Pulmapelien matemaattinen mallintaminen ja ratkaiseminen

Jenni Lampainen

Pro gradu -tutkielma
Toukokuu 2024

Tarkastajat:
Kaisa Joki
Stefan Emet

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Jenni Lampainen: Pulmapelien matemaattinen mallintaminen ja ratkaiseminen
Pro gradu -tutkielma, 77 s., 39 liites.
Sovellettu matematiikka
Toukokuu 2024

Tässä tutkielmassa tarkastellaan erilaisia pulmapelejä sekä niiden matemaattista mallintamista ja ratkaisemista. Lähemmin käsitellään tiilitys- ja reunasovitusongelmia, erilaisia sudokuja sekä summafunktioon perustuvia pulmapelejä. Tiilitysongelmissa annetut epäsymmetriset palat tulee sijoittaa pelilaudalle siten, että koko alue tulee peitetyksi. Puolestaan reunasovitusongelmissa annetut symmetriset palat tulee sijoittaa pelilaudalle siten, että vierekkäisten palojen reunojen kuviot sopivat yhteen. Sudoku on tunnettu pulmapeli, jossa luvut 1–9 tulee sijoittaa ruudukkoon annettujen sääntöjen mukaisesti. Summafunktioon perustuvien pulmapelien keskeinen idea on se, että tietyssä rivissä tai sarakkeessa esiintyvät luvut tai symbolien lukumäärät summautuvat annettuun vihjenumeroon.

Tutkielma alkaa katsauksella pulmapelien historiaan, jonka jälkeen tutustutaan keskeisimpiin käsitteisiin ja mallinnustapoihin. Tämän jälkeen siirrytään tarkastelemaan lähemmin neljää edellä mainittua pulmapelikategoriaa. Jokaisesta pulmapelistä muodostetaan optimointitehtävä, jonka mallinnus käydään yksityiskohtaisesti läpi. Joillekin pulmapeleille muodostetaan muutama vaihtoehtoinen malli. Lopuksi pulmapeliongelmia ratkaistaan numeerisesti GAMS-ohjelmistolla, ja saatuja tuloksia vertaillaan keskenään. Yhtäläisyyksiä ja eroja etsitään sekä kaikkien tarkasteltujen pulmapelien väliltä, että myös yhden pulmapelityypin modifikaatioiden väliltä.

Asiasanat: pulmapelit, Eternity II, sudoku, lineaarinen binäärinen optimointi.

Sisällys

1 Johdanto

Pulmapelit ovat tehtäviä, joiden ratkaisemiseen tarvitaan ongelmanratkaisutaitoja sekä päättelykykyä. Pulmapelit perustuvat pitkälti matematiikkaan sekä logiikkaan. Niitä voidaan ratkaista pelkkää päättelykykyäkin hyödyntämällä, mutta tässä tutkielmassa erilaisille pulmapeleille esitellään matemaattiset optimointimallit. Kun nämä mallit syötetään tietokoneen numeeriseen ratkaisijaan, voidaan pulmapelin ratkaisu saada jopa sekunneissa.

Optimointi on matematiikan osa-alue, jossa tavoitteena on löytää sallituissa olosuhteissa annettuun ongelmaan paras mahdollinen ratkaisu. Tarkastelussa oleva ongelma ilmaistaan kohdefunktiona ja rajoitteina. Kohdefunktiolla mitataan ratkaisun hyvyyttä. Toisin sanoen optimointitehtävässä pyritään löytämään kohdefunktion suurin tai pienin arvo rajoitteiden määräämässä alueessa. Jatkossa pulmapelin säännöt kirjoitetaan rajoitteiksi. Esimerkiksi sääntö siitä, että jokin luku saa esiintyä yhdellä rivillä vain kerran, kirjoitetaan siis rajoitteeksi.

Vaikka monet pulmapelit, esimerkiksi sudokut, ratkeavat vaikeustasosta riippuen melko helposti vain päättelykyvyllä, on matemaattisesta optimoinnista hyötyä pulmapelien ratkaisemisessa. On nimittäin olemassa haastavia pulmapelejä, jotka eivät helposti ja nopeasti ratkea pelkällä päättelykyvyllä. Tällaisissa tapauksissa erilaisia mahdollisuuksia esimerkiksi palojen tai numeroiden sijoittamiseen on tyypillisesti niin paljon, että niitä kaikkia ei ole mitenkään mahdollista vain kokeilla läpi. Ainoaksi vaihtoehdoksi jääkin siis pulmapelin mallintaminen matemaattisesti ja tämän mallin syöttäminen tietokoneen numeeriseen ratkaisijaan. On kuitenkin hyvä tiedostaa, että aina numeerinen ratkaisijakaan ei ole ratkaisu kaikkeen. Nimittäin välillä tehtävät ovat niin vaikeita, että niiden ratkaiseminen ei äärellisessä ajassa onnistu välttämättä edes tietokoneella. Eräs esimerkki tällaisesta ongelmasta on Eternity II -palapeli, jolle on noin $1{,}15 \cdot 10^{661}$ mahdollista tapaa sijoittaa palat pelilaudalle. Tästä huolimatta optimointi on tärkeä työkalu pulmapelien, etenkin vaikeiden pulmapelien, ratkaisemisessa, koska optimoinnilla voidaan usein löytää hyvinkin nopeasti ratkaisu sellaisiin pulmapeleihin, joiden ratkaiseminen pelkällä päättelykyvyllä kestäisi kauan.

Tutkielma esittelee tarkemmin kahden yksinkertaisen tiilitysongelman sekä Eternity I -palapelin esimerkkeinä tiilitysongelmasta ja Eternity II -palapelin esimerkkinä reunasovitusongelmasta. Lisäksi Eternity II -palapelistä on kehitetty tässä tutkielmassa kaksi modifikaatiota. Tarkemmin esitellään myös neljä erilaista sudokua sekä kakuro ja japanilainen ristikko esimerkkeinä summafunktioon perustuvista pulmapeleistä. Kyseiset pulmapelit on valittu tarkasteltaviksi siksi, että ne antavat kattavan kuvan sellaisista pulmapeleistä, joita on mahdollista mallintaa matemaattisella

optimoinnilla.

Tutkielman rakenne on seuraava: Luvussa 2 käydään lyhyesti läpi tarkasteltavien pulmapelien historiaa. Luvussa 3 esitellään keskeisiä käsitteitä ja mallinnustapoja. Tämän jälkeen käydään läpi tiilitys- ja reunasovitusongelmia, sudokuja sekä summafunktioon perustuvia pulmapelejä luvuissa 4-7. Kyseisistä pulmapelikategorioista käydään läpi myös konkreettisia pulmapelejä, joista muodostetaan optimointitehtävät. Luvussa 8 nämä optimointitehtävät ratkaistaan ohjelmistoa käyttäen ja saatuja numeerisia tuloksia vertaillaan keskenään. Tutkielman viimeisessä luvussa on lyhyt yhteenveto käsitellyistä pulmapeleistä ja saaduista tuloksista. Lisäksi liitteeseen A on koottu optimointitehtävien ratkaisemisessa käytetyt GAMS-mallit.

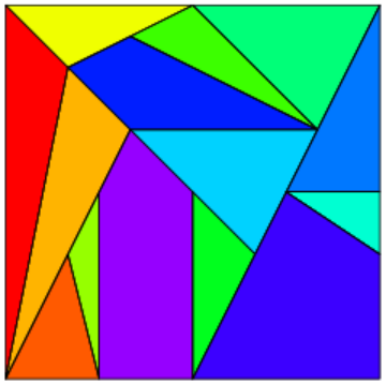
2 Pulmapelien historia

Pulmapeli (*puzzle*) on laaja käsite, joka pitää sisällään esimerkiksi arvoitukset, sanaristikot, palapelit ja sudokut. Tämä tutkielma tarkastelee nimenomaan palapelejä ja numeerisia, logiikkaan sekä kombinatoriikkaan, perustuvia pulmapelejä (kuten sudokuja). Toisin sanoen tarkastellaan pulmapelejä, joita voidaan mallintaa matemaattisesti. Seuraavaksi käydään lyhyesti läpi pulmapelien historiaa keskittyen erityisesti jatkossa käsiteltäviin pulmapeleihin.

Maailman ensimmäisenä matemaattisena pulmapelinä pidetään Ostomakhionia (*Ostomachion*) [9, 32], jota Arkhimedes (287-212 eaa.) tutki antiikin aikaan. Ostomakhion on 14 osainen palapeli. Paloista 11 on kolmioita, kaksi nelikulmioita ja yksi viisikulmio. Tarkoituksena on sijoittaa nämä palat neliön muotoiseen alustaan ja mahdollisia ratkaisuja on monta erilaista. Itse asiassa on osoitettu, että erilaisia ratkaisuja on 536 kappaletta [35], kun toisistaan peilaamalla tai rotatoimalla saadut ratkaisut oletetaan samoiksi. Kuvassa 1 on esitetty yksi ratkaisu Ostomakhionille.

Jigsaw-palapelit [36] ovat suosituin sekä tunnetuin palapelien muoto. Useimmissa tällaisissa palapeleissä on annettu kuva, ja palat tulee sijoittaa siten, että kyseinen kuva muodostuu. Tällöin jokainen pala sopii täsmälleen yhteen kohtaan. Ensimmäiset Jigsaw-palapelit tehtiin 1760-luvulla puulevyistä pistosahaa (*jig saw*) käyttämällä. Tästä juontaakin kyseisten palapelien nimi. Aluksi palapelejä käytettiin lähinnä kouluissa opetuksen apuna, ja vasta 1900-luvun alussa puisia Jigsaw-palapelejä alettiin tuottaa aikuisille. Nykyään palapelit tehdään pääsääntöisesti pahvista.

Reunasovituspalapelit (*edge-matching puzzles*) [11, 25] kehitettiin 1890-luvulla ja näiden idea on melko vastaava kuin Jigsaw-palapelien. Reunasovituspalapelissä tavoitteena on järjestää useita identtisesti muotoiltuja, mutta eri tavoin kuvioituja paloja siten, että vierekkäisten palojen vastakkaiset reunat muodostavat kuvion.



Kuva 1: Yksi ratkaisu Ostomakhionille, joka on maailman ensimmäinen matemaattinen pulmapeli. Kuva on peräisin sivustolta [32].

Tyypillisesti nämä palat ovat neliöitä. Lisäksi reunasovituspalapeleissä on jokin tietty muoto (ikään kuin alusta), johon palat tulee laittaa. Reunasovituspalapelin palojen vierekkäisistä sivuista voidaan pyrkiä muodostamaan esimerkiksi yhtenäinen maisema tai hahmo. Kuitenkin yksinkertaisissa reunasovitusongelmissa palojen reunat ovat jonkin värisiä, ja tavoitteena on sijoittaa samanväriset reunat vierekkäin. On myös olemassa haastavia abstrakteja reunasovituspalapelejä. Näissä palojen reunoilla on värin lisäksi merkki ($+$ tai $-$), ja vierekkäisillä reunoilla täytyy olla sama väri mutta eri merkit. Reunasovituspalapelit ovat haastavia tavallisiin Jigsaw-palapeleihin verrattuna seuraavien syiden vuoksi. Reunasovituspalapeleissä ei ole annettu kuvaa, jonka mukaan palojen sijainteja voisi päätellä. Lisäksi se, että kahden reunasovituspalapelin palan reunojen kuviot/värit sopivat yhteen, ei takaa sitä, että palojen tulee olla vierekkäin. Toisin on Jigsaw-palapeleissä, jossa palojen sopiminen vierekkäin takaa niiden olevan vierekkäin. Ainoastaan reunasovitusongelman lopullisen ratkaisun tuottaminen takaa paikallisten osien ratkaisujen oikeellisuuden.

Monikulmiopalapelien (*polyform packing puzzles*) [11] idea on myös melko vastaava kuin Jigsaw-palapelien. Tavoitteena on täyttää monikulmion muotoinen pelilauta kokonaan paloilla, jotka ovat myös monikulmiota. Tässä ei siis ole tarkoitus muodostaa paloista ja näiden väreistä mitään kuviota, ainoastaan täyttää pelilauta kokonaan. Vaikka monikulmiopalapelit siis perustuvat Jigsaw-palapelien ideaan, on näillä kahdella palapelillä myös seuraava ero. Monikulmiopalapeleissä nimittäin se, että kaksi palaa sopivat yhteen, ei takaa niiden kuuluvat yhteen. Kahdella palalla on myös useita tapoja sopia yhteen rotatoitumisen takia. Monikulmiopalapelit ovat peräisin vuodelta 1965, jolloin ensimmäinen monikulmiopalapeli julkaistiin [18]. Kuitenkin edellä esiteltyä Ostomakhionia voidaan pitää aivan ensimmäisenä monikulmiopalapelinä. Nämä palapelit nousivat suosioon vuonna 1999 julkaistun Eter-

nity I -palapelin innostamana, jonka pelilauta on 12-kulmio. Monikulmiopalapelit kuuluvat tiilitysongelmien (*tiling problems*) joukkoon.

Esitellyt kolme palapelityyppiä ovat laskennalliselta tehokkuudeltaan samanlaisia [11]. Nimittäin jokainen näistä palapelityypeistä voidaan muuttaa vastaavaksi toisen tyyppiseksi palapeliksi pienellä koon muuttamisella. Lisäksi esitellyt palapelityypit ovat laskennallisesti hankalia: NP-täydellisiä [16]. Tämä tarkoittaa sitä, että ei ole löydetty algoritmia, joka ratkaisisi tämän tyyppisiä palapelejä polynomisessa ajassa.

Siirrytään nyt tarkastelemaan muita pulmapelejä kuin palapelejä. Sudoku [2, 28] on laajalti tunnettu logiikkaan perustuva pulmapeli, jossa tarkoituksena on sijoittaa ruudukkoon numeroita annettujen sääntöjen mukaisesti. Sveitsiläinen matemaatikko Leonhard Euler kehitti vuonna 1783 latinalaisen neliön [1, 31], jota pidetään sudokun ensimmäisenä versiona. Kuitenkin vasta 1970-luvulla amerikkalainen Dell Magazines -lehti kehitti näiden pohjalta ristikon, joka perustui nykyisen sudokun sääntöihin. Sudokut eivät siis ole japanilainen keksintö, vaikka nimi kyseiseen maahan viittaakin. Nimittäin vuonna 1983 japanilainen Nikoli-yhtiö toi Dell Magazinesin ristikot Japaniin nimellä "Suuji Wa Dokushin Ni Kagiru", mutta pitkä nimi päätettiin lyhentää sudokuksi. Alkuperäinen japanilainen nimi on suomennettuna "numerot voivat esiintyä vain kerran". Tämä tiivistääkin sudokun pääidean. Vuonna 2004 honkongilainen Wayne Gould toi sudokut Englantiin ja antoi The Times -lehdelle ilmaiseksi ohjelman, jolla sudokuja oli mahdollista luoda. Tästä alkoikin sudokun maailmanvalloitus. Se, että sudokut levisivät nopeasti ympäri maapallon johtuikin pääosin siitä, että sudoku ei ole riippuvainen kielestä toisin kuin esimerkiksi perinteiset sanaristikot.

Vaikka sudoku on kenties tunnetuin matemaattinen pulmapeli, ei se ole vanhin. Nimittäin esimerkiksi kakuro (*cross sums*) [10] on kehitetty vuonna 1966 eli muutama vuosi ennen sudokua. Kakuron kehittäjä on kanadalainen Jacob E. Funk, pulmapeleistään tunnetun Dell Magazines -lehden työntekijä. Kyseinen lehti on kehittänyt siis sekä sudokut että kakurot. Yhteistä sudokuille ja kakuroille on se, että ruudukkoihin täytetään lukuja yhden ja yhdeksän väliltä. Japanilainen ristikko (*nonogram*) [29, 37] on puolestaan pulmapeli, jossa numeroiden sijaan ruudukkoa väritetään. Vuonna 1987 japanilainen graafinen suunnittelija Non Ishida sai idean japanilaisesta ristikosta. Sattumalta japanilainen arvoituksia intohimoisesti ratkaiseva Tetsuya Nishio kehitti japanilaisen ristikon myös samaan aikaan. Kumpikin julkaisi oman kehitelmänsä eri lehdessä. Kaiken kaikkiaan japanilainen ristikko on tässä tarkastelluista pulmapeleistä uusin.

3 Keskeiset työkalut ja mallinnustavat

Seuraavaksi esitellään keskeisiä mallinnukseen, ja erityisesti matemaattiseen optimointiin, liittyviä asioita. Optimointitehtävän yleinen muoto on

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & f(\boldsymbol{z}) \\ \text{s. t.} \quad & g_i(\boldsymbol{z}) \leq 0, \quad i = 1, \ldots, N_{ie} \\ & h_j(\boldsymbol{z}) = 0, \quad j = 1, \ldots, N_e \\ & \boldsymbol{z} \in Z_1 \times Z_2, \end{aligned} \tag{1}$$

missä päätösmuuttujavektori $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2)^\top$ sisältää sekä jatkuvat muuttujat $\boldsymbol{z}_1 \in \mathbb{R}^n$ että diskreetit muuttujat $\boldsymbol{z}_2 \in \mathbb{Z}^m$. Tehtävässä on N_{ie} epäyhtälörajoitetta ja N_e yhtälörajoitetta. Oletetaan, että optimointitehtävän kohdefunktio f on funktio avaruudesta $\mathbb{R}^n \times \mathbb{Z}^m$ reaalilukujoukkoon. Lisäksi oletetaan, että epäyhtälörajoitteiden funktiot g_i ovat funktioita avaruudesta $\mathbb{R}^n \times \mathbb{Z}^m$ reaalilukujoukkoon kaikilla $i = 1, \ldots, N_{ie}$ ja yhtälörajoitteiden funktiot h_j ovat vastaavasti funktioita avaruudesta $\mathbb{R}^n \times \mathbb{Z}^m$ reaalilukujoukkoon kaikilla $j = 1, \ldots, N_e$. Lisäksi usein on tapana olettaa, että kaikki tehtävän funktiot ovat jatkuvasti differentioituvia joukossa $Z_1 \times Z_2$. Tässä tyypillisesti Z_1 on avaruuden $\mathbb{R}^n$ kompakti osajoukko ja Z_2 avaruuden $\mathbb{Z}^m$ äärellinen diskreetti osajoukko. Nämä kaksi joukkoa rajoittavat päätösmuuttujavektorin $\boldsymbol{z}$ arvoja. Yhdessä kaikki rajoitteet muodostavatkin *sallitun alueen*

$$\begin{aligned} S = \{ & \boldsymbol{z} \in \mathbb{R}^n \times \mathbb{Z}^m \mid g_i(\boldsymbol{z}) \leq 0, \forall i = 1, \ldots, N_{ie}; \\ & h_j(\boldsymbol{z}) = 0, \forall j = 1, \ldots, N_e; \ \boldsymbol{z} \in Z_1 \times Z_2 \}, \end{aligned}$$

joka on osajoukko päätösmuuttujavektorin avaruudesta $\mathbb{R}^n \times \mathbb{Z}^m$. Joukko S sisältää ne pisteet, jotka toteuttavat kaikki rajoitteet.

Yllä kohdefunktion f pienintä arvoa etsitään sallitussa alueessa S. *Globaali optimi* on piste $\boldsymbol{z}^* \in S$, joka toteuttaa ehdon $f(\boldsymbol{z}^*) \leq f(\boldsymbol{z})$ kaikilla $\boldsymbol{z} \in S$. Näin ollen globaali optimi antaa kohdefunktiolle parhaan arvon sallitussa alueessa. Kuitenkin vaikka optimointitehtävän yleisessä muodossa (1) kohdefunktiota minimoidaan, voidaan minimointitehtävä muuttaa helposti maksimointitehtäväksi ja päinvastoin. Tämä tapahtuu kertomalla kohdefunktio luvulla -1. Toinen tärkeä huomio on se, että joskus kohdefunktiolla ja sen minimoinnilla tai maksimoinnilla ei ole merkitystä. Tällaisessa tilanteessa jokainen rajoitteet toteuttava sallittu ratkaisu on itse asiassa optimointitehtävän paras ratkaisu. Tällöin kohdefunktio voidaan siis valita vapaasti ja on ihan sama, minimoidaanko vai maksimoidaanko sitä.

Optimointitehtävän (1) luokittelu pohjautuu päätösmuuttujavektorin $\boldsymbol{z}$ määrittelyjoukkoon sekä funktioiden f, g_i ja h_j luonteeseen. Mikäli Z_2 on tyhjä joukko, on

kyseessä *jatkuva optimointitehtävä*. Mikäli Z_1 on tyhjä joukko, on kyseessä *diskreetti optimointitehtävä*. Lisäksi diskreetti optimointitehtävä voidaan vielä luokitella *kokonaisluku-* tai *binäärioptimointitehtäväksi* muuttujien luonteen perusteella. Mikäli kaikki muuttujat ovat binäärisiä, on kyseessä binäärioptimointitehtävä. Vastaavasti kyseessä on kokonaislukuoptimointitehtävä, kun kaikki muuttujat saavat kokonaislukuarvoja. Kun kumpikaan edellä mainituista joukoista Z_1 tai Z_2 ei ole tyhjä, on kyseessä *sekalukuoptimointitehtävä*. Puolestaan, jos kaikki tehtävän funktiot f, g_i ja h_j ovat lineaarisia, on kyseessä *lineaarinen optimointitehtävä*. Jos yksikin funktio on epälineaarinen, on kyseessä *epälineaarinen optimointitehtävä*.

Tässä tutkielmassa käsiteltävät optimointitehtävät luokitellaan joukkoihin MILP (*Mixed-Integer Linear Programming*), MINLP (*Mixed-Integer Non-Linear Programming*) ja BLP (*Binary Linear Programming*). MILP-joukon tehtävät ovat lineaarisia sekalukuoptimointitehtäviä, joissa samanaikaisesti sallitaan sekä jatkuvat että diskreetit muuttujat. MINLP-joukon tehtävät on muuten vastaavia, mutta mukana pitää olla vähintään yksi epälineaarinen funktio (kohdefunktio tai rajoite). BLP-joukon tehtävät ovat lineaarisia ja sisältävät ainoastaan binäärisiä muuttujia.

Tärkeä työkalu jatkossa on epälineaaristen rajoitteiden linearisointi. Tässä tutkielmassa tarvitaan erityisesti itseisarvon sekä kahden binäärimuuttujan tulon linearisointia. Muotoa $|x_i - x_j| \geq a$, missä $x_i, x_j \in \mathbb{Z}$ ja $a > 0$, olevan itseisarvorajoitteen linearisoimiseksi on määriteltävä binäärinen päätösmuuttuja

$$y_i^j = \begin{cases} 1, & \text{jos } x_i - x_j \geq a \\ 0, & \text{jos } x_j - x_i \geq a. \end{cases}$$

Epälineaarinen epäyhtälö $|x_i - x_j| \geq a$ voidaan tällöin kirjoittaa kahtena lineaarisena epäyhtälönä muodossa

$$\begin{aligned} x_i - x_j &\geq a - M \cdot (1 - y_i^j) \\ x_j - x_i &\geq a - M \cdot y_i^j, \end{aligned}$$

missä M on riittävän suuri positiivinen reaaliluku. Puolestaan kahden binäärimuuttujan tulon sisältävät rajoitteet voidaan linearisoida seuraavalla tavalla. Olkoon rajoitteen sisältämä tulo muotoa $x_i \cdot x_j$, missä $x_i, x_j \in \{0, 1\}$. Tällöin rajoitteessa oleva tulo voidaan korvata muuttujalla s_i^j, kun malliin otetaan mukaan lineaariset lisärajoitteet

$$\begin{aligned} s_i^j &\geq 1 + (x_i + x_j - 2) \\ s_i^j &\leq \frac{1}{2}(x_i + x_j) \\ s_i^j &\in \{0, 1\}. \end{aligned}$$

Keskeinen käsite jatkossa on lisäksi ongelman *NP-täydellisyys* [16]. Tällä tarkoitetaan sellaista ongelmaa, jolle ei ole löydetty algoritmia, joka ratkaisisi kyseisen ongelman polynomisessa ajassa. Esimerkiksi klassinen optimointitehtävä, kauppamatkustajaongelma (*travelling salesman problem*) [14], on NP-täydellinen ongelma. Itse luokalla NP (*Nondeterministic Polynomial*) tarkoitetaan sellaisten ongelmien joukkoa, jotka ratkeavat polynomisessa ajassa epädeterministisellä Turingin koneella. NP-täydelliset ongelmat ovat tämän joukon vaikeimmat ongelmat. Puolestaan luokalla P (*Polynomial*) tarkoitetaan sellaisten ongelmien joukkoa, jotka voidaan ratkaista polynomisessa ajassa deterministisellä Turingin koneella. Mikäli jollekin NP-täydelliselle ongelmalle löydettäisiin deterministisellä Turingin koneella polynomiaikainen ratkaisu, tarkoittaisi tämä sitä, että polynomiaikainen ratkaisu löydettäisiin kaikille muillekin ongelmille luokassa NP. Tällöin olisi P=NP. Kyseistä väitettä ei kuitenkaan ole pystytty todistamaan.

4 Tiilitysongelmat

Tiilitysongelma (*tiling problem*) [24] on päätöksenteko-ongelma, joka pyrkii vastaamaan kysymykseen: onko määrätylle alueelle olemassa vähintään yksi kelvollinen tiilitys t annetulla äärellisellä tiilisarjalla T. Tässä äärellisellä tiilisarjalla T tarkoitetaan äärellistä joukkoa paloja, joista jokaista voi käyttää niin monta kertaa kuin halutaan. Tiilityksellä t puolestaan tarkoitetaan tapahtumaa, jossa tiilisarjan T paloja sijoitetaan tiettyyn (joko äärelliseen tai äärettömään) alueeseen niin, että alue peittyy kokonaan. Toisin sanoen tiilitysongelmassa tavoitteena on täyttää annettu alue käyttäen ainoastaan äärellistä määrää erilaisia paloja eli tiiliä. Tiilitysongelma, jossa on tavoitteena peittää äärellisellä tiilisarjalla ääretön alue, on todistettu ratkaisettomaksi [3, 33]. On siis avoin ongelma, voidaanko äärellisellä tiilisarjalla tiilittää ääretön alue. Äärellisille alueille tämä ongelma on NP-täydellinen [26]. Tiilitysongelmasta on olemassa myös erikoistapaus [23, 30], jossa jokaista tiiltä on käytettävä täsmälleen kerran. Tiilitysongelmasta käytetään myös nimeä domino-ongelma (*domino problem*). Tässä luvussa tarkastellaan erityisesti tiilitysongelman erikoistapausta, mutta myös esimerkkiä yleisestä tiilitysongelmasta.

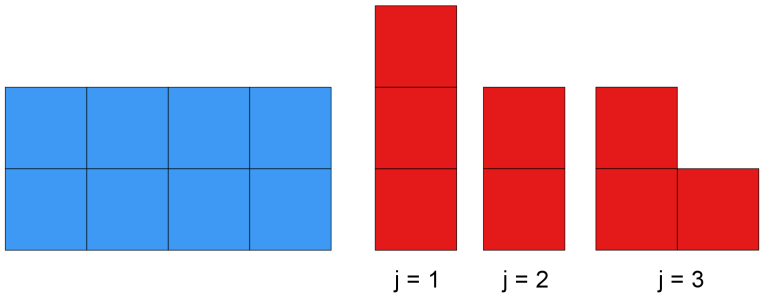
4.1 Yksinkertainen tiilitysongelma

Tässä aliluvussa esitellään eräs tiilitysongelman erikoistapaus, johon on otettu vaikutteita artikkelista [17]. Yksinkertaisessa tiilitysongelmassa pelilauta on peitettävä kolmella palalla, joiden yhteenlaskettu pinta-ala on yhtä suuri kuin pelilaudan pinta-ala. Toisin sanoen tiiliä on kolme kappaletta ja jokaista käytetään vain kerran. Mer-

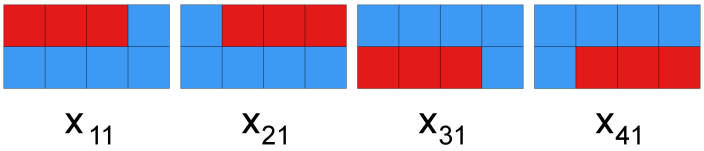
kitään kyseisiä tiiliä eli paloja indeksillä j. Kuvassa 2 on esitetty pelilauta sinisellä ja palat punaisella. Yksinkertaisen tiilitysongelman mallinnuksessa tarvitaan binäärisiä päätösmuuttujia x_{ij}, jotka kertovat, sijoitetaanko pala j pelilaudalle paikkaan i. Tätä varten määritellään palojen j paikoille i joukot I_j siten, että $I_1 = \{1, 2, 3, 4\}$, $I_2 = \{1, 2, \ldots, 10\}$ ja $I_3 = \{1, 2, \ldots, 12\}$. Olkoon

$$x_{ij} = \begin{cases} 1, & \text{jos pala } j \text{ sijoitetaan pelilaudalle paikkaan } i \\ 0, & \text{muuten,} \end{cases}$$

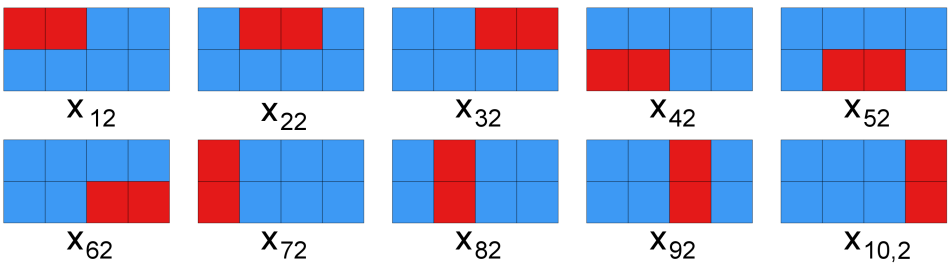
missä $j = 1, 2, 3$ ja $i \in I_j$. Kuvissa 3-5 on havainnollistettu palojen mahdollisia sijainteja pelilaudalla.



Kuva 2: Yksinkertaisen tiilitysongelman pelilauta sekä siihen sijoitettavat palat $j = 1, 2, 3$.

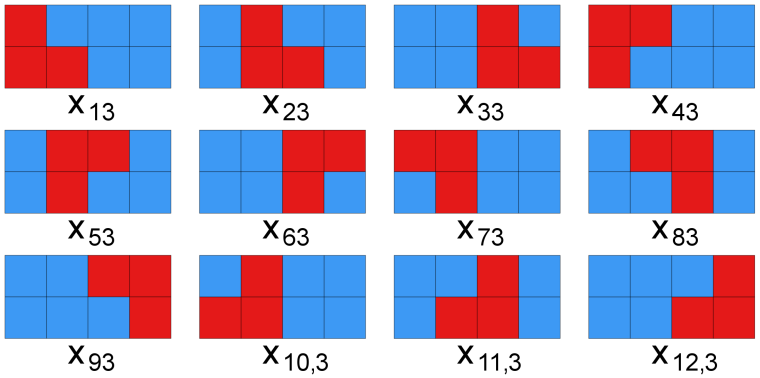


Kuva 3: Palan $j = 1$ mahdolliset sijainnit $i = 1, 2, 3, 4$ pelilaudalla.



Kuva 4: Palan $j = 2$ mahdolliset sijainnit $i = 1, 2, \ldots, 10$ pelilaudalla.

Muodostetaan seuraavaksi mallissa tarvittavat rajoitteet. Ensinnäkin on huolehdittava, että yhdessä pelilaudan ruudussa on vain yksi pala. Toisin sanoen mitään



Kuva 5: Palan $j = 3$ mahdolliset sijainnit $i = 1, 2, \ldots, 12$ pelilaudalla.

kahta palaa ei saa sijoittaa päällekkäin. Esimerkkinä tarkastellaan pelilaudan vasemman yläkulman ruutua. Kuvista 3-5 huomataan, että kun muuttujat x_{11}, x_{12}, x_{72}, x_{13}, x_{43} ja x_{73} saavat arvon yksi, peittyy tarkasteltava pelilaudan ruutu. On siis huolehdittava, että korkeintaan yksi näistä muuttujista saa arvon yksi, joten muodostetaan rajoite

$$x_{11} + x_{12} + x_{72} + x_{13} + x_{43} + x_{73} \leq 1.$$

Vastaava rajoite täytyy kirjoittaa myös muille pelilaudan ruuduille ja näin ollen kyseisiä rajoitteita on yhteensä kahdeksan kappaletta.

Tiilitysongelmassa vaaditaan, että jokainen pelilaudan ruuduista peittyy täsmälleen yhdellä palalla j. Edellä esitellyt rajoitteet takaavat kuitenkin vain sen, että jokainen ruutu peittyy korkeintaan yhdellä palalla. On siis mahdollista, että pelilaudan ruutuja jää "tyhjäksi", eikä niihin sijoiteta mitään palaa j. Ongelma kuitenkin poistuu, kun maksimoidaan kohdefunktiota

$$\sum_{j=1}^{3} \sum_{i \in I_j} x_{ij}.$$

Tällä nimittäin yhdessä edellä esiteltyjen rajoitteiden kanssa taataan, että jokainen pelilaudan ruutu peittyy täsmälleen yhdellä palalla j, jos ylipäätään on mahdollista löytää koko pelilaudan täyttävä tiilitys käyttäen jokaista palaa täsmälleen kerran.

Mallissa pitää varmistaa myös, että jokaista palaa todellakin käytetään tarkalleen kerran. Tätä varten tarvitaan rajoite

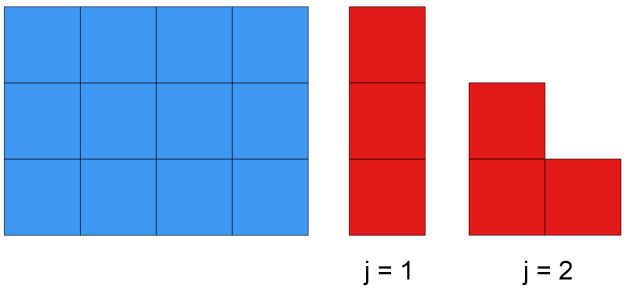
$$\sum_{i \in I_j} x_{ij} \leq 1, \quad j = 1, 2, 3,$$

joka pitää huolta siitä, että jokaista palaa käytetään enintään kerran. Yhdessä kohdefunktion maksimoinnin kanssa rajoite kuitenkin takaa, että jokainen pala sijoitetaan pelilaudalle tasan kerran, jos tämä vain on mahdollista.

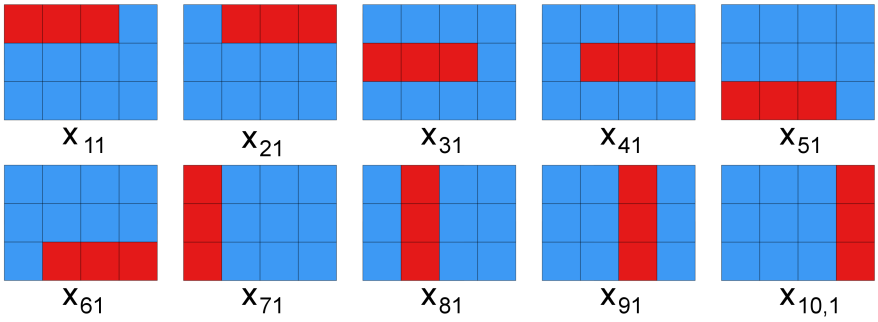
Keräämällä edellä esitetty yhteen voidaan yksinkertainen tiilitysongelma kirjoittaa lineaarisena binäärisenä optimointitehtävänä

$$
\begin{aligned}
\max_{x_{ij}} \quad & \sum_{j=1}^{3} \sum_{i \in I_j} x_{ij} \\
\text{s. t.} \quad & \sum_{i \in I_j} x_{ij} \leq 1, \quad j = 1, 2, 3 \\
& x_{11} + x_{12} + x_{72} + x_{13} + x_{43} + x_{73} \leq 1 \\
& x_{11} + x_{21} + x_{12} + x_{22} + x_{82} + x_{23} + x_{43} + x_{53} + x_{73} + x_{83} + x_{10,3} \leq 1 \\
& x_{11} + x_{21} + x_{22} + x_{32} + x_{92} + x_{33} + x_{53} + x_{63} + x_{83} + x_{93} + x_{11,3} \leq 1 \\
& x_{21} + x_{32} + x_{10,2} + x_{63} + x_{93} + x_{12,3} \leq 1 \\
& x_{31} + x_{42} + x_{72} + x_{13} + x_{43} + x_{10,3} \leq 1 \\
& x_{31} + x_{41} + x_{42} + x_{52} + x_{82} + x_{13} + x_{23} + x_{53} + x_{73} + x_{10,3} + x_{11,3} \leq 1 \\
& x_{31} + x_{41} + x_{52} + x_{62} + x_{92} + x_{23} + x_{33} + x_{63} + x_{83} + x_{11,3} + x_{12,3} \leq 1 \\
& x_{41} + x_{62} + x_{10,2} + x_{33} + x_{93} + x_{12,3} \leq 1 \\
& x_{ij} \in \{0, 1\}, \quad j = 1, 2, 3, \ i \in I_j.
\end{aligned}
$$

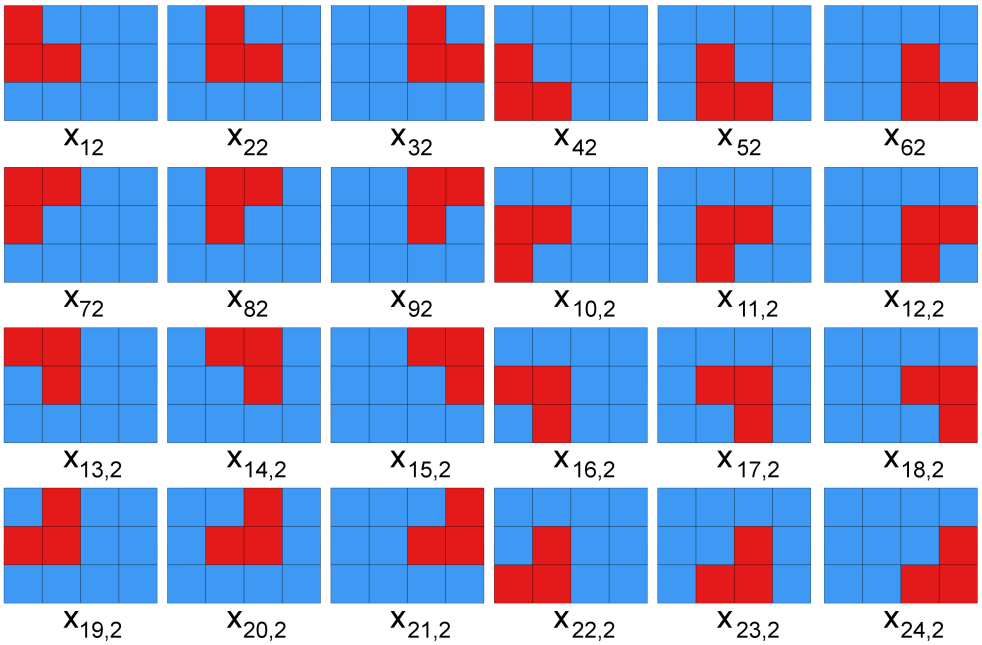
Lisäksi yksinkertainen tiilitysongelma voidaan mallintaa myös lineaarisena yhtälöryhmänä. Tällöin yllä esitellyn mallin epäyhtälöt korvataan yhtäsuuruuksilla, minkä seurauksena kohdefunktio voidaan poistaa, sillä mikä tahansa yhtälöryhmän toteuttava ratkaisu on yksinkertaisen tiilitysongelman ratkaisu.



