Aalto-yliopisto
Perustieteiden korkeakoulu
Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Lineaaristen monitavoiteoptimointitehtävien ratkaiseminen Bensonin algoritmilla

kandidaatintyö
28.03.2013

Juho Andelmin

Aalto-yliopisto

AALTO-YLIOPISTO PERUSTIETEIDEN KORKEAKOULU PL 11000, 00076 Aalto http://www.aalto.fi	KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Juho Andelmin

Työn nimi: Lineaaristen monitavoiteoptimointitehtävien ratkaiseminen Bensonin algoritmilla

Tutkinto-ohjelma: Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Pääaine: Systeemitieteet	**Pääaineen koodi:** F3010

Vastuuopettaja: Prof. Raimo P. Hämäläinen

Työn ohjaaja: TkT Juuso Liesiö

Tiivistelmä:

Lineaarisessa monitavoiteoptimointitehtävässä (MOLP, Multiple Objective Linear Programming) on useita lineaarisia kohdefunktioita, joita optimoidaan lineaaristen yhtälö- ja epäyhtälörajoitteiden vallitessa. MOLP-tehtävällä ei tavallisesti ole yksikäsitteistä optimiratkaisua, joka optimoisi kaikki kohdefunktiot samanaikaisesti. Sen sijaan tehtävän ratkaisujoukko koostuu tehokkaista pisteistä, eli niistä käyvistä päätösavaruuden pisteistä (päätösmuuttujien arvoista), joista siirtyminen mihin tahansa toiseen käypään pisteeseen huonontaa vähintään yhden kohdefunktion arvoa. Tunnetuin algoritmi tehokkaiden pisteiden tunnistamiseen on päätösavaruudessa operoiva monitavoite-Simplex, jonka laskenta-aika kasvaa eksponentiaalisesti päätösmuuttujien lukumäärän suhteen.

Tässä työssä tarkastellaan Bensonin algoritmia, joka ratkaisee MOLP-tehtävät suoraan tavoiteavaruudessa päätösavaruuden sijaan. Algoritmin soveltuvuutta käytännön ongelmiin tutkitaan resurssien allokointitehtävän avulla. Resurssien allokointitehtävässä käytetään hyväksi tehokkuusanalyysiä (DEA, Data Envelopment Analysis), jonka pohjalta muodostetaan kaksi MOLP-tehtävää erilaisilla oletuksilla. Lisäksi Bensonin algoritmin tuloksia verrataan monitavoite-Simplexillä saatuihin tuloksiin. Tulosten perusteella Bensonin algoritmi ratkaisee molemmat resurssien allokointitehtävät alle kymmenessä sekunnissa, kun taas monitavoite-Simplexillä ensimmäisen tehtävän ratkaisemiseen kuluu useita päiviä ja toinen osoittautuu liian suureksi ratkaistavaksi järkevässä ajassa. Käytännön ongelmissa kohdefunktioita on usein huomattavasti vähemmän kuin päätösmuuttujia, jolloin myös tavoiteavaruuden dimensio on merkittävästi pienempi kuin päätösavaruuden dimensio. Näin ollen Bensonin algoritmi vaikuttaa erittäin lupaavalta vaihtoehdolta sovelluksissa vastaantulevien MOLP-tehtävien ratkaisemiseen.

Päivämäärä: 31.03.2013	**Kieli:** suomi	**Sivumäärä:** 25

Avainsanat: optimointi, monitavoiteoptimointi, lineaarinen, Benson, MOLP, tehokkuusanalyysi, DEA, resurssien allokointi

Sisältö

Lista käytetyistä merkeistä ja lyhennyksistä

Λ	Sallittujen painovektorien joukko
$\mathcal{P}$	Laajennettu kuvajoukko $\mathcal{P} = P(\mathcal{X}) + \mathbb{R}^p_{\geq}$
$\mathcal{X}$	Päätösavaruuden käypä joukko
$\mathcal{X}_E$	Päätösavaruuden käyvän joukon tehokkaat pisteet
$\mathbb{R}^p_{>}$	$\{y \in \mathbb{R}^p : y > 0\}$
$\mathbb{R}^p_{\geq}$	$\{y \in \mathbb{R}^p : y \geq 0\}$
bd$\mathcal{P}$	Joukon $\mathcal{P}$ reunapisteet
conv(S)	Joukon S konveksi peite
int$\mathcal{P}$	Joukon $\mathcal{P}$ sisäpisteet
$P \in \mathbb{R}^{p \times n}$	MOLP-tehtävän kohdefunktioiden kerroinmatriisi
$P(\mathcal{X})$	Tavoiteavaruuden käypä joukko
$P(\mathcal{X}_E)$	Tavoiteavaruuden käyvän joukon ei-dominoidut pisteet
$x \in \mathbb{R}^n$	Päätösmuuttujien vektori
y^I	MOLP-tehtävän ideaalipiste
DEA	Data Envelopment Analysis, tehokkuusanalyysi
LP	Linear Programming, lineaarinen optimointi
MOLP	Multiple Objective Linear Programming, lineaarinen monitavoiteoptimointi

1 Johdanto

Lineaarinen monitavoiteoptimointi (MOLP, Multiple Objective Linear Programming) on kahden tai useamman lineaarisen kohdefunktion samanaikaista optimointia lineaaristen yhtälö- ja epäyhtälörajoitteiden määräämässä *käyvässä* alueessa. Lineaariset rajoitteet muodostuvat MOLP-tehtävän päätösmuuttujille asetetuista ehdoista. MOLP-tehtävissä on tavallisesti erilaisia tai jopa ristiriitaisia tavoitteita; esimerkiksi syöpäkasvaimen sädehoidon lineaarisessa optimointimallissa pyritään maksimoimaan kasvaimeen kohdistettua säteilyannosta ja toisaalta minimoimaan terveeseen kudokseen ja kriittisiin elimiin osuvan säteilyn määrää (Ehrgott, 2005).

MOLP-tehtävälle ei tavallisesti löydy yksikäsitteistä optimiratkaisua, missä jokainen kohdefunktio saavuttaisi optimiarvonsa. Optimiratkaisun sijasta MOLP-tehtävän ratkaisujoukko koostuu tehokkaista eli *Pareto-optimaalisista* pisteistä. Nämä ovat päätösavaruuden käypiä pisteitä (eli päätösmuuttujien arvoja), joista siirtyminen mihin tahansa toiseen käypään pisteeseen huonontaa vähintään yhden kohdefunktion arvoa. MOLP-tehtävä saadaan ratkaistua laskemalla tehokkaat pisteet ja esittämällä ne päätöksentekijälle (DM, Decision Maker), joka valitsee lopullisen ratkaisun preferenssiensä perusteella (Benson, 1998).

MOLP-tehtävien tehokkaiden pisteiden laskemisessa käytetyt metodit perustuvat pääosin yhden kohdefunktion lineaaristen optimointitehtävien ratkaisemiseen kehitettyyn Simplex-algoritmiin, jonka laskenta-aika on pahimmassa tapauksessa eksponentiaalinen (Ehrgott and Wiecek, 2005). Lisäksi tehokkaiden pisteiden lukumäärä kasvaa eksponentiaalisesti tehtävän koon mukaan (Steuer, 1986), joten monitavoitteiset Simplex-algoritmit muuttuvat tehottomiksi isoissa tehtävissä. Tämä on motivoinut tutkijoita kehittämään vaihtoehtoisia lähestymistapoja suurten MOLP-tehtävien ratkaisemiseksi.

Benson (1998) esittää artikkelissaan algoritmin MOLP-tehtävien ratkaisemiseen tavoiteavaruudessa päätösavaruuden sijaan. Tavoiteavaruuden dimensio on käytännön sovelluksissa usein huomattavasti pienempi kuin päätösavaruuden dimensio; tällöin voidaan olettaa, että tehtävän ratkaiseminen tavoiteavaruudessa ei ole yhtä työlästä verrattuna päätösavaruudessa ratkaisemiseen. On myös todettu, että päätöksentekijä valitsee tehtävän ratkaisun usein mieluummin tavoitearvojen kuin niihin johtaneiden päätösten perusteella (Benson and Sayin, 1997).

Tässä työssä esitetään Bensonin algoritmin toimintaperiaate MOLP-tehtävien ratkaisemiseen tavoiteavaruudessa. Lisäksi tutkitaan Bensonin algoritmin tehokkuutta ja soveltuvuutta käytännön ongelmiin, ja verrataan tuloksia monitavoite-Simplexillä saatuihin tuloksiin.

Kappaleessa 2 tutustutaan lineaarisen monitavoiteoptimoinnin käsitteisiin ja MOLP-tehtävien ratkaisujen karakterisointiin. Kappaleessa 3 pohditaan päätösavaruudessa ratkaisemisen haasteita, sekä esitetään Bensonin algoritmi tehtävien ratkaisemiseen tavoiteavaruudessa. Kappaleessa 4 tutkitaan Bensonin algoritmin tehokkuutta ensin numeerisella esimerkillä ja tämän jälkeen käytännön sovelluksessa, sekä verrataan tuloksia monitavoite-Simplexillä saatuihin tuloksiin. Yhteenveto työstä on esitetty kappaleessa 5.

2 Lineaarinen monitavoiteoptimointi

2.1 Lineaarinen monitavoiteoptimointitehtävä

MOLP-tehtävä voidaan esittää optimointitehtävänä, missä tavoitteena on minimoida p lineaarista kohdefunktiota $P_i : \mathbb{R}^n \to \mathbb{R}, i = 1 \dots p$, lineaaristen rajoitusehtojen määrittelemässä alueessa $\mathcal{X} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$, missä $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ ja tavoitevektorit P_i muodostavat kohdefunktioiden kerroinmatriisin $P \in \mathbb{R}^{p \times n}$ rivit. Vektorin $x \in \mathbb{R}^n$ alkioita kutsutaan päätösmuuttujiksi. MOLP-tehtävä on tällöin muotoa

$$\operatorname*{v\text{-}min}_{x \in \mathcal{X}} P(\mathcal{X}), \quad \mathcal{X} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}, \tag{1}$$

missä $P(\mathcal{X}) = \{Px : x \in \mathcal{X}\}$ on tehtävän *tavoiteavaruuden* käypä joukko ja $\mathcal{X} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ muodostaa *päätösavaruuden* käyvän joukon; molemmat $\mathcal{X}$ ja $P(\mathcal{X})$ ovat konvekseja ja suljettuja (Ehrgott, 2005).

Tehtävällä (1) ei tavallisesti ole yksikäsitteistä optimiratkaisua, vaan sen sijasta tehtävän ratkaisujoukko koostuu päätösavaruuden käyvän alueen *tehokkaista* pisteistä $\mathcal{X}_E$, joita kutsutaan myös *Pareto-optimaalisiksi* pisteiksi.

