaan alla oleva verkko.

Kuinka kauan pizzan valmistus vähintään kestää, kun samaan aikaan käynnissä olevien vaiheiden määrä ei rajoita sitä? Tämä selviää etsimällä verkosta pisin tie alkusolmusta loppusolmuun. Tähän voisi käyttää Leipälän esittelemää algoritmia monisteen [7] sivulta 114 tai kaikki polut voisi luetteloida ja valita niistä pisimmän. Tällä kertaa ongelmamme on kuitenkin niin pieni, että pisin tie on helppoa nähdä ja päätellä. Selvästi tarvitsee tarkastella vain solmua 7 edeltävää osaa ja nuoli vaiheiden 1 ja 6 välillä pistää silmään muita paljon suurempana: se on jopa suurempi kuin kolmen muun peräkkäisen nuolen painot. Niinpä ei ole yllätys, että pisin tie on 1-6-7, jonka pituus on 51. Tämä on myös kriittinen polku.

Missä vaiheessa on viimeinen hetki alkaa lämmittää uunia, ilman että se hidastaa pizzan valmistumista? Pizzan valmistamiseen kuluu vähintään 51 minuuttia. Uunin pitää olla valmis ennen vaihetta 7, josta valmistumiseen kuluu vielä 13 minuuttia, joten uunin lämmitykseen on aikaa $51 - 13 = 38$



Kuva 4.5: Esimerkin 20 tilennetta kuvaava graafi.

minuuttia. Uunin lämmitykseen tarvitaan kuitenkin vain 20 min, joten sen aloittamista voidaan viivyttää $38 - 20 = 18$ minuuttia pizzan valmistamisen aloittamisesta. Tarkistuslaskulla $18 + 20 + 13 = 51$ voidaan todeta, että uunin lämmitys todella on aloitettava viimeistään 18 minuuttia pizzan valmistuksen alettua. Siispä voidaan sanoa, että uunin lämmityksellä on 18 min pelivaraa.

Tässä vaiheessa ei keskitytä tämän enempää pisimmän tien löytämiseen tai pelivaraan. Mainittakoon kuitenkin, että pelivaraa on useita tyyppejä, pisimmän tien löytämiseen tarvittava algoritmi ei välttämättä ole monimutkainen ja että pisimpään tiehen voivat liittyä myös eri työvaiheiden vaatimat resurssit. Lisätietoa saa muun muassa Leipälän monisteesta [7] sivuilta 113-118.

Virtausongelmat

Tarkkaan ottaen verkko-ongelmat ovat erikoistapauksia virtausongelmista. Tähänastisissa tapauksissa on vain oletettu, että kun löydetään paras tie, sitä voidaan käyttää miten paljon tahansa. Tämä ei kuitenkaan aina vastaa todellisuutta: tiellä ei voi kulkea rajattomasti autoja ilman että se ruuhkautuu. Virtausongelmissa nuolilla on painon sijaan *kustannus* ja *kapasiteetti*, jotka kuvaavat reitin kulkemisen hintaa tavarayksikköä kohden ja tavaran tai vastaavan suurinta sallittua määrää. Lisäksi virtausongelmissa esiintyy uudentyyppisiä solmuja: *lähteitä*, jotka tuottavat ”tyhjästä” verkossa virtaavaa tavaraa, ja *nieluja*, jotka hävittävät sitä tavaraa ”tyhjiin”. Näin verkko esittää tavaran kulkua, eikä sen valmistusta tai kulutusta.

Yleisin virtausongelma lienee sanahirviö *minimikustannusvirtausongelma*, jossa pyritään kuljettamaan lähteiden tuotanto nieluihin niin, että kapasiteetteja ei ylitetä ja kustannukset ovat mahdollisimman alhaiset. Tällai-

nen ongelma voisi olla tehdas, joka toimittaa tuotteita useisiin kauppoihin. Kaupat ovat eri etäisyyksillä, joten kuljetukset niihin ovat eri hintaisia, ja myös kysynnät vaihtelevat kauppojen kesken. Jos tämä kuulostaa liian yksinkertaiselta, lisätään yksi tai kaksi tehdasta ja muutama jakelukeskus, niin virtausongelman selvittäminen alkaa vaatia asiaan perehtymistä.

Maksimivirtausongelmassa pyritään selvittämään, kuinka suuri jatkuva virtaus on mahdollinen verkossa. Tämä edellyttää, että jokaisesta solmusta lähtee yhtä suuri virtaus kuin siihen tulee ja että nuolille on määrätty maksimikapasiteetit.Tällainen tieto on hyödyllistä, kun suunnitellaan esimerkiksi evakuointisuunnitelmia: Kuinka kauan kestää siirtää tietty ihmismassa ulos rakennuksesta tai kuinka monta ihmistä rakennukseen voidaan ottaa, jotta heidät voidaan saada ulos tietyssä ajassa.

On olemassa myös muita virtausongelmia. Lyhimmän tien ongelma voidaan ratkaista virtausongelmana, jos lähde tuottaa yhden tuotteen ja nielu tarvitsee sen. Lyhimmän tien löytämiseen on kuitenkin erityisesti siihen kehitettyjä tehokkaita menetelmiä. On myös *ajasta riippuvia virtauksia*, joita voidaan tutkia muodostamalla joka tilanteesta oma solmunsa. Tällaiset ongelmat tulevat helposti hyvin suuriksi. Viimeisenä mainittakoon *usean tuotteen virtausongelmat*, jolloin verkossa liikkuu useita tuotteita, jotka ovat mahdollisesti eri lähteistä. Esimerkkinä tällaisesta ongelmasta voisi käyttää usean alueen järjestelmää, jossa joiltakin alueilta mennään toisille töihin ja halutaan selvittää työmatkojen pituutta ja sitä mahdollisesti lyhentäviä keinoja. Virtausongelmia on käsitelty enemmän muun muassa Leipälän monisteessa [7] sivuilla 136-153

Luku 5

Optimointimenetelmistä

On olemassa suuri määrä erilaisia optimointimenetelmiä. Tämän on saattanut huomata jo johdannossa olleesta ohjelmistoihin keskittyvästä osiosta 1.2. Lieneekin aiheellista pohtia, miksi on näin monia menetelmiä? Yksinkertainen vastaus on: Koska ei ole keksitty täydellistä menetelmää. Kaikki menetelmät vaativat tehtävältä jotakin, vähintään esitysmuodon suhteen. Jotkin menetelmät vaativat toisia enemmän, mutta ovatkin juuri tietyn tyyppisten tehtävien ratkaisemisessa ylivoimaisia. Marko Mäkelä[1] on sanonut: "Jokaiselle optimointimenetelmälle voidaan konstruoida esimerkkitehtävä, jonka se ratkaisee paremmin kuin mikään muu menetelmä." Toisaalta Pursiheimo toteaa, että jokaiselle optimointialgoritmille voidaan kehittää tehtävä, jota se ei ratkaise ainakaan tehokkaasti [9].

On siis tärkeää valita tehtävän ratkaisemiseen oikeanlainen menetelmä, algoritmi ja ohjelmisto. Tämä valinta voi olla vaikea, mikäli eri menetelmien tai ohjelmien ominaisuudet eivät ole selkeästi tiedossa. Siksi valinnan tekemiseen on kehitetty apuvälineitä, kuten ohjelmien ja menetelmien ominaisuuksien taulukoita, joita löytyy muiden muassa sivulta 5 ja kirjasta [1]. Näiden lisäksi on olemassa myös niin kutsuttuja asiantuntijaohjelmistoja, jotka pystyvät tunnistamaan optimointitehtävän ja ehdottamaan tai valitsemaan parhaan ratkaisumenetelmän. Eräs tällainen on sisällytetty Excelin optimointialgoritmeihin. Osiossa 6.2.1 on myös kerrottu lyhyesti algoritmien paremmuuteen vaikuttavia asioita.

Tässä luvussa käydään pääpiirteittäin läpi joitakin yleisimpiä ja kiinnostavimpia optimointimenetelmiä. Tavoitteena on antaa tiivis kokonaiskuva kustakin menetelmästä, sen tehokkuudesta ja soveltuvuudesta. Yksinkertaisimpia menetelmiä käsitellään hieman syvällisemmin.

[1] Optimoinnin professori Turun yliopistosta.

5.1 Lähes intuitiivisista optimointimenetelmistä

Monet optimointimenetelmät tarvitsevat tuekseen runsasta matemaattista ymmärrystä, ja tämä tekee niistä tehokkaita. On kuitenkin olemassa menetelmiä, jotka vaativat vain vähän matemaattista perehtymistä ja mielikuvitusta. Tässä osiossa käsitellään menetelmiä, jotka ovat ehkäpä kaikkein yksinkertaisimpia ja vaativat vähiten esitietoja.

5.1.1 Raa'an voiman menetelmä ja täydellinen luettelointi

Kaikkein yksinkertaisin tapa löytää paras ratkaisu ongelmaan on tutkia kaikki mahdolliset ratkaisut ja sen jälkeen vain valita niistä paras. Tätä kutsutaan *täydelliseksi luetteloinniksi*. Tämän menetelmän suuri ongelma on siinä, että usein optimoitavalla funktiolla on äärettömästi arvoja, eikä niitä siis voi kaikkia luetteloida. Täydellisestä luetteloinnista tulee huomattavasti käyttökelpoisempi, jos otetaan käyttöön jokin tapa rajata luetteloitavien pisteiden joukkoa. Esimerkiksi osiossa 2.2.3 mainittujen kärkipisteiden salliminen luetteloitaviksi ja muiden jättäminen pois pienentää äärettömän joukon helposssa tapauksessa alle kymmeneen pisteeseen ja hankalassakin tapauksessa vain tuhansiin pisteisiin. Jos funktion derivaatta tai gradientti (selitetään sivulla 79) voidaan laskea, voidaan luetella niiden nollakohdat tai tehdä jopa merkkikaavio ja luetella vain järkevät pisteet, jolloin tehtävän koko pienenee selvästi. On kuitenkin mahdollista, että tehtävän pienentäminen on hankalaa tai että derivaatalla on ääretön määrä nollakohtia, kuten trigonometristen funktioiden tapauksessa. Yksinkertainen ei usein ole tehokas.

Hyvin lähellä täydellistä luettelointia on *raa'an voiman menetelmä* (engl. brute force), jossa tutkitaan suuri määrä ongelmaan liittyviä arvoja ja valitaan niistä jonkin kriteerin mukaan paras. Tällä tavalla ei saada välttämättä selville optimia, mutta sille voidaan saada hyvä likiarvo. Tämä kuitenkin vaatii, että tiedetään optimin sijoittuvan jollekin välille tai alueelle, ja se vaatii kohdefunktiolta hieman säännöllisyyttä. Jos nämä ehdot täyttyvät, voidaan raa'an voiman menetelmää käyttää apuna haarukoitaessa funktion arvoja, kuten seuraavaksi nähdään.

Esimerkiksi jos onnistutaan paikantamaan optimi välille $[0, 1000]$, voidaan laskea funktion arvot $0, 1, 2, 3, \ldots, 999, 1000$, jolloin optimin sijainti saadaan välille, jonka pituus on yksi. Nyt tälle välille voidaan laskea arvoja niin, että kahden arvon etäisyys on vaikkapa 0,000001 ja näistä arvoista voidaan valita paras. Sitä voi jo väittää hyväksi likiarvoksi, mutta jos tarkkuus ei riitä, voidaan prosessia toistaa kunnes käytettävän laskumenetelmän erottelukyky tulee ongelmaksi. Tällainen menetelmä ei tietenkään sovi kovin hyvin muille kuin tietokoneille, mutta koska tietokoneet ovat nykyään melko tehokkaita, menetelmä on tiettyyn rajaan asti käyttökelpoinen. Jos funktion arvot ovat helppoja laskea, miljoonan arvon laskemiseen ja läpikäymiseen voi kulua kotikäytössä olevalta tietokoneelta alle kaksi sekuntia. Tämä ai-

ka tietenkin riippuu käytettävästä koneesta, ohjelmistosta ja laskettavasta funktiosta. Jos funktio on tietokoneelle hankala, aika voi olla huomattavasti pidempi. Koska yllä on käsitelty raa'an voiman menetelmää yksiulotteisessa tapauksessa, käsitellään sitä seuraavaksi moniulotteisesti.

