

582456 Approksimointialgoritmit

kevät 2010
Jyrki Kivinen

Kurssin asema opetuksessa

- laajuus 8 op
- kelpaa syventäviin opintoihin algoritmien ja koneoppimisen (ja vanhalla algoritmien) erikoistumislinjalla.
- esitietoina *Algoritmien suunnittelu ja analyysi* ja *Diskreetti optimointi* tai vastaavat tiedot
- luennot ti 8–10, pe 12–14; harjoitukset ma 14–16

Kurssin suorittaminen, arvostelu, materiaalit

Maksimipistemäärä 60 pistettä:

- kaksi kurssikoetta $20 + 20 = 40$ pistettä
- laskuharjoitukset 10 pistettä
- artikkelitiivistelmä ja esitelmä 10 pistettä

Läpikäsyraja noin 30 pistettä, arvosanan 5/5 raja noin 50 pistettä.

Kurssi perustuu kirjaan

Vijay V. Vazirani: *Approximation Algorithms*.

Sisältö

1. Johdanto
peruskäsitteitä
2. Perustekniikat
kokoelma edustavia esimerkkiongelmia
3. Lineaarinen ohjelmointi
pyöristäminen ja primaali-duaali-menetelmä
4. PCP-teoreeman sovellukset
approksimoinnin mahdottomuustuloksia

1. Johdanto

Monista tärkeistä ongelmista tiedetään, että niille on olemassa polynomisessa ajassa toimiva ratkaisualgoritmi, jos ja vain jos $P = NP$.

Koska käytännössä ei ole realistista ruveta todistamaan $P = NP$, täytyy tinkiä jostain:

ei-polynomiset algoritmit: peruuttava etsintä, branch-and-bound
heuristiikat, joiden tuottaman ratkaisun laadusta ei ole takuita
approksimointialgoritmit, joiden tuottama ratkaisu ei ole optimaalinen, mutta todistettavasti jossain mielessä lähellä optimaalista.

Esimerkki: solmupeiteongelma (vertex cover)

Olkoon annettuna verkko $G = (V, E)$ ja solmujen kustannusfunktio $c: V \rightarrow \mathbb{Q}_+$.

Verkon **solmupeite** on mikä tahansa joukko $V' \subseteq V$, joka sisältää ainakin toisen päätepisteen kaikista kaarista $e \in E$. Joukkopeitteen **kustannus** on $\sum_{v \in V'} c(v)$.

Jos $c(v) = 1$ kaikilla $v \in V$, puhumme **lukumääräisestä** solmupeiteongelmasta.

Tunnetusti päätösongelma

Syöte: verkko G , kustannusfunktio c , rationaaliluku p

Kysymys: onko verkolla G solmupeite, jonka kustannus on korkeintaan p

on NP-täydellinen.

Tarkastellaan nyt seuraavaa algoritmia lukumääräisesti pienen(?) solmupeitteen löytämiseksi:

1. Etsi jokin verkon G maksimaalinen pariutus $M \subseteq E$.
2. Palauta V' , joka sisältää pariutuksen M kaikkien kaarien kummankin päätepisteen.

Muistutukseksi:

- **Pariutus** on joukko kaaria, jossa millään kahdella kaarella ei ole yhteistä päätepistettä.
- Pariutus M on **maksimaalinen**, jos mikään $M' \supset M$ ei ole pariutus.

Jokin maksimaalinen pariutus on helppo löytää käymällä kaaret läpi mielivaltaisessa järjestyksessä ja ottamalla mukaan kaari, jos sen päätepisteitä ei vielä ole käytetty.

Lause 1.1: Edellä esitetty algoritmi tuottaa solmupeitteen, jonka koko on korkeintaan 2 kertaa pienimmän solmupeitteen koko.

Todistus: Joukko V' on solmupeite: jos jonkin kaaren kumpikaan päätepiste ei olisi joukossa V' , niin M ei olisi maksimaalinen pariutus.

Olkoon P mielivaltainen solmupeite. Erityisesti P sisältää ainakin toisen päätepisteen jokaisesta pariutuksen M kaaresta. Koska M on pariutus, nämä ovat erillisiä, joten $|P| \geq |M|$. Siis $|V'| = 2|M| \leq 2|P|$. \square

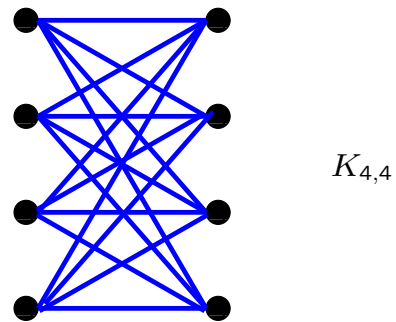
Keskeinen havainto tässä oli alaraja solmupeitteen koolle: jos M on pariutus, niin minkä tahansa solmupeitteen (ja erityisesti siis pienimmän) koko on ainakin $|M|$.

Edellinen tulos approksimoinnista vakiokertoimen 2 tarkkuudella ei ehkä tunnu kovin vahvalta. Voisiko kertoimen saada pienemmäksi kuin 2

1. käyttämällä samaa algoritmia, mutta tekemällä analyysin tarkemmin
2. laatimalla uuden algoritmin, mutta perustamalla analyysin samaan alarajaan (solmupeitteen koko on vähintään maksimaalisen pariutuksen koko)
3. käyttämällä jotain kokonaan muuta periaatetta?

Kolmas vaihtoehto on merkittävä avoin ongelma. Sen sijaan kaksi ensimmäistä vaihtoehtoa eivät tuo parannusta, kuten seuraavaksi näemme.

Esimerkki 1.2: Olkoon $K_{n,n}$ täydellinen $2n$ -solmuinen kaksijakoinen (bipartite) verkko:



Edelläesitetty algoritmi valitsee peitteeseen kaikki $2n$ solmua, vaikka riittäisi valita joko vasemmanpuoleiset tai oikeanpuoleiset n solmua. \square

Tällaisten ns. tiukkojen esimerkkien tutkiminen on usein hyvin valaisevaa algoritmien toiminnan ja analyysin ymmärtämiseksi.

Esimerkki 1.3: Olkoon K_n täydellinen n solmun verkko. Kun n on pariton, niin maksimaalisessa pariutuksessa on $(n - 1)/2$ kaarta ja pienimmässä solmupeitteessä on $n - 1$ solmua.

Siis jos P on pienin solmupeite ja M maksimaalinen pariutus, niin

$$|P| \geq |M| \geq \frac{1}{2} |P|,$$

ja kumpikin epäyhtälö voi olla tiukka. Siis pelkästään tarkastelemalla maksimaalisten pariutusten kokoja ei päästä parempaan approksimointitarkkuuteen kuin 2. \square

Palataan vielä kysymykseen, onko approksimointi vakiokertoimen 2 tarkkuudella tyydyttävää. Tämä riippuu tietysti sovelluksesta. Edellä esitetyn perusteella paremman approksimointitakuun saavuttaminen olisi merkittävä avoin ongelma. Toisaalta käytännössä voi olla, että algoritmin pahimpia tapauksia ("tiukat esimerkit") ei juuri esiinny. Jos taas niitä esiintyy, niin algoritmi ja sen analyysi ovat luultavasti hyödyllisiä ongelman ymmärtämiseksi.

Luokan NP määritelmä

Luokka NP määritellään perinteisesti epädeterminististen Turingin koneiden avulla. Seuraava ekvivalentti määritelmä on usein kätevämpi.

Kieli L kuuluu luokkaan NP, jos ja vain jos on olemassa sellaiset polynomi p ja polynomisessa ajassa toimiva deterministinen Turingin kone M (tarkastaja, verifier), että

- jos $x \in L$, niin on olemassa ainakin yksi sellainen y , että $|y| \leq p(|x|)$ ja M hyväksyy syötteen $\langle x, y \rangle$
- jos $x \notin L$, niin ei ole olemassa sellaista y , että $|y| \leq p(|x|)$ ja M hyväksyy syötteen $\langle x, y \rangle$

Tässä $\langle x, y \rangle$ on jokin tapa koodata merkkijonopari (x, y) yhdeksi merkkijonoksi.

Tapauksen " $x \in L$ " merkkijonoa y sanotaan sertifikaatiksi, todistajaksi, todistukseksi jne.

Esimerkki 1.4: Tarkastellaan NP-täydellistä solmupeiteongelmaa

$$VC = \{ \langle G, k \rangle \mid \text{verkolla } G \text{ on } k\text{-alkiainen solmupeite} \}.$$

Muodostetaan M , joka hyväksyy merkkijonon $\langle x, y \rangle$, jos ja vain jos x on muotoa $\langle G, k \rangle$ ja y on sellainen joukko verkon G solmuja, että sen koko on k ja se muodostaa solmupeitteen. \square

Kuten yllä, NP-täydellisten ongelmien ratkaisemisessa vaikea osa on yleensä sertifiikaatin löytäminen.

Jos $L \in \text{NP}$ edellisen määritelmän mukaan, niin on helppo konstruoida polynomisessa ajassa toimiva epädeterministinen Turingin kone N kielelle L . Kone N arvaa ensin epädeterministisesti merkkijonon y ja simuloi sitten määritelmän mukaista konetta M .

Kääntäen oletetaan, että polynomisessa ajassa toimiva epädeterministinen Turingin kone N tunnistaa kielen L . Muodostetaan kone M , joka syötteellä $\langle x, y \rangle$ toimii seuraavasti:

- Simuloi koneen N laskentaa syötteellä x .
- Kun kone N tulee tilanteeseen, jossa on useita mahdollisia seuraajatilanteita, lue merkkijonosta y seuraava merkki ja päätä sen perusteella, mikä seuraajista valitaan.

Selvästi M täyttää edellisen määritelmän vaatimukset.

NP-optimointiongelmat

NP-optimointiongelma Π koostuu seuraavista komponenteista:

1. Kelvollisten tapausten (valid instances) joukko D_Π on jokin polynomisessa ajassa tunnistettavissa oleva kieli.
2. Jokaiselle kelvolliselle tapaukselle $I \in D_\Pi$ sen käypien (feasible) ratkaisujen joukko $S_\Pi(I)$, jolle pätee
 - $S_\Pi(I) \neq \emptyset$
 - ratkaisun $s \in S_\Pi(I)$ koko $|s|$ on polynominen tapauksen koon $|I|$ suhteen
 - on olemassa polynomisessa ajassa toimiva algoritmi, joka annetuilla s ja I ratkaisee, päteekö $s \in S_\Pi(I)$.

3. polynomisessa ajassa laskettava **kustannusfunktio** obj_Π , joka liittää ei-negatiivisen rationaaliluvun $\text{obj}_\Pi(I, s)$ jokaiseen pariin (I, s) , missä $s \in S_\Pi(I)$
4. tieto siitä, onko Π **minimointi-** vai **maksimointiongelma**.

Huomioita:

- Tarkastelemme vain ongelmia, joissa esiintyvät luvut ovat rationaalisia.
- Ongelman tapaukset jne. oletetaan koodatuiksi merkkijonoiksi jollain järkevällä tavalla. Merkinnät $|I|$ jne. tarkoittavat tällaisen koodin pituutta. Ainoa huomionarvoinen seikka on, että kokonaisluvun k koodin pituudeksi oletetaan $O(\log k)$, ts. koodaus **ei** ole unaarinen.

Esimerkki 1.5: Solmupeiteongelmassa

- Kelvollinen tapaus koostuu verkosta $G = (V, E)$ ja kustannusfunktioista $c: V \rightarrow \mathbb{Q}_+$.
- Tapauksen $I = (G, c)$ käypiä ratkaisuja ovat verkon G solmupeitteet.
- Kustannusfunktio on

$$\text{obj}_{\Pi}(I, s) = \sum_{v \in s} c(v).$$

- Kysymyksessä on minimointiongelma.

□

Minimointiongelman tapauksessa

$$\text{OPT}_{\Pi}(I) = \min \{ \text{obj}_{\Pi}(I, s) \mid s \in S_{\Pi}(I) \}$$

on paras saavutettavissa oleva kustannusfunktion arvo tapaukselle $I \in D_{\Pi}$.

Optimaalinen ratkaisu on sellainen $s \in S_{\Pi}(I)$, että $\text{obj}_{\Pi}(I, s) = \text{OPT}_{\Pi}(I)$.

Maksimointiongelmalta tietysti

$$\text{OPT}_{\Pi}(I) = \max \{ \text{obj}_{\Pi}(I, s) \mid s \in S_{\Pi}(I) \}.$$

Jos selvyys ei kärsi, merkitsemme usein vain "OPT" jne.

Olkoon Π minimointiongelma ja δ jokin funktio $\mathbb{Z}_+ \rightarrow \mathbb{Q}_+$.

Jos polynomisessa ajassa toimiva algoritmi A syötteellä $I \in D_\Pi$ tuottaa käyvän ratkaisun $A(I) \in S_\Pi(I)$, jolla

$$\text{obj}_\Pi(I, A(I)) \leq \delta(|I|)\text{OPT}(I),$$

niin A on δ -*approksimointialgoritmi* ongelmalle Π .

Toisinaan sallimme myös satunnaisalgoritmit. Tällöin ehto tulee muotoon

$$\Pr [\text{obj}_\Pi(I, A(I)) \leq \delta(|I|)\text{OPT}(I)] \geq \frac{1}{2},$$

missä $\Pr[...]$ on todennäköisyys yli algoritmin tekemien satunnaisvalintojen.

Maksimointiongelman tapauksessa δ -approksimointialgoritmin ehto tulee luonnollisesti muotoon

$$\text{obj}_\Pi(I, A(I)) \geq \delta(|I|)\text{OPT}(I).$$

Siis kummassakin tapauksessa $\delta = 1$ vastaa tarkkaa ratkaisua.

(Kirjallisuudessa toisinaan approksimointitarkkuus määritellään suhteellisen virheen kautta, mikä antaa eri numeroarvot kuin tämä määritelmä.)

Funktion δ argumenttina on usein jokin muu ongelman kokoa kuvaava suure kuin sen koko, esim. joukkopeiteongelman tapauksessa peitettävän perusjoukon alkioiden lukumäärä.

Polynomiset palautukset

NP-täydelliset ongelmat ovat kaikki määritelmän mukaan polynomisesti palautettavissa toisiinsa ja siis jossain mielessä ekvivalentteja. Kuitenkin niiden approksimoituvuudessa on suuria eroja, kuten jatkossa näemme. Polynomiselta palautukselta nimittäin vaaditaan vain, että se kuvaa "kyllä"-tapaukset "kyllä"-tapauksiksi ja "ei"-tapaukset "ei"-tapauksiksi. Mitään vaatimuksia ei aseteta sen suhteen, kuinka paljon "ei"-tapaukset voivat olla pielessä.

Approksimointisuhteen säilyttävä palautus minimointiongelma Π_1 minimointiongelmaan Π_2 koostuu kahdesta funktiosta. Ensimmäinen, f , muuntaa tapaukset; siis f on funktio $D_{\Pi_1} \rightarrow D_{\Pi_2}$. Toinen funktio, g , muuntaa ratkaisut; siis jos $I_2 = f(I_1)$ ja $s_2 \in S_{\Pi_2}(I_2)$, niin $g(I_1, s_2) \in S_{\Pi_1}(I_1)$. Lisäksi kun $I_2 = f(I_1)$ ja $s_1 = g(I_1, s_2)$, vaaditaan

- $\text{OPT}_{\Pi_2}(I_2) \leq \text{OPT}_{\Pi_1}(I_1)$
- $\text{obj}_{\Pi_1}(I_1, s_1) \leq \text{obj}_{\Pi_2}(I_2, s_2)$.

Helposti nähdään, että tällainen palautus yhdistettynä δ -approksimointialgoritmiin ongelmalle Π_2 antaa δ -approksimointialgoritmin ongelmalle Π_1 .

Itsepalautuvuus (Self-reducibility)

Esim. (lukumääräinen) solmupeiteongelma voidaan esittää kolmena eri versiona:

Päätösongelma:

Annettu: verkko G , luonnollinen luku k

Kysymys: onko verkolla G solmupeite, jonka koko on k

Optimointiongelma:

Annettu: verkko G

Kysymys: mikä on verkon G pienimmän solmupeitteen koko

Etsintäongelma:

Annettu: verkko G

Kysymys: mikä on verkon G pienin solmupeite

Selvästi etsintäversion ratkaisualgoritmi antaa ratkaisualgoritmin optimointiversiolle, ja optimointiversion ratkaisualgoritmi päätösversiolle.

Itse asiassa myös käänteinen pätee.

Oletetaan ensin, että algoritmi $VC(G, k)$ palauttaa polynomisessa ajassa ratkaisun päätösongelmaan. Käyttämällä algoritmia VC alirutiinina voidaan optimointiongelma ratkaista esim. binäärihaulla polynomisessa ajassa.

Oletetaan nyt, että $\text{VC-opt}(G)$ palauttaa verkon G pienimmän solmupeitteen **koon** polynomisessa ajassa. Pienin solmupeite voidaan löytää seuraavalla algoritmilla:

- Valitse mielivaltainen solmu $v \in V$. Muodosta G' poistamalla verkosta G solmu v ja siihen liittyvät kaaret.
- Jos $\text{VC-opt}(G') = \text{VC-opt}(G) - 1$, niin muodosta rekursiivisesti verkon G' pienin solmupeite S' ja palauta $S' \cup \{v\}$.
- Muuten muodosta G'' poistamalla verkosta G solmu v , kaikki sen naapurit ja näihin solmuihin liittyvät kaaret. Muodosta rekursiivisesti verkon G'' pienin solmupeite S'' ja palauta $S'' \cup N(v)$, missä $N(v)$ on solmun v naapurien joukko.

Algoritmin haarat vastaavat kahta tapausta:

- Ehto $VC\text{-opt}(G') = VC\text{-opt}(G) - 1$ pätee, jos ja vain jos v kuuluu johonkin verkon G pienimpään solmupeitteeseen S . Poistamalla v joukosta S siitä saadaan verkon G' pienin solmupeite.
- Jos v ei kuulu mihinkään pienimpään solmupeitteeseen, niin jokainen pienin solmupeite sisältää kaikki solmun v naapurit. Yhdistämällä nämä verkon G'' pienimpään solmupeitteeseen saadaan verkon G pienin solmupeite.

Siis edellinen algoritmi ratkaisee etsintäongelman polynomisessa ajassa, jos optimointiongelmalle on polynominen ratkaisualgoritmi.

Edellisen yleistämiseksi määrittelemme nyt NP-optimointiongelman itsepalautuvuuden.

Tarkastelemissamme ongelmissa tapauksen I ratkaisu s voidaan esittää joukkona **atomeita**. Esim. solmupeiteongelman tapauksessa atomeja ovat väittämät, että tietty solmu kuuluu tai ei kuulu tarkasteltavaan solmupeitteeseen. Atomit voidaan tässä tapauksessa (kuten yleensäkin jatkossa vastaantulevissa tapauksissa) esittää $O(\log |I|)$ bitillä. Tällöin sanomme, että ongelman **rakeisuus** (granularity) on $O(\log |I|)$.

Olkoon ongelman Π rakeisuus $O(\log |I|)$. Ongelma Π on **itsepalautuva**, jos on olemassa polynomisessa ajassa toimiva algoritmi A ja polynomisessa ajassa laskettavat funktiot f ja g , jotka toteuttavat seuraavat ehdot:

1. Algoritmi A saa syötteenä tapauksen I ja atomin α . Tulosteena A tuottaa tapauksen I_α , jolle $|I_\alpha| < |I|$.
2. Olkoon $S(I | \alpha)$ niiden käypien ratkaisujen $s \in S_\Pi(I)$ joukko, jotka ovat yhteensopivia atomin α kanssa. Funktio $s \mapsto f(I, \alpha, s)$ on bijektio joukolta $S(I_\alpha)$ joukkoon $S(I | \alpha)$.

3. Olkoot s'_1 ja s'_2 kaksi käypää ratkaisua ongelmaan I_α , ja $f(I, \alpha, s'_1) = s_1$ ja $f(I, \alpha, s'_2) = s_2$. Jos $\text{obj}_\Pi(I_\alpha, s'_1) \leq \text{obj}_\Pi(I_\alpha, s'_2)$, niin $\text{obj}_\Pi(I, s_1) \leq \text{obj}_\Pi(I, s_2)$.
4. Jos ongelman I_α optimaalinen kustannus on r , niin ongelman $S(I | \alpha)$ optimaalinen kustannus on $g(I, \alpha, r)$.

Lause 1.6: Olkoon Π itsepalautuva NP-optimointiongelma. Jos käytettävissä on polynomisessa ajassa toimiva algoritmi \mathcal{B} ongelman päätösversiolle, niin voimme muodostaa polynomisessa ajassa toimivan algoritmin etsintäversiolle.

Todistus: Kuten edellä todettiin, voimme ensin binäärihakua soveltamalla muodostaa algoritmin \mathcal{B}' ongelman optimointiversiolle.

Olkoot nyt A , f ja g kuten itsepalautuvuuden määritelmässä. Olkoon I ongelman tapaus. Etsimme ensin atomin α , jolle

$$g(I, \alpha, \text{OPT}(I_\alpha)) = \text{OPT}(I).$$

Tämä voidaan tehdä polynomisessa ajassa, kun käytettävissä on algoritmi \mathcal{B}' ja atomien koko on logaritminen. Tällainen α on yhteensopiva tapauksen I jonkin optimaalisen ratkaisun kanssa.

Olkoon nyt $I_\alpha = A(I, \alpha)$. Lasketaan seuraavaksi ongelman I_α ratkaisu rekursiivisesti; olkoon tämä ratkaisu s' . Koska $|I_\alpha| < |I|$, tämä rekursio päättyy polynomisessa ajassa. Nyt $f(I, \alpha, s')$ on optimaalinen ratkaisu alkuperäiseen ongelmaan I . \square

Ei-sertifikaatit ja min-max-relaatiot

Vaativuusluokka co-NP koostuu niistä kielistä, joiden komplementti kuuluu luokkaan NP. Selvästi

$$P \subseteq NP \cap \text{co-NP}.$$

On merkittävä avoin ongelma, onko sisältyvyys aito vai päteekö

$$P \stackrel{?}{=} NP \cap \text{co-NP}.$$

Toinen merkittävä avoin ongelma on, päteekö

$$NP \stackrel{?}{=} \text{co-NP}.$$

Jos L kuuluu luokkaan co-NP, sanomme, että sillä on polynomisen kokoiset ei-sertifikaatit.

(Luokan NP kielillä on määritelmän mukaan polynomisen kokoiset (kyllä-)sertifikaatit. Ei-sertifikaatti osoittaa, että $x \notin L$ eli on sama asia kuin kyllä-sertifikaatti kielelle \bar{L} .)

Tarkastellaan nyt seuraavia päätösongelmia:

- Onko verkon G lukumääräisesti pienimmän solmupeitteen koko korkeintaan k solmua?
- Onko verkon G lukumääräisesti suurimman pariutuksen koko ainakin l kaarta?

Ongelmilla on selvästi polynomisen kokoiset sertifiikaatit (yksinkertaisesti verkon pienin solmupeite ja maksimipariutus) eli ne kuuluvat luokkaan NP.

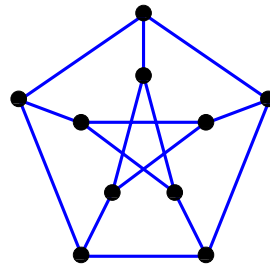
Königin-Egerváryn lauseen mukaan

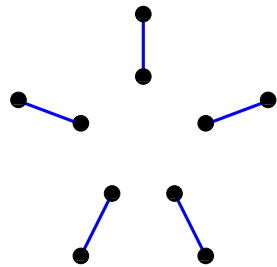
Jos verkko on kaksijakoinen, niin sen maksimipariutuksen koko on sama kuin pienimmän solmupeitteen koko.