Määritelmä 1. *Piste $\hat{x} \in \mathcal{X}$ on tehokas, jos ei ole olemassa $x \in \mathcal{X}$ siten, että $Px \leq P\hat{x}$ ja $Px \neq P\hat{x}$; tällöin merkitään $\hat{x} \in \mathcal{X}_E$.*

Tehokkaat pisteet ovat optimaalisia siinä mielessä, että siirtyminen tehokkaasta pisteestä mihin tahansa toiseen käypään pisteeseen huonontaa vähintään yhden kohdefunktion arvoa.

Vastaavasti määritellään tehtävän tavoiteavaruuden käyvän alueen tehokkaat pisteet, joita kutsutaan yleisemmin *ei-dominoiduiksi* pisteiksi.

Määritelmä 2. *Piste $\hat{y} \in P(\mathcal{X})$ on ei-dominoitu, jos ei ole olemassa $y \in P(\mathcal{X})$ siten, että $y \leq \hat{y}$ ja $y \neq \hat{y}$; tällöin merkitään $\hat{y} \in P(\mathcal{X}_E)$.*

Tehokkaiden pisteiden joukko $\mathcal{X}_E$ muodostaa tehtävän (1) mahdolliset ratkaisut päätösavaruudessa. Tavoitteena on tavallisesti ratkaista kaikki tehokkaat pisteet ja esittää ne päätöksentekijälle, joka valitsee lopullisen ratkaisun preferenssiensä perusteella. Ei-dominoidut pisteet saadaan tällöin laskettua suoraan tehokkaista pisteistä: $P(\mathcal{X}_E) = \{Px : x \in \mathcal{X}_E\}$. Vaihtoehtoisesti voidaan laskea ei-dominoitujen pisteiden joukko $P(\mathcal{X}_E)$ suoraan tavoiteavaruudessa ja esittää se päätöksentekijälle tehokkaan joukon $\mathcal{X}_E$ sijasta.

Lisäksi voidaan määritellä *heikosti tehokkaat* pisteet $\mathcal{X}_{WE}$ ja näitä vastaavat *heikosti ei-dominoidut* pisteet $P(\mathcal{X}_{WE})$.

Määritelmä 3. *Piste $\hat{x} \in \mathcal{X}$ on heikosti tehokas, jos ei ole olemassa $x \in \mathcal{X}$ siten, että $Px < P\hat{x}$; tällöin merkitään $\hat{x} \in \mathcal{X}_{WE}$ ja vastaava tavoiteavaruuden piste $\hat{y} = P\hat{x} \in P(\mathcal{X}_{WE})$ on heikosti ei-dominoitu.*

Määritelmien perusteella kaikki tehokkaat pisteet ovat myös heikosti tehokkaita. Käytännössä ollaan kiinnostuneita tehokkaista pisteistä, mutta heikosti tehokkaiden pisteiden avulla voidaan karakterisoida monitavoiteoptimointitehtävien ratkaisuja. Erityisesti jos jokin ehto pätee heikosti tehokkaille pisteille $\mathcal{X}_{WE}$, pätee se myös tehokkaille pisteille $\mathcal{X}_E$, sillä $\mathcal{X}_E \subset \mathcal{X}_{WE}$. Vastaavanlainen yhteys on myös heikosti ei-dominoitujen pisteiden ja ei-dominoitujen pisteiden välillä.

2.2 MOLP-tehtävän skalarisointi

Monitavoiteoptimointitehtävä voidaan *skalarisoida* yhden kohdefunktion optimointitehtäväksi, jonka ratkaisulla on yhteys alkuperäiseen tehtävään. Tästä on usein hyötyä, sillä skalarisoitujen tehtävien avulla voidaan esimerkiksi karakterisoida tai laskea alkuperäisen tehtävän ratkaisuja.

MOLP-tehtävä (1) voidaan muuttaa yhden kohdefunktion LP-tehtäväksi (Linear Programming) painottamalla kohdefunktioita ei-negatiivisella painovektorilla $\lambda \in \mathbb{R}^p_{\geq}$, $\lambda \neq 0$ ja minimoimalla niiden summaa, jolloin saadaan skalarisoitu LP-tehtävä

$$\min_{x \in \mathcal{X}} \lambda^T Px, \quad \mathcal{X} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}. \tag{2}$$

Painovektori λ voidaan normalisoida määrittelemällä $\sum_{i=1}^{p} \lambda_i = 1$. Jos jokin painoista $\lambda_i = 0$, ei vastaavalla kohdefunktiolla P_i ole vaikutusta lopputulokseen; tämän takia onkin usein tarpeellista olettaa kaikkien painojen olevan positiivisia. Seuraava lause yhdistää MOLP-tehtävän (1) tehokkaan pisteen ja skalarisoidun LP-tehtävän (2) optimiratkaisun.

Lause 1. (Isermann, 1974) *Piste $\hat{x} \in \mathcal{X}$ on tehokas ratkaisu tehtävälle* (1) *jos ja vain jos on olemassa $\lambda \in \mathbb{R}^p_>$ siten, että*

$$\lambda^T P\hat{x} \leq \lambda^T Px, \quad \forall x \in \mathcal{X}. \tag{3}$$

Lauseen 1 perusteella jokaisella MOLP-tehtävän (1) tehokkaalla pisteellä $\hat{x} \in \mathcal{X}_E$ on olemassa positiivinen painovektori $\lambda \in \mathbb{R}^p_>$, joka vastaa skalarisoidun LP-tehtävän (2) optimiratkaisua. MOLP-tehtävän ratkaisuja voidaan täten laskea generoimalla positiivisia painovektoreita $\lambda \in \mathbb{R}^p_>$ ja ratkaisemalla niitä vastaavat tehtävän (2) optimiratkaisut. Muodostetaan tehtävän (2) duaali:

$$\max_{u \in \mathcal{U}} b^T u, \quad \mathcal{U} = \{u \in \mathbb{R}^m : A^T u \leq P^T \lambda\}. \tag{4}$$

Vahvan duaalisuuden perusteella tehtävien (2) ja (4) optimiratkaisut yhtyvät, mikäli kyseiset ratkaisut ovat olemassa.

Toinen tapa skalarisoida MOLP-tehtävä (1) on määritellä kohdefunktioille P_i, $i = 1, \ldots, p$ yhteinen referenssimuuttuja $z \in \mathbb{R}$ ja kiinnittää piste $y \in \mathbb{R}^p$ siten, että

$$\begin{aligned} P_1 x &\leq y_1 + z \\ P_2 x &\leq y_2 + z \\ &\cdots \\ P_p x &\leq y_p + z. \end{aligned} \tag{5}$$

Määrittelemällä vektori $e = (1, \ldots, 1)^T \in \mathbb{R}^p$ saadaan (5) kirjoitettua muotoon $Px \leq y + ez$. Minimoimalla referenssimuuttujan z arvoa saadaan jokaiselle $y \in \mathbb{R}^p$ skalarisoitu LP-tehtävä ($\mathrm{P}_2(y)$) (Löhne, 2011).

$$(\mathrm{P}_2(y)) \qquad \min_{(x,z) \in \mathcal{S}} z,$$
$$\mathcal{S} = \{(x, z) \in \mathbb{R}^n \times \mathbb{R} : Ax = b, Px - ez \leq y, x \geq 0\}.$$

Tehtävän duaaliksi saadaan ($\mathrm{D}_2(y)$).

$$(\mathrm{D}_2(y)) \qquad \max_{(u,\lambda)\in\mathcal{T}} b^T u - y^T\lambda,$$
$$\mathcal{T} = \{(u,\lambda) \in \mathbb{R}^m \times \mathbb{R}^p : A^T u \leq P^T\lambda, e^T\lambda = 1, \lambda \geq 0\}.$$

Seuraava lause muodostaa yhteyden tehtävän (1) heikosti tehokkaiden pisteiden $\mathcal{X}_{WE}$ ja skalarisoidun LP-tehtävän ($\mathrm{P}_2(y)$) optimiratkaisujen välille.

Lause 2. (Löhne, 2011) *Piste $\hat{x} \in \mathcal{X}$ on heikosti tehokas jos ja vain jos on olemassa $y \in \mathbb{R}^p$ siten, että $(\hat{x}, 0)$ on optimiratkaisu tehtävälle* ($\mathrm{P}_2(y)$).

Lause 3 yhdistää tehtävän (1) heikosti ei-dominoidut pisteet $P(\mathcal{X}_{WE})$ ja tehtävän ($\mathrm{P}_2(y)$) optimiratkaisut.

Lause 3. (Löhne, 2011) *Piste $\hat{y} \in P(\mathcal{X})$ on heikosti ei-dominoitu jos ja vain jos on olemassa $x \in \mathbb{R}^n$ siten, että $(x, 0)$ on optimiratkaisu tehtävälle* ($\mathrm{P}_2(\hat{y})$).

Vahvan duaalisuuden perusteella tehtävien ($\mathrm{P}_2(y)$) ja ($\mathrm{D}_2(y)$) optimiratkaisut yhtyvät, mikäli ne ovat olemassa: jos piste $\hat{y}$ on heikosti ei-dominoitu, saadaan tehtävän ($\mathrm{D}_2(\hat{y})$) optimiratkaisuksi $b^T u = \hat{y}^T\lambda$.

Lauseiden 2 ja 3 perusteella voidaan päätellä, että heikosti ei-dominoidulle pisteelle $\hat{y} \in P(\mathcal{X}_{WE})$ saadaan tehtävän ($\mathrm{P}_2(\hat{y})$) optimiratkaisuna heikosti tehokas piste $\hat{x} \in \mathcal{X}_{WE}$ siten, että $\hat{y} = P\hat{x}$. Luonnollisesti tämä pätee myös tehokkaille ja ei-dominoiduille pisteille, sillä $\mathcal{X}_E \subset \mathcal{X}_{WE}$ ja $P(\mathcal{X}_E) \subset P(\mathcal{X}_{WE})$. Tästä on hyötyä, jos ratkaistaan MOLP-tehtäviä suoraan tavoiteavaruudessa ja ollaan kiinnostuneita päätösmuuttujien arvoista; tehtävän ($\mathrm{P}_2(y)$) avulla saadaan laskettua ei-dominoituja pisteitä $P(\mathcal{X}_E)$ vastaavat tehokkaat pisteet $\mathcal{X}_E$.

3 Bensonin algoritmi MOLP-tehtävien ratkaisemiseen tavoiteavaruudessa

3.1 Päätösavaruudessa ratkaisemisen haasteet

Lineaarinen monitavoiteoptimointitehtävä (1) voidaan ratkaista päätösavaruudessa laskemalla tehokkaiden pisteiden joukko $\mathcal{X}_E$ ja esittämällä se päätöksentekijälle, joka valitsee parhaan vaihtoehdon preferenssiensä perusteella.