Kuva 6: Yleisen tiilitysongelman pelilauta sekä sen tiilityksessä käytettävät palat $j = 1$ ja $j = 2$.



Kuva 7: Palan $j = 1$ mahdolliset sijainnit $i = 1, 2, \ldots, 10$ pelilaudalla.



Kuva 8: Palan $j = 2$ mahdolliset sijainnit $i = 1, 2, \ldots, 24$ pelilaudalla.

4.2 Yleinen tiilitysongelma

Tässä aliluvussa tarkastellaan yleistä tiilitysongelmaa [24], jossa jokaista tiilisarjan T tiiltä saa käyttää miten monta kertaa tahansa. Erityisesti esitetään malli kuvassa 6 olevalle tiilitysongelmalle, jossa 3×4-kokoinen pelilauta on tiilitettävä kahta erilaista tiiltä käyttäen. Muodostetaan malli käyttäen vastaavia merkintöjä kuin edellä esitellyssä yksinkertaisessa tiilitysongelmassa. Tällöin palojen $j = 1, 2$ paikkoja i esittävät joukot $I_1 = \{1, 2, \ldots, 10\}$ ja $I_2 = \{1, 2, \ldots, 24\}$. Lisäksi binäärinen päätösmuuttuja x_{ij} kertoo, sijoitetaanko pala j pelilaudalle sijaintiin i. Palojen mahdollisia sijainteja on havainnollistettu kuvissa 7 ja 8.

Yleisen tiilitysongelman malliksi saadaan lineaarinen binäärinen optimointiteh-

tävä

$$
\begin{aligned}
\max_{x_{ij}} \quad & \sum_{j=1}^{2} \sum_{i \in I_j} x_{ij} \\
\text{s. t.} \quad & x_{11} + x_{71} + x_{12} + x_{72} + x_{13,2} \leq 1 \\
& x_{11} + x_{21} + x_{81} + x_{22} + x_{72} + x_{82} + x_{13,2} + x_{14,2} + x_{19,2} \leq 1 \\
& x_{11} + x_{21} + x_{91} + x_{32} + x_{82} + x_{92} + x_{14,2} + x_{15,2} + x_{20,2} \leq 1 \\
& x_{21} + x_{10,1} + x_{92} + x_{15,2} + x_{21,2} \leq 1 \\
& x_{31} + x_{71} + x_{12} + x_{42} + x_{72} + x_{10,2} + x_{16,2} + x_{19,2} \leq 1 \\
& x_{31} + x_{41} + x_{81} + x_{12} + x_{22} + x_{52} + x_{82} + x_{10,2} + x_{11,2} + \\
& \qquad x_{13,2} + x_{16,2} + x_{17,2} + x_{19,2} + x_{20,2} + x_{22,2} \leq 1 \\
& x_{31} + x_{41} + x_{91} + x_{22} + x_{32} + x_{62} + x_{92} + x_{11,2} + x_{12,2} + \\
& \qquad x_{14,2} + x_{17,2} + x_{18,2} + x_{20,2} + x_{21,2} + x_{23,2} \leq 1 \\
& x_{41} + x_{10,1} + x_{32} + x_{12,2} + x_{15,2} + x_{18,2} + x_{21,2} + x_{24,2} \leq 1 \\
& x_{51} + x_{71} + x_{42} + x_{10,2} + x_{22,2} \leq 1 \\
& x_{51} + x_{61} + x_{81} + x_{42} + x_{52} + x_{11,2} + x_{16,2} + x_{22,2} + x_{23,2} \leq 1 \\
& x_{51} + x_{61} + x_{91} + x_{52} + x_{62} + x_{12,2} + x_{17,2} + x_{23,2} + x_{24,2} \leq 1 \\
& x_{61} + x_{10,1} + x_{62} + x_{18,2} + x_{24,2} \leq 1 \\
& x_{ij} \in \{0, 1\}, \quad j = 1, 2, \ i \in I_j,
\end{aligned}
$$

missä esimerkiksi ensimmäinen rajoite varmistaa, että pelilaudan vasemman yläkulman ruutu peittyy korkeintaan yhdellä palalla. Puolestaan kohdefunktion maksimointi pitää huolen siitä, että kyseinen ruutu peittyy täsmälleen yhdellä palalla. Yhdessä kaikki 12 rajoitetta varmistavat, että jokainen pelilaudan ruutu peittyy täsmälleen yhdellä tiilellä.

Yleisen tiilitysongelman mallista saadaan helposti muokattua malli tiilitysongelman erikoistapaukselle, jossa jokaista palaa tulee käyttää täsmälleen kerran. Tällöin tarvitaan rajoitteet

$$\sum_{i \in I_j} x_{ij} \leq 1, \quad j = 1, 2.$$

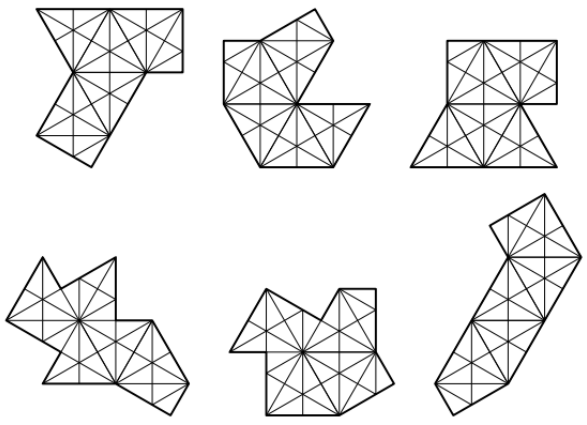
Vastaavalla tavalla voidaan muodostaa rajoitteet palojen muillekin käyttökerroille. Mikäli esimerkiksi paloja $j = 1, 2$ halutaan kumpaakin käyttää kahdesti, voidaan kirjoittaa rajoitteet

$$\sum_{i \in I_1} x_{i1} \leq 2 \quad \text{ja} \quad \sum_{i \in I_2} x_{i2} \leq 2.$$

4.3 Eternity I

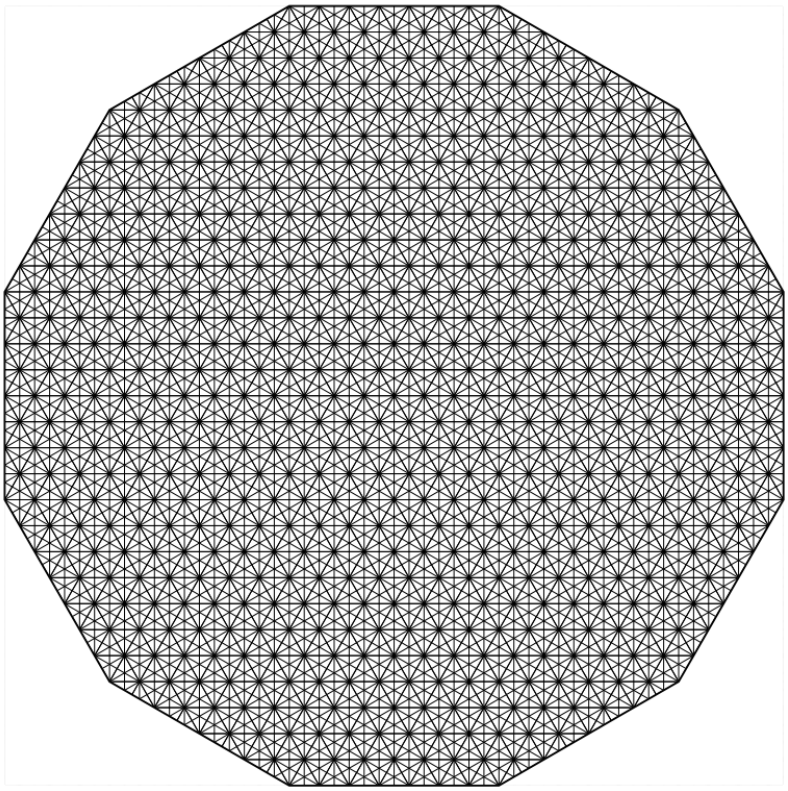
Eternity I on vaikea kombinatorinen tiilitysongelma, jossa tavoitteena on sijoittaa kaikki annetut 209 palaa pelilaudalle. Lisäksi jokaista palaa pitää käyttää tasan kerran eli kyseessä on yleisen tiilitysongelman erikoistapaus. Palapeli julkaistiin kesäkuussa 1999, ja sen on kehittänyt Christopher Monckton. Eternity I -palapelin ratkaisemisesta haastavan tekee se, että kaikki 209 palaa ovat monikulmioita, epäsymmetrisiä ja keskenään erilaisia. Edes palojen rotatointi ei tee mistään kahdesta palasta samanlaista. Muutama pala on esitetty kuvassa 9. Puolestaan pelilauta on melkein säännöllinen 12-kulmio ja palat on sijoitettava siten, että koko lauta peittyy. Pelilauta on esitetty kuvassa 10. Monckton kehitti palapelin sillä ajatuksella, että se on mahdoton ratkaista edes tietokoneella kokeilemalla. Tästä juontaakin palapelin nimi Eternity eli vapaasti suomennettuna ikuisuus.

Heti Eternity I -palapelin ilmestyttyä vuonna 1999 se sai osakseen paljon huomiota, sillä pelin ensimmäisenä oikein ratkaisseelle luvattiin miljoonan punnan palkkio. Lokakuussa vuonna 2000 palapelille esitettiinkin kaksi eri ratkaisua. Ensimmäisen löysivät Cambridgen matemaatikot Alex Selby sekä Oliver Riordan ja toisen shakin pelaaja Guenter Stertenbrink. Näistä ratkaisuista kumpikaan ei kuitenkaan vastannut Moncktonin julkistamatonta ratkaisua, jolle muutaman palan sijainti oli annettu vihjeenä. Mainittujen ratkaisujen jälkeen uusia ratkaisuja ei ole löydetty, eikä edelleenkään siis sellaista ratkaisua, joka sisältäisi vihjepalat oikeilla paikoilla.



Kuva 9: Kuusi kappaletta Eternity I -palapelin paloja. Kuva on otettu artikkelista [6].

Tarkastellaan seuraavaksi lähemmin sitä, miten Eternity I -palapeli voidaan mallintaa matemaattisella optimoinnilla. Esiteltävä malli ja siinä käytettävät merkinnät pohjautuvat artikkeliin [6]. Palapelin tavoitteena on siis täyttää alue R (pelilauta) käyttämällä $T = 209$ kappaletta keskenään erilaisia paloja. Näitä paloja merkitään



Kuva 10: Eternity I -palapelin pelilauta. Kuva on otettu artikkelista [6].

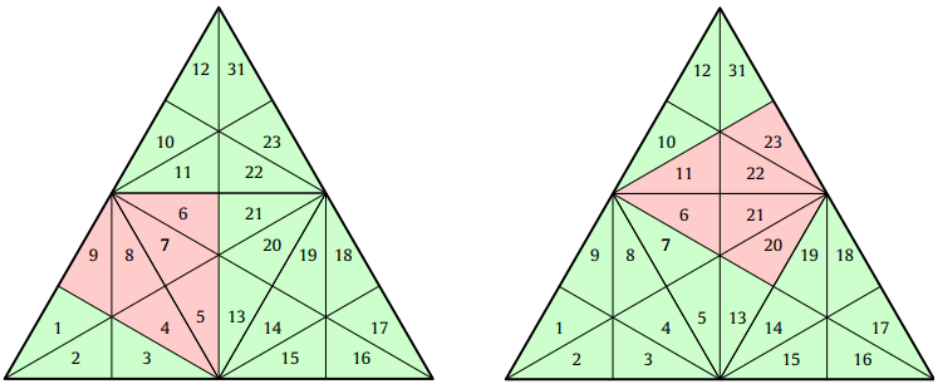
jatkossa indeksillä i, missä $i = 1, 2, \ldots, T$. Jokainen pala i koostuu 12 samankokoisesta suorakulmaisesta kolmiosta, joiden kulmat ovat 30°, 60° ja 90° (*polydrafter*). Toisaalta jokainen pala i koostuu 36 pelilaudan pienestä kolmiosta, joiden kokonaismäärää merkitään jatkossa parametrilla e. Tämä parametri on kiinnitettävä ennen ongelman ratkaisemista. Saadaan yhtälö $e = 36T$, sillä jokainen pelilaudan pieni kolmio pitää peittyä yhdellä paloista. Eternity I -palapelissä $e = 7524$, koska $T = 209$. Kaikki pelilaudan R pienet kolmiot voidaankin numeroida käyttäen lukuja $1, 2, \ldots, e$ ja näitä kolmioita merkitään jatkossa indeksillä k eli $k = 1, 2, \ldots, e$.

On hyvä tiedostaa, että palan sijoittaminen pelilaudalle R ei ole yksinkertainen tehtävä. Nimittäin palaa i voidaan kiertää eri asentoihin ja palalla on s_i erilaista mahdollista rotaatiota. Rotaatioita merkitään jatkossa indeksillä j. Lisäksi rotaatiossa j olevalla palalla i on w_{ij} erilaista mahdollista sijaintia z. Määritellään binäärivektorin $\boldsymbol{v}^{ijz}$ komponentit

$$v_k^{ijz} = \begin{cases} 1, & \text{jos pelilaudan kolmio } k \text{ peittyy palalla } i, \text{ joka on rotaatiossa } j \\ & \text{ja sijainnissa } z \\ 0, & \text{muuten,} \end{cases}$$

missä $i = 1, 2, \ldots, T$, $j = 1, 2, \ldots, s_i$, $z = 1, 2, \ldots, w_{ij}$ ja $k = 1, 2, \ldots, e$. Vektorin

$\boldsymbol{v}^{ijz}$ pituus on e. Seuraavaksi havainnollistetaan vektoria $\boldsymbol{v}^{ijz}$ kuvan 11 tilanteessa. Oletetaan, että kuvan pala on nimenomaan pala 1, jolloin $i = 1$. Lisäksi tapauksessa a) oletetaan, että kyseessä on rotaatio $j = 1$ ja sijainti $z = 1$. Näin ollen $v_k^{111} = 1$, kun $k \in \{4, 5, 6, 7, 8, 9\}$. Oletetaan sitten, että kuvan 11 tapauksessa b) on rotaatio $j = 2$ ja sijainti $z = 1$. Tällöin $v_k^{121} = 1$, kun $k \in \{6, 11, 20, 21, 22, 23\}$.



Kuva 11: Esimerkki Eternity I -palapelin palan rotatoitumisesta. Pala on merkitty punaisella. Luonnollisilla luvuilla on puolestaan numeroitu pienet pelilaudan kolmiot. Lisäksi tässä on kuvattu vain osa koko pelilaudasta, joten mukana on vain osa numeroista (puuttuu esimerkiksi 24–30). Kuva on otettu artikkelista [6].

Olkoon I_i joukko palan i binäärivektoreita $\boldsymbol{v}^{ijz}$. Joukkoon kuuluvat kaikki mahdolliset palan i rotaatiot j ja sijainnit z pelilaudalla R. Näin ollen

$$I_i = \{\boldsymbol{v}^{ijz} \mid j = 1, 2, \ldots, s_i,\ z = 1, 2, \ldots, w_{ij}\},\ \text{missä } i = 1, 2, \ldots, T.$$

Binäärivektorien $\boldsymbol{v}^{ijz}$ kokonaismäärä n summaa puolestaan yhteen kaikkien palojen i binäärivektorit. Siis $n = \sum_{i=1}^{T} \sum_{j=1}^{s_i} w_{ij}$. Erityisesti ennen tiilitysongelman rajoitteiden esittämistä täytyy muodostaa kaikki mahdolliset binäärivektorit $\boldsymbol{v}^{ijz}$ eli toisin sanoen jokaisen palan i joukko I_i. Tätä varten täytyy ensin määritellä jokaiselle palalle i kaikki mahdolliset rotaatiot ja sijainnit. Tämän jälkeen voidaan muodostaa binäärivektorit $\boldsymbol{v}^{ijz}$.

Tehtävän mallinnuksessa tarvitaan lisäksi binäärisiä päätösmuuttujia x_{ijz}, jotka kertovat, sijoitetaanko pala i pelilaudalle rotaatiossa j sijaintiin z. Olkoon siis

$$x_{ijz} = \begin{cases} 1, & \text{jos pala } i \text{ sijoitetaan pelilaudalle rotaatiossa } j \text{ sijaintiin } z \\ 0, & \text{muuten,} \end{cases}$$

missä $i = 1, 2, \ldots, T$, $j = 1, 2, \ldots, s_i$ ja $z = 1, 2, \ldots, w_{ij}$. Tiilitysongelman rajoitteet voidaankin nyt kirjoittaa yhtälöryhmänä

$$\sum_{i=1}^{t} \sum_{j=1}^{s_i} \sum_{z=1}^{w_{ij}} x_{ijz} \boldsymbol{v}^{ijz} = \mathbf{1}$$

$$\sum_{j=1}^{s_i} \sum_{z=1}^{w_{ij}} x_{ijz} = 1, \quad i = 1, 2, \ldots, T$$

$$x_{ijz} \in \{0, 1\}, \quad i = 1, 2, \ldots, T,\ j = 1, 2, \ldots, s_i,\ z = 1, 2, \ldots, w_{ij}.$$

Yllä vektorin $\mathbf{1}$ pituus on e ja jokainen sen komponenteista on ykkönen. Ensimmäinen tiilitysongelman rajoite vaatii, että jokainen pelilaudan R pieni kolmio peittyy täsmälleen yhdellä palalla. Toinen rajoite puolestaan takaa sen, että jokainen pala i rotatoidaan ja sijoitetaan pelilaudalle tarkalleen kerran. Optimointitehtävissä on aina vielä lisäksi mukana kohdefunktio, mutta nyt se voi olla mikä tahansa, esimerkiksi vakiofunktio, sillä jokainen rajoitteet toteuttava sallittu ratkaisu on tehtävän ratkaisu. Ei siis ole väliä minimoidaanko vai maksimoidaanko vapaasti valittavaa kohdefunktiota. Esitelty malli on lineaarinen binäärinen optimointitehtävä eli kyseessä on BLP-tehtävä.

Eternity I -palapeliä ratkaistaessa on yllä esitellyssä mallissa 7733 rajoitetta sekä 1 372 296 muuttujaa. Lisäksi lähteessä [6] on todettu, että kolmen kuukauden laskemisenkaan jälkeen CPLEX-ratkaisija ei onnistunut löytämään Eternity I -palapelille sallittua ratkaisua. Kyseisen palapelin ratkaisseet Selby ja Riordan löysivät oman ratkaisunsa takautuvalla hakumenetelmällä (*backtracking method*). Kyseessä on rekursiivinen hakualgoritmi, joka käy läpi kaikki optimointitehtävään sopivat ratkaisut yksitellen. Takautuva hakumenetelmä on siis eräänlainen brute force -menetelmä. Hakupuun jokaisessa solmussa takautuva hakumenetelmä poistaa sellaiset ratkaisuehdokkaat, joita ei varmuudella pystytä täydentämään tehtävän sallituksi ratkaisuksi. Tällöin hakumenetelmä etenee rekursiivisesti tarkistamaan vain sellaisia ratkaisuja, jotka ylipäätään ovat sallittuja. Selby ja Riordan käyttivät myös tilastollisia tekniikoita muodoltaan kaikkein haastavimpien palojen tunnistamiseksi, jotta kyseiset palat voitiin sijoittaa pelilaudalle aikaisessa vaiheessa. Käytetty ratkaisumenetelmä suunniteltiin nimenomaan Eternity I -palapeliä varten, eikä se näin ollen sovellu muihin vaikeisiin tiilitysongelmiin.

5 Reunasovitusongelmat

Reunasovitusongelmat (*edge-matching problems*) [11, 25] ovat palapelejä, jossa tavoitteena on sijoittaa annetut palat pelilaudalle siten, että vierekkäisten palojen

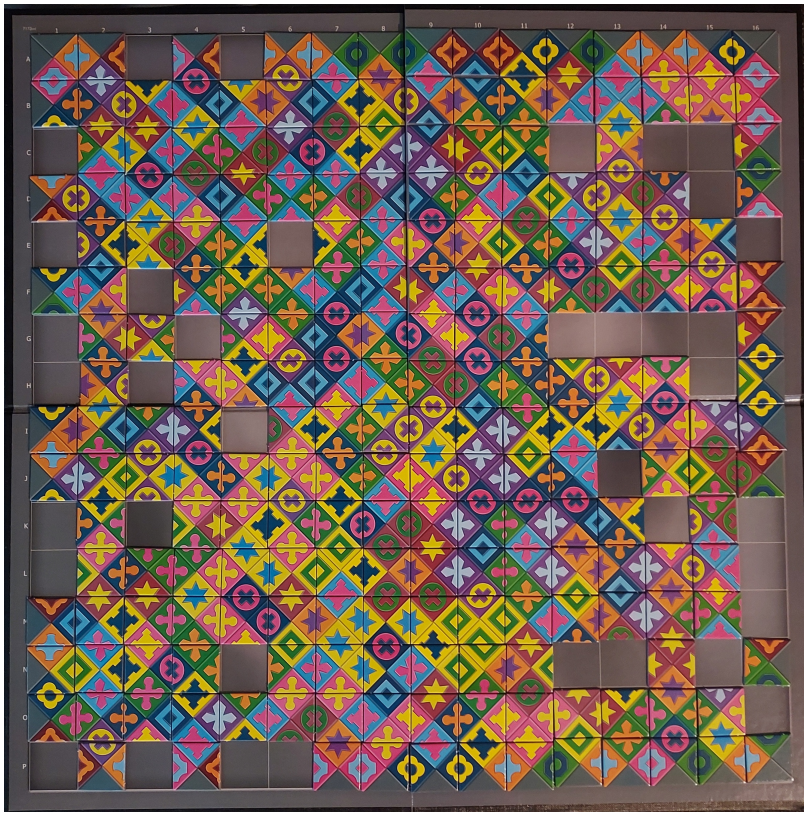
vastakkaiset reunat sopivat yhteen. Tällä tarkoitetaan sitä, että esimerkiksi aina saman väriset reunat ovat vastakkain. On olemassa myös reunasovitusongelmia, joissa reunojen on muodostettava esimerkiksi tietty kuvio. Reunasovitusongelmien palat ovat kaikki saman muotoisia. Usein palat ovat neliöitä. Haastavia reunasovitusongelmista tekee se, että yksi pala voi sopia moneen paikkaan pelilaudalla. Kuitenkin vaikka kaksi palaa sopisivat vierekkäin, ei se takaa niiden olevan lopullisessa ratkaisussa vierekkäin. Jatkossa tarkastellaan sellaisia reunasovitusongelmia, joissa vierekkäisten palojen vastakkaisten reunojen on oltava samanväriset.

5.1 Eternity II

Eternity II on reunasovitusongelma, jossa tavoitteena on sijoittaa 256 neliönmuotoista palaa 16×16-kokoiselle pelilaudalle siten, että vierekkäisten palojen reunat sopivat yhteen. Palapelissä on 22 eri kuvion ja värin kombinaatiota eli niin sanotusti väriä, sekä pelilaudan reunojen harmaa väri. Eternity II on julkaistu 28.7.2007 ja sen on kehittänyt, kuten Eternity I -palapelinkin, Christopher Monckton. Palapeli on muodostettu siten, että sitä on lähes mahdotonta ratkaista algoritmilla, joka käy kaikki palojen mahdolliset kombinaatiot läpi. Toisin sanoen kyseessä on NP-täydellinen ongelma. Lisäksi Eternity II -palapelin ratkaisemisesta on järjestetty kilpailu, jossa palapelin ensimmäisenä ratkaisseelle luvattiin kahden miljoonan dollarin palkintosumma. Kilpailu päättyi 31.12.2010 ilman voittajaa. Tämänkään jälkeen kukaan ei ole onnistunut ratkaisemaan palapeliä kokonaan. Moncktonkaan ei ole julkistanut omaa ratkaisuaan, joten Eternity II pysyy edelleen avoimena ongelmana. Kuva 12 havainnollistaa Eternity II -palapeliä.

Palapelissä on siis kaikkiaan 256 neliönmuotoista palaa, ja jokaisessa palassa on neljä kolmiota siten, että jokaisessa palan reunassa on yksi väri. Reuna- ja kulmapalat on merkitty harmailla kolmioilla ja harmaiden reunojen on oltava pelilaudan ulkoreunassa. Kulmapaloja on neljä ja reunapaloja 56 kappaletta, jolloin muita paloja (mikään sivu ei ole harmaa) on $14^2 = 196$ kappaletta. Lisäksi jokaisessa palassa on uniikki värien järjestys. Toisin sanoen mitkään palat eivät edes rotatoituna ole samanlaiset. Näin ollen on olemassa yhteensä $256 \cdot 4 = 1024$ mahdollisuutta yksittäisen ruudun täyttämiseen. Tässä kerroin neljä merkitsee neljää eri mahdollista palan asentoa eli rotaatiota. Eternity II sisältää myös yhden valmiiksi sijoitetun palan. Kyseinen pala sijaitsee laudan keskiosassa ja se on sijoitettava annettuun kohtaan annetussa rotaatiossa.

Tarkastellaan seuraavaksi $n \times n$-kokoista Eternity II -palapeliä, ja mallinnetaan kyseinen ongelma optimointitehtävänä. Erityisesti esitellään kaksi vaihtoehtoista mallinnustapaa. Molemmissa malleissa käytetään samoja merkintöjä indeksien osal-

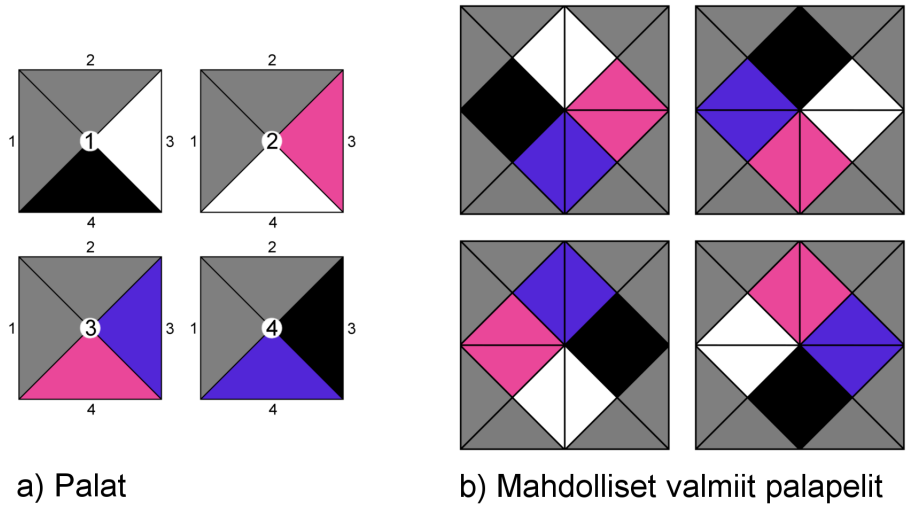


Kuva 12: Keskeneräinen Eternity II -palapeli.

ta, jotta mallien vertailu olisi mahdollisimman helppoa. Käydään seuraavaksi läpi kyseiset indeksit sekä muut käytettävät merkinnät. Tämän jälkeen esitellään kaksi optimointimallia Eternity II -palapelille.

Aluksi jokainen pala pitää numeroida. Olkoon palojen joukko $K = \{1, 2, 3, \ldots, n^2 - 1, n^2\}$, jolloin yksittäinen pala $k \in K$. Täytyy myös päättää, mitkä palojen rotaatiot ovat niin kutsuttuja "perusasentoja". Palojen ollessa perusasennoissa numeroidaan reunat parametrilla $l = 1, 2, 3, 4$. Jatkossa $l = 1$ on palan vasemmanpuoleinen reuna, $l = 2$ yläreuna, $l = 3$ oikeanpuoleinen reuna ja $l = 4$ alareuna. On huomattava, että vaikka palaa rotatoidaan, on edelleen esimerkiksi palan vasemmanpuoleisen reunan numerona 1. Palojen rotatointi ei siis vaikuta reunojen numeroihin, vaan ainoastaan siihen, mikä väri tulee olemaan palan milläkin sivulla. Merkitään paloissa esiintyviä värejä joukon $P = \{1, 2, 3, ..., p_{max}\}$ alkioilla, missä p_{max} on värien lukumäärä. Lisäksi joukko $P_0 = P \cup \{0\}$ koostuu esitellyistä väreistä ja harmaasta reunasta eli väristä 0. Puolestaan pelilaudan rivejä merkitään indeksillä i ja sarakkeita indeksillä j. Koska nyt tarkastelussa on $n \times n$-kokoinen Eternity II, niin $i = 1, 2, \ldots, n$ ja $j = 1, 2, \ldots, n$. Pelilaudan rivin i ja sarakkeen j ruudulle käytetään jatkossa merkintää (i, j). Lisäksi jatkoa varten määritellään joukot $S_n = \{1, 2, \ldots, n\}$ ja $S_{n-1} = \{1, 2, \ldots, n - 1\}$.

Tarkastellaan esimerkkinä merkintöjä kuvan 13 2×2-kokoisessa Eternity II -palapelissä, jossa on reunan harmaan lisäksi neljä väriä. Palojen numerot ovat $k = 1, 2, 3, 4$ ja ne vastaavat kuvan 13 a)-kohdan palojen numeroita. Samassa kuvassa on esitetty palojen perusasennot sekä reunojen numerointi parametrilla $l = 1, 2, 3, 4$. Puolestaan palojen reunojen värejä voidaan merkitä esimerkiksi seuraavasti: 0 (harmaa eli reuna), 1 (valkoinen), 2 (musta), 3 (pinkki), 4 (liila). Lisäksi 2×2-kokoisessa Eternity II -palapelissä $i = 1, 2$ ja $j = 1, 2$.



Kuva 13: Neljän palan Eternity II -palapeli.

5.1.1 Mallinnustapa 1

Ensimmäisen mallin esitystapa pohjautuu artikkeliin [34]. Tässä indeksillä $m = 0, 1, 2, 3$ merkitään palan rotatoitumista. Arvo $m = 0$ merkitsee sitä, että pala on perusasennossa, eikä sitä ole kierretty ollenkaan. Puolestaan arvo $m = 1$ tarkoittaa sitä, että palaa on kierretty $1 \cdot 90°$ kellon suuntaan. Näin ollen $m = 2$ vastaa 180 asteen kiertoa ja $m = 3$ puolestaan 270 asteen kiertoa kellon suuntaan. Lisäksi indeksillä $p \in P_0$ merkitään edellä numeroituja värejä.

Mallia varten tarvitaan parametrit

$$c_{lkmp} = \begin{cases} 1, & \text{jos palan } k \text{ rotaatiolla } m \text{ reunan } l \text{ väri on } p \\ 0, & \text{muuten,} \end{cases}$$

missä $l = 1, 2, 3, 4$, $k \in K$, $m = 0, 1, 2, 3$ ja $p \in P_0$. Lisäksi tarvitaan päätösmuuttujat

$$x_{kijm} = \begin{cases} 1, & \text{jos pala } k \text{ on ruudussa } (i, j) \text{ rotaatiolla } m \\ 0, & \text{muuten,} \end{cases}$$

missä $k \in K$, $i \in S_n$, $j \in S_n$ ja $m = 0, 1, 2, 3$,

$$h_{ij} = \begin{cases} 1, & \text{jos ruudun } (i,j) \text{ oikea reuna ei sovi} \\ 0, & \text{muuten,} \end{cases}$$

missä $i \in S_n$ ja $j \in S_{n-1}$ sekä

$$v_{ij} = \begin{cases} 1, & \text{jos ruudun } (i,j) \text{ alareuna ei sovi} \\ 0, & \text{muuten,} \end{cases}$$

missä $i \in S_{n-1}$ ja $j \in S_n$. Tarkastellaan seuraavaksi lähemmin mallissa tarvittavaa kohdefunktiota sekä rajoitteita.

Minimoitava kohdefunktio on muotoa

$$\sum_{i \in S_n} \sum_{j \in S_{n-1}} h_{ij} + \sum_{i \in S_{n-1}} \sum_{j \in S_n} v_{ij}.$$

Ensimmäisessä summassa käydään läpi kaikki paitsi pelilaudan viimeisen sarakkeen ruudut ja näin ollen nyt minimoidaan niiden ruutujen määrää, joiden oikea reuna ($l = 3$) ei ole sopiva viereisen ruudun vasemman reunan kanssa. Jälkimmäisessä summassa käydään läpi kaikki paitsi pelilaudan viimeisen rivin ruudut ja tästä syystä minimoidaan niiden ruutujen määrää, joiden alareuna ($l = 4$) ei ole sopiva alapuolella olevan ruudun yläreunan kanssa. Kaiken kaikkiaan kohdefunktion minimoinnilla tavoitellaan tilannetta, jossa mahdollisimman vähän palojen reunoja rikkoo värien muodostaman kuvion. Ideaalitilanteessa kohdefunktion arvo on nolla, sillä tällöin palapeli on ratkaistu.

Esitellään seuraavaksi tarvittavat rajoitteet. Rajoite

$$\sum_{i \in S_n} \sum_{j \in S_n} \sum_{m=0}^{3} x_{kijm} = 1, \quad k \in K$$

pitää huolen siitä, että jokainen pala sijoitetaan täsmälleen yhteen ruutuun täsmälleen yhdessä rotaatiossa. Summa käy ensinnäkin läpi kaikki paikat ja rotaatiot palalle k. Koska rajoitteen oikean puolen arvon on oltava tasan yksi, on summattavista binäärimuuttujista myös tarkalleen yhden oltava yksi. Siis pala k esiintyy tasan yhdessä ruudussa (i, j) ja yhdessä rotaatiossa m, kuten halutaankin. Rajoite

$$\sum_{k \in K} \sum_{m=0}^{3} x_{kijm} = 1, \quad i \in S_n,\ j \in S_n$$

puolestaan vaatii, että täsmälleen yksi pala yhdessä rotaatiossa sijoitetaan ruutuun (i, j). Vasemman puolen summa käy läpi kaikki palat ja kaikki palojen rotaatiot

ja varmistaa, että tiettyyn ruutuun (i, j) valitaan täsmälleen yksi pala yhdessä rotaatiossa. Nämä kaksi esiteltyä rajoitetta saattavat vaikuttaa vaativan samaa asiaa. Kuitenkin niiden ero on seuraava. Ensimmäinen rajoite vaatii, että jokainen pala sijoitetaan yhteen ruutuun. Toinen rajoite vaatii, että jokaisessa ruudussa on täsmälleen yksi pala. Erona on siis se, että ensimmäinen rajoite ei estä useamman palan sijoitusta samaan paikkaan. Jälkimmäinen rajoite ei puolestaan estä sijoittamasta yhtä tiettyä palaa useaan eri ruutuun.

Seuraavaksi tarvitaan rajoitteet, jotka varmistavat sen, että päätösmuuttuja h_{ij} saa arvon 1, jos ruuduissa (i, j) ja $(i, j+1)$ olevien palojen vastakkaiset reunat eivät ole yhteensopivat. Rajoitteet

$$\sum_{k\in K}\sum_{m=0}^{3} c_{3kmp}x_{kijm} - \sum_{k\in K}\sum_{m=0}^{3} c_{1kmp}x_{ki,j+1,m} \leq h_{ij}, \quad i \in S_n,\ j \in S_{n-1},\ p \in P$$

$$-\sum_{k\in K}\sum_{m=0}^{3} c_{3kmp}x_{kijm} + \sum_{k\in K}\sum_{m=0}^{3} c_{1kmp}x_{ki,j+1,m} \leq h_{ij}, \quad i \in S_n,\ j \in S_{n-1},\ p \in P$$

takaavat tämän. Tässä siis tarkastellaan ruudussa (i, j) olevan palan reunaa 3 (oikea reuna) ja ruudussa $(i, j + 1)$ olevan palan reunaa 1 (vasen reuna). Puolestaan rajoitteet

$$\sum_{k\in K}\sum_{m=0}^{3} c_{4kmp}x_{kijm} - \sum_{k\in K}\sum_{m=0}^{3} c_{2kmp}x_{k,i+1,jm} \leq v_{ij}, \quad i \in S_{n-1},\ j \in S_n,\ p \in P$$

$$-\sum_{k\in K}\sum_{m=0}^{3} c_{4kmp}x_{kijm} + \sum_{k\in K}\sum_{m=0}^{3} c_{2kmp}x_{k,i+1,jm} \leq v_{ij}, \quad i \in S_{n-1},\ j \in S_n,\ p \in P$$

takaavat sen, että päätösmuuttuja v_{ij} saa arvon 1 silloin, kun ruuduissa (i, j) ja $(i+1, j)$ olevien palojen vastakkaiset reunat eivät ole yhteensopivat. Ruudussa (i, j) olevalle palalle tämä tarkoittaa reunaa 4 (alareuna) ja ruudussa $(i + 1, j)$ olevalle palalle reunaa 2 (yläreuna).

Lopuksi tarvitaan vielä rajoitteet, jotka takaavat sen, että palojen harmaat reunat $(p = 0)$ ovat pelilaudan reunoilla. Tällaiset rajoitteet ovat

$$\sum_{k\in K}\sum_{m=0}^{3} c_{2km0}x_{k1jm} = 1, \quad j \in S_n$$

$$\sum_{k\in K}\sum_{m=0}^{3} c_{4km0}x_{knjm} = 1, \quad j \in S_n$$

$$\sum_{k\in K}\sum_{m=0}^{3} c_{1km0}x_{ki1m} = 1, \quad i \in S_n$$

ja

$$\sum_{k \in K} \sum_{m=0}^{3} c_{3km0} x_{kinm} = 1, \quad i \in S_n.$$

Ensimmäinen rajoite vaatii, että pelilaudan ylimmällä rivillä olevien palojen yläreunan ($l = 2$) on oltava harmaa. Toinen rajoite puolestaan vaatii, että pelilaudan alarivillä olevien palojen alareunan ($l = 4$) on oltava harmaa. Kolmannessa rajoitteessa tarkistetaan vasemmanpuoleisimman sarakkeen palojen vasemmat reunat ($l = 1$) ja neljännessä rajoitteessa oikeanpuoleisimman sarakkeen palojen oikeat reunat ($l = 3$). Näin ollen rajoitteet yhdessä tarkistavat sen, että palapelin ulkoreunat ovat harmaat.