Siis kaksijakoisiin verkkoihin rajoitettuina yo. ongelmilla on myös ei-sertifiikaatit eli ne kuuluvat luokkaan $NP \cap co-NP$ (ja itse asiassa luokkaan P). Solmupeiteongelman ei-sertifiikaatti on kokoa $k + 1$ oleva pariutus, ja paritusongelman ei-sertifiikaatti kokoa $l - 1$ oleva solmupeite.

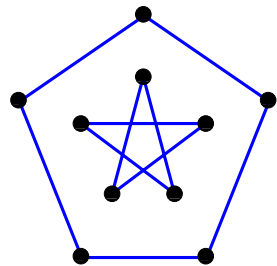
Maksimipariutusongelma kuuluu luokkaan P myös yleisillä verkoilla (Edmonds 1965). Sen sijaan yleisen solmupeiteongelman ei tiedetä kuuluvan luokkaan co-NP.

Jos Königin-Egerváryn lause pätsi myös yleisillä verkoilla, niin polynominen algoritmi maksimipariutukselle antaisi tietysti polynomisen algoritmin solmupeitteelle. Allaoleva Petersenin verkko kuitenkin osoittaa, että pienin solmupeite voi olla suurempi kuin maksimipariutus.





Maksimipariutuksessa on viisi kaarta.



Verkossa on kaksi erillistä viiden mittaista sykliä, joiden peittämiseen tarvitaan ainakin $3 + 3$ solmua.

Siis maksimipariutuksen ei-sertifikaatit eivät ole solmupeitteitä.

Verkon $G = (V, E)$ **pariton joukkopeite** C koostuu sellaisista

- joukosta parittoman kokoisia solmujoukkoja V_1, \dots, V_k ja
- joukosta solmuja v_1, \dots, v_l ,

että jokaisella kaarella

- ainakin toinen päätepiste on jokin solmuista v_i tai
- kumpikin päätepiste kuuluu samaan joukkoon V_j .

Peitteen **paino** on

$$w(C) = l + \sum_{i=1}^k (|S_i| - 1)/2.$$

Voidaan osoittaa, että maksimipariutuksen koko on sama kuin pienin parittoman joukkopeitteen paino. Parittomista joukkopeitteistä saadaan siis maksimipariutusongelman ei-sertifikaatit.

Vaikka maksimipariutus voi olla pienempi kuin pienin solmupeite, ero ei voi olla mielivaltaisen suuri. Olemme edellä todenneet, että jos M on maksimipariutus ja U pienin solmupeite, niin

$$|M| \leq |U| \leq 2|M|.$$

Olkoon nyt $k < |U|/2$. Verkon maksimiparituksessa M on ainakin $|U|/2 > k$ kaarta. Tästä pariutuksesta seuraa, että solmupeitteen koko on suurempi kuin k . Siis M on ei-sertifikaatti, joka osoittaa, että (G, k) on solmupeiteongelman ei-tapaus.

Jos minimointiongelmallä Π on ei-sertifikaatti kaikille tapauksille (I, B) , joilla $B < \text{OPT}(I)/\alpha$, sanomme, että ongelmalla on **α -approksimatiiviset** ei-sertifikaatit.

Jos ongelmalla on α -approksimointialgoritmi, sen tulosteet antavat polynomisessa ajassa laskettavissa olevat α -approksimatiiviset ei-sertifikaatit.

Sertifikaatti verifioidaan toteamalla, että sen kustannus on suurempi kuin αB ja se todella on mainitun approksimointialgoritmin tuloste, jolloin tiedämme optimikustannuksen olevan suurempi kuin B .

2. Perustekniikat

Esimerkkejä

- algoritmitekniikoista
- analyysitekniikoista
- millaisia tuloksia voidaan saavuttaa.

Joukkopeite (Set Cover)

Tapaus:

- perusjoukko U
- kokoelma $\mathcal{S} = \{S_1, \dots, S_k\}$ osajoukkoja $S_i \subseteq U$
- kustannusfunktio $c: \mathcal{S} \rightarrow \mathbb{Q}_+$

Käyvät ratkaisut: joukkopeitteet eli kokoelmat $\mathcal{R} \subseteq \mathcal{S}$, joilla
 $\cup_{S \in \mathcal{R}} S = U$

Kustannus: joukkopeitteen \mathcal{R} kustannus on $\sum_{S \in \mathcal{R}} c(S)$
(minimointitehtävä)

Perusratkaisu tässä, kuten monessa muussakin ongelmassa, on **ahne** (greedy) algoritmi. Se saavuttaa approksimointisuhteen $O(\log n)$, missä $n = |U|$.

Vaihtoehtoinen ratkaisu perustuu **kerrostamiseen** (layering).

Ahne joukkopeitealgoritmi

1. $C \leftarrow \emptyset$
2. Toista kunnes $C = U$:
 - (a) Laske jokaisen osajoukon $S \in \mathcal{S}$ yksikkökustannus

$$u(S) = \frac{c(S)}{|S - C|}.$$

- (b) Olkoon S_* jokin osajoukko, jonka yksikkökustannus on pienin.
- (c) Valitse S_* mukaan joukkopeitteeseen.
- (d) Kaikilla $e \in S_* - C$ aseta $\text{price}(e) = u(S_*)$.

3. Palauta peitteeseen valitut osajoukot.

(Arvoja $\text{price}(e)$ tarvitaan vain analyysissä, ne eivät vaikuta algoritmin suoritukseen.)

Olkoon e_1, \dots, e_n alkioiden numerointi siinä järjestyksessä, kuin ne tulevat peitetyiksi; tasatilanteiden järjestyksellä ei ole väliä.

Lemma 2.1: Kaikilla $1 \leq k \leq n$ pätee

$$\text{price}(e_k) \leq \frac{\text{OPT}}{n - k + 1}.$$

Todistus: Millä tahansa iteraatiokierroksella vielä peittämättömät alkiot voitaisiin peittää kokonaiskustannuksella korkeintaan OPT käyttämällä optimipeitteen joukkoja. Voimme arvioida

$$\frac{\text{OPT}}{|U - C|} = \sum_{S \in \mathcal{R}} \frac{c(S)}{|U - C|} = \sum_{S \in \mathcal{R}} \frac{|S - C|}{|U - C|} u(S) \geq \sum_{S \in \mathcal{R}} \frac{|S - C|}{\sum_{S' \in \mathcal{R}} |S' - C|} u(S),$$

missä \mathcal{R} on optimipeite. Koska siis optimipeitteen joukkojen yksikkökustannusten painotettu keskiarvo on korkeintaan $\frac{\text{OPT}}{|U - C|}$, niin parhaan yksittäisen joukon yksikkökustannus on korkeintaan $\frac{\text{OPT}}{|U - C|}$.

Juuri ennen kuin e_k tuli peitetyksi, joukossa C oli korkeintaan $k - 1$ alkiota. Siis

$$\text{price}(e_k) \leq \frac{\text{OPT}}{|U - C|} \leq \frac{\text{OPT}}{n - k + 1}. \quad \square$$

Lause 2.2: Ahne joukkopeitealgoritmi on H_n -approksimointialgoritmi, missä $H_n = 1 + 1/2 + 1/3 + \dots + 1/n \sim \ln n$.

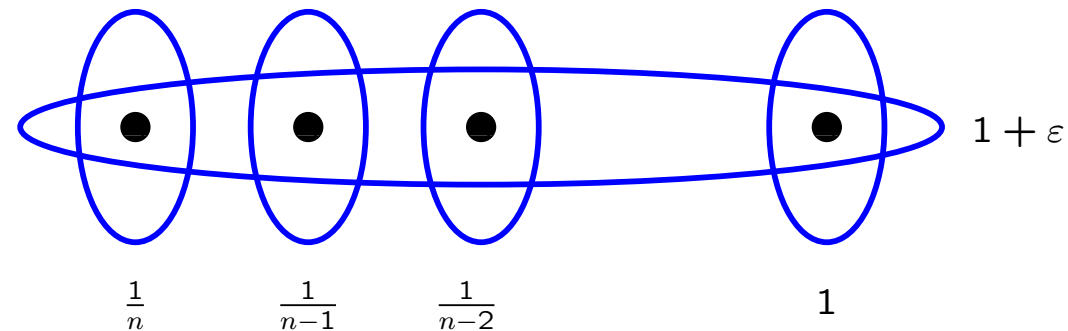
Todistus: Jakamalla kunkin peitteeseen valitun osajoukon S_* kustannus tasan peitetyksi tulleiden alkioiden kesken nähdään, että peitteen kokonaiskustannus on

$$\sum_{k=1}^n \text{price}(e_k) \leq \text{OPT} \cdot \sum_{k=1}^n \frac{1}{n - k + 1} = H_n \cdot \text{OPT},$$

missä on sovellettu edellistä lemmaa. \square

PCP-tekniikalla voidaan osoittaa, että jos lukumääräiselle joukkopeiteongelmalle olisi $(1 - \delta) \ln n$ -approksimointialgoritmi jollain $\delta > 0$, niin luokan NP ongelmat voitaisiin ratkaista deterministisesti ajassa $n^{O(\log \log n)}$ (ts. vain hyvin lievästi ylilynomisessa ajassa) (Feige 1998).

Esimerkki 2.3: Seuraava esimerkki, jossa kuvaan on merkitty joukkojen kustannukset, osoittaa, että edellinen raja ahneelle algoritmille on tiukka:



Optimipeitteessä on yksi joukko, kustannuksena $1 + \varepsilon$.

Ahne algoritmi valitsee yksi kerrallaan kaikki muut joukot, kustannuksena $1/n + 1/(n - 1) + \dots + 1 = H_n$. \square

Joukkopeite kerrostamalla

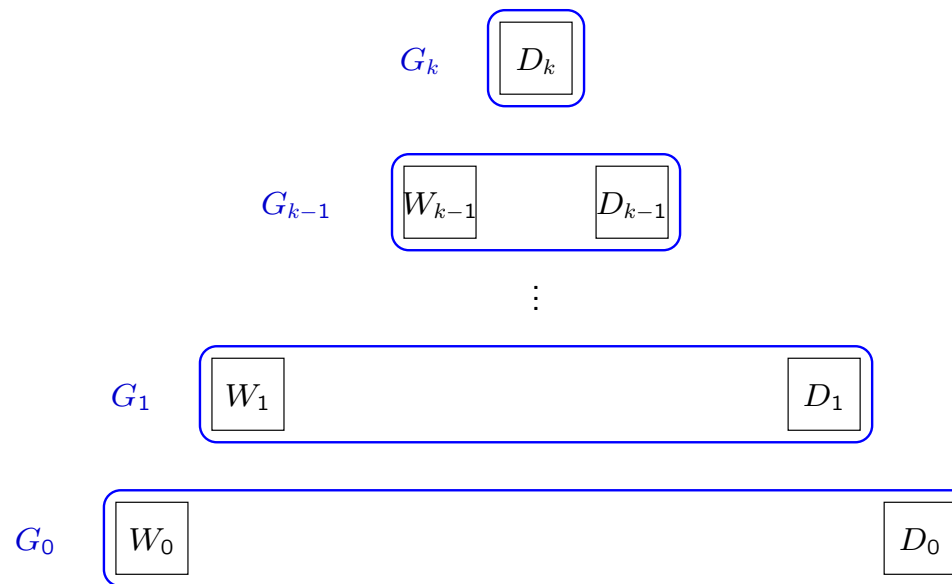
Perusjoukon alkion x frekvenssi on niiden osajoukkojen $S \in \mathcal{S}$ lukumäärä, joilla $x \in S$. Kerrostamistekniikalla saadaan minimijoukkopeitteelle f -approksimaatio, missä f on suurin näistä frekvenseistä.

Esitämme ratkaisun solmupeiteongelmaan, joka vastaa tapausta $f = 2$. (Verkon kaaret vastaavat perusjoukon alkioita ja solmut osajoukkoja.) Yleistäminen jää harjoitustehtäväksi.

Jos painofunktio $w: V \rightarrow \mathbb{Q}_+$ toteuttaa ehdon $w(v) = c \cdot \deg(v)$ kaikilla v , missä c on vakio ja $\deg(v)$ solmun v asteluku (naapurien lukumäärä), sanomme verkkoa **asteen mukaiseksi painoksi**. Perusideana on jakaa verkon kustannusfunktio "viipaleisiin", jotka ovat asteen mukaisia.

Algoritmi on seuraava:

1. Alusta $i \leftarrow 0$ ja $G_0 \leftarrow G$.
2. Olkoon D_i verkon G_i nolla-asteisten solmujen joukko; poista ne verkosta.
3. Jos verkkoon ei jäänyt solmuja, palauta solmupeite $C = W_0 \cup \dots \cup W_{i-1}$.
4. Jäljelle jääneiden solmujen joukossa laske $c = \min \{ w(v)/\deg(v) \}$.
Olkoon $t_i(v) = c \cdot \deg(v)$.
5. Olkoon W_i niiden solmujen v joukko, joilla $w(v) = t_i(v)$. Poista ne verkosta G_i .
6. Jäljellejääneillä solmuilla aseta $w(v) \leftarrow w(v) - c \cdot \deg(v)$. Ne muodostavat nyt verkon G_{i+1} .
7. Aseta $i \leftarrow i + 1$ ja palaa kohtaan 2.



Algoritmin palauttama solmupeite on $C = W_0 \cup \dots \cup W_{k-1}$.

Peitteen ulkopuolelle jää $V - C = D_0 \cup \dots \cup D_k$.

Lemma 2.4: Kerrostavan algoritmin palauttama joukko C on solmupeite.

Todistus: Tehdään vastaoletus, että kaari (u, v) ei tule peitetyksi. Siis $u \in D_i$ ja $v \in D_j$ joillakin i, j . Oletetaan esim. että $i \leq j$. Nyt verkossa G_i on edelleen mukana solmut u ja v ja siis myös kaari (u, v) , joten solmun u asteluku ei ole nolla. \square

Jäljellä on approksimointikertoimen arvioiminen asteenmukaisuutta käyttäen. Perustulos tässä on seuraava:

Lemma 2.5: Olkoon $w(v)$ asteenmukainen painofunktio. Nyt $w(V) \leq 2 \cdot \text{OPT}$.

Todistus: Siis $w(v) = c \cdot \text{deg}(v)$. Olkoon U optimipeite. Koska U peittää kaikki kaaret,

$$w(U) = c \cdot \sum_{v \in U} \text{deg}(v) \geq c|E|.$$

Toisaalta

$$w(V) = c \cdot \sum_{v \in V} \text{deg}(v) = 2c|E|.$$

□

Lause 2.6: Kerrostavan algoritmin palauttaman solmupeitteen kustannus on $w(C) \leq 2\text{OPT}$.

Todistus: Jos $v \in C$, niin $v \in W_j$ jollain j ja

$$w(v) = \sum_{i \leq j} t_i(v). \quad (*)$$

Jos taas $v \notin C$, niin $v \in D_j$ jollain j ja

$$w(v) \geq \sum_{i < j} t_i(v). \quad (**)$$

Olkoon C_* optimipeite. Koska $G_i = (V_i, E_i)$ on muodostettu verkosta G jättämällä pois solmuja (ja niihin rajoittuvia kaaria), niin $C_* \cap V_i$ on verkon G_i solmupeite. Kun varustetaan tämä verkko painoilla t_i ja sovelletaan edellistä lemmaa, saadaan

$$t_i(C \cap V_i) \leq t_i(V_i) \leq 2t_i(C_* \cap V_i).$$

"Viipaloimalla" painot $w(v)$ edellä esitetyllä tavalla saadaan

$$w(C) \stackrel{(*)}{=} \sum_{i=0}^{k-1} t_i(C \cap V_i) \leq 2 \sum_{i=0}^{k-1} t_i(C_* \cap V_i) \stackrel{(**)}{\leq} 2w(C_*).$$

□

Kerros 2 on tässäkin tiukka. Tämä nähdään soveltamalla algoritmia täydelliseen kaksijakoiseen verkkoon $K_{n,n}$, missä kunkin solmun paino on 1. Algoritmi poimii peitteeseen kaikki solmut, vaikka riittäisi valita vain toinen puoli. \square

Lyhin ylimerkkijono joukkopeitteellä

Annettu: joukko $S = \{s_1, \dots, s_n\} \subseteq \Sigma^+$ merkkijonoja

Löydettävä: lyhin merkkijono s , joka sisältää kaikki merkkijonot s_i osamerkkijonoinaan (ts. $s = u_i s_i v_i$ joillakin u_i ja v_i)

Tälle NP-kovalle lyhimmän ylimerkkijonon (shortest superstring) ongelmalle itse asiassa tunnetaan $2\frac{1}{2}$ -approksimointialgoritmi (Sweedyk 2000).

Esitämme tässä $O(\log n)$ -approksimointialgoritmin esimerkkinä joukkopeitteen moninaisista sovelluksista.

Oletamme jatkossa (yleisyyttä rajoittamatta), että mikään s_i ei ole minkään toisen s_j osamerkkijono.

Tarkastellaan ensin lyhyesti [ahnetta algoritmia](#).

Kahden merkkijonon s ja t [päällekkäisyys](#) (overlap) on pisimmän sellaisen merkkijonon pituus, joka on sekä merkkijonon s loppuosa että merkkijonon t alkuosa.

Algoritmi pitää yllä merkkijonojoukkoa T . Aluksi $T = S$. Kullakin iteraatiokierroksella algoritmi poimii joukosta kaksi merkkijonoa, joiden päällekkäisyys on suurin mahdollinen, ja korvaa ne niiden lyhimmillä ylimerkkijonolla.

Tämän algoritmin tiedetään saavuttavan approksimointisuhteen $3\frac{1}{2}$ ([Kaplan ja Shafir 2005](#)). Arvellaan, että se itse asiassa saavuttaa approksimointisuhteen 2, mutta tätä ei ole todistettu. Osoitamme tässä vain, että suhde ei voi olla parempi kuin 2. Tämä nähdään esimerkiksi

$$\{ ab^k, b^k c, b^{k+1} \}.$$

Jos algoritmi valitsee ensin merkkijonot ab^k ja $b^k c$, lopputulos on $ab^k cb^{k+1}$, joka on melkein kaksinkertainen optimitulokseen $ab^{k+1}c$ verrattuna.

Siirrymme nyt joukkopeitettä soveltavaan algoritmiin.

Kun $s_i, s_j \in S$ ja k on sellainen, että merkkijonon s_i viimeiset k merkkiä ovat samat kuin merkkijonon s_j viimeiset k merkkiä, olkoon σ_{ijk} merkkijono, joka saadaan laittamalla s_i ja s_j tämän mukaisesti päällekkäin. Tässä siis

- $s_i = uv$
- $s_j = vw$
- $|v| = k$
- $\sigma_{ijk} = uvw$.

Olkoon M kaikkien tällä tavoin määriteltyjen merkkijonojen σ_{ijk} joukko.

Merkkijonolle π olkoon

$$\text{set}(\pi) = \{s \in S \mid s \text{ on merkkijonon } \pi \text{ osamerkkijono}\}.$$

Muodostamme joukkopeitetapauksen, jossa perusjoukko on $S = S$ ja osajoukkoina ovat joukot $\text{set}(\pi)$, missä $\pi \in S \cup M$. Joukon $\text{set}(\pi)$ kustannus on $|\pi|$ (merkkijonon π pituus).

Algoritmi (Lyhin ylimerkkijono joukkopeitteellä)

1. Muodosta joukkopeiteongelma kuten edellä ja ratkaise ahneella algoritmilla. Olkoon $\text{set}(\pi_1), \dots, \text{set}(\pi_k)$ saatu ratkaisu.
2. Palauta $s = \pi_1 \dots \pi_k$.

Olkoon OPT ylimerkkijono-ongelman minimikustannus ja OPT_S joukkopeiteongelman.

Seuraavasta lauseesta ja ahneen joukkopeitealgoritmin H_n -approksimointikertoimesta seuraa, että yo. algoritmi on $2H_n$ -approksimointialgoritmi.

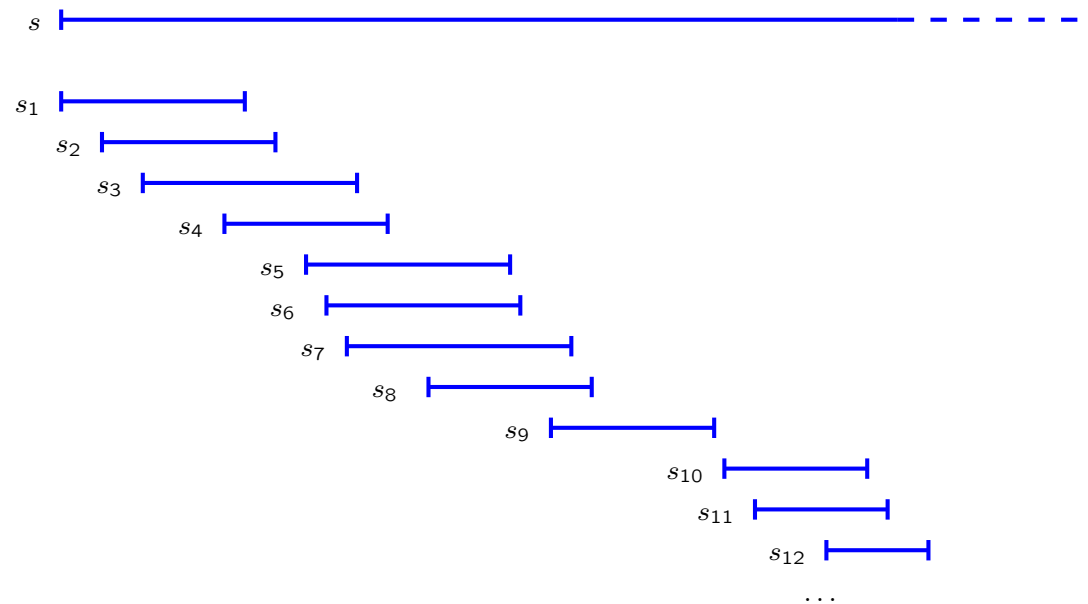
Lause 2.7: $\text{OPT} \leq \text{OPT}_S \leq 2\text{OPT}$.

Todistus: Olkoon optimijoukkopeite $\{\text{set}(\rho_1), \dots, \text{set}(\rho_l)\}$ ja $s = \rho_1 \dots \rho_l$. Joukkopeitteen kustannus on $|s|$.

Toisaalta s on käypä ratkaisu ylimerkkijono-ongelmaan, ja sen kustannus on $|s|$. Siis $\text{OPT} \leq |s| = \text{OPT}_S$.

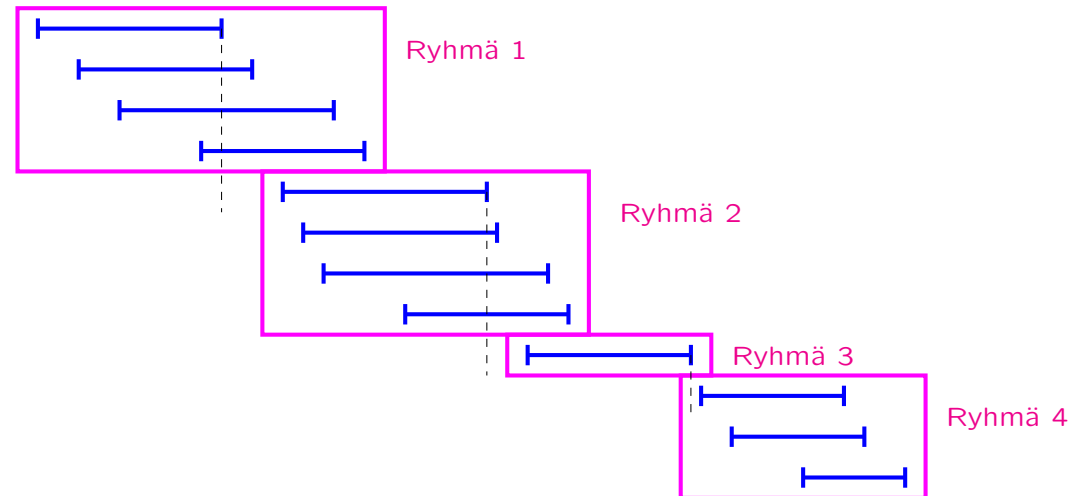
Olkoon toisaalta s merkkijonojen s_1, \dots, s_n lyhin ylimerkkijono; siis $|s| = \text{OPT}$. Muodostamme joukkopeiteongelmalle ratkaisun, jonka kustannus on korkeintaan $2|s|$.