Käytännössä $\mathcal{X}_E$ kasvaa usein kuitenkin niin suureksi, että päätöksentekijän on vaikea valita mieleisensä ratkaisu tehokkaiden pisteiden joukosta. On myös havaittu, että päätöksentekijä valitsee tehtävän ratkaisun mieluummin tavoitearvojen kuin niihin johtaneiden päätösten perusteella (Benson and Sayin, 1997). Tarvittaessa ei-dominoidut tavoitearvot saadaan laskettua suoraan tehokkaista pisteistä: $P(\mathcal{X}_E) = \{P\hat{x} : \hat{x} \in \mathcal{X}_E\}$.

Päätösavaruudessa ratkaisemisen suurimmaksi haasteeksi osoittautuu laskennallinen tehokkuus. Tehokkaiden pisteiden lukumäärä kasvaa eksponentiaalisesti tehtävän koon mukaan (Benson, 1998), ja käytännön ongelmissa päätösavaruuden dimensio on usein merkittävästi suurempi kuin tavoiteavaruuden dimensio. Lisäksi on havaittu, että tavallisesti monta eri tehokasta pistettä kuvautuu samaksi ei-dominoiduksi pisteeksi aiheuttaen suuren määrän turhia laskutoimituksia (Dauer, 1987). Tällöin voidaan olettaa, että MOLP-tehtävän ratkaiseminen päätösavaruudessa on paljon työläämpää verrattuna tavoiteavaruuden ei-dominoitujen pisteiden $P(\mathcal{X}_E)$ suoraan laskemiseen.

Tästä huolimatta useimmat kirjallisuudessa esitetyt metodit MOLP-tehtävien ratkaisemiseen operoivat päätösavaruudessa laskien tehokkaiden pisteiden joukon $\mathcal{X}_E$. Nämä metodit perustuvat usein Simplex-algoritmin laajennuksiin tai sisäpistemenetelmiin (Steuer, 1986; Ehrgott, 2005), jotka muuttuvat varsin tehottomiksi tehtävän koon kasvaessa.

3.2 Bensonin algoritmi

Benson (1998) esittää artikkelissaan algoritmin (Benson's Outer Approximation Algorithm), joka ratkaisee MOLP-tehtävän (1) ei-dominoitujen pisteiden joukon $P(\mathcal{X}_E)$ suoraan tavoiteavaruudessa. Englanninkielisestä nimestään huolimatta Bensonin algoritmi ratkaisee kaikki ei-dominoidut pisteet tarkasti; algoritmista on tosin kehitetty myös variantti, joka ratkaisee ei-dominoitujen pisteiden arvot likimääräisesti mutta tehokkaammin (Shao and Ehrgott, 2008).

Tässä työssä tutkitaan Bensonin algoritmin variaatiota, johon on tehty parannuksia (Löhne, 2011). Toisin kuin alkuperäisessä algoritmissa, päätösavaruuden käyvän joukon $\mathcal{X}$ ei tarvitse olla rajoitettu; riittää, että $\mathcal{X}$ ei ole tyhjä joukko. Lisäksi paranneltu algoritmi tuottaa pelkästään (heikosti) ei-dominoituja pisteitä, joten jokaisen pisteen tarkistaminen algoritmin ajon jälkeen ei ole tarpeellista.

Merkitään kahden joukon S_1 ja S_2 summaa seuraavasti:

$$S_1 + S_2 = \{s^1 + s^2 : s^1 \in S_1, s^2 \in S_2\}. \tag{6}$$

Määritellään MOLP-tehtävän (1) tavoiteavaruuden käyvän joukon $P(\mathcal{X})$ *laajennettu kuvajoukko* $\mathcal{P} = P(\mathcal{X}) + \mathbb{R}^p_{\geq}$.

Lause 4. (Löhne, 2011) *Laajennettu kuvajoukko* $\mathcal{P} = P(\mathcal{X}) + \mathbb{R}^p_{\geq}$ *on polyhedri, jonka jokainen reunapiste on heikosti ei-dominoitu.*

Merkitään $\mathcal{P}_E = \{y \in \mathcal{P} : \textit{ei ole olemassa } y' \in \mathcal{P} \textit{ s.e. } y' \leq y\}$. $\mathcal{P}_E$ on määritelmän 2 mukaisesti joukon $\mathcal{P}$ ei-dominoitujen pisteiden joukko.

Lause 5. (Ehrgott, 2005) *Olkoon* $P(\mathcal{X}_E)$ *joukon* $P(\mathcal{X})$ *ei-dominoitujen pisteiden joukko. Tällöin* $P(\mathcal{X}_E) = \mathcal{P}_E$.

Lauseen 5 perusteella joukkojen $P(\mathcal{X})$ ja $\mathcal{P}$ ei-dominoidut pisteet yhtyvät, josta voidaan päätellä, että $\mathbb{R}^p_{\geq}$ lisääminen tehtävän (1) tavoiteavaruuden käypään joukkoon ei muuta ei-dominoitujen pisteiden joukkoa. Tämä on tärkeä ominaisuus, jota käytetään hyväksi Bensonin algoritmissa.

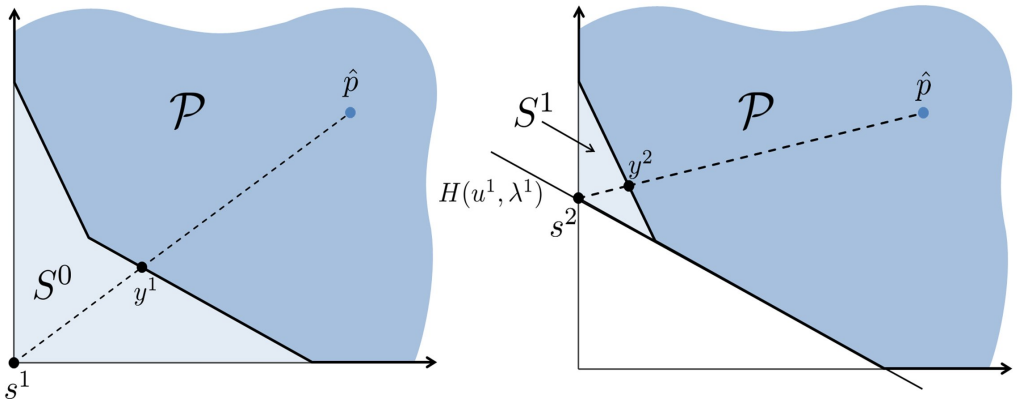
Oletetaan, että tehtävän (1) laajennettu kuvajoukko $\mathcal{P} = P(\mathcal{X}) + \mathbb{R}^p_{\geq}$ on alhaalta rajoitettu, eli on olemassa piste $\hat{y} \in \mathbb{R}^p$ siten, että $\hat{y} \leq y$ kaikille $y \in \mathcal{P}$. Määritellään tehtävän ideaalipiste y^I, joka antaa tiukan alarajan tavoiteavaruuden käyvälle joukolle.

Määritelmä 4. *Piste* $y^I = (y^I_1, \ldots, y^I_p)$, $y^I_k = \min\{y_k : y \in \mathcal{P}\}$ *on MOLP-tehtävän* (1) *ideaalipiste.*

Bensonin algoritmin idea on esitetty kuvassa 1. Algoritmi etsii aluksi laajennetun kuvajoukon sisäpisteen $\hat{p} \in \text{int}\mathcal{P}$ ja muodostaa polyhedraalin joukon $\mathcal{S}^0 = \{y^I\} + \mathbb{R}^p_{\geq}$, jolle pätee $\mathcal{P} \subseteq S^0$. Iteraation k aikana algoritmi valitsee jonkin kulmapisteen $s^k \in \mathcal{S}^{k-1}$, $s^k \notin \mathcal{P}$, ja etsii reunapisteen $y^k \in \text{bd}\mathcal{P}$, joka sijaitsee sisäpisteen $\hat{p}$ ja kulmapisteen s^k välisellä janalla. Tämän jälkeen algoritmi muodostaa kannattelevan hypertason joukolle $\mathcal{P}$ pisteeseen y^k skalarisoidun LP-tehtävän ($\text{D}_2(y^k)$) optimiratkaisun avulla.

Lause 6. (Löhne, 2011) *Olkoon piste* $\hat{y} \in \text{bd}\mathcal{P}$. *Tällöin tehtävän* ($\text{D}_2(\hat{y})$) *optimiratkaisu* $(\bar{u}, \bar{\lambda}) \in \mathcal{T}$ *on olemassa ja* $H(\bar{u}, \bar{\lambda}) = \{y \in \mathbb{R}^p : \bar{\lambda}^T y = b^T \bar{u}\}$, $\hat{y} \in H(\bar{u}, \bar{\lambda})$, *on kannatteleva hypertaso joukolle* $\mathcal{P}$.

Uusi polyhedri S^k saadaan polyhedrin S^{k-1} ja hypertason $H(u^k, \lambda^k)$ rajaaman, joukon $\mathcal{P}$ sisältävän, puoliavaruuden leikkauksena: $S^k = S^{k-1} \cap \{y \in \mathbb{R}^p : \lambda^T y \geq b^T u\}$. Algoritmi suorittaa äärellisen määrän iterointeja, kunnes jonkin iteraation k aikana ei löydy sellaista kulmapistettä $s^k \in S^{k-1}$, joka



Kuva 1: Bensonin algoritmin toimintaperiaate. Vasemmassa kuvassa polyhedrin $S^0 = \{y^I\} + \mathbb{R}^p_{\geq}$ kulmapiste $s^1 \notin \mathcal{P}$ vastaa ideaalipistettä y^I. Algoritmi muodostaa janan pisteiden s^1 ja $\hat{p}$ välille ja etsii reunapisteen $y^1 \in \text{bd}\mathcal{P}$. Oikeassa kuvassa on muodostettu hypertaso $H(u^1, \lambda^1)$ ratkaisemalla skalarisoitu LP-tehtävä $(\text{D}_2(y^1))$ ja tehty leikkaus $S^1 = S^0 \cap \{H(u^1, \lambda^1) \geq 0\}$. Lisäksi kuvassa on havainnollistettu seuraavaa iteraatiota etsimällä polyhedrin S^1 kulmapiste $s^2 \notin \mathcal{P}$ ja tätä vastaava reunapiste $y^2 \in \text{bd}\mathcal{P}$.

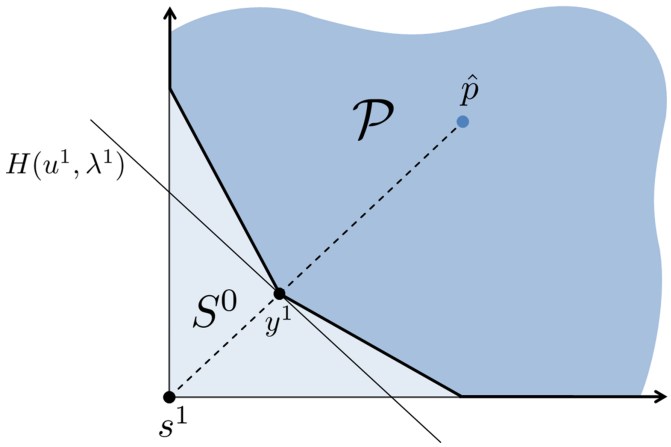
ei kuulu joukkoon $\mathcal{P}$. Tästä seuraa, että $S^{k-1} = \mathcal{P}$, ja algoritmi palauttaa joukon $\mathcal{P}$ kulmapisteet, jotka ovat seuraavan lauseen mukaan ei-dominoituja.

