Tampereen yliopisto

Sami Forss

GRAAFITEOREETTINEN JOHDANTO UNKARILAISEEN ALGORITMIIN

Tekniikan ja luonnontieteiden tiedekunta
Kandidaatintyö
Heinäkuu 2019

TIIVISTELMÄ

Sami Forss: Graafiteoreettinen johdanto unkarilaiseen algoritmiin
Kandidaatintyö
Tampereen yliopisto
Tekniikka ja luonnontieteet, TkK
Heinäkuu 2019

Tässä kandidaatintyössä käsitellään unkarilainen algoritmi graafiteoriaa hyödyntäen. Unkarilainen algoritmi on yksi sovitusongelman ratkaisevista graafiteorian metodeista. Sovitusongelman tavoitteena on löytää optimaalinen sovitus painotetulle kaksijakoiselle graafille.

Työn tavoitteena on todistaa, että unkarilainen algoritmi toimii. Algoritmi hyödyntää ratkaisun löytämisessä sekä yhtäsuuruus aligraafin olemassaoloa että algoritmin sisäistä toimintoa, joka kasvattaa sovituksen kokoa. Algoritmin toiminta osoitetaan käyttäen työssä todistettavia näihin käsitteisiin ja toimintoihin liittyviä graafiteoreettisia tuloksia. Työn lopussa esitellään yksinkertainen esimerkki unkarilaisen algoritmin toiminnasta käytännössä.

Avainsanat: unkarilainen algoritmi, graafiteoria, sovitusongelma, kaksijakoinen graafi, maksimisovitus, täydellinen sovitus, solmupainotus, yhtäsuuruus aligraafi

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

$G \setminus e$	graafista G on poistettu särmä e
$G \setminus v$	graafista G on poistettu solmu v ja siihen yhteydessä olevat särmät
$S \cap T$	joukkojen S ja T leikkaus
$S \cup T$	joukkojen S ja T yhdiste eli unioni
$W \subset V$	joukko W on joukon V aito osajoukko
$W \subseteq V$	joukko W on joukon V osajoukko
$\emptyset$	tyhjä joukko
$\sum_{a \in A} a$	joukon A alkioiden summa
$\{u, v\}$	alkioiden u ja v muodostama pari
$u \in V$	u on joukon V alkio
$\|V\|$	joukon V koko eli alkioiden määrä
TAU	Tampereen yliopisto (engl. Tampere University)

1 JOHDANTO

Ihmisille on ominaista halu mallintaa abstrakteja asioita, mikä teknologisen kehityksen aikana on helpottanut varsinkin tiedon hahmottamista. Yksi menestynyt ihmisen kehittämä tapa mallintaa tietoa on graafiteoria, jota hyödynnetään nykyään varsinkin tietotekniikan ja matematiikan opetuksessa. Graafiteorialle on ominaista tutkia algoritmien tehokkuuksia ja pyrkiä tuottamaan mahdollisimman tehokkaita algoritmeja ratkaisemaan ongelmia.

Tässä työssä perehdytään unkarilaiseen algoritmiin graafiteorian kautta. Algoritmi pystyy ratkaisemaan sovitusongelman, jossa tarkoituksena on sovittaa n kappaletta töitä n kappaleelle tekijöitä. Sovitusongelma voidaan kuvata kaksijakoisena graafina, jossa solmut kuvaavat sekä töitä että työntekijöitä ja painotetut särmät solmujen välillä ilmaisevat työntekijän tehokkuutta kyseiseen työhön. Unkarilaisen algoritmin merkittävyys ja monipuolisuus näkyy monissa käytännön tutkimuskohteissa. Sitä hyödynnetään esimerkiksi kappaleen lentoradan alkutilan määrittämiseen [1], maalien sovittamisessa aseille sodankäynnissä [2] ja ominaisuuksiltaan sopivan työntekijän valinnassa asiakaspalvelussa [6].

Työn tarkoituksena on saada lukijalle selvä kuva siitä, miksi unkarilainen algoritmi todella toimii. Sovitusongelman esittäminen graafiteoreettisesti antaa selkeän matemaattisen viitekehyksen tarkastella ongelmaa ja sen ratkaisevaa unkarilaista algoritmia. Työssä tarvittavat lauseet ja niiden todistukset on pyritty jäsentämään niin, että työn lopussa esiteltävän unkarilaisen algoritmin pseudokoodin toiminta olisi mahdollisimman helppo ymmärtää.

Työ on suunnattu matematiikkaa yliopistotasolla opiskeleville henkilöille, joilla on matematiikan perus-opintojen lisäksi tietämystä graafiteoriasta. Tästä syystä työssä esiteltävät graafieteorian peruskäsitteet ja -tulokset, eivät ole riittävä kokonaisuus tuottamaan täyttä ymmärrystä graafiteoriasta, vaan ne toimivat pohjana unkarilaiselle algoritmille.

Tämän kandidaatintyön luvussa 2 esitellään sovitusongelma, jonka unkarilainen algoritmi pystyy ratkaisemaan. Luvussa 3 esitellään työssä tarvittavat graafiteorian määritelmät. Luvun 4 tarkoituksena on sekä perehdyttää lukija graafien sovituksiin että todistaa muutama lause, joihin unkarilainen algoritmi pohjautuu. Lukujen 3 ja 4 teoria mahdollistaa sovitusongelman esittämisen graafina. Lopuksi luvussa 5 käydään yksytyiskohtaisesti läpi esimerkki unkarilaisen algoritmin toiminnasta.

2 SOVITUSONGELMA

Työssä esiteltävä unkarilainen algoritmi ratkaisee sovitusongelman, joka tässä luvussa esitellään yleisellä matemaattisella tasolla. Sovitusongelman lähtökohtana on kuinka sovittaa n kappaletta töitä n kappaleelle tekijöitä mahdollisimman tehokkaasti. Matemaattisesti ongelma voidaan kuvata seuraavilla lausekkeilla ja yhtälöillä [3].

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}; \tag{2.1a}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, ..., n; \tag{2.1b}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, ..., n; \tag{2.1c}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j = 1, ..., n. \tag{2.1d}$$

Jos muuttujat i ja j kuvaavat työntekijöitä ja työtehtäviä, yhtälöt (2.1b) ja (2.1c) rajoittavat yhdelle tekijälle yhden työn ja yhteen työhön yhden tekijän [3]. Käytännössä voidaan ajatella, että jokainen työntekijä pystyy tekemään vain yhden työn, mutta he tekevät eri työt eri tehokkuuksilla. Työssä käsitellään unkarilainen algoritmi maksimitehokkuuden kannalta, jolloin lauseke (2.1a) etsii maksimin. Saman lausekkeen muuttuja c_{ij} ilmaisee työn x_{ij} tehokkuutta.

Esimerkki 2.1. Alla olevassa taulukossa on esitetty yksinkertainen sovitusongelma, joka koostuu neljästä työntekijästä ja neljästä työstä.

Taulukko 2.1. *Esimerkki sovitusongelmasta*

	1'	2'	3'	4'
1	2	5	6	1
2	3	4	3	2
3	1	4	3	3
4	2	2	7	2

Taulukon rivit 1-4 kuvaavat työntekijöitä ja sarakkeet 1’-4’ töitä. Taulukon keskellä olevat numerot kuvaavat kyseisen rivin eli työntekijän tehokkuutta kyseiseen sarakkeeseen eli työhön. Täten esimerkiksi taulukon solu, jonka arvo on 7, vastaa työntekijän numero 4 tehokkuutta työhön 3. Tarkoituksena on siis yhdistää työntekijät ja työt niin, että yhteenlaskettu tehokkuus olisi mahdollisimman suuri.

Tässä kandidaatintyössä käsitellään koko ajan edellä mainitun esimerkin 2.1 mukaista sovitusongelmaa. Sovitusongelma esitetään seuraavassa luvussa hyödyntäen graafeja ja lopuksi luvussa 5 ongelma ratkaistaan hyödyntäen unkarilaista algoritmia.

3 JOHDATUS GRAAFEIHIN

Tässä luvussa esitellään työssä käytettävät graafiteorian peruskäsitteet ja niihin liittyvät notaatiot. Työssä seurataan Dieter Jungnickelin kirjan Graphs, Networks and Algorithms [4] merkintöjä, jotka ovat pääosin standardimerkintöjä graafiteorialle.

Määritelmä 3.1. *Yksinkertainen graafi (simple graph)* G on pari (V, E), joka koostu kahdesta joukosta V ja E. Joukko $V \neq \emptyset$ on äärellinen, ja sen alkiota kutsutaan *solmuiksi* (*vertex*). Joukko E on äärellinen, ja se koostuu järjestämättömistä pareista $\{u, v\}$, missä $u, v \in V$ ja $u \neq v$. Joukon E alkioita kutsutaan *särmiksi* (*edge*).

Yksinkertainen graafi $G = (V, E)$ ei salli silmukoita, eli solmuista ei voi olla särmiä itseensä. Tämän vuoksi oletetaan ehto $u \neq v$. Työssä käsitellään vain yksinkertasia graafeja, jolloin graafeissa kahden eri solmun $u, v \in V$ välillä voi olla vain yksi särmä. Multigraafit sallisivat myös useamman särmän samojen solmujen $u, v \in V$ välillä. Huomautuksena, että jos parit $\{u, v\}$ olisivat järjestettyjä, olisi kyseessä suunnattu graafi.

Määritelmä 3.2. Särmän $e = \{u, v\} \in E$ *päätesolmut (end vertices)* ovat $u \in V$ ja $v \in V$.