Kun yhdistetään kohdefunktio ja kaikki esitellyt rajoitteet, saadaan lineaarinen

binäärinen optimointitehtävä

$$
\begin{aligned}
\min_{x_{kijm},h_{ij},v_{ij}} \quad & \sum_{i\in S_n}\sum_{j\in S_{n-1}} h_{ij} + \sum_{i\in S_{n-1}}\sum_{j\in S_n} v_{ij} \\
\text{s.t.} \quad & \sum_{i\in S_n}\sum_{j\in S_n}\sum_{m=0}^{3} x_{kijm} = 1, \quad k \in K \\
& \sum_{k\in K}\sum_{m=0}^{3} x_{kijm} = 1, \quad i \in S_n,\ j \in S_n \qquad (2) \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{3kmp}x_{kijm} - \sum_{k\in K}\sum_{m=0}^{3} c_{1kmp}x_{ki,j+1,m} \le h_{ij}, \quad i \in S_n,\ j \in S_{n-1},\ p \in P \\
& -\sum_{k\in K}\sum_{m=0}^{3} c_{3kmp}x_{kijm} + \sum_{k\in K}\sum_{m=0}^{3} c_{1kmp}x_{ki,j+1,m} \le h_{ij}, \quad i \in S_n,\ j \in S_{n-1},\ p \in P \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{4kmp}x_{kijm} - \sum_{k\in K}\sum_{m=0}^{3} c_{2kmp}x_{k,i+1,jm} \le v_{ij}, \quad i \in S_{n-1},\ j \in S_n,\ p \in P \\
& -\sum_{k\in K}\sum_{m=0}^{3} c_{4kmp}x_{kijm} + \sum_{k\in K}\sum_{m=0}^{3} c_{2kmp}x_{k,i+1,jm} \le v_{ij}, \quad i \in S_{n-1},\ j \in S_n,\ p \in P \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{2km0}x_{k1jm} = 1, \quad j \in S_n \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{4km0}x_{knjm} = 1, \quad j \in S_n \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{1km0}x_{ki1m} = 1, \quad i \in S_n \\
& \sum_{k\in K}\sum_{m=0}^{3} c_{3km0}x_{kinm} = 1, \quad i \in S_n \\
& x_{kijm} \in \{0,1\}, \quad k \in K,\ i \in S_n,\ j \in S_n,\ m = 0,1,2,3 \\
& h_{ij} \in \{0,1\}, \quad i \in S_n,\ j \in S_{n-1} \\
& v_{ij} \in \{0,1\}, \quad i \in S_{n-1},\ j \in S_n.
\end{aligned}
$$

5.1.2 Mallinnustapa 2

Seuraavaksi esitellään toinen malli. Sen merkinnät ja esitystapa pohjautuvat monisteeseen [13]. Olkoot mallissa tarvittavat parametrit

$$c_{kl} = \text{palan } k \text{ reunan } l \text{ väri perusasennossa,}$$

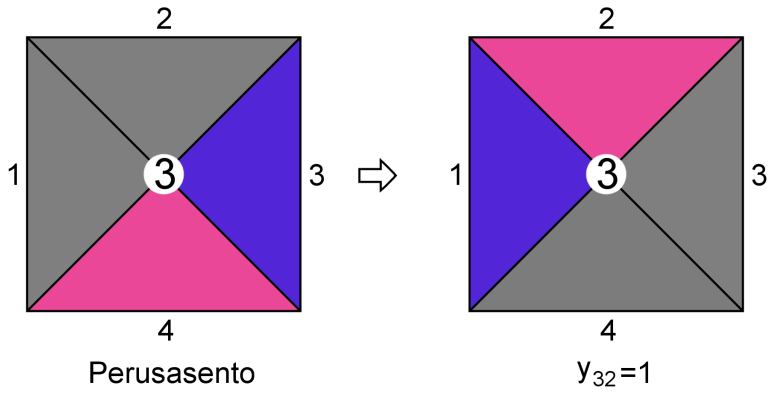
missä $k \in K$ ja $l = 1, 2, 3, 4$. Tässä $c_{kl} \in P_0$ eli värit on numeroitu edellä kerrotulla tavalla. Lisäksi tarvittavat päätösmuuttujat ovat

$$x_{ijk} = \begin{cases} 1, & \text{jos pala } k \text{ on ruudussa } (i, j) \\ 0, & \text{muuten,} \end{cases}$$

$$y_{km} = \begin{cases} 1, & \text{jos palaa } k \text{ on kierretty kellon suuntaan } m \cdot 90° \\ 0, & \text{jos pala } k \text{ on perusasennossa,} \end{cases}$$

$$r_{kl} \in [0, p_{max}], \quad \text{palan } k \text{ reunan } l \text{ väri, kun pala on paikallaan,}$$

missä $i \in S_n$, $j \in S_n$, $k \in K$, $m = 1, 2, 3$ ja $l = 1, 2, 3, 4$. Kuva 14 havainnollistaa päätösmuuttujaa y_{km} tilanteessa, jossa $k = 3$ ja $m = 2$. Erityisesti siis valinnalla $y_{32} = 1$ palaa 3 kierretään $2 \cdot 90° = 180°$.



Kuva 14: Eternity II -palapelin palan rotatoituminen.

Kuten edellä on jo todettu, tavoitteena on sijoittaa kaikki palapelin palat paikalleen. Tämä saavutetaan maksimoimalla kohdefunktiota

$$\sum_{i \in S_n} \sum_{j \in S_n} \sum_{k \in K} x_{ijk},$$

joka kertoo pelialustaan asetettujen palojen lukumäärän.

Tarkastellaan seuraavaksi optimointitehtävässä tarvittavia rajoitteita. Rajoite

$$\sum_{i \in S_n} \sum_{j \in S_n} x_{ijk} \leq 1, \quad k \in K \tag{3}$$

takaa sen, että pala k on korkeintaan yhdessä ruudussa (i, j). Lisäksi rajoite

$$\sum_{k \in K} x_{ijk} \leq 1, \quad i \in S_n, \ j \in S_n$$

pitää huolen siitä, että ruutuun (i, j) sijoitetaan korkeintaan yksi pala. On hyvä huomata, että kummassakin rajoitteessa esintyvä summa voi olla arvoltaan nolla tai yksi. Kuitenkin kohdefunktion maksimointi takaa, että molempien summien arvo on yksi, jos palapelille on ratkaisu olemassa. Tällöin siis jokainen pala k on täsmälleen yhdessä ruudussa. Samoin jokaiseen ruutuun (i, j) sijoitetaan täsmälleen yksi pala.

Jokainen pala k voi luonnollisesti olla vain yhdessä rotaatiossa. Tämän takaa rajoite

$$y_{k1} + y_{k2} + y_{k3} \leq 1, \quad k \in K.$$

Jos siis kaikki muuttujat y_{km} ovat nollia, on pala k perusasennossa. Lisäksi reunapalojen on sijoituttava pelilaudan reunoille siten, että palojen harmaat reunat ovat laudan ulkoreunalla. Nämä ehdot toteutuvat rajoitteilla

$$\begin{aligned}
r_{k1} &\leq M \left(1 - \sum_{i \in S_n} x_{i1k}\right), \quad k \in K \\
r_{k2} &\leq M \left(1 - \sum_{j \in S_n} x_{1jk}\right), \quad k \in K \\
r_{k3} &\leq M \left(1 - \sum_{i \in S_n} x_{ink}\right), \quad k \in K \\
r_{k4} &\leq M \left(1 - \sum_{j \in S_n} x_{njk}\right), \quad k \in K,
\end{aligned} \tag{4}$$

kun $M > 0$ on riittävän suuri vakio. Voidaan valita esimerkiksi $M = p_{max}$, joka on pienin sallittu luku tehtävälle, jossa suurin värin arvo on $p_{max} \in P_0$. Esitellyissä neljässä rajoitteessa käydään kussakin läpi yksi tilanteista, jossa pala k on sijoitettu jollekin pelilaudan reunoista, jotta voidaan taata oikean sivun palasta olevan harmaa. Tarkastellaan esimerkkinä viimeistä rajoitetta tilanteessa $k = 1$. Jos kyseinen pala sijoitetaan ruutuun $(n, 1)$, niin $x_{n11} = 1$ ja $x_{nj1} = 0$, kaikilla $j \in S_n \setminus \{1\}$. Tällöin rajoite saa muodon $r_{14} \leq 0$, jolloin palan 1 alareunan (reuna $l = 4$) värin on oltava harmaa (väri 0). Vastaavasti minkä tahansa pelilaudan alareunassa olevan palan alareuna on harmaa, kuten kuuluukin. Mikäli pala 1 sijoitettaisiin esimerkiksi ruutuun $(1, 1)$, saisi viimeinen rajoite muodon $r_{14} \leq M$. Tällöin rajoite toteutuisi aina, eikä siis palan 1 alareunan (reuna $l = 4$) väriä määrää kyseinen rajoite. Kaiken kaikkiaan nämä rajoitteet määräävät ainoastaan palapelin ulkoreunat harmaiksi.

Jotta rotatoituneen palan k reunojen värit saadaan määritettyä, tarvitaan ra-

joitteet

$$
\begin{aligned}
r_{k1} &= c_{k1}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k4}y_{k1} + c_{k3}y_{k2} + c_{k2}y_{k3}, \quad k \in K \\
r_{k2} &= c_{k2}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k1}y_{k1} + c_{k4}y_{k2} + c_{k3}y_{k3}, \quad k \in K \\
r_{k3} &= c_{k3}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k2}y_{k1} + c_{k1}y_{k2} + c_{k4}y_{k3}, \quad k \in K \\
r_{k4} &= c_{k4}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k3}y_{k1} + c_{k2}y_{k2} + c_{k1}y_{k3}, \quad k \in K.
\end{aligned}
$$

Tarkastellaan esimerkkinä kuvan 13 a)-kohdan palaa 1 eli tilannetta $k = 1$. Nyt $c_{11} = 0,\ c_{12} = 0,\ c_{13} = 1$ ja $c_{14} = 2$ sekä oletetaan, että palaa 1 kierretään 90°. Tällöin $y_{11} = 1,\ y_{12} = 0$ ja $y_{13} = 0$. Ensimmäinen rajoite saadaan nyt muotoon

$$
\begin{aligned}
r_{11} &= c_{11}(1 - y_{11} - y_{12} - y_{13}) + c_{14}y_{11} + c_{13}y_{12} + c_{12}y_{13} \\
&= 0 \cdot (1 - 1 - 0 - 0) + 2 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 = 2,
\end{aligned}
$$

joten 90 astetta rotatoidun palan 1 vasemmanpuoleisen reunan (reuna $l = 1$) väri on musta.

Lopuksi tarvitaan vielä rajoitteet, jotka takaavat sen, että vierekkäisten palojen vastakkain olevat reunat ovat samanväriset. Rajoitteet

$$
|r_{k3} - r_{t1}| \leq M \left(1 - \sum_{i \in S_n} \sum_{j \in S_{n-1}} x_{ijk} x_{i,j+1,t} \right), \quad k, t \in K,\ k \neq t \tag{5}
$$

$$
|r_{k4} - r_{t2}| \leq M \left(1 - \sum_{i \in S_{n-1}} \sum_{j \in S_n} x_{ijk} x_{i+1,jt} \right), \quad k, t \in K,\ k \neq t \tag{6}
$$

varmistavat tämän, kun $M > 0$ on riittävän suuri vakio. Koska nyt suurin värin arvo on p_{max}, on oltava $M \geq p_{max}$. Rajoitteessa (5) tarkastellaan vaakasuunnassa vastakkain olevia reunoja. Rajoite (6) puolestaan käsittelee pystysuunnassa vastakkain olevia reunoja. Näissä rajoitteissa oikealla puolella kaarisulkujen sisällä olevat lausekkeet saavat aina arvoikseen 0 tai 1. Kaarisulkulausekkeen arvo 0 tarkoittaa sitä, että palat k ja t sijaitsevat vierekkäin. Puolestaan kaarisulkulausekkeen arvo 1 tarkoittaa sitä, että palat k ja t eivät sijaitse vierekkäin. Jos kiinnitetyillä arvoilla k ja t rajoitteen (5) kaarisulkujen lauseke saa arvon 0, saa rajoitteen (6) kaarisulkujen lauseke arvon 1. Vastaavasti jos kiinnitetyillä arvoilla k ja t rajoitteen (6) kaarisulkujen lauseke saa arvon 0, saa rajoitteen (5) kaarisulkujen lauseke arvon 1. Erityisesti näiden kahden rajoitteen kaarisulkujen lausekkeet eivät koskaan saa samanaikaisesti arvoa 0. Nimittäin kahdella palalla ei voi samanaikaisesti olla vastakkaisia reunoja sekä pysty- että vaakasuunnassa. Kuitenkin rajoitteiden (5) ja (6) kaarisulkulausekkeet voivat saada samaan aikaan arvon 1. Tällöin paloilla k ja t ei ole yhtään yhteistä reunaa eli ne eivät sijaitse vierekkäin.

Edellä on nähty, että epäyhtälön (5) tai (6) oikean puolen kaarisulkulausekkeen arvo on 0, jos palat k ja t ovat vierekkäin. Tällöin kyseisen epäyhtälön koko oikean puolen arvoksi tulee 0. Tällöin myös epäyhtälön vasen puoli on pakotettu saamaan saman arvon, mikä tarkoittaa sitä, että palojen k ja t vastakkaisten reunojen on oltava samanväriset. Toisaalta jos epäyhtälön oikean puolen kaarisulkujen lauseke saa arvon 1, on kyseisen epäyhtälön oikean puolen arvo M. Näin ollen, jos M on riittävän suuri luku, niin tämän rajoitteen toteuttavat kaikki mahdolliset värit palojen k ja t vastakkaisille reunoille. Kyseinen rajoite on siis paloille k ja t niin kutsuttu turha rajoite, koska palat eivät todellisuudessa ole sivu- ja/tai pystysuunnassa vierekkäin.

Nyt kaikki Eternity II -palapelin mallintamiseen tarvittavat rajoitteet on esitelty. Kaikki muut rajoitteet ovat lineaarisia sekä sileitä eli jatkuvasti differentioituvia, paitsi rajoitteet (5) ja (6), jotka ovat itseisarvofunktion takia epäsileitä eli eivät kaikkialla jatkuvasti differentioituvia. Tehdään seuraavaksi näistä kahdesta rajoitejoukosta sileitä. Jokainen rajoite (5) voidaan kirjottaa kahtena epäyhtälönä muodossa

$$r_{k3} - r_{t1} \leq M \left(1 - \sum_{i \in S_n} \sum_{j \in S_{n-1}} x_{ijk} x_{i,j+1,t} \right), \quad k, t \in K,\ k \neq t \quad \text{ja} \tag{7}$$

$$-r_{k3} + r_{t1} \leq M \left(1 - \sum_{i \in S_n} \sum_{j \in S_{n-1}} x_{ijk} x_{i,j+1,t} \right), \quad k, t \in K,\ k \neq t. \tag{8}$$

Vastaavasti jokainen rajoite (6) saadaan muotoon

$$r_{k4} - r_{t2} \leq M \left(1 - \sum_{i \in S_{n-1}} \sum_{j \in S_n} x_{ijk} x_{i+1,jt} \right), \quad k, t \in K,\ k \neq t \quad \text{ja} \tag{9}$$

$$-r_{k4} + r_{t2} \leq M \left(1 - \sum_{i \in S_{n-1}} \sum_{j \in S_n} x_{ijk} x_{i+1,jt} \right), \quad k, t \in K,\ k \neq t. \tag{10}$$

Kohdefunktio ja kaikki rajoitteet voidaan nyt koota yhteen optimointitehtäväksi. Lisäksi valitaan $M = p_{max}$, koska ratkaistavana on p_{max} eri värin Eternity II.

Lopputuloksena saadaan epälineaarinen sekalukuoptimointitehtävä

$$
\begin{aligned}
\max_{x_{ijk},y_{km},r_{kl}} \quad & \sum_{i\in S_n}\sum_{j\in S_n}\sum_{k\in K} x_{ijk} \\
\text{s.t.} \quad & \sum_{i\in S_n}\sum_{j\in S_n} x_{ijk} \le 1, \quad k \in K \\
& \sum_{k\in K} x_{ijk} \le 1, \quad i \in S_n,\ j \in S_n \\
& y_{k1} + y_{k2} + y_{k3} \le 1, \quad k \in K \\
& r_{k1} \le p_{max}\left(1 - \sum_{i\in S_n} x_{i1k}\right), \quad k \in K \\
& r_{k2} \le p_{max}\left(1 - \sum_{j\in S_n} x_{1jk}\right), \quad k \in K \\
& r_{k3} \le p_{max}\left(1 - \sum_{i\in S_n} x_{ink}\right), \quad k \in K \\
& r_{k4} \le p_{max}\left(1 - \sum_{j\in S_n} x_{njk}\right), \quad k \in K \\
& r_{k1} = c_{k1}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k4}y_{k1} + c_{k3}y_{k2} + c_{k2}y_{k3}, \quad k \in K \\
& r_{k2} = c_{k2}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k1}y_{k1} + c_{k4}y_{k2} + c_{k3}y_{k3}, \quad k \in K \\
& r_{k3} = c_{k3}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k2}y_{k1} + c_{k1}y_{k2} + c_{k4}y_{k3}, \quad k \in K \\
& r_{k4} = c_{k4}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k3}y_{k1} + c_{k2}y_{k2} + c_{k1}y_{k3}, \quad k \in K \\
& r_{k3} - r_{t1} \le p_{max}\left(1 - \sum_{i\in S_n}\sum_{j\in S_{n-1}} x_{ijk}x_{i,j+1,t}\right), \quad k,t \in K,\ k \ne t \\
& -r_{k3} + r_{t1} \le p_{max}\left(1 - \sum_{i\in S_n}\sum_{j\in S_{n-1}} x_{ijk}x_{i,j+1,t}\right), \quad k,t \in K,\ k \ne t \\
& r_{k4} - r_{t2} \le p_{max}\left(1 - \sum_{i\in S_{n-1}}\sum_{j\in S_n} x_{ijk}x_{i+1,jt}\right), \quad k,t \in K,\ k \ne t \\
& -r_{k4} + r_{t2} \le p_{max}\left(1 - \sum_{i\in S_{n-1}}\sum_{j\in S_n} x_{ijk}x_{i+1,jt}\right), \quad k,t \in K,\ k \ne t \\
& x_{ijk} \in \{0,1\}, \quad i \in S_n,\ j \in S_n,\ k \in K \\
& y_{km} \in \{0,1\}, \quad k \in K,\ m = 1,2,3 \\
& r_{kl} \in [0, p_{max}], \quad k \in K,\ l = 1,2,3,4.
\end{aligned}
\tag{11}
$$

Jotta esitellystä epälineaarisesta sekalukuoptimointitehtävästä (11) saadaan line-

aarinen, on rajoitteet (7)-(10) linearisoitava luvussa 3 esitetyllä tavalla. Näin ollen kyseisten rajoitteiden jokainen sellaisen termi, jossa on kahden erilaisen binäärimuuttujan tulo pitää linearisoida. Linearisoituna rajoitteet (7) ja (8) ovat muotoa

$$r_{k3} - r_{t1} \leq M \left(1 - \sum_{i \in S_n} \sum_{j \in S_{n-1}} s_{ijk}^{i,j+1,t} \right), \quad k,t \in K,\ k \neq t \quad \text{ja}$$

$$-r_{k3} + r_{t1} \leq M \left(1 - \sum_{i \in S_n} \sum_{j \in S_{n-1}} s_{ijk}^{i,j+1,t} \right), \quad k,t \in K,\ k \neq t.$$

Tämän lisäksi tarvitaan lisäehdot

$$s_{ijk}^{i,j+1,t} \geq 1 + (x_{ijk} + x_{i,j+1,t} - 2), \quad i \in S_n,\ j \in S_{n-1},\ k,t \in K,\ k \neq t$$
$$s_{ijk}^{i,j+1,t} \leq \frac{1}{2}(x_{ijk} + x_{i,j+1,t}), \quad i \in S_n,\ j \in S_{n-1},\ k,t \in K,\ k \neq t$$
$$s_{ijk}^{i,j+1,t} \in \{0,1\}, \quad i \in S_n,\ j \in S_{n-1},\ k,t \in K,\ k \neq t.$$

Vastaavalla tavalla on linearisoitava rajoitteet (9)-(10). Yhteensä esitellyissä neljässä epälineaarisessa rajoitteessa on kiinnitetyille paloille k ja t erilaisia epälineaarisia termejä $2n(n-1) = 2n^2 - 2n$ kappaletta, joista jokainen on linearisoitava. Lisäksi pitää vielä huomioida se, että palat k ja t voidaan valita $n^2(n^2-1) = n^4 - n^2$ tavalla ja jokaisella valinnalla on $2n^2 - 2n$ linearisoitavaa termiä. Tästä syystä linearisoidussa Eternity II -palapelin mallinnustavan 2 mallissa on $(2n^2-2n)(n^4-n^2) = 2(n^6-n^5-n^4+n^3)$ lisämuuttujaa ja $2 \cdot 2(n^6-n^5-n^4+n^3) = 4(n^6-n^5-n^4+n^3)$ lisärajoitetta.

5.1.3 Mallinnustapojen vertailua

Tässä aliluvussa vertaillaan edellä esiteltyjen Eternity II -palapelin mallistapoja 1 ja 2. Erityisesti tarkastellaan mallien välisiä eroja rajoite-, muuttuja- ja parametrimäärissä. Taulukkoon 1 on listattu päätösmuuttujien, parametrien ja rajoitteiden määriä erikokoisilla Eternity II -palapeleillä, kun on käytetty mallinnustavan 1 lineaarista mallia (2). Vastaavat tiedot mallinnustavan 2 epälineaariselle mallille (11) on listattu taulukkoon 2 ja mallin (11) linearisoidulle versiolle taulukkoon 3. Taulukoiden värit-sarake kertoo, kuinka monta väriä on reunan harmaan lisäksi mukana tarkastellussa mallissa. Taulukossa 1 on siis pääsääntöisesti oletettu, että jokainen palapeli koostuu neljästä väristä sekä harmaasta reunasta. Puolestaan taulukoissa 2 ja 3 annetut tulokset eivät riipu värien määrästä, ja näin ollen ne ovat voimassa mille tahansa värimäärille.

Tarkastellaan ensin mallinnustavan 1 tuloksia. Taulukosta 1 voidaan huomata sekä päätösmuuttujien että parametrien määrän hurja kasvu jo pienillä pelilaudan

Taulukko 1: Päätösmuuttujien, parametrien ja rajoitteiden määrät Eternity II -palapelin koon kasvaessa käytettäessä mallinnustavan 1 lineaarista mallia (2).

Koko	**Päätösmuuttujat**	**Parametrit**	**Rajoitteet**	**Värit**
2×2	68	320	48	4
4×4	1048	1280	240	4
10×10	40180	8000	1680	4
16×16	262 624	20480	4416	4
16×16	262 624	94208	21696	22
$n \times n$	$4n^4 + 2n^2 - 2n$	$16n^2(p_{max} + 1)$	$2n^2(1 + 2p_{max}) + 4n(1 - p_{max})$	p_{max}

Taulukko 2: Päätösmuuttujien, parametrien ja rajoitteiden määrät Eternity II -palapelin koon kasvaessa käytettäessä mallinnustavan 2 epälineaarista mallia (11).

Koko	**Päätösmuuttujat**	**Parametrit**	**Rajoitteet**	**Värit**
2×2	44	16	92	p_{max}
4×4	368	64	1136	p_{max}
10×10	10700	400	40700	p_{max}
16×16	67328	1024	263 936	p_{max}
$n \times n$	$n^4 + 7n^2$	$4n^2$	$4n^4 + 7n^2$	p_{max}

Taulukko 3: Päätösmuuttujien, parametrien ja rajoitteiden määrät Eternity II -palapelin koon kasvaessa käytettäessä mallinnustavan 2 mallin (11) lineaarista versiota.

Koko	**Päätösmuuttujat**	**Parametrit**	**Rajoitteet**	**Värit**
2×2	92	16	188	p_{max}
4×4	6128	64	12656	p_{max}
10×10	1 792 700	400	3 604 700	p_{max}
16×16	31 401 728	1024	62 932 736	p_{max}
$n \times n$	$2n^6 - 2n^5 - n^4 + 2n^3 + 7n^2$	$4n^2$	$4n^6 - 4n^5 + 4n^3 + 7n^2$	p_{max}

koon muutoksilla. Puolestaan rajoitteiden määrän kasvu on melko maltillista. On hyvä huomata, että värien määrä ei vaikuta päätösmuuttujien määrään, ainoastaan parametrien ja rajoitteiden määrään. Mallinnustavalla 1 alkuperäisessä 22 värin ja harmaan reunan Eternity II -palapelissä on 262 624 päätösmuuttujaa, 94208 para-

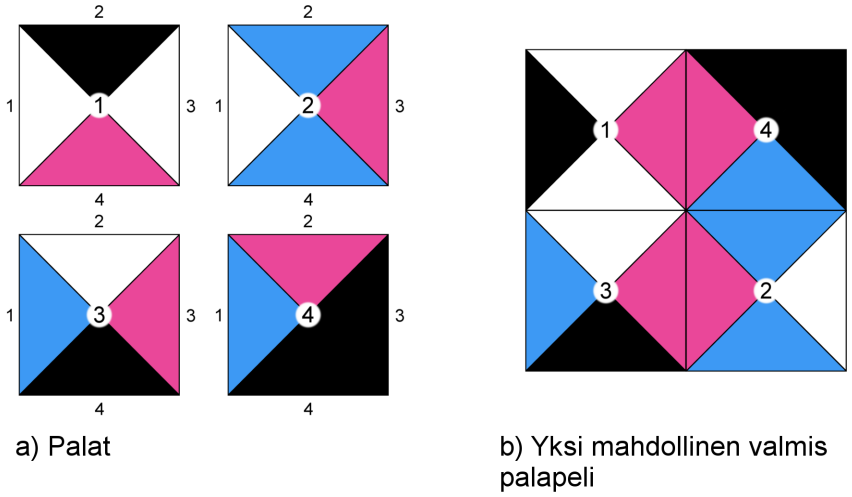
metria ja 21696 rajoitetta.

Seuraavaksi tarkastellaan mallinnustavan 2 lineaarisen ja epälineaarisen mallin tuloksien eroja mallinnustapaan 1. On hyvä huomata, että mallinnustavassa 2 värien määrä ei myöskään vaikuta päätösmuuttujien määrään. Värien määrä ei vaikuta edes parametrien tai rajoitteiden määrään, kuten mallissa (2). Taulukosta 2 huomataan, että epälineaarisella mallilla jo pelilaudan kasvu 2×2-kokoisesta 4×4-kokoiseen kasvattaa päätösmuuttujien määrän yli kahdeksankertaiseksi. Lineaarisella mallilla kasvu on yli 66-kertainen ja mallinnustavalla 1 yli 15-kertainen. Havaitaankin, että päätösmuuttujien määrä on epälineaarisella mallinnustavalla 2 paljon pienempi ja lineaarisella mallinnustavalla 2 erittäin paljon suurempi kuin mallinnustavalla 1. Lisäksi on hyvä huomata, että pieni osa mallinnustavan 2 muuttujista on jatkuvia. Tehtävän ratkaisemisen kannalta jatkuvat muuttujat ovatkin binäärisiä muuttujia toivotumpia. Vaikka päätösmuuttujien määrää on saatu mallinnustavan 2 epälineaarisessa mallissa pienennettyä mallinnustavan 1 malliin (2) verrattuna (2×2-kokoisella palapelillä pienennys 24 päätösmuuttujaa), niin edelleen päätösmuuttujien määrä nousee nopeasti ja on isommille ongelmille hyvin suuri. Mallinnustavassa 2 (sekä lineaarisessa että epälineaarisessa mallissa) on saatu pienennettyä myös parametrien määrää mallinnustapaan 1 verrattuna. Esimerkiksi 2×2-kokoisella palapelillä pienennys on 304 parametria. Puolestaan rajoitteita on mallinnustavan 1 mallissa vähemmän kuin mallinnustavan 2 malleissa. Pienimmällä 2×2-kokoisella palapelillä mallinnustavan 1 mallissa on 44 rajoitetta vähemmän kuin mallinnustavan 2 epälineaarisessa mallissa ja 140 rajoitetta vähemmän kuin mallinnustavan 2 lineaarisessa mallissa.

On hyvä havaita, että taulukoiden 1-3 toiseksi viimeiset rivit esittävät alkuperäistä Eternity II -palapeliä, ja seuraava tarkastelu on tehty nimenomaan tälle Eternity II -palapelille. Mallinnustavan 1 mallissa on lähes neljää kertaa (195 296 kappaletta) enemmän päätösmuuttujia kuin mallinnustavan 2 epälineaarisessa mallissa ja yli 119 kertaa (31 139 104 kappaletta) vähemmän päätösmuuttujia kuin mallinnustavan 2 lineaarisessa mallissa. Puolestaan parametreja mallinnustavassa 1 on 92 kertaa (93184 kappaletta) enemmän kuin mallinnustavan 2 malleissa. Kuitenkin mallinnustavan 2 epälineaarisessa mallissa on rajoitteita yli 12 kertaa (242 240 kappaletta) enemmän kuin mallinnustavassa 1. Mallinnustavan 2 lineaarisessa mallissa on puolestaan rajoitteita yli 2900 kertaa (62 911 040 kappaletta) enemmän kuin mallinnustavan 1 mallissa.

Yhteenvetona mallinnustapojen vertailusta voidaan siis havaita seuraavaa. Päätösmuuttujien ja parametrien määrä on mallinnustavassa 1 huomattavasti suurempi kuin mallinnustavan 2 epälineaarisessa mallissa. Tilanne on kuitenkin päinvastai-

nen rajoitteiden määrissä. Puolestaan mallinnustavan 2 lineaarisessa mallissa ovat sekä päätösmuuttujien että rajoitteiden määrät mallinnustavan 1 mallia merkittävästi suuremmat. Näin ollen lineaaristen mallien välisessä vertailussa huomataan mallinnustavan 1 mallin olevan mallinnustavan 2 lineaarista mallia parempi. Mallinnustavan 1 huonoutena voidaan pitää sitä, että värien määrä vaikuttaa parametrien ja rajoitteiden määrään. Puolestaan tavassa 2 värien määrällä ei ole vaikutusta päätösmuuttujiin, parametreihin eikä rajoitteisiin. Kuitenkin mallinnustavan 2 huonoutena voidaan pitää epälineaarisuutta ja sitä, että linearisointi nostaa muuttujien ja rajoitteiden määriä rajusti. Koska molemmilla mallinnustavoilla on sekä hyvät että huonot puolensa, ei ole yksinkertaista laittaa malleja paremmuusjärjestykseen. Kuitenkin mallinnustavan 2 epälineaarisen mallin voidaan todeta vaikuttavan mallinnustavan 2 lineaarista mallia paremalta. Lineaarinen malli on yleisesti epälineaarista helpompi ratkaista. Kuitenkin lineaarisessa mallissa on nyt moninkertaisesti enemmän päätösmuuttujia ja rajoitteita epälineaariseen malliin verrattuna. Esimerkiksi 16×16-kokoisessa Eternity II -palapelissä on mallinnustavan 2 lineaarisessa mallissa päätösmuuttujia yli 466 kertaa enemmän ja rajoitteita yli 238 kertaa enemmän kuin mallinnustavan 2 epälineaarisessa mallissa, joten lineaarista mallinnustapaa 2 voidaan pitää epälineaarista huonompana. Joka tapauksessa sekä mallinnustavat 1 että 2 ovat liian monimutkaisia siihen, että niillä voitaisiin ratkaista alkuperäinen 16×16-kokoinen 22 värin ja harmaan reunan Eternity II -palapeli. Tämä johtuu päätösmuuttujien, ja erityisesti binääristen päätösmuuttujien, suuresta määrästä. Näin ollen Eternity II -palapeliä kannattaisi yrittää mallintaa jollakin muulla kuin esitellyillä tavoilla.



Kuva 15: Eternity II -palapeli, jossa ei ole määrättyjä reunapaloja.

5.2 Eternity II yleistyksiä

Tässä aliluvussa tarkastellaan Eternity II -palapelin yleistyksiä ja muunnoksia. Erityisesti tarkastellaan mallia (11), ja muokataan sitä tilanteeseen sopivaksi. Tästä syystä päätösmuuttujat ovat vastaavat kuin aliluvun 5.1.2 mallissa.

5.2.1 Ei reunoja -Eternity

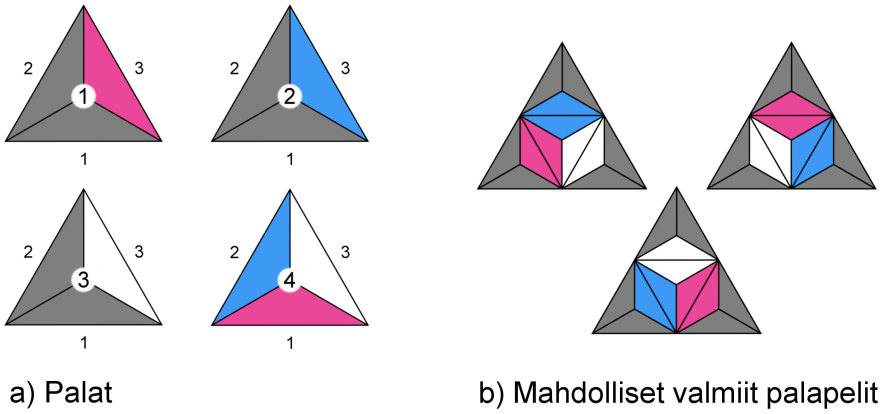
Muokataan seuraavaksi edellä tarkasteltua mallia (11) tilanteeseen, jossa Eternity-palapelistä puuttuvat reunat. Toisin sanoen palapelin reunoilla ei ole harmaata, vaan mitkä tahansa värit sopivat reunoihin. Kuva 15 havainnollistaa tällaista Eternity II -palapeliä.

Uutta mallia varten riittää, että esitellystä mallista (11) poistetaan ne neljä rajoitetta, jotka koskevat harmaita ulkoreunoja. Nämä ovat rajoitteet (4). Lopputuloksena saadaan epälineaarinen sekalukuoptimointitehtävä

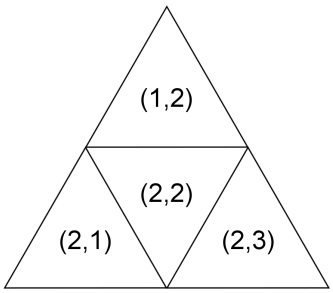
$$
\begin{aligned}
\max_{x_{ijk},y_{km},r_{kl}} \quad & \sum_{i\in S_n}\sum_{j\in S_n}\sum_{k\in K} x_{ijk} \\
\text{s.t.} \quad & \sum_{i\in S_n}\sum_{j\in S_n} x_{ijk} \le 1, \quad k \in K \\
& \sum_{k\in K} x_{ijk} \le 1, \quad i \in S_n,\ j \in S_n \\
& y_{k1} + y_{k2} + y_{k3} \le 1, \quad k \in K \\
& r_{k1} = c_{k1}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k4}y_{k1} + c_{k3}y_{k2} + c_{k2}y_{k3}, \quad k \in K \\
& r_{k2} = c_{k2}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k1}y_{k1} + c_{k4}y_{k2} + c_{k3}y_{k3}, \quad k \in K \\
& r_{k3} = c_{k3}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k2}y_{k1} + c_{k1}y_{k2} + c_{k4}y_{k3}, \quad k \in K \\
& r_{k4} = c_{k4}(1 - y_{k1} - y_{k2} - y_{k3}) + c_{k3}y_{k1} + c_{k2}y_{k2} + c_{k1}y_{k3}, \quad k \in K \\
& r_{k3} - r_{t1} \le p_{max}\left(1 - \sum_{i\in S_n}\sum_{j\in S_{n-1}} x_{ijk}x_{i,j+1,t}\right), \quad k,t \in K,\ k \ne t \\
& -r_{k3} + r_{t1} \le p_{max}\left(1 - \sum_{i\in S_n}\sum_{j\in S_{n-1}} x_{ijk}x_{i,j+1,t}\right), \quad k,t \in K,\ k \ne t \\
& r_{k4} - r_{t2} \le p_{max}\left(1 - \sum_{i\in S_{n-1}}\sum_{j\in S_n} x_{ijk}x_{i+1,jt}\right), \quad k,t \in K,\ k \ne t \\
& -r_{k4} + r_{t2} \le p_{max}\left(1 - \sum_{i\in S_{n-1}}\sum_{j\in S_n} x_{ijk}x_{i+1,jt}\right), \quad k,t \in K,\ k \ne t \\
& x_{ijk} \in \{0,1\}, \quad i \in S_n,\ j \in S_n,\ k \in K \\
& y_{km} \in \{0,1\}, \quad k \in K,\ m = 1,2,3 \\
& r_{kl} \in [0, p_{max}], \quad k \in K,\ l = 1,2,3,4.
\end{aligned}
$$

5.2.2 Kolmio-Eternity

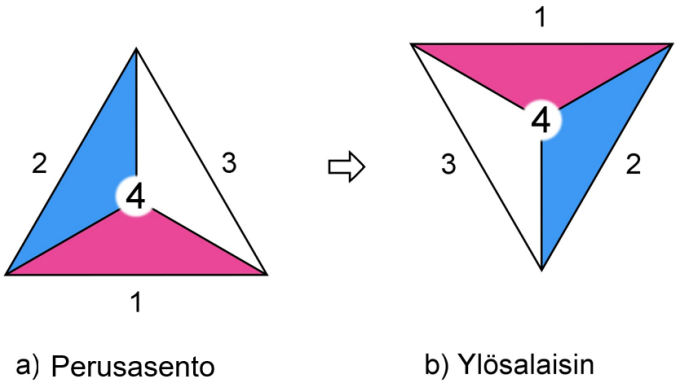
Esitellään seuraavaksi malliin (11) pohjautuva Eternity II -malli, jossa palat ja pelilauta ovat tasasivuisten kolmioiden muotoiset. Kuvassa 16 on havainnollistettu tällaista palapeliä. Lisäksi kuvaan on merkitty reunojen numerot palojen ollessa perusasennossa. Kun kolmio-Eternity koostuu neljästä palasta, numeroidaan pelilaudan ruudut kuvan 17 esittämällä tavalla. Yleisesti ruudut $(1, j)$ vastaavat ensimmäistä riviä, $(2, j)$ toista riviä, $(i, 1)$ ensimmäistä saraketta ja niin edelleen. Lisäksi jatkossa käytetään termiä "ylösalaisin", jolla viitataan kuvan 18 esittämään tilanteeseen. On siis huomattava, että esimerkiksi kuvan 17 ruutuun $(2, 2)$ sijoitettava pala on ylösalaisin, jolloin sen reunojen numerointi on erilainen kuin muiden palojen.