Lause 7. (Löhne, 2011) *Laajennetun kuvajoukon $\mathcal{P} = P(\mathcal{X}) + \mathbb{R}^p_{\geq}$ jokainen kulmapiste on ei-dominoitu.*

Lause 8. (Löhne, 2011) *Olkoon piste $y \in \mathbb{R}^p$ ja $S^{k-1} \subseteq \mathbb{R}^p$ polyhedraali konveksi joukko siten, että $\mathcal{P} \subseteq S^{k-1} \subseteq \{y\} + \mathbb{R}^p_{\geq} = S^0$. Olkoon $\mathcal{E}^{k-1}$ joukon S^{k-1} kulmapisteiden joukko; tällöin $S^{k-1} = \text{conv}(\mathcal{E}^{k-1} + \mathbb{R}^p_{\geq})$.*

Lauseen 8 mukaan algoritmissa muodostetut kuvajoukon $\mathcal{P}$ approksimaatiot $S^0, \ldots, S^{k-1}$ saadaan konstruoitua suoraan niiden kulmapisteiden joukkojen $\mathcal{E}^0, \ldots, \mathcal{E}^{k-1}$ ja joukon $\mathbb{R}^p_{\geq}$ summien konvekseina peitteinä, sillä approksimaatioiden ekstreemisuunnat ovat aina yksikkövektorit $e^k \in \mathbb{R}^p$. Koska algoritmin päätyttyä $S^{k-1} = \mathcal{P}$, saadaan $\mathcal{P}$ konstruoitua suoraan algoritmin palauttamien kulmapisteiden avulla.

Bensonin algoritmin englanninkieliseen nimeen (Benson's Outer Approximation Algorithm) viitaten voidaan ajatella, että alussa muodostettu polyhedri S^0 approksimoi kuvajoukkoa $\mathcal{P}$ ulkopuolelta, ja tavoitteena on tarkentaa tätä approksimaatiota iteraatioiden aikana muodostamalla uusia polyhedrejä $S^0 \supset S^1 \supset \cdots \supset S^{k-1}$, kunnes lopulta $S^{k-1} = \mathcal{P}$.



Kuva 2: Esimerkki tilanteesta, joka voi johtaa degeneroituneeseen yhtälöesitykseen joukon $\mathcal{P}$ reunasta. Piste y^1 on joukon $\mathcal{P}$ kulmapiste, ja ratkaisemalla skalarisoitu LP-tehtävä ($\mathrm{D}_2(y^1)$) voidaan päätyä kuvan tilanteeseen, missä ratkaisua vastaava hypertaso $H(u^1, \lambda^1)$ tukee joukkoa $\mathcal{P}$ ainoastaan pisteessä y^1.

Algoritmi tallentaa polyhedrin S^{k-1} esityksen iteraation k aikana kahdella eri tavalla: äärellisellä määrällä yhtälörajoituksia ja piste-esityksenä. Polyhedrin S^{k-1} piste-esitys on tarkka kuvaus joukon $\mathcal{P}$ kulmapisteistä algoritmin päätyttyä, mutta iteraatioiden aikana saattaa tulla vastaan redundantteja yhtälörajoituksia, jolloin yhtälöesitys joukon $\mathcal{P}$ reunasta on degeneroitunut. Esimerkki yhtälörajoitusten degeneroituvuudesta on esitetty kuvassa 2. Degeneroituvuus ei vaikuta tässä työssä tutkittavan algoritmin tuloksiin, mutta saattaa hidastaa algoritmin suoritusaikaa. Tätä voidaan tutkia jatkossa muodostamalla tehtäviä, joissa päädytään degeneroituneeseen yhtälöesitykseen joukon $\mathcal{P}$ reunasta tietyllä sisäpisteen $\hat{p}$ arvolla; muuttamalla pisteen $\hat{p}$ arvoa sopivasti ja suorittamalla algoritmi uudestaan päästään eroon degeneroituvuudesta, ja vertailemalla suoritusaikoja voidaan estimoida, miten yhtälöesityksen degeneroituvuus vaikuttaa algoritmin tehokkuuteen.

Bensonin algoritmi on esitetty kaaviossa 1. Alustuksen kohdassa 1 sisäpiste $\hat{p} \in \mathcal{P}$ saadaan laskettua etsimällä jokin päätösavaruuden käypä piste $x \in \mathcal{X}$ ja valitsemalla jokin $\alpha > 0$ siten, että $\hat{p} = Px + \alpha e$, missä $e = (1, \ldots, 1) \in \mathbb{R}^p$. Kohdassa 2 ideaalipiste y^I voidaan ratkaista myös duaalitehtävän (4) avulla; ideaalipisteen olemassaolon takaa oletus, että kuvajoukko $\mathcal{P}$ on alhaalta rajoitettu. Kohdassa 3 muodostetulle joukolle S^0 pätee ideaalipisteen määritelmän (4) perusteella $\mathcal{P} \subseteq S^0$.

Iteraatiovaiheen kohdassa 4 voidaan tarkistaa kuuluuko jokin kulmapiste $s^k \in S^0$ joukkoon $\mathcal{P}$ laskemalla skalarisoidun LP-tehtävän ($\mathrm{P}_2(s^k)$), tai sen

Kaavio 1 Bensonin Algoritmi (Löhne, 2011)

Syöte: Tehtävän (1) matriisit

Palauttaa: Kaikki tehtävän (1) ei-dominoidut kulmapisteet ja niiden väliset tahkot

Alustus:

1: Etsi jokin joukon $\mathcal{P} = P(X) + \mathbb{R}^p_{\geq}$ sisäpiste $\hat{p} \in \text{int}\mathcal{P}$.

2: Laske ideaalipiste y^I skalarisoidun LP-tehtävän (2) avulla sijoittamalla yksikkövektorit $e^k, k = 1, \ldots, p$ vuorotellen tehtävän painovektoriksi λ.

3: Muodosta polyhedraali joukko $S^0 = \{y^I\} + \mathbb{R}^p_{\geq}$. Tallenna joukko S^0 sekä piste-esityksenä joukon kulmapisteistä että yhtälöesityksenä joukon reunoista. Aseta $k = 1$.

Iteraatiovaihe:

4: Jos kaikki joukon S^{k-1} kulmapisteet kuuluvat joukkoon $\mathcal{P}$, siiry kohtaan 8. Muuten valitse jokin kulmapiste $s^k \in S^{k-1}$, joka ei kuulu joukkoon $\mathcal{P}$.

5: Laske $\alpha^k \in (0, 1)$ siten, että kulmapisteen s^k ja sisäpisteen $\hat{p}$ välisellä janalla oleva piste $y^k = \alpha^k s^k + (1 - \alpha^k)\hat{p}$ on joukon $\mathcal{P}$ reunapiste.

6: Laske skalarisoidun LP-tehtävän ($\text{D}_2(y^k)$) optimiratkaisu (u^k, λ^k) ja muodosta kannatteleva hypertaso $H(u^k, \lambda^k) = \{y \in \mathbb{R}^p : {\lambda^k}^T y = b^T u^k\}$ joukolle $\mathcal{P}$ reunapisteeseen y^k.

7: Muodosta S^k tekemällä leikkaus $S^k = S^{k-1} \cap \{H(u^k, \lambda^k) \geq 0\}$, aseta $k = k + 1$ ja palaa kohtaan 4.

Tulokset:

8: S^{k-1} sisältää tehtävän (1) ei-dominoidut kulmapisteet ja (mahdollisesti degeneroituneen) yhtälöesityksen joukon $\mathcal{P}$ reunasta.

duaalin ($D_2(s^k)$), optimiarvo μ^*; lauseiden 2 ja 3 perusteella $s^k \in \mathcal{P}$ jos ja vain jos $\mu^* = 0$. Kohdassa 5 määritelty α^k saadaan laskettua LP-tehtävästä

$$\alpha^k = \max\{\alpha : \alpha s^k + (1-\alpha)\hat{p} \geq Px, x \in \mathcal{X}\}. \tag{7}$$

Selvästi jokaiselle $x \in \mathcal{X}$, $\alpha = 0$ on käypä ratkaisu tehtävälle (7). Lisäksi $\mathcal{X}$ ei ole tyhjä joukko, joten tehtävälle on olemassa optimiratkaisu. Koska kulmapiste s^k ei kuulu joukkoon $\mathcal{P}$ ja piste $\hat{p}$ on joukon $\mathcal{P}$ sisäpiste, voidaan päätellä, että optimiratkaisua α^k vastaava piste $y^k = \alpha^k s^k + (1-\alpha^k)\hat{p}$ on joukon $\mathcal{P}$ reunapiste.

Iteraatiovaiheen kohdassa 6 piste y^k on joukon $\mathcal{P}$ reunapiste, joten lauseen 6 perusteella skalarisoidun LP-tehtävän ($D_2(y^k)$) optimiratkaisu on olemassa ja $H(u^k, \lambda^k) = \{y \in \mathbb{R}^p : {\lambda^k}^T y = b^T u^k\}$, $y^k \in H(u^k, \lambda^k)$ on kannatteleva hypertaso joukolle $\mathcal{P}$. Kohdassa 7 uusi polyhedri S^k saadaan muodostettua hypertason $H(u^k, \lambda^k)$ rajaaman, joukon $\mathcal{P}$ sisältävän, puoliavaruuden ja polyhedrin S^{k-1} leikkauksena. Tämän perusteella $\mathcal{P} \subseteq S^k \subseteq S^{k-1}$, joten jokainen iteraatio tarkentaa joukon $\mathcal{P}$ approksimaatiota S^k.

Algoritmi päättyy, kun kaikki joukon S^{k-1} kulmapisteet kuuluvat joukkoon $\mathcal{P}$ jolloin $S^{k-1} = \mathcal{P}$. Algoritmi palauttaa siis joukon $\mathcal{P}$ kulmapisteet, jotka ovat lauseen 7 mukaan ei-dominoituja; lisäksi lauseen 8 perusteella $\mathcal{P}$ saadaan konstruoitua algoritmin palauttamien kulmapisteiden avulla.

Lauseen 5 mukaan tehtävän (1) ei-dominoitujen pisteiden joukko on identtinen kuvajoukon $\mathcal{P}$ ei-dominoitujen pisteiden kanssa, joten algoritmi ratkaisee tehtävän (1) tavoiteavaruuden käyvän joukon ei-dominoidut ekstreemipisteet (kulmapisteet). Lisäksi algoritmi palauttaa joukon $\mathcal{P}$ reunan yhtälöesityksenä, josta saadaan ratkaistua tehtävän ei-dominoitujen kulmapisteiden väliset tahkot.