Määritelmä 3.3. *Painotettu graafi (weighted graph)* on graafin $G = (V, E)$ yleistys, jossa jokaiseen särmään $e = \{u, v\} \in E$ on liitetty luku. Tätä lukua sanotaan särmän *painoksi (weight)*, ja sitä merkitään $w(e)$ tai $w(\{u, v\})$. Graafin kokonaispainolla tarkoitetaan graafin $G = (V, E)$ särmien painojen summaa, jota merkitään $w(G) = \sum_{e \in E} w(e)$.

Eräs tapa hahmottaa särmien paino käytännössä on verrata graafia tiekarttaan, joka sisältää kaupunkeja. Ajatellaan kahta kaupunkia eli päätesolmua ja niiden välistä tietä eli särmää. Tällä tiellä on pituus, joka vastaan graafin kyseisen särmän painoa. Kaikkien teiden yhteispituus on täten sama kuin koko graafin paino. Jos kahden kaupungin välissä on useampi niitä yhdistävä tie, voidaan tilannetta kuvata jo mainitun multigraafin avulla tai yksinkertaistaa tilannetta valitsemalla esimerkiksi lyhin tie. Tällöin graafista muodostuu yksinkertainen graafi.

Määritelmä 3.4. *Kaksijakoinen graafi (bipartite graph)* on graafi $G = (V, E)$, jonka solmujoukko voidaan jakaa kahteen sellaiseen epätyhjään erilliseen joukkoon S ja T, että jokaisen särmän $e = \{u, v\} \in E$ toinen solmu kuuluu joukkoon S ja toinen joukkoon T. Joukkojen S ja T yhdiste on koko solmujoukko V ja leikkaus on tyhjäjoukko $\emptyset$.

Määritelmä 3.5. *Täydellinen kaksijakoinen graafi (complete bipartite graph)* $K_{m,n} = (V, E)$ on erityinen kaksijakoinen graafi 3.4, jossa solmujen välillä on särmä täsmälleen silloin, kun ne kuuluvat eri joukkoihin S tai T. Graafin $K_{m,n}$ alaindeksit m ja n kertovat solmujoukkojen S ja T koot eli $|S| = m$ ja $|T| = n$.

Edellä mainitun täydellisen kaksijakoisen graafin solmujoukkojen S ja T koot eivät ole riippuvaisia toisistaan, ja ne sisältävät aina vähintään yhden alkion. Useat työn todistukset ja luvun 5 esimerkki seuraavat kuitenkin lähdettä [4], jossa unkarilainen algoritmi käsitellään niin, että solmujoukkojen koot ovat samat eli $|S| = |T|$.

Määritelmä 3.6. Olkoot $G = (V, E)$ ja $H = (W, F)$ sellaisia graafeja, että $W \subseteq V$ ja $F \subseteq E$. Silloin graafi H on graafin G *aligraafi (subgraph)*. Jos lisäksi $W \subset V$ tai $F \subset E$, niin H on aito aligraafi.

Edellisestä määritelmästä 3.6 on hyvä huomata, että myös aligraafi toteuttaa graafin määritelmän 3.1. Täten aligraafin solmujoukko ei voi olla tyhjä, joten $W \neq \emptyset$. Aligraafi on siis graafin osa, joka myös itse on graafi.

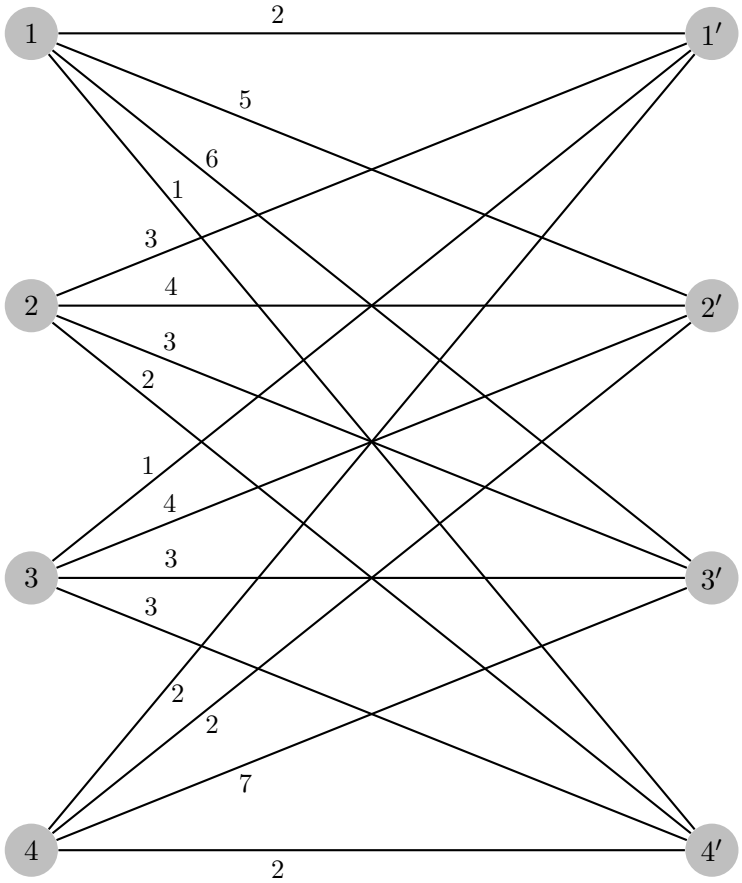
Määritelmä 3.7. *Solmupeite (vertex cover)* on graafin $G = (V, E)$ solmujoukon sellainen osajoukko $V' \subset V$, että jokaisen graafin $G = (V, E)$ särmän $e = \{u, v\} \in E$ päätesolmuista vähintään toinen on solmujoukossa V'.

Määritelmä 3.8. Graafia $G = (V, E)$ tai sen aligraafia kutsutaan *poluksi (path)* solmusta u_1 solmuun u_n tai vain poluksi, jos sen solmut voidaan laittaa järjestykseen u_1, u_2, ..., u_n siten, että sen särmät ovat $\{u_1, u_2\}$, $\{u_2, u_3\}$, ..., $\{u_{n-1}, u_n\}$ ja $u_i \neq u_j$, jos $i < j < n$. Jos $u_1 = u_n$, niin polkua kutsutaan *piiriksi (circuit)*.

Määritelmä 3.9. Graafi $G = (V, E)$ on *yhdistetty (connected)*, jos sen jokaisesta solmusta on polku muihin solmuihin.

Polut ja piirit ovat välttämättä yhdistettyjä graafeja. Kuitenkaan kaikki yhdistetyt graafit eivät ole polkuja eivätkä piirejä. Esimerkiksi hiiliatomin tetraedrinen rakenne voidaan kuvata graafina, joka on yhdistetty, mutta ei polku eikä piiri. Yhdistetyssä graafissa ei ole irrallisia aligraafeja.

Esimerkki 3.10. Alla olevassa kuvassa 3.1 on esitetty painotettu täydellinen kaksijakoinen graafi $K_{4,4} = (S \cup T, E)$. Graafin solmujoukko $V = S \cup T$ koostuu joukoista $S = \{1, 2, 3, 4\}$ ja $T = \{1', 2', 3', 4'\}$. Jo esiteltyjen määritelmien mukaan graafi on täten myös yksinkertainen ja yhdistetty. Särmien vieressä olevat numerot esittävät särmien painoja.



Kuva 3.1. *Painotettu täydellinen kaksijakoinen graafi* $(K_{4,4}, w)$

Kuvan graafi on esimerkin 2.1 taulukko kuvattuna graafiksi. Se toimii myös täten luvun 5 esimerkkinä, kun työssä sovelletaan unkarilaista algoritmia sovitusongelman ratkaisemiseksi.

4 GRAAFIEN MAKSIMISOVITUS

Tämän luvun lähtökohtana on esitellä lukijalle tarvittavat tiedot unkarilaisen algoritmin ymmärtämiseen. Lisäksi todistetaan graafiteoreettisia tuloksia, joiden avulla osoitetaan, miksi kyseinen algoritmi toimii. Jotta luvussa 2 esitelty sovitusongelma voitaisiin ratkaista graafien avulla, tarvitaan lisäksi muutama käsite, jotka esitellään seuraavaksi.

4.1 Graafien sovitukset

Määritelmä 4.1. *Sovitus (matching)* on Graafin $G = (V, E)$ särmäjoukon E sellainen osajoukko $M \subseteq E$, että jokainen $v \in V$ on enintään yhden joukon M särmän päätesolmu.

Määritelmä 4.2. *Täydellinen sovitus (perfect matching)* on sovitus M, jossa jokainen graafin $G = (V, E)$ solmu $v \in V$ on jonkin särmäjoukon M särmän päätesolmu.

Määritelmästä 4.1 seuraa, että graafin $G = (V, E)$ sovituksessa M täytyy esiintyä päätesolmuina parillinen määrä solmuja. Edelleen määritelmän 4.2 mukaisesti täytyy täydellisessä sovituksessa jokaisen solmun $v \in V$ olla enintään ja vähintään yhden särmän $e = \{u, v\} \in E$ päätesolmu. Kaksijakoisessa graafissa tämä tarkoittaa myös sitä, että graafin $G = (S \cup T, E)$ solmujoukkojen kokojen $|S|$ ja $|T|$ täytyy olla yhtäsuuret. Graafeissa voi tietenkin olla pariton määrä solmuja, jolloin määritelmän 4.2 mukainen täydellinen sovitus ei sovi kuvaamaan sovituksen mahtavuutta. Seuraava määritelmä antaa keinon kuvata kyseistä tilannetta.