Kuva 16: Kolmion muotoinen Eternity II -palapeli.



Kuva 17: Neljän palan kolmio-Eternityn pelilaudan ruutujen numerointi.



Kuva 18: Kolmion muotoisen palan rotatoituminen siten, että se on "ylösalaisin".

Seuraavaksi muodostetaan malli neljän palan kolmio-Eternitylle. Mallin muodostus suuremmalle palapelille menee vastaavalla tavalla. Jatkossa käytetään nimitystä $n \times n$-kokoinen kolmio-Eternity yleisestä kolmio-Eternitystä, jonka yksi sivu muodostuu n kappaleesta paloja. Tällöin $n \times n$-kokoisessa kolmio-Eternityssä on yhteensä n^2 palaa, joten perinteisten $n \times n$-kokoisten Eternity II -pulmapelien vastine on $n \times n$-kokoinen kolmio-Eternity. Aluksi määritellään indeksijoukot J_i, jotka kertovat, mitä

sarakkeita j riviin i kuuluu. Indeksijoukot ovat kuvan 16 esimerkissä

$$J_1 = \{2\} \text{ ja } J_2 = \{1, 2, 3\}.$$

Vastaavasti $n \times n$-kokoiselle kolmio-Eternitylle indeksijoukot ovat

$$J_i = \{1 + (n - i), \ldots, 2n - 1 - (n - i)\}, \quad i = 1, 2, ..., n.$$

Lisäksi määritellään palojen joukko $K = \{1, 2, \ldots, n^2\}$, jolloin yksittäinen pala $k \in K$. Neljän palan eli 2×2-kokoisen kolmio-Eternityn palojen joukko on $K = \{1, 2, 3, 4\}$.

Neljän palan kolmio-Eternityn mallia varten tarvitaan parametrit

$$c_{kl} = \text{palan } k \text{ reunan } l \text{ väri perusasennossa,}$$

missä $k \in K$ ja $l = 1, 2, 3$. Tässä $c_{kl} \in \{0, 1, 2, 3\}$, missä 0 tarkoittaa harmaata, 1 valkoista, 2 pinkkiä ja 3 sinistä. Lisäksi tarvitaan päätösmuuttujat

$$x_{ijk} = \begin{cases} 1, & \text{jos pala } k \text{ on ruudussa } (i, j) \\ 0, & \text{muuten,} \end{cases}$$

$$y_{km} = \begin{cases} 1, & \text{jos palaa } k \text{ on kierretty kellon suuntaan } m \cdot 120^\circ \\ 0, & \text{jos pala } k \text{ on perusasennossa,} \end{cases}$$

$$r_{kl} \in [0, 3], \quad \text{palan } k \text{ reunan } l \text{ väri, kun pala on paikallaan,}$$

missä $i = 1, 2$, $j \in J_i$, $k \in K$, $m = 1, 2$ ja $l = 1, 2, 3$. Kohdefunktio on muuten samanlainen kuin mallissa (11), mutta summien indeksien arvot muuttuvat hieman. Maksimoitava kohdefunktio on nyt

$$\sum_{i=1}^{2} \sum_{j \in J_i} \sum_{k \in K} x_{ijk}.$$

Esitellään seuraavaksi tarvittavat rajoitteet. Rajoite

$$\sum_{i=1}^{2} \sum_{j \in J_i} x_{ijk} \leq 1, \quad k \in K$$

varmistaa sen, että pala k on korkeintaan yhdessä ruudussa (i, j). Lisäksi rajoite

$$\sum_{k \in K} x_{ijk} \leq 1, \quad i = 1, 2, \ j \in J_i$$

pitää huolen siitä, että ruutuun (i, j) sijoitetaan korkeintaan yksi pala. Kohdefunktion maksimointi takaa sen, että kummankin rajoitteen vasen puoli saa arvokseen

tasan yksi, jos ongelmalle on olemassa ratkaisu. Tällöin jokainen pala k on täsmälleen yhdessä ruudussa. Vastaavasti jokaiseen ruutuun sijoitetaan täsmälleen yksi pala. Lisäksi jokainen pala k voi olla vain yhdessä rotaatiossa. Tämän takaa rajoite

$$y_{k1} + y_{k2} \leq 1, \quad k \in K.$$

Jotta rotatoituneen palan k reunojen värit saadaan määritettyä, tarvitaan rajoitteet

$$\begin{aligned}
r_{k1} &= c_{k1}(1 - y_{k1} - y_{k2}) + c_{k3}y_{k1} + c_{k2}y_{k2}, \quad k \in K \\
r_{k2} &= c_{k2}(1 - y_{k1} - y_{k2}) + c_{k1}y_{k1} + c_{k3}y_{k2}, \quad k \in K \\
r_{k3} &= c_{k3}(1 - y_{k1} - y_{k2}) + c_{k2}y_{k1} + c_{k1}y_{k2}, \quad k \in K.
\end{aligned}$$

Lisäksi palojen harmaiden reunojen on oltava pelilaudan reunalla. Nämä ehdot toteutuvat rajoitteilla

$$\begin{aligned}
r_{k1} &\leq M(1 - x_{21k} - x_{23k}), \quad k \in K \\
r_{k2} &\leq M(1 - x_{21k} - x_{12k}), \quad k \in K \\
r_{k3} &\leq M(1 - x_{12k} - x_{23k}), \quad k \in K,
\end{aligned}$$

kun $M > 0$ on riittävän suuri vakio. Koska tarkastelussa on pieni neljän palan tehtävä, jossa suurin värin arvo on 3, on $M = 3$ pienin sallittu arvo parametrille M.

Lopuksi esitellään vielä rajoitteet, jotka takaavat sen, että vierekkäisten palojen vastakkain olevat reunat ovat samanväriset. Seuraavat rajoitteet varmistavat kyseisen ehdon

$$\begin{aligned}
r_{k1} - r_{t1} &\leq M(1 - x_{12k}x_{22t}), \quad k, t \in K,\ k \neq t \\
-r_{k1} + r_{t1} &\leq M(1 - x_{12k}x_{22t}), \quad k, t \in K,\ k \neq t \\
r_{k2} - r_{t2} &\leq M(1 - x_{22k}x_{23t}), \quad k, t \in K,\ k \neq t \\
-r_{k2} + r_{t2} &\leq M(1 - x_{22k}x_{23t}), \quad k, t \in K,\ k \neq t \\
r_{k3} - r_{t3} &\leq M(1 - x_{21k}x_{22t}), \quad k, t \in K,\ k \neq t \\
-r_{k3} + r_{t3} &\leq M(1 - x_{21k}x_{22t}), \quad k, t \in K,\ k \neq t,
\end{aligned} \tag{12}$$

kun $M > 0$ on riittävän suuri vakio. Koska edelleen suurin värin arvo on 3, on oltava $M \geq 3$. Tarkastellaan vielä lähemmin kahta ensimmäistä näistä rajoitteista. Molemmissa rajoitteissa keskitytään ruuduissa $(1, 2)$ ja $(2, 2)$ oleviin paloihin. Mikäli pala k on ruudussa $(1, 2)$ ja pala t ruudussa $(2, 2)$, niin rajoitteet muuttuvat muotoon $r_{k1} - r_{t1} \leq 0$ ja $-r_{k1} + r_{t1} \leq 0$ Tämä on sama asia kuin ehto $0 \leq |r_{k1} - r_{t1}| \leq 0$, jolloin ruudussa $(1, 2)$ olevan palan reuna 1 ja ruudussa $(2, 2)$ olevan palan reuna

1 ovat pakosta samanväriset. On hyvä huomata, että verrattujen reunojen numerot ovat samat siksi, että ruudun $(2,2)$ pala on ylösalaisin. Tällöin reunojen numerointi poikkeaa normaalista edellä kerrotun mukaisesti.

Kootaan kohdefunktio sekä kaikki esitellyt rajoitteet yhteen. Valitaan lisäksi $M = 3$. Lopputulokseksi saadaan epälineaarinen sekalukuoptimointitehtävä

$$
\begin{aligned}
\max_{x_{ijk}, y_{km}, r_{kl}} \quad & \sum_{i=1}^{2} \sum_{j \in J_i} \sum_{k \in K} x_{ijk} \\
\text{s. t.} \quad & \sum_{i=1}^{2} \sum_{j \in J_i} x_{ijk} \leq 1, \quad k \in K \\
& \sum_{k \in K} x_{ijk} \leq 1, \quad i = 1,2,\ j \in J_i \\
& y_{k1} + y_{k2} \leq 1, \quad k \in K \\
& r_{k1} = c_{k1}(1 - y_{k1} - y_{k2}) + c_{k3} y_{k1} + c_{k2} y_{k2}, \quad k \in K \\
& r_{k2} = c_{k2}(1 - y_{k1} - y_{k2}) + c_{k1} y_{k1} + c_{k3} y_{k2}, \quad k \in K \\
& r_{k3} = c_{k3}(1 - y_{k1} - y_{k2}) + c_{k2} y_{k1} + c_{k1} y_{k2}, \quad k \in K \\
& r_{k1} \leq 3 \cdot (1 - x_{21k} - x_{23k}), \quad k \in K \\
& r_{k2} \leq 3 \cdot (1 - x_{21k} - x_{12k}), \quad k \in K \\
& r_{k3} \leq 3 \cdot (1 - x_{12k} - x_{23k}), \quad k \in K \\
& r_{k1} - r_{t1} \leq 3 \cdot (1 - x_{12k} x_{22t}), \quad k,t \in K,\ k \neq t \\
& -r_{k1} + r_{t1} \leq 3 \cdot (1 - x_{12k} x_{22t}), \quad k,t \in K,\ k \neq t \\
& r_{k2} - r_{t2} \leq 3 \cdot (1 - x_{22k} x_{23t}), \quad k,t \in K,\ k \neq t \\
& -r_{k2} + r_{t2} \leq 3 \cdot (1 - x_{22k} x_{23t}), \quad k,t \in K,\ k \neq t \\
& r_{k3} - r_{t3} \leq 3 \cdot (1 - x_{21k} x_{22t}), \quad k,t \in K,\ k \neq t \\
& -r_{k3} + r_{t3} \leq 3 \cdot (1 - x_{21k} x_{22t}), \quad k,t \in K,\ k \neq t \\
& x_{ijk} \in \{0,1\}, \quad i = 1,2,\ j \in J_i,\ k \in K \\
& y_{km} \in \{0,1\}, \quad k \in K,\ m = 1,2 \\
& r_{kl} \in [0,3], \quad k \in K,\ l = 1,2,3,
\end{aligned}
\tag{13}
$$

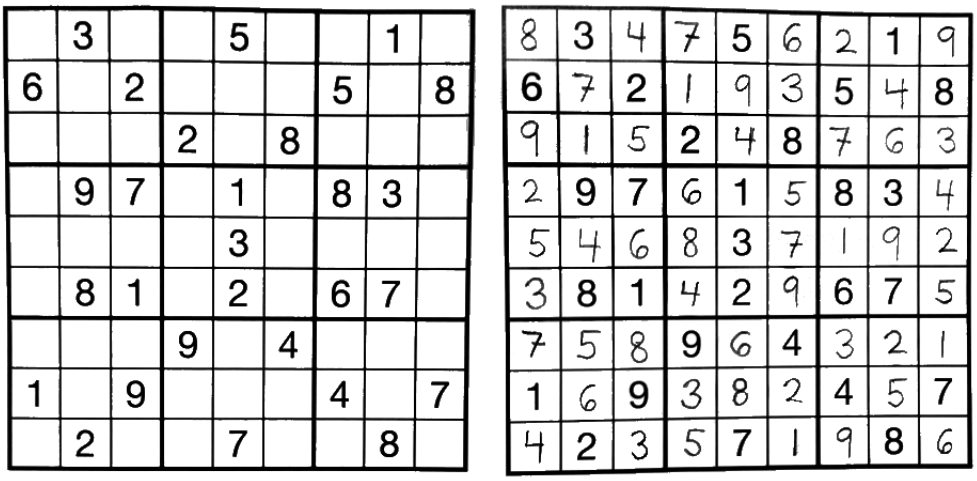
jossa on 36 päätösmuuttujaa, 12 parametria ja 108 rajoitetta. Vastaavassa $n \times n$-kokoisen kolmio-Eternityn mallissa on $n^4 + 5n^2$ päätösmuuttujaa, $3n^2$ parametria ja $6n^4 + 3n^2$ rajoitetta. Jotta esitellystä epälineaarisesta optimointitehtävästä saadaan lineaarinen, pitää rajoitteissa (12) olevat binäärimuuttujien tulot linearisoida. Linearisointi tapahtuu vastaavalla tavalla kuin edellä esitellyssä perinteisen Eternity II -palapelin mallinnustavassa 2. Kun neljäpalaisen kolmio-Eternityn malli linearisoidaan, tulee uusia päätösmuuttujia 36 kappaletta ja uusia rajoitteita 72 kappaletta.

6 Sudoku

Sudoku on yksi tunnetuimmista matemaattisista pulmapeleistä. Sudokusta on kehitetty vuosien saatossa lukuisia erilaisia versioita. Nämä kaikki kuitenkin pohjautuvat sudokun perusideaan: luvut 1–9 esiintyvät jokaisella rivillä, jokaisessa sarakkeessa ja jokaisessa alimatriisissa täsmälleen kerran. Tässä luvussa tarkastellaan niin perinteistä sudokua kuin sen modifikaatioitakin.

6.1 Perinteinen sudoku

Perinteinen sudoku koostuu 9×9-matriisista, joka on jaettu yhdeksään 3×3-alimatriisiin. Tavoitteena on täyttää 9×9-matriisi siten, että jokaisessa sarakkeessa, rivissä ja 3×3-alimatriisissa on luvut 1–9 täsmälleen kerran. Lisäksi sudokuun on valmiiksi täytetty vihjenumeroita. Yksi esimerkki perinteisestä sudokusta ja sen ratkaisusta on esitetty kuvassa 19. Tarkastellaan seuraavaksi perinteisen sudokun mallintamista ja esitetään kolme vaihtoehtoista mallinnustapaa. Esitystavat ovat vastaavia kuin monisteessa [13].



Kuva 19: Esimerkki perinteisestä sudokusta ja sen ratkaisusta.

6.1.1 Mallinnustapa 1

Muodostetaan sudoku-ongelmalle ensimmäinen malli. Merkitään rivejä muuttujalla i ja sarakkeita muuttujalla j. Olkoot päätösmuuttujat

$$x_{ijk} = \begin{cases} 1, & \text{jos luku } k \text{ on ruudussa } (i,j) \\ 0, & \text{muuten,} \end{cases}$$

missä $i = 1, \ldots, 9$, $j = 1, \ldots, 9$ ja $k = 1, \ldots, 9$. Nyt binäärisiä päätösmuuttujia on $9^3 = 729$ kappaletta.

Seuraavaksi tarkastellaan tarvittavia rajoitteita ja kohdefunktiota. Ensinnäkin pitää varmistaa, että jokaisessa ruudussa on tasan yksi numero. Rajoitteet

$$\sum_{k=1}^{9} x_{ijk} \leq 1, \quad i, j \in \{1, \ldots, 9\}$$

takaavat sen, että jokaisessa ruudussa on korkeintaan yksi numero. Lisäksi, kun kohdefunktioksi valitaan

$$\max_{x_{ijk}} \quad \sum_{i=1}^{9} \sum_{j=1}^{9} \sum_{k=1}^{9} x_{ijk},$$

pitää maksimointi huolen siitä, että jokaiseen ruutuun tulee tarkalleen yksi numero. Jotta varmistetaan, että kaikissa riveissä on jokainen luku 1–9 täsmälleen kerran, tarvitaan rajoite

$$\sum_{j=1}^{9} x_{ijk} \leq 1, \quad i, k \in \{1, \ldots, 9\}.$$

Ehto kuitenkin takaa ainoastaan sen, että jokainen numero on rivissä korkeintaan yhden kerran. Kuten edellä, pitää kohdefunktion maksimointi huolen siitä, että jokaista lukua on täsmälleen yksi per rivi. Vastaavalla tavalla voidaan varmistaa, että jokainen luku 1–9 sijaitsee jokaisessa sarakkeessa täsmälleen kerran. Tällöin rajoite on muotoa

$$\sum_{i=1}^{9} x_{ijk} \leq 1, \quad j, k \in \{1, \ldots, 9\}.$$

Vielä tarvitaan ehdot, jotka varmistavat, että jokaisessa 3×3-alimatriisissa on luvut 1–9 täsmälleen kerran. Tästä syystä jokaiselle yhdeksästä alimatriisista kirjoitetaan oma rajoitejoukko, joiksi saadaan

$$\sum_{i=1}^{3} \sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=1}^{3} \sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=1}^{3} \sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1, \ldots, 9\}$$

$$\sum_{i=4}^{6} \sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=4}^{6} \sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=4}^{6} \sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1, \ldots, 9\}$$

$$\sum_{i=7}^{9} \sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=7}^{9} \sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=7}^{9} \sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1, \ldots, 9\}.$$

Näissä rajoitteissa taataan, että jokaista numeroa 1–9 on alimatriisissa nolla tai yksi kappale. Kuitenkin kohdefunktion maksimointi varmistaa taas jokaisen luvun sijaitsevan alimatriisissa tarkalleen kerran.

Kaikki yllä esitellyt rajoitteet ovat lineaarisia. Kun rajoitteet ja kohdefunktio sekä tieto päätösmuuttujan binäärisyydestä kootaan yhteen, saadaan lineaarinen binäärinen optimointitehtävä. Se on muotoa

$$
\begin{aligned}
\max_{x_{ijk}} \quad & \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} x_{ijk} \\
\text{s.t.} \quad & \sum_{k=1}^{9} x_{ijk} \leq 1, \quad i,j \in \{1,\ldots,9\} \\
& \sum_{j=1}^{9} x_{ijk} \leq 1, \quad i,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{ijk} \leq 1, \quad j,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& x_{ijk} \in \{0,1\}, \quad i,j,k \in \{1,\ldots,9\}.
\end{aligned}
\tag{14}
$$

Optimointitehtävälle saadaan ratkaisu, kun löydetään sellaiset binääristen päätösmuuttujien x_{ijk} arvot, jotka toteuttavat kaikki rajoitteet ja joilla kohdefunktion arvo on 81.

On huomattava, että optimointitehtävä (14) ei sisällä sudokun vihjenumeroita. Ne kuitenkin saadaan malliin helposti mukaan seuraavalla tavalla. Oletetaan, että ruudussa (i,j) on vihjenumero $m \in \{1,\ldots,9\}$. Tällöin optimointitehtävään voidaan lisätä rajoitteet

$$x_{ijm} = 1 \quad \text{ja} \quad x_{ijk} = 0, \ k \in \{1,\ldots,9\} \setminus \{m\}.$$

Nämä rajoitteet on kirjoitettava jokaiselle annetulle vihjenumerolle. Tätä havainnollistetaan esittämällä kuvan 19 sudokun vasemman ylänurkan 3×3-alimatriisin

vihjenumerot. Vihjenumeroiden rajoitteet voidaan kirjoittaa muodossa

$$
\begin{aligned}
&x_{123} = 1 \\
&x_{12k} = 0, \quad k \in \{1,2,4,5,6,7,8,9\} \\
&x_{216} = 1 \\
&x_{21k} = 0, \quad k \in \{1,2,3,4,5,7,8,9\} \\
&x_{232} = 1 \\
&x_{23k} = 0, \quad k \in \{1,3,4,5,6,7,8,9\}.
\end{aligned}
$$

Huomataan kuitenkin, että muotoa $x_{ijk} = 0, k \in \{1,\ldots,9\} \setminus \{m\}$, olevat rajoitteet ovat turhia. Optimointitehtävän (14) rajoite $\sum_{k=1}^{9} x_{ijk} \leq 1$ kiinnitetyille $i,j \in \{1,\ldots,9\}$ nimittäin takaa sen, että tietyssä ruudussa (i,j) vain yhdellä arvolla k voi $x_{ijk} = 1$. Toisin sanoen tämä ehto varmistaa, että yhdessä ruudussa voi olla vain yksi numero. Näin ollen vihjenumeroiden rajoitteet riittää kirjoittaa muodossa $x_{ijm} = 1$, kun m on paikkaan (i,j) annettu vihjenumero.

6.1.2 Mallinnustapa 2

Sudoku voidaan mallintaa myös lineaarisena yhtälöryhmänä. Kyseinen malli vastaa optimointitehtävää (14) muuten, mutta kohdefunktiota ei tarvita ja epäyhtälömerkit rajoitteissa korvataan yhtäsuuruuksilla. Näin voidaan tehdä seuraavin perustein. Optimointitehtävässä (14) kohdefunktion maksimointi varmistaa kaikkien rajoitteiden aktiivisuuden eli sen, että rajoitteiden vasen ja oikea puoli ovat samansuuruiset. Kun kohdefunktio poistetaan, on rajoitteiden aktiivisuus taattava eri tavalla. Tämä varmistetaan juurikin yhtäsuuruusrajoitteilla. Kun nämä muutokset tehdään,

saadaan lineaarinen yhtälöryhmä

$$
\begin{aligned}
&\sum_{k=1}^{9} x_{ijk} = 1, \quad i,j \in \{1,\dots,9\} \\
&\sum_{j=1}^{9} x_{ijk} = 1, \quad i,k \in \{1,\dots,9\} \\
&\sum_{i=1}^{9} x_{ijk} = 1, \quad j,k \in \{1,\dots,9\} \\
&\sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} = 1,\ \sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} = 1,\ \sum_{i=1}^{3}\sum_{j=7}^{9} x_{ijk} = 1, \quad k \in \{1,\dots,9\} \\
&\sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} = 1,\ \sum_{i=4}^{6}\sum_{j=4}^{6} x_{ijk} = 1,\ \sum_{i=4}^{6}\sum_{j=7}^{9} x_{ijk} = 1, \quad k \in \{1,\dots,9\} \\
&\sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} = 1,\ \sum_{i=7}^{9}\sum_{j=4}^{6} x_{ijk} = 1,\ \sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} = 1, \quad k \in \{1,\dots,9\} \\
&x_{ijk} \in \{0,1\}, \quad i,j,k \in \{1,\dots,9\}.
\end{aligned}
\tag{15}
$$

On hyvä huomata, että vaikka epäyhtälöt vaihdettiin yhtälöiksi, kuvaavat rajoitteet edelleen vastaavia asioita kuin optimointitehtävässä (14). Yhtälöryhmälle saadaan ratkaisu, kun löydetään sellaiset binääriset päätösmuuttujien x_{ijk} arvot, jotka toteuttavat kaikki yhtälöt. Lisäksi optimointitehtävään on otettava mukaan sudokun vihjenumerot vastaavalla tavalla kuin edellä esitellyssä mallinnustavassa 1. Tällöin mikä tahansa yhtälöryhmän ratkaisu on sudokun ratkaisu.

Esitelty lineaarinen yhtälöryhmä on mahdollista täydentää optimointitehtäväksi. Tällöin yhtälöt muodostavat optimointitehtävän rajoitteet. Näiden lisäksi pitää valita tehtävälle vielä kohdefunktio. Se voidaan kuitenkin valita vapaasti, sillä kaikki rajoitteet toteuttavat sallitut ratkaisut ovat itse sudokun ratkaisuja. Ei siis ole merkitystä minimoidaanko vai maksimoidaanko vapaasti valittavaa kohdefunktiota.

6.1.3 Mallinnustapa 3

Seuraavaksi esitellään kolmas tapa sudokun mallintamiseen. Olkoot päätösmuuttujat

$$x_{ij} = \text{ruutuun } (i,j) \text{ sijoitettu numero,}$$

missä $i = 1,\dots,9$, $j = 1,\dots,9$ ja $x_{ij} \in \{1,\dots,9\}$. Kyseiset kokonaislukumuuttujat voidaan esittää muodossa

$$x_{ij} = \beta_{ij0} \cdot 2^0 + \beta_{ij1} \cdot 2^1 + \beta_{ij2} \cdot 2^2 + \beta_{ij3} \cdot 2^3,$$

missä $\beta_{ijk} \in \{0,1\}$, $i = 1,\ldots,9$, $j = 1,\ldots,9$ ja $k = 0,1,2,3$. Lisäksi kokonaislukumuuttuja x_{ij} voidaan olettaa jatkuvaksi, kun käytetään yllä olevaa binääristä esitystapaa.

Muodostetaan seuraavaksi tarvittavat sudoku-ongelman rajoitteet. Rajoitteisiin on otettava mukaan edellä kerrottu päätösmuuttujien muotoilu. Tästä saadaan ehdot

$$\begin{aligned} &x_{ij} = \beta_{ij0} + 2\beta_{ij1} + 4\beta_{ij2} + 8\beta_{ij3}, \quad i,j \in \{1,\ldots,9\} \\ &\beta_{ijk} \in \{0,1\}, \quad i,j \in \{1,\ldots,9\},\ k \in \{0,1,2,3\}. \end{aligned}$$

Kyseiset rajoitteet takaavat päätösmuuttujien x_{ij} olevan kokonaislukuja väliltä $[0,15]$. Vielä tarvitaan kuitenkin ehdot, jotka varmistavat päätösmuuttujien arvojen olevan yhdestä yhdeksään. Nämä vaatimukset voidaan kirjoittaa muodossa

$$x_{ij} \geq 1, \quad i,j \in \{1,\ldots,9\} \quad \text{ja} \quad x_{ij} \leq 9, \quad i,j \in \{1,\ldots,9\}. \tag{16}$$

Mikäli päätösmuuttujia x_{ij} ei halua esittää binäärimuuttujien avulla, riittää tällöin ensinnäkin vaatia ainoastaan rajoitteet (16), jotka takaavat päätösmuuttujien saavan arvoja yhdestä yhdeksään. Lisäksi jokaisen päätösmuuttujan on oltava kokonaislukumuuttuja.

Jotta varmistetaan, että jokaisella vaakarivillä i on jokainen numero 1–9 täsmälleen kerran, tarvitaan rajoite

$$|x_{ij} - x_{ik}| \geq 1, \quad i = 1,\ldots,9,\ j = 1,\ldots,8,\ k = j+1,\ldots,9.$$

Vastaavasti rajoite

$$|x_{ij} - x_{kj}| \geq 1, \quad i = 1,\ldots,8,\ j = 1,\ldots,9,\ k = i+1,\ldots,9,$$

takaa sen, että jokaisessa sarakkeessa j esiintyy kukin numero vain kerran. Vielä tarvitaan vastaavat rajoitteet jokaiselle 3×3-alimatriisille. Näiksi saadaan

$$\begin{aligned} &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{1,2,3\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{4,5,6\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{7,8,9\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{1,2,3\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{4,5,6\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{7,8,9\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{1,2,3\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{4,5,6\} \\ &|x_{ij} - x_{kl}| \geq 1, \quad i<k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{7,8,9\}, \end{aligned}$$

joissa jokainen rajoite vastaa yhtä alimatriisia. On hyvä huomata, että alimatriisin samalla vaaka- tai pystyrivillä olevia alkioita ei tarvitse enää erikseen vertailla keskenään, sillä edellä esitellyissä rajoitteissa on jo käyty läpi kaikki samojen vaaka- ja pystyrivien alkiot. Näin ollen yhdessä rivi- ja sarakerajoitteiden kanssa nämä ehdot takaavat sen, että missään 3×3-ruudukossa ei ole kahta samaa numeroa.

Kaikki yllä olevat rajoitteet yhdessä mallintavat sudoku-ongelman. Lopputuloksena saadaankin epäyhtälöryhmä

$$
\begin{aligned}
& x_{ij} = \beta_{ij0} + 2\beta_{ij1} + 4\beta_{ij2} + 8\beta_{ij3}, \quad i,j \in \{1,\dots,9\} \\
& x_{ij} \geq 1, \quad i,j \in \{1,\dots,9\} \\
& x_{ij} \leq 9, \quad i,j \in \{1,\dots,9\} \\
& |x_{ij} - x_{ik}| \geq 1, \quad i = 1,\dots,9,\ j = 1,\dots,8,\ k = j+1,\dots,9 \\
& |x_{ij} - x_{kj}| \geq 1, \quad i = 1,\dots,8,\ j = 1,\dots,9,\ k = i+1,\dots,9 \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{1,2,3\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{4,5,6\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{7,8,9\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{1,2,3\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{4,5,6\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{7,8,9\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{1,2,3\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{4,5,6\} \\
& |x_{ij} - x_{kl}| \geq 1, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{7,8,9\} \\
& \beta_{ijk} \in \{0,1\}, \quad i,j = 1,\dots,9,\ k = 0,1,2,3.
\end{aligned}
\tag{17}
$$

Mikä tahansa epäyhtälöryhmän ratkaisu on ratkaisu tarkastelussa olevalle sudokutehtävälle, kunhan vihjenumerot on kiinnitetty.

Yllä oleva epäyhtälöryhmä on epälineaarinen ja epäsileä, sillä se sisältää epälineaarisia ja epäsileitä itseisarvofunktioita. Hyvä puoli kuitenkin on, että rajoitteet $|x_{ij} - x_{kl}| \geq 1$ pystytään linearisoimaan luvussa 3 esitetyllä tavalla. Linearisointia varten määritellään binääriset päätösmuuttujat

$$
y_{ij}^{kl} = \begin{cases} 1, & \text{jos } x_{ij} - x_{kl} \geq 1 \\ 0, & \text{jos } x_{kl} - x_{ij} \geq 1. \end{cases}
$$

Tällöin alkuperäinen epäyhtälö $|x_{ij} - x_{kl}| \geq 1$ voidaan kirjoittaa kahtena epäyhtä-

lönä muodossa

$$x_{ij} - x_{kl} \geq 1 - M \cdot (1 - y_{ij}^{kl})$$
$$x_{kl} - x_{ij} \geq 1 - M \cdot y_{ij}^{kl},$$

missä M on riittävän suuri positiivinen reaaliluku. Sudokun tapauksessa vakion M on toteutettava ehto $M \geq 10$, joten voidaan valita esimerkiksi $M = 10$. Muunnoksen jälkeen kaikki rajoitteet ovat lineaariset sekä sileät, ja tarkastelussa on lineaarinen

sileä epäyhtälöryhmä

$$
\begin{aligned}
& x_{ij} = \beta_{ij0} + 2\beta_{ij1} + 4\beta_{ij2} + 8\beta_{ij3}, \quad i,j \in \{1,\dots,9\} \\
& x_{ij} \geq 1, \quad i,j \in \{1,\dots,9\} \\
& x_{ij} \leq 9, \quad i,j \in \{1,\dots,9\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i = 1,\dots,9,\ j = 1,\dots,8,\ k = j+1,\dots,9 \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i = 1,\dots,8,\ j = 1,\dots,9,\ k = i+1,\dots,9 \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{1,2,3\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{4,5,6\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{7,8,9\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{1,2,3\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{4,5,6\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{7,8,9\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{1,2,3\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{4,5,6\} \\
& x_{ij} - x_{kl} \geq 1 - 10 \cdot (1 - y_{ij}^{kl}), \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{7,8,9\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i = 1,\dots,9,\ j = 1,\dots,8,\ k = j+1,\dots,9 \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i = 1,\dots,8,\ j = 1,\dots,9,\ k = i+1,\dots,9 \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{1,2,3\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{4,5,6\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{1,2,3\},\ j,l \in \{7,8,9\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{1,2,3\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{4,5,6\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{4,5,6\},\ j,l \in \{7,8,9\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{1,2,3\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{4,5,6\} \\
& x_{kl} - x_{ij} \geq 1 - 10 \cdot y_{ij}^{kl}, \quad i < k,\ j \neq l,\ i,k \in \{7,8,9\},\ j,l \in \{7,8,9\} \\
& \beta_{ijk} \in \{0,1\}, \quad i,j = 1,\dots,9,\ k = 0,1,2,3 \\
& y_{ij}^{kl} \in \{0,1\}, \quad i,j,k,l = 1,\dots,9.
\end{aligned}
\tag{18}
$$

Siinä on sekä 810 päätösmuuttujaa että rajoitetta enemmän kuin epälineaarisessa epäsileässä mallissa (17).

Kuten yhtälöryhmässä (15), myös nyt voidaan kirjoittaa optimointitehtävä, jonka rajoitteina on esitelty epäyhtälöryhmä. Kohdefunktio sekä sen minimointi tai mak-

simointi voidaan valita vapaasti, koska niin kuin jo edellä on huomautettu, kaikki rajoitteet toteuttavat sallitut ratkaisut ovat itse sudokun ratkaisuja. Saatava optimointitehtävä on lineaarinen sekalukuoptimointitehtävä (eli MILP-tehtävä), kun itseisarvorajoitteet on korvattu niiden linearisoiduilla versioilla.

6.1.4 Mallinnustapojen vertailua

Tässä aliluvussa vertaillaan keskenään perinteiselle sudokulle esiteltyjen kolmen mallinnustavan muuttuja- ja rajoitemääriä. Koska näissä malleissa ei tarvita parametreja, ei niiden määriä luonnollisestikaan vertailla. Mallinnustavoissa 1 ja 2 on sekä muuttujia että rajoitteita sama määrä. Muuttujia on 729 kappaletta ja rajoitteita 324 kappaletta. Mallinnustapojen 1 ja 2 rajoitteiden ja muuttujien määrät ovat samat, koska mallissa (15) on vain korvattu mallin (14) epäyhtälöt yhtäsuuruuksilla. Näin ollen rajoitteiden ja muuttujien määrät pysyvät samoina.

Mallinnustavan 3 epälineaarisessa mallissa on muuttujia 405 ja rajoitteita 1053 kappaletta. Puolestaan mallinnustavan 3 lineaarisessa mallissa on 1215 muuttujaa ja 1863 rajoitetta. Mallinnustavan 3 lineaarinen malli on siis näiden lukujen valossa huonompi malli sudokulle kuin mallinnustavat 1 ja 2, sillä muuttujia ja rajoitteita on mallinnustavan 3 lineaarisessa mallissa enemmän kuin mallinnustapojen 1 ja 2 malleissa. Koska nämä kaikki kolme mallia ovat lineaarisia, on vertailu on mielekäs. Vaikka mallinnustavan 3 epälineaarisessa mallissa on muuttujia mallinnustapojen 1 ja 2 malleja vähemmän, tuo epälineaarisuus oman lisähaasteensa kyseisen mallin ratkaisemiseen.

Tarkastellaan seuraavaksi sitä, mistä erot mallien rajoitemäärissä johtuvat. Perinteisen sudokun mallinnustavan 3 epälineaarisessa mallissa on jo alun perin enemmän rajoitteita kuin mallinnustapojen 1 ja 2 malleissa. Tämä johtuu siitä, että rivi-, sarake- ja alimatriisirajoitteiden lisäksi mallissa (17) tarvitaan rajoitteet, jotka vaativat sijoitettujen lukujen olevan kokonaislukuja väliltä $1-9$. Puolestaan mallinnustavan 3 lineaarisen mallin (18) suuri rajoitteiden määrä johtuu siitä, että rivejä, sarakkeita ja alimatriiseja koskevat 11 rajoiteryhmää pitivät alun perin sisällään itseisarvon ja näin ollen ne jouduttiin linearisoimaan. Tästä aiheutui sekä rajoitteita että binäärimuuttujia 810 kappaletta lisää.

Lopuksi tarkastellaan vielä sitä, mistä muuttujien eri määrät johtuvat. Tavan 3 sudokussa päätösmuuttujan x_{ij} arvo kertoo ruutuun (i, j) sijoitettavan luvun. Päätösmuuttuja voi siis saada kokonaislukuarvon yhdestä yhdeksään, mikä on taattu linkittämällä jatkuva päätösmuuttuja binääriesitykseen. Muissa sudokumalleissa binäärisen päätösmuuttujan x_{ijk} arvo kertoo, sijoitetaanko luku k ruutuun (i, j). Mallinnustavoissa 1 ja 2 tarvitaan yhdeksän binäärimuuttujaa yhtä ruutua (i, j) kohti.

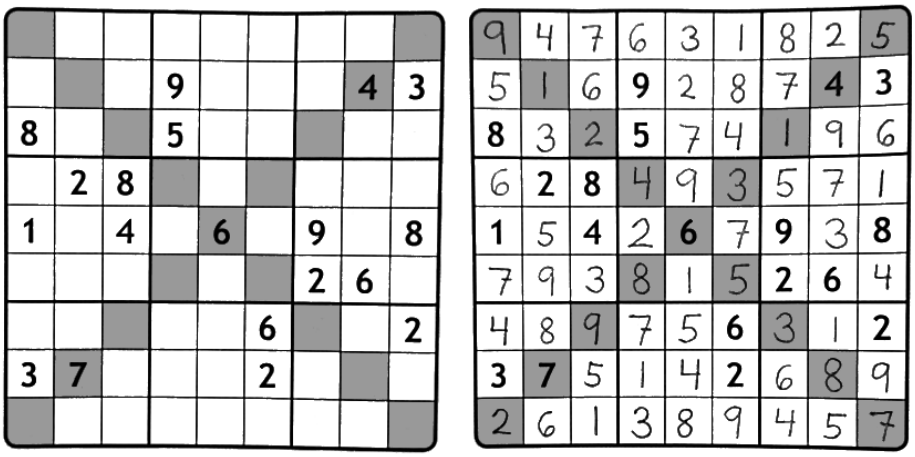
Puolestaan mallinnustavan 3 epälineaarisessa versiossa yhtä ruutua kohden tarvitaan 4 binäärimuuttujaa. Mallinnustavan 3 lineaariselle mallille ei kuitenkaan voida suoraan sanoa, kuinka monta binäärimuuttujaa yhtä ruutua kohti tarvitaan. Nimittäin linearisoinnista tulevat binäärimuuttujat eivät suoraan linkity vain yhteen ruutuun, sillä esimerkiksi ruudun $(1, 1)$ arvoa joudutaan vertaamaan yhteensä 20 muun ruudun kanssa eli kyseinen ruutu on mukana 20 eri binäärimuuttujassa. Mikäli lisämuuttujien lukumäärä 810 jaetaan ruutujen lukumäärällä 81, niin keskimäärin voidaan ajatella yhtä ruutua kohden tulevan 10 lisämuuttujaa. Linearisoinnin tuomat lisämuuttujat yhdessä alkuperäisten muuttujien kanssa tekevät sen, että mallinnustavan 3 lineaarisessa mallissa on eniten muuttujia. Tämä selittää myös osaltaan sitä, miksi perinteisen sudokun tavan 3 lineaarinen malli ei ole niin toimiva ja tehokas.