Oletetaan merkkijonot s_i numeroiduksi siinä järjestyksessä, missä niiden vasemmanpuoleisimpien esiintymien alkukohtat ovat merkkijonossa s . Koska oletuksen mukaan mikään s_i ei ole toisen s_j osajono, nämä alkukohtat ovat erilliset ja myös vastaavien esiintymien loppuosat ovat samassa järjestyksessä.

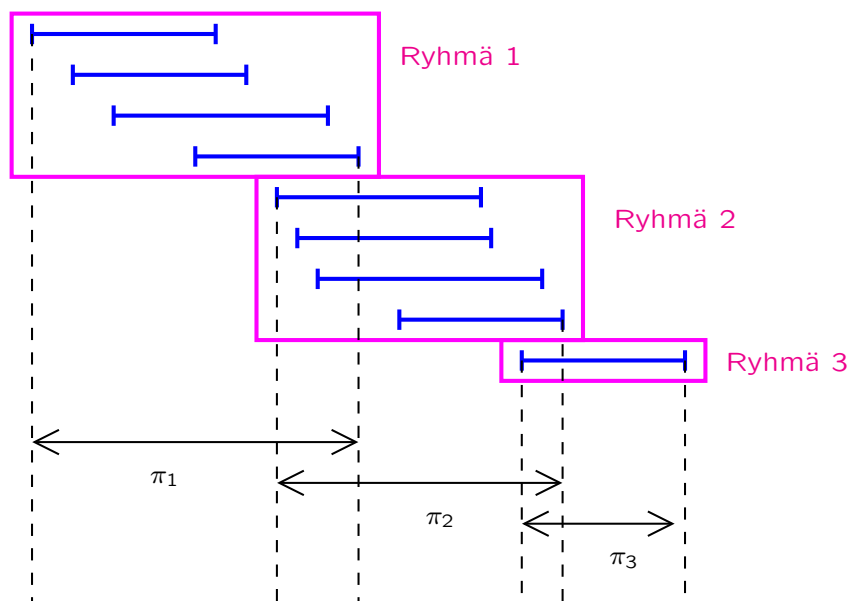


Ryhmitellään nyt merkkijonot s_1, \dots, s_n . Ryhmään 1 tulee s_1 ja sen kanssa päällekkäiset merkkijonot. Ryhmään 2 tulee ensimmäinen ryhmän 1 ulkopuolelle jäänyt merkkijono ja kaikki sen kanssa päällekkäiset, jne. Ryhmän koko voi olla myös 1.

(Tarkastelemme koko ajan merkkijonojen vasemmanpuoleisimpia esiintymiä.)



Muodostetaan nyt merkkijonot $\pi_1, \pi_2, \pi_3, \dots$ siten, että merkkijono π_r ulottuu ryhmän r ensimmäisen merkkijonon alusta ryhmän r viimeisen merkkijonon loppuun. Jos ryhmässä r on vain yksi merkkijono, niin $\pi_r \in S$. Muuten koska saman ryhmän merkkijonot ovat päällekkäisiä, π_r on muotoa σ_{ijk} joillain i, j, k . Siis joukot $\text{set}(\pi_r)$ ovat joukkopeiteongelman osajoukkoja. Joukko $\{\pi_r\}$ on joukkopeiteongelman ratkaisu, jonka kustannus on $\sum_r |\pi_r|$.



Koska merkkijonot π_r ja π_{r+1} voivat mennä päällekkäin, voi olla $\sum_r \pi_r > |s|$. Sen sijaan π_r ja π_{r+2} eivät voi mennä päällekkäin. Nimittäin ryhmän $r + 1$ ensimmäisen merkkijonon loppukohta on

- merkkijonon π_r loppumiskohdan oikealla puolella ja
- merkkijonon π_{r+2} alkamiskohdan vasemmalla puolella.

Siis jokainen merkkijonon s positio tulee lasketuksi korkeintaan kahteen merkkijonoon π_r , joten

$$\text{OPT}_S \leq \sum_r |\pi_r| \leq 2|s| = 2\text{OPT}.$$

□

Steiner-puut

Steiner-puuongelma on seuraava:

Annettu: Verkko $G = (V, E)$, kustannusfunktio $w: E \rightarrow \mathbb{Q}_+$,
solmujen ositus pakollisiin ja Steiner-solmuihin

Löydettävä: kustannukseltaan pienin verkon G puu, joka sisältää
ainakin kaikki pakolliset solmut.

Jos kustannusfunktio toteuttaa kolmioepäyhtälön

$$w(a, c) \leq w(a, b) + w(b, c)$$

kaikilla $a, b, c \in V$, kysymyksessä on metrinen Steiner-puuongelma.

Lause 2.8: Steiner-puuongelmasta on approksimointisuhteen säilyttävä palautus metriseen Steiner-puuongelmaan.

Todistus: Annetusta yleisestä Steiner-puuongelman tapauksesta I muodostetaan metrisen ongelman tapaus I' seuraavasti: Jos tapauksen I verkko on $G = (V, E)$, niin tapauksen I' verkko G' on täydellinen verkko solmujoukossa V . Jako pakollisiin ja Steiner-solmuihin pysyy ennallaan. Tapauksen I' kustannukseksi $w'(u, v)$ asetetaan kustannusfunktion w mukainen lyhin polunpituus verkossa G solmusta u solmuun v .

Tämä on selvästi polynomisessa ajassa laskettava kuvaus yleisen Steiner-puuongelman tapauksista metrisiin.

Palautuksen määritelmän mukaan pitää osoittaa, että

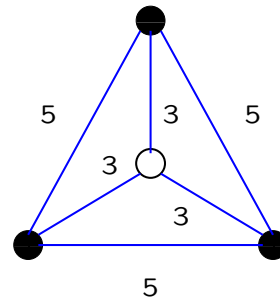
1. tapauksen I' optimikustannus ei ole suurempi kuin tapauksen I
2. tapauksen I' kelvollisesta ratkaisusta s' saadaan polynomisessa ajassa tapauksen I kelvollinen ratkaisu s , jonka kustannus ei ole suurempi.

Ehto 1 seuraa suoraan, koska kaikki tapauksen I kaaret ovat myös tapauksessa I' korkeintaan saman painoisina.

Ehto 2 toteutuu ottamalla alustavasti jokaista ratkaisun s' kaarta kohti ratkaisuun s vastaava polku, jolla on sama kustannus. Jos tämä tuottaa syklejä, poistetaan kaaria, kunnes s on syklitön; tämä ei kasvata kustannusta. \square

Rajoitumme siis ongelman metriseen versioon, koska edellisen lauseen mukaan sen approksimoituvuus on sama kuin yleisen version. Ongelman tarkka ratkaiseminen tiedetään NP-kovaksi.

Jos Steiner-solmujen joukko on tyhjä, ratkaisu on yksinkertaisesti pienin virittävä puu. Yleensä pakollisten solmujen pienin virittävä puu ei kuitenkaan ole optimaalinen ratkaisu Steiner-puuongelmaan:



Kuitenkin pienimmän virittävän puun etsiminen tuottaa 2-approksimointialgoritmin:

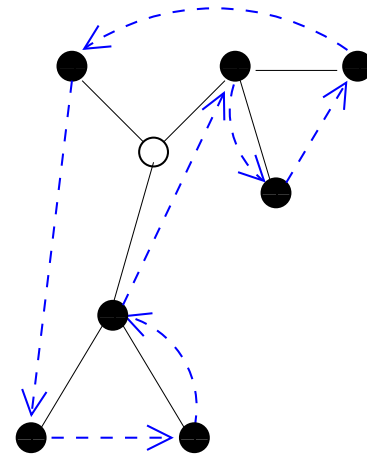
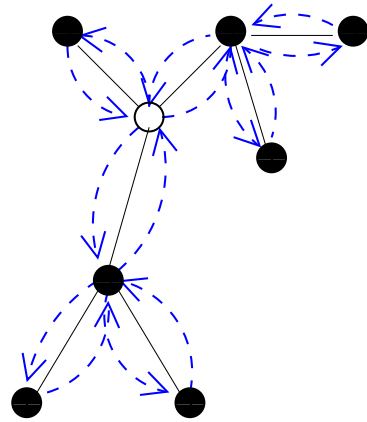
Lause 2.9: Pakollisten solmujen pienimmän virittävän puun kustannus on korkeintaan $2 \cdot \text{OPT}$.

Todistus: Tarkastellaan Steiner-puuta, jonka kustannus on OPT .

Muodostetaan siitä ensin suunnattu Eulerin kehä kulkemalla jokaista kaarta kumpaankin suuntaan. Kustannus on $2 \cdot \text{OPT}$.

Oikaistaan nyt edellistä Eulerin kehää ohittamalla Steiner-solmut ja pakollisten solmujen moninkertaiset esiintymät. Metrisyyden takia kustannus ei kasva. Saadaan pakollisten solmujen Hamiltonin kehä, jonka kustannus on korkeintaan $2 \cdot \text{OPT}$.

Poistamalla tästä kehästä yksi kaari saadaan pakollisten solmujen virittävä puu, jonka kustannus on korkeintaan $2 \cdot \text{OPT}$. \square



Siis 2-approksimointialgoritmi metriselle Steiner-puuongelmalle saadaan yksinkertaisesti etsimällä pienin virittävä puu pakollisten solmujen joukossa.

Approksimointisuhteen raja 2 on tiukka. Tiukka esimerkki saadaan verkosta, jossa on yksi Steiner-solmu ja n pakollista solmua. Kaaren (u, v) kustannus on

- $w(u, v) = 2$ jos u ja v ovat pakollisia
- $w(u, v) = 1$ jos u tai v on Steiner-solmu.

Kustannusfunktio toteuttaa kolmioepäyhtälön. Pienimmän virittävän puun kustannus on $2(n - 1)$ ja optimaalisen Steiner-puun n .

Kauppamatkustajan ongelma (TSP)

Tapaus: täydellinen verkko G , kullekin kaarelle e ei-negatiivinen paino $w(e) \in \mathbb{Q}_+$.

Käyvät ratkaisut: verkon G Hamiltonin kehät, ts. syklit jotka käyvät jokaisessa solmussa tasan kerran

Kustannus: kehällä olevien kaarten painojen summa (minimointitehtävä).

Ongelma on **metrinen**, jos painofunktio toteuttaa kolmioepäyhtälön

$$w(a, c) \leq w(a, b) + w(b, c) \quad \text{kaikilla } a, b, c.$$

Yleisessä muodossaan TSP ei ole approksimoitavissa, jos $P \neq NP$.

Lause 2.10: Jos $P \neq NP$, niin kauppamatkustajan ongelmalla ei ole $\alpha(n)$ -approksimointialgoritmiä millään polynomisessa ajassa laskettavissa olevalla α .

Todistus: Tehdään vastaoletus, että polynomisessa ajassa toimiva algoritmi A on α -approksimointialgoritmi TSP:lle. Muodostamme tästä polynomisessa ajassa toimivan ratkaisun Hamiltonin kehä -ongelmalle.

Olkoon $G = (V, E)$ Hamiltonin kehä -ongelman tapaus. Muodostetaan täydellinen verkko G' , jonka solmujoukko on V . Kaaren (u, v) kustannus on

- $w(u, v) = 1$ jos $(u, v) \in E$
- $w(u, v) = n\alpha(n)$ jos $(u, v) \notin E$.

Jos verkossa G on Hamiltonin kehä, niin tämä kehä on myös TSP-ongelman optimiratkaisu. Sen kustannus on n .

Jos verkossa G ei ole Hamiltonin kehää, niin TSP-tapauksen optimikustannus on ainakin $n\alpha(n)$.

Siis verkossa G on Hamiltonin kehä, jos ja vain jos A palauttaa ratkaisun, jonka kustannus on korkeintaan $n\alpha(n)$. \square

Metristä TSP:tä voidaan approksimoida samantapaisilla oikopolkutekniikoilla kuin Steiner-puuongelman analyysissä.

Algoritmi (metrinen TSP virittävällä puulla)

1. Muodosta verkolle G pienin virittävä puu T .
2. Muodosta suunnattu verkko G' ottamalla puun T kaaret kumpaankin suuntaan (yhteensä $2(n - 1)$ kaarta).
3. Muodosta verkossa G' Eulerin kehä P .
4. Muodosta Hamiltonin kehä C ohittamalla kehällä P solmut, joissa on jo käyty. Palauta C .

Lause 2.11: Edellinen algoritmi on 2-approksimointialgoritmi metriselle kauppamatkustajan ongelmalle.

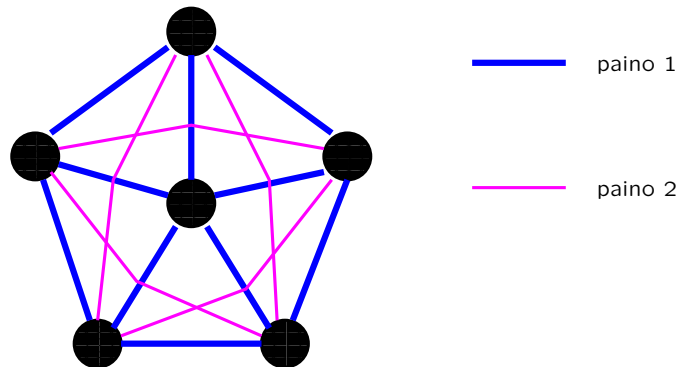
Todistus: Perushavainto on, että pienimmän virittävän puun T kustannus on samalla yläraja TSP:n minimikustannukselle OPT .

Siis Eulerin kehän P kustannus on korkeintaan $2 \cdot OPT$.

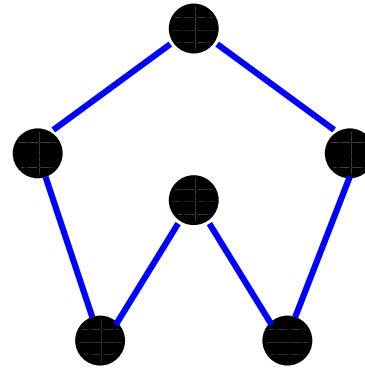
Kolmioepäyhtälön takia solmujen ohittaminen ei kasvata polun kustannusta, joten Hamiltonin kehän C kustannus on sekin korkeintaan $2 \cdot OPT$. \square

Approksimaatiokerroin 2 nähdään tiukaksi tarkastelemalla $n + 1$ solmun verkkoa, jossa on

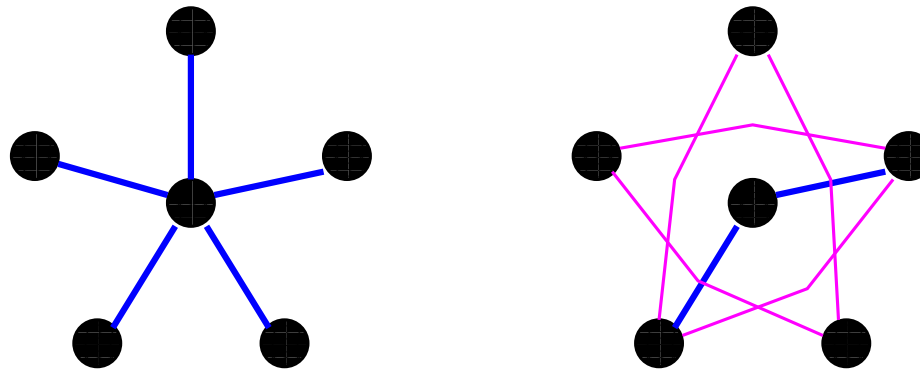
- n -sakarainen tähti, kaarten paino 1
- tähden sakaroita yhdistävät kaaret, paino 1
- muiden kaarten paino 2



Optimikustannus on $n + 1$:



Pahimmassa tapauksessa virittäväksi puuksi valitaan tähti ja oikaisut tehdään sellaisessa järjestyksessä, että kustannukseksi tulee $2n + 2$:



Edellistä algoritmia parantamalla voidaan saavuttaa approksimointisuhde $3/2$:

Algoritmi (metrinen TSP virittävällä puulla (Cristofides 1976))

1. Muodosta verkolle G pienin virittävä puu T .
2. Muodosta minimikustannuksinen täydellinen pariutus M niille solmuille, joiden asteluku puussa T on pariton.
3. Muodosta Eulerin kehä P kaarille $T \cup M$.
4. Muodosta Hamiltonin kehä C ohittamalla kehällä P solmut, joissa on jokin asteluku pariton. Palauta C .

Huom. paritonasteisten solmujen lukumäärä puussa T on parillinen, koska astelukujen summa $2|T|$ on parillinen.

Pikakertaus: Eulerin kehä

- Eulerin kehä on (suunnatun tai suuntaamattoman) verkon kehä, joka käyttää jokaista **kaarta** tasan kerran
- Suuntaamattomassa verkossa on Eulerin kehä, jos ja vain jos se on yhtenäinen ja jokaisen solmun asteluku on parillinen.
- Suunnatussa verkossa on Eulerin kehä, jos ja vain jos se on yhtenäinen ja jokaisen solmun tuloaste on sama kuin lähtöaste.
- Eulerin kehän löytämiseen on yksinkertainen ja nopea algoritmi.

Lemma 2.12: Olkoon C optimiratkaisu metriseen TSP-tapaukseen (V, E, w) , ja $V' \subseteq V$ joukko, jossa on parillinen määrä solmuja. Nyt joukolla V' on täydellinen pariutus M , jonka kustannukselle pätee $w(M) \leq w(C)/2$.

Todistus: Muodostetaan C' oikaisemalla C ohi niiden solmujen, jotka eivät kuulu joukkoon V' . Metrisyyden takia $w(C') \leq w(C)$.

Laitetaan polulta C' joka toinen kaari joukkoon M' , joka toinen joukkoon M'' . Sekä M' että M'' ovat solmujoukon V' täydellisiä pariutuksia. Koska $w(M') + w(M'') = w(M' \cup M'') = w(C')$, pätee $\min \{w(M'), w(M'')\} \leq w(C') \leq w(C)$. \square

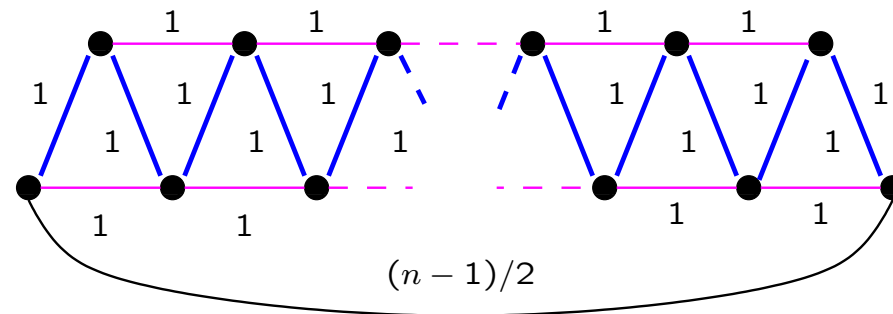
Lause 2.13: Cristofidesin algoritmi on $3/2$ -approksimointialgoritmi metriselle kauppamatkustajan ongelmalle.

Todistus: Kehän C' kaaret kuuluvat joko puuhun T tai pariutukseen M .

Kuten edellisen algoritmin yhteydessä todettiin, $w(T) \leq \text{OPT}$, missä OPT on TSP-tapauksen minimikustannus.

Edellisen lemmän perusteella $w(M) \leq \text{OPT}/2$. \square

Seuraava esimerkki osoittaa, että approksimointisuhteen raja $3/2$ on tiukka:



Verkossa on n solmua; pitkän kaaren paino on valittu siten, että kolmioepäyhtälö toteutuu. (Kuvasta puuttuvien kaarten kustannus on kuvaan merkityistä kaarista saatava lyhin polun pituus.)

Pahimmassa tapauksessa virittävään puuhun valitaan paksut kaaret. Parittomiksi jää vasemman- ja oikeanpuoleisin solmu, jotka pariutetaan. Kustannukseksi tulee $(n - 1) + (n - 1)/2$.

Optimaalisen TSP-reitin kustannus on n .

Paremmen approksimointisuhteen kuin $3/2$ saavuttaminen metrisessä TSP:ssä on avoin ongelma.

Euklidisessa kauppamatkustajan ongelmassa oletetaan vahvemmin, että solmut ovat pisteitä avaruudessa \mathbb{R}^d ja kaarten painot solmujen euklidisia etäisyyksiä. Tälle ongelmalla on mahdollista löytää $(1 + \varepsilon)$ -approksimointialgoritmi millä tahansa $\varepsilon > 0$, mutta aikavaativuus on eksponentiaalinen parametrin $1/\varepsilon$ suhteen (**Arora 1996**).

k -keskusongelma (k -center)

Tapaus:

- täydellinen verkko $G = (V, E)$
- painofunktio $\text{cost}: E \rightarrow \mathbb{Q}_+$
- luonnollinen luku k

Käyvät ratkaisut: joukot $S \subseteq V$, joissa on tasan k solmua

Kustannusfunktio: minimoitava

$$\max_{v \in V} \min_{u \in S} \text{cost}(u, v).$$

Tulkinta: Ajatellaan, että solmut ovat kaupunkeja ja S on niiden kaupunkien joukko, joihin perustetaan varasto. Lauseke $\min_{u \in S} \text{cost}(u, v)$ antaa kaupunkia v lähinnä olevan varaston etäisyyden. Halutaan sijoitella k varastoa niin, että pahimman tapauksen etäisyys lähimpään varastoon minimoituu.

Tarkastelemme jatkossa **metristä** k -keskusongelmaa, jossa kaarten painot toteuttavat kolmioepäyhtälön. Ei-metrisessä tapauksessa ongelmalla ei ole $\alpha(n)$ -approksimointialgoritmia millään polynomisessa ajassa laskettavalla α .

Esitämme parametriseen karsintaan (parametric pruning) perustuvan 2-aproksimointialgoritmin.

Perusidea on, että jos tietäisimme optimiratkaisun arvon OPT , voisimme yksinkertaistaa ongelmaa karsimalla pois osia, jotka eivät tule kysymykseen. Erityisesti metrisessä k -keskusongelmassa jos $OPT \leq B$, niin ratkaisu pysyy ennallaan, kun poistamme kaikki kustannukseltaan yli B olevat kaaret.

Käytännössä emme tietenkään tiedä arvoa OPT , ja kuten itsepalautusten yhteydessä totesimme, sen löytäminen on suunnilleen yhtä vaikeaa kuin koko ongelman ratkaiseminen. Siksi kokeilemme jonoa arvauksia.

Edellisen perusteella OPT on jonkin kaaren paino. Olkoon $E = \{e_1, \dots, e_m\}$, missä $\text{cost}(e_1) \leq \text{cost}(e_2) \leq \dots \leq \text{cost}(e_m)$. Merkitään $G_i = (V, \{e_1, \dots, e_i\})$.

Verkon $H = (U, F)$ **dominoiva joukko** on sellainen solmujoukko S , että $S \cup (\cup_{v \in S} N(v)) = V$, missä $N(v)$ on solmun v naapurusto; siis jos solmu v ei itse kuulu joukkoon S , niin ainakin jokin sen naapureista kuuluu. Olkoon $\text{dom}(H)$ pienimmän dominoivan joukon koko; tämän laskeminen on NP-kova ongelma.

Verkon G_i dominoivat joukot, joiden koko on k , vastaavat suoraan k -keskusongelman käypiä ratkaisuja, joiden kustannus on korkeintaan $\text{cost}(e_i)$. Tällainen on olemassa, jos ja vain jos verkossa G_i kaikki solmut voidaan peittää k :lla tähdellä (muotoa $K_{1,p}$ olevalla osaverkolla).