Määritelmä 4.3. Kaksijakoisen graafin $G = (S \cup T, E)$ sovituksen maksimaalista kardinaliteettia rajoittaa sen pienemmän solmujoukon koko eli $\min\{|S|, |T|\}$. Kardinaliteetiltaan tälläistä sovitusta kutsutaan *täydeksi sovitukseksi (complete matching)*.

Edellä mainituista määritelmistä 4.2 ja 4.3 seuraa, että jokainen täydellinen sovitus on myös täysi sovitus, mutta täysi sovitus ei välttämättä ole täydellinen sovitus. Kun täyden sovituksen määritelmässä graafin $G = (S \cup T, E)$ solmujoukkojen S ja T koot ovat yhtäsuuret, saadaan täydellinen sovitus.

Määritelmä 4.4. Olkoon $G = (V, E)$ kaksijakoinen graafi, jonka särmille pätee *painofunktio (weight funktion)* $w : E \to \mathbb{R}$. Graafin G sovituksen M paino $w(M)$ määritellään

$$w(M) = \sum_{e \in M} w(e).$$

Sovitus M on *maksimisovitus (maximal weighted matching)*, jos jokaiselle graafin G sovitukselle $M^{'}$ pätee $w(M) \geq w(M^{'})$.

Maksimisovitus ei tietenkään voi sisältää negatiivisia särmien painoja, joten oletetaan kaikkien painojen $w(e)$ olevan positiivisia [4]. Jos graafin $G = (V, E)$ sisältää negatiivisia särmäpainoja $w(e)$, voidaan jokaiseen särmään lisätä pienimmän särmän painon itseisarvo, jolloin lopulta jokainen särmäpaino on positiivinen.

Nyt määritelmien 4.1-4.4 avulla voidaan esimerkin 2.1 ongelma esittää myös graafeja käyttäen. Oletetaan ensin, että graafi $G = (V, E)$ on täydellinen kaksijakoinen graafi $K_{n,n}$, eli graafi voidaan jakaa kahteen yhtäsuureen solmujoukkoon kuvastamaan töitä ja tekijöitä, kuten esimerkissä 3.10. Sovittamalla tähän graafiin täydellinen sovitus ja maksimisovitus, saadaan kaavat (2.1a)-(2.1d) mallinnettua graafeiksi. Graafin $G = (V, E)$ erilliset solmujoukot kuvastavat tekijät ja työt erillisinä joukkoina. Täydellisen sovituksen määritelmä 4.2 yhdistää yhden työn ja yhden tekijän eli kattaa yhtälöt (2.1b) ja (2.1c). Määritelmän 4.4 mukainen maksimisovitus on graafiteorian vastaavuus kaavalle (2.1a).

Esimerkki 4.5. Jos esimerkin 3.10 graafista valitaan esimerkiksi särmät $\{1, 1'\}, \{2, 2'\}$ ja $\{3, 3'\}$, täyttää se sovituksen määritelmän, mutta sovitus ei ole täydellinen, sillä solmut $4 \in S$ ja $4' \in T$ eivät kuulu sovitukseen. Lisäämällä sovitukseen särmä $\{4, 4'\}$ saadaan siitä täydellinen sovitus, mutta sovitus ei ole maksimisovitus, sillä voidaan löytää useita sovituksia, joilla on suurempi paino. Luvussa 5 löydetään esimerkin 3.10 graafille maksimisovitus.

4.2 Solmupainotus

Jotta luvun 5 unkarilaisen algoritmin toiminta olisi matemaattisesti täsmällisesti perusteltu, tarvitaan muutama lause tukemaan edellä mainittua teoriaa. Luvun seuraavat lauseet ja niiden todistukset on esitetty lähteessä [4] ellei toisin mainita, ja ne käsitellään kyseisen kirjan tapaan.

Olkoon $G = (V, E)$ täydellinen kaksijakoinen graafi $K_{n,n}$, jonka solmujoukko $V = S \cup T$. Täten kaksijakoisuuden perusteella $S \cap T = \emptyset$. Olkoot joukot $S = \{1, ..., n\}$ ja $T = \{1', ..., n'\}$. Määritellään ei-negatiivinen painofunktio w symmetrisen matriisin $W = (w_{ij})$ avulla seuraavasti: alkio w_{ij} on särmän $\{i, j'\}$ paino. Reaaliarvoista vektoriparia $\vec{u} = (u_1, ..., u_n)$ ja $\vec{v} = (v_1, ..., v_n)$ kutsutaan *mahdolliseksi solmupainotukseksi (feasible node-weighting)*, jos seuraava yhtälö pätee:

$$u_i + v_j \geq w_{ij} \quad \forall i, j = 1, ..., n. \tag{4.1}$$

Toisin sanoen särmien päätesolmujen arvojen summan täytyy aina olla vähintään kyseisen särmän paino. Tietenkin mahdollinen solmupainotus voidaan valita varmasti suuremmaksi kuin särmien painot, mutta pian huomaamme, miksi näin ei tehdä. Merkitään kaikki mahdolliset solmupainotukset $(\vec{u}, \vec{v})$ joukoksi $\vec{F}$.

Lause 4.6. *Olkoon maksimisovitus D. Jokaiselle mahdolliselle solmupainotukselle $(\vec{u}, \vec{v})$ ja jokaiselle graafin G sovitukselle M on voimassa*

$$w(M) \leq w(D) \leq \sum_{i=1}^{n} (u_i + v_i) \tag{4.2}$$

Todistus. Laskemalla yhteen yhtälö (4.1) yli kaikkien sovituksen M särmien saadaan epäyhtälö $w(M) \leq \sum_{i=1}^{n}(u_i+v_i)$. Maksimisovitus D on jo määritelmän 4.4 mukaan vähintään minkä tahansa muun sovituksen paino. Koska mahdollisen solmupainotuksen täytyy päteä kaikille sovituksille, pätee se täten myös maksimisovitukselle. □

Lauseesta 4.6 seuraa suoraan, että jos pystymme löytämään yhtäsuuruuden mahdollisen solmupainotuksen $(\vec{u}, \vec{v})$ ja täydellisen sovituksen M välille, täytyy sovituksen M olla välttämättä maksimisovitus D. Luvun 5 unkarilaisen algoritmin päämäärä on löytää täydellinen sovitus, jolla yhtäsuuruus saavutetaan. Tarkastellaan seuraavaksi lausetta 4.6 rajatummassa muodossa, mihin tarvitsemme avuksi seuraavat kaksi lausetta. Ensimmäinen lause ja sen todistus on esitelty lyhyesti lähteessä [7] ja laajemmin lähteen [4] sivulla 214, joten käydään se läpi lähdettä [4] mukaillen.

Otetaan seuraavia lauseita varten käyttöön kaksi uutta merkintää. Merkitään graafin $G = (V, E)$ sovituksen maksimaalista kardinaliteettia kirjaimella ν ja määritelmän 3.7 mukaisen solmupeitetteen minimaalista kardinaliteettia kirjaimella τ. Nämä vastaavat särmien määrää maksimaalisessa sovituksessa ja solmujen määrää minimaalisessa solmupeitteessä.

Apulause 4.7. *Jos $\nu(G) = \tau(G)$, M on graafin G maksimisovitus ja W saman graafin minimisolmupeite, niin jokainen minimisolmupeitteen W solmu on täsmälleen yhden sovituksen M särmän päätepiste.*

Todistus. Koska minimisolmupeitteestä W löytyy jokaisen särmän päätepiste, niin erityisesti kaikille särmille $e = \{u, v\} \in M$ on voimassa $u \in W$ tai $v \in W$. Koska maksimisovituksen M särmillä ei saa olla yhteisiä päätepisteitä, niin on oltava, että kaikilla särmillä $e = \{u, v\} \in M$ vain toinen ehdoista $u \in W$ tai $v \in W$ toteutuu, mutta ei molemmat. Täten jokainen minimisolmupeitteen W solmu on jonkin maksimisovituksen M särmän päätepiste. □

Lause 4.8. *(Königin lause (König's theorem)) Maksimaalisen sovituksen M särmien määrä vastaa minimaalisen solmupeitteen solmujen määrää kaksijakoisessa graafissa $G = (V, E)$ eli $\nu(G) = \tau(G)$.*

Todistus. Todistetaan väite kahdessa osassa. Todistetaan ensin, että kaksijakoiselle graafille on voimassa $\nu(G) \leq \tau(G)$. Koska muuttuja $\nu(G)$ vastaa särmien määrää maksimaalisessa sovituksessa, käydään läpi graafin $G = (V, E)$ sovituksen särmät yksi kerrallan. Vähintään toisen särmän päätesolmuista on kuuluttava minimaaliseen solmupeitteeseen määritelmän 3.7 mukaisesti. Tällöin jokaisen särmän kohdalla kasvaa solmupeitteen koko vähintään yhdellä, koska määritelmän 4.1 mukaan jokainen solmu on korkeintaan yhden sovituksen särmän päätesolmu. Tästä seuraa epäyhtälö $\nu(G) \leq \tau(G)$.

Todistetaan seuraavaksi, että kaksijakoisille graafeille on voimassa $\nu(G) \geq \tau(G)$. Tehdään vastaoletus, että epäyhtälö $\nu(G) < \tau(G)$ toteutuu. Olkoon $G = (V, E)$ solmujen suhteen minimaalinen vastaesimerkki, jossa on pienin mahdollinen määrä särmiä. On helppo todeta, että tällöin graafi G on yhdistetty eikä se ole piiri eikä polku. Tällöin graafilla on yksi solmu, jonka aste on vähintään 3, ja olkoon tämä solmu $u \in V$. Olkoon tämän solmun vierussolmu $v \in V$. Koska graafi $G = (V, E)$ valittiin minimaaliseksi vastaesimerkiksi, toteuttaa graafi $G \setminus v$ yhtälön $\nu(G \setminus v) = \tau(G \setminus v)$.