6.2 Modifioitu sudoku

Tässä luvussa esitellään kolme muunnosta perinteisestä sudokusta: sudoku-x, pariton-parillinen sudoku ja killer sudoku. Kaikki nämä sudokut sisältävät perinteisen sudokun säännöt. Jokaisessa on kuitenkin mukana vielä omat lisärajoitteensa.

6.2.1 Sudoku-x

Sudoku-x [2] on perinteinen sudoku, jossa myös kummallakin nurkasta nurkkaan kulkevalla lävistäjällä on numerot 1–9. Lisäksi sudoku-x ei ratkea ilman lävistäjien numeroiden apua. Kuvassa 20 on esitetty eräs sudoku-x ja sen ratkaisu. Tarkastellaan seuraavaksi sitä, miten perinteistä sudokua mallintavasta optimointitehtävästä (14) saadaan muokattua malli sudoku-x:lle.



Kuva 20: Eräs sudoku-x ja sen ratkaisu.

Koska sudoku-x sisältää kaikki perinteisenkin sudokun säännöt, voidaan optimointitehtävä (14) ottaa mukaan malliin sellaisenaan. Kuitenkin vielä tarvitaan rajoitteet, joilla taataan kummallakin lävistäjällä olevan luvut 1–9. Sudokun vasemmasta yläkulmasta lähtevän lävistäjän rajoite voidaan kirjoittaa muodossa

$$\sum_{i=1}^{9} x_{iik} \leq 1, \quad k \in \{1, \ldots, 9\}.$$

Toisen lävistäjän rajoite voidaan puolestaan kirjoittaa muodossa

$$\sum_{i=1}^{9} x_{i,10-i,k} \leq 1, \quad k \in \{1, \ldots, 9\}.$$

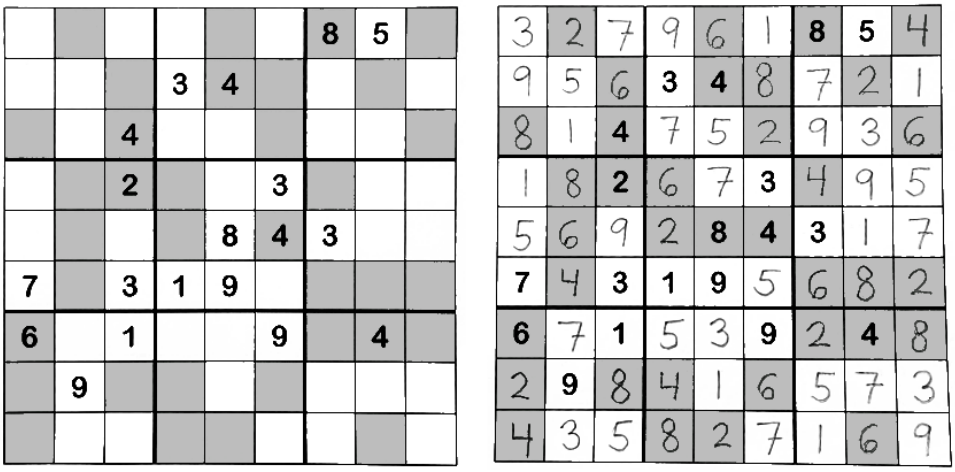
Nyt lävistäjien rajoitteet voidaan lisätä perinteistä sudokua mallintavaan optimointitehtävään. Tällöin saadaan ratkaistavaksi lineaarinen binäärinen optimointitehtävä

$$\begin{aligned}
\max_{x_{ijk}} \quad & \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} x_{ijk} \\
\text{s.t.} \quad & \sum_{k=1}^{9} x_{ijk} \leq 1, \quad i,j \in \{1,\ldots,9\} \\
& \sum_{j=1}^{9} x_{ijk} \leq 1, \quad i,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{ijk} \leq 1, \quad j,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{iik} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{i,10-i,k} \leq 1, \quad k \in \{1,\ldots,9\} \\
& x_{ijk} \in \{0,1\}, \quad i,j,k \in \{1,\ldots,9\}.
\end{aligned}$$

Kuten edelläkin, voidaan kohdefunktio poistaa, jos rajoitteiden epäyhtälöt korvataan yhtäsuuruuksilla. Tällöin sudoku-x:n ratkaisu saadaan lineaarisesta yhtälöryhmästä.

6.2.2 Pariton-parillinen sudoku

Pariton-parillinen sudoku (*odd/even sudoku*) [38] on muuten perinteinen sudoku, mutta parittomien ja parillisten numeroiden paikat ovat ennalta määrättyjä. Kuvassa 21 on esitetty eräs pariton-parillinen sudoku ja sen ratkaisu. Siinä harmaisiin ruutuihin tulee parillinen luku ja valkoisiin ruutuihin pariton luku. Muokataan seuraavaksi perinteisen sudokun mallia (14) pariton-parillinen sudokulle sopivaksi.



Kuva 21: Eräs pariton-parillinen sudoku ja sen ratkaisu.

Pariton-parillinen sudoku on siis nimenomaan perinteinen sudoku, johon on vain lisätty ehdot parittomien ja parillisten lukujen sijainnille. Tämän takia riittää, että optimointitehtävään (14) lisätään parittomien ja parillisten lukujen sijainteja koskevat rajoitteet. Muuten perinteisen sudokun mallia ei siis tarvitse muokata. Parittomien ja parillisten lukujen sijainteja varten tarvitaan kaksi joukkoa. Olkoot joukossa I_h kaikki harmaat ruudut (i, j) ja joukossa I_v kaikki valkoiset ruudut (i, j). Lisäksi määritellään kaksi joukkoa itse luvuille. Olkoot $J_e = \{2, 4, 6, 8\}$ ja $J_o = \{1, 3, 5, 7, 9\}$. Rajoitteet

$$\sum_{k \in J_o} x_{ijk} = 0, \quad \sum_{k \in J_e} x_{ijk} = 1, \quad (i,j) \in I_h \quad \text{ja}$$
$$\sum_{k \in J_o} x_{ijk} = 1, \quad \sum_{k \in J_e} x_{ijk} = 0, \quad (i,j) \in I_v$$

takaavat parillisten ja parittomien numeroiden sijoittuvan oikeisiin ruutuihin. Kuitenkin rajoitteisiin voidaan vaihtaa $\leq$-merkki, sillä kohdefunktion maksimointi takaa rajoitteiden olevan aktiivisia. Kun nämä rajoitteet lisätään perinteisen sudokun malliin, saadaan pariton-parillinen sudokusta lineaarinen binäärinen optimointiteh-

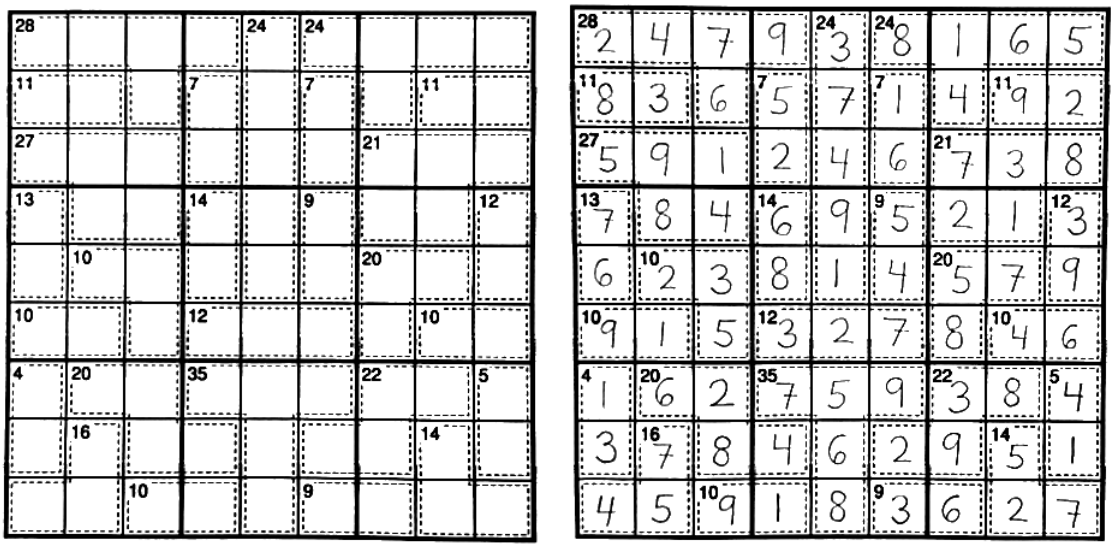
tävä

$$
\begin{aligned}
\max_{x_{ijk}} \quad & \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} x_{ijk} \\
\text{s.t.} \quad & \sum_{k=1}^{9} x_{ijk} \leq 1, \quad i,j \in \{1,\ldots,9\} \\
& \sum_{j=1}^{9} x_{ijk} \leq 1, \quad i,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{ijk} \leq 1, \quad j,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{k\in J_o} x_{ijk} \leq 0, \quad \sum_{k\in J_e} x_{ijk} \leq 1, \quad (i,j) \in I_h \\
& \sum_{k\in J_o} x_{ijk} \leq 1, \quad \sum_{k\in J_e} x_{ijk} \leq 0, \quad (i,j) \in I_v \\
& x_{ijk} \in \{0,1\}, \quad i,j,k \in \{1,\ldots,9\}.
\end{aligned}
$$

6.2.3 Killer sudoku

Killer sudoku on pulmapeli, jossa yhdistyvät sudokun ja kakuron (kts. luku 7.1) ominaisuudet. Perinteisen 9×9-kokoisen killer sudokun säännöt ovat seuraavat. Jokaisessa rivissä ja sarakkeessa ovat luvut 1–9 täsmälleen kerran. Lisäksi jokaisessa 3×3-alimatriisissa ovat kyseiset luvut myös täsmälleen kerran. Nämä säännöt vastaavat perinteisen sudokun sääntöjä. Lisäksi killer sudokun ruudukko on jaettu kahden tai useamman ruudun sisältäviin osioihin. Osiot eivät ole keskenään yhdenmuotoisia ja niiden muoto voi vaihdella suurestikin. Jokainen osio on merkitty numerolla. Tämä numero kertoo sen, mikä tulos osion ruutujen yhteenlaskusta on saatava. Lisäksi yhdessä osiossa mikään luku ei saa esiintyä kuin korkeintaan kerran. Kuvassa 22 on esimerkki killer sudokusta ja sen ratkaisusta. Siinä osioita on merkitty katkoviivoilla.

Koska killer sudokun perustana on perinteinen sudoku, voidaan jälleen hyödyntää optimointitehtävää (14) nykyisen mallin pohjana. Tarvitaan kuitenkin vielä rajoitteet eri osioiden summille. Näitä varten esitellään muutamia merkintöjä. Kun



Kuva 22: Esimerkki killer sudokusta ja sen ratkaisusta.

rivejä merkitään muuttujalla i ja sarakkeita muuttujalla j, voidaan jokainen yksittäinen ruutu kirjoittaa muodossa (i, j), missä $i, j \in \{1, \ldots, 9\}$. Olkoon T koko killer sudokussa olevien osioiden määrä. Yksittäisestä osiosta käytetään merkintää $t \in \{1, \ldots, T\}$. Lisäksi merkitään osioon t kuuluvia ruutuja joukolla R_t ja kyseisen osion summauksen arvoa parametrilla r_t.

Esiteltyjä merkintöjä käyttäen voidaan nyt kirjoittaa vaadittavat rajoitteet. Ensimmäiseksi tarvitaan ehto, joka vaatii, että yksittäisissä osioissa ovat luvut 1–9 korkeintaan kerran. Rajoite

$$\sum_{(i,j)\in R_t} x_{ijk} \leq 1, \quad t \in \{1, \ldots, T\},\ k \in \{1, \ldots, 9\}$$

takaa tämän. Tämän lisäksi osioissa olevien numeroiden pitää summautua annettuun arvoon. Voidaankin kirjoittaa

$$\sum_{(i,j)\in R_t} \sum_{k=1}^{9} k \cdot x_{ijk} = r_t, \quad t \in \{1, \ldots, T\},$$

joka varmistaa vaatimuksen. Kun nämä kaksi rajoitetta lisätään perinteisen sudokun

malliin, saadaan lopputuloksena lineaarinen binäärinen optimointitehtävä

$$
\begin{aligned}
\max_{x_{ijk}} \quad & \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} x_{ijk} \\
\text{s.t.} \quad & \sum_{k=1}^{9} x_{ijk} \leq 1, \quad i,j \in \{1,\ldots,9\} \\
& \sum_{j=1}^{9} x_{ijk} \leq 1, \quad i,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{9} x_{ijk} \leq 1, \quad j,k \in \{1,\ldots,9\} \\
& \sum_{i=1}^{3}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=1}^{3}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=4}^{6}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=4}^{6}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{i=7}^{9}\sum_{j=1}^{3} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=4}^{6} x_{ijk} \leq 1, \ \sum_{i=7}^{9}\sum_{j=7}^{9} x_{ijk} \leq 1, \quad k \in \{1,\ldots,9\} \\
& \sum_{(i,j)\in R_t} x_{ijk} \leq 1, \quad t \in \{1,\ldots,T\}, \ k \in \{1,\ldots,9\} \\
& \sum_{(i,j)\in R_t}\sum_{k=1}^{9} k \cdot x_{ijk} = r_t, \quad t \in \{1,\ldots,T\} \\
& x_{ijk} \in \{0,1\}, \quad i,j,k \in \{1,\ldots,9\},
\end{aligned}
$$

joka mallintaa killer sudokua. On hyvä huomata, että killer sudokua ei voida mallintaa lineaarisena yhtälöryhmänä. Nimittäin mikäli rajoitteeseen

$$\sum_{(i,j)\in R_t} x_{ijk} \leq 1, \quad t \in \{1,\ldots,T\}, \ k \in \{1,\ldots,9\}$$

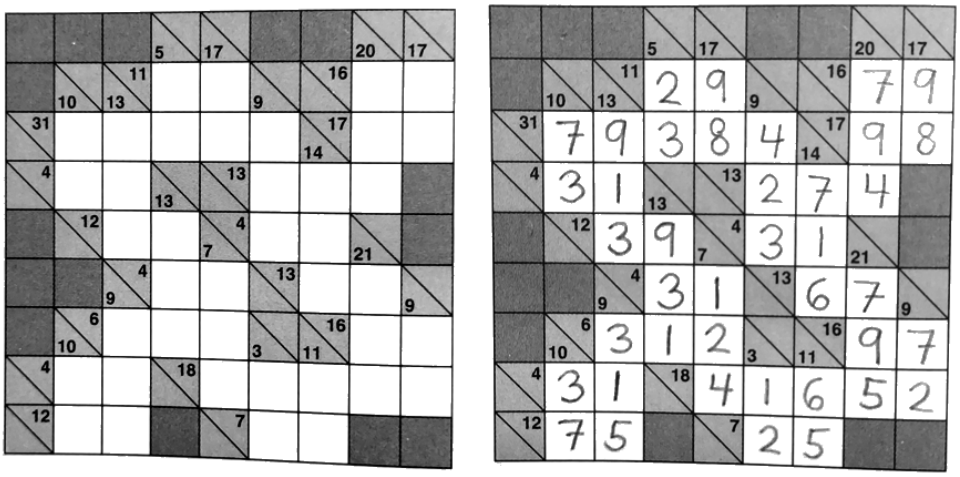
vaihdetaan yhtäsuuruus, sisältyisi jokainen luku 1–9 jokaiseen osioon. Tämä ei tietysti ole mahdollista, sillä killer sudokussa voi olla osioita, jotka sisältävät esimerkiksi vain kolme lukua.

7 Summafunktioon perustuvat pulmapelit

Edellä on jo nähty esimerkki summafunktioon perustuvasta pulmapelistä. Nimittäin killer sudokussa summafunktio on keskeisessä roolissa. Tarkastellaan seuraavaksi vielä kahta summafunktioon perustuvaa pulmapeliä, kakuroa ja japanilaista ristikkoa.

7.1 Kakuro

Kakuro (*cross sums*) [10] on klassinen lukujen yhteenlaskuun perustuva pulmapeli, joka on ollut olemassa jo vuosikymmenien ajan. Se koostuu $n \times n$-kokoisesta ruudukosta, jonka tyhjiin ruutuihin on tarkoituksena sijoittaa kokonaisluku 1–9. Peräkkäisille pysty- ja vaakasuuntaisille tyhjille ruuduille on annettu arvo. Näihin ruutuihin sijoittavien lukujen summan on oltava yhtä suuri kuin annettu arvo. Lisävaatimuksena on se, että summassa olevien lukujen on oltava erilaisia. Eräs kakuro ja sen ratkaisu on esitetty kuvassa 23. Tarkastellaan seuraavaksi sitä, miten kakuro voidaan mallintaa matemaattisesti.



Kuva 23: Esimerkki kakurosta ja sen ratkaisusta.

Mallinnetaan kakuro-ongelmaa optimointitehtävänä, jonka merkinnät perustuvat pitkälti artikkeliin [22]. Kuten edelläkin, merkitään rivejä indeksillä i ja sarakkeita indeksillä j. Tällöin jokainen ruutu voidaan ilmaista muodossa (i, j). Olkoon C joukko, joka sisältää kaikki tyhjät ruudut (i, j). Lisäksi olkoon T koko ruudukossa olevien summien lukumäärä. On hyvä huomata, että erikseen ei siis tarvitse erotella vaaka- ja pystysuuntaisia summia, vaan yhteinen parametri tarkastelussa riittää. Olkoon merkintä yksittäiselle summalle $t \in \{1, \ldots, T\}$. Summassa t yhteenlaskettavat ruudut kuuluvat joukkoon R_t, joka on joukon C osajoukko, ja kyseisen summan arvo on r_t. Yleensä summat t numeroidaan kakuron summia noudattaen vasemmalta oikealle ja ylhäältä alas kulkien. Mikäli samassa ruudussa on kaksi vihjenumeroa, merkitään ylempää summanumeroa suuremmalla numerolla.

Havainnollistetaan merkintöjä tarkastelemalla kuvan 23 kakuroa. Kyseisen kakuron ruudukko on 9×9-kokoinen ja summia on $T = 32$ kappaletta. Tutkitaan lähemmin oikean yläkulman summaa ja merkitään $t = 4$, $R_4 = \{(2, 9), (3, 9)\}$ ja $r_4 = 17$. Vastaavasti vasemman alakulman summalle merkitään $t = 31$, $R_{31} = \{(9, 2), (9, 3)\}$

ja $r_{31} = 12$.

Olkoot päätösmuuttujat

$$x_{ijk} = \begin{cases} 1, & \text{jos luku } k \text{ on ruudussa } (i,j) \\ 0, & \text{muuten,} \end{cases}$$

missä $k = 1, \ldots, 9$ ja $(i,j) \in C$. Kakuro voidaan kirjoittaa lineaarisena binäärisenä optimointitehtävänä

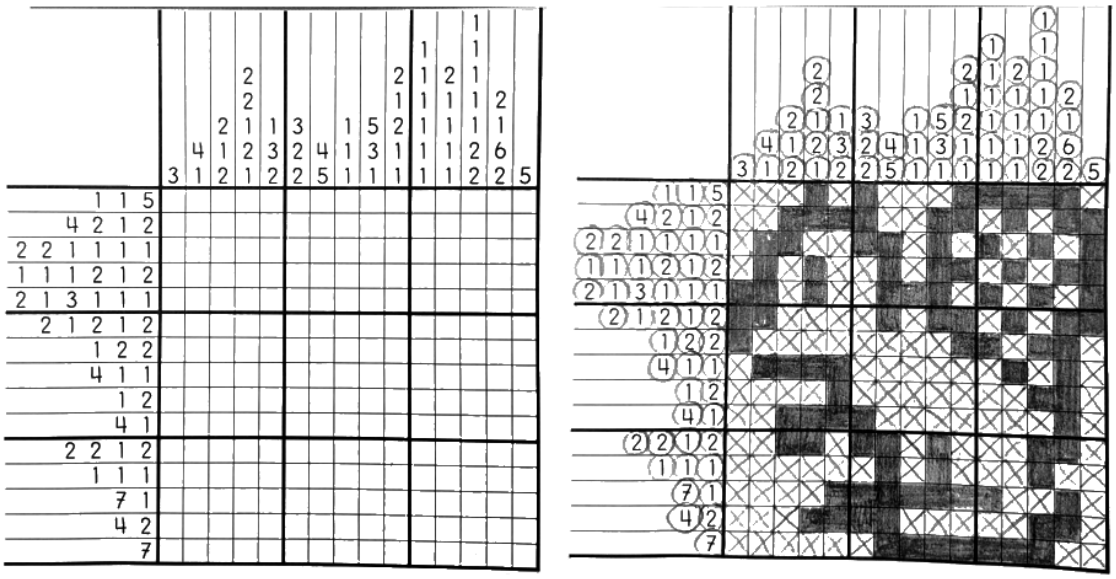
$$\begin{aligned} \max_{x_{ijk}} \quad & \sum_{(i,j)\in C} \sum_{k=1}^{9} x_{ijk} \\ \text{s. t.} \quad & \sum_{k=1}^{9} x_{ijk} \leq 1, \quad (i,j) \in C && (19) \\ & \sum_{(i,j)\in R_t} x_{ijk} \leq 1, \quad t = 1, \ldots, T,\ k = 1, \ldots, 9 && (20) \\ & \sum_{(i,j)\in R_t} \sum_{k=1}^{9} k \cdot x_{ijk} = r_t, \quad t = 1, \ldots, T && (21) \\ & x_{ijk} \in \{0,1\}, \quad (i,j) \in C,\ k = 1, \ldots, 9. && (22) \end{aligned}$$

Rajoite (19) takaa sen, että jokaiseen tyhjään ruutuun sijoitetaan korkeintaan yksi luvuista 1–9. Kuitenkin kohdefunktion maksimointi varmistaa, että tyhjään ruutuun sijoitetaan täsmälleen yksi luvuista 1–9. Rajoite (20) puolestaan pitää huolen siitä, että summataan yhteen vain ja ainoastaan eri lukuja. Toisin sanoen jokainen luvuista 1–9 esiintyy yhdessä summassa korkeintaan kerran. Rajoitteessa (21) vaaditaan, että summan tulos vastaa annettua lukua. Lisäksi viimeinen rajoite (22) asettaa päätösmuuttujat binäärisiksi. On hyvä huomata, että kakuroa ei voida mallintaa lineaarisena yhtälöryhmänä. Nimittäin mikäli rajoitteessa (20) on mukana yhtäsuuruus, esiintyisi jokainen luvuista 1–9 jokaisessa summassa. Tämä ei tietysti ole mahdollista, koska kakuro sisältää summia, joissa yhteenlaskettavia lukuja on esimerkiksi vain kaksi. Mallinnettaessa kuvan 23 kakuroa esitellyllä mallilla, on sekä muuttujia että rajoitteita 360 kappaletta.

7.2 Japanilainen ristikko

Japanilainen ristikko (*nonogram*) [29] on logiikkapeli, jossa tarkoituksena on värittää ruudukko tietyin säännöin. Ristikon värien määrä vaihtelee, mutta tässä tarkastellaan ainoastaan kaksiväristä (musta ja valkoinen) japanilaista ristikkoa. Ruudukon jokainen ruutu on siis joko musta tai valkoinen. Jokaisen rivin viereen on merkitty mustien ruutujen muodostamien yhtäjaksoisten jonojen pituudet kyseisessä rivissä.

Vastaavasti jokaisen sarakkeen yläpuolelle on merkitty mustien ruutujen muodostamien yhtäjaksoisten jonojen pituudet kyseisessä sarakkeessa. Yhtäjaksoisia mustia jonoja erottaa aina vähintään yksi valkoinen ruutu. Lisäksi on huomattava, että mustat jonot ovat rivissä/sarakkeessa siinä järjestyksessä, kuin ne on vihjeissä ilmoitettu. Tavoitteena on siis päätellä kaikkien mustien ruutujen sijainnit. Japanilainen ristikko on NP-täydellinen ongelma. Ei siis ole löydetty polynomisessa ajassa toimivaa algoritmia, joka ratkaisisi japanilaisen ristikon. Eräs japanilainen ristikko ja sen ratkaisu on esitetty kuvassa 24. Tarkastellaan seuraavaksi optimointitehtävän luomista japanilaiselle ristikolle. Esiteltävä binäärinen optimointitehtävä ja siinä käytettävät merkinnät pohjautuvat artikkeliin [5].



Kuva 24: Esimerkki japanilaisesta ristikosta ja sen ratkaisusta. Valkoisia ruutuja on ratkaisussa merkitty symbolilla X.

Kuten edellä, merkitään ruudukon rivejä indeksillä i ja sarakkeita indeksillä j. Tällöin yksittäinen ruutu saa merkinnän (i, j). Kun merkitään rivin/sarakkeen vihjeiden koostuvan vektoreista $(s_1, s_2, \ldots, s_k)^\top$, niin kyseisessä rivissä/sarakkeessa on oltava k mustien ruutujen jonoa, joiden pituudet ovat $s_i,\ i = 1, 2, \ldots, k$. Näin ollen ensimmäisenä on s_1 ruudun mittainen jono, toisena s_2 ruudun mittainen jono ja niin edelleen. Tässä ensimmäisellä tarkoitetaan rivien tapauksessa vasenta reunaa ja sarakkeiden tapauksessa yläreunaa. On hyvä huomata, että rivin/sarakkeen ei tarvitse alkaa mustalla ruudulla, kuten ei myöskään loppua. Mustien jonojen tarvitsee ainoastaan olla vektorin määräämässä järjestyksessä ja jonoja täytyy erottaa vähintään yhdellä valkoisella ruudulla.

Selkeyden vuoksi taulukossa 4 on listattu ensin kaikki mallissa tarvittavat para-

Taulukko 4: Japanilaisen ristikon mallinnuksessa tarvittavat parametrit.

m	rivien lukumäärä
n	sarakkeiden lukumäärä
k_i^r	mustien jonojen määrä rivillä i
k_j^c	mustien jonojen määrä sarakkeessa j
$(s_{i,1}^r, s_{i,2}^r, \ldots, s_{i,k}^r)^\top$	vihjenumerot rivillä i
$(s_{j,1}^c, s_{j,2}^c, \ldots, s_{j,k}^c)^\top$	vihjenumerot sarakkeessa j
$e_{i,t}^r$	numeroarvoltaan pienin sellainen sarake j, että rivin i jono t voidaan sijoittaa riville niin, että jonon vasemmanpuoleisin ruutu on sarakkeessa j
$l_{i,t}^r$	numeroarvoltaan suurin sellainen sarake j, että rivin i jono t voidaan sijoittaa riville niin, että jonon vasemmanpuoleisin ruutu on sarakkeessa j
$e_{j,t}^c$	numeroarvoltaan pienin sellainen rivi i, että sarakkeen j jono t voidaan sijoittaa sarakkeeseen niin, että jonon ylimmäisin ruutu on rivillä i
$l_{j,t}^c$	numeroarvoltaan suurin sellainen rivi i, että sarakkeen j jono t voidaan sijoittaa sarakkeeseen niin, että jonon ylimmäisin ruutu on rivillä i

metrit. Lisäksi yläindeksin merkintä r tarkoittaa riviä (*row*) ja c saraketta (*column*). Seuraavaksi esitellään tarvittavat päätösmuuttujat, jotka ovat kaikki binäärisiä. Ensimmäinen päätösmuuttuja on

$$z_{ij} = \begin{cases} 1, & \text{jos ruutu } (i,j) \text{ on musta} \\ 0, & \text{jos ruutu } (i,j) \text{ on valkoinen,} \end{cases}$$

missä $i = 1, \ldots, m$ ja $j = 1, \ldots, n$. Kyseinen päätösmuuttuja siis ilmaisee sen, minkä väriseksi jokainen ruutu väritetään. Muut kaksi päätösmuuttujaa kertovat sen, mihin rivien ja sarakkeiden mustat jonot sijoittuvat. Olkoot

$$y_{itj} = \begin{cases} 1, & \text{jos rivin } i \text{ jonon } t \text{ vasemmanpuoleisin ruutu on sarakkeessa } j \\ 0, & \text{muuten,} \end{cases}$$

missä $i = 1, \ldots, m$, $t = 1, \ldots, k_i^r$ ja $j = e_{i,t}^r, \ldots, l_{i,t}^r$. Lisäksi olkoot

$$x_{jti} = \begin{cases} 1, & \text{jos sarakkeen } j \text{ jonon } t \text{ ylimmäisin ruutu on rivillä } i \\ 0, & \text{muuten,} \end{cases}$$

missä $j = 1, \ldots, n$, $t = 1, \ldots, k_j^c$ ja $j = e_{j,t}^c, \ldots, l_{j,t}^c$.

Tarkastellaan seuraavaksi tarvittavia rajoitteita. Ensimmäiseksi tarvitaan rajoitteita mustilla jonoille. Rajoite

$$\sum_{j=e_{i,t}^r}^{l_{i,t}^r} y_{itj} = 1, \quad i = 1, \ldots, m,\ t = 1, \ldots, k_i^r$$

takaa sen, että rivin i jono t on kyseisellä rivillä täsmälleen kerran. Vasemmanpuoleinen summa käy nimittäin läpi kaikki ne sarakkeet j, joissa on mahdollista olla rivin i jonon t vasemmanpuoleisin ruutu. Mukaan tarvitaan myös rajoite

$$y_{itj} \leq \sum_{j'=j+s_{i,t}^r+1}^{l_{i,t+1}^r} y_{i,t+1,j'}, \quad i = 1, \ldots, m,\ t = 1, \ldots, k_i^r - 1,\ j = e_{i,t}^r, \ldots, l_{i,t}^r.$$

Rajoite takaa sen, että rivin i jono $t+1$ on sijoitettu vastaavalla rivillä olevan jonon t oikealle puolelle ja myös riittävän kauaksi, jotta väliin jää vähintään yksi valkoinen ruutu. Vastaavat kaksi rajoitetta sarakkeiden mustille jonoille ovat

$$\sum_{i=e_{j,t}^c}^{l_{j,t}^c} x_{jti} = 1, \quad j = 1, \ldots, n,\ t = 1, \ldots, k_j^c$$

ja

$$x_{jti} \leq \sum_{i'=i+s_{j,t}^c+1}^{l_{j,t+1}^c} x_{j,t+1,i'}, \quad j = 1, \ldots, n,\ t = 1, \ldots, k_j^r - 1,\ i = e_{j,t}^c, \ldots, l_{j,t}^c.$$

Toiseksi tarvitaan vielä rajoitteita, joilla taataan mustien ruutujen linkittyvän oikein mustien jonojen sijaintien kanssa. Rajoite

$$z_{ij} \leq \sum_{t=1}^{k_i^r} \sum_{j'=\max\{e_{it}^r, j-s_{it}^r+1\}}^{\min\{l_{it}^r, j\}} y_{itj'}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n$$

varmistaakin, että jos ruutu (i, j) on musta, niin vähintään yksi rivin i jonoista peittää kyseisen ruudun. Rajoitteen lähemmässä tarkastelussa huomataan, että jos ruutu (i, j) on musta, niin vähintään yhden rivin i jonon on sijoituttava siten, että ruutu (i, j) kuuluu jonoon. Rajoitteen oikeanpuolen jälkimmäisen summamerkinnän ala- ja ylärajat pitävät huolen siitä, että rivin i jonossa t tarkastellaan vain sarakkeet, joissa samanaikaisesti sekä $e_{it}^r \leq j' \leq l_{it}^r$ että $j - s_{it}^r + 1 \leq j' \leq j$. Epäyhtälöt $e_{it}^r \leq j' \leq l_{it}^r$ varmistavat, että rivin i jonolle t katsotaan ylipäätään vain alunperin sallittuja sarakkeita. Puolestaan epäyhtälöt $j - s_{it}^r + 1 \leq j' \leq j$ vaativat, että tarkastelussa

ovat ainoastaan ne sarakkeet j', joista alkava jono t voi rivillä i ylipäätään peittää kohdan (i, j). Vastaava asia sarakkeiden mustille ruuduille on

$$z_{ij} \leq \sum_{t=1}^{k_j^c} \sum_{i'=\max\{e_{jt}^c, i-s_{jt}^c+1\}}^{\min\{l_{jt}^c, i\}} x_{jti'}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

Näin ollen, jos ruutu (i, j) on musta, niin vähintään yksi sarakkeen j jonoista peittää kyseisen ruudun.

Lopuksi tarvitaan vielä rajoitteet, jotka estävät sen, että valkoisiin ruutuihin ei osu rivien eikä sarakkeiden mustia jonoja. Tätä varten määritellään ensin indeksijoukot

$$J_{ijt} = \{j' \in \mathbb{N} \mid e_{it}^r \leq j' \leq l_{it}^r \text{ ja } j - s_{it}^r + 1 \leq j' \leq j\} \quad \text{ja}$$
$$I_{ijt} = \{i' \in \mathbb{N} \mid e_{jt}^c \leq i' \leq l_{jt}^c \text{ ja } i - s_{jt}^c + 1 \leq i' \leq i\}.$$

Rajoitteiksi saadaan

$$z_{ij} \geq y_{itj'}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n,\ t = 1, \ldots, k_i^r,\ j' \in J_{ijt}$$
$$z_{ij} \geq x_{jti'}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n,\ t = 1, \ldots, k_j^c,\ i' \in I_{ijt}.$$

Kaikki edellä esitellyt rajoitteet yhdessä muodostavat japanilaiselle ristikolle mallin. Eksaktia kohdefunktiota ei varsinaisesti siis tarvita, koska mikä tahansa rajoitteet toteuttava sallittu ratkaisu on japanilaisen ristikon ratkaisu. Näin ollen

lopputuloksena saadaan lineaarinen binäärinen optimointitehtävä

$$
\begin{aligned}
\max_{x_{jti},y_{itj},z_{ij}} \quad & f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \\
\text{s.t.} \quad & \sum_{j=e^r_{i,t}}^{l^r_{i,t}} y_{itj} = 1, \quad i = 1,\dots,m,\ t = 1,\dots,k^r_i \\
& \sum_{i=e^c_{j,t}}^{l^c_{j,t}} x_{jti} = 1, \quad j = 1,\dots,n,\ t = 1,\dots,k^c_j \\
& y_{itj} \le \sum_{j'=j+s^r_{i,t}+1}^{l^r_{i,t+1}} y_{i,t+1,j'}, \quad i = 1,\dots,m,\ t = 1,\dots,k^r_i - 1,\ j = e^r_{i,t},\dots,l^r_{i,t} \\
& x_{jti} \le \sum_{i'=i+s^c_{j,t}+1}^{l^c_{j,t+1}} x_{j,t+1,i'}, \quad j = 1,\dots,n,\ t = 1,\dots,k^r_j - 1,\ i = e^c_{j,t},\dots,l^c_{j,t} \\
& z_{ij} \le \sum_{t=1}^{k^r_i} \sum_{j'=\max\{e^r_{it},j-s^r_{it}+1\}}^{\min\{l^r_{it},j\}} y_{itj'}, \quad i = 1,\dots,m,\ j = 1,\dots,n \\
& z_{ij} \le \sum_{t=1}^{k^c_j} \sum_{i'=\max\{e^c_{jt},i-s^c_{jt}+1\}}^{\min\{l^c_{jt},i\}} x_{jti'}, \quad i = 1,\dots,m,\ j = 1,\dots,n \\
& z_{ij} \ge y_{itj'}, \quad i = 1,\dots,m,\ j = 1,\dots,n,\ t = 1,\dots,k^r_i,\ j' \in J_{ijt} \\
& z_{ij} \ge x_{jti'}, \quad i = 1,\dots,m,\ j = 1,\dots,n,\ t = 1,\dots,k^c_j,\ i' \in I_{ijt} \\
& z_{ij} \in \{0,1\}, \quad i = 1,\dots,m,\ j = 1,\dots,n \\
& y_{itj} \in \{0,1\}, \quad i = 1,\dots,m,\ t = 1,\dots,k^r_i,\ j = e^r_{i,t},\dots,l^r_{i,t} \\
& x_{jti} \in \{0,1\}, \quad j = 1,\dots,n,\ t = 1,\dots,k^c_j,\ j = e^c_{j,t},\dots,l^c_{j,t},
\end{aligned}
$$

jonka kohdefunktio f voidaan valita vapaasti.

8 Numeerinen ratkaiseminen

Tässä luvussa tarkastellaan edellä esiteltyjen pulmapelien numeerista ratkaisemista. Erityisesti keskitytään siihen, miten esimerkiksi tehtävän koko ja ratkaisijan valinta vaikuttavat mallien ratkaisemiseen. Aluksi käytetyt ratkaisijat esitellään ja eri pulmapelejä vertaillaan yleisesti keskenään. Tämän jälkeen tarkastellaan vielä tarkemmin sekä Eternity-pulmapelejä että sudokuja. Tarkasteltavista pulmapeleistä on tehty GAMS-mallit [15], jotka on esitetty liitteessä A.

8.1 Ratkaisijat ja mallien yleinen vertailu

Mallien ratkaisemiseen on käytetty GAMS-ohjelmiston versiota 45.7.0 ja ratkaisijoita SCIP [4] (versio 8.0), LINDOGlobal [27] (versio 14.0.5099.279), CPLEX [7] (versio 22.1.1.0) sekä DICOPT [20] (versio 3.170). Näistä ratkaisijoista SCIP ja LINDOGlobal ovat globaaleja ratkaisijoita, kun taas CPLEX ja DICOPT ovat lokaaleja ratkaisijoita. SCIP ja LINDOGlobal on valittu globaaleiksi ratkaisijoiksi siksi, että ne ovat keskenään melko erilaisia menetelmiä, mutta molemmat ratkaisevat kattavasti erilaisia tehtävätyyppejä (esim. BLP, MILP ja MINLP). Puolestaan CPLEX on hyvin tunnettu lineaaristen tehtävien ratkaisija, minkä vuoksi se on valittu lineaaristen tehtävien lokaaliksi ratkaisijaksi. Koska osalle pulmapeleistä on esitetty myös MINLP-malli, on epälineaarisille sekalukuoptimointitehtäville soveltuva lokaali ratkaisija DICOPT valittu mukaan. Globaalit ratkaisijat löytävät aina globaalin optimin, kunhan laskenta-aikaa on käytössä tarpeeksi. Lokaaleilla ratkaisijoilla globaali optimi voidaan löytää esimerkiksi aina silloin, kun tehtävä on lineaarinen. Osalla ratkaisijoista myös esimerkiksi epälineaarisen tehtävän konveksisuus takaa globaalin optimin löytymisen. Muussa tapauksessa lokaalin ratkaisijan antaman ratkaisun voidaan taata olevan ainoastaan lokaali optimi. Näin ollen yleisille epälineaarisille tehtäville ei ratkaisun optimaalisuutta voida lokaalilla ratkaisijalla taata.