Kuten kappaleen 2.2 lopussa todettiin, jos ollaan kiinnostuneita ei-dominoituja pisteitä vastaavista päätösavaruuden tehokkaista pisteistä, saadaan ei-dominoitua pistettä $\hat{y} \in P(\mathcal{X}_E)$ vastaava tehokas piste $\hat{x} \in \mathcal{X}_E$ laskettua skalarisoidun LP-tehtävän ($P_2(\hat{y})$) optimiarvosta $(\hat{x}, 0)$: tällöin $P\hat{x} = \hat{y}$. Koska päätösavaruuden dimensio on käytännön ongelmissa usein suuri, on päätöksentekijän vaikea vertailla kaikkia ratkaisuja keskenään. Vertailua voidaan helpottaa esittämällä päätöksentekijälle tehokkaiden pisteiden minimi- ja maksimiarvot, joista saadaan käsitys kyseisten arvojen vaihteluväleistä.

4 Bensonin algoritmin testaaminen

Tässä kappaleessa tutkitaan Bensonin algoritmin tehokkuutta, ja vertaillaan saatuja tuloksia monitavoite-Simplexillä (Ehrgott, 2005) laskettuihin tuloksiin. Vertailussa käytetään Löhnen (2011; 2012) ohjelmoimaa Bensonin algoritmin toteutusta, ja tuloksia vertaillaan Nummenpalon (2012) ohjelmoiman monitavoite-Simplex algoritmin tuloksiin. Jotta laskenta-ajat olisivat vertailukelpoisia, ratkaistaan Bensonin algoritmissa ei-dominoitujen pisteiden lisäksi myös vastaavat tehokkaat pisteet, ja monitavoite-Simplexissä tehokkaita pisteitä vastaavat ei-dominoidut pisteet.

Bensonin algoritmin eri vaiheissa ratkaistaan yhden kohdefunktion LP-tehtäviä. Löhnen toteuttamassa Bensonin algoritmin variaatiossa on mukana kolme LP-ratkaisijaa välivaiheiden laskemiseksi, joista tässä käytetään avoimen lähdekoodin GLPK (GNU Linear Programming Kit) pakettia (Makhorin, 2006). Nummenpalon ohjelmoiman monitavoite-Simplex algoritmin toteutuksessa käytetään lp_solve pakettia (Berkelaar et al., 2004), jonka tehokkuuden on raportoitu olevan samaa luokkaa kuin GLPK:n LP-ratkaisijan (Meindl and Templ, 2012).

4.1 Numeerinen esimerkki

Tutkitaan aluksi algoritmin tehokkuutta yksinkertaisella esimerkillä, missä päätösavaruuden käypä joukko $\mathcal{X}$ muodostaa hyperkuution avaruudessa $\mathbb{R}^n$ eli $\mathcal{X} = [0,1]^n$, ja tavoitteina on minimoida muuttujia x_i ja $-x_i, i = 1, \ldots, n$ (Ehrgott, 2005). Tehtävä on tällöin muotoa

$$\begin{aligned} \min \quad & x_i, \quad i = 1, \ldots, n \\ \min \quad & -x_i, \quad i = 1, \ldots, n \\ \text{s.e.} \quad & x_i \leq 1, \quad i = 1, \ldots, n \\ & -x_i \leq 1, \quad i = 1, \ldots, n. \end{aligned} \tag{8}$$

Tehtävässä (8) on n kappaletta päätösmuuttujia ja $2n$ kappaletta sekä rajoitteita että kohdefunktioita. Tehtävän rajoitteiden muodostaman hyperkuution jokainen kulmapiste on tehokas, joten päätösavaruuden tehokkaiden pisteiden lukumäärä on 2^n. Lisäksi jokaista tehokasta pistettä vastaavat kohdefunktioiden arvot ovat yksilöllisiä, joten myös tavoiteavaruuden ei-dominoitujen pisteiden lukumäärä on 2^n.

Taulukko 1: Bensonin algoritmin ja monitavoite-Simplexin 10 ajon keskimääräiset suoritusajat sekunteina tehtävälle (8) parametrin n eri arvoilla.

n	8	9	10	11	12	13
Bensonin algoritmi	1.55	5.22	17.56	62.40	239.15	983.77
Monitavoite-Simplex	0.50	1.09	2.62	6.83	20.03	74.90

Käytännön ongelmissa kohdefunktioiden lukumäärä on tavallisesti merkittävästi pienempi kuin rajoitteiden ja päätösmuuttujien lukumäärä; usein jopa kertaluokkaa 10-100 tai enemmän (Benson, 1998). Yksi Bensonin algoritmin merkittävimmistä eduista monitavoite-Simplexiin nähden on juuri kohdefunktioiden pieni lukumäärä päätösmuuttujien ja rajoitteiden lukumäärään verrattuna, joten on mielenkiintoista vertailla algoritmien tehokkuuksia tehtävän (8) parametrin n eri arvoilla.

Taulukossa 1 on esitetty Bensonin algoritmin ja monitavoite-Simplexin suoritusajat sekunteina tehtävälle (8) parametrin n eri arvoilla. Suoritusajat on laskettu 10 ajon keskiarvona ja testaaminen on suoritettu 3.4 GHz neliydinprosessorilla ja 8 GB keskusmuistilla varustetulla tietokoneella.

Taulukosta 1 nähdään, että Bensonin algoritmi muuttuu nopeasti tehottomaksi tehtävälle (8): algoritmin suoritusaika yli kolminkertaistuu kasvattamalla parametrin n arvoa yhdellä. Tämä voidaan selittää katsomalla kaaviota 1, jossa algoritmin iteraatiovaiheissa 4, 5 ja 6 lasketaan yhden kohdefunktion LP-tehtävät, joiden lukumäärä on verrannollinen ei-dominoitujen kulmapisteiden lukumäärään; lisäksi jokaiselle ei-dominoidulle kulmapisteelle $\hat{y} \in P(\mathcal{X}_E)$ lasketaan vastaavat tehokkaat pisteet LP-tehtävän ($\mathrm{P}_2(\hat{y})$) avulla. Parametrin n kasvaessa yhdellä algoritmissa suoritettavien LP-tehtävien koko kasvaa ja määrä kaksinkertaistuu, mikä vaikuttaa algoritmin suoritusaikaan huomattavasti.

Taulukon 1 perusteella monitavoite-Simplex ratkaisee tehtävän (8) huomattavasti tehokkaammin kuin Bensonin algoritmi, mutta tästä ei voida vielä muodostaa johtopäätöksiä algoritmien tehokkuuksista käytännössä. Tavallisesti käytännön ongelmissa tavoiteavaruuden dimensio on huomattavasti pienempi kuin päätösavaruuden dimensio, ja lisäksi monta eri tehokasta pistettä vastaa usein yhtä ei-dominoitua pistettä (Dauer, 1987). Tehtävä (8) on siis todella epäedullinen Bensonin algoritmin kannalta, mutta se on epäedullinen myös monitavoite-Simplexin kannalta tehokkaiden pisteiden lukumäärän takia, joten tuloksista näkee hyvin, miten algoritmien tehokkuudet vaihtelevat huonolla syötteellä.

4.2 Sovellus: Resurssien tehokas allokointi

Tutkitaan Bensonin algoritmin tehokkuutta käytännön sovelluksessa ja verrataan tuloksia monitavoite-Simplexillä saatuihin tuloksiin. Esimerkkisovelluksena käytetään resurssien allokointitehtävää, missä tarkastellaan erään suomalaisen kauppaketjun 25 eri kaupan olemassa olevien resurssien ja lisäresurssien allokointia kauppojen välillä (Korhonen and Syrjänen, 2004). Resurssien allokointimallissa käytetään apuna tehokkuusanalyysiä (DEA, Data Envelopment Analysis; Charnes et al., 1978), minkä pohjalta saadaan rakennettua MOLP-tehtävä resurssien järkevään allokointiin *päätöksentekoyksiköiden*, tässä tapauksessa kauppaketjun kauppojen, tehokkuuksien perusteella.

Oletetaan, että päätöksentekoyksiköitä on n kappaletta, ja jokaisella yksiköllä on m kappaletta panoksia ja p kappaletta tuotoksia. Merkitään kaikkien yksiköiden panosten havaitut arvot matriisiin $X \in \mathbb{R}_{>}^{m\times n}$ ja tuotosten havaitut arvot matriisiin $Y \in \mathbb{R}_{>}^{p\times n}$ siten, että matriisin X sarakevektori x_i vastaa päätöksentekoyksikön i panosten lukumääriä ja matriisin Y sarakevektori y_i yksikön i tuotosten lukumääriä. Määritellään *tuotantomahdollisuusjoukko* T, joka koostuu päätöksentekoyksiköiden havaituista arvoista ja niiden semipositiivisista (≥ 0) lineaarikombinaatioista painovektoreilla $\lambda \in \Lambda$:

$$T = \{(x, y) \in \mathbb{R}_{\geq}^{m+p} : x \geq X\lambda, y \leq Y\lambda, \lambda \in \Lambda\}. \tag{9}$$

Λ on lineaarikombinaatioiden sallittujen painovektorien joukko. Tuotantomahdollisuusjoukko T muodostaa käyvän alueen virtuaalisille päätöksentekoyksiköille. Oletetaan, että tavoitteena on minimoida kaikki panokset ja samalla maksimoida kaikki tuotokset tuotantomahdollisuusjoukon määräämässä alueessa. Tämä voidaan esittää MOLP-tehtävänä

$$\underset{(x,y)\in T}{\text{v-min}}\ (x, -y). \tag{10}$$

MOLP-tehtävän (10) päätösavaruuden tehokkaat pisteet muodostavat tuotantomahdollisuusjoukon T *tehokkaan rintaman*. Päätöksentekoyksikön i tuotospainotteinen tehokkuus θ_i saadaan laskettua LP-tehtävän

$$\max\{\Theta_i : (x_i, \Theta_i y_i) \in T\} \tag{11}$$

optimiarvosta Θ_i^*: tällöin $\theta_i = 1/\Theta_i^*$.

Tehokkaalla rintamalla olevan päätöksentekoyksikön tuotospainotteinen tehokkuus saavuttaa arvon 1, ja tehottoman yksikön tuotospainotteinen tehokkuus riippuu sen etäisyydestä tehokkaaseen rintamaan. Tuotantomahdollisuusjoukkoa tutkimalla voidaan päätellä, mitä muutoksia tehoton yksikkö voi suorittaa päästäkseen lähemmäs tehokasta rintamaa ja parantaakseen täten tehokkuuttaan.

Tässä työssä tarkastellaan kahta eri tapaa määritellä painovektorien joukko Λ tuotantomahdollisuusjoukolle T, jotka johtavat kahteen eri DEA-malliin. CCR-malli (Charnes et al., 1978) saadaan, kun valitaan $\Lambda = \mathbb{R}^n_>$; mallissa tuotantomahdollisuusjoukolle pätee *vakioskaalatuotto-oletus*: jos $(x, y) \in T$, myös $(tx, ty) \in T$ kaikilla $t \in \mathbb{R}_\geq$.