Olkoon i^* pienin indeksi, jolla G_i :n solmut voidaan peittää k :lla tähdellä. Siis $\text{OPT} = \text{cost}(e_{i^*})$.

Verkon $H = (U, F)$ **neliö** on $H^2 = (V, F')$, missä F' sisältää kaaren (u, v) , jos $u \neq v$ ja verkossa H on korkeintaan kahden kaaren mittainen polku solmusta u solmuun v .

Lemma 2.14: Olkoon I riippumaton joukko verkossa H^2 . Tällöin

$$|I| \leq \text{dom}(H).$$

Todistus: Olkoon D verkon H pienin dominoiva joukko. Siis H :n solmut voidaan peittää $|D|$:llä tähdellä. Verkossa H^2 jokainen näistä tähdistä muodostaa klikin. Siis I voi sisältää korkeintaan yhden solmun kustakin tähdestä. \square

Algoritmi (metrinen k -keskus)

1. Muodosta $G_1^2, G_2^2, G_3^2, \dots, G_m^2$.
2. Muodosta jokaisessa G_i^2 maksimaalinen riippumaton joukko M_i .
3. Olkoon j pienin sellainen, että $|M_j| \leq k$.
4. Palauta M_j .

Analyysi perustuu seuraavaan alarajaan:

Lemma 2.15: Algoritmin mukainen j toteuttaa ehdon

$$\text{cost}(e_j) \leq \text{OPT}.$$

Todistus: Kaikilla $i < j$ pätee $|M_i| > k$, eli verkossa G_i^2 on yli k -solmuinen riippumaton joukko. Edellisen lemmän mukaan tällöin $\text{dom}(G_i) > k$, joten $i^* > i$.

Siis $i^* \geq j$. \square

Lause 2.16: Edellinen algoritmi on 2-aproksimointialgoritmi metriselle k -keskusongelmalle.

Todistus: Maksimaalinen riippumaton joukko I on aina myös dominoiva, sillä muuten $I \cup \{v\}$ olisi riippumaton jollain $v \notin I$.

Siis valitsemalla keskipisteiksi joukon M_j solmut saadaan verkon G_j^2 solmut peitetyksi k :lla tähdellä.

Kolmioepäyhtälön nojalla näissä tähdissä olevien kaarien painot ovat korkeintaan $2 \cdot \text{cost}(e_j)$. \square

Esimerkki 2.17: Tiukka esimerkki saadaan tarkastelemalla $n + 1$ solmun verkkoa, jossa

- solmusta $n + 1$ on jokaiseen muuhun solmuun kaari kustannukseltaan 1
- kaikkien solmujen $1, \dots, n$ välisten kaarten kustannus on 2.

Kun $k = 1$, optimiratkaisu muodostuu solmusta $n + 1$ ja $\text{OPT} = 1$.

Algoritmin palauttama arvo on $j = n$. Verkko G_n^2 on koko verkon käsittävä klikki. Jos joukoksi M_n on valittu jokin solmuista $1, \dots, n$, algoritmin tuottama kustannus on 2. \square

Itse asiassa pätee

Lause 2.18: Jos $P \neq NP$, niin metrisellä k -keskusongelmalla ei ole $(2 - \varepsilon)$ -approksimointialgoritmiä millään $\varepsilon > 0$.

Todistus: Oletetaan, että A on $(2 - \varepsilon)$ -approksimointialgoritmi, ja johdetaan tästä polynomisessa ajassa toimiva algoritmi dominoivalle joukolle.

Olkoon (G, k) dominoivan joukon tapaus, missä $G = (V, E)$. Muodostetaan täydellinen verkko $G' = (V, E')$, missä

$$\text{cost}(u, v) = \begin{cases} 1 & \text{jos } (u, v) \in E \\ 2 & \text{jos } (u, v) \notin E. \end{cases}$$

Kolmioepäyhtälö toteutuu. Lisäksi

- Jos $\text{dom}(G) \leq k$, niin optimaalisen k -keskuksen kustannus on 1. Myös algoritmin A löytämä kustannus on 1, sillä se on ainoa mahdollisuus kertoimen $(2 - \varepsilon)$ sisällä optimista.
- Jos $\text{dom}(G) > k$, niin optimaalisen k -keskuksen kustannus on 2, ja A :n löytämä kustannus on tietysti vähintään 2.

Siis algoritmilla A voidaan ratkaista dominoiva joukko -ongelma. \square

Painotettu keskustensijoitteluongelma

Tapaus:

- täydellinen verkko $G = (V, E)$
- solmujen painofunktio $w: V \rightarrow \mathbb{Q}_+$
- kaarten painofunktio $\text{cost}: E \rightarrow \mathbb{Q}_+$
- painoraja $W \in \mathbb{Q}_+$

Käyvät ratkaisut: joukot $S \subseteq V$, joilla $w(S) := \sum_{v \in S} w(v) \leq W$

Kustannusfunktio: minimoitava

$$\max_{v \in V} \min_{u \in S} \text{cost}(u, v).$$

Painottamaton ongelma siis on erikoistapaus, jossa $w(v) = 1$ kaikilla v (ja $W = k$).

Yleistetään vastaavasti dominoiva joukko -ongelma painotetuille verkoille. Olkoon $w\text{dom}(G)$ pienin dominoivan joukon paino verkossa G .

Tarkastellaan samoja joukkoja G_i kuin aiemmin. Olkoon i^* pienin i , jolla $w\text{dom}(G_i) \leq W$. Kuten edellä, optimaalisen keskusratkaisun kustannus on $\text{OPT} = \text{cost}(e_{i^*})$.

Olkoon H solmupainoilla varustettu verkko ja I riippumaton joukko verkossa H^2 . Solmuilla $u \in I$ olkoon $s(u)$ painoltaan pienin solmu joukossa $\{u\} \cup N(u)$, missä $N(u)$ on solmun u naapurusto verkossa H . Asetetaan

$$S = \{s(u) \mid u \in I\}.$$

Lemma 2.19: $w(S) \leq \text{wdom}(H)$.

Todistus: Olkoon D verkon H minimipainoinen dominoiva joukko. Siis verkon H solmut voidaan peittää tähdillä, joiden keskipisteinä on joukon D solmut.

Jokainen näistä verkon H tähdistä vastaa klikkiä verkossa H^2 . Siis I voi sisältää vain yhden solmun kustakin tähdestä. Jos $u \in I$ ja $c(u)$ on solmun u sisältävän tähden keskipiste, niin $w(s(u)) \leq w(c(u))$. Siis

$$\sum_{u \in I} w(s(u)) \leq \sum_{u \in I} w(c(u)) \leq \sum_{v \in D} w(v).$$

□

Algoritmi (painotettu metrinen keskustensijoitteluongelma)

1. Muodosta $G_1^2, G_2^2, G_3^2, \dots, G_m^2$.
2. Muodosta jokaisessa G_i^2 maksimaalinen riippumaton joukko M_i .
3. Muodosta $S_i = \{s_i(u) \mid u \in M_i\}$, missä $s_i(u)$ on painoltaan pienin solmu joukossa $\{u\} \cup N(u)$ (naapurusto verkon G_i mukaan).
4. Olkoon j pienin sellainen, että $w(S_j) \leq W$.
5. Palauta S_j .

Lemma 2.20: $\text{cost}(e_j) \leq \text{OPT}$.

Todistus: Analoginen lemmän 2.15 kanssa. \square

Lause 2.21: Edellinen algoritmi on 3-approksimointialgoritmi.

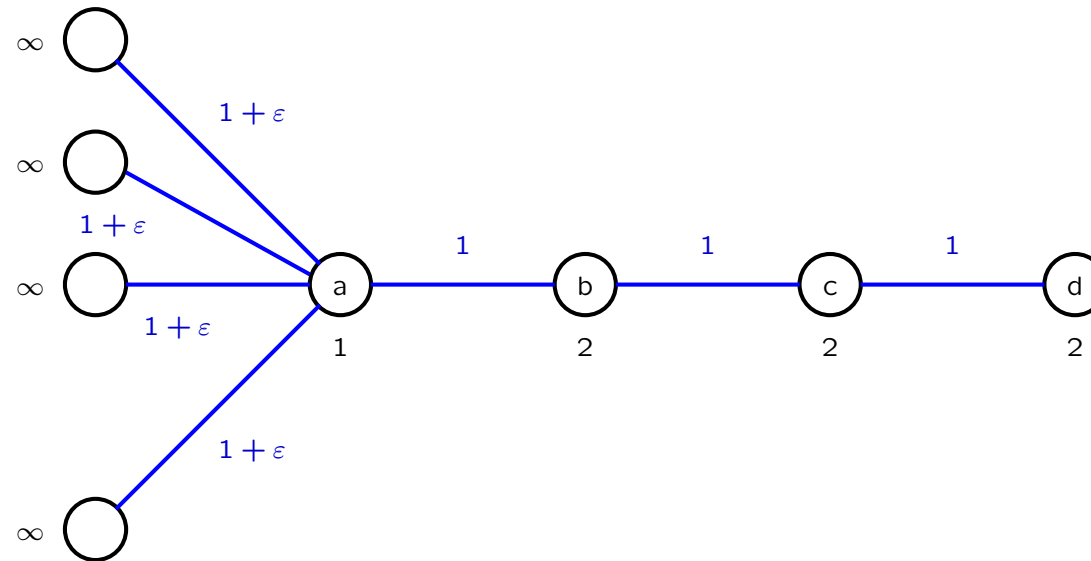
Todistus: Koska M_j on verkon G_j^2 dominoiva joukko, voimme peittää solmujoukon V verkon G_j^2 tähdillä, joiden keskipisteinä ovat joukon M_j solmut. Verkon G_j^2 kaarten kustannukset ovat korkeintaan $2 \cdot \text{cost}(e_j)$.

Olkoon v mielivaltainen solmu. Oletetaan, että se kuuluu tähteen, jonka keskus on u . Siis $\text{cost}(u, v) \leq 2 \cdot \text{cost}(e_j)$.

Solmu $s_j(u)$ on solmun u naapuri verkossa G_j , joten $\text{cost}(s_j(u), u) \leq \text{cost}(e_j)$. Kolmioepäyhtälön nojalla $\text{cost}(s_j(u), v) \leq 3 \cdot \text{cost}(e_j)$.

Siis solmut $s_j(u)$ antavat keskustensijoitteluongelmalle ratkaisun, jonka kustannus on korkeintaan $3 \cdot \text{cost}(e_j) \leq 3 \cdot \text{OPT}$. \square

Esimerkki 2.22: Allaoleva tiukka esimerkki osoittaa, että algoritmille ei saada parempaa approksimointisuhdetta kuin 3 (kuvasta puuttuvien kaarten painot saadaan minimipolunpituuksista):



Arvolla $W = 3$ optimiratkaisu on $\{a, c\}$ ja kustannus $1 + \epsilon$.

Olkoon n on ∞ -painoisten solmujen lukumäärä. Kaikilla $i < n + 3$ joukko S_i sisältää ∞ -painoisen solmun. Tilanteessa $i = n + 3$ eräs maksimaalinen riippumaton joukko on $\{b\}$, jolloin $s(b) = a$. Siis algoritmi tulostaa joukon $\{a\}$ kustannuksella 3. \square

Approksimointiskeemat

Olkoon Π NP-optimointiongelma ja sen kustannusfunktio f_Π .
Approksimointiskeema ongelmalle Π on algoritmi, joka

- saa syötteenä parin (I, ε) , missä I on ongelman Π tapaus ja $\varepsilon \in \mathbb{Q}_+$
- tulostaa sellaisen tapaukselle I käyvän ratkaisun s , että
- minimointiongelmassa $f_\Pi(I, s) \leq (1 + \varepsilon)\text{OPT}$
- maksimointiongelmassa $f_\Pi(I, s) \geq (1 - \varepsilon)\text{OPT}$.

Approksimointiskeema on **polynominen** (polynomial time approximation scheme, **PTAS**), jos sen aikavaativuus on polynominen tapauksen koon $|I|$ suhteen.

Jos aikavaativuus on lisäksi polynominen parametrin $1/\varepsilon$ suhteen, approksimointiskeema on **täysin polynominen** (fully polynomial time approximation scheme, **FPTAS**).

Siis esim. aikavaativuus $O(n^{1/\varepsilon})$ riittää PTAS:aan, mutta ei FPTAS:aan.

Repunpakkausongelma (knapsack)

Tapaus:

- joukko esineitä $S = \{a_1, \dots, a_n\}$
- esineiden **koot** $\text{size}(a_i) \in \mathbb{Z}_+$, $i = 1, \dots, n$
- esineiden **tuotot** $\text{profit}(a_i) \in \mathbb{Z}_+$, $i = 1, \dots, n$
- "repun koko" $B \in \mathbb{Z}_+$

Käyvät tapaukset: esinejoukot $R \subseteq S$ joilla

$$\text{size}(R) := \sum_{r \in R} \text{size}(r) \leq B$$

Kohdefunktio: maksimoitava $\text{profit}(R) := \sum_{r \in R} \text{profit}(r)$

Yleisyyttä rajoittamatta oletamme, että $\text{size}(a_i) \leq B$ kaikilla i .

Ongelma on NP-kova.

Ilmeinen ahne algoritmi valitse alkioita tuotto/koko-suhteen mukaisessa järjestyksessä, kunnes reppu on täysi. Tässä se ei kuitenkaan toimi hyvin.

Pseudopolynominen ratkaisu repunpakkausongelmaan

Merkitään $P = \max \{ \text{profit}(a_i) \}$. Selvästi $\text{OPT} \leq nP$.

Olkoon $A(i, p)$ pienin koko $\text{size}(R)$ sellaiselle R , että

- $\text{profit}(R) = p$ (huom. yhtäsuuruus)
- $R \subseteq \{ a_1, \dots, a_i \}$ (missä esineiden numerointi on mielivaltainen mutta kiinteä).

Merkitsemme $A(i, p) = \infty$ jos tällaista R ei ole olemassa, erityisesti jos $p < 0$.

Algoritmi (tarkka ratkaisu repunpakkausongelmalle)

1. Alusta $A(i, 0) \leftarrow 0$ kun $i = 0, \dots, n$ ja $A(0, p) \leftarrow \infty$ kun $p = 1, \dots, P$.
2. Toista arvoilla $i = 1, \dots, n$:

Toista arvoilla $p = 1, \dots, nP$:

$$A(i, p) \leftarrow \min \{ A(i-1, p), \text{size}(a_i) + A(i, p - \text{profit}(a_i)) \}$$

3. Palauta suurin p , jolla $A(n, p) \leq B$.

Aikavaativuus on $O(n^2P)$.

Edellisen algoritmin aikavaativuus $O(n^2P)$ ei ole polynominen. Polynomisuushan määritellään **syöteen koon** suhteen. Koska luvut oletetaan esitettyksi **binäärikoodattuina**, tuottojen koot ovat $O(\log P)$.

Sanomme **pseudopolynomiseksi** tällaisia algoritmeja, joiden aikavaativuus olisi polynominen, jos syötteeseen kuuluvat luvut olisikin koodattu unaarisesti, ts. kokonaisluvun p kooksi laskettaisiin $O(p)$ eikä $O(\log p)$.

Pseudopolynominen algoritmi siis antaa polynomisen ratkaisun sellaisille tapauksille, joissa esiintyvät luvut ovat polynomisia syöteen koon suhteen.

Esim. kauppamatkustajan ongelmalle ei ole pseudopolynomista algoritmia, jos $P \neq NP$. Tämä nähdään siitä, että ongelman NP-täydellisyystodistuksessa (palautus Hamiltonin kehästä) riittää käyttää vakiokokoisia painoja.

Jos pseudopolynominen algoritmi on olemassa, siitä saadaan usein FPTAS esittämällä syöte rajoitetulla tarkkuudella.

Algoritmi (FPTAS repunpakkausongelmalle)

1. Olkoon $K = (\varepsilon P)/n$, missä $P = \max \{ \text{profit}(a_i) \}$ ja ε on syötteenä saatu tarkkuusvaatimus.
2. Muodosta skaalatut tuotot

$$\text{profit}'(a_i) = \left\lfloor \frac{\text{profit}(a_i)}{K} \right\rfloor.$$

3. Ratkaise ongelma tarkalla algoritmilla käyttäen skaalattuja tuottoja. Olkoon ratkaisu S' .
4. Tulosta S' .

Lause 2.23: Algoritmin aikavaativuus on polynominen repunpakkaustapauksen koon ja parametrin $1/\varepsilon$ suhteen.

Todistus: Skaalatun ongelman suurin tuotto on $P' \leq P/K = n/\varepsilon$.

Tarkan algoritmin aikavaativuus on $O(n^2 P') = O(n^3/\varepsilon)$. \square

Lause 2.24: Algoritmin tuottama ratkaisu toteuttaa ehdon

$$\text{profit}(S') \geq (1 - \varepsilon)\text{OPT}.$$

Todistus: Olkoon O optimiratkaisu alkuperäiseen ongelmaan. Kaikilla a pätee

$$\frac{\text{profit}(a)}{K} - 1 \leq \text{profit}'(a) \leq \frac{\text{profit}(a)}{K}.$$

Siis erityisesti

$$K \cdot \text{profit}'(O) \geq \text{profit}(O) - nK.$$

Koska $\text{profit}'(S') \geq \text{profit}'(O)$, saadaan

$$\begin{aligned} \text{profit}(S') &\geq K \cdot \text{profit}'(S') \\ &\geq K \cdot \text{profit}'(O) \\ &\geq \text{profit}(O) - nK \\ &\geq \text{OPT} - \varepsilon P \\ &\geq (1 - \varepsilon) \cdot \text{OPT}, \end{aligned}$$

sillä $\text{OPT} \geq P$. \square

Vahva NP-kovuus

Ongelma on **vahvasti** NP-kova, jos se säilyisi NP-kovana, vaikka tapaukseen kuuluvien lukujen koodaus muutettaisiin unaariseksi. Edellä esitetyn perusteella tämä näyttäisi liittyvän FPTAS:n olemassaoloon ja pseudopolynomisiin algoritmeihin.

Lause 2.25: Olkoon $|I_u|$ tapauksen I koko, kun lukujen koodaus on muutettu unaariseksi. Tarkastellaan NP-minimointiongelmaa Π , jonka kohdefunktio on kokonaislukuarvoinen ja jolle $\text{OPT}(I) < p(|I_u|)$ jollain polynomilla p . Jos ongelmalla Π on FPTAS, niin sillä on myös tarkka pseudopolynomisen algoritmi.

Todistus: Olkoon FPTAS:n aikavaativuus $q(|I|, 1/\varepsilon)$ jollain polynomilla q .

Pseudopolynomisen algoritmi syötteellä I kutsuu FPTAS:ää parametreillä (I, ε) , missä $\varepsilon = 1/p(|I_u|)$. Saadun ratkaisun kustannus on korkeintaan

$$(1 + \varepsilon)\text{OPT}(I) < \text{OPT}(I) + \varepsilon p(|I_u|) = \text{OPT}(I) + 1.$$

Koska kohdefunktio on kokonaislukuarvoinen, kustannuksen on siis oltava tasan OPT .

Aikavaativuus $q(|I|, p(|I_u|))$ on suureen $|I_u|$ suhteen polynomisen, eli pseudopolynomisen. \square

Huom. jos oletettaisiin vahvemmin $\text{OPT}(I) < p(|I|)$, saataisiin polynominen algoritmi, mutta tämä rajaa pois monia kiinnostavia ongelmia.

Korollari 2.26: Tarkastellaan edellisen lauseen oletukset toteuttavaa ongelmaa Π . Jos Π on vahvasti NP-kova, sillä ei ole FPTAS:ää, olettaen $P \neq \text{NP}$.

Todistus: Jos FPTAS on olemassa, edellinen lause antaa pseudopolynomisen algoritmin. Vahvan NP-kovuuden määritelmästä seuraa tällöin $P = \text{NP}$. \square

Laatikonpakkausongelma (bin packing)

Tapaus: jono esineiden kokoja a_1, \dots, a_n , missä $0 < a_i < 1$

Käyvät ratkaisut: esineiden sijoittelut laatikoihin B_i siten, että kuhunkin laatikkoon tulevien esineiden yhteenlaskettu koko on korkeintaan 1

Kohdefunktio: minimoitava laatikoiden lukumäärä.

Vaihtoehtoinen tulkinta: Halutaan ostaa mahdollisimman vähän vakiomittaisia lautoja niin, että niistä voidaan sahata halutun mittaiset pätkät.

Algoritmi (First Fit -heuristiikka laatikonpakkaukseen)

Käy esineet läpi mielivaltaisessa järjestyksessä.

Pidä yllä järjestettyä listaa laatikoista B_i .

Jos vuorossa oleva esine ei mahdu mihinkään vajaan laatikkoon, lisää uusi tyhjä laatikko listan loppuun ja käytä sitä. Muuten laita esine ensimmäiseen laatikkoon, jossa on riittävästi tilaa. \square

Lause 2.27: First Fit on 2-aproksimointialgoritmi.

Todistus: Olkoon m käytettyjen laatikoiden lukumäärä. First Fit -periaate pitää huolen, että mahdollisesti yhtä lukuunottamatta kaikki laatikot tulevat enemmän kuin puolilleen. Siis

$$\sum_{i=1}^n a_i > \frac{1}{2} \cdot (m - 1).$$

Toisaalta

$$\text{OPT} \geq \sum_{i=1}^n a_i.$$

Siis

$$m < 2 \cdot \text{OPT} + 1,$$

joten $m \leq 2 \cdot \text{OPT}$. \square

Lause 2.28: Jos $P \neq NP$, niin laatikonpakkausongelmalla ei ole $(3/2 - \varepsilon)$ -approksimointialgoritmia millään $\varepsilon > 0$.

Todistus: Olkoon algoritmin A approksimointisuhde $(3/2 - \varepsilon)$. Muodostetaan polynominen algoritmi NP-täydelliselle PARTITION-ongelmalle:

Annettu: jono positiivisia lukuja b_1, \dots, b_n

Kysymys: voidaanko jono jakaa kahteen osajonoon, joiden kummankin summa on $\frac{1}{2} \sum_{i=1}^n b_i$

Muodostetaan PARTITION-tapauksesta b_1, \dots, b_n laatikonpakkaustapaus a_1, \dots, a_n asettamalla

$$a_i = \frac{2b_i}{\sum_{j=1}^n b_j}.$$

Siis laatikonpakkauksen optimikustannus on ainakin 2, ja yhtäsuurus pätee, jos ja vain jos PARTITION-ongelman vastaus on "kyllä".

Jos optimikustannus on 2, niin $(3/2 - \varepsilon)$ -approksimointialgoritmi löytää sen, sillä seuraava mahdollinen kokonaislukuratkaisu 3 olisi jo liian huono approksimaatio. \square

Edellinen negatiivinen tulos siis saatiin tarkastelemalla tapauksia, joissa OPT on hyvin pieni (2 tai 3). Seuraava tulos osoittaa, että laatikonpakkauksella on **asymptoottinen** PTAS. Toisin sanoen approksimointisuhde voidaan saada mielivaltaisen lähelle ykköstä, kun OPT kasvaa.

Lause 2.29: Millä tahansa kiinteällä $0 < \varepsilon < 1/2$ laatikonpakkauksongelmalle on polynominen algoritmi, jonka tuottaman ratkaisun kustannus on korkeintaan $(1 + 2\varepsilon) \cdot \text{OPT} + 1$.

Todistus koostuu kolmesta vaiheesta:

1. tarkka ratkaisu, kun jonossa a_1, \dots, a_n on vain vakiomäärä eri suurien arvoja
2. edellisen perusteella pyöristämiseen perustuva approksimatiivinen ratkaisu, kun mikään a_i ei ole kovin pieni
3. pienten esineiden liittäminen edelliseen ratkaisuun.

Syntyvä algoritmi ei ole kovin käytännöllinen, sillä aikavaativuus on korkea etenkin parametrin $1/\varepsilon$ suhteen.