Oletetaan seuraavaksi, että graafien G ja $G \setminus v$ maksimaalisten sovitusten kardinaliteetit noudattavat seuraavaa epäyhtälöä $\nu(G \setminus v) < \nu(G)$. Tällöin graafin $G \setminus v$ solmupeitteeseen voidaan lisätä solmu v, mikä tuottaa graafin G solmupeitteen. Minimaalisuuden takia graafin $G \setminus v$ solmupeitteen minimaalinen kardinaliteetti on suurempi tai yhtä suuri kuin graafin G minimaalinen kardinaliteetti vähennettynä yhdellä. Tästä saadaan epäyhtälö $\tau(G)-1 \leq \tau(G\setminus v)$, joka voidaan muokata seuraavaksi yhtälöksi $\tau(G) \leq \tau(G\setminus v)+1$. Yhdistämällä tämä aikaisemman epäyhtälön kanssa saadaan epäyhtälö $\tau(G) \leq \tau(G \setminus v) + 1 = \nu(G \setminus v) + 1 \leq \nu(G)$. Epäyhtälö on ristiriidassa vastaoletuksen kanssa.

Koska epäyhtälö $\nu(G \setminus v) < \nu(G)$ ei voi toteutua, jää tarkasteltavaksi tapaus $\nu(G \setminus v) = \nu(G)$. Tämä tarkoittaa, että on olemassa graafin G maksimaalinen sovitus M, jossa yksikään särmä ei ole yhteydessä solmuun v. Koska solmun u aste on vähintään kolme, graafissa G voidaan valita särmä $e \notin M$, joka on yhteydessä solmuun u, mutta ei solmuun v. Tällöin graafin G minimaalisuudesta johtuen aligraafi $G \setminus e$ toteuttaa lauseen ja saadaan yhtälö $\nu(G) = \nu(G \setminus e) = \tau(G \setminus e)$. Olkoon W' graafin $G \setminus e$ solmupeite, jonka kardinaliteetti on $\nu(G)$. Sovelletaan graafin $G \setminus e$ solmupeitteeseen W' ja sovitukseen M edellä todistettua apulausetta 4.7, jolloin solmu v ei voi kuulua solmupeitteeseen W', sillä yksikään sovituksen M särmä ei ole yhteydessä solmuun v. Siten solmupeitteeseen W' täytyy sisältyä särmän $uv \neq e$ päätesolmu u ja W' on täten myös solmupeite graafille G. Tämä johtaa yhtälöön $\nu(G) = \nu(G \setminus e) = \tau(G \setminus e) = \tau(G)$, joka on ristiriidassa vastaoletuksen kanssa.

Täten kahden edellisen kohdan perusteella oletus $\nu(G \setminus v) < \nu(G)$ on väärä, joten välttämättä $\nu(G) \geq \tau(G)$. Koska myös $\nu(G) \leq \tau(G)$, seuraa siitä, että kaksijakoiselle graafille pätee $\nu(G) = \tau(G)$. □

Lause 4.9. *Olkoon $G = (S \cup T, E)$ kaksijakoinen graafi, jossa $|S| \geq |T|$. Olkoon $J \subset T$, ja käytetään joukon J vierussolmujen joukolle merkintää $\Gamma(J)$. Graafilla G on olemassa*

täysi sovitus jos ja vain jos

$$|\Gamma(J)| \geq |J| \quad \forall J \subset T. \tag{4.3}$$

Todistus. Todistetaan lause kahdessa osassa. Oletetaan ensin, että M on graafin G täysi sovitus ja J joukon T mikä tahansa osajoukko. Olkoon $E(J)$ täyden sovituksen särmien joukko, jonka särmät ovat yhteydessä joukon J solmuihin. Tällöin joukon $E(J)$ päätesolmut, jotka kuuluvat joukkoon S, muodostavat kardinaliteetiltaan $|J|$ kokoisen joukon $\Gamma(J)$ osajoukon, jolloin $|\Gamma(J)| \geq |J|$ toteutuu.

Oletetaan seuraavaksi, että yhtälö toteutuu, mutta graafin G sovituksen maksimaalinen kardinaliteetti on pienempi kuin $|T|$. Edellisen lauseen 4.8 nojalla kaksijakoiselle graafille on voimassa $\nu(G) = \tau(G)$, joten myös minimaalisen solmupeitteen koko on pienempi kuin $|T|$. Merkitään solmupeitettä $X = S' \cup T'$. Tämä solmupeite toteuttaa seuraavat ehdot $S' \subset S$, $T' \subset T$ ja $|S'| + |T'| < |T|$. Tällöin jokaisen särmän uv, jonka päätesolmu v kuuluu joukkoon $T \setminus T'$, päätesolmu u kuuluu joukkoon S', eli solmujen määrä joukossa S' on vähintään solmujen määrä joukossa $\Gamma(T \setminus T')$. Yhdistämällä edellä mainitut yhtälöt, saadaan epäyhtälön

$$|\Gamma(T \setminus T')| \leq |S'| < |T| - |T'| = |T \setminus T'| \tag{4.4}$$

mukainen ristiriita oletuksen kanssa, jolloin yhtälö (4.3) toteutuu. □

Määritelmä 4.10. Olkoon $(\vec{u}, \vec{v})$ mahdollinen solmupainotus ja $H_{\vec{u},\vec{v}}$ graafin $G = (V, E)$ aligraafi, jonka särmiä $e \in E$ ovat vain ne $\{ij'\}$, jotka toteuttavat ehdon $u_i + v_j = w_{ij}$. Tällöin $H_{\vec{u},\vec{v}}$ on *yhtäsuuruus aligraafi (equality subgraph)*.

Lause 4.11. *Olkoon $H = H_{\vec{u},\vec{v}}$ yhtäsuuruus aligraafi mahdolliselle solmupainotukselle $(\vec{u}, \vec{v}) \in \vec{F}$ ja olkoon graafin maksimisovitus D. Yhtälö*

$$\sum_{i=1}^{n} (u_i + v_i) = w(D) \tag{4.5}$$

on voimassa täsmälleen silloin, kun aligraafilla H on täydellinen sovitus. Tässä tapauksessa jokainen aligraafin H täydellinen sovitus M on maksimisovitus D graafille $G = (S \cup T, E)$.

Todistus. Olkoon $\sum_{i,j=1}^{n}(u_i + v_j) = w(D)$ ja oletetaan ensiksi, että H ei sisällä täydellistä sovitusta. Hyödynnetään seuraavaksi lausetta 4.9, mutta käännetään joukot S ja T toisinpäin. Olkoot $J \subset S$ ja merkitään niiden solmujen $j' \in T$ joukkoa, jotka ovat yhteydessä johonkin joukon J solmuihin $i \in J$, joukolla $\Gamma(J)$. Näin ollen lauseen 4.9 mukaan on olemassa joukon S osajoukko J, jolla pätee $|\Gamma(J)| < |J|$. Olkoot lisäksi

$$\delta = \min\{u_i + v_j - w_{ij} : i \in J,\ j' \notin \Gamma(J)\} \tag{4.6}$$

ja määritellään uusi mahdollinen solmupainotus $(\vec{u'}, \vec{v'}) \in \vec{F'}$ seuraavasti:

$$u'_i = \begin{cases} u_i - \delta & \text{jos } i \in J \\ u_i & \text{jos } i \notin J \end{cases} \tag{4.7}$$

$$v'_j = \begin{cases} v_j + \delta & \text{jos } j' \in \Gamma(J) \\ v_j & \text{jos } j' \notin \Gamma(J). \end{cases} \tag{4.8}$$

Huomataan, että ehto $u'_i + v'_j \geq w_{ij}$ pätee, kun $i \in J$ ja $j' \notin \Gamma(J)$, sillä $u'_i = u_i - \delta$ ja $\delta \leq u_i + v_j - w_{ij}$, jolloin $w_{ij} \leq (u_i - \delta) + v_j = u'_i + v'_j$. Jos $i \notin J$ tai $j' \in \Gamma(J)$, niin $u'_i + v'_j = u_i + v_j \geq w_{ij}$, joten myös $(\vec{u'}, \vec{v'}) \in \vec{F'}$ toteuttaa yhtälön (4.2) ja on siten mahdollinen solmupainotus. Tästä kuitenkin seuraa ristiriita alkuperäisen väittämän kanssa, sillä

$$w(D) \leq \sum_{i,j=1}^{n} (u'_i + v'_j) = \sum_{i,j=1}^{n} (u_i + v_j) - \delta|J| + \delta|\Gamma(J)| = w(D) - \delta(|J| - |\Gamma(J)|) < w(D). \tag{4.9}$$

Oletetaan seuraavaksi, että yhtäsuuruus aligraafi H sisältää täydellisen sovituksen M. Tällöin yhtälössä (4.1) saavutetaan yhtäsuuruus kaikilla $\{ij'\} \in M$. Täten summaamalla yli kaikkien sovituksen särmien saadaan yhtäsuuruus myös yhtälössä (4.2). Näin ollen jokainen yhtäsuuruus aligraafin H täydellinen sovitus on maksimisovitus painotetulle graafille (G, w). □

5 UNKARILAINEN ALGORITMI

Tässä luvussa käydään läpi unkarilainen algoritmi, joka löytää täydelliselle kaksijakoiselle graafille maksimisovituksen. Algoritmi käsitellään askel kerrallaan ja demonstroidaan sen toimintaa esimerkin avulla. Edellisen luvun lauseiden avulla osoitamme, että algoritmi todella toimii. Tämän algoritmin kehitti amerikkalainen matemaatikko Harold Kuhn unkarilaisten matemaatikkojen Dénes Königin ja Jenö Egerváryn töiden pohjalta [4]. Alkuperäisen Kuhnin tieteellisen julkaisun löytää lähteestä [5].