Tämän aliluvun mallien yleisessä vertailussa GAMS-mallit on muodostettu nimenomaan niille pulmapeleille, jotka on esitetty tutkielman kuvissa, kun taas seuraavissa aliluvuissa ratkaistaan myös muunlaisia Eternity II -pulmapelejä ja sudokuja. Toisin sanoen esimerkiksi kakuron GAMS-malli mallintaa kuvan 23 kakuroa. Eternity II -palapeleistä on tässä tarkasteltu ainoastaan 2×2-kokoisia versioita, sillä yleisessä vertailussa tehtävän koko on haluttu pitää pienenä. Tarkemmat viitteet ratkaistuja ongelmia esittäviin kuviin on koottu taulukkoon 5. Tarkastelusta on jätetty pois Eternity I -palapeli, Eternity II -palapelin mallinnustapa 1 ja japanilainen ristikko. Eternity I -palapelin kohdalla syy on epäsymmetristen palojen koodaamisen työläys ja vaikeus sekä se, että kolmen kuukauden laskentakaan ei ole tuottanut pulmapelille ratkaisua viitteessä [6]. Lisäksi pelkästään 2×2-kokoisessa Eternity II -palapelissä on tavan 1 mallissa jo 320 parametria ja tämä tekee mallista melko työlään koodata. Koska Eternity II -palapelille on esitelty myös huomattavasti vähemmän parametreja sisältävä tavan 2 malli, on numeerisissa tuloksissa tarkasteltu ainoastaan sille saatuja tuloksia. Japanilainen ristikko on puolestaan päätetty jättää numeerisista tarkasteluista pois mallin koodaamisen monimutkaisuuden vuoksi. Nimittäin mallin rajoitteet ja erilaiset indeksijoukot ovat haastavia kirjoittaa vähänkin isommalle tehtävälle.

Tutkielmassa on esitelty kahden tyyppisiä optimointitehtäviä: "tavallisia" koh-

defunktiollisia tehtäviä sekä yhtälöryhmiä, joissa kohdefunktiota ei tarvita. Jälkimmäisessä tilanteessa on GAMS-malleihin valittu maksimoitavaksi kohdefunktioksi vakiofunktio 1. Kirjoitetut mallit on syötetty NEOS-serverin [8, 12, 19] versioon 6.0 ratkaistavaksi, koska GAMS-ohjelmiston demolisenssi ei pysty ratkaisemaan seuraavassa aliluvussa tarkasteltavia 3×3-kokoisia ja sitä suurempia Eternity II -palapelejä.

Taulukko 5: Mallien laskenta-ajat sekunteina. Viimeisessä sarakkeessa on viite tarkasteltavan pulmapelin kuvaan.

Malli	**SCIP**	**LINDOGlobal**	**Lokaali ratkaisija**	**Kuva**
Yksinkertainen tiilitysongelma	0,092	0,031	0,028 (CPLEX)	2
Yleinen tiilitysongelma	0,092	0,031	0,029 (CPLEX)	6
Eternity II, tapa 2, epälin. (4 palaa)	0,109	0,079	0,061 (DICOPT)	13
Eternity II, tapa 2, lin. (4 palaa)	0,102	0,069	0,037 (CPLEX)	13
Ei reunoja -Eternity, epälin. (4 palaa)	0,119	0,190	0,127* (DICOPT)	15
Ei reunoja -Eternity, lin. (4 palaa)	0,154	0,250	0,035 (CPLEX)	15
Kolmio-Eternity, epälin. (4 palaa)	0,110	0,045	0,086 (DICOPT)	16
Kolmio-Eternity, lin. (4 palaa)	0,140	0,104	0,031 (CPLEX)	16
Perinteinen sudoku, tapa 1	0,109	0,058	0,038 (CPLEX)	19
Perinteinen sudoku, tapa 2	0,098	0,037	0,031 (CPLEX)	19
Perinteinen sudoku, tapa 3	0,191	0,120	0,082 (CPLEX)	19
Sudoku-x	0,115	0,114	0,044 (CPLEX)	20
Pariton-parillinen sudoku	0,124	1,057	0,040 (CPLEX)	21
Killer sudoku	0,842	1,139	0,625 (CPLEX)	22
Kakuro	0,226	0,082	0,040 (CPLEX)	23

* löydetty ratkaisu ei ole globaali optimi

Taulukkoon 5 on koottu yleisessä vertailussa GAMS-mallien ratkaisemiseen menevät laskenta-ajat sekunteina. Laskenta-aika on se aika, joka kuluu käännöksen valmistumisesta tuloksen saamiseen. Tässä kaikille malleille on käytetty globaaleja ratkaisijoita SCIP ja LINDOGlobal, joilla saatujen ratkaisujen tiedetään olevan globaaleja optimeja, koska menetelmien suoritus päättyi äärellisessä ajassa ilman pakotusta. Lisäksi jokaiselle mallille on käytetty yhtä lokaalia ratkaisijaa, joka lineaarisen mallin tapauksessa on CPLEX ja epälineaarisen tapuksessa DICOPT. Kaikissa muissa paitsi ei reunoja -Eternityn epälineaarisessa mallissa lokaalit ratkaisijat onnistuivat löytämään tehtävälle globaalin optimin eli parhaan mahdollisen ratkaisun. Tämä olikin yleisesti odotettu tulos, sillä tarkasteltujen tehtävien koko on pieni ja näin ollen lokaalit ratkaisijat eivät joudu laskennassa niin helposti vaikeuksiin.

Taulukko 6: Mallien päätösmuuttujien ja rajoitteiden määrät sekä tehtävätyyppi.

Malli	**Päätös-muuttujat**	**Rajoitteet**	**Tehtävä-tyyppi**
Yksinkertainen tiilitysongelma	26	11	BLP
Yleinen tiilitysongelma	34	12	BLP
Eternity II, tapa 2, epälin. (4 palaa)	44	92	MINLP
Eternity II, tapa 2, lin. (4 palaa)	92	188	MILP
Ei reunoja -Eternity, epälin. (4 palaa)	44	76	MINLP
Ei reunoja -Eternity, lin. (4 palaa)	92	172	MILP
Kolmio-Eternity, epälin. (4 palaa)	36	108	MINLP
Kolmio-Eternity, lin. (4 palaa)	72	180	MILP
Perinteinen sudoku, tapa 1	729	324	BLP
Perinteinen sudoku, tapa 2	729	324	BLP
Perinteinen sudoku, tapa 3	1215	1701	MILP
Sudoku-x	729	342	BLP
Pariton-parillinen sudoku	729	486	BLP
Killer sudoku	729	594	BLP
Kakuro	360	360	BLP

Tarkastellaan taulukon 5 laskenta-aikoja lähemmin. Kaikkien mallien laskenta-ajat ovat välillä 0,028–1,139 sekuntia. Lisäksi keskenään saman kategorian pulmapelien laskenta-ajat ovat hyvin lähellä toisiaan. Esimerkiksi kaikki eri Eternity-mallien laskenta-ajat ovat ratkaisijalla SCIP 0,052 sekunnin päässä toisistaan. Kuitenkin perinteisen sudokun kolmesta mallista erottuu mallinnustavan 3 malli. Nimittäin jokaisella ratkaisijalla perinteisen sudokun tavan 3 mallin laskenta-aika on noin kaksin-

tai kolminkertainen perinteisen sudokun mallinnustapoihin 1 ja 2 verrattuna. Jonkin verran pidempi laskenta-aika perinteisen sudokun mallinnustavassa 3 johtuu luultavasti ainakin osittain mallin suuresta rajoitteiden ja muuttujien määrästä. Tarkemmin kaikkien ratkaistujen pulmapelimallien muuttujien ja rajoitteiden määrät sekä tehtävätyypit onkin esitetty taulukossa 6.

Neljässä mallissa ratkaisijalla LINDOGlobal saadut laskenta-ajat ovat hieman pidempiä kuin ratkaisijalla SCIP saadut. Puolestaan 11 mallissa tilanne on päinvastainen. Lisäksi molemmat ratkaisijat takaavat globaalin optimin ja toimivat kaikille tarkastelussa mukana oleville tehtävätyypeille (BLP, MILP ja MINLP). Loppupeleissä ratkaisijoiden SCIP ja LINDOGlobal tuottamat laskenta-ajat ovat kuitenkin kaikille malleille samaa suuruusluokkaa, joten erot laskenta-ajoissa ovat hyvin pieniä. Näin ollen varsinaisia johtopäätöksiä ei tarkastelluille pulmapeleille toimivammasta globaalista ratkaisijasta pystytä tekemään, vaikka LINDOGlobal onkin melko monessa tapauksessa aivan vähän ratkaisijaa SCIP nopeampi. Lokaalilla ratkaisijalla DICOPT saadut laskenta-ajat ovat hyvin lähellä globaaleilla ratkaisijoilla saatuja laskenta-aikoja. Kuitenkaan ratkaisijan DICOPT laskenta-aika ei ole missään mallissa molempia globaaleja ratkaisijoita pidempi. Puolestaan lokaalilla ratkaisijalla CPLEX saadut laskenta-ajat ovat kaikilla tarkastelluilla lineaarisilla malleilla lyhyemmät kuin globaaleilla ratkaisijoilla saadut, vaikka jälleen saadut laskenta-ajat ovat samaa suuruusluokkaa. Voidaan todeta, että lineaarisille tehtäville kannattaa käyttää ratkaisijaa CPLEX, koska kyseinen ratkaisija vaikuttaa pystyvän hyödyntämään tehtävän lineaarisuutta paremmin kuin yleiset globaalit menetelmät. On myös hyvä muistaa, että lineaarisissa tehtävissä CPLEX takaa ratkaisun globaalin optimaalisuuden. Kaiken kaikkiaan voidaan todeta, että ratkaisijan valinnalla ei näytä olevan kovinkaan suurta vaikutusta melko pienten pulmapelien ratkaisemisessa.

8.2 Eternity-pulmapelit

Tässä aliluvussa tarkastellaan Eternity-pulmapelejä tarkemmin ja erityisesti keskitytään siihen, miten pulmapelin koko, tyyppi ja värien määrä vaikuttaa ratkaisun löytymiseen sekä esimerkiksi laskenta-aikaan. Vertailussa ovat perinteinen Eternity II -palapeli, ei reunoja -Eternity ja kolmio-Eternity. Erityisesti kyseisten pulmapelimallien epälineaarisia versioita vertaillaan keskenään. Perinteisen Eternity II -pulmapelin mallinnustavan 2 avulla tarkastellaan myös sitä, miten epälineaarisen ja linearisoidun mallin tulokset eroavat toisistaan.

Vertailu aloitetaan tarkastelemalla 2×2-, 3×3- ja 4×4-kokoisia epälineaarisia malleja Eternity-pulmapeleistä. Tarkasteluissa on perinteisessä Eternity II -pulmapelissä sekä ei reunoja -Eternityssä reunan harmaan lisäksi 4 väriä ja kolmio-

Eternityssä 3 väriä. Ratkaisijoina on käytetty menetelmiä SCIP sekä LINDOGlobal, ja mallien laskenta-ajat on koottu taulukkoon 7. Kyseisestä taulukosta havaitaan, että laskenta-aika nousee pulmapelin koon kasvaessa sekä ratkaisijalla SCIP että LINDOGlobal. Esimerkiksi kun käytetään ratkaisijaa SCIP ja siirrytään saman tehtävän sisällä 2×2-kokoisesta palapelistä yhtä suurempaan eli 3×3-kokoiseen palapeliin, niin laskenta-aika yli kuusinkertaistuu perinteisellä Eternity II -pulmapelillä ja lähes kymmenkertaistuu kolmio-Eternityllä. Puolestaan ei reunoja -Eternityllä laskenta-aika kasvaa lähes 17-kertaiseksi. Ratkaisijalla LINDOGlobal pulmapelin kasvu 2×2-kokoisesta 3×3-kokoiseen kasvattaa laskenta-ajan perinteisellä Eternity II -palapelillä yli 11-kertaiseksi, kolmio-Eternityllä yli 47-kertaiseksi ja ei reunoja -Eternityllä yli 110-kertaiseksi. Kun siirrytään 3×3-kokoisista malleista 4×4-kokoisiin malleihin, ovat laskenta-ajat ei reunoja -Eternityllä ja kolmio-Eternityllä yli kymmenkertaiset sekä perinteisellä Eternity II -palapelillä lähes 19-kertaiset ratkaisijalla SCIP. Vastaavasti ratkaisijaa LINDOGlobal käytettäessä ovat laskenta-ajat perinteisellä Eternity II -palapelillä yli 226-kertaiset, kolmio-Eternityllä lähes 139-kertaiset ja ei reunoja -Eternityllä lähes 262-kertaiset.

Molemmilla ratkaisijoilla kaiken kokoisissa malleissa ei reunoja -Eternity on hitain eli vaikein ratkaista. Puolestaan ratkaisijalla SCIP 2×2- ja 3×3-kokoisissa malleissa perinteinen Eternity II -palapeli on nopein eli helpoin ratkaista ja 4×4-kokoisissa malleissa kolmio-Eternity. Ratkaisijalla LINDOGlobal 2×2-kokoisista malleista kolmio-Eternity on nopein ratkaista sekä 3×3- ja 4×4-kokoisista malleista perinteinen Eternity II. Kaiken kaikkiaan ratkaisija SCIP tuottaa ratkaisijaa LINDOGlobal nopeammin ratkaisun kaikissa muissa malleissa paitsi 2×2-kokoisissa perinteisen Eternity II -palapelin ja kolmio-Eternityn malleissa. Mitä suuremmaksi pulmapelin koko kasvaa, sitä suuremmaksi kasvaa myös ratkaisijoiden välinen laskenta-aikojen ero. Esimerkiksi 3×3-kokoisella ei reunoja -Eternityllä ratkaisijan LINDOGlobal laskenta-aika on yli yhdeksän kertaa suurempi kuin ratkaisijalla SCIP saatu laskenta-aika, mutta 4×4-kokoiselle ei reunoja -Eternitylle ratkaisijan LINDOGlobal laskenta-aika on jo lähes 248-kertainen ratkaisijan SCIP laskenta-aikaan nähden. Ratkaisijaa SCIP voidaan siis pitää ratkaisijaa LINDOGlobal toimivampana vaihtoehtona Eternity-pulmapeleille.

Tuntuisi luontevalta olettaa, että ei reunoja -Eternity olisi helpompi ratkaista kuin perinteinen Eternity II. Tämä johtuu siitä, että ei reunoja -Eternity pohjautuu perinteiseen Eternityyn, mutta harmaita reunoja koskevat rajoitteet on poistettu. Tuntuisi myös luontevalta, että kolmio-Eternity olisi helpoin tarkasteltavista kolmesta pulmapelistä. Nimittäin kolmio-Eternityssä paloilla on vain kolme reunaa, jotka pitää sovittaa viereisten palojen reunoihin, kun muissa kahdessa mallissa on paloil-

Taulukko 7: Eri kokoisten epälineaaristen Eternity-mallien laskenta-ajat sekunteina.

Malli	**Ratkaisija**	2×2	3×3	4×4
Eternity II, tapa 2	SCIP	0,109	0,686	12,957
	LINDOGlobal	0,079	0,879	199,204
Ei reunoja -Eternity	SCIP	0,119	2,216	22,193
	LINDOGlobal	0,190	20,993	5494,167
Kolmio-Eternity	SCIP	0,110	1,073	11,716
	LINDOGlobal	0,045	2,128	295,420

la neljä reunaa. Saadut tulokset eivät kuitenkaan vastaa näihin odotuksiin, mikä todennäköisesti johtuu tehtävien pienestä koosta sekä mahdollisesti myös siitä, että tehdyt odotukset eivät vastaa todellisuutta. Kaiken kaikkiaan voidaan kuitenkin todeta, että tehdyn tarkastelun perusteella ei reunoja -Eternity on vaikein ratkaista. Lisäksi mallien koon kasvaessa kasvaa luonnollisesti laskenta-aikakin. Kasvu ei kuitenkaan ole lineaarista, vaan eksponentiaalista.

Taulukko 8: Eri kokoisten epälineaaristen ja lineaaristen Eternity II -mallien laskenta-ajat sekunteina.

Malli	**Ratkaisija**	2×2	3×3	4×4	5×5
Eternity II, tapa 2, epälin.	SCIP	0,109	0,686	12,957	2830,214
	LINDOGlobal	0,079	0,879	199,204	$-$*
Eternity II, tapa 2, lin.	SCIP	0,102	0,710	13,265	3803,484
	LINDOGlobal	0,069	2,339	404,116	$-$*
	CPLEX	0,037	0,139	1,235	479,620

* ratkaisija ei löytänyt ratkaisua kahdeksassa tunnissa

Taulukossa 8 on esitetty neljän värin ja harmaan reunan perinteisen Eternity II -pulmapelin laskenta-aikoja eri kokoisille mallinnustavan 2 epälineaarisille ja lineaarisille malleille eri ratkaisijoilla. Tarkastellaan näitä tuloksia lähemmin. Kuten edelläkin, pulmapelin koon kasvaessa kasvaa laskenta-aikakin jokaisella ratkaisijalla, ja tämä kasvu on eksponentiaalista. Pienillä 2×2- ja 3×3-kokoisilla malleilla laskenta-ajat ovat melko samaa suuruusluokkaa ja alle sekunnin kaikilla ratkaisijoilla paitsi menetelmällä LINDOGlobal lineaarisen 3×3-kokoisen Eternity II -palapelin mallissa, jossa laskenta-aika on 2,339 sekuntia. Myös 4×4-kokoisen epälineaarisen ja lineaarisen Eternity II -pulmapelin laskenta-ajat ovat ratkaisijalla SCIP samaa suuruusluokkaa, noin kymmenen sekuntia. Kuitenkin vastaavalle pulmapelille havaitaan ero epälineaarisen ja lineaarisen mallin välillä ratkaisijalla LINDOGlobal, koska li-

neaarisen mallin laskenta-aika on yli kaksinkertainen epälineaarisen mallin laskenta-aikaan verrattuna. Lisäksi, koska laskenta-ajat ovat 4×4-kokoisilla Eternity II -pulmapeleillä ratkaisijalla LINDOGlobal satojen sekuntien luokkaa, ovat laskenta-ajat merkittävästi ratkaisijalla SCIP saatuja laskenta-aikoja pidemmät. Kuitenkin 5×5-kokoisilla malleilla ratkaisijalla SCIP on sekä epälineaarisen että lineaarisen mallin laskenta-aika tunnin luokkaa eli hyvin merkittävästi edellisiä pidempi. Ratkaisija LINDOGlobal ei ratkaissut epälineaarista eikä lineaarista 5×5-kokoista Eternity II -palapeliä kahdeksassa tunnissa. Myös 6×6-kokoista perinteistä Eternity II -pulmapeliä on yritetty ratkaista, mutta laskenta on keskeytetty ratkaisijoilla SCIP, LINDOGlobal ja CPLEX kahdeksan tunnin kohdalla tuloksettomana.

Taulukosta 8 huomataan, että ainoastaan 2×2-kokoisilla malleilla lineaarinen malli antaa epälineaarista mallia nopeammat laskenta-ajat globaaleilla ratkaisijoilla SCIP ja LINDOGlobal. Kuitenkin ero on ratkaisijalla SCIP vain 0,007 sekuntia ja ratkaisijalla LINDOGlobal vain 0,010 sekuntia, eli erot ovat hyvin pienet. Muun kokoisilla Eternity II -pulmapeleillä epälineaarinen malli ratkeaa globaaleilla ratkaisijoilla lineaarista mallia nopeammin. Erityisesti 5×5-kokoisilla malleilla havaitaan selkeä ero, sillä ratkaisijalla SCIP on lineaarisen mallin laskenta-aika 973,27 sekuntia epälineaarisen mallin laskenta-aikaa pidempi. On kuitenkin hyvä huomata, että lokaalilla ratkaisijalla CPLEX saadut laskenta-ajat ovat merkittävästi globaaleilla ratkaisijoilla saatuja laskenta-aikoja lyhyemmät erityisesti suuremmilla Eternity II -malleilla. Tehtyjen havaintojen perusteella tehokkain ratkaisija Eternity II -pulmapelin epälineaariseen malliin on SCIP ja lineaariseen mallin CPLEX.

Vaikka yleisesti lineaariset mallit ovat epälineaarisia helpompia ratkaista, niin linearisointi tuo mukanaan myös lisämuuttujia ja -rajoitteita. Esimerkiksi 5×5-kokoisessa lineaarisessa mallissa on 24000 muuttujaa ja 48000 rajoitetta enemmän kuin vastaavassa epälineaarisessa mallissa, mikä tarkoittaa 31 kertaa enemmän muuttujia ja lähes 19 kertaa enemmän rajoitteita. Hurja muuttujien ja rajoitteiden määrien kasvu selittää sitä, miksi lineaarinen Eternity II -palapelin malli on epälineaarista vaikeampi ratkaista varsinkin, kun tarkastellaan alkuperäistä 16×16-kokoista palapeliä. Tästä syystä jatkotarkasteluissa keskitytään nimenomaan perinteisen Eternity II -pulmapelin epälineaariseen versioon. Jatkotarkastelut tehdään myös globaalilla ratkaisijalla SCIP, sillä sen laskenta-ajat ovat etenkin suuremmilla Eternity II -pulmapeleillä merkittävästi toista globaalia ratkaisijaa LINDOGlobal lyhyemmät.

Tarkastellaan seuraavaksi värien määrän vaikutusta Eternity II -pulmapelien ratkaisemiseen. Tarkastelu tehdään ainoastaan 4×4-kokoiselle perinteisen Eternity II -pulmapelin epälineaariselle mallille ja värien määrille neljä, kuusi ja kahdeksan, mis-

sä harmaata ulkoreunaa ei lasketa yhdeksi väriksi. Tulokset on koottu taulukkoon 9. Havaitaan, että 4×4-kokoisessa mallissa laskenta-aika kasvaa 1,07 kertaiseksi (0,849 sekuntia) siirryttäessä neljästä väristä kuuteen ja 1,29 kertaiseksi (3,963 sekuntia) siirryttäessä kuudesta väristä kahdeksaan. Laskenta-ajan kasvu on kuitenkin niin vähäistä, että käytännössä värien määrällä ei ole vaikutusta erittäin pienien Eternity II -palapelien ratkaisemiseen ja sitä kautta laskenta-aikoihin. Tämä johtuu siitä, että perinteisen Eternity II -palapelin mallinnustavassa 2 värien määrä ei vaikuta muuttujien, parametrien eikä rajoitteiden määrään.

Taulukko 9: Perinteisen Eternity II -palapelin laskenta-ajat, kun värien määrä muuttuu ja malli on epälineaarinen. Ratkaisijana on SCIP.

Värien määrä	**Laskenta-aika**	**Koko**
4	12,957	4×4
6	13,806	4×4
8	17,769	4×4

Tarkastellaan vielä sitä, helpottaako tarkempien kulma-, reuna- ja sisäpaloja koskevien rajoitteiden lisääminen Eternity II -pulmapelien ratkaisua 4×4-kokoisissa epälineaarisissa malleissa, joissa on 4 väriä reunan harmaan lisäksi. Tällöin voidaan poistaa alkuperäisen mallin rajoitteet (3), joissa taataan jokaisen palan k olevan korkeintaan yhdessä paikassa. Nyt siis korvataan alkuperäiset rajoitteet (3) uusilla tarkemmilla rajoitteilla, joissa huomioidaan ylipäätään ne paikat, joihin pala on mahdollista sijoittaa. Nimittäin esimerkiksi kulmapalaa ei voi sijoittaa muualle kuin pelilaudan kulmiin. On siis vaadittava, että muuttuja x_{ijk} on nolla, jos palaa k ei ole mahdollista sijoittaa ruutuun (i, j). Olkoon kulmapalojen joukko K_{kulma}, reunapalojen joukko K_{reuna} ja sisäpalojen joukko K_{sisa}. Lisättävät rajoitteet ovat perinteiselle 4×4-kokoiselle Eternity II -palapelille muotoa

$$
\begin{aligned}
&x_{11k} + x_{14k} + x_{41k} + x_{44k} \leq 1, \quad k \in K_{kulma} \\
&x_{12k} + x_{13k} + x_{21k} + x_{24k} + x_{31k} + x_{34k} + x_{42k} + x_{43k} \leq 1, \quad k \in K_{reuna} \\
&x_{22k} + x_{23k} + x_{32k} + x_{33k} \leq 1, \quad k \in K_{sisa} \\
&x_{12k} + x_{13k} + x_{21k} + x_{24k} + x_{31k} + x_{34k} + x_{42k} + x_{43k} + x_{22k} + x_{23k} + \\
&\qquad x_{32k} + x_{33k} = 0, \quad k \in K_{kulma} \\
&x_{11k} + x_{14k} + x_{41k} + x_{44k} + x_{22k} + x_{23k} + x_{32k} + x_{33k} = 0, \quad k \in K_{reuna} \\
&x_{11k} + x_{14k} + x_{41k} + x_{44k} + x_{12k} + x_{13k} + x_{21k} + x_{24k} + x_{31k} + x_{34k} + \\
&\qquad x_{42k} + x_{43k} = 0, \quad k \in K_{sisa}
\end{aligned}
$$

ja 4×4-kokoiselle kolmio-Eternitylle muotoa

$$
\begin{aligned}
&x_{14k} + x_{41k} + x_{47k} \leq 1, \quad k \in K_{kulma} \\
&x_{23k} + x_{25k} + x_{32k} + x_{36k} + x_{42k} + x_{43k} + x_{44k} + x_{45k} + x_{46} \leq 1, \quad k \in K_{reuna} \\
&x_{24k} + x_{33k} + x_{34k} + x_{35k} \leq 1, \quad k \in K_{sisa} \\
&x_{23k} + x_{25k} + x_{32k} + x_{36k} + x_{42k} + x_{43k} + x_{44k} + x_{45k} + x_{46} + x_{24k} + \\
&\qquad x_{33k} + x_{34k} + x_{35k} = 0, \quad k \in K_{kulma} \\
&x_{14k} + x_{41k} + x_{47k} + x_{24k} + x_{33k} + x_{34k} + x_{35k} = 0, \quad k \in K_{reuna} \\
&x_{14k} + x_{41k} + x_{47k} + x_{23k} + x_{25k} + x_{32k} + x_{36k} + x_{42k} + x_{43k} + x_{44k} + \\
&\qquad x_{45k} + x_{46} = 0, \quad k \in K_{sisa}.
\end{aligned}
$$

Esimerkiksi kulmapalarajoitteet takavat sen, että jokainen kulmapala sijoitetaan korkeintaan yhteen kulmaan. Kuitenkin jälleen kohdefunktion maksimointi varmistaa, että jokainen kulmapala sijoitetaan täsmälleen yhteen kulmista. Lisättävien rajoitteiden ansiosta osa muuttujista saadaan automaattisesti nolliksi, sillä esimerkiksi kulmapalat eivät ikinä voi sijaita laudan keskellä.

Kun uudet rajoitteet lisätään, saadaan ratkaisijalla SCIP perinteisen Eternity II -palapelin epälineaarisen mallin laskenta-ajaksi 1,708 sekuntia ja kolmio-Eternityn laskenta-ajaksi 0,835 sekuntia. Kulma-, reuna- ja sisäpalarajoitteiden lisääminen nopeuttaa siis perinteisen Eternityn ratkaisemista 11,249 sekuntia ja kolmio-Eternityn ratkaisemista 10,881 sekuntia. On hyvä huomata, että vaikka lisätyt rajoitteet takaavat osan muuttujien olevan nollia, tuovat uudet ehdot mukanaan myös 16 uutta rajoitetta. Nimittäin 4×4-kokoisella Eternity-palapelillä on poistettavia rajoitteita (3) 16 kappaletta ja uusia tilalle tulevia rajoitteita 32 kappaletta. Kuitenkin rajoitteiden kasvusta huolimatta saadaan 4×4-kokoisten perinteisen Eternity II -palapelin laskenta-aikaa lyhennettyä lähes kymmenesosaan ja kolmio-Eternityn laskenta-aikaa yli kymmenesosaan alkuperäisestä.

8.3 Sudokut

Tässä aliluvussa tarkastellaan vielä edellä esitettyä tarkemmin eri tyyppisten ja vaikeusasteisten sudokujen ratkaisemista. Aluksi vertaillaan yleisesti tämän tutkielman neljää eri sudokutyyppiä: perinteistä sudokua, sudoku-x:ää, pariton-parillinen sudokua ja killer sudokua. Vertailussa käytetään taulukon 5 tuloksia. Lisäksi perinteisen sudokun mallinnustavan 3 mallia ei enää oteta mukaan näihin tarkasteluihin, koska kyseinen malli on perinteisen sudokun malleista haastavin ratkaista jokaiselle taulukon 5 ratkaisijalle. Esimerkiksi käytettäessä ratkaisijaa SCIP tuottaa perinteisen

sudokun mallinnustapa 2 pienimmän laskenta-ajan 0,098 sekuntia, kun mallinnustavan 1 laskenta-aika on 0,109 sekuntia. Sudoku-x:n laskenta-aika on 0,115 sekuntia. Puolestaan pariton-parillinen sudokun laskenta-aika on 0,124 sekuntia ja killer sudokun laskenta-aika 0,842 sekuntia. Ainakin siis laskenta-aikojen valossa perinteinen sudoku ja sudoku-x ovat hieman helpompia pulmapelejä ratkaisijalle SCIP kuin pariton-parillinen sudoku ja killer sudoku. Kuitenkin erot ovat niin pieniä (alle sekunnin), että mitään kauaskantoisia johtopäätöksiä ei voida tehdä. Lisäksi esimerkiksi katsomalla lokaalin ratkaisijan CPLEX tuloksia, on sille sudoku-x "haastavampi" ongelma kuin pariton-parillinen sudoku, mutta ero on vain 0,004 sekuntia eli jälleen hyvin pieni. Kuitenkin CPLEX vie jokaisessa sudokumallissa vähemmän laskenta-aikaa kuin kumpikaan globaaleista ratkaisijoista. Näin ollen sudokujen ratkaisemiseen kannattaa käyttää ratkaisijaa CPLEX. Toisaalta, koska laskenta-ajat ovat korkeintaan sekunnin luokkaa, ovat mallit periaatteessa yhtä vaikeita tai tässä tapauksessa helppoja ratkaista.

Kun ei huomioida perinteisen sudokun mallinnustapaa 3, on kaikissa eri sudokumalleissa yhtä monta muuttujaa (729 kappaletta). Lisäksi perinteisen sudokun mallinnustavoissa 1 ja 2 rajoitteiden määrä on sama (324 kappaletta). Perinteisen sudokun mallinnustavassa 3 muuttujia on 1215 ja rajoitteita 1701. Puolestaan sudoku-x:n rajoitteiden määrä on 342, pariton-parillinen sudokun 486 ja killer sudokun 594. Tietokoneelle malli on yleensä sitä haastavampi ja hitaampi ratkaista, mitä enemmän rajoitteita on. Ratkaistuille malleille asia pätee suurimmaksi osaksi, kun ratkaisija on SCIP. Kuitenkin perinteisen sudokun mallinnustavan 3 malli näyttää olevan nopeampi ratkaista kuin killer sudoku, vaikka killer sudokussa on 1107 rajoitetta vähemmän kuin perinteisen sudokun mallinnustavassa 3. Lisäksi vaikka ratkaisijalla SCIP sudoku-x ratkeaa 0,009 sekuntia pariton-parillinen sudokua nopeammin, on asia toisin päin ratkaisijalla CPLEX, jolla mallien välinen laskenta-aikojen ero on 0,004 sekuntia. Huomion arvoista on kuitenkin se, että tarkastellut pulmapelit ovat yksinkertaisia ja kooltaan melko pieniä tehtäviä, joten pienillä eroilla mallien rajoitteiden määrissä ei ole suurta merkitystä ratkaisuaikaan. Kuitenkin, koska perinteisen sudokun mallinnustavassa 3 on merkittävästi enemmän rajoitteita ja muuttujia kuin muissa sudokumalleissa, on tavan 3 laskenta-aikakin muita perinteisen sudokun malleja pidempi.

Vertaillaan seuraavaksi eri vaikeusasteisten sudokujen laskenta-aikoja. Tässä keskitason sudokuilla tarkoitetaan samoja sudokuja, jotka on esitetty pulmapelien mallien yhteydessä olevissa kuvissa. Viitteet näihin kuviin on koottu taulukkoon 5. Vaikealla sudokulla puolestaan tarkoitetaan sellaista sudokua, joka on sudokukirjassa määritelty vaikeaksi. Poikkeuksena on perinteinen sudoku, jonka kohdalla on tutkit-

tu nimenomaan maailman vaikenta sudokua [21]. Kyseisen sudokun on kehittänyt suomalainen matemaatikko Arto Inkala vuonna 2012. Ei ole tiedossa, että kukaan olisi kehittänyt vaikeampaa sudokua kuin Inkala. Eri tasoisten sudokujen laskenta-ajat on esitetty taulukossa 10, kun ratkaisija on SCIP ja CPLEX.

Taulukko 10: Eri tasoisten sudokumallien laskenta-ajat sekunteina.

Malli	**Ratkaisija**	**Keskitasoinen**	**Vaikea**
Perinteinen sudoku, tapa 1	SCIP	0,109	0,579
	CPLEX	0,038	0,075
Perinteinen sudoku, tapa 2	SCIP	0,098	0,134
	CPLEX	0,031	0,037
Perinteinen sudoku, tapa 3	SCIP	0,191	12,491
	CPLEX	0,082	0,490
Sudoku-x	SCIP	0,115	0,117
	CPLEX	0,044	0,046
Pariton-parillinen sudoku	SCIP	0,124	0,133
	CPLEX	0,040	0,051
Killer sudoku	SCIP	0,842	0,273
	CPLEX	0,625	0,201

Niin kuin jo edellä nähtiin, keskitasoisissa sudokuissa perinteinen sudoku (mallinnustavat 1 ja 2) sekä sudoku-x olivat lähes aina helpompia kuin parillinen-pariton sudoku ja killer sudoku. Kuitenkaan varsinaisesti laskenta-ajoissa ei ollut mitään suuria eroja. Taulukosta 10 voidaan havaita, että vaikea killer sudoku on hieman helpompi ratkaista kuin keskitasoinen killer sudoku sekä ratkaisijalla SCIP että CPLEX. Muissa malleissa vaikeiden sudokujen ratkaiseminen kestää molemmilla ratkaisijoilla pidempään kuin keskitasoisten sudokujen, eli vaikeat sudokut ovat keskitasoisia vaikeampia. Kuitenkin yleisesti alle sekunnin eroa laskenta-ajoissa ei voida pitää merkitsevänä, ja näin ollen tehtävät ovat aivan yhtä helppoja ratkaista. Kuitenkin huomionarvoista on se, että perinteisen sudokun mallinnustavalla 3 kestää vaikean sudokun ratkaiseminen menetelmällä SCIP yli 65-kertaisen ajan keskitasoiseen sudokuun verrattuna. Ratkaisijalla CPLEX vastaava laskenta-aikojen ero on ainoastaan lähes kuusinkertainen. Lisäksi erityisen kiinnostavaa on, että sudokukirjan keskitasoinen sudoku sekä maailman vaikein sudoku ovat laskenta-ajan perusteella tietokoneelle lähes samantasoisia ongelmia mallinnustavoilla 1 ja 2, mutta mallinnustavalla 3 erot ovat merkittävät erityisesti ratkaisijalla SCIP.

On hyvä huomata, että edellä kuvatut tulokset on saatu vain yksittäisiä sudoku-

ja tarkastelemalla ja kattavampia johtopäätöksiä varten pitäisi ratkaista useita eri sudokuja. Lisäksi, vaikka saatujen numeeristen tulosten perusteella on esimerkiksi mahdollista todeta, että keskitasoinen perinteinen sudoku (mallinnustapa 2) on hieman helpompi ongelma kuin keskitasoinen pariton-parillinen sudoku, niin kyseessä on kuitenkin vain 0,026 sekunnin ero laskenta-ajoissa ratkaisijalla SCIP ja 0,009 sekunnin ero ratkaisijalla CPLEX. Erot ovat siis hyvin vähäisiä. Kaiken kaikkiaan laskenta-aikojen erot molemmilla ratkaisijoilla kaikkien tarkasteltujen sudokujen välillä ovat hyvin pienet. Myös perinteisen sudokun mallinnustavan 3 laskenta-ajat ovat hyvin pientä suuruusluokkaa, vaikka ne erottuvatkin muista perinteisten sudokujen laskenta-ajoista.

9 Yhteenveto

Tässä työssä on tarkasteltu erilaisia pulmapelejä, niiden matemaattista mallintamista sekä ratkaisemista. Erityisesti on tutkittu erilaisia tiilitys- ja reunasovitusongelmia sekä sudokuja ja summafunktioon perustuvia pulmapelejä. Tiilitys- ja reunasovitusongelmien keskiössä ovat olleet Eternity-pulmapelit, jotka ovat alkuperäisen kokoisina lähes mahdottomia ratkaista. Nämä neljä pulmapelityyppiä luovat laajan kuvan erilaisista matemaattisista pulmapeleistä. Tarkastelluista ongelmista NP-täydellisiä ovat tiilitys- ja reunasovitusongelmat sekä japanilainen ristikko, joka on esimerkki summafunktioon perustuvasta pulmapelistä. Näin ollen suurin osa työssä tarkastelluista pulmapeleistä tiedetään jo teoriassa haastaviksi. Kuitenkin NP-täydellisille ongelmille voidaan löytää ratkaisuja järkevässä äärellisessä ajassa, kunhan rajaudutaan vain riittävän pienen kokoisiin ongelmiin.

Aluksi tutkielmassa on käyty läpi jokaista tarkasteltavaa pulmapeliä mallintava optimointitehtävä. Perinteiselle Eternity II -palapelille ja perinteiselle sudokulle on lisäksi esitetty muutama vaihtoehtoinen malli. Jokaiselle pulmapelille on saatu kirjoitettua ainakin yksi lineaarinen malli, jossa on käytetty binäärisiä muuttujia mahdollisesti yhdessä jatkuvien muuttujien kanssa. Pääsääntöisesti pulmapelien matemaattiset mallit on esitetty nimenomaan binäärisinä lineaarisina optimointitehtävinä. Esitellyistä tehtävistä on kirjoitettu GAMS-mallit, jolloin tehtävien ratkaisemiseen on voitu käyttää valmiita ohjelmistoja. Työn lopussa eri mallien numeerisia tuloksia on vertailtu keskenään.