Esimerkki 21. Tilassa on kolme yhtä voimakasta äänilähdettä, joiden xy-koordinaatit ovat (0,0), (300,300) ja (400,200), yksikkönä cm. Mikrofoni halutaan asettaa niin, että sen äänilähteistä mitattujen etäisyyksien neliöiden summa on pienin mahdollinen. Eli mihin pisteeseen se tulisi sijoittaa, kun paremmuuden mittana käytetään etäisyyden neliötä? Etäisyys selviää Pythagoraan lauseella ja merkitsemällä mikrofonin sijaintia parilla (x, y), saadaan seuraava rajoittamaton optimointitehtävä:

$$\begin{aligned}\min \quad & \sqrt{x^2+y^2}^2 + \sqrt{(300-x)^2+(300-y)^2}^2 + \sqrt{(400-x)^2+(200-y)^2}^2 \\ & = x^2+y^2+(300-x)^2+(300-y)^2+(400-x)^2+(200-y)^2.\end{aligned}$$

Valitaan nyt kokeiltavat pisteet niin, että x:n arvot ovat välillä $[0, 400]$ ja kokeillaan niistä arvoja 50, 100, 150, 200, 250, 300 ja 350. Samalla tavoin y:n arvoiksi väliltä $[0, 300]$ valitaan 50, 100, 150, 200 ja 250. Näistä saadaan muodostettua taulukko 5.1 funktion arvoille.

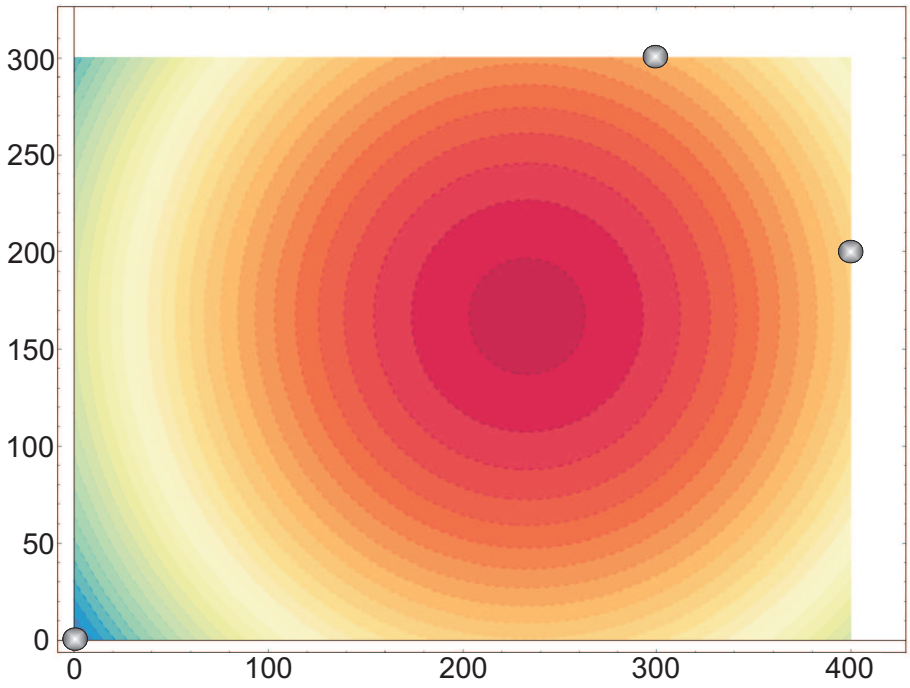
y:n	x:n arvot						
arvot	50	100	150	200	250	300	350
50	275000	227500	195000	177500	175000	187500	215000
100	247500	200000	167500	150000	147500	160000	187500
150	235000	187500	155000	137500	135000	147500	175000
200	237500	190000	157500	140000	137500	150000	17750
250	255000	207500	175000	157500	155000	167500	195000

Taulukko 5.1: Esimerkissä 21 käytetyn funktion arvoja.

Taulukon 5.1 arvot on laskettu OpenOfficella ja saman voi tehdä tietenkin myös Excelillä. Funktio esimerkiksi vasemman yläkulman arvon laskemiseen on `=B$2^2+$A3^2+(300-B$2)^2+(300-$A3)^2+(400-B$2)^2+(200-$A3)^2`. Taulukkolaskentaohjelmien kaavojen kopioinnin avulla suuri määrä soluja voidaan käsitellä nopeasti. Myöskin pienimmän arvon löytäminen helpottuu `=min(solu;solu)`-funktiolla ja kyseisen arvon voi hakea kaikkien arvojen joukosta Etsi–toiminnolla. Taulukon arvoista pienin on 135000, joka saadaan kun $x = 250$ ja $y = 150$. Jos tutkitaan väli $x \in (200, 300)$ ja $y \in (100, 200)$ 10 cm jaolla, saadaan tarkempi tulos 133400 arvoilla $(x, y) = (230, 170)$. Tutkimalla näiden ympäristö 1 cm jaolla, saadaan tulos 133334 arvoilla $(x, y) = (233, 167)$. Näin on saatu hyvä tulos yksinkertaisella menetelmällä, joka on tarpeeksi tehokas kotikäyttöön.

Jos ei haluta käyttää aikaa haun tarkentamiseen, voidaan suoraan luoda 1 cm jako alkaen kohdasta $(x, y) = (51, 51)$ kohtaan $(x, y) = (349, 249)$, jol-

loin saadaan 59501 arvoa. OpenOfficella ja melko tavallisella tietokoneella[2] kaikkien solujen arvojen laskeminen ja esittäminen kesti muutamia sekunteja. Tämä on osoitus siitä, että menetelmä on yllättävän käyttökelpoinen eikä vaadi erityisosaamista tai -ohjelmistoja.



Kuva 5.1: Kuva esimerkin 21 tilanteesta.

On myös syytä huomata, että täydellisen luetteloinnin ja raa'an voiman menetelmät löytävät globaalin minimin eivätkä jää jumiin lokaaliin minimiin, mikä on ongelmana useissa muissa menetelmissä.

5.1.2 Nelder–Mead

Seuraavaksi käsitellään Nelder–Meadin optimointimenetelmä[3] kaksiulotteisessa tapauksessa. Tällöin optimoitavan funktion arvoja voidaan kuvata tasona kolmiulotteisessa avaruudessa, kuten esimerkiksi paraboloidin pintana tai arkisempana esimerkkinä maan pinnanmuotoina luonnossa. Näin ollen funktion kulkua voidaan verrata kartan korkeuskäyriin ja optimointia syvimmän kuopan tai korkeimman mäen etsimiseen. Minimoitaessa eli kuoppaa etsittäessä on melko intuitiivista asettaa pallo tai vastaava pinnalle ja

[2] i3 3.07 GHz prosessori, 4 Gt DDR3 muistia, Windows 7 ja useita taustalla olevia prosesseja

[3] Tunnetaan myös Nelder–Meadin simpleksimenetelmänä sekä Nelderin ja Meadin polytooppihakuna.

katsoa, mihin suuntaan se alkaa vieriä. Pallo vierii kunnes se on pohjalla, joka vastaa likimain (lokaalia) minimiä. Tämän menetelmän kannalta on tärkeä, että pallon koko ja pinnanmuodot ovat järkevissä rajoissa: Pöydällä olevan lusikan alimman kohdan etsiminen jalkapallolla ei ole järkevää. [8]

Miten tätä sitten voi soveltaa matematiikassa? Nelder–Meadin optimointimenetelmän yksinkertaisin versio korvaa edellä mainitun pallon kolmiolla, joka kääntyilee (engl. flip) sivujensa varassa vierien kohti pohjaa. Tämä onnistuu laskemalla funktion arvo kolmion kärjissä ja vertaamalla arvoja toisiinsa. Yksinkertaisimmillaan kolmio on aina saman kokoinen, mutta menetelmän tehokkaammissa muodoissa kolmion muoto ja koko muuttuvat sen mukaan, millaista aluetta tutkitaan. Paneudummekin seuraavaksi itse menetelmän vaiheisiin ja laskuihin, joista runko on esitetty alla.

1. Valitaan aloituskolmio ABC ja lasketaan minimoitavan funktion $f(x, y)$ arvot sen kärkipisteissä.

2. Tutkitaan minkä kärkipisteen arvo on huonoin.

3. Korvataan huonoin piste paremmalla.

4. Toistetaan edellisiä vaiheita kunnes ollaan tyytyväisiä tulokseen.

Ensimmäisessä vaiheessa saatu kolmio voi olla ulkopuolelta annettu, itse päätelty tai vain satunnaiset kolme pistettä. Optimoitava funktio $f(x, y)$ täytyy olla tiedossa, jotta optimoinnissa olisi mitään mieltä. Jos alkuperäinen tehtävä on maksimointitehtävä, se voidaan muuttaa minimointitehtäväksi osiossa 3.1.1 esitetyllä tavalla sivulta 31 alkaen. Näin ensimmäisen vaiheen mahdollisesti työläin vaihe on laskea funktion $f(x, y)$ arvot pisteissä $A = (x_1, y_1)$, $B = (x_2, y_2)$ ja $C = (x_3, y_3)$, missä alaindeksi osoittaa vain eri x ja y koordinaattien mahdollisesti eroavan toisistaan. Funktion muoto voi tehdä arvon laskemisesta työlästä, vaikka tämä onkin melko harvinainen ongelma. Kuitenkin esimerkiksi hyvin suuria arvoja saavan funktion tapauksessa voi esiintyä ongelmia laskimen tai tietokoneen muistin ja tarkkuuden kanssa. Samoja ongelmia voi esiintyä, jos funktion arvot ovat hyvin lähellä toisiaan.

Toisessa vaiheessa oleva kärkipisteiden vertailu on yksinkertaista: Mitä suurempi arvo, sitä huonompi piste. Merkintöjen helpottamiseksi sanotaan, että A on paras, C on huonoin ja B siltä väliltä.

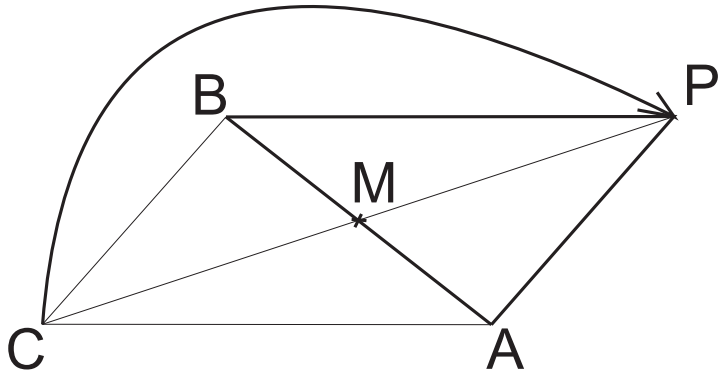
Kolmas vaihe on selvästi monimutkaisin edellä mainituista, koska siihen sisältyy päätöksenteko parhaasta uudesta pisteestä huonoimman tilalle. Tämän avuksi lasketaan janan AB keskipiste

$$M = \frac{A + B}{2} = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2} \right).$$

PEILAUS PISTEESEEN P (kuva 5.2): Koska funktion arvot pienenevät siirryytäessä huonoimmasta pisteestä C parhaaseen pisteeseen A ja myös

siirryttäessä pisteestä B pisteeseen A, vielä paremmat arvot saattavat olla pisteestä C nähden sivun AB toisella puolella. Näin ollen on järkevää kokeilla kolmion peilausta sivun AB suhteen. Näin saadaan kolmio ABP, missä P on uusi testattava piste. Matemaattisesti P saadaan piirtämällä jana CM, missä M on edellä mainittu AB:n keskipiste ja d on janan pituus. Jatketaan nyt janaa alkaen pisteestä M poispäin pisteestä C ja päädytään pisteeseen P. Jos pisteet esitetään vektoreina, pisteen P kaava on

$$P = M + (M - C) = 2M - C.$$



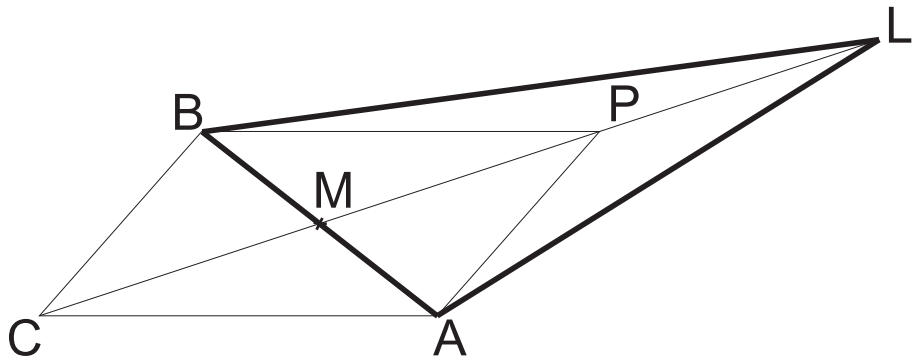
Kuva 5.2: Nelder-Meadin peilaus pisteeseen P.