BCC-mallissa (Banker et al., 1984) painovektorien joukko on puolestaan $\Lambda = \{\lambda \in \mathbb{R}^n_> : \sum_{i=1}^n \lambda_i = 1\}$. Tällöin tuotantomahdollisuusjoukon T tehokas rintama on paloittain lineaarinen ja muodostuu tehokkaiden päätöksentekoyksiköiden konvekseista kombinaatioista. Tehokkaan rintaman eri palojen pisteillä on erisuuret marginaalituottavuudet, joten BCC-mallissa on *muuttuva skaalatuotto*.

Jos oletetaan, että panokset $x \in \mathbb{R}^m_\geq$ ovat allokoitavia resursseja ja tuotokset $y \in \mathbb{R}^p_\geq$ maksimoitavia kohdefunktioita, saadaan yleinen resurssien allokointitehtävä formuloitua seuraavasti (Korhonen and Syrjänen, 2004):

$$\begin{aligned}
\underset{\Delta y_i, \Delta x_i, \lambda_i}{\text{v-max}} \quad & \Delta y = \Delta y_1 + \cdots + \Delta y_n && && (12)\\
\text{s.e.} \quad & y_i + \Delta y_i \leq Y\lambda_i, && i = 1, \ldots, n && (13)\\
& x_i + \Delta x_i \geq X\lambda_i, && i = 1, \ldots, n && (14)\\
& \lambda_i \in \Lambda, && i = 1, \ldots, n && (15)\\
& \alpha_i \leq \Delta x_i \leq \beta_i, && i = 1, \ldots, n && (16)\\
& \Delta x_1 + \cdots + \Delta x_n \leq r. && && (17)
\end{aligned}$$

Tehtävä (12) - (17) on MOLP-tehtävä, jossa maksimoidaan päätöksentekoyksiköiden tuotosten muutoksien summaa Δy siten, että allokoinnin jälkeiset resurssien ja tuotosten arvot $(x_i + \Delta x_i, y_i + \Delta y_i)$, $i = 1, \ldots, n$, on rajoitettu tuotantomahdollisuusjoukon (9) sisälle. Lisäksi resurssien muutokset Δx_i, $i = 1, \ldots, n$, on rajoitettu välille $[\alpha_i, \beta_i]$, ja budjettirajoitteen r avulla voidaan säätää resurssien kokonaismuutosta joko vähentämällä resursseja $(r < 0)$, kasvattamalla niitä $(r > 0)$ tai olla muuttamatta niiden määrää $(r = 0)$. Jos $r = 0$, ei tehtävään tule lisäresursseja ulkopuolelta, ja saadaan resurssien uudelleenallokointitehtävä päätöksentekoyksiköiden kesken.

Taulukossa 2 on esitetty suomalaisen kauppaketjun 25 kaupan olemassa olevat resurssit ja havaitut tuotokset (Korhonen and Syrjänen, 2004). Alkuperäistä mallia on yksinkertaistettu ottamalla huomioon ainoastaan kaksi resurssia, henkilötyötunnit ja kaupan koko, ja kaksi tuotosta, myynti ja voitto.

Nyt kaupat ovat päätöksentekoyksiköitä, ja päätöksentekijänä on kauppaketjun johto, joka pyrkii allokoimaan vapaat resurssit mahdollisimman tehokkaasti kauppojen kesken siten, että kokonaismyynti ja voitot maksimoituvat. Oletetaan, että yksittäisen kaupan resurssit voivat laskea enintään 10% ja nousta 30%, jolloin tehtävän (12) - (17) resurssien muutosten Δx_i rajoitteet ovat $\alpha_i = -0.1x_i$ ja $\beta_i = 0.3x_i$, $i = 1, \ldots, n$. Oletetaan lisäksi, että resurssien kokonaismäärä voi kasvaa enintään 1%, jolloin tehtävän budjettirajoitteeksi saadaan $r = 0.01 \sum_{i=i}^{n} x_i$. Kauppaketjun tuotokset maksimoivat resurssien allokointistrategiat saadaan ratkaistua laskemalla tehtävän (12) - (17) ei-dominoituja pisteitä vastaavat resurssien muutokset Δx_i, $i = 1, \ldots, n$.

4.2.1 Allokointi yksiköiden suhteellisella skaalauksella

Oletetaan aluksi, että päätöksentekoyksiköt voivat muuttaa toimintaansa ainoastaan olemassa olevien resurssien ja havaittujen tuotosten suhteellisella skaalauksella, ja että tuotantomahdollisuusjoukko (9) perustuu CCR-malliin, jolloin $\Lambda = \mathbb{R}^n_>$. Oletetaan lisäksi, että resurssit ja tuotokset muuttuvat samassa suhteessa, minkä ansiosta yksiköiden CCR-tehokkuudet eivät muutu resurssien allokoinnin aikana. Tämän perusteella riittää tutkia ainoastaan resurssien ja tuotosten suhteellisia muutoksia, ja allokointitehtäväksi saadaan tehtävä (12) - (17), jonka rajoitteet (13) - (15) korvataan rajoitteilla

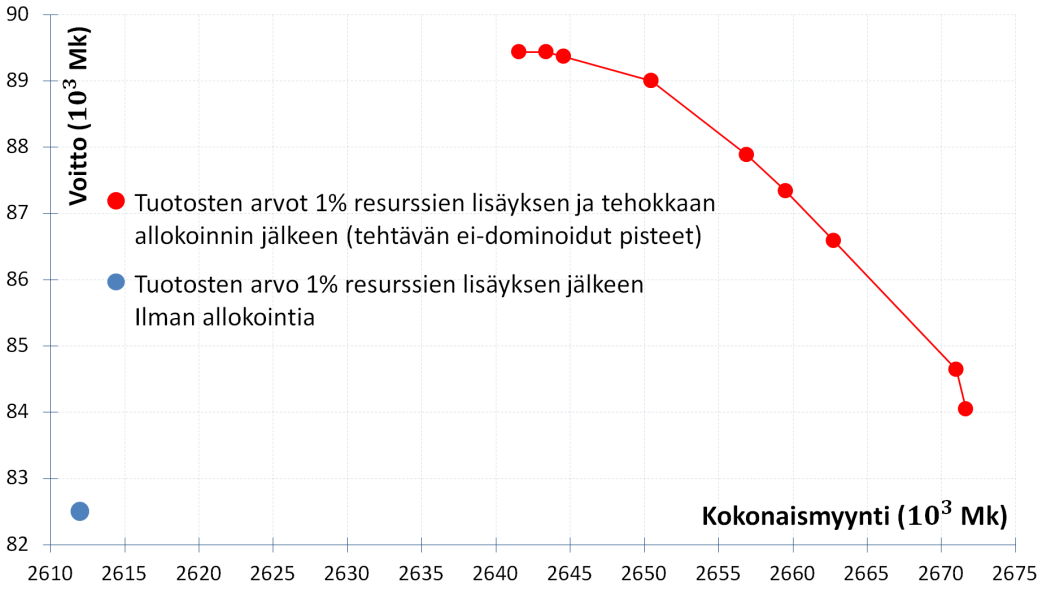
$$\begin{aligned} \Delta y_i &\leq \delta_i y_i, \quad i = 1, \ldots, n \\ \Delta x_i &\geq \delta_i x_i, \quad i = 1, \ldots, n. \end{aligned} \tag{18}$$

Bensonin algoritmi ratkaisee tehtävän noin 0.1 sekunnissa 3.4 GHz neliydinprosessorilla ja 8 GB keskusmuistilla varustetulla tietokoneella. Algoritmi palauttaa 9 ei-dominoitua pistettä, jotka maksimoivat kokonaismyynnin ja voitot, ja laskemalla ei-dominoituja pisteitä vastaavat päätösavaruuden tehokkaat pisteet saadaan 11 tehokasta resurssien allokointistrategiaa.

Kuvassa 3 on esitetty ei-dominoitujen tuotosparien arvot ja vertailun vuoksi tuotosten arvo 1% kokonaisresurssien lisäyksellä ilman allokointia. Tuloksista nähdään, että resurssien tehokas allokointi parantaa sekä kokonaismyyntiä että voittoa huomattavasti verrattuna pelkkään resurssien kasvattamiseen.

Taulukko 2: Suomalaisen kauppaketjun 25 kaupan resurssit, tuotokset ja suhteelliset DEA-tehokkuudet (Korhonen and Syrjänen, 2004)

Kauppa i	Henkilötyötunnit (10^3 h) x_{1i}	Koko (10^3 m^2) x_{2i}	Myynti (10^6 Mk) y_{1i}	Voitto (10^6 Mk) y_{2i}	CCR-tehokkuus θ_i	BCC-tehokkuus θ_i
1	79.1	4.99	115.3	1.71	0.801	0.821
2	60.1	3.30	75.2	1.81	0.688	0.772
3	126.7	8.12	225.5	10.39	1	1
4	153.9	6.70	185.6	10.42	0.919	1
5	65.7	4.74	84.5	2.36	0.707	0.769
6	76.8	4.08	103.3	4.35	0.778	0.806
7	50.2	2.53	78.8	0.16	0.902	1
8	44.8	2.47	59.3	1.30	0.727	1
9	48.1	2.32	65.7	1.49	0.803	1
10	89.7	4.91	163.2	6.26	1	1
11	56.9	2.24	70.7	2.80	0.921	1
12	112.6	5.42	142.6	2.75	0.745	0.824
13	106.9	6.28	127.8	2.70	0.657	0.673
14	54.9	3.14	62.4	1.42	0.625	0.736
15	48.8	4.43	55.2	1.38	0.622	0.803
16	59.2	3.98	95.9	0.74	0.890	0.978
17	74.5	5.32	121.6	3.06	0.897	0.930
18	94.6	3.69	107.0	2.98	0.814	0.817
19	47.0	3.00	65.4	0.62	0.765	0.969
20	54.6	3.87	71.0	0.01	0.715	0.804
21	90.1	3.31	81.2	5.12	0.855	0.858
22	95.2	4.25	128.3	3.89	0.847	0.876
23	80.1	3.79	135.0	4.73	1	1
24	68.7	2.99	98.9	1.86	0.929	0.973
25	62.3	3.10	66.7	7.41	1	1
Yhteensä	2 586.1	81.69	1 901.5	102.94		