Tutkielmassa erilaisia pulmapelejä on mallinnettu luonnollisestikin erilailla. Lisäksi tietty pulmapeli, kuten Eternity II ja sudoku, on mallinnettu usealla eri tavalla, mikä on johtanut useaan vaihtoehtoiseen optimointitehtävään. Mallinnustavasta riippuen muuttujien, parametrien ja rajoitteiden määrissä voi kuitenkin olla suu-

ria eroja. Tämän takia kaikki mallinnustavat eivät välttämättä ole yhtä helppoja ratkaista, sillä mitä enemmän tehtävässä on päätösmuuttujia, parametreja ja rajoitteita, sitä hankalampi se on. Lisäksi epälineaariset mallit ovat tavallisesti lineaarisia malleja hankalampia ratkaista. Esimerkiksi perinteisen sudokun kohdalla yhden kolmesta mallinnustavasta on havaittu olevan muita perinteiselle sudokulle esiteltyjä mallinnustapoja huonompi. Nimittäin mallin suuri päätösmuuttujien ja rajoitteiden määrä nostaa laskenta-ajan noin kaksinkertaiseksi muiden perinteisen sudokun mallien laskenta-aikoihin verrattuna.

Tässä työssä on erityisesti tarkasteltu pieniä pulmapelejä ja kyseisten mallien numeerisista tuloksista on havaittu, että vaikka eri pulmapelien mallit ovat erilaisia, ratkeavat ne pääsääntöisesti (hieman isompia Eternity II -palapelejä lukuun ottamatta) saman suuruusluokan ajassa. Lisäksi on havaittu laskenta-aikojen eksponentiaalinen kasvu NP-täydellisten pulmapelien koon kasvaessa. Erityisesti NP-täydellisen Eternity II -pulmapelin ratkaisemisessa on onnistuttu pienentämällä alkuperäisen pulmapelin kokoa huomattavasti. Kuitenkaan alkuperäiselle suurelle NP-täydelliselle Eternity II -palapelille ei tutkielman malleilla ole mahdollista löytää ratkaisua järkevässä äärellisessä ajassa.

Tutkielmassa on nähty, että matemaattinen optimointi on tehokas keino mallintaa pulmapelejä, joiden koko on melko pieni. Erityisesti on havaittu binäärimuuttujien hyödyllisyys pulmapelejä mallinnettaessa. Kuitenkin binäärimuuttujien suuri määrä voi aiheuttaa haasteita mallien ratkaisemisessa. Erityisesti mallien linearisoinnista aiheutuvat lisäbinäärimuuttujat voivat aiheuttaa sen, että epälineaarinen tehtävä on helpompi ratkaista kuin vastaava lineaarinen tehtävä. Tälläinen havainto on tehty esimerkiksi toisessa perinteisen Eternity II -palapelin mallinnustavoista, sillä kyseisen epälineaarisen tehtävän linearisoinnista aiheutuu miljoonia uusia binäärimuuttujia ja rajoitteita, kun tarkastellaan alkuperäistä suurikokoista Eternity II -palapeliä. Kuten edellä on todettu, tutkielmassa tarkastelluille NP-täydellisille pulmapeleille on onnistuttu löytämään globaalit optimit ainoastaan, koska tehtävien kokoa on pienennetty tarvittaessa huomattavasti. Jotta esimerkiksi alkuperäiselle suurikokoiselle NP-täydelliselle Eternity II -palapelille voitaisiin löytää ratkaisu, tulisi numeeristen ratkaisijoiden ja tietokoneiden kehittyä entisestään. On kuitenkin lähes mahdotonta sanoa, millainen kehitys riittäisi siihen, että 17 vuotta ratkaisemattomana pysynyt Eternity II -pulmapeli saataisiin vihdoin ratkaistua.

Kirjallisuutta

[1] B. Bagchi: Latin squares. *Resonance*, Vol. 17(9), 895–902, 2012.

[2] A. Bartlett, T. Chartier, A. Langville, T. Rankin: An integer programming model for the sudoku problem. *Journal of Online Mathematics and its Applications*, Vol. 8(1), 2008.

[3] R. Berger: The undecidability of the domino problem. *American Mathematical Society*, Vol. 66(1), 1966.

[4] K. Bestuzheva, M. Besancon, W. Chen et al: *The SCIP Optimization Suite 8.0*, Optimization Online, 2021. `http://www.optimization-online.org/DB_HTML/2021/12/8728.html`

[5] R. A. Bosch: Painting by numbers. *Optima*, Vol. 65(1), 16–17, 2001.

[6] J. Burkardt, M. R. Garvie: An integer linear programming approach to solving the Eternity puzzle. *Theoretical Computer Science*, Vol. 975(1), 2023.

[7] CPLEX, IBM ILOG: V12. 1: User's Manual for CPLEX. *International Business Machines Corporation*, Vol. 46(53), 157, 2009. `https://www.ibm.com/docs/en/icos/22.1.1?topic=optimizers-users-manual-cplex`

[8] J. Czyzyk, M. P. Mesnier, J. J. Moré: The NEOS Server. *IEEE Journal on Computational Science and Engineering*, Vol. 5(3), 68–75, 1998.

[9] M. Danesi: Puzzles and Mathematics. *Ahmes' Legacy: Puzzles and the Mathematical Mind*, 1–44, 2018.

[10] R. P. Davies: *An investigation into the solution to, and evalution of, Kakuro puzzles*. Pro gradu -tutkielma, University of South Wales, Iso-Britannia, 2009.

[11] E. D. Demaine, M. L. Demaine: Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. *Graphs and combinatorics*, Vol. 23(1), 195–208, 2007.

[12] E. Dolan: *The NEOS Server 4.0 Administrative Guide*. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[13] S. Emet, K. Joki: *Mixed-integer non-linear programming*. Luentomoniste, Turun yliopisto, 2021.

[14] M. M. Flood: The traveling-salesman problem. *Operations research*, Vol. 4(1), 61–75, 1956.

[15] GAMS Development Corporation. General Algebraic Modeling System (GAMS), Release 44.3.0, 2023.

[16] M. R. Garey, D. S. Johnson, L. Stockmeyer: Some simplified NP-complete problems. *Proceedings of the sixth annual ACM symposium on Theory of computing*, 47–63, 1974.

[17] M. Garvie, J. Burkardt: A new mathematical model for tiling finite regions of the plane with polyominoes. *Contributions to Discrete Mathematics*, Vol. 15(2), 95–131, 2020.

[18] S. W. Golomb: *Polyominoes: puzzles, patterns, problems, and packings*. Princeton University Press, New Jersey, 1996.

[19] W. Gropp, J. J. Moré: Optimization Environments and the NEOS Server. *Approximation Theory and Optimization*, M. D. Buhmann, A. Iserles, eds., Cambridge University Press, 167–182, 1997.

[20] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, E. Kalvelagen: *GAMS/DICOPT: A discrete continuous optimization package*, Vol. 11(1), 2002.

[21] A. Harish: *Can you solve the hardest-ever sudoku?* ABC News Network, 2012. Viitattu 22.1.2024. `https://abcnews.go.com/blogs/headlines/2012/06/can-you-solve-the-hardest-ever-sudoku`

[22] S. Hartmann: Puzzle-solving smartphone puzzle apps by mathematical programming. *Informs Transactions on Education*, Vol. 18(1), 127–141, 2018.

[23] J. Kari: The tiling problem revisited (extended abstract). *Machines, Computations, and Universality*, J. Durand-Lose, M. Margenstern, eds., Lecture Notes in Computer Science, Vol. 4664(1), 72–79, Springer, Berlin, Heidelberg, 2007.

[24] J. Kari: *Tilings and patterns*. Luentomoniste, Turun yliopisto, 2023.

[25] S. Z. Kovalsky, D. Glasner, R. Basri: *A global approach to solving edge-matching puzzles*. arXiv.org, 2015. `https://arxiv.org/pdf/1409.5957.pdf`

[26] H. R. Lewis: Complexity of solvable cases of the decision problem for the predicate calculus. *19th Annual Symposium on Foundations of Computer Science*, 35–47, 1978.

[27] Y. Lin, L. Schrage: The global solver in the LINDO API. *Optimization Methods and Software*, Vol. 25(4–5), 657–668, 2009.

[28] H. Lloyd, M. Crossley, M. Sinclair, M. Amos: *J-POP: Japanese puzzles as optimization problems*. IEEE Transactions on Games, Vol. 14(3), 391–402, 2021.

[29] L. Mingote, F. Azevedo: Colored nonograms: an integer linear programming approach. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, Vol. 5816(1), 213–224, 2009.

[30] C. Moore, J. M. Robson: Hard tiling problems with simple tiles. *Discrete and Computational Geometry*, Vol. 26(1), 573–590, 2001.

[31] N. Oswald, J. Steuding: A hidden orthogonal Latin square in a work of Euler from 1770. *The Mathematical intelligencer*, Vol. 42(4), 23–29, 2020.

[32] M. Pitici: *Archimedes' Stomachion*. Geometric Dissections, 2008. Viitattu 30.1.2024. `https://pi.math.cornell.edu/~mec/GeometricDissections/1.2%20Archimedes%20Stomachion.html`

[33] R. M. Robinson: Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, Vol. 12(1), 177–209, 1971.

[34] F. Salassa, W. Vancroonenburg, T. Wauters, F. Della Croce, G. V. Berghe: *MILP and max-clique based heuristics for the Eternity II puzzle*. arXiv.org, 2017. `https://arxiv.org/pdf/1709.00252.pdf`

[35] E. W. Williams: *Stomachion*. 2002. Viitattu 9.3.2024. `https://mathworld.wolfram.com/Stomachion.html`

[36] A. D. Williams: *The Jigsaw puzzle: piecing together a history*. Berkley Books, New York, 2004.

[37] C. H. Yu, H. L. Lee, L. H. Chen: An efficient algorithm for solving nonograms. *Applied Intelligence*, Vol. 35(1), 18–31, 2011.

[38] H. Yu, Y. Tang, C. Zong: Solving odd even sudoku puzzles by binary integer linear programming. *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, IEEE, 2226–2230, 2016.

A GAMS-mallit

Yksinkertainen tiilitysongelma

```
Sets
j "palat" /1*3/
i "paikat" /1*12/

C(i,j) "palojen mahdolliset paikat" /1.1, 2.1, 3.1, 4.1, 1.2, 2.2,
3.2, 4.2, 5.2, 6.2, 7.2, 8.2, 9.2, 10.2, 1.3, 2.3, 3.3, 4.3, 5.3,
6.3, 7.3, 8.3, 9.3, 10.3, 11.3, 12.3/ ;

Variables
x(i,j) "sijoitetaanko pala j pelilaudan paikkaan i"
z "kohdefunktiomuuttuja" ;

Binary variables x ;

Equations
obj "kohdefunktio"
raj1(j), raj2, raj3, raj4, raj5, raj6, raj7, raj8, raj9;

obj.. z =E= sum((i,j) $(C(i,j)),(x(i,j))) ;
raj1(j).. sum(i $(C(i,j)), x(i,j)) =L= 1 ;
raj2.. x('1','1')+x('1','2')+x('7','2')+x('1','3')+x('4','3')+
x('7','3') =L= 1 ;
raj3.. x('1','1')+x('2','1')+x('1','2')+x('2','2')+x('8','2')+
x('2','3')+x('4','3')+x('5','3')+x('7','3')+x('8','3')+x('10','3') =L= 1 ;
raj4.. x('1','1')+x('2','1')+x('2','2')+x('3','2')+x('9','2')+
x('3','3')+x('5','3')+x('6','3')+x('8','3')+x('9','3')+x('11','3') =L= 1 ;
raj5.. x('2','1')+x('3','2')+x('10','2')+x('6','3')+x('9','3')+
x('12','3') =L= 1 ;
raj6.. x('3','1')+x('4','2')+x('7','2')+x('1','3')+x('4','3')+
x('10','3') =L= 1 ;
raj7.. x('3','1')+x('4','1')+x('4','2')+x('5','2')+x('8','2')+
x('1','3')+x('2','3')+x('5','3')+x('7','3')+x('10','3')+x('11','3') =L= 1 ;
raj8.. x('3','1')+x('4','1')+x('5','2')+x('6','2')+x('9','2')+
x('2','3')+x('3','3')+x('6','3')+x('8','3')+x('11','3')+x('12','3') =L= 1 ;
raj9.. x('4','1')+x('6','2')+x('10','2')+x('3','3')+x('9','3')+
x('12','3') =L= 1;

Model tiilitys "yksinkertainen tiilitysongelma" /all/ ;

Solve tiilitys using MIP maximizing z ;
```

Yleinen tiilitysongelma

```
Sets
j "palat" /1*2/
i "paikat" /1*24/

C(i,j) "palojen mahdolliset paikat" /1.1, 2.1, 3.1, 4.1, 5.1,
6.1, 7.1, 8.1, 9.1, 10.1, 1.2, 2.2, 3.2, 4.2, 5.2, 6.2, 7.2,
8.2, 9.2, 10.2, 11.2, 12.2, 13.2, 14.2, 15.2, 16.2, 17.2,
18.2, 19.2, 20.2, 21.2, 22.2, 23.2, 24.2/ ;

Variables
x(i,j) "sijoitetaanko pala j pelilaudan paikkaan i"
z "kohdefunktiomuuttuja" ;

Binary variables x ;

Equations
obj "kohdefunktio"
raj1, raj2, raj3, raj4, raj5, raj6, raj7, raj8, raj9, raj10,
raj11, raj12 ;

obj.. z =E= sum((i,j) $(C(i,j)),(x(i,j))) ;
raj1.. x('1','1')+x('7','1')+x('1','2')+x('7','2')+x('13','2') =L= 1 ;
raj2.. x('1','1')+x('2','1')+x('8','1')+x('2','2')+x('7','2')+
x('8','2')+x('13','2')+x('14','2')+x('19','2') =L= 1 ;
raj3.. x('1','1')+x('2','1')+x('9','1')+x('3','2')+x('8','2')+
x('9','2')+x('14','2')+x('15','2')+x('20','2') =L= 1 ;
raj4.. x('2','1')+x('10','1')+x('9','2')+x('15','2')+x('21','2') =L= 1 ;
raj5.. x('3','1')+x('7','1')+x('1','2')+x('4','2')+x('7','2')+
x('10','2')+x('16','2')+x('19','2') =L= 1 ;
raj6.. x('3','1')+x('4','1')+x('8','1')+x('1','2')+x('2','2')+
x('5','2')+x('8','2')+x('10','2')+x('11','2')+x('13','2')+x('16','2')+
x('17','2')+x('19','2')+x('20','2')+x('22','2') =L= 1 ;
raj7.. x('3','1')+x('4','1')+x('9','1')+x('2','2')+x('3','2')+
x('6','2')+x('9','2')+x('11','2')+x('12','2')+x('14','2')+x('17','2')+
x('18','2')+x('20','2')+x('21','2')+x('23','2') =L= 1 ;
raj8.. x('4','1')+x('10','1')+x('3','2')+x('12','2')+x('15','2')+
x('18','2')+x('21','2')+x('24','2') =L= 1 ;
raj9.. x('5','1')+x('7','1')+x('4','2')+x('10','2')+x('22','2') =L= 1 ;
raj10.. x('5','1')+x('6','1')+x('8','1')+x('4','2')+x('5','2')+
x('11','2')+x('16','2')+x('22','2')+x('23','2') =L= 1 ;
raj11.. x('5','1')+x('6','1')+x('9','1')+x('5','2')+x('6','2')+
x('12','2')+x('17','2')+x('23','2')+x('24','2') =L= 1 ;
raj12.. x('6','1')+x('10','1')+x('6','2')+x('18','2')+x('24','2') =L= 1 ;

Model tiilitys "yleinen tiilitysongelma" /all/ ;

Solve tiilitys using MIP maximizing z ;
```

Eternity II, mallinnustapa 2

Esitellään aluksi $n \times n$-kokoisen epälineaarisen ja lineaarisen Eternity II -pulmapelin GAMS-mallit. Tämän jälkeen esitetään eri kokoisten mallien kiinnitetyt indeksien ja parametrien arvot.

$n \times n$-kokoinen epälineaarinen malli

```
Scalar
p_max "suurin väri" /p_max/
n "pelilaudan yhden sivun koko" /n/ ;

Sets
k "palat" /1*n^2/
l "palan reunat" /1*4/
i "rivit" /1*n/
j "sarakkeet" /1*n/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;

Variables
x(i,j,k) "onko pala k ruudussa (i,j)"
y(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
r(k,l) "palan k reunan l väri, kun pala on paikallaan"
z "kohdefunktiomuuttuja" ;

Binary variables x, y ;
Positive variables r ;

r.up(k,l) = p_max ;
```

```
Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k,t)
raj13(k,t), raj14(k,t), raj15(k,t) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(k).. sum((i,j), x(i,j,k)) =L= 1 ;
raj2(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj3(k).. y(k,'1')+y(k,'2')+y(k,'3') =L= 1 ;
raj4(k).. r(k,'1') =L= p_max * (1-sum(i,x(i,'1',k))) ;
raj5(k).. r(k,'2') =L= p_max * (1-sum(j,x('1',j,k))) ;
raj6(k).. r(k,'3') =L= p_max * (1-sum(i,x(i,'n',k))) ;
raj7(k).. r(k,'4') =L= p_max * (1-sum(j,x('n',j,k))) ;
raj8(k).. r(k,'1') =E= c(k,'1')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'4')*y(k,'1')+c(k,'3')*y(k,'2')+c(k,'2')*y(k,'3') ;
raj9(k).. r(k,'2') =E= c(k,'2')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'1')*y(k,'1')+c(k,'4')*y(k,'2')+c(k,'3')*y(k,'3') ;
raj10(k).. r(k,'3') =E= c(k,'3')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'2')*y(k,'1')+c(k,'1')*y(k,'2')+c(k,'4')*y(k,'3') ;
raj11(k).. r(k,'4') =E= c(k,'4')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'3')*y(k,'1')+c(k,'2')*y(k,'2')+c(k,'1')*y(k,'3') ;

raj12(k,t) $(ord(k)<>ord(t)).. r(k,'3')-r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), x(i,j,k)*x(i,j+1,t))) ;
raj13(k,t) $(ord(k)<>ord(t)).. -r(k,'3')+r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), x(i,j,k)*x(i,j+1,t)));
raj14(k,t) $(ord(k)<>ord(t)).. r(k,'4')-r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), x(i,j,k)*x(i+1,j,t))) ;
raj15(k,t) $(ord(k)<>ord(t)).. -r(k,'4')+r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), x(i,j,k)*x(i+1,j,t))) ;

Model EternityII "Eternity II tapa 2, epalineaarinen" /all/ ;

Solve EternityII using MINLP maximizing z ;
```

$n \times n$-kokoinen lineaarinen malli

```
Scalar
p_max "suurin väri" /p_max/
n "pelilaudan yhden sivun koko" /n/ ;

Sets
k "palat" /1*n^2/
l "palan reunat" /1*4/
i "rivit" /1*n/
j "sarakkeet" /1*n/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;

Variables
x(i,j,k) "onko pala k ruudussa (i,j)"
y(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
r(k,l) "palan k reunan l väri, kun pala on paikallaan"
s1(i,j,k,t) "binäärimuuttuja 1 linearisointia varten"
s2(i,j,k,t) "binäärimuuttuja 2 linearisointia varten"
z "kohdefunktiomuuttuja" ;

Binary variables x, y, s1, s2 ;
Positive variables r ;

r.up(k,l) = p_max ;
```

```
Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k,t)
raj13(k,t), raj14(k,t), raj15(k,t), raj16(i,j,k,t)
raj17(i,j,k,t), raj18(i,j,k,t), raj19(i,j,k,t) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(k).. sum((i,j), x(i,j,k)) =L= 1 ;
raj2(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj3(k).. y(k,'1')+y(k,'2')+y(k,'3') =L= 1 ;
raj4(k).. r(k,'1') =L= p_max * (1-sum(i,x(i,'1',k))) ;
raj5(k).. r(k,'2') =L= p_max * (1-sum(j,x('1',j,k))) ;
raj6(k).. r(k,'3') =L= p_max * (1-sum(i,x(i,'n',k))) ;
raj7(k).. r(k,'4') =L= p_max * (1-sum(j,x('n',j,k))) ;
raj8(k).. r(k,'1') =E= c(k,'1')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'4')*y(k,'1')+c(k,'3')*y(k,'2')+c(k,'2')*y(k,'3') ;
raj9(k).. r(k,'2') =E= c(k,'2')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'1')*y(k,'1')+c(k,'4')*y(k,'2')+c(k,'3')*y(k,'3') ;
raj10(k).. r(k,'3') =E= c(k,'3')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'2')*y(k,'1')+c(k,'1')*y(k,'2')+c(k,'4')*y(k,'3') ;
raj11(k).. r(k,'4') =E= c(k,'4')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'3')*y(k,'1')+c(k,'2')*y(k,'2')+c(k,'1')*y(k,'3') ;

raj12(k,t) $(ord(k)<>ord(t)).. r(k,'3')-r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), s1(i,j,k,t))) ;
raj13(k,t) $(ord(k)<>ord(t)).. -r(k,'3')+r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), s1(i,j,k,t))) ;
raj14(k,t) $(ord(k)<>ord(t)).. r(k,'4')-r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), s2(i,j,k,t))) ;
raj15(k,t) $(ord(k)<>ord(t)).. -r(k,'4')+r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), s2(i,j,k,t))) ;
raj16(i,j,k,t) $(ord(k)<>ord(t) and ord(j)<n).. s1(i,j,k,t) =G=
1+(x(i,j,k)+x(i,j+1,t)-2) ;
raj17(i,j,k,t) $(ord(k)<>ord(t) and ord(j)<n).. s1(i,j,k,t) =L=
0.5 * (x(i,j,k)+x(i,j+1,t)) ;
raj18(i,j,k,t) $(ord(k)<>ord(t) and ord(i)<n).. s2(i,j,k,t) =G=
1+(x(i,j,k)+x(i+1,j,t)-2) ;
raj19(i,j,k,t) $(ord(k)<>ord(t) and ord(i)<n).. s2(i,j,k,t) =L=
0.5 * (x(i,j,k)+x(i+1,j,t)) ;

Model EternityII "Eternity II, tapa 2, lineaarinen" /all/ ;

Solve EternityII using MIP maximizing z ;
```

2×2-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /2/ ;

Sets
k "palat" /1*4/
l "palan reunat" /1*4/
i "rivit" /1*2/
j "sarakkeet" /1*2/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=0 ;
c('1','2')=0 ;
c('1','3')=1 ;
c('1','4')=2 ;
c('2','1')=0 ;
c('2','2')=0 ;
c('2','3')=3 ;
c('2','4')=1 ;
c('3','1')=0 ;
c('3','2')=0 ;
c('3','3')=4 ;
c('3','4')=3 ;
c('4','1')=0 ;
c('4','2')=0 ;
c('4','3')=2 ;
c('4','4')=4 ;
```

3×3-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /3/ ;

Sets
k "palat" /1*9/
l "palan reunat" /1*4/
i "rivit" /1*3/
j "sarakkeet" /1*3/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=2 ; c('1','2')=2 ; c('1','3')=4 ;
c('1','4')=1 ; c('2','1')=3 ; c('2','2')=3 ;
c('2','3')=0 ; c('2','4')=0 ; c('3','1')=4 ;
c('3','2')=0 ; c('3','3')=4 ; c('3','4')=2 ;
c('4','1')=0 ; c('4','2')=0 ; c('4','3')=3 ;
c('4','4')=2 ; c('5','1')=1 ; c('5','2')=4 ;
c('5','3')=0 ; c('5','4')=0 ; c('6','1')=3 ;
c('6','2')=0 ; c('6','3')=1 ; c('6','4')=2 ;
c('7','1')=0 ; c('7','2')=4 ; c('7','3')=1 ;
c('7','4')=0 ; c('8','1')=0 ; c('8','2')=2 ;
c('8','3')=4 ; c('8','4')=1 ; c('9','1')=0 ;
c('9','2')=3 ; c('9','3')=1 ; c('9','4')=3 ;
```

4×4-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /4/ ;

Sets
k "palat" /1*16/
l "palan reunat" /1*4/
i "rivit" /1*4/
j "sarakkeet" /1*4/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c('1','1')=0 ; c('1','2')=0 ; c('1','3')=1 ;
c('1','4')=3 ; c('2','1')=3 ; c('2','2')=1 ;
c('2','3')=1 ; c('2','4')=4 ; c('3','1')=4 ;
c('3','2')=2 ; c('3','3')=1 ; c('3','4')=1 ;
c('4','1')=0 ; c('4','2')=2 ; c('4','3')=2 ;
c('4','4')=1 ; c('5','1')=0 ; c('5','2')=4 ;
c('5','3')=2 ; c('5','4')=0 ; c('6','1')=0 ;
c('6','2')=3 ; c('6','3')=4 ; c('6','4')=1 ;
c('7','1')=2 ; c('7','2')=2 ; c('7','3')=1 ;
c('7','4')=1 ; c('8','1')=0 ; c('8','2')=0 ;
c('8','3')=2 ; c('8','4')=3 ; c('9','1')=3 ;
c('9','2')=4 ; c('9','3')=4 ; c('9','4')=0 ;
c('10','1')=0 ; c('10','2')=2 ; c('10','3')=2 ;
c('10','4')=4 ; c('11','1')=1 ; c('11','2')=2 ;
c('11','3')=0 ; c('11','4')=3 ; c('12','1')=4 ;
c('12','2')=2 ; c('12','3')=4 ; c('12','4')=0 ;
c('13','1')=4 ; c('13','2')=3 ; c('13','3')=1 ;
c('13','4')=3 ; c('14','1')=3 ; c('14','2')=3 ;
c('14','3')=0 ; c('14','4')=0 ; c('15','1')=3 ;
c('15','2')=0 ; c('15','3')=4 ; c('15','4')=3 ;
c('16','1')=0 ; c('16','2')=1 ; c('16','3')=4 ;
c('16','4')=2 ;
```

5×5-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan koko nxn" /5/ ;

Sets
k "palat" /1*25/
l "palan reunat" /1*4/
i "rivit" /1*5/
j "sarakkeet" /1*5/
m "rotaatio" /1*3/ ;
alias(k,t);
Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=4 ; c('1','2')=1 ; c('1','3')=2 ; c('1','4')=2 ;
c('2','1')=4 ; c('2','2')=3 ; c('2','3')=4 ; c('2','4')=3 ;
c('3','1')=0 ; c('3','2')=0 ; c('3','3')=1 ; c('3','4')=2 ;
c('4','1')=0 ; c('4','2')=1 ; c('4','3')=2 ; c('4','4')=3 ;
c('5','1')=4 ; c('5','2')=1 ; c('5','3')=2 ; c('5','4')=1 ;
c('6','1')=0 ; c('6','2')=3 ; c('6','3')=2 ; c('6','4')=3 ;
c('7','1')=3 ; c('7','2')=4 ; c('7','3')=0 ; c('7','4')=3 ;
c('8','1')=2 ; c('8','2')=1 ; c('8','3')=2 ; c('8','4')=3 ;
c('9','1')=1 ; c('9','2')=0 ; c('9','3')=2 ; c('9','4')=3 ;
c('10','1')=0 ; c('10','2')=0 ; c('10','3')=4 ; c('10','4')=2 ;
c('11','1')=1 ; c('11','2')=0 ; c('11','3')=3 ; c('11','4')=2 ;
c('12','1')=0 ; c('12','2')=4 ; c('12','3')=1 ; c('12','4')=0 ;
c('13','1')=3 ; c('13','2')=4 ; c('13','3')=0 ; c('13','4')=3 ;
c('14','1')=1 ; c('14','2')=1 ; c('14','3')=2 ; c('14','4')=4 ;
c('15','1')=3 ; c('15','2')=1 ; c('15','3')=4 ; c('15','4')=1 ;
c('16','1')=3 ; c('16','2')=3 ; c('16','3')=2 ; c('16','4')=0 ;
c('17','1')=4 ; c('17','2')=3 ; c('17','3')=0 ; c('17','4')=0 ;
c('18','1')=4 ; c('18','2')=3 ; c('18','3')=4 ; c('18','4')=2 ;
c('19','1')=0 ; c('19','2')=1 ; c('19','3')=1 ; c('19','4')=4 ;
c('20','1')=1 ; c('20','2')=1 ; c('20','3')=1 ; c('20','4')=0 ;
c('21','1')=2 ; c('21','2')=1 ; c('21','3')=4 ; c('21','4')=4 ;
c('22','1')=3 ; c('22','2')=2 ; c('22','3')=1 ; c('22','4')=2 ;
c('23','1')=0 ; c('23','2')=2 ; c('23','3')=2 ; c('23','4')=3 ;
c('24','1')=0 ; c('24','2')=2 ; c('24','3')=4 ; c('24','4')=3 ;
c('25','1')=4 ; c('25','2')=0 ; c('25','3')=4 ; c('25','4')=4 ;
```

Ei reunoja -Eternity

Kuten edellä, esitellään aluksi $n \times n$-kokoisen epälineaarisen ja lineaarisen ei reunoja -Eternityn GAMS-malli. Tämän jälkeen esitetään eri kokoisten mallien kiinnitetyt indeksien ja parametrien arvot.

$n \times n$-kokoinen epälineaarinen malli

```
Scalar
p_max "suurin väri" /p_max/
n "pelilaudan yhden sivun koko" /n/ ;

Sets
k "palat" /1*n^2/
l "palan reunat" /1*4/
i "rivit" /1*n/
j "sarakkeet" /1*n/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;

Variables
x(i,j,k) "onko pala k ruudussa (i,j)"
y(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
r(k,l) "palan k reunan l väri, kun pala on paikallaan"
z "kohdefunktiomuuttuja" ;

Binary variables x, y ;
Positive variables r ;

r.up(k,l) = p_max ;
```

```
Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k,t), raj9(k,t), raj10(k,t), raj11(k,t) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(k).. sum((i,j), x(i,j,k)) =L= 1 ;
raj2(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj3(k).. y(k,'1')+y(k,'2')+y(k,'3') =L= 1 ;
raj4(k).. r(k,'1') =E= c(k,'1')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'4')*y(k,'1')+c(k,'3')*y(k,'2')+c(k,'2')*y(k,'3') ;
raj5(k).. r(k,'2') =E= c(k,'2')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'1')*y(k,'1')+c(k,'4')*y(k,'2')+c(k,'3')*y(k,'3') ;
raj6(k).. r(k,'3') =E= c(k,'3')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'2')*y(k,'1')+c(k,'1')*y(k,'2')+c(k,'4')*y(k,'3') ;
raj7(k).. r(k,'4') =E= c(k,'4')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'3')*y(k,'1')+c(k,'2')*y(k,'2')+c(k,'1')*y(k,'3') ;

raj8(k,t) $(ord(k)<>ord(t)).. r(k,'3')-r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), x(i,j,k)*x(i,j+1,t))) ;
raj9(k,t) $(ord(k)<>ord(t)).. -r(k,'3')+r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), x(i,j,k)*x(i,j+1,t))) ;
raj10(k,t) $(ord(k)<>ord(t)).. r(k,'4')-r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), x(i,j,k)*x(i+1,j,t))) ;
raj11(k,t) $(ord(k)<>ord(t)).. -r(k,'4')+r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), x(i,j,k)*x(i+1,j,t))) ;

Model EiReunoja "Ei reunoja -Eternity, nxn-ruudukko" /all/ ;

Solve EiReunoja using MINLP maximizing z ;
```

$n \times n$-kokoinen lineaarinen malli

```
Scalar
p_max "suurin väri" /p_max/
n "pelilaudan yhden sivun koko" /n/ ;

Sets
k "palat" /1*n^2/
l "palan reunat" /1*4/
i "rivit" /1*n/
j "sarakkeet" /1*n/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;

Variables
x(i,j,k) "onko pala k ruudussa (i,j)"
y(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
r(k,l) "palan k reunan l väri, kun pala on paikallaan"
s1(i,j,k,t) "binäärimuuttuja 1 linearisointia varten"
s2(i,j,k,t) "binäärimuuttuja 2 linearisointia varten"
z "kohdefunktiomuuttuja" ;

Binary variables x, y, s1, s2 ;
Positive variables r ;

r.up(k,l) = p_max ;

Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k,t), raj9(k,t), raj10(k,t), raj11(k,t), raj12(i,j,k,t)
raj13(i,j,k,t), raj14(i,j,k,t), raj15(i,j,k,t) ;
```

```
obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(k).. sum((i,j), x(i,j,k)) =L= 1 ;
raj2(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj3(k).. y(k,'1')+y(k,'2')+y(k,'3') =L= 1 ;
raj4(k).. r(k,'1') =E= c(k,'1')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'4')*y(k,'1')+c(k,'3')*y(k,'2')+c(k,'2')*y(k,'3') ;
raj5(k).. r(k,'2') =E= c(k,'2')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'1')*y(k,'1')+c(k,'4')*y(k,'2')+c(k,'3')*y(k,'3') ;
raj6(k).. r(k,'3') =E= c(k,'3')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'2')*y(k,'1')+c(k,'1')*y(k,'2')+c(k,'4')*y(k,'3') ;
raj7(k).. r(k,'4') =E= c(k,'4')*(1-y(k,'1')-y(k,'2')-y(k,'3'))+
c(k,'3')*y(k,'1')+c(k,'2')*y(k,'2')+c(k,'1')*y(k,'3') ;

raj8(k,t) $(ord(k)<>ord(t)).. r(k,'3')-r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), s1(i,j,k,t))) ;
raj9(k,t) $(ord(k)<>ord(t)).. -r(k,'3')+r(t,'1') =L= p_max *
(1-sum((i,j) $(ord(j)<n), s1(i,j,k,t))) ;
raj10(k,t) $(ord(k)<>ord(t)).. r(k,'4')-r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), s2(i,j,k,t))) ;
raj11(k,t) $(ord(k)<>ord(t)).. -r(k,'4')+r(t,'2') =L= p_max *
(1-sum((i,j) $(ord(i)<n), s2(i,j,k,t))) ;

raj12(i,j,k,t) $(ord(k)<>ord(t) and ord(j)<n).. s1(i,j,k,t) =G=
1+(x(i,j,k)+x(i,j+1,t)-2) ;
raj13(i,j,k,t) $(ord(k)<>ord(t) and ord(j)<n).. s1(i,j,k,t) =L=
0.5 * (x(i,j,k)+x(i,j+1,t)) ;
raj14(i,j,k,t) $(ord(k)<>ord(t) and ord(i)<n).. s2(i,j,k,t) =G=
1+(x(i,j,k)+x(i+1,j,t)-2) ;
raj15(i,j,k,t) $(ord(k)<>ord(t) and ord(i)<n).. s2(i,j,k,t) =L=
0.5 * (x(i,j,k)+x(i+1,j,t)) ;

Model EiReunojaLIN "Ei reunoja -Eternity, lineaarinen" /all/ ;

Solve EiReunojaLIN using MIP maximizing z ;
```

2×2-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /2/ ;

Sets
k "palat" /1*4/
l "palan reunat" /1*4/
i "rivit" /1*2/
j "sarakkeet" /1*2/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=1 ;
c('1','2')=2 ;
c('1','3')=1 ;
c('1','4')=3 ;
c('2','1')=1 ;
c('2','2')=4 ;
c('2','3')=3 ;
c('2','4')=4 ;
c('3','1')=4 ;
c('3','2')=1 ;
c('3','3')=3 ;
c('3','4')=2 ;
c('4','1')=4 ;
c('4','2')=3 ;
c('4','3')=2 ;
c('4','4')=2 ;
```

3×3-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /3/ ;

Sets
k "palat" /1*9/
l "palan reunat" /1*4/
i "rivit" /1*3/
j "sarakkeet" /1*3/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=2 ; c('1','2')=2 ; c('1','3')=4 ;
c('1','4')=1 ; c('2','1')=3 ; c('2','2')=3 ;
c('2','3')=1 ; c('2','4')=2 ; c('3','1')=4 ;
c('3','2')=3 ; c('3','3')=4 ; c('3','4')=2 ;
c('4','1')=4 ; c('4','2')=3 ; c('4','3')=3 ;
c('4','4')=2 ; c('5','1')=1 ; c('5','2')=4 ;
c('5','3')=1 ; c('5','4')=1 ; c('6','1')=3 ;
c('6','2')=2 ; c('6','3')=1 ; c('6','4')=2 ;
c('7','1')=3 ; c('7','2')=4 ; c('7','3')=1 ;
c('7','4')=4 ; c('8','1')=2 ; c('8','2')=2 ;
c('8','3')=4 ; c('8','4')=1 ; c('9','1')=4 ;
c('9','2')=3 ; c('9','3')=1 ; c('9','4')=3 ;
```

4×4-kokoinen ruudukko

```
Scalar
p_max "suurin väri" /4/
n "pelilaudan yhden sivun koko" /4/ ;

Sets
k "palat" /1*16/
l "palan reunat" /1*4/
i "rivit" /1*4/
j "sarakkeet" /1*4/
m "rotaatio" /1*3/ ;
alias(k,t);

Parameters
c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=1 ; c('1','2')=2 ; c('1','3')=1 ;
c('1','4')=3 ; c('2','1')=3 ; c('2','2')=1 ;
c('2','3')=1 ; c('2','4')=4 ; c('3','1')=4 ;
c('3','2')=2 ; c('3','3')=1 ; c('3','4')=1 ;
c('4','1')=3 ; c('4','2')=2 ; c('4','3')=2 ;
c('4','4')=1 ; c('5','1')=2 ; c('5','2')=4 ;
c('5','3')=2 ; c('5','4')=2 ; c('6','1')=4 ;
c('6','2')=3 ; c('6','3')=4 ; c('6','4')=1 ;
c('7','1')=2 ; c('7','2')=2 ; c('7','3')=1 ;
c('7','4')=1 ; c('8','1')=1 ; c('8','2')=4 ;
c('8','3')=2 ; c('8','4')=3 ; c('9','1')=3 ;
c('9','2')=4 ; c('9','3')=4 ; c('9','4')=3 ;
c('10','1')=3 ; c('10','2')=2 ; c('10','3')=2 ;
c('10','4')=4 ; c('11','1')=1 ; c('11','2')=2 ;
c('11','3')=3 ; c('11','4')=3 ; c('12','1')=4 ;
c('12','2')=2 ; c('12','3')=4 ; c('12','4')=2 ;
c('13','1')=4 ; c('13','2')=3 ; c('13','3')=1 ;
c('13','4')=3 ; c('14','1')=3 ; c('14','2')=3 ;
c('14','3')=4 ; c('14','4')=1 ; c('15','1')=3 ;
c('15','2')=1 ; c('15','3')=4 ; c('15','4')=3 ;
c('16','1')=4 ; c('16','2')=1 ; c('16','3')=4 ;
c('16','4')=2 ;
```

Kolmio-Eternity

GAMS-mallin merkinnät poikkeavat aliluvun 5.2.2 mallin merkinnöistä seuraavalla tavalla. Alla muuttuja y vastaa mallin (13) muuttujaa x, muuttuja r muuttujaa y, muuttuja f muuttujaa r ja indeksi n indeksiä t.