LAAJENNUS PISTEESEEN L (kuva 5.3): Jos funktion arvo pisteessä P oli parempi kuin pisteessä C, ollaan liikuttu oikeaan suuntaan. On siis järkevää kokeilla liikkua vielä pidemmälle siihen suuntaan. Niinpä pidennämme janaa MP pisteeseen L, jolloin muodostuu laajennettu kolmio ABL. Piste L löytyy siirtymällä pisteestä P etäisyys d poispäin pisteestä M. Jos funktion arvo pisteessä L on pienempi kuin pisteessä P, ollaan löydetty parempi piste. Vektorikaava pisteelle L on

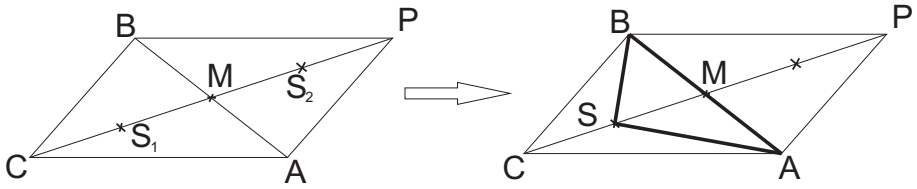
$$L = P + (P - M) = 2P - M.$$

SUPISTAMINEN PISTEESEEN S (kuva 5.4): Jos funktion arvot pisteissä P ja C olivat samat, pitää kokeilla jotakin muuta pistettä. Pisteen M pitäisi olla parempi, mutta se ei kelpaa, koska tarvitsemme kolmion. Kokeillaan siis pisteitä läheltä pistettä M. Valitaan pisteet S_1 ja S_2 siten, että ne ovat janojen CM ja MP keskipisteet. Lasketaan funktion arvot näissä pisteissä ja kutsutaan pienemmän arvon pistettä nimellä S ja saadaan uusi kolmio ABS.

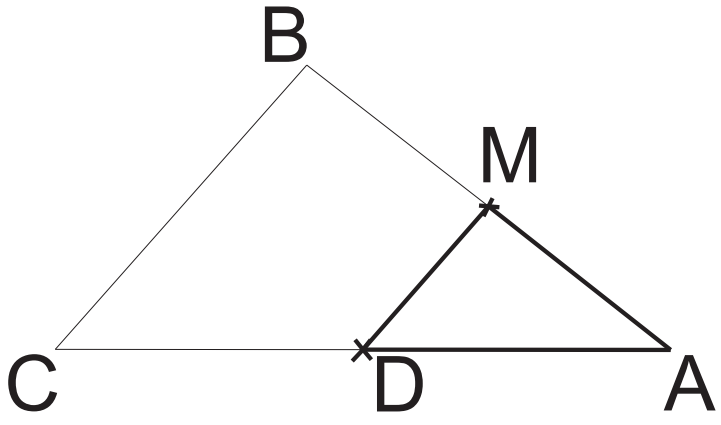
KUTISTAMINEN KOHTI PISTETTÄ A (kuva 5.5): Jos funktion arvo pisteessä S ei ole pienempi kuin pisteessä C, kolmiota täytyy kutistaa kohti pistettä A. Tämä tapahtuu korvaamalla piste B pisteellä M ja piste C pisteellä D, jossa D on janan AC keskipiste.



Kuva 5.3: Nelder-Meadin laajennus pisteeseen L.



Kuva 5.4: Nelder-Meadin supistus pisteeseen S, kun arvo pisteessä S_1 on pienempi kuin pisteessä S_2.



Kuva 5.5: Nelder-Meadin kutistus kohti pistettä A.

Näin ollaan saatu käsiteltyä kaikki tapaukset ja voidaan siirtyä neljänteen vaiheeseen, jossa lähinnä tarkistetaan, ollaanko tyytyväisiä tulokseen vai aloitetaanko uusi kierros vaiheesta 2. On useita tapauksia joissa voidaan todeta uudet kierrokset turhiksi, kuten esimerkiksi jos kaikissa kärkipisteissä saadaan sama funktion arvo tai jos nämä arvot ovat riittävän lähellä toisiaan.

Esimerkki 22. Lasketaan Nelder-Meadin algoritmilla kahdeksan kierrosta, kun lähdetään pisteistä (1,1), (2,0) ja (1,5;0) ja minimoitava kohdefunktio on $f(x, y) = (x - 2)^4 + (x - 2y)^2$. Saadaan taulukot 5.2 ja 5.3. Arvot on laskettu OpenOfficella ja ne ovat likiarvoja, jotta taulukon koko saataisiin pidettyä aisoissa.

i	Piste 1			Piste 2			Piste 3			M	
	X	Y	arvo	X	Y	arvo	X	Y	arvo	X	Y
1	1	1	2	2	0	4	1,5	0	2,31	1,25	0,5
2	1	1	2	1,63	0,25	1,29	1,5	0	2,31	1,31	0,63
3	1	1	2	1,63	0,25	1,29	1,41	0,31	0,73	1,52	0,28
4	1,26	0,64	0,30	1,63	0,25	1,29	1,41	0,31	0,73	1,33	0,48
5	1,26	0,64	0,30	1,04	0,70	0,99	1,41	0,31	0,73	1,33	0,48
6	1,26	0,64	0,30	1,19	0,59	0,44	1,41	0,31	0,73	1,22	0,62
7	1,26	0,64	0,30	1,19	0,59	0,44	1,31	0,46	0,37	1,29	0,56

Taulukko 5.2: Esimerkissä 22 saadut pisteet, jotka muodostavat joka vaiheessa tutkittavan kolmion.

i	P		CM		CP		Toiminto
	X	Y	X	Y	X	Y	
1	0,5	1	1,63	0,25	1,25	0	Supistaminen CM
2	1,13	1,25	1,41	0,31	1,31	0	Supistaminen CM
3	2,03	-0,44	1,26	0,64	1,52	0,28	Supistaminen CM
4	1,04	0,7					Peilaus
5	1,63	0,25	1,19	0,59	1,52	0,28	Supistaminen CM
6	1,04	0,92	1,31	0,46	1,11	0,75	Supistaminen CM
7	1,39	0,51	1,24	0,57	1,29	0,55	Peilaus

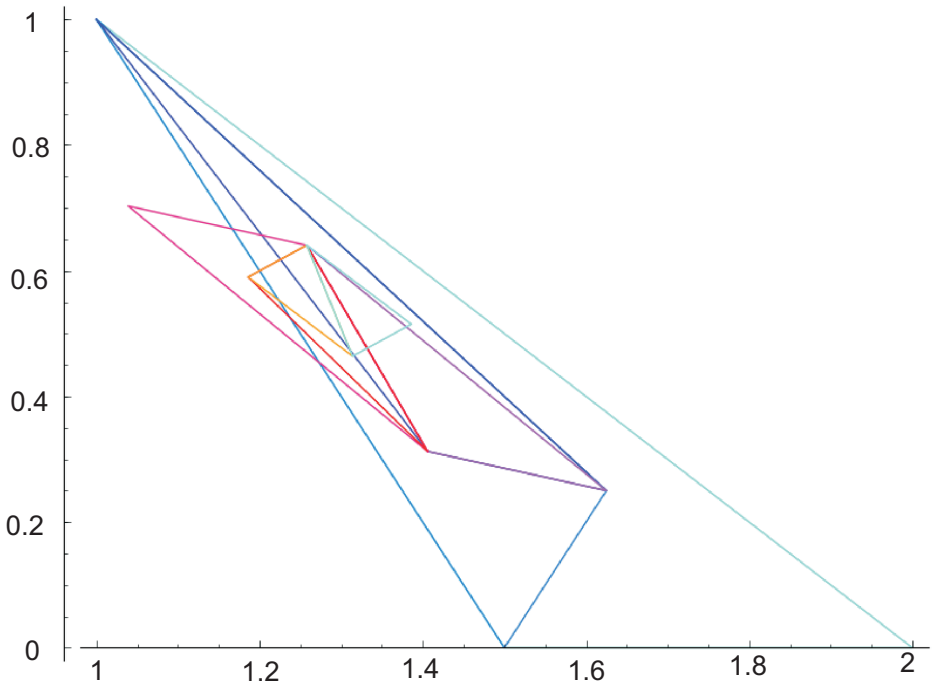
Taulukko 5.3: Esimerkissä 22 lasketut muut kuin kolmioon kuuluvat pisteet ja toiminto siirryttäessä seuraavaan vaiheeseen.

Seitsemännen kierroksen jälkeen saadut pisteet ja funktion arvot niissä on esitetty taulukossa 5.4. Menetelmän vaiheet näkyvät kuvassa 5.6.

Käytetyllä funktiolla on globaali minimi kohdassa $(2, 1)$, jossa arvo on 0. Funktio sisältää vinon laakson, jossa eteneminen on hankalaa monille algoritmeille ja myös Nelder-Meadin menetelmälle. Tämän esimerkin tavoitteena

Piste 1			Piste 2			Piste 3		
X	Y	arvo	X	Y	arvo	X	Y	arvo
1,258	0,641	0,304	1,386	0,515	0,269	1,314	0,464	0,371

Taulukko 5.4: Esimerkissä 22 lasketut tutkittavan kolmion pisteet seitsemännen kierroksen jälkeen.



Kuva 5.6: Kuva esimerkin 22 kulusta. Eri vaiheiden kolmiot ovat erivärisiä.

oli kuitenkin menetelmän toiminnan havainnollistaminen eikä minimin löytäminen. Esimerkiksi osoitteesta

`https://svn.broadinstitute.org/CellProfiler/trunk/oldCPA/jCPAnalyst/Statistics/NelderMead.java`

löytyvän algoritmin muunnelmalla[4] minimi 32 kierroksen jälkeen oli luokkaa 0,00000016.

Olemme käsitelleet Nelder-Meadin metodia vain kaksiulotteisessa tapauksessa, mutta sen käyttö ei ole rajoittunut niihin. Itse asiassa se sopii moniulotteisiin tapauksiin, kunhan optimoitava funktio täyttää tietyt ehdot [33]. Tällöin käytetään kolmion sijasta simpleksiä [35], mutta menetelmän periaatteet eivät eroa kaksiulotteisesta tapauksesta. Itse asiassa kolmio on simpleksi

[4]Muunnelma tarkoittaa tässä, että koodiin oli syötetty käsiteltävä funktio ja alkupisteet.

kaksiulotteiselle tapaukselle. Tästä syystä käsiteltyä menetelmää kutsutaan myös Nelder–Meadin simpleksimenetelmäksi.