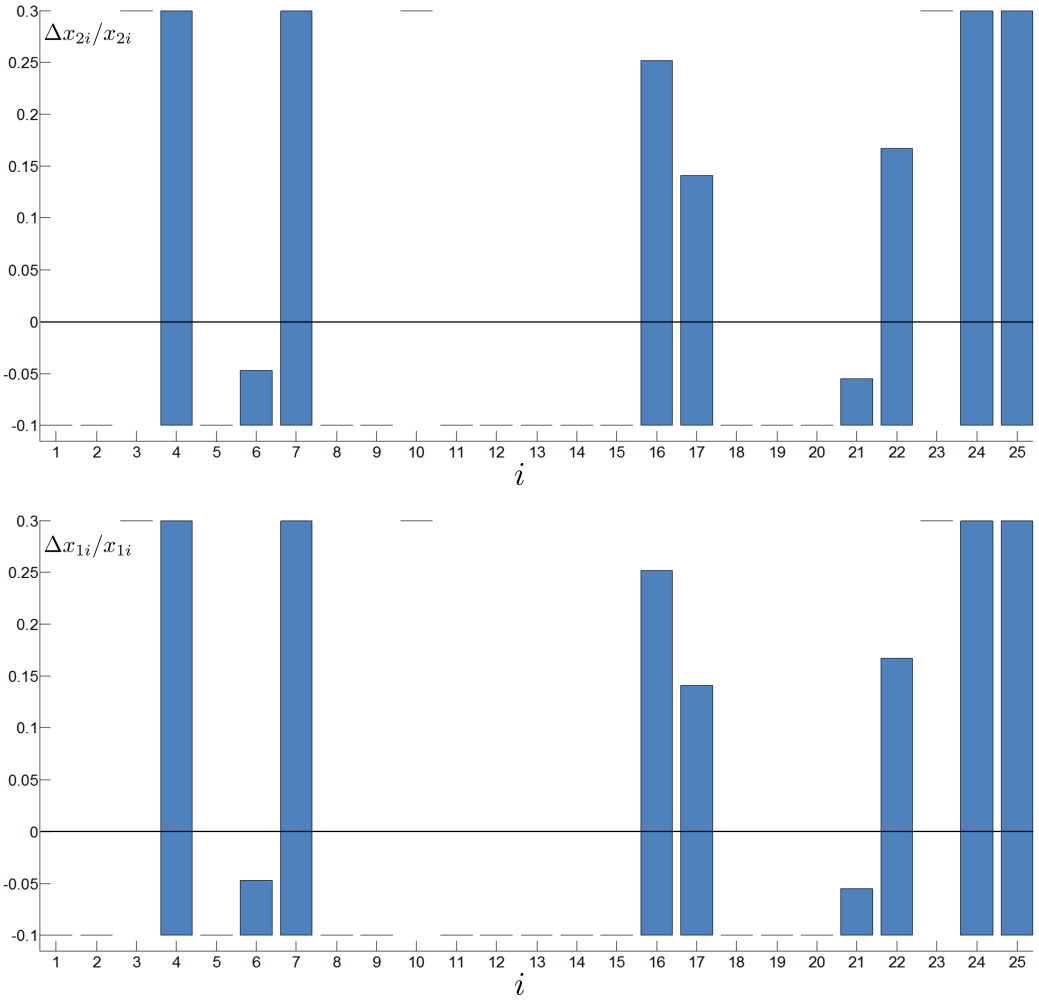


Kuva 3: Resurssien allokointitehtävän ratkaisut tavoiteavaruudessa, kun kokonaisresurssit kasvavat 1%. Lisäksi kuvassa on esitetty tuotosten arvot 1% resurssien lisäyksellä ilman allokointia.

Tulosten perusteella kauppaketjun johto voi halutessaan parantaa kokonaismyyntiä noin 2.3%, voittoa 8.4% tai molempia tasaisesti preferenssiensä perusteella.

Ei-dominoituja pisteitä vastaavista tehokkaista pisteistä saadaan selville resurssien muutosten arvot, joiden avulla kauppaketjun johto voi tutkia eri ratkaisujen vaikutuksia yksittäisten kauppojen toimintaan. Rajoitusten (18) ansiosta molemmat resurssit muuttuvat samassa suhteessa. Kuvassa 4 on esitetty molempien resurssien suhteellisten muutosten, $\Delta x_{1i}/x_{1i}$ ja $\Delta x_{2i}/x_{2i}$, $i = 1, \ldots, 25$, vaihteluvälit tehtävän ratkaisuille. Kuvasta nähdään, että useimmista kaupoista poistetaan aina maksimimäärä resursseja tehokkaampien kauppojen käyttöön. Tehokkaisiin kauppoihin 3, 10 ja 23 lisätään aina suurin sallittu määrä resursseja. Kaupoista 6 ja 21 poistetaan resursseja vaihtelevasti jokaisessa ratkaisussa. Toisaalta kauppojen 4, 7, 16, 17, 22, 24 ja 25 resurssit voivat joko kasvaa tai laskea; resurssien muutokset riippuvat siitä, panostetaanko enemmän kokonaismyyntiin vai voittoon.

Yllättäen monitavoite-Simplexillä tehtävän ratkaisemiseen kuluu useita päiviä, joten algoritmien suoritusaikojen välinen ero on uskomattoman suuri. Tämä voidaan selittää tutkimalla tehtävän päätös- ja tavoiteavaruuden dimensioita sekä rajoitteiden lukumäärää. Tehtävässä on 125 päätösmuuttu-



Kuva 4: Resurssien x_{1i} (henkilötyötunnit) ja x_{2i} (kaupan koko) suhteelliset muutokset $\Delta x_{1i}/x_{1i}$ ja $\Delta x_{2i}/x_{2i}$, $i = 1, \ldots, 25$, tehtävän ratkaisuille

jaa ja kaksi kohdefunktiota, joten päätösavaruuden dimensio on yli 50 kertaa suurempi kuin tavoiteavaruuden dimensio. Lisäksi tehtävän epäyhtälörajoitteiden lukumäärä on yli 200. Päätösavaruuden tehokkaiden pisteiden lukumäärä kasvaa eksponentiaalisesti suhteessa päätösmuuttujien ja rajoitteiden lukumäärään, joten tehokkaita pisteitä on huomattavasti enemmän kuin ei-dominoituja pisteitä, joiden lukumäärä puolestaan riippuu pääosin tavoiteavaruuden dimensiosta (Benson, 1998; Dauer, 1987). Koska useimmat tehokkaat pisteet kuvautuvat samoiksi ei-dominoiduiksi pisteiksi, tekee monitavoite-Simplex todella paljon turhaa työtä.

4.2.2 Allokointi yksiköiden muuttumattomilla tehokkuuksilla

Oletetaan seuraavaksi, että yksiköt eivät voi parantaa tehokkuuksiaan resurssien allokoinnin aikana, ja että tuotantomahdollisuusjoukko (9) perustuu BCC-malliin, jolloin $\Lambda = \{\lambda \in \mathbb{R}^n_> : \sum_{i=1}^n \lambda_i = 1\}$. Taulukossa 2 on esitetty LP-tehtävän (11) avulla lasketut kauppojen tuotospainotteiset BCC-tehokkuudet θ_i, $i = 1, \ldots, n$. Tehokkuusrajoitukset saadaan tuotua resurssien allokointimalliin kertomalla tehtävän (12) - (17) rajoitteen (13) oikea puoli tehokkuusluvulla θ_i, jolloin vastaavaksi rajoitteeksi saadaan

$$y_i + \Delta y_i \leq \theta_i Y \lambda_i, \quad i = 1, \ldots, n. \tag{19}$$

Lisäksi, koska tuotantomahdollisuusjoukko perustuu BCC-malliin, tehtävän (12) - (17) rajoite (15) kirjoitetaan muotoon

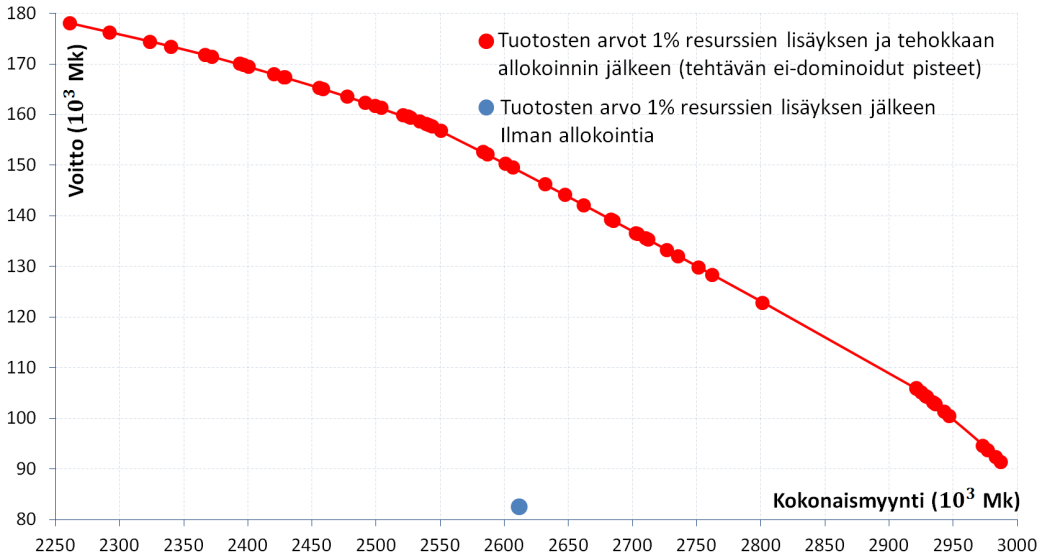
$$\begin{aligned} e^T \lambda_i &= 1, \quad i = 1, \ldots, n \\ \lambda_i &\geq 0, \quad i = 1, \ldots, n \end{aligned} \tag{20}$$

missä $e = (1, \ldots, 1) \in \mathbb{R}^n$.

Nyt tehtävän päätösmuuttujina ovat resurssien muutosten Δx_i lisäksi painovektorien λ_i komponentit, joita on yhteensä $25^2 = 625$ kappaletta. Päätösavaruuden dimensio siis kasvaa huomattavasti verrattuna edelliseen esimerkkiin, ja kuten saattaa arvata, on tehtävää lähes mahdotonta ratkaista monitavoite-Simplexillä tehokkaiden pisteiden huiman lukumäärän takia.

Toisaalta Bensonin algoritmi ratkaisee tehtävän alle 10 sekunnissa, mikä vahvistaa ajatusta algoritmin paremmuudesta monitavoite-Simplexiin verrattuna isoissa käytännön ongelmissa. Tämä tukee myös oletusta, jonka mukaan Bensonin algoritmin ratkaisutehokkuus ei riipu merkittävästi päätösmuuttujien ja rajoitteiden lukumäärästä vaan enemmänkin tavoiteavaruuden dimensiosta. Oletusta voitaisiin tutkia generoimalla satunnaisia tehtäviä erilaisilla tavoitefunktioiden lukumäärillä ja vertailemalla algoritmin suoritusaikoja.