Lause 2.30: Olkoon $\varepsilon > 0$ ja $K \in \mathbb{N}$ kiinnitetty. Tarkastellaan laatikonpakkausta rajoitettuna tapauksiin, joissa esineiden koot ovat ainakin ε ja erilaisia kokoja esiintyy korkeintaan K kappaletta. Tälle ongelmalle on tarkka polynominen ratkaisualgoritmi.

Todistus: Yhteen laatikkoon tulee korkeintaan $M = \lfloor 1/\varepsilon \rfloor$ esinettä. Erilaisia laatikkotyyppjä (laatikossa olevia esinekokojen yhdistelmiä) on korkeintaan

$$R = \binom{M + K}{M}.$$

Ajatellaan nimittäin laatikkotyyppiä, jossa on r_i kappaletta kokonumeroa i olevia esineitä, $i = 1, \dots, K$. Tällainen tyyppi voidaan esittää bittijonona

$$1^{n_1} 0 1^{n_2} 0 \dots 0 1^{n_K} 0 1^r,$$

missä $r = M - n_1 + \dots + n_K$. Siis laatikkotyyppjä on yhtä monta kuin $M + K$ bitin jonoja, joissa nolliä on tasan K .

Selvästi riittää rajoittua tarkastelemaan pakkauksia, joissa on korkeintaan n laatikkoa. Tällaisia voidaan muodostaa korkeintaan

$$P = \binom{n+R}{R} = O(n^R)$$

erilaista, kun otetaan huomioon ainoastaan eri laatikkotyyppien lukumäärät (ei laatikoiden järjestystä). Päättely on samantyylinen kuin lukuvulle R .

Koska oleellisesti erilaisia käyviä ratkaisuja on siis polynominen määrä, voidaan käydä nämä läpi raa'alla voimalla ja valita paras. \square

Huom. edellisen algoritmin polynominen aikavaativuus on erittäin korkea-asteinen, kun ε on pieni.

Lause 2.31: Olkoon $\varepsilon > 0$ kiinnitetty. Tarkastellaan laatikonpakkausta rajoitettuna tapauksiin, joissa kaikki koot ovat ainakin ε . Ongelmalla on $(1 + \varepsilon)$ -approksimointialgoritmi.

Todistus: Tarkastellaan laatikonpakkaustapausta I . Jos $n < 1/\varepsilon^2$, voidaan suoraan soveltaa edellistä algoritmia vakiolla $K = \lceil 1/\varepsilon^2 \rceil$. Oletetaan siis $n\varepsilon^2 \geq 1$.

Seuraavaksi esineet jaetaan Q esinettä käsittäviin ryhmiin, missä

$$Q = \lfloor n\varepsilon^2 \rfloor \geq n/(2\varepsilon^2).$$

Jos n/Q ei ole kokonaisluku, jätetään viimeinen ryhmä vajaaksi. Olkoon K ryhmien lukumäärä. Tälle saadaan yläraja $K \leq \lceil 2/\varepsilon^2 \rceil$.

Ryhmät jaetaan koon mukaan. Ryhmään 1 tulee pienimmät Q esinettä, ryhmään 2 seuraavaksi pienimmät Q esinettä jne. Ryhmäraja voi sattua sellaiseen kohtaan, että samankokoisia esineitä tulee kahteen eri ryhmään, mutta tämä ei haittaa.

Kun $k = 1, \dots, K$, olkoon p_k suurin koko, joka tulee ryhmään k . Muodostetaan laatikonpakkaustapaus J tapauksesta I asettamalla jokaisen ryhmään k tulleen esineen kooksi p_k . Siis kokoja pyöristetään ylöspäin siten, että saman ryhmän esineiden koot menevät päällekkäin.

Tapauksessa J on nyt vakiomäärä K eri kokoja. Edellisen lauseen algoritmilla sille saadaan optimaalinen ratkaisu. Olkoon $\text{OPT}(J)$ tämän ratkaisun kustannus. Koska painoja pyöristettiin ylöspäin, sama pakkaus on käypä ratkaisu myös ongelmaan I . Osoitamme, että $\text{OPT}(J) \leq (1 + \varepsilon) \cdot \text{OPT}(I)$. Siis tämä ongelman J ratkaisu on riittävän hyvä approksimaatio ongelmaan I .

Väitteen $\text{OPT}(J) \leq (1 + \varepsilon) \cdot \text{OPT}(I)$ todistamiseksi tarkastellaan tapausta J' , jossa ryhmään kuuluvien esineiden koko asetetaan samaksi kuin ryhmän **pienimmän** esineen koko. Koska kokoja nyt pyöristettiin alaspäin, $\text{OPT}(J') \leq \text{OPT}(I)$. Muodostamme tapauksen J' optimiratkaisusta s' tapauksen J ratkaisun s , jonka kustannus ei ole paljon suurempi.

Kun $1 \leq k \leq K - 1$, olkoon f_k bijektio ryhmästä k ryhmään $k + 1$. Koska ryhmän k suurin koko ei voi olla suurempi kuin ryhmän $k + 1$ pienin koko, esineen a koko tapauksessa J ei voi olla suurempi kuin esineen $f_k(a)$ koko tapauksessa J' . Voimme siis ratkaisussa s laittaa ryhmän k alkion a siihen laatikkoon, johon $f_k(a)$ meni ratkaisussa s' .

Siis $\text{OPT}(J')$ laatikkoa riittää tapauksen J ensimmäisille $K - 1$ ryhmälle. Viimeinen ryhmä mahtuu triviaalisti Q laatikkoon. Saadaan

$$\text{OPT}(J) \leq \text{OPT}(J') + Q \leq \text{OPT}(I) + Q.$$

Oletuksen mukaan kaikki koot ovat vähintään ε , joten $\text{OPT}(I) \geq n\varepsilon$. Siis $Q = \lfloor n\varepsilon^2 \rfloor \leq \varepsilon \cdot \text{OPT}(I)$, joten

$$\text{OPT}(J) \leq \text{OPT}(I) + Q \leq (1 + \varepsilon) \cdot \text{OPT}(I).$$

□

Viimeinen askel käsittelee alle ε olevat koot.

Lause 2.32: Olkoon I mielivaltainen laatikonpakkaustapaus ja I' saatu poistamalla siitä kaikki kooltaan alle ε olevat esineet, missä $0 < \varepsilon \leq 1/2$ on vakio. Olkoon s' tapauksen I' ratkaisu, jonka kustannus on korkeintaan $(1 + \varepsilon) \cdot \text{OPT}(I')$. Muodostetaan tapauksen I ratkaisu sijoittelemalla kooltaan alle ε olevat esineet First Fit -menetelmällä käyttäen alkutilanteena ratkaisua s' . Ratkaisun s kustannus on korkeintaan $(1 + 2\varepsilon) \cdot \text{OPT}(I) + 1$.

Huom. Tapauksen I' ratkaisu s' saadaan edellisen lauseen mukaan. Pienet esineet sijoitellaan mahdollisuuksien mukaan ratkaisussa s' syntyneisiin vajaisiin laatikoihin, ja uusia laatikoita ratkaisuun s otetaan vasta tarpeen vaatiessa.

Todistus: Jos kooltaan alle ε olevat esineet onnistutaan kaikki sijoittelemaan ratkaisun s' vajaisiin laatikoihin, ratkaisun s kustannus on sama kuin ratkaisun s' ja siis riittävän hyvä.

Oletetaan nyt, että ratkaisussa s tarvittiin ainakin yksi uusi laatikko ja kaikkiaan M laatikkoa. Vain viimeisessä laatikossa voi olla tyhjää tilaa ε tai enemmän. Siis laatikoissa olevien esineiden yhteenlaskettu koko on ainakin $(M - 1)(1 - \varepsilon)$.

Koska esineiden yhteenlaskettu koko on alaraja optimikustannukselle,

$$M \leq \frac{\text{OPT}(I)}{1 - \varepsilon} + 1 \leq (1 + 2\varepsilon)\text{OPT}(I) + 1,$$

missä on käytetty oletusta $\varepsilon \leq 1/2$. \square

Kootaan vielä algoritmi yhteen:

Algoritmi (laatikonpakkauksen asymptoottinen PTAS)

Syöte: tapaus I , tarkkuusparametri $0 < \varepsilon \leq 1/2$

1. Muodosta tapaus I' poistamalla tapauksesta I kooltaan alle ε olevat esineet.
2. Muodosta tapaus J pyöristämällä ylöspäin esineiden kokoja niin, että eri kokoja jää $K = O(1/\varepsilon^2)$ (kuten lauseen 2.31 todistuksessa).
3. Muodosta tapaukselle J tarkka ratkaisu s' (kuten lauseen 2.30 todistuksessa). Tämä on samalla approksimatiivinen ratkaisu tapaukselle I' .
4. Muodosta ratkaisusta s' ratkaisu s sijoittelemalla kooltaan alle ε olevat esineet First Fit -menetelmällä. Tulosta s .

3. Lineaarinen ohjelmointi approksimoinnissa

Monet kombinatoriset optimointiongelmat voidaan esittää lineaarisina kokonaislukuohjelmina. Lineaarinen ohjelmointi kokonaislukumuuttujilla on NP-kovaa, mutta ongelma voidaan **löysentää** (relax) sallimalla murtolukuratkaisut. Ilman kokonaislukurajoitteita lineaarinen ohjelmointiongelma tunnetusti voidaan ratkaista polynomisessa ajassa.

Tästä saadaan kaksi approksimointistrategiaa:

- etsitään löysennetylle ongelmalle ratkaisu ja **pyöristetään** mahdolliset murtoluvut kokonaisluvuiksi
- käytetään lineaarisen ohjelman **duaalia** tuottamaan approksimoinnissa tarvittavia alarajoja optimiratkaisun arvolle.

Lineaarinen ohjelmointi

(Lineaarisen ohjelmoinnin perusteita on käsitelty melko yksityiskohtaisesti kurssilla *Diskreetti optimointi*, joten ne esitetään tässä kertauksenomaisesti.)

Tavoitteena on löytää n muuttujalle x_1, \dots, x_n sellaiset (rationaaliluku)arvot, että lineaarinen funktio

$$\sum_{j=1}^n c_j x_j$$

minimoituu. Lisäksi vaaditaan, että muuttujien arvot ovat **ei-negatiivisia** ja toteuttavat joukon lineaarisia rajoitteita

$$\sum_{j=1}^n a_{ij} x_j \geq b_j, \quad i = 1, \dots, m.$$

Sanomme tätä lineaarisen ohjelman **standardimuodoksi** (terminologia vaihtelee, toisinaan tätä sanotaan kanoniseksi muodoksi). Esim. lineaarisia yhtäsuuruusrajoitteita ja negatiiviset arvot sallivia muuttujia sisältävät ongelmat voidaan helposti muuntaa tähän muotoon.

Esitämme tämän yleensä matriisimuodossa

$$\begin{array}{ll} \text{minimoi} & \mathbf{c}^\top \mathbf{x} \\ \text{ehdolla} & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{array}$$

missä vektorit $\mathbf{c} \in \mathbb{Q}_+^n$ ja $\mathbf{b} \in \mathbb{Q}_+^m$ ja matriisi $A \in \mathbb{Q}_+^{m \times n}$ siis ovat ongelman syöte.

Ylläolevan standardimuotoisen minimointiongelman **duaaliongelma** on muuttujien y_1, \dots, y_m minimointiongelma

$$\begin{array}{ll} \text{maksimoi} & \mathbf{b}^\top \mathbf{y} \\ \text{ehdolla} & \mathbf{A}^\top \mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{array}$$

Siis duaaliongelmassa on yksi muuttuja kutakin alkuperäisen **primaaliongelman** rajoitetta kohti ja yksi rajoite kutakin primaaliongelman muuttujaa kohti.

Jos duaaliongelmalle muodostetaan duaaliongelma samalla periaatteella, saadaan takaisin alkuperäinen primaaliongelma.

Duaaliongelman tärkeys ilmenee seuraavasta:

Lause 3.1 (Lineaarisen ohjelmoinnin vahva duaalisuus):

Primaaliongelmalla on äärellinen optimi, jos ja vain jos duaaliongelmalla on. Jos tällöin x^* on primaaliongelman ja y^* duaaliongelman optimiratkaisu, niin

$$c^T x = b^T y.$$

Todistus: Helppo suunta on pian esitettävä heikko duaalisuus; vaikean suunnan todistus sivuutetaan. \square

Jos tarkastellaan lineaarisen ohjelmoinnin päätösversiota

Annettu: c , b , A , rationaaliluku α

Kysymys: onko lineaarisen minimointiongelman optimiarvo korkeintaan α

niin

- "kyllä"-sertifikaattina voidaan käyttää primaaliongelman optimiratkaisua
- "ei"-sertifikaattina voidaan käyttää duaaliongelman optimiratkaisua.

Siis ongelma kuuluu luokkaa $NP \cap co-NP$. (Tosin muuten tiedetään sen kuuluvan jopa luokkaan P .)

Lause 3.2 (Lineaarisen optimoinnin heikko duaalisuus): Jos $x \in \mathbb{Q}_+^n$ ja $y \in \mathbb{Q}_+^m$ ovat primaali- ja duaaliongelman käypiä ratkaisuja, niin

$$c^T x \geq b^T y.$$

Todistus: Koska y toteuttaa duaaliongelman rajoitteet ja $x_j \geq 0$, pätee

$$c_j x_j \geq \left(\sum_{i=1}^m a_{ij} y_i \right) x_j \quad \text{kaikilla } j = 1, \dots, n.$$

Koska x toteuttaa primaaliongelman rajoitteet ja $y_i \geq 0$, pätee

$$\left(\sum_{j=1}^n a_{ij} x_j \right) y_i \geq b_i y_i \quad \text{kaikilla } i = 1, \dots, m.$$

Laskemalla yhteen saadaan

$$\begin{aligned}\sum_{j=1}^n c_j x_j &\geq \sum_{j=1}^n \left(\sum_{i=1}^m a_{ij} y_i \right) x_j \\ &= \sum_{i=1}^m \left(\sum_{j=1}^n a_{ij} x_j \right) y_i \\ &\geq \sum_{i=1}^m b_i y_i.\end{aligned}$$

□

Jos edellisessä erityisesti x ja y ovat optimiratkaisuja, niin vahvan duaalisuuden takia kaikissa arvioissa pitää päteä yhtäsuuruus. Siis

Lause 3.3 (Komplementaarisuusehdot): Olkoon x ja y primaali- ja duaaliongelman käypiä ratkaisuja. Nyt x ja y ovat kumpikin optimaalisia, jos ja vain jos kumpikin seuraavista pätee:

- kaikilla $1 \leq i \leq m$: jos $y_i \neq 0$ niin $\sum_{j=1}^n a_{ij}x_j = b_i$
- kaikilla $1 \leq j \leq n$: jos $x_j \neq 0$ niin $\sum_{i=1}^m a_{ij}y_i = c_j$.

□

Siis duaalimuuttuja y_i voi optimiratkaisussa saada nolosta poikkeavan arvon vain, jos vastaavassa primaaliratkaisussa rajoite numero i on "aktiivinen".

Duaalisuus ja min-max-relaatiot

Tarkastelemme esimerkkinä maksimivuo-ongelmaa ja siihen liittyvää minimileikkausongelmaan.

Maksimivuo-ongelmassa syötteenä on vuoverkko eli

- suunnattu verkko $G = (V, E)$
- lähde $s \in V$ ja nielu $t \in V$
- kaarten kapasiteetit $c(e) \in \mathbb{Q}_+$ kaikilla $e \in E$.

Tehtävänä on maksimoida kokonaisvuo solmusta s solmuun t siten, että

- vuo kunkin kaaren e yli on korkeintaan $c(e)$
- kaikissa muissa solmuissa kuin s ja t solmusta lähtevä vuo on sama kuin solmuun tuleva vuo.

Olkoon (X, \bar{X}) vuoverkon solmujen ositus, jolla $s \in X$ ja $t \in \bar{X}$. Kaaret joukosta X joukkoon \bar{X} muodostavat vuoverkon **leikkauksen**. Sen **kapasiteetti** $c(X, \bar{X})$ on näiden kaarten yhteenlaskettu kapasiteetti.

Selvästi minkä tahansa leikkauksen kapasiteetti on yläraja maksimivuolle.

Jos sattuu olemaan niin, että kelvollisessa vuossa vuon arvo on sama kuin jonkin leikkauksen kapasiteetti, niin kyseinen vuo on maksimivuo ja kyseinen leikkaus minimileikkaus (kapasiteetiltaan pienin leikkaus).

Tunnetun maksimivuo-minimileikkauseen mukaan itse asiassa verkossa aina maksimivuo ja minimileikkauksen arvot ovat samat.

Esitetään asia lineaarisena ongelmana. Muuttuja f_{ij} esittää vuota solmusta i solmuun j .

Yksinkertaistamme tilannetta lisäämällä solmusta t takaisin solmuun s kaaren, jonka kapasiteetti on ääretön. Tällöin vuot lähteestä nieluun muuttuvat suljetuiksi kierroiksi. Etuna on, että

- kaikki solmut toteuttavat saman vuon säilymisehdon: tuleva kokonaisvuo on yhtä kuin lähtevä kokonaisvuo
- vuon säilymisehdon takaamiseksi riittää, että kaikissa solmuissa tuleva kokonaisvuo on **enintään** yhtä kuin lähtevä kokonaisvuo.

Nimittäin jos suljetussa verkossa missään solmussa ei katoa vuota, sitä ei voi myöskään missään syntyä.

Saamme lineaarisen maksimointiongelman

maksimoi	f_{ts}	
ehdolla	$f_{ij} \leq c(i, j)$	kaikilla $(i, j) \in E$
	$\sum_{j:(j,i) \in E} f_{ji} - \sum_{j:(i,j) \in E} f_{ij} \leq 0$	kaikilla $i \in V$
	$f_{ij} \geq 0$	kaikilla $(i, j) \in E$.

Duaaliin otamme muuttujan d_{ij} kaaren (i, j) kapasiteettirajoitelle ja muuttujan p_i solmun i säilymisrajoitteelle. Duaaliongelmaksi tulee

minimoi	$\sum_{(i,j) \in E} c(i, j)d_{ij}$	
ehdolla	$d_{ij} - p_i + p_j \geq 0$	kaikilla $(i, j) \in E$
	$p_s - p_t \geq 1$	
	$d_{ij} \geq 0$	kaikilla $(i, j) \in E$
	$p_i \geq 0$	kaikilla $i \in V$.

Duaaliongelman intuitiivinen tulkinta ei ole heti ilmeinen. Asian selvittämiseksi tehdään siitä kokonaislukuohjelma, jossa muuttujat on rajoitettu 0-1-arvoiksi:

$$\begin{array}{ll}
 \text{minimoi} & \sum_{(i,j) \in E} c(i,j)d_{ij} \\
 \text{ehdolla} & d_{ij} - p_i + p_j \geq 0 \quad \text{kaikilla } (i,j) \in E \\
 & p_s - p_t \geq 1 \\
 & d_{ij} \in \{0, 1\} \quad \text{kaikilla } (i,j) \in E \\
 & p_i \in \{0, 1\} \quad \text{kaikilla } i \in V.
 \end{array}$$

Duaaliongelman on tämän kokonaislukuongelman **löysennys** (relaxation), jossa sallitaan myös murtolukuratkaisut.

(Itse asiassa löysennykseen tulee myös ehdot $p_i \leq 1$ ja $d_{ij} \leq 1$ kaikille muuttujille. On kuitenkin helppo nähdä, että näiden rajoitteiden poistaminen ei muuta optimiratkaisua.)

Olkoon (d^*, p^*) kokonaislukuongelman optimiratkaisu.

Rajoitus 0-1-arvoihin ja ehto $p_s^* - p_t^* \geq 1$ implikoivat $p_s^* = 1$ ja $p_t^* = 0$.
Määritellään siis leikkaus, jossa $X = \{i \mid p_i^* = 1\}$ ja $\bar{X} = \{i \mid p_i^* = 0\}$.

Jos (i, j) on kaari joukosta X joukkoon \bar{X} , niin ehdosta $d_{ij}^* - p_i^* + p_j^* \geq 0$ seuraa $d_{ij}^* = 1$. Muilla kaarilla d_{ij}^* voi rajoitteiden puolesta olla 0 tai 1, mutta $d_{ij}^* = 0$ minimoi kohdefunktion.

Siis kohdefunktion arvo on leikkauksen (X, \bar{X}) kapasiteetti, ja koska kysymyksessä oli optimiratkaisu, tämä on minimileikkaus.

Siis duaalin kokonaislukuversio antaa minimileikkausongelman. Entä itse duaali, eli kokonaislukuongelman löysennys?

Murtolukuleikkaus (fractional cut) liittyy kaariin (i, j) sellaiset **etäisyydet** $d_{ij} \geq 0$, että millä tahansa polulla lähteestä s nieluun t näiden etäisyyksien summa on ainakin 1.

Duaalin minkä tahansa käyvän ratkaisun arvot d_{ij} antavat murtolukuleikkauksen. Tarkastellaan nimittäin polkua P solmusta s solmuun t . Laskemalla yhteen rajoitteet $d_{ij} - p_i + p_j \geq 0$ kaikilla polun P kaarilla saadaan

$$\sum_{(i,j) \in P} d_{ij} - p_s + p_t \geq 0,$$

sillä alku- ja loppusolmua lukuunottamatta jokainen solmu on yhtä monta kertaa lähtö- ja tulosolmuna. Koska $p_s - p_t \geq 1$, saamme $\sum_{(i,j) \in P} d_{ij} \geq 1$.

Sanomme murtolukuleikkauksen **kapasiteetiksi** arvoa

$$\sum_{(i,j) \in E} c(i,j) d_{ij}$$

eli vastaavaa duaaliongelman kohdefunktion arvoa. Koska kokonaislukuongelma on rajoittuneempi kuin sen löysennys, voisi kuvitella, että pienin murtolukuleikkaus voisi olla pienempi kuin pienin leikkaus. Niin ei kuitenkaan voi olla, sillä

- maksimivuo-minimileikkauseen mukaan pienin leikkaus on sama kuin maksimivuo
- vahvan dualisuuden takia myös pienin murtolukuleikkaus on sama kuin maksimivuo!

Selitys on maksimivuo-ongelmaa esittävän lineaarisen ohjelman erityisominaisuuksissa.

Lineaarisen ohjelman käypien ratkaisujen joukko muodostaa konveksin monitahokkaan (hypertasojen rajaaman alueen). Tiedetään, että optimiratkaisuksi on aina mahdollista valita jokin tämän monitahokkaan **kärkipiste** (olettaen että äärellinen optimi on olemassa).

Minimimurtolukuleikkauksen lineaarisella ohjelmalla voidaan osoittaa olevan se erityisominaisuus, että käypien ratkaisujen monitahokkaan kaikkien kärkipisteiden koordinaatit ovat **kokonaislukuarvoisia**. Vastaava ominaisuus on monella muullakin ongelmalla, joille pätee jokin min-max-relaatio. Sen sijaan NP-kovilla ongelmilla näin ei yleensä ole, joten löysennyksen ratkaiseminen ei anna tarkkaa ratkaisua kokonaislukuversiolle.

Approksimointi pyöristämällä

Monet NP-kovat ongelmat voidaan esittää lineaarisina kokonaislukuohjelmina. Kokonaislukuohjelman löysennöksellä on usein jokin luonnollinen tulkinta alkuperäisen NP-ongelman murtolukuversiona. Joka tapauksessa saadaan seuraava yleismenettely approksimointiin:

1. Muodosta ongelmasta kokonaislukuohjelma.
2. Ratkaise kokonaislukuohjelman **löysennös** tarkasti.
3. **Pyöristä** saadun ratkaisun muuttujien arvot kokonaisluvuiksi jollain sopivalla menettelyllä.
4. Osoita, että pyöristäminen ei huononna ratkaisua liiaksi.