5.1.3 Viivahausta

Viivahaku tarkoittaa sitä, että yritetään löytää minimi yksiulotteiselle funktiolle, kuten paraabelille. Tähän on olemassa useita erilaisia menetelmiä, joiden teho ja vaatimukset eroavat suuresti toisistaan. Se on usein osana muita optimointialgoritmeja ja niin myös nopeimman laskeutumisen menetelmää.

Vanha tuttu derivaatan nollakohtien tarkastelu on yksi vaihtoehto, mutta se vaatii kohdefunktion derivoituvuutta, eikä derivaatan laskeminenkaan saa olla kovin vaikeaa. Jos tunnetaan funktion kasvun ja vähenemisen säännöllisyyttä ja tiedetään minimin sijaitsevan jollakin välillä, voidaan tutkia välin pisteitä ja saada väli kutistumaan. Näin voidaan selvittää minimin summittainen sijainti.

Viivahakuun palataan tarkemmin osiossa 5.2. Toistaiseksi riittää kun ymmärrämme viivahaun idean ja mahdollisuuden.

Gradientista

Gradientti on derivaatan moniulotteinen vastine, ja sen laskeminen on hyvin samankaltaista derivaatan laskemiseen nähden. Tämä johtuu siitä, että gradientti koostuu funktion osittaisderivaatoista. Koska derivointi ei välttämättä ole tuoreimmassa muistissa, käytetään hetki asian selventämiseen.

Yksiulotteisessa, eli muotoa $f(x) = y$ olevassa, tapauksessa derivaatta kuvaa funktion arvon muutoksen nopeutta ja on myös funktio. Emme tässä käytä aikaa varsinaiseen derivointiin, koska esimerkiksi Maol-taulukoista löytyy useita derivoimiskaavoja, ja matemaattiset ohjelmistot sekä jotkin laskimet pystyvät derivointiin. Sen sijaan käsittelemme osittaisderivaatan käsitettä, joka voi olla hieman vähemmän tunnettu. Osittaisderivaatta kuvaa derivoitavan funktion muutosnopeutta sen muuttujan suhteen, jonka suhteen osittaisderivaatta on laskettu. Osittaisderivaattoja laskettaessa vain derivoitaessa mainittu muuttuja katsotaan muuttujaksi ja kaikki muut muuttujat vakioiksi. Luonnollisesti tämä vaatii vähintään kahden muuttujan funktiota, koska muuten osittaisderivaatta ei eroa normaalista derivaatasta. On myös selvää, että funktion on oltava jatkuva ja derivoituva kunkin muuttujan suhteen, jotta muuttujan osittaisderivaatta voitaisiin laskea. Osittaisderivaatat voidaan laskea kaikille muuttujille erikseen, kuten seuraavassa esimerkissä on tehty. Sen arvo voidaan myös määrittää kussakin pisteessä erikseen samoin kuin derivaatan. Funktion $f(x, y)$ osittaisderivaatalle muuttujan x suhteen pisteessä (x_1, y_1) voidaan käyttää seuraavia merkintöjä, joista viimeinen on yleisimmin käytetty

$$D_x f(x_1, y_1) = \frac{d}{dx} f(x_1, y_1) = \frac{\partial}{\partial x} f(x_1, y_1).$$

Esimerkki 23. Lasketaan osittaisderivaatat funktiolle $f(x,y) = x^2 + xy + 3y^2$.

Osittaisderivaatta x:n suhteen on $\frac{\partial}{\partial x} f(x,y) = \frac{\partial f(x,y)}{\partial x} = 2x + y$. Tämä seuraa siitä, että derivoitaessa x:n suhteen $D_x x^2 = 2x$, $D_x xy = 1 \cdot y$ ja $D_x 3y^2 = 0$ koska se on vakio.

Osittaisderivaatta y:n suhteen on $\frac{\partial}{\partial y} f(x,y) = x + 6y$. Tämä seuraa siitä, että derivoitaessa y:n suhteen $D_y x^2 = 0$ (vakio), $D_y xy = x \cdot 1$ ja $D_y 3y^2 = 3 \cdot 2y$.

Merkkiä, joka näyttää kuutosen ja d-kirjaimen risteytykseltä, kutsutaan nimellä *doo* ja se on yleinen merkintä osittaisderivaatalle. Nyt kun osittaisderivaattojen laskeminen ei enää ole täyttä hepreaa, voidaan siirtyä gradienttiin. Siirtymä ei ole suuri, koska gradientti koostuu funktion kaikista osittaisderivaatoista. Käytetään funktion gradientille pisteessä x_1, y_1 merkintää

$$\nabla f(x_1, y_1) = \left(\frac{\partial}{\partial x} f(x_1, y_1), \frac{\partial}{\partial y} f(x_1, y_1) \right).$$

kärjellään seisovaa kolmiota kaavan alussa kutsutaan *nablaksi* ja se on yleinen gradientin merkki. Huomaa, että suluissa oleva osa on vektori. Yleensä vektorit nimetään painetussa tekstissä lihavoinnilla ja käsin kirjoitetussa tekstissä vektorin yläpuolelle vedettynä viivana. Tässä teoksessa käytetään kuitenkin käsin kirjoitetun tekstin merkintää. On myös syytä huomata, että vektoreissa termien järjestyksellä on väliä, eli

$$\left(\frac{\partial}{\partial x} f(x_1, y_1), \frac{\partial}{\partial y} f(x_1, y_1) \right) \neq \left(\frac{\partial}{\partial y} f(x_1, y_1), \frac{\partial}{\partial x} f(x_1, y_1) \right).$$

Esimerkki 24. Esimerkissä 23 laskettiin osittaisderivaatat funktiolle $f(x,y) = x^2 + xy + 3y^2$ ja saatiin tulokseksi $\frac{\partial}{\partial x} f(x,y) = 2x + y$ ja $\frac{\partial}{\partial y} f(x,y) = x + 6y$. Yhdistämällä nämä saadaan funktion gradientti $\nabla f(x,y) = (2x + y, x + 6y)$. Nyt voimme esimerkin vuoksi laskea funktion gradientin pisteessä (1,2) ja saamme $\nabla f(1,2) = (2 \cdot 1 + 2, 1 + 6 \cdot 2) = (4, 13)$. Tämä on siis vektori $4\bar{i} + 13\bar{j}$, missä $\bar{i}$ ja $\bar{j}$ ovat x- ja y-akseleiden suuntaiset yksikkövektorit.

Gradientista ja differentioituvuudesta on luonnollisesti paljon enemmänkin tietoa saatavilla esimerkiksi Ylisen teoksessa [14], mutta tällä kertaa emme paljon enempää tarvitse. Vain sen maininnan, että tietyssä pisteessä funktio kasvaa jyrkimmin gradientin suuntaan ja vähenee jyrkimmin negatiivisen gradientin suuntaan.

5.1.4 Nopeimman laskeutumisen menetelmä eli NLM

Nopeimman laskeutumisen menetelmä (engl. steepest descent), eli lyhyesti NLM, noudattaa hyvin samantyylisiä periaatteita kuin Nelder–Meadin menetelmä kappaleessa 5.1.2. Nyt kaksiulotteisessa tapauksessa pallon tai kolmion vierittämisen sijaan pyrimme kuitenkin vain löytämään parhaan etenemissuunnan. Voisi ajatella, että pinnalle asetetaan vesivaaka, jolla voidaan

selvittää jyrkimmän nousun tai laskun suunta. Kuten yllä on mainittu, matemaattisesti jyrkimmän nousun suunta on sama kuin funktion gradienttivektorin suunta. Koska gradientti voi olla moniulotteinen vektori, voi sen hahmottaminen olla hyvin hankalaa. Tästä johtuen käsittelemme tässä vain kaksiulotteista tapausta, joka on yksinkertaisin gradienttivektorin tapaus. Matemaattisesti moniulotteiset tapaukset eivät useinkaan ole huomattavasti hankalampia.

NLM on yksi perustavimmista optimointimenetelmistä. Se on ollut olemassa pitkään ja on nykyään vanhentunut, kun tehokkaampia menetelmiä on kehitetty. Menetelmä on kuitenkin helppo ymmärtää, ja usein muiden menetelmien tehokkuutta verrataan siihen. Seuraavaksi on esitetty menetelmän vaiheet, joita käsitellään tarkemmin sen jälkeen.

1. Lopetusta varten valitaan sopivan pieni luku $\varepsilon > 0$. Tämä kuvaa etäisyyttä, jonka päähän minimistä halutaan päästä.

2. Valitaan aloituspiste $x_1 = x_k$. Asetetaan kierros– eli iteraatiolaskuriin arvo $k = 1$.

3. Tarkistetaan ehto $\|\nabla f(x_k)\| \leq \varepsilon$. Jos tämä pitää paikkansa, on päästy lähelle minimiä, voidaan lopettaa ja palauttaa käyttäjälle arvo λ_k. Muutoin otetaan suunta

$$d_k = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}.$$

   Nyt d_k on yksikkövektori (eli vektori, jonka pituus on yksi) suuntaan, johon funktio vähenee nopeiten pisteessä x_k

4. Määritellään funktio $\theta(\lambda) = f(x_k + \lambda d_k)$. Suoritetaan viivahaku tämän funktion minimin löytämiseksi kun $\lambda \geq 0$. Merkitään viivahaun tulosta λ_k:lla.

5. Siirrytään pisteeseen $x_{k+1} = x_k + \lambda_k d_k$ ja asetetaan arvo $k \leftarrow k + 1$. Siirrytään uudelleen vaiheeseen 3 ja suoritetaan kolmea viimeistä vaihetta kunnes vaiheen 3 ehdon tarkistus pitää paikkansa.

Vaiheista 1 ja 2 tuskin tarvitsee sanoa muuta kuin, että kreikkalaista kirjainta *epsilon* "ε" käytetään matematiikassa usein merkkinä pienelle positiiviselle luvulle. Sen arvo on usein jopa hyvin lähellä nollaa. NLM:n tapauksessa mitä pienempi ε on, sitä lähemmäksi minimiä halutaan ja sitä kauemmin menetelmän algoritmia käytetään. Muuttuja k on indeksi, joka kertoo menossa olevan suorituskierroksen numeron.

Vaiheessa 3 testataan onko lähtöpiste tarpeeksi lähellä minimiä. Näin algoritmia ei ole pakko suorittaa yhtään kierrosta, jos aloituspiste on valittu hyvin. Tämä testataan kullakin kierroksella sen kierroksen aloituspisteellä.

Testauksen mekanismina on tarkkaan ottaen gradienttivektorin pituus, eli jos funktion muutosnopeus mihinkään suuntaan ei ole yli tai yhtäsuuri kuin epsilon, voidaan todeta minimin olevan riittävän lähellä. Tätä arvoa kutsutaan myös vektorin *normiksi*, ja sen matemaattinen merkintä on ylläkin näkyvä $\|vektori\|$. Kaksiulotteisessa tapauksessa vektorin normin voi yleensä laskea Pythagoraan lauseella ja moniulotteisissa tapauksissa saman lauseen laajennuksella. On syytä huomata, että jos käsiteltävä piste x_k on lokaali minimi, niin $\nabla f(x_k) = 0$ ja samoin $\|\nabla f(x_k)\| = 0$.

Jos aloituspiste ei ole kyllin lähellä minimiä, valitaan siis suunta, josta minimiä lähdetään etsimään. Koska funktio kasvaa nopeiten gradienttinsa suuntaan, se selvästikin vähenee nopeiten negatiivisen gradientin suuntaan.

Vaiheessa 3 määritellään funktio, joka on yksiulotteinen poikkileikkaus alkuperäisestä funktiosta. Poikkileikkaus tehdään lähtien kierroksen alkuperäisestä pisteestä ja negatiivisen gradientin suuntaisesti. Tämän uuden funktion minimikohta negatiivisen gradientin suunnassa kertoo, miten pitkälle siihen suuntaan voidaan kulkea ilman että alkuperäisen funktion arvot alkavat kasvaa. Viivahaun palauttama arvo λ_k kertoo viivahaun tuloksen, eli minimikohdan sijainnin. Viivahaun menetelmää sinänsä ei ole määritetty, vaan se voidaan toteuttaa parhaaksi katsotulla tavalla.