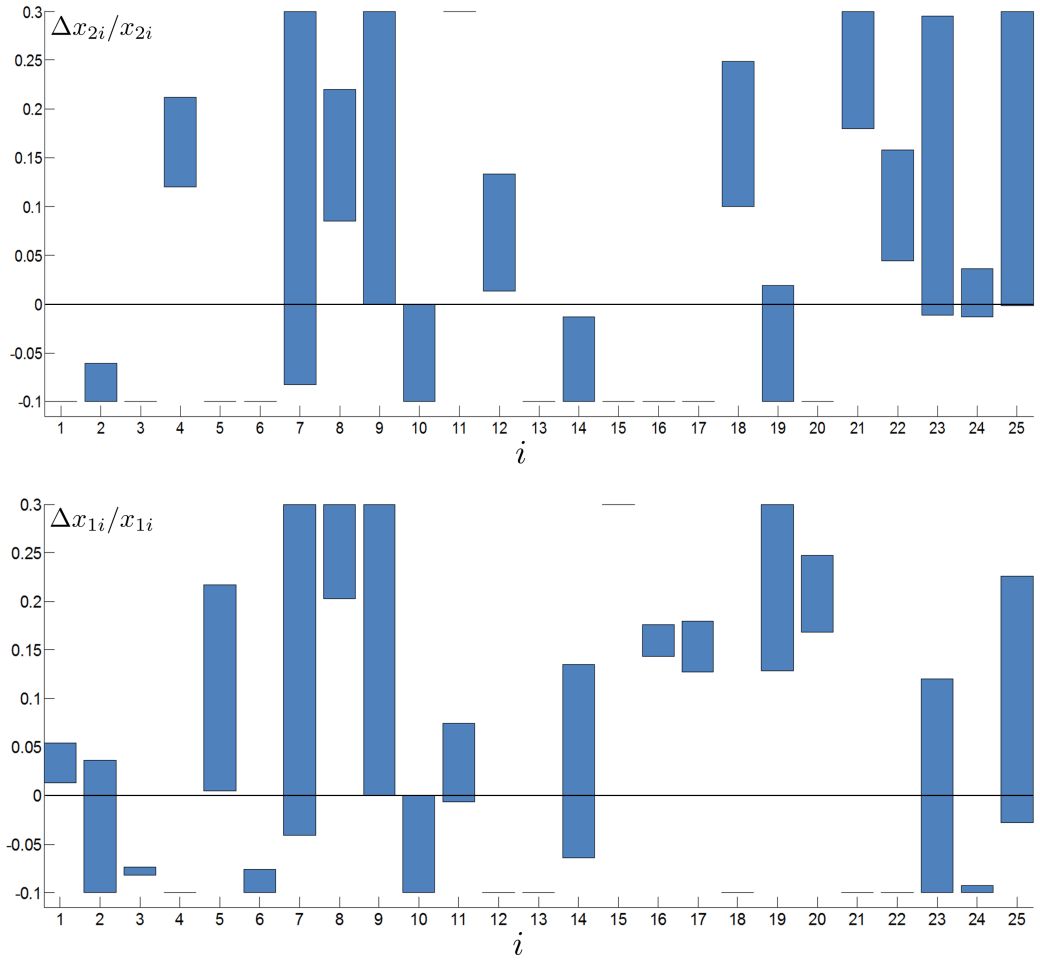
Algoritmi palauttaa 58 ei-dominoitua pistettä, ja laskemalla vastaavat päätösavaruuden tehokkaat pisteet saadaan yhteensä 150 tehokasta resurssien allokointistrategiaa. Kuvassa 5 on esitetty ei-dominoitujen tuotosparien arvot sekä tuotosten arvo 1% kokonaisresurssien lisäyksellä ilman allokointia. Samoin kuin edellisessä esimerkissä, resurssien tehokas allokointi parantaa sekä kokonaismyyntiä että voittoa merkittävästi verrattuna pelkkään resurssien



Kuva 5: Resurssien allokointitehtävän ratkaisut tavoiteavaruudessa, kun kokonaisresurssit kasvavat 1%. Lisäksi kuvassa on esitetty tuotosten arvot 1% resurssien lisäyksellä ilman allokointia.

kasvattamiseen. Nyt kauppaketjun johto voi halutessaan parantaa kokonaismyyntiä 14.4%, voittoa jopa 115.9% tai molempia tasaisesti preferenssiensä perusteella. Malli tosin olettaa kauppojen pystyvän muuttamaan kokoaan vaivattomasti, mikä ei käytännössä ole välttämättä mahdollista. Koko voitaisiinkin korvata jollain toisella resurssilla tai poistaa mallista kokonaan, mutta se sopii kuitenkin mallin havainnollistamiseen.

Kuvissa 6 ja 7 on esitetty tehokkaita resurssien allokointistrategioita vastaavat resurssien suhteelliset muutokset. Kuvassa 6 on esitetty resurssin x_1 (henkilötyötunnit) suhteellisten muutosten $\Delta x_{1i}/x_{1i}$, $i = 1, \ldots, 25$ vaihteluvälit ja kuvassa 7 resurssin x_2 (kaupan koko) suhteellisten muutosten $\Delta x_{2i}/x_{2i}$, $i = 1, \ldots, 25$ vaihteluvälit tehtävän ratkaisuille. Kuvien perusteella yhteenkään kauppaan ei lisätä aina maksimimäärää (30%) molempia resursseja, ja ainoastaan kaupasta 13 poistetaan aina suurin sallittu määrä (10%) molempia resursseja. Muutoin resursseja liikutellaan paljon vapaammin verrattuna edellisen tehtävän ratkaisuihin: esimerkiksi kauppaan 15 lisätään aina maksimimäärä henkilötyötunteja, ja toisaalta siitä vähennetään aina suurin sallittu määrä kaupan pinta-alaa.



Kuva 6: Resurssien x_{1i} (henkilötyötunnit) ja x_{2i} (kaupan koko) suhteelliset muutokset $\Delta x_{1i}/x_{1i}$ ja $\Delta x_{2i}/x_{2i}$, $i = 1, \ldots, 25$, tehtävän ratkaisuille

5 Yhteenveto

Työssä tarkasteltiin lineaarisen monitavoiteoptimointitehtävän ratkaisukäsitteitä, ja tutkittiin MOLP-tehtävien ratkaisemista päätös- ja tavoiteavaruudessa. Erityisesti työssä pohdittiin päätösavaruudessa ratkaisemisen haasteita ja esiteltiin Bensonin algoritmi MOLP-tehtävien ratkaisemiseen tavoiteavaruudessa. Algoritmin toimintaa tutkittiin ensin numeerisella esimerkillä, ja sen soveltuvuutta käytännön ongelmien ratkaisemiseen kahden resurssien allokointitehtävän avulla (Korhonen and Syrjänen, 2004). Lisäksi Bensonin algoritmia verrattiin kirjallisuudessa usein mainittuun monitavoite-Simplex -algoritmiin (Ehrgott, 2005) vertailemalla algoritmien suoritusaikoja.

Bensonin algoritmi osoittautui tehottomaksi monitavoite-Simplexiin verrattuna esimerkkitehtävässä, missä kohdefunktioita on saman verran kuin päätösmuuttujia, ja jokaista tehokasta pistettä vastaa yksi ei-dominoitu piste. Käytännön ongelmissa on kuitenkin tavallisesti erilainen rakenne; ensinnäkin kohdefunktioita on normaalisti huomattavasti vähemmän kuin päätösmuuttujia, ja toiseksi usein monta tehokasta pistettä kuvautuu samaksi ei-dominoiduksi pisteeksi (Benson, 1998; Dauer, 1987). Nämä kaksi tulosta pitävät paikkansa työssä tarkastelluissa resurssien allokointitehtävissä, ja niiden vaikutus Bensonin algoritmin tehokkuuteen osoittautui yllättävän suureksi.

Bensonin algoritmi ratkaisi molemmat allokointitehtävät vaivattomasti sekunneissa; toisaalta monitavoite-Simplexillä ensimmäinen tehtävä ratkesi vasta useiden päivien ajon jälkeen, ja toinen tehtävä osoittautui liian suureksi ratkaistavaksi järkevässä ajassa. Tulosten perusteella Bensonin algoritmi soveltuu erinomaisesti isojen käytännön ongelmien ratkaisemiseen, joissa on vähän kohdefunktioita verrattuna päätösmuuttujien lukumäärään. Käytännössä kohdefunktioita onkin tavallisesti vähän, mutta väite algoritmin tehokkuudesta yleisesti isoissa käytännön ongelmissa vaatii lisätutkimuksia.

Viitteet

Banker, R. D., Charnes, A., and Cooper, W. W. (1984). Some models for estimating technical and scale inefficiencies in data envelopment analysis. *Management science*, 30(9):1078–1092.

Benson, H. (1998). An outer approximation algorithm for generating all efficient extreme points in the outcome set of a multiple objective linear programming problem. *Journal of Global Optimization*, 13:1–24.

Benson, H. P. and Sayin, S. (1997). Towards finding global representations of the efficient set in multiple objective mathematical programming. *Naval Research Logistics (NRL)*, 44(1):47–67.

Berkelaar, M., Eikland, K., and Notebaert, P. (2004). lp_solve 5.5, Open source (Mixed-Integer) Linear Programming system. http://lpsolve.sourceforge.net/5.5/. Accessed: 01/12/2012.

Charnes, A., Cooper, W. W., and Rhodes, E. (1978). Measuring the efficiency of decision making units. *European journal of operational research*, 2(6):429–444.

Dauer, J. P. (1987). Analysis of the objective space in multiple objective linear programming. *Journal of Mathematical Analysis and Applications*, 126(2):579 – 593.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer Berlin, second edition.

Ehrgott, M. and Wiecek, M. (2005). Mutiobjective programming. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, pages 667–722. Springer New York.

Isermann, H. (1974). Proper efficiency and the linear vector maximum problem. *Operations Research*, 22(1):189–191.

Korhonen, P. and Syrjänen, M. (2004). Resource allocation based on efficiency analysis. *Management Science*, 50(8):1134–1144.

Löhne, A. (2011). *Vector Optimization with Infimum and Supremum*. Springer Heidelberg, Dordrecht, London, Newyork, first edition.

Löhne, A. (2012). bensolve-1.2. Matlab implementation of Benson's algorithm to solve linear vector optimization problems. http://ito.mathematik.uni-halle.de/~loehne/page_en/downloads.php. Accessed: 03/01/2013.

Makhorin, A. (2006). GLPK (GNU Linear Programming Kit). http://www.gnu.org/software/glpk/. Accessed: 05/02/2013.

Meindl, B. and Templ, M. (2012). Analysis of commercial and free and open source solvers for linear optimization problems.

Nummenpalo, J. (2012). *Lineaaristen monitavoiteoptimointitehtävien ratkaiseminen*. Kandidaatintyö, Aalto-yliopisto.

Shao, L. and Ehrgott, M. (2008). Approximating the nondominated set of an molp by approximately solving its dual problem. *Mathematical Methods of Operations Research*, 68:469–492.

Steuer, R. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application*. John Wiley and Sons, New York.