5.1 Algoritmin läpikäynti

Esiteltävän unkarilaisen algoritmin pseudokoodi sisältää useita muuttujia ja algoritmille ominaisia merkintöjä, jotka eivät välttämättä ole yleisesti käytössä. Alla olevaan taulukkoon on koottu algoritmin tärkeimmät merkinnät ja niiden selitykset. Muuttujien väliset yhteydet ja niiden toiminta algoritmin ajon aikana on selitetty tarkemmin itse algoritmin jälkeen.

Taulukko 5.1. *Unkarilaisen algoritmin merkinnät*

Merkintä	Selitys
$mate(i) = j'$	särmä ij' kuuluu sovitukseen M
$nrex$	avoimien solmujen määrä sovituksessa M
u_i	solmun $i \in S$ solmupaino
v_j	solmun $j' \in T$ solmupaino
Q	joukko käsiteltävistä solmuista $i \in S$
$m(i)$	totuusarvo $true$, jos solmu $i \in S$ vielä käsiteltävänä
δ_j	solmun j' potentiaali eli pienin arvo lausekkeelle $u_i + v_j - w_{ij}$
$p(j)$	ensimmäinen solmu $i \in S$, jossa potentiaali δ_j saavutetaan
aug	totuusarvo onko toiminto AUGMENT suoritettu

Olkoon graafi $G = (V, E)$ täydellinen kaksijakoinen graafi, jossa solmujoukko $V = S \cup T$ on jaettu kahteen yhtäsuureen osaan $S = \{1, \ldots, n\}$ ja $T = \{1', \ldots, n'\}$. Jokaiseen särmään ij' on liitetty positiivinen paino w_{ij}. Alla oleva unkarilainen algoritmi löytää optimaalisen sovituksen graafille $G = (V, E)$. Algoritmi löytyy lähteen [4] sivuilta 423-424.

Algoritmi 1 UNKARILAINEN ALGORITMI Osa 1

```
1: procedure HUNGARIAN(N,W;MATE)
2:     for v ∈ V do
3:         mate(v) ← 0
4:     end for
5:     for i = 1 to n do
6:         u_i ← max{w_ij : j = 1, ..., n}; v_i ← 0
7:     end for
8:     nrex ← n;
9:     while nrex ≠ 0 do
10:        for i = 1 to n do
11:            m(i) ← false; p(i) ← 0; δ_i ← ∞
12:        end for
13:        aug ← false; Q ← {i ∈ S : mate(i) = 0};
14:        repeat
15:            remove an arbitrary vertex i from Q; m(i) ← true; j ← 1;
16:            while aug = false and j ≤ n do
17:                if mate(i) ≠ j' then
18:                    if u_i + v_j − w_ij < δ_j then
19:                        δ_j ← u_i + v_j − w_ij; p(j) ← i;
20:                        if δ_j = 0 then
21:                            if mate(j') = 0 then
22:                                AUGMENT(mate, p, j'; mate);
23:                                aug ← true; nrex ← nrex − 1
24:                            else Q ← Q ∪ {mate(j')}
25:                            end if
26:                        end if
27:                    end if
28:                end if
29:                j ← j + 1
30:            end while
31:            if aug = false and Q = ∅ then
32:                J ← {i ∈ S : m(i) = true}; K ← {j' ∈ T : δ_j = 0};
33:                δ ← min{δ_j : j' ∈ T \ K};
34:                for i ∈ J do
35:                    u_i ← u_i − δ
36:                end for
37:                for j' ∈ K do
38:                    v_j ← v_j + δ
39:                end for
40:                for j' ∈ T \ K do
41:                    δ_j ← δ_j − δ
42:                end for
43:                X ← {j' ∈ T \ K : δ_j = 0}
44:                if mate(j') ≠ 0 for all j' ∈ X then
45:                    for j' ∈ X do
46:                        Q ← Q ∪ {mate(j')}
47:                    end for
```

Algoritmi 1 UNKARILAINEN ALGORITMI Osa 2

```
48:                 else choose j' ∈ X with mate(j') = 0;
49:                     AUGMENT(mate, p, j'; mate);
50:                     aug ← true; nrex ← nrex − 1
51:                 end if
52:             end if
53:         until aug = true
54:     end while
55: end procedure
```

Algoritmi 1 UNKARILAINEN ALGORITMI Osa 3

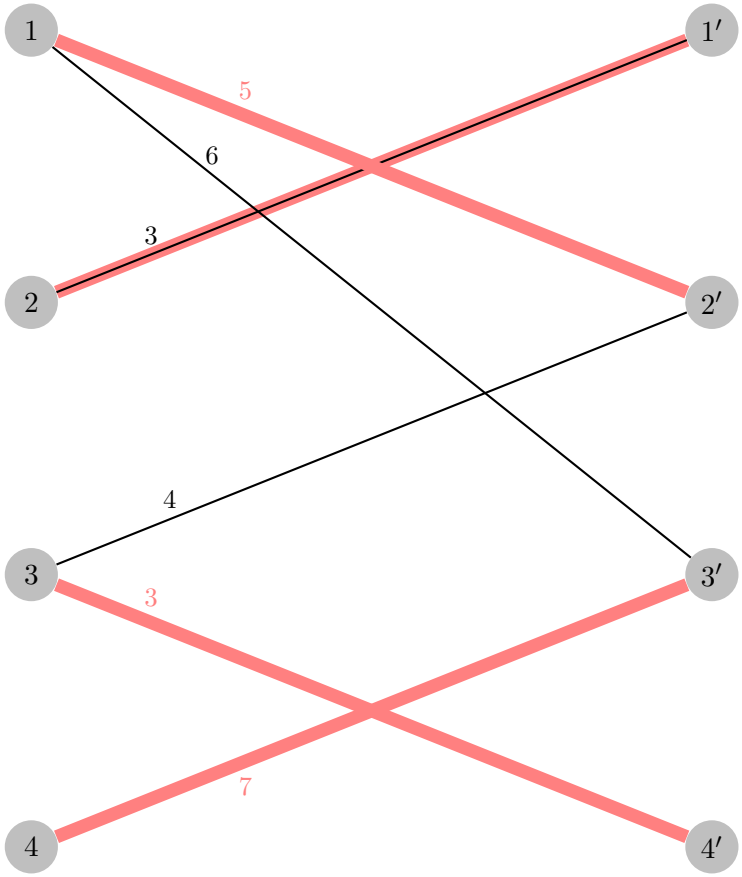
```
56: procedure AUGMENT(MATE,P,J';MATE)
57:     repeat
58:         i ← p(j); mate(j') ← i; next ← mate(i); mate(i) ← j';
59:         if next ≠ 0 then
60:             j' ← next
61:         end if
62:     until next = 0
63: end procedure
```

Käydään seuraavaksi läpi algoritmin merkintöjä, sillä useat niistä ovat ominaisia vain lähteelle [4]. Muuttuja $mate$ on *taulukko (array)*, jossa $mate(i) = j$ ja $mate(j) = i$, kun särmä ij kuuluu sovitukseen M. Jos solmu k on *avoin (exposed)*, se ei ole yhteydessä sovituksen M yhteenkään solmuun, jolloin $mate(k) = 0$. Kyseinen solmu k ei siis ole yhdenkään särmän päätesolmu sovituksessa. Algoritmin riveillä 2-4 alustetaan alkutilanteessa jokainen solmu avoimeksi, jolloin sovitus on tyhjä. Koska kyseessä on määritelmän 4.1 mukainen sovitus, voi jokainen solmu olla yhteydessä vain yhteen solmuun taulukossa $mate$. Muuttuja $nrex$ kuvaa avoimien solmujen määrää sovituksen M näkökulmasta. Algoritmin rivillä 8 muuttuja $nrex$ alustetaan arvoksi n eli yhtäsuureksi, kun solmujen määrä graafin solmujoukoissa S ja T. Rivin 9 silmukka loppuu, kun muuttuja $nrex$ saa arvon 0, jolloin algoritmi päättyy. Algoritmin riveiltä 23 ja 50 huomataan, että muuttujan $nrex$ arvo vähenee yhdellä joka kerta, kun *toiminto (procedure)* AUGMENT suoritetaan, joten algoritmi päättyy äärellisessä ajassa.

Toiminto AUGMENT on unkarilaiselle algoritmille ominainen prosessi, joka on esitetty algoritmin riveillä 56-63. Toiminto lähtee liikkeelle avoimesta solmusta ja päättyy avoimeen solmuun muodostaen yhden solmuparin lisää sovitukseen. Täten jokainen toiminto AUGMENT kasvattaa sovituksen kardinaliteettia ja vie sovitusta kohti täydellistä sovitusta.

Esimerkki 5.1. Alla olevassa kuvassa 5.1 on esitetty yksinkertainen esimerkki toiminnosta AUGMENT. Mustalla piirretyt särmät ovat osa sovitusta ennen toimintoa AUGMENT ja punaisella piirretyt särmät toiminnon jälkeen. Kuvasta nähdään, että sovituksen koko on kasvanut yhdellä toiminnon AUGMENT jälkeen, sillä punaisia särmiä on yksi enemmän kuin mustia. On hyvä huomata että toiminto AUGMENT ei ole kohdistunut särmään $\{2, 1'\}$, joka pysyy ennallaan myös toiminnon jälkeen. Kuvan tilanne on osa luvun 5 esi-

merkkiä, jossa sovitusongelma ratkaistaan unkarilaista algoritmia hyödyntäen.