Approksimointi primaali-duaaliskeemalla

Tässä ajatuksena on hyödyntää sitä, että duaalin käypä ratkaisu antaa alarajan primaalin optimiarvolle. Duaalin käypää ratkaisua ja primaalin kokonaislukuratkaisua rakennetaan samanaikaisesti jollain ongelman erikoispiirteitä hyödyntävällä algoritmilla. Saatuja duaali- ja primaaliratkaisua vertaamalla saadaan raja approksimointisuhteelle.

Joukkopeiteongelman ahne algoritmi on esimerkki tästä, kuten pian näemme.

Kummallakaan edellisistä menetelmistä ei selvästikään voi todistaa parempaa approksimointisuhdetta kuin **kokonaislukurako** (integrality gap)

$$\sup_I \frac{\text{OPT}(I)}{\text{OPT}_f(I)},$$

missä $\text{OPT}_f(I)$ on tapauksen I murtolukuversion optimikustannus. (Tässä oletetaan minimointiongelma.)

Minimileikkaus oli esimerkki ongelmasta, jonka kokonaislukurako on 1. Useilla NP-kovilla ongelmilla sekä pyöristysmenetelmällä että primaali-duaaliskeemalla on mahdollista saavuttaa kokonaislukurako.

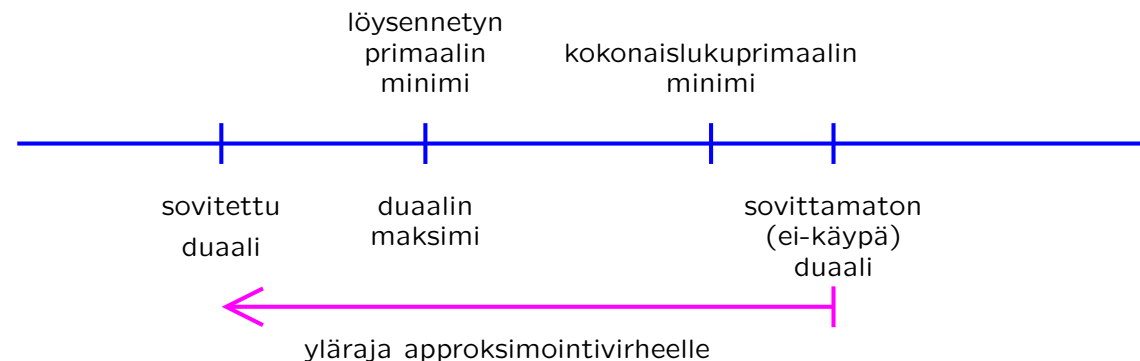
Tällöin menetelmien vertailussa huomio kiinnittyy algoritmien aikavaativuuteen. Lineaarinen ohjelma voidaan ratkaista polynomisessa ajassa, jos rajoitteiden määrä on polynominen tai on annettu polynomisaikainen **separointioraakkeli**, joka kertoo ei-käyvälle pisteelle jonkin rajoitteen, jota se rikkoo. Polynomisuudesta huolimatta tämä ei yleensä ole läheskään yhtä nopeaa kuin varta vasten laadittu esim. ahne algoritmi.

Primaali-duaaliskeeman tuottamat algoritmit ovat myös usein ymmärrettävämpiä, niitä on helppo virittää ja muunnella.

Joukkopeite duaalinsovituksella (dual fitting)

Ideana tässä tekniikassa on löytää (minimointiongelman) duaalille ratkaisu, joka **ei ole käypä** ja joka on **maksaa täysin** vastaavan kokonaislukuohjelman, ts. sen duaaliratkaisun arvo on **yläraja** kokonaislukuarvoisen primaaliohjelman arvolle.

Tätä ei-käypää duaaliratkaisua "sovitetaan" kertomalla se sopivalla vakiolla, jolloin se tulee käyväksi. Duaalisuuden takia sovitetun ratkaisun arvo on **alaraja** primaalin arvolle. Sovituksessa käytetty kerroin antaa ylärajan approksimointisuhteelle.



Olkoon siis joukkopeiteongelman perusjoukko U , osajoukkokokoelma \mathcal{S} ja osajoukon $S \in \mathcal{S}$ kustannus $c(S)$. Esitämme ratkaisun muuttujilla $x_S \in \{0, 1\}$, missä $x_S = 1$ jos S kuuluu valittuun joukkopeitteeseen.

Kokonaislukuohjelmaksi tulee

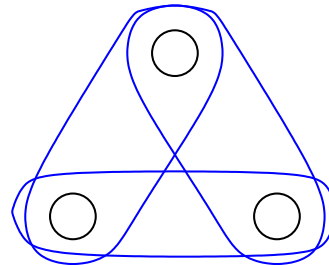
$$\begin{array}{ll} \text{minimoi} & \sum_{S \in \mathcal{S}} c(S)x_S \\ \text{ehdolla} & \sum_{S: e \in S} x_S \geq 1 \quad \text{kaikilla } e \in U \\ & x_S \in \{0, 1\} \quad \text{kaikilla } S \in \mathcal{S}. \end{array}$$

Löysennyksessä muuttujien arvorajoiksi tulee $0 \leq x_S \leq 1$, mutta ylärajat voidaan jättää pois muuttamatta optimia:

$$\begin{array}{ll} \text{minimoi} & \sum_{S \in \mathcal{S}} c(S)x_S \\ \text{ehdolla} & \sum_{S: e \in S} x_S \geq 1 \quad \text{kaikilla } e \in U \\ & x_S \geq 0 \quad \text{kaikilla } S \in \mathcal{S}. \end{array}$$

Löysennyksen arvo (pienin "murtopeite") voi olla aidosti pienempi kuin kokonaislukuohjelman (pienin joukkopeite).

Esimerkki 3.4: Olkoon $U = \{a, b, c\}$ ja $\mathcal{S} = \{\{a, b\}, \{a, c\}, \{b, c\}\}$. Jokaisen joukon kustannus on 1.



Optimaalinen kokonaislukuarvoinen peite saadaan valitsemalla mitkä tahansa kaksi joukkoa. Siis kustannukseksi tulee 2.

Paras murtopeite saadaan valitsemalla kaikki kolme joukkoa, kukin painolla $1/2$. Siis kustannukseksi tulee $3/2$. \square

Otamme duaalin muuttujiksi y_e vastaamaan alkioon e liittyvää rajoitetta:

$$\begin{array}{ll} \text{maksimoi} & \sum_{e \in U} y_e \\ \text{ehdolla} & \sum_{e: e \in S} y_e \leq c(S) \quad \text{kaikilla } S \in \mathcal{S} \\ & y_e \geq 0 \quad \text{kaikilla } e \in U. \end{array}$$

Intuitiivisesti pakkaamme alkioon e tavaramäärän y_e . Tavaramäärä yritetään maksimoida, mutta joukkoon S tuleva tavaramäärä saa olla korkeintaan $c(S)$.

Sekä primaalissa että duaalissa kohdefunktion kertoimet, rajoitematriisin alkiot ja rajoitteiden oikeat puolet ovat kaikki ei-negatiivisia. Nämä ehdot täyttävää minimointiongelmaa sanotaan **peitto-ongelmaksi** ja maksimointiongelmaa **pakkausongelmaksi**. Tällaiset peitto-pakkausparit ovat tärkeä erikoistapaus lineaarisista ohjelmista.

Esitämme nyt tutulle ahneelle algoritmille duaalinsovitusanalyysin. Algoritmi on siis seuraava (kalvolta 41):

1. $C \leftarrow \emptyset$

2. Toista kunnes $C = U$:

(a) Laske jokaisen osajoukon $S \in \mathcal{S}$ yksikkökustannus

$$u(S) = \frac{c(S)}{|S - C|}.$$

(b) Olkoon S_* jokin osajoukko, jonka yksikkökustannus on pienin.

(c) Valitse S_* mukaan joukkopeitteeseen.

(d) Kaikilla $e \in S_* - C$ aseta $\text{price}(e) = u(S_*)$.

3. Palauta peitteeseen valitut osajoukot.

Sovellamme duaalinsovitusmenetelmää seuraavasti:

1. Valitsemme duaalimuuttujille alustavat arvot $y'_e = \text{price}(e)$.
2. Toteamme, että ratkaisu y'_e maksaa täysin algoritmin tuottaman ratkaisun, eli sen kohdefunktion arvo on yläraja saadun joukkopeitteen (kokonaislukuongelman ratkaisun) kustannukselle.
3. Toteamme, että alustavista arvoista sovitettu ratkaisu $y_e = y'_e / H_n$ on duaalin käypä ratkaisu.
4. Siis algoritmin tuottaman joukkopeitteen kustannus on korkeintaan H_n kertaa optimikustannus.

Arvojen $y'_e = \text{price}(e)$ summa on selvästi sama kuin saadun joukkopeitteen kustannus, sillä valitun joukon kustannus jaetaan aina tasan sen peittämien (uusien) alkioiden hinnoiksi.

Analyysin oleellinen kohta on seuraava:

Lemma 3.5: Arvot $y_e = \text{price}(e)/H_k$ ovat käypä ratkaisu duaaliin.

Todistus: Tarkastellaan joukkoa S , missä $|S| = k$. Numeroidaan sen alkiot siinä järjestyksessä, kuin ne tulevat peitetyksi: e_1, \dots, e_k .

Tarkastellaan hetkeä, jolloin e_i tulee peitetyksi. Juuri tätä ennen S sisältää vielä ainakin $k - i + 1$ peittämätöntä alkiota. Siis $u(S) \leq c(S)/(k - i + 1)$. Koska algoritmi peitti alkion e_i mahdollisimman kustannustehokkaasti, $\text{price}(e_i) \leq c(S)/(k - i + 1)$. Laskemalla yhteen arvot $y_{e_i} = \text{price}(e_i)$ saadaan

$$\sum_{i=1}^k y_{e_i} \leq \frac{c(S)}{H_n} \cdot \left(\frac{1}{k} + \frac{1}{k-1} + \dots + \frac{1}{1} \right) = \frac{c(S)}{H_n} \cdot H_k \leq c(S).$$

□

Korollaari 3.6: Ahneen joukkopeitealgoritmin approksimointikerroin on korkeintaan H_n .

Todistus: Olkoon OPT optimijoukkopeitteen kustannus ja OPT_f optimaalisen murtopeitteen kustannus eli (löysennety) primaalin arvo.

Koska \mathbf{y} on duaalin käypä ratkaisu, dualisuusperiaatteesta seuraa $\sum_{e \in U} y_e \leq \text{OPT}_f$.

Saadun joukkopeitteen kustannus on siis

$$\sum_{e \in U} y'_e = H_n \cdot \sum_{e \in U} y_e \leq H_n \cdot \text{OPT}_f \leq H_n \cdot \text{OPT}.$$

□

Seuraava esimerkki osoittaa, että kokonaislukurako voi olla $\Omega(\log n)$, joten tällä alarajatekniikalla ei päästä sen parempiin approksimointisuhteisiin.

Esimerkki 3.7: Valitaan $n = 2^k - 1$ jollain $k \in \mathbb{N}$ ja $U = \{e_1, \dots, e_n\}$.

Joukkojen määrittelemiseksi tulkitaan ensin alkioiden indeksit i bittijonoiksi, joissa on k bittiä. Samastetaan nämä sitten k -ulotteisiin vektoreihin kerroinkuntana $\text{GF}(2)$ (kokonaisluvut modulo 2). Indeksejä i ja j vastaavien vektorien sisätulo $i \cdot j$ on siis 1, jos vastaavissa bittijonoissa on pariton määrä kohtia, joissa kummassakin jonossa on ykkönen.

Määritellään n joukkoa S_1, \dots, S_n siten, että $e_j \in S_i$ jos $i \cdot j = 1$. Jokaisen joukon kustannus on 1.

Jos i on annettu ja halutaan j , jolla $i \cdot j = 1$, voidaan vektorin j ensimmäiset $k - 1$ komponenttia valita mielivaltaisesti ja sitten valita viimeinen niin, että ehto täyttyy. Jokaisessa joukossa on siis $2^{k-1} = (n + 1)/2$ alkiota, ja jokainen alkio kuuluu $(n + 1)/2$ joukkoon. Optimaalinen murtopeite siis valitsee jokaisen joukon painolla $2/(n + 1)$. Kustannukseksi tulee $2n/(n + 1) = O(1)$.

Tarkastellaan nyt mielivaltaista p osajoukon yhdistettä, missä $p < k$. Osoitamme, että tämä ei voi peittää koko perusjoukkoa, joten optimaalisen kokonaislukuratkaisun arvo on ainakin $k = \Omega(\log n)$.

Olkoon näitä p joukkoa vastaavat vektorit i_1, \dots, i_p . Koska avaruus on k -ulotteinen ja $p < k$, voimme löytää vektorin j , joka on kohtisuorassa kaikkia näitä vektoreita kohtaan eli $j \cdot i_q = 0$. Vastaava e_j ei kuulu mihinkään joukoista S_{i_1}, \dots, S_{i_p} . \square

Joukkopeiteongelman yleistyksiä

Edellä esitetyn tapainen analyysi soveltuu mm. seuraaviin joukkopeiteongelman yleistyksiin:

Joukkomonipeite: Jokaiselle alkiolle e on lisäksi annettu **peittovaatimus** r_e , joka kertoo, **montako kertaa** se on peitettävä. Saman joukon saa sisällyttää peitteeseen useita kertoja, jolloin myös sen kustannus kerrotaan vastaavasti.

Monijoukkomonipeite: Monipeite muodostetaan **monijoukoista** eikä joukoista; siis yksi $S \in \mathcal{S}$ voi peittää alkion e useita kertoja. Käytetään tästä alkion e lukumäärästä monijoukossa S merkintää $M(S, e)$. Oletamme $M(S, e) \leq r_e$ kaikilla S ja e .

Kokonaislukupeiteongelmat: Yleinen muoto on

$$\begin{array}{ll} \text{minimoi} & \mathbf{c}^\top \mathbf{x} \\ \text{ehdolla} & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \in \mathbb{N}^n \end{array}$$

missä A , c ja b sisältävät vain ei-negatiivisia arvoja.

Rajoitettu joukkomonipeite

Analysoimme duaalinsovitusmenetelmällä joukkomonipeiteongelman version, jossa samaa joukkoa saa käyttää vain kerran. Olkoon siis r_e alkion e peittovaatimus ja x_S indikaattorimuuttuja sille, onko joukko S valittu peitteeseen:

$$\begin{array}{lll} \text{minimoi} & \sum_{S \in \mathcal{S}} c(S)x_S & \\ \text{ehdolla} & \sum_{S: e \in S} x_S \geq r_e & \text{kaikilla } e \in U \\ & x_S \in \{0, 1\} & \text{kaikilla } S \in \mathcal{S}. \end{array}$$

Tällä kertaa löysennyksessä ehtoja $x_s \leq 1$ ei voi jättää pois turhina. Ne ovat ainoa este saman joukon valitsemiselle useita kertoja. Siis löysennykseen tulee perusjoukkopeitteeseen verrattuna $|S|$ uutta ehtoa:

minimoi	$\sum_{S \in \mathcal{S}} c(S)x_S$	
ehdolla	$\sum_{S:e \in S} x_S \geq r_e$	kaikilla $e \in U$
	$-x_S \geq -1$	kaikilla $S \in \mathcal{S}$
	$x_S \geq 0$	kaikilla $S \in \mathcal{S}$.

Duaaliin tulee vastaavasti $|S|$ uutta muuttujaa z_S :

$$\begin{array}{ll} \text{maksimoi} & \sum_{e \in U} r_e y_e - \sum_{S \in \mathcal{S}} z_S \\ \text{ehdolla} & \sum_{e: e \in S} y_e - z_S \leq c(S) \quad \text{kaikilla } S \in \mathcal{S} \\ & y_e \geq 0 \quad \text{kaikilla } e \in U \\ & z_S \geq 0 \quad \text{kaikilla } S \in \mathcal{S}. \end{array}$$

Intuitiivisesti z_S kertoo, paljonko joukkoa S on ylipakattu. Ylipakkaaminen pienentää kohdefunktiota, mutta kertoimien r_e takia voi silti olla edullista.

Jos r_e on suurempi kuin alkion e sisältävien joukkojen $S \ni e$ lukumäärä, niin alkioita e kannattaa ylipakata rajatta. Tällöin duaalin optimiarvo on ääretön ja primaalilla ei ole käypiä ratkaisuja.

Algoritmi on samanlainen ahne algoritmi kuin joukkopeitteen perusversiolle. Olkoon jollain ajanhetkellä $C \subseteq U$ täysin peitettyjen alkioiden joukko. Siis $e \in C$, jos peitteeseen on jo valittu r_e joukkoa S , joilla $e \in S$. Kun $C = U$, algoritmi pysähtyy.

Joukon $S \in \mathcal{S}$ yksikkökustannukseksi määritellään

$$\frac{c(S)}{|S - C|}.$$

Kullakin iteraatiolla peitteeseen liitetään yksikkökustannukseltaan pienin joukko S .

Kun joukko S on juuri liitetty peitteeseen, jaetaan sen kustannus tasan peitetyille alkiolle asetetaan kaikilla $e \in S - C$

$$\text{price}(e, j_e) = \frac{c(S)}{|S - C|},$$

missä $j_e \leq r_e$ kertoo, monettako kertaa e tuli nyt peitetyksi.

Muodostetaan nyt duaaliongelman alustava (siis ei välttämättä käypä) ratkaisu asettamalla

$$\begin{aligned}y'_e &= \text{price}(e, r_e) \\z'_S &= \sum_{e \in S-C} (\text{price}(e, r_e) - \text{price}(e, j_e)),\end{aligned}$$

missä C ja j_e vastaavat tilannetta kun S valittiin peitteeseen. Jos S ei kuulu peitteeseen, asetetaan $z'_S = 0$. Koska yksikkökustannukset kasvavat algoritmi edetessä, pätee $\text{price}(e, j_e) \leq \text{price}(e, r_e)$, joten $z'_S \geq 0$.

Lemma 3.8: Duaaliratkaisu $(\mathbf{y}', \mathbf{z}')$ maksaa täysin algoritmin tuottaman joukkopeitteen.

Todistus: Joukkopeitteen kustannus on

$$\sum_{e \in U} \sum_{j=1}^{r_e} \text{price}(e, j).$$

Pitää siis osoittaa, että duaalin kohdefunktio saa tämän arvon.
Kohdefunktion arvoksi tulee

$$\begin{aligned} \sum_{e \in U} r_e y'_e - \sum_{S \in \mathcal{S}} z'_S &= \sum_{e \in U} r_e \text{price}(e, r_e) - \sum_{S \in \mathcal{S}} \left(\sum_{e \in S-C} (\text{price}(e, r_e) - \text{price}(e, j_e)) \right) \\ &= \sum_{e \in U} \sum_{j=1}^{r_e} \text{price}(e, j). \end{aligned}$$

□

Arvot $y_e = y'_e/H_n$ ja $z_S = z'_S/H_n$ antavat käyvän ratkaisun duaaliongelmalle.

Todistus: Tarkastellaan joukkoa S ja numeroidaan sen alkiot e_1, \dots, e_k siinä järjestyksessä, kuin ne tulevat kokonaan peitetyiksi.

Ensimmäinen tapaus on, että S ei tullut valituksi peitteeseen. Kun alkio e_i tuli kokonaan peitetyksi, joukosta S oli peittämättä ainakin $k - i + 1$ alkioita. Valittu joukko oli ainakin yhtä hyvä kuin S , joten

$$\text{price}(e_i, r_{e_i}) \leq \frac{c(S)}{k - i + 1}.$$

Koska $z_S = 0$, saamme

$$\begin{aligned} \sum_{e \in S} y_e - z_S &= \frac{1}{H_n} \sum_{i=1}^k \text{price}(e_i, r_{e_i}) \\ &\leq \frac{c(S)}{H_n} \left(\frac{1}{k} + \frac{1}{k-1} + \dots + \frac{1}{1} \right) \\ &\leq c(S) \end{aligned}$$

eli rajoite toteutuu.

Tarkastellaan nyt tapausta, että S valitaan, kun siitä on jo peitetty $k' < k$ alkioita. Peitetyksi tulevalle alkioille $e \in \{e_{k'+1}, \dots, e_k\}$ olkoon taas j_e indeksi sille, monettako kertaa se nyt peittyy. Saadaan

$$\begin{aligned} \sum_{e \in S} y_e - z_S &= \frac{1}{H_n} \left(\sum_{i=1}^k \text{price}(e_i, r_{e_i}) - \sum_{i=k'+1}^k (\text{price}(e_i, r_{e_i}) - \text{price}(e_i, j_{e_i})) \right) \\ &= \frac{1}{H_n} \left(\sum_{i=1}^{k'} \text{price}(e_i, r_{e_i}) + \sum_{i=k'+1}^k \text{price}(e_i, j_{e_i}) \right) \\ &= \frac{1}{H_n} \left(\sum_{i=1}^{k'} \text{price}(e_i, r_{e_i}) + c(S) \right). \end{aligned}$$

Arvoilla $1 \leq i \leq k$ kun e_i tuli lopullisesti peitetyksi, S ei vielä ollut valittu, joten taas $\text{price}(e_i, r_{e_i}) \leq c(S)/(k - i + 1)$. Sijoittamalla tämä edelliseen saadaan

$$\sum_{e \in S} y_e - z_S \leq \frac{c(S)}{H_n} \left(\sum_{i=1}^{k'} \frac{1}{k - i + 1} + 1 \right) \leq c(S). \quad \square$$

Lause 3.9: Ahne algoritmi antaa H_n -approksimaation rajoitetulle joukkomonipeiteongelmalle.

Todistus: Olkoon OPT optimikustannus. Edellisten lemموjen mukaan ahneen ratkaisun kustannus on

$$\sum_{e \in U} r_e y'_e - \sum_{S \in \mathcal{S}} z'_S = H_n \cdot \left(\sum_{e \in U} r_e y_e - \sum_{S \in \mathcal{S}} z_S \right).$$

Koska (\mathbf{y}, \mathbf{z}) on duaalin käypä ratkaisu, sen arvo on primaalin alaraja. Siis

$$H_n \cdot \left(\sum_{e \in U} r_e y_e - \sum_{S \in \mathcal{S}} z_S \right) \leq H_n \cdot \text{OPT}.$$

□

Edellisestä seuraa erityisesti, että ongelman kokonaislukurako on korkeintaan H_n . Tämä itse asiassa ei enää olisi tilanne, jos tarkastelisimme rajoitettua **monijoukkomonipeitettä**. "Rajoitettu" siis tarkoitti, että kutakin (moni)joukkoa saa käyttää vain kerran.

Joukkopeite pyöristämällä

Muistetaan, että alkion e frekvenssi on niiden joukkojen S lukumäärä, joilla $e \in S$. Olkoon taas f frekvenssien maksimi.

Seuraava yksinkertainen algoritmi saavuttaa approksimointisuhteen f .

Algoritmi (joukkopeitteen yksinkertainen pyöristys)

1. Muodosta joukkopeiteongelmasta kokonaislukuohjelma ja sille edelleen löysennös.
2. Ratkaise löysennös tarkasti. Olkoon ratkaisuna muuttujat $(x'_S)_{S \in \mathcal{S}}$.
3. Aseta $x_S = 1$ jos $x'_S \geq 1/f$ ja $x_S = 0$ muuten. Palauta ratkaisu $(x_S)_{S \in \mathcal{S}}$.

Lause 3.10: Edellinen algoritmi saavuttaa approksimointisuhteen f .

Todistus: Koska $x_S \leq f x'_S$ kaikilla S , ratkaisun kustannus pyörityksessä kasvaa korkeintaan kertoimella f .

Tarkastellaan mielivaltaista alkiota e . Ratkaisu (x'_S) on käypä, joten $\sum_{S:e \in S} x'_S \geq 1$. Alkio e kuuluu korkeintaan f joukkoon, joten $x'_S \geq 1/f$ ainakin yhdellä S , jolla $e \in S$. Tämä S tulee myös joukkopeitteeseen pyörityksen tuloksena. \square

Sama approksimointisuhde saavutettiin kerrostamalla (s. 45; harjoitus 2.3).