Vaiheessa 4 otetaan viivahaun tulos uudeksi alkupisteeksi seuraavalle kierrokselle ja siirrytään siihen. Indeksi k päivitetään ja siirrytään uudelleen vaiheeseen kolme. Näin syntynyttä silmukkaa toistetaan kunnes vaiheen 3 lopetusehto tulee voimaan ja ollaan lähellä minimiä.

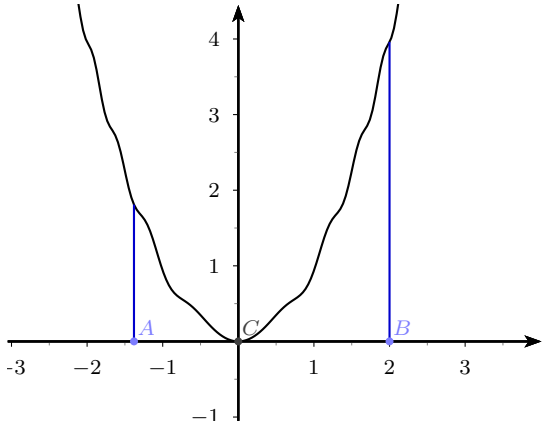
Lukijat ovat saattaneet algoritmin käsittelyn aikana tai jo aiemmin huomata, että NLM ei toimi kaikille funktioille, ja itse asiassa se vaatii kohdefunktiolta melko paljon. Ensinnäkin se vaatii funktion differentioituvuuden, jotta osittaisderivaatat voidaan laskea. Toiseksi se vaatii globaalisen optimin löytymiselle, että tutkittavan funktion arvojoukko on konveksi, eli että joukon kahden pisteen välinen jana kuuluu myös joukkoon. Tätä on havainnollistettu kuvassa 5.8. Jos tämä ehto ei ole voimassa, voidaan viivahaussa vahingossa ylittää sallittujen pisteiden alueen raja [6] [30]. Kolmanneksi globaalisen minimin löytämiseksi vaaditaan tutkittavan funktion olevan unimodaalinen minimin suhteen. Jos unimodaalisuus on uusi termi, seuraava määritelmä selvittää asiaa.

Määritelmä 1. Reaalilukuarvoja saava funktio on unimodaalinen välillä $[a, b]$, jos jollekin pisteelle $c \in (a, b)$ pätee, että $f(x)$ on aidosti vähenevä välillä $[a, c)$ ja aidosti kasvava välillä $(c, b]$. Tai vastaavasti aidosti kasvava välillä $[a, c)$ ja aidosti vähenevä välillä $(c, b]$. [2]

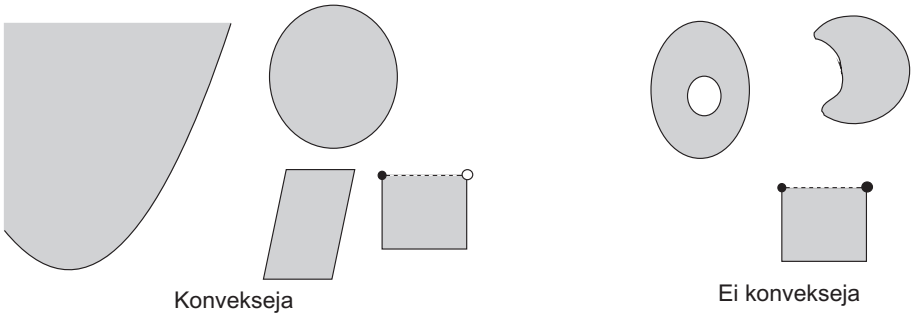
Toinen tapa sanoa sama asia on, että "$f(x)$ on unimodaalinen, jos jokaiselle parille x_1, x_2, jossa x_2 tuottaa kohdefunktiolle paremman arvon, on $\Delta x = x_2 - x_1$ parantava suunta pisteestä x_1." [6] Tässä x:t ovat siis funktion määrittelyjoukon pisteitä, usein lukuja tai (x, y)-pareja eli tason pisteitä. Hieman eksoottisemmissa funktioissa ne voivat kuitenkin olla esimerkiksi

usean luvun muodostamia vektoreita tai matriiseja. Tässä teoksessa näihin tapauksin ei kuitenkaan kosketa.

Unimodaalisuutta havainnollistaa kuva 5.7, mutta myös moniulotteiset funktiot voivat olla unimodaalisia. Sivulla 26 olevan esimerkin 7 funktio ei siis ole unimodaalinen minimin suhteen, mutta sivun 15 esimerkin 2 funktio on.



Kuva 5.7: Esimerkki minimoinnin suhteen unimodaalisesta funktiosta.



Kuva 5.8: Konvekseja ja ei konvekseja joukkoja. Algebrallisesti konveksiuden ehto joukolle $S \subset \mathbb{R}^n$ on $\lambda x + (1-\lambda)y \in S \ \forall x, y \in S \ \forall \lambda \in [0,1]$.

Kaikkien näiden vaatimusten jälkeen on masentavaa, mutta totta, että NLM on melko hidas menetelmä. Tämä johtuu siitä, että gradientin luonteen mukaan algoritmin peräkkäiset kulkusuunnat ovat kohtisuorassa toisiaan vastaan, jos on käytetty tarkkaa viivahakua. Näin ollen esimerkiksi banaanin tai spiraalin muotoiset ”laaksot”, joista minimiä etsitään, ovat algoritmille myrkkyä: niiden tutkiminen vaatii suuren määrän kohtisuoria suorituskierroksia ja kuluttaa näin ollen runsaasti aikaa. Onkin kehitetty useita gradienttiin perustuvia menetelmiä, joissa pyritään pois peräkkäisten suuntien kohtisuoruudesta ja näin saadaan nopeutettua kaarevien laaksojen tut-

kimista. NLM on kuitenkin menetelmien klassikko, ja muiden menetelmien tehoa verrataan usein siihen.

5.2 Yksiulotteisesta optimoinnista

Yksiulotteisen funktion optimoinnin idea on melko yksinkertainen, mutta menetelmät sen toteuttamiseen voivat olla hyvinkin hienostuneita. Tämän tutkielman luonteen mukaisesti käymme kuitenkin läpi lähinnä yksinkertaisimmat menetelmät. Lisää tietoa aiheesta saa esimerkiksi Pursiheimon monisteesta [9].

Miksi sitten yksiulotteisen funktion minimointiin eli viivahakuun on kehitetty useita menetelmiä, kun sen idea on harvinaisen helppo ymmärtää? Koska viivahakua tarvitaan monissa tilanteissa ja monenlaisissa tehtävissä. Moniulotteisten tehtävien minimoinnissa monet menetelmät etenevät askel askeleelta ottaen minimoitavasta funktiosta yksiulotteisen "poikkileikkauksen" lupaavaan suuntaan ja suorittavat sen jälkeen viivahaun tähän suuntaan. Koska moniulotteiset funktiot voivat olla hyvinkin eksoottisia, löytyy niiden pinnoilta lukemattomia erilaisia poikkileikkauksia, joille voi olla tarpeen suorittaa viivahaku. Koska ei ole olemassa yhtä parasta tapaa minimin löytämiseen, on kehitetty useita, joiden joukosta toivottavasti löytyy kuhunkin tilanteeseen kohtuullisen hyvä menetelmä.

5.2.1 Viivahaku derivaattojen avulla

Funktion minimin etsimisestä derivaatan avulla tulee luultavasti ainakin pitkän matematiikan lukeneille mieleen derivaatan nollakohtien tarkastelu sekä muistojen että idean tasolla. Se menetelmä onkin käyttökelpoinen, mikäli derivaatan nollakohta saadaan ratkaistua. Näin ei kuitenkaan aina ole ja seuraavaksi esiteltävä *bisektiomenetelmä* käyttääkin derivaatan arvoja vain muutamassa pisteessä. Tehokkaampia ja monimutkaisempia menetelmiä ovat esimerkiksi Newtonin menetelmä, Regula falsi ja Kuutiopolynomi.

Bisektiomenetelmä muistuttaa hyvin paljon kohdassa 5.2.2 esiteltävää puolitusmenetelmää. Bisektiomenetelmä siis tarvitsee derivoituvan kohdefunktion ja etsintävälin alku- ja loppupisteen. Menetelmän idea on tutkia annetun välin $[a, b]$ keskipisteessä c kohdefunktion derivaattaa. Jos tutkitussa pisteessä funktio on kasvava eli derivaatta on positiivinen, etsitään minimiä välin alkupäästä $[a, c]$. Jos derivaatta taas on negatiivinen, etsitään minimiä välin loppupäästä $[c, b]$. Tätä jatketaan kunnes minimin sijainti on saatu selville riittävällä tarkkuudella. Jos käy niin, että $f'(c) = 0$, voidaan todeta minimin löytyneen ja lopettaa.

Olisi myös suotavaa, että minimoitava funktio olisi etsintävälillä minimin suhteen unimodaalinen, koska menetelmässä itsessään ei ole estettä löytää lokaali minimi, vaikka tutkitulla välillä olisi myös globaali minimi. Tällaisissa tapauksissa on onnenkauppaa, sattuuko menetelmä löytämään lokaalin

vai globaalin minimin. On jopa mahdollista, että menetelmä löytää lokaalin maksimin minimin sijaan, jos maksimi sattuu olemaan jonkin välin keskipisteessä, jolloin derivaatta olisi nolla.

Esimerkki 25. Etsitään minimi funktiolle $f(x) = 0,6x^6 + 0,5x^5 - x^4 - x^3 + x^2 + 1,5x$ väliltä $[-1,5; 1,5]$ käyttäen puolitusmenetelmää derivaatan avulla. Minimin sijainti halutaan rajata välille, jonka pituus on pienempi kuin $\varepsilon = 0,025$.

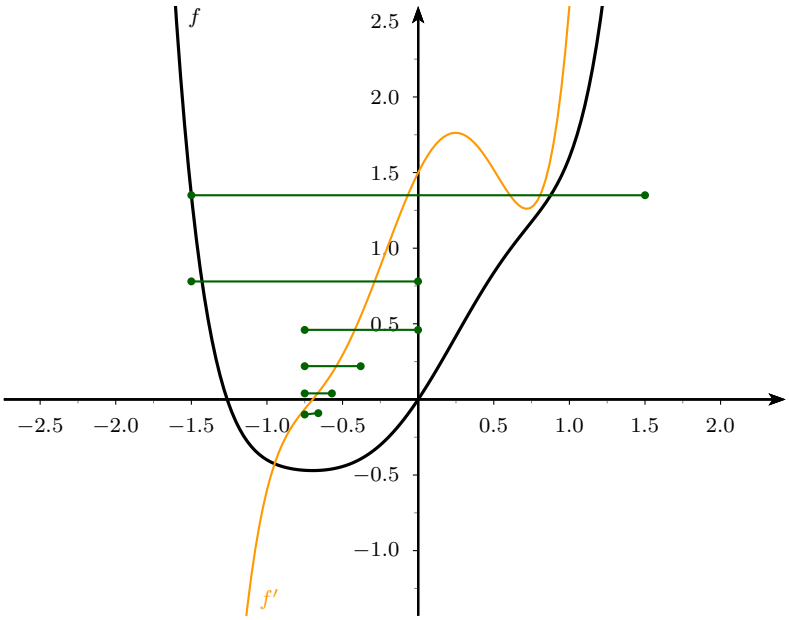
Ensimmäisenä kannattaa laskea derivaatta ja huomataan sen olevan $f'(x) = 3,6x^5+2,5x^4-4x^3-3x^2+2x+1,5$. Taulukossa 5.5 on esitetty puolitusmenetelmän toiminta. Sarakkeissa 'A' ja 'B' on välin päätepisteet, sarakkeessa 'C' keskipiste ja sarakkeessa 'ε' on välin pituus, joka pyritään saamaan alle epsilonin. Muissa sarakkeissa on funktion arvoja välin pisteissä ihan vain mielenkiinnon vuoksi kahden desimaalin tarkkuudella. Arvot ovat likiarvoja ja ne on laskettu OpenOfficella.