Kuva 5.1. *Esimerkki toiminnosta AUGMENT*

Unkarilaisen algoritmin alussa alustetaan mahdollinen solmupainotus $(\vec{u}, \vec{v}) \in \vec{F}$, joka usein valitaan seuraavasti

$$v_1 = ... = v_n = 0 \quad \text{ja} \quad u_i = \max\{w_{ij} : j = 1, ..., n\} \quad (\text{kun} \quad i = 1, ..., n). \tag{5.1}$$

Algoritmissa tämä on toteutettu riveillä 5-7, ja se varmistaa lähtökohdaksi, että yhtälön (4.1) mukainen ehto toteutuu. Lisäksi valinta varmistaa jonkin yhtäsuuruus aligraafin muodostumisen.

Edellä mainitut rivit 2-8 alustavat lähtötilanteen algoritmille, mutta algoritmin runkoa varten tarvitsemme vielä muutamia määritelmiä. Algoritmin muuttuja δ_j kuvaa jokaisen solmun $j' \in T$ *potentiaalia (potential)*, joka on sen hetkinen pienin arvo lausekkeelle $u_i + v_j - w_{ij}$. Tämän lisäksi muuttuja $p(j)$ ilmaisee ensimmäisen solmun $i \in S$, jossa edellä mainittu potentiaali δ_j saavutetaan. *Joukon (set)* Q tehtävä on pitää kirjaa, mitkä solmut halutaan vielä käsitellä, ja muuttujan $m(i)$ totuusarvo *true* ilmaisee, että solmu i on vielä käsiteltävänä. Viimeisenä muuttujan aug totuusarvo ilmaisee, onko toiminto AUGMENT suoritettu.

Käsitellään seuraavaksi unkarilaisen algoritmin runko, jota toistetaan useampaan otteeseen sovitusta muodostettaessa. Rivit 10-13 alustavat algoritmissa tarvittavien paramet-

rien arvot jokaisessa silmukassa uudelleen, missä huomiota kannattaa kiinnittää varsinkin joukkoon Q, johon lisätään vain joukon S avoimet solmut. Rivin 14 silmukasta poistutaan vasta, kun parametri *aug* saa arvon *tosi*, jolloin toiminto AUGMENT on suoritettu. Yksinkertaisuuden vuoksi tässä työssä rivillä 15 poistetaan aina luvultaan pienin solmu joukosta Q, vaikka algoritmissä poisto tehdäänkin satunnaiselle solmulle.

Rivin 16 silmukan tehtävä on etsiä joukon S solmuille optimaalinen pari joukosta T. Ensimmäisellä kerralla rivien 17 ja 18 ehdot toteutuvat välttämättä rivien 10-13 alustusten perusteella, jolloin ensimmäinen solmu $i \in S$ yhdistyy solmuun $j^{'} \in T$, jonka potentiaali on pienin eli $\delta_j = 0$. Jos jokaisella solmulla $i \in S$ pienin potentiaali toteutuu eri solmun $j^{'} \in T$ kohdalla, yhtäsuuruus aligraafi sisältää täydellisen sovituksen ja ongelma ratkeaa heti lauseen 4.11 mukaisesti. Muussa tapauksessa rivillä 24 joukkoa Q kasvatetaan yhdellä solmulla, mutta rivien 10-13 alustuksia ei toteuteta. Tämä mahdollistaa eri arvoja muuttujille $p(j)$, jolloin toimintoon AUGMENT voidaan kuitenkin päästä, ja sovituksen kardinaliteetti kasvaa jälleen yhdellä.

Jos joukosta Q on poistettu jokainen solmu $i \in S$, mutta muuttuja aug on epätosi, ei täydellistä sovitusta ole syntynyt. Tämä tarkoittaa, että yhtäsuuruus aligraafilla ei ole täyttä sovitusta, joten lauseen 4.9 nojalla on olemassa sellainen joukon S osajoukko J, että $|\Gamma(J)| < |J|$. Algoritmi muodostaa rivillä 32 kyseisen osajoukon J ja myös joukon K, jonka alkioille $j' \in T$ on voimassa yhtälö $u_i + v_j = w_{ij}$, sillä $\delta_j = 0$. Yhtäsuuruus aligraafin määritelmästä seuraa, että joukko $K = \Gamma(J)$. Algoritmin rivin 24 mukaan jokaisen solmun $j' \in K$ vierussolmu $i \in S$, joka toteuttaa yhtälön $mate(j') = i$ kuuluu joukkoon J. Tämän lisäksi joukkoon J kuluvat kaikki avoimet solmut $i \in S$. Näin ollen joukot K ja J toteuttavat epäyhtälön $|K| < |J|$.

Tämän jälkeen algoritmin riveillä 34-38 muutetaan mahdollista solmupainotusta $(\vec{u}, \vec{v})$ lauseen 4.11 todistuksen mukaisesti, mikä vähentää summaa $\sum(u_i + v_i)$. Rivillä 33 muuttujalle δ alustetaan uusi arvo, mikä varmistaa, että vähintään yhdelle uudelle solmulle $j' \in T$ saavutetaan potentiaali $\delta_j = 0$. Tällöin joukon K koko kasvaa vähintään yhdellä. Tätä toimintoa toistetaan kunnes saavutetaan tilanne $|K| \geq |J|$, jolloin muodostunut aligraafi H sisältää täydellisen sovituksen. Tämä sovitus on samalla lauseen 4.11 mukaan maksimisovitus graafille G.

Edellä esitelty unkarilainen algoritmi olettaa käsiteltävän graafin olevan täydellinen kaksijakoinen graafi. Jos käsiteltävä graafi ei olisikaan täydellinen kaksijakoinen graafi, voitaisiin se kuitenkin täydentää, jolloin algoritmi toimisi kuten sen on tarkoitus. Koska algoritmi etsii maksimisovituksen, voitaisiin puuttuvat särmät lisätä minimaalisin painoin, jolloin niillä ei olisi vaikutusta lopulliseen maksimisovitukseen. Jos graafin solmujoukkojen koot eivät olisi samat vaan graafi olisi muotoa $(K_{n,m}, w)$ eli $n \neq m$, voitaisiin pienempi joukko laajentaa isomman kokoiseksi. Tällöin solmujen lisäksi on lisättävä tarvittavat särmät kuten edellä. Löytyneestä sovituksesta täytyy lopulta muistaa poistaa särmät, jotka ovat yhteydessä lisättyihin solmuihin.

5.2 Esimerkki algoritmin käytöstä

Tässä aliluvussa käydään läpi yksinkertainen, mutta kattava esimerkki unkarilaisen algoritmin toiminnasta käytännössä. Esimerkissä algoritmin avulla etsitään painotetulle täydelliselle kaksijakoiselle graafille $(K_{4,4}, w)$ täydellinen sovitus.

Graafin solmujoukko $V = S \cup T$ on siis jaettu kahteen yhtäsuureen osaan $S = \{1, 2, 3, 4\}$ ja $T = \{1', 2', 3', 4'\}$. Käsiteltävä graafi $(K_{4,4}, w)$ on kuvattu kuvassa 3.1, mutta yksinkertaisuutensa vuoksi esitetään graafin painofunktio kuitenkin matriisina eikä työlle ominaisesti graafina. Esimerkissä läpikäytävän graafin painofunktio on esitetty alla olevassa matriisissa A.

$$A = \begin{pmatrix} 2 & 5 & 6 & 1 \\ 3 & 4 & 3 & 2 \\ 1 & 4 & 3 & 3 \\ 2 & 2 & 7 & 2 \end{pmatrix} \begin{matrix} 6 \\ 4 \\ 4 \\ 7 \end{matrix}$$

$$\begin{matrix} 0 & 0 & 0 & 0 & v \backslash u \end{matrix}$$

Matriisia tulkitaan niin, että ensimmäisen rivin numerot ovat solmun $1 \in S$ ja joukon $T = \{1', 2', 3', 4'\}$ välisten särmien painoja, toinen rivi solmuun $2 \in S$ yhteydessä olevien särmien painoja ja niin edelleen. Sarakkeet ovat päinvastoin joukon $T = \{1', 2', 3', 4'\}$ solmuihin yhteydessä olevien särmien painoja. Esimerkiksi matriisin neljännen rivin kolmas numero 7 on särmän $w_{ij'} = w_{43}$ paino. Matriisin A oikealle reunalle on valittu solmujoukon $S = \{1, 2, 3, 4\}$ solmuille mahdolliset solmupainot $u_1 - u_4$ yhtälön (5.1) mukaisesti. Sama on toteutettu myös solmujoukolle $T = \{1', 2', 3', 4'\}$ solmupainoille $v_1 - v_4$ matriisin alapuolelle. Mahdollisten solmupainojen esitteäminen matriisin yhteydessä on järkevää, sillä ne tulevat muuttumaan algoritmin läpikäynnin aikana.

Unkarilaisen algoritmin rivit 2-13 alustavat parametreille alkuarvot. Sovituksessa ei siis ole alussa yhtään särmää ja esimerkin tapauksessa muuttuja $nrex$ saa arvon 4. Tämän lisäksi tehdään alla olevan taulukon mukaiset toimenpiteet.

$$\begin{matrix} 1' & 2' & 3' & 4' & j' \\ \infty & \infty & \infty & \infty & \delta_j \\ - & - & - & - & p(j) \end{matrix} \tag{5.2}$$

Jokaisen joukon $T = \{1', 2', 3', 4'\}$ solmun potentiaali valitaan siis äärettömäksi, joten huomataan ettei ole solmuja, missä tämä toteutuisi. Lisäksi muodostetaan joukko $Q = \{1, 2, 3, 4\}$, johon kuuluvat kaikki joukon $S = \{1, 2, 3, 4\}$ solmut, sillä jokainen niistä on vielä avoin. Tässä esimerkissä joukosta Q poistetaan aina pienin alkio selkeyden kannalta.