Esimerkki 3.11: Muodostetaan edelliselle tiukka esimerkki.

Perusjoukoksi valitaan $U = \{1, \dots, n\}^k$ joillain n ja k . Siis $|U| = n^k$.

Muodostamme nk joukkoa $S_{i,j}$, $i = 1, \dots, n$, $j = 1, \dots, k$. Alkio $(x_1, \dots, x_k) \in U$ kuuluu joukkoon $S_{i,j}$, jos $x_j = i$.

Siis jokainen alkio kuuluu k joukkoon, joten $f = k$. Jokaisessa joukossa on n^{k-1} alkioita.

Optimiratkaisu löysennökseen on $x_S = 1/k$ kaikilla S . Tämän pyöristäminen valitsee kaikki joukot, kustannuksena nk .

Optimaalinen kokonaislukuratkaisu saadaan valitsemalla esim. $x_S = 1$ kun $S = S_{1,2}, S_{2,1}, \dots, S_{n,1}$. Kustannukseksi tulee n . \square

Joukkopeite satunnaisella pyöristämisellä

Luonnollinen tulkinta murtolukuarvolle $0 \leq x'_S \leq 1$ on, että se esittää joukon S todennäköisyyttä kuulua peitteeseen.

Tarkastelemme siis satunnaista pyöristystä, jossa jokaisen S kohdalla heitetään painotettua lanttia, ja $x_S = 1$ todennäköisyydellä x'_S .

Algoritmin perusidea on tehdä $O(\log n)$ tällaista pyöristystä toisistaan riippumatta, ja valita S peitteeseen jos x_S sai arvon 1 edes kerran. Tällöin peitteeseen tulevien joukkojen määrän odotusarvo on $O(\log n) \cdot \text{OPT}_f$, missä OPT_f on löysennöksen optimikustannus.

Tarkemmin algoritmi on seuraava:

Apualgoritmi (joukkopeite pyöristyksiä yhdistämällä)

1. Muodosta kokonaislukuohjelma ja sen löysennös. Ratkaise löysennös tarkasti; olkoon ratkaisu $x = p$.
2. Muodosta $2 \lceil \ln n \rceil$ kokonaislukuohjelman ratkaisua $C_1, \dots, C_{2 \lceil \ln n \rceil}$ siten, että muista valinnoista riippumatta joukko S tulee peitteeseen C_i aina todennäköisyydellä p_S .
3. Palauta $C' = C_1 \cup \dots \cup C_{2 \lceil \ln n \rceil}$.

Lause 3.12: Kun $n \geq 4$, niin edellisen apualgoritmin palauttama C' on ainakin todennäköisyydellä $1/2$ joukkopeite, jonka kustannus on korkeintaan $2 \lceil \ln n \rceil \text{OPT}$.

Ennen todistusta todetaan, että vakiotodennäköisyys riittää siihen, että algoritmia iteroimalla saadaan hyviä tuloksia.

Korollaari 3.13: Olkoon $\delta > 0$. Toistetaan edellistä apualgoritmia $\lceil \log_2(1/\delta) \rceil$ kertaa. Saatujen kokoelmien C' joukossa on ainakin todennäköisyydellä $1 - \delta$ ainakin yksi kustannukseltaan korkeintaan $2 \lceil \ln n \rceil \text{OPT}$ oleva joukkopeite.

Todistus: Todistus, että $\lceil \log_2(1/\delta) \rceil$ epäonnistutaan saamaan halutunlainen C' , on korkeintaan

$$\left(\frac{1}{2}\right)^{\log_2(1/\delta)} = \delta. \quad \square$$

Korollaari 3.14: Toistetaan apualgoritmia, kunnes saadaan korkeintaan kokoa $2 \lceil \ln n \rceil \text{OPT}$ oleva joukkopeite C' . Tarvittavien toistojen lukumäärän odotusarvo on korkeintaan 2.

Todistus: Tarvittavien toistojen lukumäärä on geometrisesti jakautunut parametrillä $p \geq 1/2$. \square

Lauseen 3.12 todistus: Johdamme ensin vakioalarajan mielivaltaisen alkion $a \in U$ todennäköisyydelle tulla annetun pyörityksen C_i peittämäksi. Siis kun pyörityksiä toistetaan, alkion todennäköisyys pysyä peittämättömänä menee nopeasti nolleen. Tämän takia C' on joukkopeite ainakin todennäköisyydellä $1/4$.

Sen jälkeen arvioimme joukon C' kustannusta.

Todistuksen kummassakin osassa perustana on havainto, että kaikilla $i = 1, \dots, 2 \lceil \ln n \rceil$ pätee

$$c(C_i) := \mathbf{E} \left[\sum_{S \in C_i} c(S) \right] = \sum_{S \in \mathcal{S}} p_S c(S) = \mathbf{OPT}_f.$$

Tarkastellaan nyt alkiota $a \in U$, joka kuuluu k joukkoon. Olkoot näiden todennäköisyydet tulla valituksi p_1, \dots, p_k . Koska \mathbf{p} antaa murtolukupitteen,

$$\sum_{i=1}^k p_i \geq 1.$$

Todennäköisyys, että a ei tule tietyn kokoelman C_i peittämäksi, on

$$\prod_{i=1}^k (1 - p_i) \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e},$$

sillä pidettäessä lukujen $1 - p_i$ summa vakiona niiden tulo saa maksimiarvon, kun kaikki luvut ovat yhtä suuria.

Suoritetaan nyt $2 \lceil \ln n \rceil$ riippumatonta toistoa jollain vakiolla d .

Todennäköisyys, että a ei tule kertaakaan peitettyksi, on korkeintaan

$$\left(\frac{1}{e}\right)^{2 \ln n} = \frac{1}{n^2}.$$

Todennäköisyys, että n alkiosta **jokin** jää peittämättä, on korkeintaan $n \cdot (1/n^2) = 1/n \leq 1/4$.

Koska $\mathbf{E}[c(C_i)] = \text{OPT}_f$, niin $\mathbf{E}[c(C')] \leq 2(\lceil \ln n \rceil) \text{OPT}_f$.

Markovin epäyhtälön mukaan ei-negatiivisia arvoja saavalle satunnaismuuttujalle X pätee

$$\Pr[X \geq t] \leq \frac{\mathbf{E}[X]}{t}$$

kaikilla $t > 0$. Siis

$$\Pr[c(C') \geq 8 \lceil \ln n \rceil \text{OPT}_f] \leq \frac{1}{4}.$$

Todennäköisyys, että C' epäonnistuu ainakin toisen ehdon toteuttamisessa, on korkeintaan $1/4 + 1/4 = 1/2$. \square

Solmupeite ja puolikokonaislukuratkaisut

Jos lineaarisen ohjelman ratkaisussa muuttujien arvot ovat puolikokonaislukuja $x \in \{0, 1/2, 1, 3/2, 2, 5/2, \dots\}$, pyöristäminen huonontaa kohdefunktion arvoa korkeintaan kertoimella 2. Lisäksi jos kysymyksessä on pakkaus- tai peitto-ongelma, niin pyöristäessä vastaavasti alas- tai ylöspäin ratkaisu pysyy käypänä.

Lineaarisen ohjelman optimiarvo saavutetaan aina jossain käyvän alueen kärkipisteessä (ja mahdollisesti muissakin pisteissä). Sanomme näitä kärkipisteissä sijaitsevia käypiä ratkaisuja **kantaratkaisuiksi**. Kantaratkaisuja ovat tasan ne käyvät ratkaisut, joita ei voi esittää kahden käyvän ratkaisun x' ja x'' aitona konveksina kombinaationa eli muodossa $\lambda x' + (1 - \lambda)x''$, missä $0 < \lambda < 1$ (ks. *Diskreetti optimointi*).

Aiemmin totesimme, että minimileikkausongelman löysennöksen kantaratkaisut olivat kokonaislukuarvoisia. Monille NP-koville ongelmilla voidaan heikommin osoittaa, että ne ovat puolikokonaislukuarvoisia. Tarkastelemme esimerkkinä solmupeiteongelmaa, joka siis on joukkopeitteen erikoistapaus $f = 2$.

Solmupainoilla $c: V \rightarrow \mathbb{Q}_+$ varustetun verkon (V, E) solmupeiteongelman kokonaislukuohjelma on

$$\begin{array}{ll} \text{minimoi} & \sum_{v \in V} c(v)x_v \\ \text{ehdolla} & x_u + x_v \geq 1 \quad \text{kaikilla } (u, v) \in E \\ & x_v \in \{0, 1\} \quad \text{kaikilla } v \in V. \end{array}$$

Löysennyksessä jälleen muuttujien ylärajat voidaan taas jättää pois vaikuttamatta optimiin:

$$\begin{array}{ll} \text{minimoi} & \sum_{v \in V} c(v)x_v \\ \text{ehdolla} & x_u + x_v \geq 1 \quad \text{kaikilla } (u, v) \in E \\ & x_v \geq 0 \quad \text{kaikilla } v \in V. \end{array}$$

Lause 3.15: Painotetun solmupeiteongelman lineaarisen löysennöksen kantaratkaisut ovat puolikokonaislukuarvoisia.

Huom. Mille tahansa käyvälle ratkaisulle on helppo löytää ainakin yhtä hyvä kantaratkaisu (ja esim. simplex tuottaa nimenomaan kantaratkaisuja). Etsimällä löysennökselle optimaalinen kantaratkaisu ja pyöristämällä ylöspäin saadaan siis 2-approksimaatio solmupeiteongelmalle.

Todistus: Olkoon x käypä ratkaisu, jonka kaikki muuttujien arvot eivät ole joukossa $\{0, 1/2, 1\}$. Määritellään

$$V_- = \{v \mid 0 < x_v < 1/2\} \quad \text{ja} \quad V_+ = \{v \mid 1/2 < x_v < 1\}.$$

Oletuksen mukaan ainakin toinen joukoista on epätyhjä.

Kun $\varepsilon > 0$, määritellään ratkaisut \mathbf{y} ja \mathbf{z} seuraavasti:

$$\mathbf{y}_v = \begin{cases} x_v - \varepsilon & \text{kun } v \in V_- \\ x_v + \varepsilon & \text{kun } v \in V_+ \\ x_v & \text{muuten} \end{cases} \quad \text{ja} \quad \mathbf{z}_v = \begin{cases} x_v + \varepsilon & \text{kun } v \in V_- \\ x_v - \varepsilon & \text{kun } v \in V_+ \\ x_v & \text{muuten} \end{cases}$$

Koska $V_- \cup V_+$ on epätyhjä, ratkaisut \mathbf{x} , \mathbf{y} ja \mathbf{z} ovat kaikki erillisiä. Siis $\mathbf{x} = \frac{1}{2}\mathbf{y} + \frac{1}{2}\mathbf{z}$ on ratkaisujen \mathbf{y} ja \mathbf{z} aito konvekssi kombinaatio. Riittää osoittaa, että riittävän pienillä ε ratkaisut \mathbf{y} ja \mathbf{z} ovat käyviä, jolloin siis \mathbf{x} ei ole kantaratkaisu.

Jos $x_v = 0$, niin $y_v = z_v = x_v$. On siis helppo valita niin pieni ε , että y ja z ovat ei-negatiivisia.

Edelleen on helppo valita ε niin pieneksi, että jos $x_u + x_v < 1$, niin myös y ja z toteuttavat tämän rajoitteen.

Lopulta jäljellä on yhtäsuuruutena pätevät rajoitteet $x_u + x_v = 1$. Tässä tapauksessa on kolme mahdollisuutta:

- $x_u = x_v = 1/2$: nyt $y_u = z_u = x_u$ ja $y_v = z_v = x_v$
- $x_u = 0$ ja $x_v = 1$: taas $y_u = z_u = x_u$ ja $y_v = z_v = x_v$
- $u \in V_-$ ja $v \in V_+$: nyt $y_u + y_v = x_u - \varepsilon + x_v + \varepsilon = 1$ ja $z_u + z_v = x_u + \varepsilon + x_v - \varepsilon = 1$.

Siis y ja z toteuttavat kaikki rajoitteet. \square

Joukkopeite primaali-duaaliskeemalla

Tässä menetelmässä rakennetaan rinnakkain käypä ratkaisu sekä primaalille että duaalille. Komplementaarisuusehdot (s. 123) ovat keskeisessä osassa. Tämän tyyppiset konstruktioit ovat tärkeitä myös tarkoissa algoritmeissa. Nyt olemme kiinnostuneita approksimaatioalgoritmeista ongelmille, joilla kokonaislukuongelman ja murtolukuongelman optimien välillä on rako. Siksi käytämme komplementaarisuusehtojen approksimatiivista versiota.

Esitämme ensin [approksimatiivisen komplementaarisuusehdon](#) yleisessä muodossa. Sitten esitämme algoritmisen periaatteen käyttäen taas joukkopeitettä esimerkkinä. Tuloksena saadaan jälleen f -approksimaatioalgoritmi, missä f on alkioiden maksimifrekvenssi.

Lähdemme siis muuttujien x_1, \dots, x_n primaaliongelmasta

$$\begin{array}{ll} \text{minimoi} & \mathbf{c}^\top \mathbf{x} \\ \text{ehdolla} & \mathbf{Ax} \geq \mathbf{b} \\ & \mathbf{x} \geq 0, \end{array}$$

ja sen muuttujat y_1, \dots, y_m sisältävästä duaalista

$$\begin{array}{ll} \text{maksimoi} & \mathbf{b}^\top \mathbf{y} \\ \text{ehdolla} & \mathbf{A}^\top \mathbf{y} \leq \mathbf{c} \\ & \mathbf{y} \geq 0. \end{array}$$

Jatkossa tilanne tulee tietysti taas olemaan se, että primaali on jotakin kombinatorista ongelmaa vastaavan kokonaislukuohjelman löysennös.

Lemma 3.16: Olkoon $\alpha \geq 1$ ja $\beta \geq 1$ vakioita ja x ja y primaalin ja duaalin käypiä ratkaisuja, jotka toteuttavat **approksimatiiviset komplementaarisuusehdot**:

Primaaliehdot: kaikilla $1 \leq j \leq n$ jos $x_j \neq 0$ niin $c_j/\alpha \leq \sum_{i=1}^m a_{ij}y_i \leq c_j$.

Duaaliehdot: kaikilla $1 \leq i \leq m$ jos $y_i \neq 0$ niin $b_i \leq \sum_{j=1}^n a_{ij}x_j \leq \beta b_i$.

Tällöin

$$\sum_{j=1}^n c_j x_j \leq \alpha \beta \sum_{i=1}^m b_i y_i.$$

Huom. jatkossa joko primaali- tai duaaliehdot pätevät tarkasti, ja vastaavasti valitsemme $\alpha = 1$ tai $\beta = 1$.

Todistus: Annamme duaalimuuttujalle y_i rahamäärän $\alpha\beta b_i y_i$. Siis kokonaisrahamäärä $\alpha\beta \sum_{i=1}^m b_i y_i$ on todistettavana olevan epäyhtälön oikea puoli.

Duaalimuuttuja y_i antaa nyt edelleen primaalimuuttujalle x_j rahamäärän $\alpha y_i a_{ij} x_j$. Kaikkiaan y_i siis tarvitsee rahamäärän

$$\alpha y_i \sum_{j=1}^n a_{ij} x_j \leq \alpha\beta b_i y_i,$$

eli duaalimuuttujien rahat riittävät tähän.

Primaalimuuttuja x_j saama kokonaisrahamäärä on

$$\alpha x_j \sum_{i=1}^m a_{ij} y_i \geq c_j x_j.$$

Siis alkuperäinen rahamäärä kattaa kustannuksen $\sum_{j=1}^n c_j x_j$. \square

Algoritmi lähtee liikkeelle antamalla primaalimuuttujille alkuarvoiksi esim. $x = 0$ ja $y = 0$. Näitä ratkaisuja ruvetaan rajoite kerrallaan parantamaan, niin että x tulee "käyvämmäksi" ja y "optimaalisemmaksi".

Toinen puoli approksimatiivisista komplementaarisuusehdoista antaa **invariantin**, joka pidetään koko ajan voimassa. Toinen puoli antaa algoritmin **lopetusehdon**. Kun algoritmi päättyy, approksimatiiviset komplementaarisuusehdot ovat kokonaan voimassa. Tällöin y antaa alarajan OPT:lle, ja edellisen lemmän mukaan x on kertoimen $\alpha\beta$ sisällä optimista.

Primaali x pidetään koko ajan kokonaislukuarvoisena, joten saamme kokonaislukuratkaisun (ja samalla ylärajan ongelman kokonaislukurajalle).

Teemme tämän nyt käytännössä joukkopeiteongelmalle. Muistetaan, että primaali on

minimoi	$\sum_{S \in \mathcal{S}} c(S)x_S$	
ehdolla	$\sum_{S: e \in S} x_S \geq 1$	kaikilla $e \in U$
	$x_S \geq 0$	kaikilla $S \in \mathcal{S}$.

ja duaali

maksimoi	$\sum_{e \in U} y_e$	
ehdolla	$\sum_{e: e \in S} y_e \leq c(S)$	kaikilla $S \in \mathcal{S}$
	$y_e \geq 0$	kaikilla $e \in U$.

Pidämme koko ajan voimassa primaaliehtot, jotka tässä pätevät tarkasti eli $\alpha = 1$. Siis

kaikilla $S \in \mathcal{S}$ jos $x_S \neq 0$ niin $\sum_{e:e \in S} y_e = c(S)$.

Tavoitteena olevan approksimaatiosuhteen mukaisesti valitsemme $\beta = f$.
Duaaliehtoksi, jonka pidämme koko ajan voimassa, tulee

kaikilla $e \in U$ jos $y_e \neq 0$ niin $\sum_{S:e \in S} x_S \leq f$.

Kun muuttujien x_S arvojoukko on $\{0, 1\}$, tämä ehto itse asiassa pätee vakion f määritelmän takia automaattisesti.

Käypyysehto $\sum_{S:e \in S} x_S \geq 1$ tulee voimaan vasta, kun algoritmin suoritus päättyy.

Algoritmi (joukkopeite primaali-duaaliskeemalla)

1. Alusta $x \leftarrow 0$ ja $y \leftarrow 0$.
2. Valitse jokin peittämätön alkio e . Kasvata muuttujaa y_e , kunnes jollain S pätee

$$\sum_{e:e \in S} y_e = c(S).$$

Aseta $x_S \leftarrow 1$ kaikilla tällaisilla S ja merkitse niiden alkiot peitettyiksi.

3. Jos kaikki alkiot $e \in U$ on peitetty, palauta joukkopeite x . Muuten palaa kohtaan 2.

Askelessa 3 koska e on peittämätön, aluksi $x_S = 0$ kaikilla $S \ni e$. Samoin aluksi $y_e = 0$, sillä kun muuttujaa y_e ruvetaan kasvattamaan, e tulee samantien peitettyksi eikä siihen enää palata.

Siis primaaliehtoja "kaikilla $S \in \mathcal{S}$ jos $x_S \neq 0$ niin $\sum_{e:e \in S} y_e = c(S)$ " siirretään yksi kerrallaan vaihtoehdosta $x_S = 0$ vaihtoehtoon $\sum_{e:e \in S} y_e = c(S)$.

Lause 3.17: Edellinen algoritmi on f -approksimointialgoritmi.

Todistus: Edellä esitetyn perusteella ratkaisupari (x, y) on käypä ja toteuttaa approksimatiiviset komplementaarisuusehdot arvoilla $\alpha = 1$ ja $\beta = f$. Väite siis seuraa lemmasta 3.16. \square

Esimerkki 3.18: Tiukka esimerkki approksimointisuhteelle saadaan valitsemalla mielivaltaisella n ja $\varepsilon > 0$

- perusjoukkoon $n + 2$ alkiota a_1, \dots, a_n, b, c
- n joukkoa $\{a_i, b\}$, $i = 1, \dots, n$ kustannuksella 1
- lisäksi joukko $\{a_1, \dots, a_n, b, c\}$ kustannuksella $1 + \varepsilon$.

Siis $f = n + 1$, ja optimikustannus on $1 + \varepsilon$.

Jos algoritmi valitsee ensimmäisenä kasvatettavaksi muuttujan y_b , se kasvaa arvoon 1 ja ehto $\sum_{e:e \in S} y_e = c(S)$ tulee todeksi kaikilla $S = \{a_i, b\}$. Peitteeseen valitaan siis kaikki nämä n joukkoa. Tämän jälkeen joka tapauksessa valitaan vielä joukko $\{a_1, \dots, a_n, b, c\}$ alkion c peittämiseksi. Algoritmi siis tuottaa kustannuksen $n + 1 + \varepsilon$. \square

Maksimitoteutuvuusongelma (Maximum satisfiability, MAX-SAT)

Tehtävänä on löytää sopivia arvoja totuusarvomuuttujille x_1, \dots, x_n .

Literaali on muuttuja x_i tai muuttujan negaatio \bar{x}_i .

Totuusarvoasetus (truth assignment) sijoittaa jokaiselle muuttujalle arvon tosi tai epätosi (joita usein merkitään 1 ja 0).

Klausuuli on literaalien disjunktio (jota ajatellaan usein joukkona C literaaleja). Klausuulin c literaalien lukumäärää merkitään $\text{size}(c)$.

konjunkttiivisessa normaalimuodossa (CNF) oleva totuusarvolauseke on klausuulien konjunktio (jota usein ajatellaan joukkona klausuuleja).

k -CNF on muoto, jossa lisäksi jokaisen klausuulin koko on k .

Tunnetussa toteutuvuusongelmassa (SAT) tehtävänä on päättää, onko annetulla CNF-lausekkeella toteuttava totuusarvoasetus. Ongelma on NP-täydellinen vielä rajoitettuna 3-CNF-kaavoihin.

Maksimitoteutuvuusongelmassa jokaiselle klausuulille on lisäksi annettu paino w_c . Tavoitteena on maksimoida toteutuvien klausuulien kokonaispaino. Tarkka ratkaisu on itse asiassa NP-kovaa jopa 2-CNF-kaavoille.

Esitämme ongelmalle kolme approksimointialgoritmia:

- Ensimmäinen takaa approksimaatiosuhteen $1/2$, mutta toimii paremmin suurilla klausuuleilla.
- Toinen takaa approksimaatiosuhteen $1 - 1/e \approx 0.632$, mutta toimii paremmin pienillä klausuuleilla.
- Kolmas yhdistää edelliset algoritmit ja takaa approksimaatiosuhteen $3/4$.

Esitämme kunkin algoritmin ensin satunnaisversiona, joka saavuttaa halutun approksimaatiosuhteen odotusarvoisesti. Sen jälkeen näytämme, miten algoritmi voidaan **derandomisoida** eli poistaa siitä satunnaisuus.

Kaikissa algoritmeissa satunnaismuuttuja W tarkoittaa toteutuvien klausuulien kokonaispainoa ja W_c klausuulin c osuutta siitä. Siis

$$W = \sum_{c \in C} W_c \quad \text{ja} \quad \mathbf{E}[W_c] = w_c \cdot \mathbf{Pr}[c \text{ toteutuu}].$$

Algoritmi pitkille klausuuleille

"Algoritmi A" yksinkertaisesti sijoittaa kullekin muuttujalle arvon 0 todennäköisyydellä $1/2$ ja 1 todennäköisyydellä $1/2$, toisista muuttujista riippumatta. Jos klausuulissa on k literaalia, sen todennäköisyys jäädä epätodeksi on $(1/2)^k$. Siis toteutumistodennäköisyys on $1 - 1/2^k$. (Jos klausuulissa on sama muuttuja sekä negaatiolla että ilman, se on aina tosi; emme jatkossa mainitse näitä erikoistapauksia.)

Merkitään $\alpha_k = 1 - 1/2^k$. Siis jos $k = \text{size}(c)$, niin $\mathbf{E}[W_c] = \alpha_k w_c$. Koska $\alpha_k \geq 1/2$, saadaan

$$\mathbf{E}[W] = \sum_{c \in C} \mathbf{E}[W_c] \geq \frac{1}{2} \sum_{c \in C} w_c \geq \frac{1}{2} \text{OPT}.$$

Siis algoritmi on $1/2$ -approksimointialgoritmi.