A	B	C	$f'(C)$	$\varepsilon < 0,025$	$f(A)$	$f(B)$	$f(C)$
-1,5	1,5	0	1,5	3	1,35	6,69	0
-1,5	0	-0,75	-0,06328	1,5	1,35	0	-0,47
-0,75	0	-0,375	0,56180	0,75	-0,47	0	-0,39
-0,75	-0,375	-0,5625	0,18525	0,375	-0,47	-0,39	-0,46
-0,75	-0,5625	-0,6563	0,05150	0,1875	-0,47	-0,46	-0,47
-0,75	-0,6563	-0,7031	-0,00659	0,0938	-0,47	-0,47	-0,47
-0,7031	-0,6563	-0,6797	0,02203	0,0469	-0,47	-0,47	-0,47
-0,7031	-0,6797	-0,6914	0,00765	0,0234	-0,47	-0,47	-0,47

Taulukko 5.5: Esimerkissä 25 toteutetun viivahaun kulku.

Nyt välin pituus on 0,0234375, eli minimikohdan sijainti on saatu riittävällä tarkkuudella ja se on välillä $[-0,703125; -0,6796875]$. Tätä havainnollistetaan kuvassa 5.9.

Jos puolitusmenetelmää ei käytettäisi, voitaisiin derivaatan nollakohta selvittää joskin hankalasti. Nollakohta on $x = -0,6976\ldots$.



Kuva 5.9: Kuva esimerkin 25 tilanteesta. Funktio $f(x)$ näkyy mustalla, $f'(x)$ keltaisella ja puolitusmenetelmän kuuden ensimmäisen vaiheen välit näkyvät vihreällä.

5.2.2 Viivahaku ilman derivaattoja

Joskus käy niin, ettei minimoitava funktio ole derivoituva kaikissa (tai missään) pisteissä, jolloin yllä mainitut menetelmät eivät toimi. Tällöin täytyy käyttää niin kutsuttuja *suoria menetelmiä*, jotka eivät ole riippuvaisia derivaatoista. Tällaisia menetelmiä ovat esimerkiksi puolitusmenetelmä, kultaisen leikkauksen menetelmä ja Fibonacci -menetelmä.

Puolitusmenetelmässä tutkitaan kahta pistettä hyvin läheltä tutkittavan välin keskipistettä ja lasketaan tutkittavan funktion arvo niissä. Näiden tietojen avulla päätellään optimin sijainti ja puolitetaan tutkittava väli joka kierroksella.[5]

Kultaisen leikkauksen menetelmässä tutkitaan kahta välin pistettä, jotka jakavat välin kultaisen leikkauksen suhteessa, eli $\frac{-1+\sqrt{5}}{2} \approx 0,618$. Menetelmä on tehokkaampi kuin puolitusmenetelmä, koska se vaatii vähemmän funktion arvon laskemisia. Tähän päästään siirtämällä edellisen kierroksen jakopiste seuraavalle kierrokselle. Menetelmää on käytetty Esimerkissä 26.

[5]Puolitusmenetelmää ja muitakin tässä kuvattuja menetelmiä voidaan soveltaa myös optimoinnin ulkopuolella esimerkiksi funktion nollakohdan hakuun, kuten Ruotsalainen [11] esittää.

Fibonacci -menetelmä voidaan osoittaa olevan paras suora menetelmä, ja se perustuu Fibonaccin lukuihin. Myös tässä menetelmässä edellisen kierroksen jakopiste siirtyy seuraavalle kierrokselle. Lisäksi väli pienenee Fibonaccin lukujen mukaisesti. Tarkempi perehtyminen menetelmään jätetään lukijan oman tutkimuksen varaan. Menetelmä on esitelty Pursiheimon monisteessa [9].

Kultaisen leikkauksen menetelmä etenee seuraavasti:

Askel 1 Huomioidaan aloitusväli $[a_1, b_1]$ ja lopetusvälin pituus L ja asetetaan ensimmäiset jakopisteet

$$\lambda_1 = a_1 + (1-\alpha)(b_1 - a_1), \mu_1 = a_1 + \alpha(b_1 - a_1),$$

missä $\alpha = \frac{-1+\sqrt{5}}{2}$. Lasketaan myös funktion arvot $f(\lambda_1)$ ja $f(\mu_1)$ sekä asetetaan $k = 1$.

Askel 2 Jos $b_k - a_k < L$, lopetetaan koska on saatu riittävä tarkkuus. Tällöin minimi sijaitsee välillä $[a_k, b_k]$. Muutoin

- Jos $f(\lambda_k) > f(\mu_k)$, siirrytään askeleeseen 3.
- Jos $f(\lambda_k) \leq f(\mu_k)$, siirrytään askeleeseen 4.

Askel 3 Asetetaan $a_{k+1} = \lambda_k$, $b_{k+1} = b_k$, $\lambda_{k+1} = \mu_k$ ja

$$\mu_{k+1} = a_{k+1} + \alpha(b_{k+1} - a_{k+1}).$$

Lasketaan arvo $f(\mu_{k+1})$ ja siirrytään askeleeseen 5.

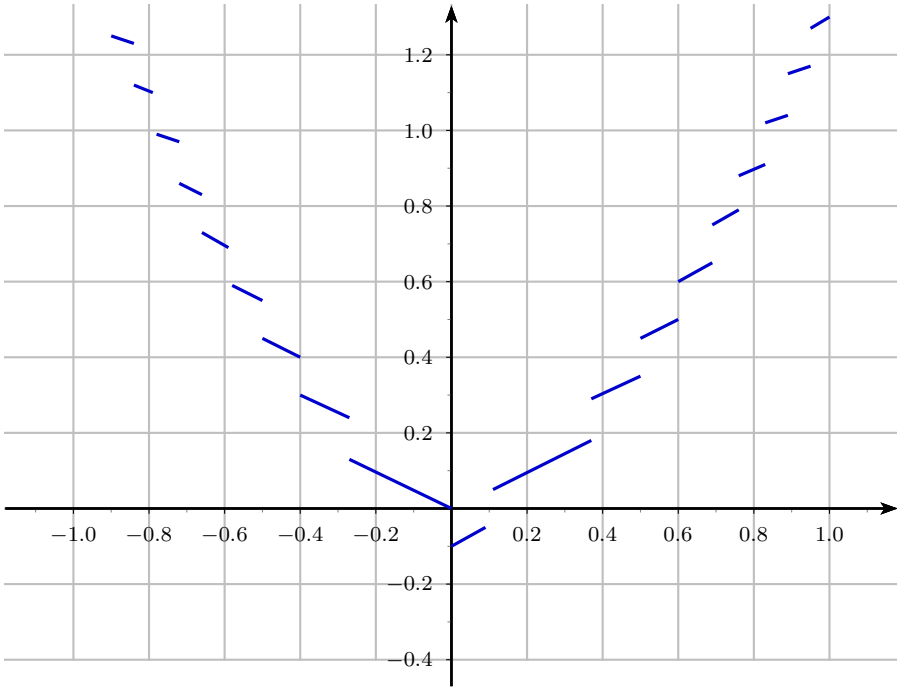
Askel 4 Asetetaan $a_{k+1} = a_k$, $b_{k+1} = \mu_k$, $\mu_{k+1} = \lambda_k$ ja

$$\lambda_{k+1} = a_{k+1} + (1-\alpha)(b_{k+1} - a_{k+1}).$$

Lasketaan arvo $f(\lambda_{k+1})$ ja siirrytään askeleeseen 5.

Askel 5 Asetetaan $k \leftarrow k+1$ ja siirrytään askeleeseen 2.

Esimerkki 26. Suoritetaan viivahaku kultaisen leikkauksen menetelmällä esimerkkifunktiolle $g(x) = 0,1\lfloor 10x^2 - x \rfloor + |0,5x|$, joka sisältää *lattiafunktion* (engl. floor function) ja itseisarvofunktion. Lattiafunktio saa sen kokonaislukuarvon, joka on pienempi tai yhtäsuuri kuin funktion argumenttina saama luku. Niinpä esimerkiksi $\lfloor 1,2 \rfloor = 1$ ja $\lfloor 15 \rfloor = 15$. Jos vielä on toiveita funktion derivoituvuudesta, ne haihtuvat katsomalla funktion kuvaajaa kuvassa 5.10. Koska funktio ei ole edes jatkuva, se ei ole myöskään derivoituva. Pyritään rajaamaan funktion minimin sijainti välille, jonka pituus on alle 0,1 aloittaen väliltä (-1;1,5).



Kuva 5.10: Kuva esimerkin 26 tilanteesta. Funktio $g(x)$ näkyy sinisellä.

Esimerkin tapauksessa siis $a_1 = -1$, $b_1 = 1,5$, $b_1 - a_1 = 2,5$, $\lambda_1 = -0,045$ ja $\mu_1 = 0,545$. Arvot esitetään kolmen desimaalin tarkkuudella, jos niitä on pyöristetty. Nyt

$$f(\lambda_1) = 0,022 \text{ ja } f(\mu_1) = 0,472.$$

Siispä $f(\lambda_1) < f(\mu_1)$ ja edetään askeleeseen 4. Tällöin

$$a_2 = a_1 = -1 \quad b_2 = \mu_1 = 0,545,$$

$$\mu_2 = \lambda_1 = -0,045$$

$$\lambda_2 = -1 + 0,382 \cdot 1,545 = -0,410$$

Nyt $f(\lambda_2) = 0,405 > f(\mu_2) = 0,023$, joten

$$a_3 = \lambda_2 = -0,410 \quad b_3 = b_2 = 0,545,$$

$$\lambda_3 = \mu_2 = -0,045$$

$$\mu_3 = -0,410 + 0,618 \cdot 0,955 = 0,180$$

Näin voidaan jatkaa kunnes seitsemännen kierroksen jälkeen välin pituus on $0,086\ldots$, eli on saavutettu riittävä tarkkuus ja tiedetään minimin sijaitsevan välillä $(-0,045; 0,041)$. Huomataan, että funktion arvoja tosiaan tarvitsee laskea vain yksi jokaisella kierroksella, ja jos se on hankalaa laskea, on hyvä, ettei sitä joudu tekemään usein.

5.3 Muista menetelmistä

Optimointimenetelmiä on suuri joukko, kuten on jo voitu arvata. Koska tämän teoksen tavoitteena on antaa perustiedot optimoinnista, kovin monien menetelmien käsittely ei ole tarkoituksenmukaista. Lisää tietoa aiheesta löytyy muun muassa Pursiheimon teoksesta Optimointialgoritmit [9], jossa käsitellään viitisenkymmentä erilaista optimointimenetelmää. Myös Leipälän monisteet [6] ja [7] käsittelevät useita erilaisia optimointimenetelmiä.

Alla käsitellään lyhyesti vielä muutama menetelmä, joista tietämisen voidaan ajatella olevan optimoinnin yleissivistystä.

5.3.1 Parantava haku sallittuun suuntaan

Monet optimointimenetelmät toimivat periaatteella, että suoritetaan parantava haku sallittuun suuntaan. Ideana on lähteä sallitusta pisteestä ja edetä pisteeseen, jossa saadaan parempi arvo kuin lähtöpisteessä ja että piste on sallittu eli kaikki rajoitukset ovat siinä voimassa. Tätä voidaan toistaa kunnes ollaan saavutettu (lokaali) optimi. Oikeastaan kaikki aiemmin esitetyt menetelmät noudattavat tätä periaatetta, ja sitä noudattavat myös kaksi seuraavaksi esiteltävää menetelmää.