Ensimmäiseksi rivillä 15 joukosta Q poistetaan solmu $i = 1$ ja asetetaan sille totuusarvo

true. Koska solmu $i = 1$ on avoin, edetään riville 18. Tämä rivin yhtälöstä $u_1 + v_1 - w_{11} = 6 + 0 - 2 = 4 < \infty$ seuraa, että $\delta_1 = 4$ ja $p(1) = 1$. Koska potentiaali $\delta_1 \neq 0$, hypätään riville 29 ja käsitellän sama silmukka arvolla $j = 2$, jolloin saadaan potentiaaliksi $\delta_2 = 1$ ja sen toteuttavaksi solmuksi jälleen $p(2) = 1$. Kolmannella silmukan kierroksella muuttujien arvoiksi saadaan $\delta_3 = 0$ ja $p(3) = 1$, jolloin suoritetaan ensimmäisitä kertaa toiminto AUGMENT. Koska sovitus on tyhjä, toiminto yhdistää vain solmut $1 \in S$ ja $3' \in T$ sovitukseen. Toiminnosta AUGMENT seuraa siis, että sovituksessa on nyt yksi särmä $\{1, 3'\}$, parametri $aug = true$ ja käsittelemättömiä solmuja on $nrex = 4 - 1 = 3$. Silmukan viimeistä solmua $j' = 4$ ei täten käsitellä, joten taulukko (5.2) jää muotoon

$1'$	$2'$	$3'$	$4'$	j'
4	1	0	∞	δ_j
1	1	1	-	$p(j)$

(5.3)

Palataan takaisin unkarilaisen algoritmin riville 9, jolloin muodostetaan jälleen alkuperäinen taulukko (5.2) ja poistetaan joukosta Q solmu $i = 2$. Suoritetaan algoritmi aivan kuten solmun $i = 1$ tapauksessa, jolloin sovitukseen lisätään särmä $\{2, 2'\}$. Solmun $i = 2$ tapauksessa saadaan parametreille δ_j ja $p(j)$ arvot

$1'$	$2'$	$3'$	$4'$	j'
1	0	∞	∞	δ_j
2	2	-	-	$p(j)$

(5.4)

Aivan kuten solmun $2 \in S$ tapauksessa, palataan jälleen takaisin algoritmin riville 9, muodostetaan alkuperäinen taulukko (5.2) ja poistetaan solmu $i = 3$ joukosta Q. Rivin 16 silmukka arvolla $j' = 1$ tuottaa kyseiseksi potentiaaliksi arvon $\delta_1 = 3$, joten silmukan kierrosta ei edetä pidemmälle. Seuraavalla silmukan kierroksella arvolla $j' = 2$ saadaan potentiaaliksi $\delta_2 = 0$. Rivin 21 ehto ei kuitenkaan toteudu, sillä sovituksessa on jo särmä $\{2, 2'\}$, joten edetään riville 24 ja lisätään solmu $mate(2') = 2$ takaisin joukkoon Q. Toteutetaan silmukka loppuun, jolloin huomataan ettei solmun $i = 3$ tapauksessa päädytty suorittamaan toimintoa AUGMENT. Joukossa Q on kuitenkin vielä alkioita, joten rivin 31 ehto ei toteudu. Nyt on hyvä huomata, ettei taulukkoa (5.2) alusteta uudelleen, sillä parametri aug pysyi epätotena. Täten jatketaan algoritmia riviltä 15 alla olevan taulukon arvoja hyödyntäen.

$1'$	$2'$	$3'$	$4'$	j'
3	0	1	1	δ_j
3	3	3	3	$p(j)$

(5.5)

Käsiteltävä joukko $Q = \{2, 4\}$ koostuu nyt kahdesta alkiosta, joista valitaan pienempi eli

$i = 2$. Toteutetaan jälleen rivin 16 silmukka kyseiselle alkiolle. Arvolla $j' = 1$ saadaan potentiaaliksi arvo $\delta_1 = 1$, joka on pienempi kuin kyseisen potentiaalin arvo taulukossa (5.5), mikä muuttaa taulukon arvoja. Tiedetään, että sovituksessa on särmä $\{2, 2'\}$, joten arvolla $j' = 2$ ei edetä riviltä 17 eteenpäin. Arvolla $j' = 3$ saadaan potentiaaliksi $\delta_3 = 0$, joka on pienempi kuin taulukon 5.5 potentiaali $\delta_3 = 1$, mutta kyseinen solmu $j' = 3$ on jo yhteydessä solmuun $i = 1$. Näin ollen lisätään joukkoon Q solmu $mate(3') = 1$. Arvolla $j' = 4$ ei saada pienempää potentiaalia, joten siirrytään takaisin riville 15 taulukon päivitetyillä arvoilla.

$1'$	$2'$	$3'$	$4'$	j'
1	0	0	1	δ_j
2	3	2	3	$p(j)$

(5.6)

Joukossa $Q = \{1, 4\}$ on jälleen kaksi alkiota, jotka otetaan käsittelyyn numerojärjestyksessä. Kummankaan alkion tapauksessa ei yhdenkään solmun $j' \in T$ potentiaali pienene suorittaessa rivin 16 silmukkaa, jolloin ei myöskään toteuteta toimintoa AUGMENT. Tämä tarkoittaa, että joukko Q on tyhjä ja edetään ensimmäistä kertaa rivin 31 ehtolauseeseen taulukon 5.6 arvoilla.

Rivin 31 ehtolausekkeessa muodostetaan kaiken kaikkiaan 3 joukkoa J, K ja X. Näiden joukkojen muodostamisessa kannattaa olla erittäin tarkkana, sillä muuten algoritmi ei toimi kuten sen pitäisi. Esimerkissämme jokainen $i \in S$ alkio on otettu käsittelyyn viimeisen toiminnon AUGMENT jälkeen, jolloin jokaisella niistä on totuusarvo $m(i) = true$. Tämä tarkoittaa, että muodostetaan joukko $J = \{1, 2, 3, 4\}$. Taulukosta (5.6) nähdään, että solmujen $2', 3' \in T$ potentiaalit δ_j ovat nollia, joten muodostetaan joukko $K = \{2', 3'\}$. Potentiaaliksi δ puolestaa valitaan pienin lopuista potentiaaleista $\{1, 1\}$ eli $\delta = 1$. Rivien 34-41 toiminnot seuraavat suoraan teorian lauseen 4.11 todistuksesta. Muodostamme siis uuden yhtäsuuruus aligraafin, jolle pyrimme löytämään täydellisen sovituksen.

Koska jokainen solmu $i \in S$ kuuluu myös joukkoon J, vähennetään niiden solmupainoista u_i arvon $\delta = 1$ verran. Sama arvo lisätään joukon $K = \{2', 3'\}$ solmujen solmupainoihin v_j. Alla on esitetty matriisi A päivitetyillä solmupainoilla u_i ja v_j.

$$A = \begin{pmatrix} 2 & 5 & 6 & 1 \\ 3 & 4 & 3 & 2 \\ 1 & 4 & 3 & 3 \\ 2 & 2 & 7 & 2 \end{pmatrix} \begin{matrix} 5 \\ 3 \\ 3 \\ 6 \end{matrix}$$

$$\begin{matrix} 0 & 1 & 1 & 0 \end{matrix} \quad v \backslash u$$

Solmupainojen lisäksi jokaisen solmun $j' \in T$, jonka potentiaali δ_j oli suurempaa kuin nolla, potentiaalista vähennetään arvon $\delta = 1$ verran. Tämä tarkoittaa, että molemmat

potentiaalit δ_1 ja δ_4 saavat arvon nolla. Jolloin muodostetaan joukko $X = \{1', 4'\}$. Kumpikaan solmuista $1', 4' \in X$ ei vielä kuulu sovitukseen, joten edetään riville 48 ja valitaan solmuista pienempi eli $j' = 1'$.

Tämä on ensimmäinen kerta, kun toiminto AUGMENT muodostaa useamman kuin yhden uuden särmän sovitukseen. Aikaisemmilla kerroilla toiminto on vain yksinkertaisesti yhdistänyt käsiteltävät solmut $i \in S$ ja $j' \in T$. Toimintoon AUGMENT mennään arvolla $j' = 1'$ ja taulukosta (5.6) saadaan arvo $i = p(j) = p(1') = 2$. Näillä arvoilla saadaan rivin 58 mukaisesti seuraavat arvot: $i = 2$, $mate(1') = mate(j') = 2$, $next = mate(i) = mate(2) = 2'$ ja $mate(2) = j' = 1'$. Toiminto ei kuitenkaan lopu, sillä $next = 2' \neq 0$, joten suoritetaan rivi 58 uudestaan arvolla $j' = 2'$. Poimitaan taulukosta arvo $i = p(2') = 3$, jolloin saadaan seuraavat arvot: $mate(2') = 3$, $next = 0$ ja $mate(3) = 2'$. Kokonaisuudessaan toiminto AUGMENT muodostaa sovitukseen siis kaksi uutta särmää $\{2, 1'\}$ ja $\{3, 2'\}$ sekä poistaa särmän $\{2, 2'\}$.