2×2-kokoinen epälineaarinen malli

```
Sets
k "palat" /1*4/
l "palan reunat" /1*3/
i "rivit" /1*2/
j "sarakkeet" /1*3/
m "rotaatio" /1*2/
t(i,j) "pelilaudan ruudut" /1.2, 2.1, 2.2, 2.3/;
alias(k,n);

Parameters c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=0 ;
c('1','2')=0 ;
c('1','3')=2 ;
c('2','1')=0 ;
c('2','2')=0 ;
c('2','3')=3 ;
c('3','1')=0 ;
c('3','2')=0 ;
c('3','3')=1 ;
c('4','1')=2 ;
c('4','2')=3 ;
c('4','3')=1 ;

Variables
y(i,j,k) "onko pala k ruudussa (i,j)"
r(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
f(k,l) "palan k reunan l väri, kun pala on paikallaan"
z "kohdefunktiomuuttuja" ;

Binary variables y, r ;
Positive variables f ;

f.up(k,l) = 3 ;
```

```
Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k,n), raj11(k,n), raj12(k,n)
raj13(k,n), raj14(k,n), raj15(k,n) ;

obj.. z =E= sum((i,j,k),(y(i,j,k)) $t(i,j)) ;
raj1(k).. sum((i,j), y(i,j,k) $t(i,j)) =L= 1 ;
raj2(i,j) $t(i,j).. sum(k, y(i,j,k)) =L= 1 ;
raj3(k).. r(k,'1')+r(k,'2') =L= 1 ;
raj4(k).. f(k,'1') =L= 3*(1-y('2','1',k)-y('2','3',k)) ;
raj5(k).. f(k,'2') =L= 3*(1-y('2','1',k)-y('1','2',k)) ;
raj6(k).. f(k,'3') =L= 3*(1-y('1','2',k)-y('2','3',k)) ;
raj7(k).. f(k,'1') =E= c(k,'1')*(1-r(k,'1')-r(k,'2'))+
c(k,'3')*r(k,'1')+c(k,'2')*r(k,'2') ;
raj8(k).. f(k,'2') =E= c(k,'2')*(1-r(k,'1')-r(k,'2'))+
c(k,'1')*r(k,'1')+c(k,'3')*r(k,'2') ;
raj9(k).. f(k,'3') =E= c(k,'3')*(1-r(k,'1')-r(k,'2'))+
c(k,'2')*r(k,'1')+c(k,'1')*r(k,'2') ;
raj10(k,n) $(ord(k)<>ord(n)).. f(k,'1')-f(n,'1') =L=
3*(1-y('1','2',k)*y('2','2',n)) ;
raj11(k,n) $(ord(k)<>ord(n)).. -f(k,'1')+f(n,'1') =L=
3*(1-y('1','2',k)*y('2','2',n)) ;
raj12(k,n) $(ord(k)<>ord(n)).. f(k,'2')-f(n,'2') =L=
3*(1-y('2','2',k)*y('2','3',n)) ;
raj13(k,n) $(ord(k)<>ord(n)).. -f(k,'2')+f(n,'2') =L=
3*(1-y('2','2',k)*y('2','3',n)) ;
raj14(k,n) $(ord(k)<>ord(n)).. f(k,'3')-f(n,'3') =L=
3*(1-y('2','1',k)*y('2','2',n)) ;
raj15(k,n) $(ord(k)<>ord(n)).. -f(k,'3')+f(n,'3') =L=
3*(1-y('2','1',k)*y('2','2',n)) ;

Model KolmioEternity "Kolmio-Eternity" /all/ ;

Solve KolmioEternity using MINLP maximizing z ;
```

2×2-kokoinen lineaarinen malli

```
Sets
k "palat" /1*4/
l "palan reunat" /1*3/
i "rivit" /1*2/
j "sarakkeet" /1*3/
m "rotaatio" /1*2/
t(i,j) "pelilaudan ruudut" /1.2, 2.1, 2.2, 2.3/;
alias(k,n);

Parameters c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=0 ;
c('1','2')=0 ;
c('1','3')=2 ;
c('2','1')=0 ;
c('2','2')=0 ;
c('2','3')=3 ;
c('3','1')=0 ;
c('3','2')=0 ;
c('3','3')=1 ;
c('4','1')=2 ;
c('4','2')=3 ;
c('4','3')=1 ;

Variables
y(i,j,k) "onko pala k ruudussa (i,j)"
r(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
f(k,l) "palan k reunan l väri, kun pala on paikallaan"
s1(k,n) "binäärimuuttuja 1 linearisointia varten"
s2(k,n) "binäärimuuttuja 2 linearisointia varten"
s3(k,n) "binäärimuuttuja 3 linearisointia varten"
z "kohdefunktiomuuttuja" ;

Binary variables y, r, s1, s2, s3 ;
Positive variables f ;
```

```
f.up(k,l) = 3 ;

Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k,n), raj11(k,n), raj12(k,n)
raj13(k,n), raj14(k,n), raj15(k,n), raj16(k,n), raj17(k,n)
raj18(k,n), raj19(k,n), raj20(k,n), raj21(k,n) ;

obj.. z =E= sum((i,j,k),(y(i,j,k)) $t(i,j)) ;
raj1(k).. sum((i,j), y(i,j,k) $t(i,j)) =L= 1 ;
raj2(i,j) $t(i,j).. sum(k, y(i,j,k)) =L= 1 ;
raj3(k).. r(k,'1')+r(k,'2') =L= 1 ;
raj4(k).. f(k,'1') =L= 3*(1-y('2','1',k)-y('2','3',k)) ;
raj5(k).. f(k,'2') =L= 3*(1-y('2','1',k)-y('1','2',k)) ;
raj6(k).. f(k,'3') =L= 3*(1-y('1','2',k)-y('2','3',k)) ;
raj7(k).. f(k,'1') =E= c(k,'1')*(1-r(k,'1')-r(k,'2'))+
c(k,'3')*r(k,'1')+c(k,'2')*r(k,'2') ;
raj8(k).. f(k,'2') =E= c(k,'2')*(1-r(k,'1')-r(k,'2'))+
c(k,'1')*r(k,'1')+c(k,'3')*r(k,'2') ;
raj9(k).. f(k,'3') =E= c(k,'3')*(1-r(k,'1')-r(k,'2'))+
c(k,'2')*r(k,'1')+c(k,'1')*r(k,'2') ;

raj10(k,n) $(ord(k)<>ord(n)).. f(k,'1')-f(n,'1') =L= 3*(1-s1(k,n)) ;
raj11(k,n) $(ord(k)<>ord(n)).. -f(k,'1')+f(n,'1') =L= 3*(1-s1(k,n)) ;
raj12(k,n) $(ord(k)<>ord(n)).. f(k,'2')-f(n,'2') =L= 3*(1-s2(k,n)) ;
raj13(k,n) $(ord(k)<>ord(n)).. -f(k,'2')+f(n,'2') =L= 3*(1-s2(k,n)) ;
raj14(k,n) $(ord(k)<>ord(n)).. f(k,'3')-f(n,'3') =L= 3*(1-s3(k,n)) ;
raj15(k,n) $(ord(k)<>ord(n)).. -f(k,'3')+f(n,'3') =L= 3*(1-s3(k,n)) ;
raj16(k,n) $(ord(k)<>ord(n)).. s1(k,n) =G=
1+(y('1','2',k)+y('2','2',n)-2) ;
raj17(k,n) $(ord(k)<>ord(n)).. s1(k,n) =L=
0.5 * (y('1','2',k)+y('2','2',n)) ;
raj18(k,n) $(ord(k)<>ord(n)).. s2(k,n) =G=
1+(y('2','2',k)+y('2','3',n)-2) ;
raj19(k,n) $(ord(k)<>ord(n)).. s2(k,n) =L=
0.5 * (y('2','2',k)+y('2','3',n)) ;
raj20(k,n) $(ord(k)<>ord(n)).. s3(k,n) =G=
1+(y('2','1',k)+y('2','2',n)-2) ;
raj21(k,n) $(ord(k)<>ord(n)).. s3(k,n) =L=
0.5 * (y('2','1',k)+y('2','2',n)) ;

Model KolmioEternityLin "Kolmio-Eternity, lin." /all/ ;

Solve KolmioEternityLin using MIP maximizing z ;

Display y.l, f.l ;
```

3×3-kokoinen epälineaarinen malli

```
Sets
k "palat" /1*9/
l "palan reunat" /1*3/
i "rivit" /1*3/
j "sarakkeet" /1*5/
m "rotaatio" /1*2/
t(i,j) "pelilaudan ruudut" /1.3, 2.2, 2.3, 2.4
3.1, 3.2, 3.3, 3.4, 3.5/;
alias(k,n);

Parameters c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=1 ; c('1','2')=0 ; c('1','3')=0 ;
c('2','1')=2 ; c('2','2')=0 ; c('2','3')=1 ;
c('3','1')=2 ; c('3','2')=1 ; c('3','3')=3 ;
c('4','1')=0 ; c('4','2')=0 ; c('4','3')=1 ;
c('5','1')=2 ; c('5','2')=3 ; c('5','3')=0 ;
c('6','1')=0 ; c('6','2')=3 ; c('6','3')=3 ;
c('7','1')=0 ; c('7','2')=2 ; c('7','3')=0 ;
c('8','1')=1 ; c('8','2')=2 ; c('8','3')=3 ;
c('9','1')=2 ; c('9','2')=3 ; c('9','3')=1 ;

Variables
y(i,j,k) "onko pala k ruudussa (i,j)"
r(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
f(k,l) "palan k reunan l väri, kun pala on paikallaan"
z "kohdefunktiomuuttuja" ;

Binary variables y, r ;
Positive variables f ;

f.up(k,l) = 3 ;
```

```
Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k,n), raj11(k,n), raj12(k,n)
raj13(k,n), raj14(k,n), raj15(k,n) ;

obj.. z =E= sum((i,j,k),(y(i,j,k)) $t(i,j)) ;
raj1(k).. sum((i,j), y(i,j,k) $t(i,j)) =L= 1 ;
raj2(i,j) $t(i,j).. sum(k, y(i,j,k)) =L= 1 ;
raj3(k).. r(k,'1')+r(k,'2') =L= 1 ;
raj4(k).. f(k,'1') =L= 3*(1-y('3','1',k)-y('3','3',k)-y('3','5',k)) ;
raj5(k).. f(k,'2') =L= 3*(1-y('1','3',k)-y('2','2',k)-y('3','1',k)) ;
raj6(k).. f(k,'3') =L= 3*(1-y('1','3',k)-y('2','4',k)-y('3','5',k)) ;
raj7(k).. f(k,'1') =E= c(k,'1')*(1-r(k,'1')-r(k,'2'))+
c(k,'3')*r(k,'1')+c(k,'2')*r(k,'2') ;
raj8(k).. f(k,'2') =E= c(k,'2')*(1-r(k,'1')-r(k,'2'))+
c(k,'1')*r(k,'1')+c(k,'3')*r(k,'2') ;
raj9(k).. f(k,'3') =E= c(k,'3')*(1-r(k,'1')-r(k,'2'))+
c(k,'2')*r(k,'1')+c(k,'1')*r(k,'2') ;
raj10(k,n) $(ord(k)<>ord(n)).. f(k,'1')-f(n,'1') =L= 3*(1-y('1','3',k)*
y('2','3',n)-y('2','2',k)*y('3','2',n)-y('2','4',k)*y('3','4',n)) ;
raj11(k,n) $(ord(k)<>ord(n)).. -f(k,'1')+f(n,'1') =L= 3*(1-y('1','3',k)*
y('2','3',n)-y('2','2',k)*y('3','2',n)-y('2','4',k)*y('3','4',n)) ;
raj12(k,n) $(ord(k)<>ord(n)).. f(k,'2')-f(n,'2') =L= 3*(1-y('2','3',k)*
y('2','4',n)-y('3','4',k)*y('3','5',n)-y('3','2',k)*y('3','3',n)) ;
raj13(k,n) $(ord(k)<>ord(n)).. -f(k,'2')+f(n,'2') =L= 3*(1-y('2','3',k)*
y('2','4',n)-y('3','4',k)*y('3','5',n)-y('3','2',k)*y('3','3',n)) ;
raj14(k,n) $(ord(k)<>ord(n)).. f(k,'3')-f(n,'3') =L= 3*(1-y('2','2',k)*
y('2','3',n)-y('3','1',k)*y('3','2',n)-y('3','3',k)*y('3','4',n)) ;
raj15(k,n) $(ord(k)<>ord(n)).. -f(k,'3')+f(n,'3') =L= 3*(1-y('2','2',k)*
y('2','3',n)-y('3','1',k)*y('3','2',n)-y('3','3',k)*y('3','4',n)) ;

Model KolmioEternity "Kolmio-Eternity, 9 palaa" /all/ ;

Solve KolmioEternity using MINLP maximizing z ;
```

4×4-kokoinen epälineaarinen malli

```
Sets
k "palat" /1*16/
l "palan reunat" /1*3/
i "rivit" /1*4/
j "sarakkeet" /1*7/
m "rotaatio" /1*2/
t(i,j) "pelilaudan ruudut" /1.4, 2.3, 2.4, 2.5, 3.2
3.3, 3.4, 3.5, 3.6, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7/ ;
alias(k,n);

Parameters c(k,l) "palan k reunan l väri perusasennossa" ;
c('1','1')=2 ; c('1','2')=0 ; c('1','3')=0 ;
c('2','1')=0 ; c('2','2')=2 ; c('2','3')=3 ;
c('3','1')=0 ; c('3','2')=2 ; c('3','3')=1 ;
c('4','1')=1 ; c('4','2')=0 ; c('4','3')=0 ;
c('5','1')=2 ; c('5','2')=1 ; c('5','3')=3 ;
c('6','1')=2 ; c('6','2')=2 ; c('6','3')=1 ;
c('7','1')=3 ; c('7','2')=0 ; c('7','3')=2 ;
c('8','1')=1 ; c('8','2')=1 ; c('8','3')=2 ;
c('9','1')=0 ; c('9','2')=2 ; c('9','3')=1 ;
c('10','1')=3 ; c('10','2')=1 ; c('10','3')=0 ;
c('11','1')=1 ; c('11','2')=3 ; c('11','3')=3 ;
c('12','1')=0 ; c('12','2')=2 ; c('12','3')=3 ;
c('13','1')=3 ; c('13','2')=3 ; c('13','3')=3 ;
c('14','1')=2 ; c('14','2')=1 ; c('14','3')=3 ;
c('15','1')=0 ; c('15','2')=0 ; c('15','3')=1 ;
c('16','1')=2 ; c('16','2')=1 ; c('16','3')=3 ;

Variables
y(i,j,k) "onko pala k ruudussa (i,j)"
r(k,m) "onko palaa k kierretty kellon suuntaan m*90 astetta"
f(k,l) "palan k reunan l väri, kun pala on paikallaan"
z "kohdefunktiomuuttuja" ;

Binary variables y, r ; Positive variables f ;
f.up(k,l) = 3 ;

Equations
obj "kohdefunktio"
raj1(k), raj2(i,j), raj3(k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k,n), raj11(k,n), raj12(k,n)
raj13(k,n), raj14(k,n), raj15(k,n) ;

obj.. z =E= sum((i,j,k),(y(i,j,k)) $t(i,j)) ;
raj1(k).. sum((i,j), y(i,j,k) $t(i,j)) =L= 1 ;
raj2(i,j) $t(i,j).. sum(k, y(i,j,k)) =L= 1 ;
raj3(k).. r(k,'1')+r(k,'2') =L= 1 ;
raj4(k).. f(k,'1') =L= 3*(1-y('4','1',k)-y('4','3',k)-y('4','4',k)-y('4','7',k)) ;
raj5(k).. f(k,'2') =L= 3*(1-y('1','4',k)-y('2','3',k)-y('3','2',k)-y('4','1',k)) ;
raj6(k).. f(k,'3') =L= 3*(1-y('1','4',k)-y('2','5',k)-y('3','6',k)-y('4','7',k)) ;
raj7(k).. f(k,'1') =E= c(k,'1')*(1-r(k,'1')-r(k,'2'))+
c(k,'3')*r(k,'1')+c(k,'2')*r(k,'2') ;
raj8(k).. f(k,'2') =E= c(k,'2')*(1-r(k,'1')-r(k,'2'))+
c(k,'1')*r(k,'1')+c(k,'3')*r(k,'2') ;
raj9(k).. f(k,'3') =E= c(k,'3')*(1-r(k,'1')-r(k,'2'))+
c(k,'2')*r(k,'1')+c(k,'1')*r(k,'2') ;
```

```
raj10(k,n) $(ord(k)<>ord(n)).. f(k,'1')-f(n,'1') =L= 3*(1-y('1','4',k)*
y('2','4',n)-y('2','3',k)*y('3','3',n)-y('2','5',k)*y('3','5',n)
-y('3','2',k)*y('4','2',n)-y('3','4',k)*y('4','4',n)-y('3','6',k)*y('4','6',n)) ;
raj11(k,n) $(ord(k)<>ord(n)).. -f(k,'1')+f(n,'1') =L= 3*(1-y('1','4',k)*
y('2','4',n)-y('2','3',k)*y('3','3',n)-y('2','5',k)*y('3','5',n)
-y('3','2',k)*y('4','2',n)-y('3','4',k)*y('4','4',n)-y('3','6',k)*y('4','6',n)) ;
raj12(k,n) $(ord(k)<>ord(n)).. f(k,'2')-f(n,'2') =L= 3*(1-y('2','4',k)*
y('2','5',n)-y('3','3',k)*y('3','4',n)-y('3','5',k)*y('3','6',n)
-y('4','2',k)*y('4','3',n)-y('4','4',k)*y('4','5',n)-y('4','6',k)*y('4','7',n)) ;
raj13(k,n) $(ord(k)<>ord(n)).. -f(k,'2')+f(n,'2') =L= 3*(1-y('2','4',k)*
y('2','5',n)-y('3','3',k)*y('3','4',n)-y('3','5',k)*y('3','6',n)
-y('4','2',k)*y('4','3',n)-y('4','4',k)*y('4','5',n)-y('4','6',k)*y('4','7',n)) ;
raj14(k,n) $(ord(k)<>ord(n)) .. f(k,'3')-f(n,'3') =L= 3*(1-y('2','3',k)*
y('2','4',n)-y('3','2',k)*y('3','3',n)-y('3','4',k)*y('3','5',n)
-y('4','1',k)*y('4','2',n)-y('4','3',k)*y('4','4',n)-y('4','5',k)*y('4','6',n)) ;
raj15(k,n) $(ord(k)<>ord(n)).. -f(k,'3')+f(n,'3') =L= 3*(1-y('2','3',k)*
y('2','4',n)-y('3','2',k)*y('3','3',n)-y('3','4',k)*y('3','5',n)
-y('4','1',k)*y('4','2',n)-y('4','3',k)*y('4','4',n)-y('4','5',k)*y('4','6',n)) ;

Model KolmioEternity "Kolmio-Eternity, 16 palaa" /all/ ;

Solve KolmioEternity using MINLP maximizing z ;
```

Perinteinen sudoku, mallinnustapa 1

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;
*annetaan vinkkinumerot
x.fx('1','2','3')=1;
x.fx('1','5','5')=1;
x.fx('1','7','1')=1;
x.fx('2','1','6')=1;
x.fx('2','3','2')=1;
x.fx('2','7','5')=1;
x.fx('2','9','8')=1;
x.fx('3','4','2')=1;
x.fx('3','6','8')=1;
x.fx('4','2','9')=1;
x.fx('4','3','7')=1;
x.fx('4','5','1')=1;
x.fx('4','7','8')=1;
x.fx('4','8','3')=1;
x.fx('5','5','3')=1;
x.fx('6','2','8')=1;
x.fx('6','3','1')=1;
x.fx('6','5','2')=1;
x.fx('6','7','6')=1;
x.fx('6','8','7')=1;
x.fx('7','4','9')=1;
x.fx('7','6','4')=1;
x.fx('8','1','1')=1;
x.fx('8','3','9')=1;
x.fx('8','7','4')=1;
x.fx('8','9','7')=1;
x.fx('9','2','2')=1;
x.fx('9','5','7')=1;
x.fx('9','8','8')=1;
```

```
Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,k), raj3(j,k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj2(i,k).. sum(j, x(i,j,k)) =L= 1 ;
raj3(j,k).. sum(i, x(i,j,k)) =L= 1 ;

*käydään pikkuruudut läpi
raj4(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj5(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj6(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj7(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj8(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj9(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj10(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj11(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj12(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;

Model sudoku1 "perinteinen sudoku, tapa 1" /all/ ;

Solve sudoku1 using MIP maximizing z ;
```

Perinteinen sudoku, mallinnustapa 2

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;
*annetaan vinkkinumerot
x.fx('1','2','3')=1;
x.fx('1','5','5')=1;
x.fx('1','7','1')=1;
x.fx('2','1','6')=1;
x.fx('2','3','2')=1;
x.fx('2','7','5')=1;
x.fx('2','9','8')=1;
x.fx('3','4','2')=1;
x.fx('3','6','8')=1;
x.fx('4','2','9')=1;
x.fx('4','3','7')=1;
x.fx('4','5','1')=1;
x.fx('4','7','8')=1;
x.fx('4','8','3')=1;
x.fx('5','5','3')=1;
x.fx('6','2','8')=1;
x.fx('6','3','1')=1;
x.fx('6','5','2')=1;
x.fx('6','7','6')=1;
x.fx('6','8','7')=1;
x.fx('7','4','9')=1;
x.fx('7','6','4')=1;
x.fx('8','1','1')=1;
x.fx('8','3','9')=1;
x.fx('8','7','4')=1;
x.fx('8','9','7')=1;
x.fx('9','2','2')=1;
x.fx('9','5','7')=1;
x.fx('9','8','8')=1;
```

```
Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,k), raj3(j,k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k) ;

obj.. z =E= 1 ;
raj1(i,j).. sum(k, x(i,j,k)) =E= 1 ;
raj2(i,k).. sum(j, x(i,j,k)) =E= 1 ;
raj3(j,k).. sum(i, x(i,j,k)) =E= 1 ;

*käydään pikkuruudut läpi
raj4(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =E= 1 ;
raj5(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =E= 1 ;
raj6(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =E= 1 ;
raj7(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =E= 1 ;
raj8(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =E= 1 ;
raj9(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =E= 1 ;
raj10(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =E= 1 ;
raj11(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =E= 1 ;
raj12(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =E= 1 ;

Model sudoku2 "perinteinen sudoku, tapa 2" /all/ ;

Solve sudoku2 using MIP maximizing z ;
```

Perinteinen sudoku, mallinnustapa 3

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
ko /0*3/ ;

alias(i,k);
alias(j,l);

Variables
x(i,j) "ruutuun (i,j) sijoitettu luku"
b(i,j,ko) "binäärimuuttuja"
y(i,j,k,l) "binäärimuuttuja"
z "kohdefunktiomuuttuja" ;

Integer variables x ;
Binary variables b, y ;

x.lo(i,j) = 1 ;
x.up(i,j) = 9 ;
*annetaan vinkkinumerot
x.fx('1','2')=3;
x.fx('1','5')=5;
x.fx('1','7')=1;
x.fx('2','1')=6;
x.fx('2','3')=2;
x.fx('2','7')=5;
x.fx('2','9')=8;
x.fx('3','4')=2;
x.fx('3','6')=8;
x.fx('4','2')=9;
x.fx('4','3')=7;
x.fx('4','5')=1;
x.fx('4','7')=8;
x.fx('4','8')=3;
x.fx('5','5')=3;
x.fx('6','2')=8;
x.fx('6','3')=1;
x.fx('6','5')=2;
x.fx('6','7')=6;
x.fx('6','8')=7;
x.fx('7','4')=9;
x.fx('7','6')=4;
x.fx('8','1')=1;
x.fx('8','3')=9;
x.fx('8','7')=4;
x.fx('8','9')=7;
x.fx('9','2')=2;
x.fx('9','5')=7;
x.fx('9','8')=8;
```

```
Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,j,k,l), raj3(i,j,k,l), raj4(i,j,k,l), raj5(i,j,k,l), raj6(i,j,k,l)
raj7(i,j,k,l), raj8(i,j,k,l), raj9(i,j,k,l), raj10(i,j,k,l), raj11(i,j,k,l)
raj12(i,j,k,l), raj13(i,j,k,l), raj14(i,j,k,l), raj15(i,j,k,l), raj16(i,j,k,l)
raj17(i,j,k,l), raj18(i,j,k,l), raj19(i,j,k,l), raj20(i,j,k,l), raj21(i,j,k,l)
raj22(i,j,k,l), raj23(i,j,k,l) ;

obj.. z =E= 1 ;

raj1(i,j).. x(i,j) =E= b(i,j,'0')+b(i,j,'1')*2+b(i,j,'2')*4+b(i,j,'3')*8 ;

raj2(i,j,k,l) $(ord(i)=ord(k) and ord(j)<=8 and
ord(l)>=ord(j)+1).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj3(i,j,k,l) $(ord(j)=ord(l) and ord(i)<=8 and
ord(k)>=ord(i)+1).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;

raj4(i,j,k,l) $(ord(i)=ord(k) and ord(j)<=8 and
ord(l)>=ord(j)+1).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj5(i,j,k,l) $(ord(j)=ord(l) and ord(i)<=8 and
ord(k)>=ord(i)+1).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
*käydään pikkuruudut läpi
raj6(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)<=3 and ord(l)<=3)..
x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj7(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)>=4 and ord(j)<=6 and ord(l)>=4 and
ord(l)<=6).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj8(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)>=7 and ord(j)<=9 and ord(l)>=7 and
ord(l)<=9).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj9(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)<=3 and
ord(l)<=3).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj10(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)>=4 and
ord(j)<=6 and ord(l)>=4 and ord(l)<=6).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj11(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)>=7 and
ord(j)<=9 and ord(l)>=7 and ord(l)<=9).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj12(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=1 and
ord(j)<=3 and ord(l)>=1 and ord(l)<=3).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj13(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=4 and
ord(j)<=6 and ord(l)>=4 and ord(l)<=6).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
raj14(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=7 and
ord(j)<=9 and ord(l)>=7 and ord(l)<=9).. x(i,j)-x(k,l) =G= 1-10*(1-y(i,j,k,l)) ;
```

```
raj15(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)<=3 and ord(l)<=3)..
x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj16(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)>=4 and ord(j)<=6 and ord(l)>=4 and
ord(l)<=6).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj17(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)<=3 and ord(k)<=3 and ord(j)>=7 and ord(j)<=9 and ord(l)>=7 and
ord(l)<=9).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj18(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)<=3 and
ord(l)<=3).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj19(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)>=4 and
ord(j)<=6 and ord(l)>=4 and ord(l)<=6).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj20(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=4 and ord(i)<=6 and ord(k)>=4 and ord(k)<=6 and ord(j)>=7 and
ord(j)<=9 and ord(l)>=7 and ord(l)<=9).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj21(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=1 and
ord(j)<=3 and ord(l)>=1 and ord(l)<=3).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj22(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=4 and
ord(j)<=6 and ord(l)>=4 and ord(l)<=6).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;
raj23(i,j,k,l) $(ord(i)<ord(k) and ord(j)<>ord(l) and
ord(i)>=7 and ord(i)<=9 and ord(k)>=7 and ord(k)<=9 and ord(j)>=7 and
ord(j)<=9 and ord(l)>=7 and ord(l)<=9).. x(k,l)-x(i,j) =G= 1-10*y(i,j,k,l) ;

Model sudoku3 "perinteinen sudoku, tapa 3" /all/ ;

Solve sudoku3 using MIP maximizing z ;

Display x.l ;
```

Sudoku-x

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;

*annetaan vinkkinumerot
x.fx('2','4','9')=1;
x.fx('2','8','4')=1;
x.fx('2','9','3')=1;
x.fx('3','1','8')=1;
x.fx('3','4','5')=1;
x.fx('4','2','2')=1;
x.fx('4','3','8')=1;
x.fx('5','1','1')=1;
x.fx('5','3','4')=1;
x.fx('5','5','6')=1;
x.fx('5','7','9')=1;
x.fx('5','9','8')=1;
x.fx('6','7','2')=1;
x.fx('6','8','6')=1;
x.fx('7','6','6')=1;
x.fx('7','9','2')=1;
x.fx('8','1','3')=1;
x.fx('8','2','7')=1;
x.fx('8','6','2')=1;
```

```
Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,k), raj3(j,k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k)
raj13(k), raj14(k) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj2(i,k).. sum(j, x(i,j,k)) =L= 1 ;
raj3(j,k).. sum(i, x(i,j,k)) =L= 1 ;

*käydään pikkuruudut läpi
raj4(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj5(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj6(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj7(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj8(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj9(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj10(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj11(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj12(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
*lävistäjien rajoitteet
raj13(k).. sum((i,j) $(ord(j)=ord(i)), x(i,j,k)) =L= 1 ;
raj14(k).. sum((i,j) $(ord(j)=10-ord(i)), x(i,j,k)) =L= 1 ;

Model sudokux "sudoku-x" /all/ ;

Solve sudokux using MIP maximizing z ;
```

Pariton-parillinen sudoku

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/
h(i,j) "harmaat ruudut" /1.2, 1.5, 1.7, 1.9, 2.3, 2.5, 2.6, 2.8, 3.1
3.3, 3.6, 3.9, 4.2, 4.3, 4.4, 4.7, 5.2, 5.4, 5.5, 5.6, 6.2, 6.7, 6.8
6.9, 7.1, 7.7, 7.8, 7.9, 8.1, 8.3, 8.4, 8.6, 9.1, 9.4, 9.5, 9.8/
v(i,j) "valkoiset ruudut" /1.1, 1.3, 1.4, 1.6, 1.8, 2.1, 2.2, 2.4
2.7, 2.9, 3.2, 3.4, 3.5, 3.7, 3.8, 4.1, 4.5, 4.6, 4.8, 4.9, 5.1, 5.3
5.7, 5.8, 5.9, 6.1, 6.3, 6.4, 6.5, 6.6, 7.2, 7.3, 7.4, 7.5, 7.6, 8.2
8.5, 8.7, 8.8, 8.9, 9.2, 9.3, 9.6, 9.7, 9.9/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;
*annetaan vinkkinumerot
x.fx('1','7','8')=1;
x.fx('1','8','5')=1;
x.fx('2','4','3')=1;
x.fx('2','5','4')=1;
x.fx('3','3','4')=1;
x.fx('4','3','2')=1;
x.fx('4','6','3')=1;
x.fx('5','5','8')=1;
x.fx('5','6','4')=1;
x.fx('5','7','3')=1;
x.fx('6','1','7')=1;
x.fx('6','3','3')=1;
x.fx('6','4','1')=1;
x.fx('6','5','9')=1;
x.fx('7','1','6')=1;
x.fx('7','3','1')=1;
x.fx('7','6','9')=1;
x.fx('7','8','4')=1;
x.fx('8','2','9')=1;

Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,k), raj3(j,k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k)
raj13(i,j), raj14(i,j), raj15(i,j), raj16(i,j) ;

obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj2(i,k).. sum(j, x(i,j,k)) =L= 1 ;
raj3(j,k).. sum(i, x(i,j,k)) =L= 1 ;
```

```
*käydään pikkuruudut läpi
raj4(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj5(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj6(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj7(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj8(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj9(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj10(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj11(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj12(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;

*parittomien ja parillisten lukujen sijainnit
raj13(i,j).. (x(i,j,'1')+x(i,j,'3')+x(i,j,'5')+x(i,j,'7')+x(i,j,'9'))
$(h(i,j)) =E= 0 ;
raj14(i,j).. (x(i,j,'2')+x(i,j,'4')+x(i,j,'6')+x(i,j,'8'))
$(h(i,j)) =L= 1 ;
raj15(i,j).. (x(i,j,'1')+x(i,j,'3')+x(i,j,'5')+x(i,j,'7')+x(i,j,'9'))
$(v(i,j)) =L= 1 ;
raj16(i,j).. (x(i,j,'2')+x(i,j,'4')+x(i,j,'6')+x(i,j,'8'))
$(v(i,j)) =E= 0 ;

Model evenodd "pariton-parillinen sudoku" /all/ ;

Solve evenodd using MIP maximizing z ;
```

Killer sudoku

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/
t "osiot" /1*27/
R(t,i,j) "mitkä ruudut (i,j) kuuluvat osioon t"
/1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.2.3, 2.1.5, 2.2.5, 2.3.5
2.4.5, 2.5.5, 3.1.6, 3.1.7, 3.1.8, 3.1.9, 3.2.7, 4.2.1, 4.2.2
5.2.4, 5.3.4, 6.2.6, 6.3.6, 7.2.8, 7.2.9, 8.3.1, 8.3.2, 8.3.3
8.4.2, 8.4.3, 9.3.7, 9.3.8, 9.3.9, 9.4.7, 9.4.8, 10.4.1, 10.5.1
11.4.4, 11.5.4, 12.4.6, 12.5.6, 13.4.9, 13.5.9, 14.5.2, 14.5.3
14.6.3, 15.5.7, 15.5.8, 15.6.7, 16.6.1, 16.6.2, 17.6.4, 17.6.5
17.6.6, 18.6.8, 18.6.9, 19.7.1, 19.8.1, 20.7.2, 20.7.3, 20.8.3
20.8.4, 21.7.4, 21.7.5, 21.7.6, 21.8.5, 21.9.5, 22.7.7, 22.7.8
22.8.6, 22.8.7, 23.7.9, 23.8.9, 24.8.2, 24.9.1, 24.9.2, 25.8.8
25.9.8, 25.9.9, 26.9.3, 26.9.4, 27.9.6, 27.9.7/ ;

Parameters
rr(t) "summan arvo" /1 28, 2 24, 3 24, 4 11, 5 7, 6 7, 7 11, 8 27
9 21, 10 13, 11 14, 12 9, 13 12, 14 10, 15 20, 16 10, 17 12
18 10, 19 4, 20 20, 21 35, 22 22, 23 5, 24 16, 25 14, 26 10, 27 9/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;

Equations
obj "kohdefunktio"
raj1(i,j), raj2(i,k), raj3(j,k), raj4(k), raj5(k), raj6(k)
raj7(k), raj8(k), raj9(k), raj10(k), raj11(k), raj12(k)
raj13(t,k), raj14(t) ;
```

```
obj.. z =E= sum((i,j,k),(x(i,j,k))) ;
raj1(i,j).. sum(k, x(i,j,k)) =L= 1 ;
raj2(i,k).. sum(j, x(i,j,k)) =L= 1 ;
raj3(j,k).. sum(i, x(i,j,k)) =L= 1 ;

*käydään pikkuruudut läpi
raj4(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj5(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj6(k).. sum((i,j) $(ord(i)>=1 and ord(i)<=3 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj7(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj8(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj9(k).. sum((i,j) $(ord(i)>=4 and ord(i)<=6 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;
raj10(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=1 and
ord(j)<=3),x(i,j,k)) =L= 1 ;
raj11(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=4 and
ord(j)<=6),x(i,j,k)) =L= 1 ;
raj12(k).. sum((i,j) $(ord(i)>=7 and ord(i)<=9 and ord(j)>=7 and
ord(j)<=9),x(i,j,k)) =L= 1 ;

raj13(t,k).. sum((i,j) $(R(t,i,j)), x(i,j,k)) =L= 1 ;
raj14(t).. sum((i,j,k) $(R(t,i,j)), ord(k)*x(i,j,k)) =E= rr(t) ;

Model killersudoku "killer sudoku" /all/ ;

Solve killersudoku using MIP maximizing z ;
```

Kakuro

```
Sets
i "vaakarivit" /1*9/
j "pystyrivit" /1*9/
k "luvut" /1*9/
t "summien indeksit" /1*32/

C(i,j) "tyhjät ruudut" /2.4, 2.5, 2.8, 2.9, 3.2, 3.3, 3.4, 3.5
3.6, 3.8, 3.9, 4.2, 4.3, 4.6, 4.7, 4.8, 5.3, 5.4, 5.6, 5.7
6.4, 6.5, 6.7, 6.8, 7.3, 7.4, 7.5, 7.8, 7.9, 8.2, 8.3, 8.5
8.6, 8.7, 8.8, 8.9, 9.2, 9.3, 9.6, 9.7/

R(t,i,j) "mitkä ruudut (i,j) kuuluvat osioon t"
/1.2.4, 1.3.4, 2.2.5, 2.3.5, 3.2.8, 3.3.8, 3.4.8, 4.2.9, 4.3.9
5.3.2, 5.4.2, 6.2.4, 6.2.5, 7.3.3, 7.4.3, 7.5.3, 8.3.6, 8.4.6
8.5.6, 9.2.8, 9.2.9, 10.3.2, 10.3.3, 10.3.4, 10.3.5, 10.3.6
11.3.8, 11.3.9, 12.4.7, 12.5.7, 12.6.7, 13.4.2, 13.4.3, 14.5.4
14.6.4, 14.7.4, 15.4.6, 15.4.7, 15.4.8, 16.5.3, 16.5.4, 17.5.6
17.5.7, 18.6.5, 18.7.5, 18.8.5, 19.6.8, 19.7.8, 19.8.8, 20.6.4
20.6.5, 21.7.3, 21.8.3, 21.9.3, 22.6.7, 22.6.8, 23.7.9, 23.8.9
24.7.3, 24.7.4, 24.7.5, 25.8.2, 25.9.2, 26.8.6, 26.9.6, 27.7.8
27.7.9, 28.8.7, 28.9.7, 29.8.2, 29.8.3, 30.8.5, 30.8.6, 30.8.7
30.8.8, 30.8.9, 31.9.2, 31.9.3, 32.9.6, 32.9.7/ ;

Parameters
rr(t) "summan t arvo" /1 5, 2 17, 3 20, 4 17, 5 10, 6 11, 7 13
8 9, 9 16, 10 31, 11 17, 12 14, 13 4, 14 13, 15 13, 16 12
17 4, 18 7, 19 21, 20 4, 21 9, 22 13, 23 9, 24 6, 25 10, 26 3
27 16, 28 11, 29 4, 30 18, 31 12, 32 7/ ;

Variables
x(i,j,k) "onko luku k ruudussa (i,j)"
z "kohdefunktiomuuttuja" ;

Binary variables x ;
Equations
obj "kohdefunktio"
raj1(i,j), raj2(t,k), raj3(t) ;

obj.. z =E= sum((i,j,k) $(C(i,j)),(x(i,j,k))) ;
raj1(i,j).. sum((k) $(C(i,j)), (x(i,j,k))) =L= 1 ;
raj2(t,k).. sum((i,j) $(R(t,i,j)), x(i,j,k)) =L= 1 ;
raj3(t).. sum((i,j,k) $(R(t,i,j)), ord(k)*x(i,j,k)) =E= rr(t) ;

Model kakuro /all/ ;

Solve kakuro using MIP maximizing z ;
```