Simplex menetelmä

Simplex menetelmän[6] kehitti vuonna 1947 George Dantzig, ja siihen on sen jälkeen laadittu useita parannuksia, joilla sen toimintaa on tehostettu ja ongelmakohtia vältetty. Menetelmän idea on lähteä liikkeelle jostakin kärkipisteestä, tutkia viereiset kärjet ja siirtyä niistä parhaaseen. Näin jatketaan kunnes on saavutettu optimi. Menetelmä toimii, koska lineaarisen optimointitehtävän ratkaisu saavutetaan aina kärkipisteessä, eli muita pisteitä ei edes tarvitse harkita. Tässä käytetään tehokkaasti hyväksi lineaarisuutta, joten epälineaaristen tehtävien ratkaisemiseen menetelmä ei sovellu.

Menetelmän mahdollisia ongelmia ovat sen vaatima kantaratkaisu ja degeneroituneet kärjet, joissa kohdefunktion arvo pysyy samana. Joillekin lineaarisen optimoinnin tehtäville on vaikeaa löytää kärkipistettä tai ainakaan kärkipistettä, joka olisi lähellä optimia. Koska simplex menetelmä liikkuu kärkipisteestä toiseen, tarvitaan lähtöpiste. Yleensä tällainen kuitenkin löydetään ennemmin tai myöhemmin.

Degeneroituneet kärjet ovat siis kärkipisteitä, joissa funktion arvo ei muutu. Jos tällaisia kärkiä on paljon vierekkäin, simplex menetelmä voi käydä niitä läpi pitkään ennen kuin löytää paremman arvon. On jopa mahdollista, että degeneroituneet kärjet edustavat optimia, ja menetelmä jää kiertämään niitä ikuisessa silmukassa. Tämä on kuitenkin onneksi harvinaista ja siihen

[6]Simplex menetelmää ei tule sekoittaa Nelder–Meadin simpleksimenetelmään. Niiden nimet ovat samankaltaiset, mutta se onkin ehkä menetelmien suurin yhteys.

on löydetty ratkaisuja. Simplex menetelmää on käyty perusteellisesti läpi Leipälän monisteessa [6].

Haku sisäpisteissä

Simplex menetelmä voi joutua degeneroituneiden kärkien vuoksi tutkimaan suuren määrän pisteitä ennen optimin löytymistä, mikä hidastaa laskuja. Eräs ratkaisu tähän on haku sisäpisteissä, johon perustuvia menetelmiä alettiin kehittää 80-luvun taitteessa. Menetelmien ideana on parantava haku sallittuun suuntaan, mutta ei kärkipisteeseen, jotta voidaan ohittaa degeneroituneita kärkiä. Tällainen menetelmä on vaikeampi kuin viereiseen kärkipisteeseen siirtyminen, mutta se voi silti olla nopeampi. Lisäongelman menetelmälle tuo se, että nopeimman kasvun tai vähenemisen suunta, eli positiivisen tai negatiivisen gradientin suunta, ei useinkaan ole sallittu. Jos menetelmien paremmuutta verrataan suuren tehtävän ratkaisemiseen kuuluvalla ajalla, haku sisäpisteissä on tehokkaampi kuin simplex menetelmä tahtävän ollessa kyllin iso, mutta käytännössä simplex menetelmä on usein nopea, koska tehtävät ovat helpohkoja.

5.3.2 Globaalista optimoinnista

Suurelle osalle optimointimenetelmistä lokaaliin minimiin juuttuminen on suuri ongelma. Lähestulkoon kaikki menetelmät löytävät oman tehtävätyyppinsä optimin, mutta vain harva menetelmä pystyy tarkistamaan, onko kyseessä lokaali vai globaali optimi. Tämä on erityisen kiusallinen ongelma, jos lokaalit optimit ympäröivät huomattavasti parempaa globaalia optimia. Siispä globaali optimointi on hankalaa ja tämä ongelma on ollut haaste optimoinnin kehittäjille. Luonnollisesti haasteeseen on vastattu. Alla on esitetty muutamia näistä vastauksista, joissa sallitaan kohdefunktion arvoa huonontavat pisteet tietyin ehdoin.

Tabuhausta

Tabuhaun ideana on kieltää palaamasta heti takaisin päin pitämällä tutkituista pisteistä ”tabulistaa”. Näin aikaisemmin tutkittuun pisteeseen ei saa palata niin kauan kuin se on on listalla, vaikka se olisi paras vaihtoehto. Näin tabuhakuun sovellettava menetelmä ei joudu helposti silmukkaan, ja se voidaan pakottaa ulos lokaalista optimista. Tietenkin tulee pitää kirjaa kullakin hetkellä parhaasta saavutetusta arvosta, jotta siihen voidaan palata optimoinnin päättyessä. Menetelmälle tulee myös asettaa jokin maksimimäärä tutkittaville pisteille, jotta se ei jatka optimin etsimistä ikuisesti.

Simuloidusta jäähdytyksestä

Simuloiduksi jäähdytykseksi kutsuttu menetelmä jäljittelee kuuman metallin jäähtymisen fysiikkaa. Myös tässä menetelmässä sallitaan huonontavat siirrot riippuen siirron huonontavuudesta ja optimointiin kuluneesta ajasta. Optimoinnin alussa huonontava siirto sallitaan suhteellisen todennäköisesti, mutta loppuvaiheessa huonontavia siirtoja ei sallita juuri lainkaan. Myös kuumassa metallissa rakenneosaset liikkuvat aluksi voimakkaasti, mutta metallin jäähtyessä liike hidastuu. Menetelmässä tulee asettaa jokin raja sille, miten pitkään optimointia jatketaan ja tulee myös määritellä, millä perusteella kokeiltavat pisteet etsitään. On myös pidettävä kirjaa parhaasta löytyneestä ratkaisusta.

Geneettisistä algoritmeista

Geneettiset algoritmit ja yleisemmin evoluutioalgoritmit pyrkivät löytämään ongelmaan optimaalisen ratkaisun jäljittelemällä luonnonvalintaa ja luonnossa esiintyviä evoluutioprosesseja[7]. Algoritmeissa luodaan pisteiden populaatio, jota muokataan sukupolvesta toiseen niin, että parhaat ratkaisut säilyvät sellaisinaan, ratkaisujen risteymistä syntyy uusia ratkaisuja ja pieni osa ratkaisuista saa satunnaisen mutaation. Populaation koko säilyy yleensä algoritmin jokaisella suorituskierroksella samana, mutta uuden populaation parhaat yksilöt ovat vähintään yhtä hyviä kuin edellisessä populaatiossa ja näin lähestytään optimia. Toisaalta populaatiossa halutaan aina pitää yllä riittävää divrsiteettiä, eli vaihtelevuutta ja monimuotoisuutta, koska kahden huonon ratkaisun risteytymä voi olla hyvä ratkaisu. Jos diversiteetti on pientä, algoritmi juuttuu helposti lokaaliin ääriarvoon ja globaali optimi jää löytymättä.

Geneettisten algoritmien hyviä puolia on niiden alhainen vaatimustaso: ne eivät vaadi kohdefunktion tai rajoitteiden jatkuvuutta. Siispä niitä voidaan käyttää, kun perinteiset menetelmät ovat suurissa vaikeuksissa. Lisäksi geneettiset algoritmit soveltuvat hyvin rinnakkaislaskentaan, mikä on iso hyöty aikakaudella, jolloin tietokoneissa on lähes poikkeuksetta moniydinprosessoreita. Geneettisten algoritmien huonoja puolia on se, että perinteiset menetelmät ovat usein geneettisiä algoritmeja tehokkaampia, jos kohdefunktio ja rajoitukset eivät ole erityisen hankalia. Rajoitteet voivat myös olla ongelmallisia geneettisille algoritmeille, kuten huomasimme esimerkin 15 lopussa. Tämä ongelma voidaan kiertää osittain käyttämällä sakkofunktioita, joista kerrotaan lyhyesti osiossa 6.2.3. Matemaattisesti geneettiset algoritmit ovat ärsyttäviä, koska niiden tehoa ei ole voitu matemaattisesti todistaa, mutta ne toimivat silti[8].

[7] Mikroevoluution jäljittely riittää, joten lukijoiden ei tarvitse hyväksyä evoluutioteoriaa tai makroevoluutiota hyväksyäkseen geneettiset algoritmit.

[8] Professori Marko Mäkelää mukaillen.

Luku 6

Muuta optimoinnista

Tähän lukuun on koottu asioita, jotka ovat mielenkiintoisia ja ne on hyvä tietää, mutta ne eivät liity kiinteästi mihinkään yksittäiseen optimointimenetelmään tai tehtävään. Asiat käsitellään melko lyhyesti.

6.1 Syvällisemmin herkkyysanalyysistä

Herkkyysanalyysillä pyritään selvittämään löydetyn optimin vakautta muuttuvassa tilanteessa. Jos esimerkiksi jotakin rajoitusta muutetaan vähemmän tiukaksi eli sitä *relaksoidaan*, optimin arvo joko paranee tai pysyy samana. Toisaalta jos jotakin rajoitetta *tiukennetaan*, optimi joko pysyy samana tai huononee. Tällaisia tilanteita voivat olla esimerkiksi tehtaan tarvitseman raaka-aineen lisääntynyt tarjonta tai tilausten yllättävä väheneminen. Relaksoimisesta voi olla hyötyä myös tiettyjen optimointimenetelmien käytössä, koska kaikki menetelmät eivät tule hyvin toimeen rajoitusten kanssa. Tästä kuitenkin enemmän sakkofunktioiden yhteydessä.

Usein on järkevää selvittää optimin muutosnopeus rajoituksen muutokseen verrattuna. Näin voidaan tehdä muodostamalla optimointitehtävän *duaalitehtävä* ja tutkimalla sitä. Duaalitehtävässä on aivan eri muuttujat kuin alkuperäisessä tehtävässä eli *primääritehtävässä* ja ne kuvaavatkin resurssiyksikön lisäyksen vaikutusta kohdefunktion arvoon. Tästä syystä niitä kutsutaan joskus *varjohinnoiksi* tai *marginaalihinnoiksi*. Positiivinen yllätys näihin liittyen on se, että ainakin jotkin versiot Excelin optimointityökalusta laskevat muuttujille myös varjohintatulkinnan. Duaalitehtävistä ja niiden muodostamisesta kerrotaan enemmän muun muassa Leipälän monisteessa [6].

6.2 Numeerisista menetelmistä

Joskus ollaan niin onnellisessa asemassa, että funktion optimin löytäminen matemaattisen tarkasti esimerkiksi derivaatan nollakohdan avulla on helppoa. Usein näin ei kuitenkaan ole, vaan joudutaan etsimään optimia numeeri-

silla menetelmillä, joita käytännössä kaikki aiemmin esitetyt menetelmätkin ovat. Numeeriset menetelmät ratkaisevat optimin sijainnin usein kuitenkin vain likimääräisesti ja menetelmien teho vaihtelee suuresti tehtävän tyypin mukaan. On siis tärkeää, että päätöksentekijä valitsee oikean menetelmän oikeaan tehtävään ja myös tarkastelee saamiaan tuloksia.

Seuraavaksi perehdytään hieman menetelmien arviointiperusteisiin. Tavoitteena on antaa lukijalle hieman keinoja arvioida ja testata erilaisten optimointimenetelmien paremmuutta, ja jotkin perusteet sopivat myös muihin ohjelmiin.

6.2.1 Algoritmien vertailuperusteista

Alla on esitetty vertailuperusteita eri algoritmeille. Niistä kaikki paitsi viimeinen sopivat myös muiden kuin optimointimenetelmien ja -algoritmien arviointiin. [9]

Yleisyys kuvaa sitä, miten monenlaisiin tilanteisiin ja tehtäviin algoritmi soveltuu. Toisaalta yleisyys kuvaa käänteisesti sitä, miten paljon vaatimuksia algoritmi asettaa tehtävilleen. Jos se asettaa runsaasti vaatimuksia ja soveltuu vain yhdentyyppisiin tehtäviin, se on käytännössä hyödytön muissa tapauksissa. Poikkeuksena edelliseen on tilanne, jossa kyseinen algoritmi on jonkin toisen ohjelman aliohjelma.