Palataan takaisin unkarilaisen algoritmin alkuun riville 9 arvolla $nrex = 1$ ja sovituksen koostuessa särmistä $\{1, 3'\}$, $\{2, 1'\}$ ja $\{3, 2'\}$. Muodostetaan jälleen taulukko (5.2), mutta tällä kertaa joukko Q koostuu vain yhdestä alkiosta $\{4\}$. Toteutetaan rivin 16 silmukka alkiolle $i = 4$ tavalliseen tapaan. Muuttujan $j' = 3$ kohdalla potentiaaliksi muodostuu $\delta_3 = 0$, mutta solmu $3' \in T$ on jo sovituksessa, joten lisätään sen pari $1 \in S$ takaisin joukkoon Q. Silmukan loputtua saadaan seuraava taulukko.

$1'$	$2'$	$3'$	$4'$	j'
4	5	0	4	δ_j
4	**4**	**4**	**4**	$p(j)$

(5.7)

Käsitellään seuraavaksi joukkoon Q lisätty solmu $1 \in S$. Kun rivin 16 silmukka toteutetaan solmulle $i = 1$, muuttuu taulukon (5.7) arvot kuten alla on esitetty. Toimintoa AUGMENT ei kuitenkaan toteuteta eikä joukkoon $Q = \emptyset$ lisätä alkioita.

$1'$	$2'$	$3'$	$4'$	j'
3	1	0	4	δ_j
1	**1**	**4**	**4**	$p(j)$

(5.8)

Ratkaisussa edetään siis toisen kerran rivin 31 ehtolauseeseen hyödyntäen saadun taulukon (5.8) arvoja. Joukosta S on käsitelty vain alkioita $1, 4 \in S$ viimeisen kahden vaiheen aikana, jolloin vain niillä on totuusarvo $m(i) = true$. Täten muodostetaan joukko $J = \{1, 4\}$. Taulukosta (5.8) nähdään, että vain solmun $3' \in T$ potentiaali on nolla, joten muodostetaan joukko $K = \{3'\}$. Potentiaaliksi δ valitaan pienin lopuista potentiaaleista $\{3, 1, 4\}$ eli potentiaali $\delta = 1$.

Rivien 34-39 mukaisesti solmut $1, 4 \in S$ saavat uudet solmupainot $u_1 = 5 - 1 = 4$ ja $u_4 = 6 - 1 = 5$. Tämän lisäksi solmu $3' \in T$ saa uuden solmupainon $v_3 = 1 + 1 = 2$.

Rivien 40-43 toiminnoista seuraa, että joukkoon $X = \{2'\}$ kuuluu vain yksi alkio, sillä ainoastaan $\delta_2 = 0$. Tästä johtuen joukkoon Q lisätään alkio $mate(2') = 3$ rivien 45-46 mukaisesti. Toimintoa AUGMENT ei kuitenkaan toteuteta, jolloin alla olevan taulukon mukaiset potentiaalit jäävät voimaan.

$$\begin{array}{cccc|c} 1' & 2' & 3' & 4' & j' \\ 2 & 0 & 0 & 3 & \delta_j \\ \mathbf{1} & \mathbf{1} & \mathbf{4} & \mathbf{4} & p(j) \end{array} \tag{5.9}$$

Algoritmissa on täten edetty siihen vaiheeseen, että sovituksessa on parit $\{1, 3'\}$, $\{2, 1'\}$ ja $\{3, 2'\}$. Tämän lisäksi vain solmu $3 \in S$ on joukkossa $Q = \{3\}$. Päivitetyt mahdolliset solmupainot u_i ja v_j ovat nähtävissä alla olevasta matriisista A.

$$A = \begin{array}{c} \begin{pmatrix} 2 & 5 & 6 & 1 \\ 3 & 4 & 3 & 2 \\ 1 & 4 & 3 & 3 \\ 2 & 2 & 7 & 2 \end{pmatrix} \\ \begin{array}{cccc} 0 & 1 & 2 & 0 \end{array} \end{array} \begin{array}{c} 4 \\ 3 \\ 3 \\ 5 \\ v \backslash u \end{array}$$

Algoritmissa palataan jälleen riville 15 ja otetaan käsittelyyn solmu $3 \in Q$. Rivin 16 silmukan tapauksessa $j = 1$ ei sovitukseen tehdä muutoksia, sillä potentiaalin δ_1 arvo ei pienene. Tapauksessa $j = 2$ ei edetä riviä 17 pidemmällä, sillä $mate(3) = 2'$. Muuttujan arvolla $j = 3$ tapahtuu sama kuin arvolla $j = 1$, mutta tapauksessa $j = 4$ saadaan uudeksi potentiaaliksi arvo $\delta_4 = 0$. Koska solmu $4' \in T$ ei kuulu vielä sovitukseen, suoritetaan toiminto AUGMENT, joka on nähtävissä myös esimerkissä 5.1. Toiminnon ensimmäisellä kierroksella muodostetaan sovitukseen uusi särmä $\{3, 4'\}$, jolloin solmu $2' \in T$ otetaan käsittelyyn ja muodostetaan pari $\{1, 2'\}$. Nyt solmu $3' \in T$ jää avoimeksi ja se yhdistetään seuraavaksi pariksi $\{4, 3'\}$. Koska solmu $4 \in S$ ei ollut ennestään osa sovitusta, loppuu toiminto AUGMENT ja muuttujille saadaan arvot $aug = true$ ja $nrex = 0$.

Edellä mainittujen muuttujien arvojen perusteella algoritmi saadaan lopulta päätökseen sovituksen särmillä $\{1, 2'\}$, $\{2, 1'\}$, $\{3, 4'\}$ ja $\{4, 3'\}$. Matriisista A voidaan nyt laskea tämän sovituksen paino $w = 5+3+3+7 = 18$, joka vastaa myös mahdollisen solmupainotuksen summaa $\sum_{i,j}(u_i+v_j) = (4+1)+(3+0)+(3+0)+(5+2) = 18$. Algoritmi siis löysi täydellisen sovituksen muodostetulle yhtäsuuruus aligraafille, mikä on lauseen 4.11 nojalla myös maksimisovitus alkuperäiselle graafille.

6 YHTEENVETO

Tässä kandidaatintyössä perehdyttiin unkarilaiseen algoritmiin graafiteorian näkökulmasta. Työn alussa, luvussa 2, käsiteltiin sovitusongelma, jonka unkarilainen algoritmi pyrkii ratkaisemaan. Luvussa 3, esiteltiin tarvittavat graafiteorian perustiedot, jotka varmistavat lukijalle pohjan ymmärtää myös työssä esiteltävät tärkeät lauseet ja todistukset. Lauseiden ja todistusten lisäksi työssä esiteltiin muutamia graafeja kuvina, joiden tarkoitus oli tuottaa mahdollisimman hyvä yleiskuvan työssä käsiteltävästä aiheesta.

Työn luvun 4 teoria käsitteli graafien eri sovituksien yhteyksiä useasta eri näkökulmasta. Luvussa esiteltiin myös harvemmin käytössä oleva graafien solmupainotus, joka kuitenkin tässä työssä on tärkeä osa unkarilaista algoritmia. Tämän lisäksi luvussa todistettiin Königin lause, jonka mukaan maksimaalisen sovituksen särmien määrä vastaa minimaalisen solmupeitteen solmujen määrää kaksijakoisessa graafissa. Tulos mahdollisti unkarilaiselle algoritmille tärkeän ominaisuuden todistamisen, jonka mukaan jokaisen aligraafin täydellinen sovitus on maksimisovitus varsinaiselle kaksijakoiselle graafille. Luvun teoria mahdollisti myös sovitusongelman esittämisen graafiteoriaa hyödyntäen.

Työn viimeisessä luvussa 5 esiteltiin unkarilainen algoritmi pseudokoodina, jota hyödynnettiin yksinkertaisen esimerkin ratkaisemiseksi. Esimerkki käytiin läpi askel kerrallaan, sillä pseudokoodi sisälsi useita kerroksia ja silmukoita, jotka hankaloittavat koodin tulkintaa. Algoritmista kuitenkin huomattiin, että se löysi nopeasti maksimisovituksen tutkittavalle kaksijakoiselle graafille $(K_{n,n}, w)$. Tarkalleen ottaen algoritmin tehokkuudeksi on todistettu $O(n^3)$ [4], joka on huomattavasti tehokkaampi ratkaisu kuin jokaisen mahdollisen sovituksen läpikäynti, minkä tehokkuus on $O(n!)$.

LÄHTEET

[1] M. Cerf. Multiple Object Trajectography Using Particle Swarm Optimization Combined to Hungarian Method (2018). URL: https://arxiv.org/ftp/arxiv/papers/1802/1802.06201.pdf. Viitattu: 4.7.2019.

[2] Y. Z. Cheng, P. C. Zhang ja B. Q. Cao. Weapon Target Assignment Problem Solving Based on Hungarian Algorithm. English. *Applied Mechanics and Materials* 713-715 (2015), 2041.

[3] K. Date ja R. Nagi. GPU-accelerated Hungarian algorithms for the Linear Assignment Problem. English. *Parallel Computing* 57 (2016), 52–72.

[4] D. Jungnickel. *Graphs, Networks and Algorithms*. English. Vol. 5. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 9783540727798;3540727795;

[5] H. W. Kuhn. The Hungarian method for the assignment problem. English. *Naval Research Logistics (NRL)* 52.1 (2005), 7–21.

[6] C. Luis Bassa ja A. M. Gil-Lafuente. The Hungarian Algorithm for Specific Customer Needs. English. 2012. painos. Vol. 286. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 363–379. ISBN: 1434-9922.

[7] R. Rizzi. A short proof of Konig's matching theorem. English. *Journal of Graph Theory* 33.3 (2000), 138–139.