Luotettavuus kuvaa sitä, saavuttaako algoritmi riittävän tarkan tuloksen järkevässä ajassa. Tietenkin luotettavuuden tutkimus on rajoitettava algoritmille sopivien tehtävien ratkaisemiseen, koska kaikille algoritmeille voidaan keksiä tehtävä, jota ne eivät pysty ratkaisemaan tarkasti ja/tai tehokkaasti.

Herkkyys syötteiden suhteen: jotkin algoritmit ovat tarkkoja siitä, millaisia syötteitä ne saavat ja pienikin muutos parametreissa tai datassa voi muuttaa algoritmin toimintaa. Tämä ei yleensä ole toivottavaa. Myöskään parametrien skaalauksella ei pitäisi olla väliä, eli jos kaikki syötteenä saatavat muuttujat kerrotaan vaikka kymmenellä, algoritmin toiminnan ei pitäisi muuttua oleellisesti. Tällöin voidaan sanoa, että algoritmi on *skaalainvariantti*. Voitaisiin oikeastaan käyttää ohjelmointiin liittyviä termejä ja sanoa, että algoritmin tulee toteuttaa omat alku- ja loppuehtonsa.

Käytetty aika kuulostaa luultavasti itsestään selvältä arviointiperusteelta, ja sitä se osittain onkin. Käytetty aika voidaan kuitenkin jakaa alakäsitteisiin. *Valmisteluaika* kuvaa sitä, kuinka kauan algoritmin aliohjelmien valmisteluun ja mahdollisiin etukäteen tehtäviin laskuihin kuluu aikaa. Algoritmi voi esimerkiksi tarvita kohdefunktion gradientin syötteenä tai aliohjelmalta ja sen laskemiseen kuluva aika voidaan ottaa

mukaan valmisteluaikaan. *Laskuajalla* tarkoitetaan aikaa, jonka algoritmi itse käyttää. Näitä aikoja voi olla joskus vaikea vertailla, mutta usein niitä verrataan syötteen tai tehtävän laajuuteen.

Muistin käyttö on aiemmin ollut tärkeä kriteeri algoritmien paremmuudessa, mutta nykyisin se on vähemmänmerkittävä. Joka tapauksessa muistia tuhlaava algoritmi on tehoton ainakin tämän kriteerin mukaan. Hyvin suurissa ja vaikeissa tehtävissä muistin käyttö voi yhä muodostua ongelmaksi.

Konvergenssi tai konvergenssiaste kertoo siitä, miten nopeasti optimointialgoritmi lähestyy optimia. Luonnollisesti mitä nopeammin, sitä parempi. Tällä on väliä erityisesti silloin, jos laskenta joudutaan keskeyttämään tai yhteen laskentakierrokseen menee paljon aikaa.

6.2.2 Testifunktioista

Edellä käsiteltiin melko teoreettisia tapoja tutkia optimointimenetelmien paremmuutta. On kuitenkin myös paljon käytännönläheisempi tapa, jolla voidaan vertailla eri menetelmiä ja se on kokeilu. Alla on esitetty kolme mahdollista kaksiulotteista funktiota, joilla voidaan testata optimointimenetelmiä ja erityisesti niiden konvergenssiä. Muitakin mahdollisia funktioita on.

- Funktio $f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2$ sisältää vinon laakson, joka on hankala monille menetelmille. Globaali minimi on kohdassa (2,1) [9]
- Rosenbrockin (banaani)funktiota $g(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$ on usein käytetty optimointitesteissä. Se sisältää kaarevan laakson, jossa globaalin minimin löytäminen on useilla menetelmillä hankalaa. Globaali minimi on kohdassa (1,1) [34]
- Himmelblau:n funktiolla $h(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ on yksi lokaali maksimi ja neljä identtistä lokaalia minimiä. Maksimi on likimain kohdassa (-0,270845;-0,923039) ja minimit likimain kohdissa (3;2), (-2,805118;3,131312), (-3,779310;-3,283186) ja (3,584428;-1,848126). Tarkat arvot voidaan laskea analyyttisesti, mutta ne ovat hankalan muotoisia. [28]

6.2.3 Sakkofunktioista

Jotkin menetelmät soveltuvat vain rajoittamattomien optimointitehtävien ratkaisemiseen, joten ei ole yllätys, että rajoitetuista optimointitehtävistä pyritään joskus tekemään rajoittamattomia. Tähän voidaan minimoinnin tapauksessa käyttää sakkofunktioita, joiden arvot sallitulla alueella ovat lähellä

nollaa ja kasvavat voimakkaasti sallitun alueen ulkopuolella. Lisäämällä tällaisen funktion arvo kohdefunktioon saadaan minimin etsinnästä tehtyä järkevää vain, kun pysytään sallitulla alueella, eli menetelmät eivät usein edes yritä päästä alueelta pois. On olemassa myös estefunktioita, joiden arvot ovat pieniä sallitun alueen sisällä, mutta lähestyvät ääretöntä kun lähestytään sallitun alueen reunaa. Siispä sakkofunktiota käytettäessä voidaan ehkä mennä ulos sallitulta alueelta, jos sakkofunktio ei saa riittävän suuria arvoja. Estefunktiota käytettäessä taas voi olla hankalaa päästä alueen rajalle asti, jos estefunktio lähestyy ääretöntä liian aikaisin tai liian loivasti. Oikein käytettyinä molemmat menetelmät ovat kuitenkin toimivia.

6.3 Loppusanat

Kuten lukija on varmasti huomannut, optimointi on hyvin laaja matematiikan osa–alue. Tutkielman esipuheessa mainittiin kyseessä olevan vain pintaraapaisu ja ilmaus on osuva, koska asioita olisi voinut käsitellä paljon syvällisemminkin. Tätä varten on kuitenkin tarjolla optimoinnin kursseja yliopistoissa ja runsaasti muuta materiaalia optimoinnista.

Optimointi on myös yhteydessä moniin muihin osa–alueisiin ja aloihin. Esimerkiksi funktioiden kulun tutkimus optimoinnissa on hyvin läheisesti yhteydessä matemaattiseen analyysiin, optimointialgoritmien vertailussa ja laatimisessa algoritminen matematiikka ja tietojenkäsittelytiede ovat tärkeässä asemassa ja verkko–optimointi liittyy graafiteoriaan. Vastaavia esimerkkejä voisi keksiä lukuisia.

Tässä vaiheessa lukijan lienee syytä miettiä seuraavaa askelta. Onko kiinnostusta johonkin optimoinnin tai matematiikan osa–alueeseen tarpeeksi, jotta sen opiskeleminen on mielekästä? Onko suurten kokonaisuuksien opiskelun sijaan mielekkäämpää hakea tietoa joistakin yksittäisistä kiinnostuksenkohteista? Vai ovatko lukijan tiedot ja taidot optimoinnissa jo riittävällä tasolla eikä lisäopintoihin ole tarvetta tai kiinnostusta? Näihin kysymyksiin voivat antaa uuden näkökulman seuraavat lainaukset.

"Live as if you were to die tomorrow. Learn as if you were to live forever."Mahatma Gandhi

"I do not feel obliged to believe that the same God who has endowed us with sense, reason, and intellect has intended us to forgo their use."Galileo Galilei

Kirjallisuutta

[1] Juha Haataja, Yrjö Leino, Jussi Rahola, Juho Ruokolainen ja Ville Savolainen, Numeeriset menetelmät käytännössä, 2002, ISBN: 952-9821-81-6, CSC, painettu Suomessa

[2] Jussi Hakanen, Epälineaarinen optimointi, luentokalvot, 2012, Jyväskylän yliopisto,
http://users.jyu.fi/ jhaka/opt/luento2.pdf

[3] Tero Harju, Lecture Notes on Graph Theory, 1994-2011, Turun yliopisto

[4] Paavo Jäppinen, Alpo Kupiainen ja Matti Räsänen, Calculus 2, 2002, ISBN: 951-1-17225-5, Otava, painettu Suomessa

[5] Matti Laaksonen, Talousmatematiikan perusteet,2011, Vaasan Yliopisto, Teknillinen tiedekunta, Matemaattisten tieteiden laitos, (s.181-192)

[6] Timo Leipälä, Matemaattinen Optimointi I, 2008, Turun yliopisto

[7] Timo Leipälä, Matemaattinen Optimointi II, 2008, Turun yliopisto

[8] John H. Mathews and Kurtis K. Fink, Numerical Methods Using Matlab, 4th Edition, 2004, ISBN: 0-13-065248-2, Prentice-Hall Inc.

[9] Ulla Pursiheimo, Optimointialgoritmit, 1998, Turun yliopisto

[10] Keijo Ruotsalainen, Este- ja sakkofunktiomenetelmät, luentokalvot, Oulun yliopisto,
http://s-mat-pcs.oulu.fi/∼keba/Optimointi/OP_penalty.pdf luettu 11.8.2013

[11] Lauri Ruotsalainen, SAGE-ohjelmisto lukion matematiikan opetuksessa, 2011, Turun yliopisto, Matematiikan laitos, Pro gradu -tutkielma

[12] Aki Taanila, Lineaarinen optimointi, 2011, Haaga-Helia ammattikorkeakoulu,
http://myy.haaga-helia.fi/ taaak/m/optim.pdf

[13] Kyösti Tarvainen, haastattelu ja optimoinnin luentomateriaalia, saatu suoraan tekijältä 2013

[14] Kari Ylinen, Usean muuttujan funktiot, 2004, Turun yliopisto

[15] https://epsstore.ti.com/OA_HTML/csksxvm.jsp?nSetId=100805 luettu 5.8.2013

[16] https://noppa.aalto.fi/noppa/kurssi/mat-2.2105/viikkoharjoitukset/Mat-2_2105_mallit_11.pdf luettu 11.8.2013

[17] https://svn.broadinstitute.org/CellProfiler/trunk/oldCPA/jCPAnalyst/Statistics/NelderMead.java luettu 5.8.2013

[18] http://www.brainyquote.com/

[19] http://www.fonecta.fi/kartat luettu 5.8.2013

[20] http://www.geogebra.org/en/upload/files/english/mojca/SolarSystem.html luettu 5.8.2013

[21] http://extensions.services.openoffice.org/en/project/NLPSolver luettu 5.8.2013

[22] http://www.ima.umn.edu/ arnold/disasters/ariane.html
ja
http://royal.pingdom.com/2009/03/19/10-historical-software-bugs-with-extreme-consequences/ luettu 5.8.2013

[23] http://www.neos-server.org/neos/ luettu 11.8.2013

[24] http://plato.asu.edu/sub/pns.html luettu 5.8.2013

[25] https://sites.google.com/site/laskenta/scilab luettu 5.8.2013

[26] http://www.tsp.gatech.edu/world/ luettu 5.8.2013

[27] http://en.wikipedia.org/wiki/Evolutionary_algorithm luettu 5.8.2013

[28] http://en.wikipedia.org/wiki/Himmelblau

[29] http://fi.wikipedia.org/wiki/Kauppamatkustajan_ongelma luettu 5.8.2013

[30] http://fi.wikipedia.org/wiki/Konveksi_joukko luettu 5.8.2013

[31] http://en.wikipedia.org/wiki/Linear_programming luettu 5.8.2013

[32] http://en.wikipedia.org/wiki/List_of_optimization_software luettu 5.8.2013

[33] http://en.wikipedia.org/wiki/Nelder-Mead_method luettu 5.8.2013

[34] http://en.wikipedia.org/wiki/Rosenbrock_function luettu 5.8.2013

[35] http://en.wikipedia.org/wiki/Simplex luettu 5.8.2013

[36] http://fi.wikipedia.org/wiki/Stokastinen luettu 5.8.2013

[37] http://en.wikipedia.org/wiki/Travelling_salesman_problem luettu 5.8.2013