Edellisestä analyysistä seuraa, että jos kaikissa klausuuleissa on ainakin k literaalia, approksimointisuhteeksi saadaan ainakin α_k . Erityisesti tapauksessa $k = 2$ saadaan suhde $3/4$.

Satunnaisuuden poistaminen

Poistamme satunnaisuuden ehdollisen odotusarvon menetelmällä. Käytämme merkintää $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i]$ muuttujan W ehdolliselle odotusarvolle, kun muuttujien x_1, \dots, x_i arvot on kiinnitetty ja muuttujien x_{i+1}, \dots, x_n arvot valitaan satunnaisesti.

Yhdelle klausuulille $\mathbf{E}[W_c \mid x_1 = a_1, \dots, x_i = a_i] = w_c$, jos jollain $1 \leq i \leq n$ siinä esiintyy literaali x_i , jolla $a_i = 1$, tai literaali \bar{x}_i , jolla $a_i = 0$. Muuten $\mathbf{E}[W_c \mid x_1 = a_1, \dots, x_i = a_i] = \alpha_k w_c$, missä k on muuttujien x_{i+1}, \dots, x_n lukumäärä klausuulissa c . Siis ehdolliset odotusarvot on helppo laskea polynomisessa ajassa.

Lähtökohtana tiedämme, että $\mathbf{E}[W] \geq \frac{1}{2}\text{OPT}$.

Tavoitteena on löytää vakiot a_1, \dots, a_n , joilla $\mathbf{E}[W \mid x_1 = a_1, \dots, x_n = a_n] \geq \frac{1}{2}\text{OPT}$. Tässä odotusarvomerkintä on vain yhdenmukaisuuden takia, mitään satunnaisuutta ei enää ole jäljellä.

Pääsemme tavoitteeseen kiinnittämällä arvoja $x_i = a_i$ yksi kerrallaan niin, että invariantti $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i] \geq \frac{1}{2}\text{OPT}$ pysyy voimassa. Tämä on mahdollista, koska

$$\begin{aligned} \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i] \\ &= \frac{1}{2} \cdot \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 0] \\ &\quad + \frac{1}{2} \cdot \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 1]. \end{aligned}$$

Riittää siis laskea $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = a_{i+1}]$ arvoilla $a_{i+1} = 0$ ja $a_{i+1} = 1$ ja valita näistä suurempi; se on vähintään $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i]$.

Algoritmin A deterministinen versio siis kiinnittää arvot $x_i = a_i$ yksi kerrallaan. Uusi arvo valitaan laskemalla odotusarvo $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = a_{i+1}]$ erikseen tapauksessa $a_{i+1} = 0$ ja $a_{i+1} = 1$ ja valitsemalla se, joka antaa suuremman arvon.

Satunnaisuuden poistamisessa sinänsä ei tarvittaisi oletusta, että alkuperäisessä satunnaisalgoritmissa muuttujien x_i arvot ovat riippumattomia. Joka tapauksessa pätee

$$\begin{aligned} \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i] \\ &= \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 0] \cdot \mathbf{Pr}[x_{i+1} = 0 \mid x_1 = a_1, \dots, x_i = a_i] \\ &\quad + \mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i, x_{i+1} = 1] \cdot \mathbf{Pr}[x_{i+1} = 1 \mid x_1 = a_1, \dots, x_i = a_i]. \end{aligned}$$

Siis $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i]$ on painotettu keskiarvo kahdesta vaihtoehdosta, joten toinen näistä on ainakin yhtä suuri.

Se, mihin edellisessä oleellisesti käytettiin riippumattomuusoletusta, oli ehdollisten odotusarvojen $\mathbf{E}[W \mid x_1 = a_1, \dots, x_i = a_i]$ laskeminen, joka ei välttämättä olisi lainkaan helppoa, jos muuttujien välillä on monimutkaisia riippuvuuksia.

Algoritmi lyhyille klausuuleille

Muodostetaan ensin kokonaislukuohjelma, jossa y_i kertoo muuttujan x_i arvon ja z_c sen, toteutuuko klausuuli c . Joukot S_c^+ ja S_c^- ovat niiden muuttujien indeksejä, jotka esiintyvät klausuulissa c vastaavasti ilman negaatiota ja negaation kanssa. Kokonaislukuohjelma on siis

$$\begin{array}{ll} \text{maksimoi} & \sum_{c \in C} w_c z_c \\ \text{ehdolla} & \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c \quad \text{kaikilla } c \in C \\ & z_c \in \{0, 1\} \quad \text{kaikilla } c \in C \\ & y_i \in \{0, 1\} \quad \text{kaikilla } i \in \{1, \dots, n\}. \end{array}$$

Löysennökseksi tulee

$$\begin{array}{ll} \text{maksimoi} & \sum_{c \in C} w_c z_c \\ \text{ehdolla} & \sum_{i \in S_c^+} y_i + \sum_{i \in S_c^-} (1 - y_i) \geq z_c \quad \text{kaikilla } c \in C \\ & 1 \geq z_c \geq 0 \quad \text{kaikilla } c \in C \\ & 1 \geq y_i \geq 0 \quad \text{kaikilla } i \in \{1, \dots, n\}. \end{array}$$

Myös tästä saatava "algoritmi B" on suoraviivainen:

1. Ratkaise löysennös tarkasti. Olkoon ratkaisu $(\mathbf{y}^*, \mathbf{z}^*)$.
2. Kaikilla i toisistaan riippumatta aseta muuttujalle x_i arvo 1 todennäköisyydellä y_i^* ja arvo 0 todennäköisyydellä $1 - y_i^*$.

Lemma 3.19: Olkoon $k = \text{size}(c)$ ja $\beta_k = (1 - (1 - 1/k)^k)$. Algoritmilla B pätee

$$\mathbf{E}[W_c] \geq \beta_k w_c z_c^*.$$

Todistus: Yleisyyttä rajoittamatta voimme olettaa, että klausuulissa c ei ole negaatioita. Numeroidaan vielä muuttujat niin, että $c = (x_1 \vee \dots \vee x_k)$.

Aritmeettista ja geometrista keskiarvoa koskevan epäyhtälön nojalla

$$\left(\prod_{i=1}^k (1 - y_i^*) \right)^{1/k} \leq \frac{1}{k} \sum_{i=1}^k (1 - y_i^*).$$

Siis todennäköisyys, että klausuuli c toteutuu, on

$$\begin{aligned} 1 - \left(\prod_{i=1}^k (1 - y_i^*) \right) &\geq 1 - \left(\frac{1}{k} \sum_{i=1}^k (1 - y_i^*) \right)^k = 1 - \left(1 - \frac{1}{k} \sum_{i=1}^k y_i^* \right)^k \\ &\geq 1 - \left(1 - \frac{z_c^*}{k} \right)^k, \end{aligned}$$

sillä (\mathbf{y}^*, z^*) on käypä ratkaisu ja siis $y_1^* + \dots + y_k^* \geq z_c^*$.

Kirjoitetaan edellinen lauseke muotoon $g(z_c^*)$, missä $g(z) = 1 - (1 - z/k)^k$. Tarkastellaan funktiota g välillä $[0, 1]$. Koska $g''(z) \leq 0$, funktio on konkaavi. Siis Jensenin epäyhtälön mukaan

$$(1 - \alpha)g(z_1) + \alpha g(z_2) \leq g((1 - \alpha)z_1 + \alpha z_2)$$

kaikilla $0 \leq \alpha \leq 1$. Valitaan $z_1 = 0$, $z_2 = 1$ ja $\alpha = z$. Koska $g(0) = 1$ ja $g(1) = \beta_k$, saadaan

$$g(z) \geq \beta_k z.$$

Siis klausuuli c toteutuu ainakin todennäköisyydellä $\beta_k z_c^*$. \square

Oletetaan, että klausuulien koot ovat korkeintaan K . Koska β_k on k :n suhteen pienenevä,

$$\begin{aligned}\mathbf{E}[W] &\geq \beta_K \sum_{c \in C} w_c z_c^* \\ &= \beta_K \text{OPT}_f \\ &\geq \beta_K \text{OPT},\end{aligned}$$

missä taas OPT ja OPT_f ovat kokonaislukuohjelman ja löysennöksen optimiarvot. Algoritmista voidaan taas poistaa satunnaisuus ehdollisten odotusarvojen menetelmällä (sivuutamme yksityiskohdat). Koska kaikilla k pätee $(1 - 1/k)^k \leq 1/e$, saamme $(1 - 1/e)$ -approksimointialgoritmin.

3/4-approksimointialgoritmi

Esitämme taas ensin suoraviivaisen satunnaisversion:

1. Valitse satunnaisesti $b = 0$ todennäköisyydellä $1/2$ ja $b = 1$ todennäköisyydellä $1/2$.
2. Jos $b = 0$, käytä (satunnaista) algoritmia A.
Jos $b = 1$, käytä (satunnaista) algoritmia B.

Siis muuttujan x_i arvoksi tulee 1 todennäköisyydellä $\frac{1}{4} + \frac{1}{2}y_i^*$, mutta nyt nämä tapahtumat eivät ole riippumattomia.

Lemma 3.20: Yhdistetyille satunnaisalgoritmile pätee

$$\mathbf{E}[W_c] \geq \frac{3}{4}w_c z_c^*.$$

Todistus: Kun $\text{size}(c) = k$, edellisten lemmojen nojalla

$$\mathbf{E}[W_c \mid b = 0] = \alpha_k w_c \geq \alpha_k w_c z_c^*$$

$$\mathbf{E}[W_c \mid b = 1] \geq \beta_k w_c z_c^*$$

Siis

$$\mathbf{E}[W_c] \geq w_c z_c^* \cdot \frac{\alpha_k + \beta_k}{2}.$$

Koska $\alpha_1 + \beta_1 = \alpha_2 + \beta_2 = 3/2$ ja arvoilla $k \geq 3$ pätee

$$\alpha_k + \beta_k \geq \frac{7}{8} + \left(1 - \frac{1}{e}\right) \geq 3/2,$$

saadaan haluttu tulos. \square

Saadaan haluttu odotusarvoinen tulos

$$\mathbf{E}[W] \geq \frac{3}{4} \sum_{c \in C} w_c z_c^* = \frac{3}{4} \text{OPT}_f \geq \frac{3}{4} \text{OPT}.$$

Lopulta poistetaan satunnaisuus, ja saadaan seuraava algoritmi:

Algoritmi ($\frac{3}{4}$ -approksimaatio maksimitoteutuvuuteen)

1. Anna tapaus algoritmille A; olkoon saatu ratkaisu τ_A .
2. Anna tapaus algoritmille B; olkoon saatu ratkaisu τ_B .
3. Valitse ratkaisuista τ_A ja τ_B parempi.

Lause 3.21: Edellinen algoritmi on $\frac{3}{4}$ -approksimointialgoritmi.

Todistus: Edellä todistimme itse asiassa, että

$$\mathbf{E}_A[W] \geq \sum_{c \in C} \alpha_{\text{size}(c)} w_c z_c^* := R_A$$
$$\mathbf{E}_B[W] \geq \sum_{c \in C} \beta_{\text{size}(c)} w_c z_c^* := R_B,$$

missä alaindeksit A ja B viittaavat vastaaviin satunnaisalgoritmeihin. Satunnaisuuden poistamisessa pidettiin yllä perusinvariantti, että kohdefunktion ehdollinen odotusarvo ei huonone. Siis myös deterministisesti saatujen ratkaisujen τ_A ja τ_B kohdefunktion arvot ovat vähintään R_A ja R_B . Vertaamalla näitä summia termeittäin edellisessä lemmassa esitettyyn tapaan saadaan haluttu tulos. \square

Koska algoritmin tuottama ratkaisu antaa alarajan OPT:lle, seuraa erityisesti, että $\text{OPT}/\text{OPT}_f \geq \frac{3}{4}$. Seuraava esimerkki osoittaa, että tämä raja on tiukka.

Esimerkki 3.22: Tarkastellaan lauseketta

$$f = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

ja annetaan jokaiselle klausuulille paino 1.

Kaikkia klausuuleja ei saa toteutumaan yhtäikää, joten $\text{OPT} = 3$.

Asettamalla $y_i = 1/2$ kaikilla i ja $z_c = 1$ kaikilla c saadaan löysennettyyn ongelmaan käypä ratkaisu, jonka arvo on 4. \square

Muodostamme vielä tiukan esimerkin algoritmillemme.

Esimerkki 3.23: Valitaan

$$f = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3)$$

ja annetaan klausuuleille painot 1, 1 ja $2 + \varepsilon$. Löysennöksen (eräs) optimi saavutetaan taas asettamalla $y_i = 1/2$ kaikilla i . Tällöin (satunnainen) algoritmi B antaa kullekin muuttujalle arvon 1 todennäköisyydellä $1/2$. Jakauma on sama kuin algoritmilla A, joten satunnaisuuden poisto algoritmeissa A ja B johtaa samoihin ratkaisuihin.

Oletetaan, että ensin kiinnitetään muuttujan x_1 arvo. Koska

$$\begin{aligned} \mathbf{E}[W \mid x_1 = 0] &= 3 + \varepsilon \\ \mathbf{E}[W \mid x_1 = 1] &= 3 + \varepsilon/2, \end{aligned}$$

algoritmi valitsee $x_1 = 0$. Tämä johtaa kokonaispainoon $3 + \varepsilon$, kun valitsemalla $x_1 = 1$ saataisiin $4 + \varepsilon$.

Toistamalla näitä kolmea klausuulia uusilla muuttujilla saadaan mielivaltaisen suuria esimerkkejä, joilla vastaava pätee. \square

Skedulointi erilaisilla rinnakkaisilla koneilla

Ongelman tapauksessa on annettu töiden joukko J ja koneiden joukko M sekä jokaiselle parille $(i, j) \in M \times J$ suoritus aika $p_{ij} \in \mathbb{Z}_+$, jonka työn j suorittaminen koneella j kestää. Tehtävänä on jakaa työt koneille minimoiden valmistusaika (makespan)

$$\max_{i \in M} \sum_{j \in J} x_{ij} p_{ij},$$

missä $x_{ij} = 1$, jos työ j on sijoitettu konelle i , ja $x_{ij} = 0$ muuten. Merkitsemme $n = |J|$ ja $m = |M|$.

Esitämme parametriseen karsintaan ja pyöristämiseen perustuvan 2-aproksimointialgoritmin.

Tässä siis töiden suoritusajat voivat vaihdella täysin mielivaltaisesti eri koneiden välillä. Tasaisilla rinnakkaisilla koneilla oletetaan jokaisella koneella i olevan nopeus s_i ja työllä j aikavaatimus p_j , ja työn j suorittaminen koneella i vie ajan p_j/s_i . Tälle versiolle tunnetaan PTAS.

Kokonaislukuohjelmassa x_{ij} kertoo, onko työ j koneella i , ja t on valmistusaika:

minimoi	t	
ehdolla	$\sum_{i \in M} x_{ij} = 1$	kaikilla $j \in J$
	$\sum_{j \in J} x_{ij} p_{ij} \leq t$	kaikilla $i \in M$
	$x_{ij} \in \{0, 1\}$	kaikilla $i \in M$ ja $j \in J$.

Ongelman kokonaislukurako on rajoittamaton:

Esimerkki 3.24: Valitaan $n = 1$ ja ainoan työn suoritusajaksi kaikilla koneilla $p_{i1} = m$. Optimikustannus on tietysti m , mutta löysennökselle saadaan kustannus $1/m$ valitsemalla $x_{i1} = 1/m$ kaikilla i . \square

Edellisessä esimerkissä kohtuuttoman hyvä murtolukuratkaisu voitaisiin estää lisäämällä yksinkertainen vaatimus

jos $p_{ij} > t$ niin $x_{ij} = 0$.

Tämä rajoite ei kuitenkaan ole lineaarinen. Sen takia otamme ongelman parametriksi arvauksen $T \in \mathbb{Z}_+$ siitä, mikä optimaalinen valmistusaika on.

Tämän perusteella muodostamme löysennetyn ongelman $LP(T)$, johon tulee vain muuttujat x_{ij} , joilla $p_{ij} \leq T$. Olkoon näiden muuttujien joukko S_T . Koska T on annettu, ongelmana on vain testata, onko olemassa käypä ratkaisu rajoitteille

$$\begin{aligned} \sum_{i:(i,j) \in S_T} x_{ij} &= 1 && \text{kaikilla } j \in J \\ \sum_{j:(i,j) \in S_T} x_{ij} p_{ij} &\leq T && \text{kaikilla } i \in M \\ x_{ij} &\geq 0 && \text{kaikilla } i \in M \text{ ja } j \in J. \end{aligned}$$

Rajoite $x_{ij} \leq 1$ voidaan taas jättää pois turhana.

Kantaratkaisujen ominaisuuksia

Olkoon T^* pienin T , jolla ohjelmalla $LP(T)$ on käypiä ratkaisuja. Selvästi $OPT \geq T^*$. Algoritmin idea on muodostaa kantaratkaisu ohjelmalle $LP(T^*)$ ja pyöristää.

Lemma 3.25: Ohjelman $LP(T)$ kantaratkaisussa korkeintaan $n + m$ muuttujalla on nolosta poikkeava arvo.

Todistus: Olkoon $r = |S_T|$ muuttujien lukumäärä. Tunnetusti käypä ratkaisu on kantaratkaisu, jos ja vain jos r lineaarisesti riippumatonta rajoitetta pätee yhtäsuuruutena. Näistä rajoitteista ainakin $r - (n + m)$ on muotoa $x_{ij} \geq 0$. \square

Sanomme, että annetussa ratkaisussa työ j on **kokonainen**, jos se on sijoitettu vain yhdelle koneelle i (siis $x_{ij} = 1$) ja muuten **jaettu** (siis $x_{ij} > 0$ ainakin kahdella eri i).

Korollaari 3.26: Ohjelman $LP(T)$ kantaratkaisussa ainakin $n - m$ työtä on kokonaisia.

Todistus: Olkoon x kantaratkaisu. Olkoon α kokonaisten ja β jaettujen töiden lukumäärä. Siis

$$\alpha + \beta = n.$$

Jokainen kokonainen työ tuottaa yhden ja jaettu työ ainakin kaksi nollasta poikkeavaa muuttujaa, joten edellisen lemmän nojalla

$$\alpha + 2\beta \leq n + m.$$

Siis $\beta \leq m$. \square

Kantaratkaisun x pyöristämisessä tarvitsemme kaksijakoista verkkoa $G = (J \cup M, E)$, missä $(j, i) \in E$, jos ja vain jos $x_{ij} \neq 0$. Olkoon lisäksi $F \subset J$ jaettujen töiden joukko ja H solmujoukon $F \cup M$ indusoima verkon G aliverkko. Siis verkossa H on kaari (i, j) , jos ja vain jos $0 < x_{ij} < 1$.

Osoitamme jatkossa, että verkolla H on **täydellinen** pariutus, ts. sellainen että jokainen joukon F työ on pariutettu. Tällaista pariutusta tarvitaan seuraavassa algoritmossa.

Algoritmossa tarvitaan myös havaintoa, että optimaalinen valmistusaika on välillä $[\alpha/m, \alpha]$, missä α on **ahneen algoritmin** tuottama valmistusaika. Ahne algoritmi sijoittaa työn j koneeseen i , jolla p_{ij} on pienin. Saatu valmistusaika α on tietysti yläraja minimille. Ahneen algoritmin kannalta pahimmassa tapauksessa kaikki työt menevät samalle koneelle, jolloin enimmillään valmistusaika voi lyhentyä kertoimella $1/m$.

Algoritmi (2-approksimaatio erilaisilla koneilla skedulointiin)

1. Etsi binäärihaulla väliltä $[\alpha/m, \alpha]$ pienin $T \in \mathbb{Z}_+$, jolla ohjelmalla $LP(T)$ on käypä ratkaisu. Olkoon tämä arvo T^* .
2. Etsi ohjelmalle $LP(T^*)$ kantaratkaisu x .
3. Sijoita ratkaisun x kokonaiset työt ratkaisun x mukaan.
4. Muodosta edellisen kalvon mukainen verkko H ja sille täydellinen pariutus \mathcal{M} .
5. Sijoita ratkaisun x jaetut työt pariutuksen \mathcal{M} mukaan.

Tarvitsemme siis menetelmän verkon H täydellisen pariutuksen löytämiseksi.

Sanomme, että verkko (V, E) on **valepuu**, jos se on yhtenäinen ja $|E| \leq |V|$.
Siis valepuu on puu tai puu plus yksi kaari, joka synnyttää tasan yhden syklin. **Valemetsä** on verkko, jonka jokainen komponentti on valepuu.

Lemma 3.27: Verkko G on valemetsä.

Todistus: Tarkastellaan mielivaltaista komponenttia G_c . Pitää siis osoittaa, että komponentissa G_c on yhtä monta kaarta kuin solmua.

Rajoitetaan $LP(T)$ ja kantaratkaisu x komponentissa G_c oleviin töihin ja koneisiin. Olkoon saatu lineaarinen ohjelma $LP_c(T)$, ja vastaava vektorin x osavektori x_c . Lopuista komponenteista muodostetaan osavektori $x_{\bar{c}}$.

Ratkaisu x_c on ohjelman $LP_c(T)$ kantaratkaisu. Jos se nimittäin olisi kahden käyvän ratkaisun x'_c ja x''_c konvekssi yhdistelmä, niin x olisi käypien ratkaisujen $(x'_c, x_{\bar{c}})$ ja $(x''_c, x_{\bar{c}})$ konvekssi yhdistelmä, vaikka oletuksen mukaan se on kantaratkaisu.

Lemman 3.25 mukaan ohjelman $LP_c(T)$ kantaratkaisussa nollostapoikkeavia muuttujia on korkeintaan yhtä monta kuin verkossa G_c solmuja. \square

Lemma 3.28: Verkolla H on täydellinen pariutus.

Todistus: Ratkaisun x kokonaista työtä vastaavaan solmuun liittyy verkossa G tasan yksi kaari. Nämä solmut ja kaaret poistetaan ja tuloksena saadaan H . Koska G oli valemetsä ja kaaria ja solmuja poistui jokaisesta komponentista saman verran, myös H on valemetsä.

Verkossa H kaikki lehdet (astetta 1 olevat solmut) ovat koneita. Pariutetaan kukin lehti vanhempansa kanssa ja poistetaan kaikki pariutetut solmut ja niihin liittyvät kaaret. Jäljelle jäävässä verkossa edelleen kaikki lehdet ovat koneita. Jos nimittäin työ j olisi muuttunut lehdeksi, niin verkosta olisi poistettu jokin työn j naapurikone i sekä koneen i naapuri $j' \neq j$. Konetta ei kuitenkaan poisteta, jos sillä on kaksi naapuria.

Voimme siis toistaa edellistä, kunnes lehtiä ei ole. Lehdetön valepuu muodostuu yhdestä syklistä. Koska verkko on kaksijakoinen, syklissä on parillinen määrä solmuja, joten nämäkin solmut voidaan helposti pariuttaa.

□

Lause 3.29: Edellinen algoritmi on 2-aproksimointialgoritmi erilaisten rinnakkaisten koneiden skedulointiin.

Todistus: Koska ohjelmalla $LP(OPT)$ on käypä ratkaisu, pätee $T^* \leq OPT$.

Ohjelman $LP(T^*)$ kantaratkaisulla x on valmistusaika korkeintaan T^* . Erityisesti siis pelkkien kokonaisten töiden skeduloiminen ratkaisun x mukaan, kuten algoritmi tekee, vie korkeintaan valmistusajan T^* .

Algoritmi lisää jaettuja töitä korkeintaan yhden konetta kohti. Koska $p_{ij} \leq T^*$ kaikilla verkon H kaarilla, minkään koneen aikavaatimus ei kasva enempää kuin T^* .

Siis kokonaisten ja jaettujen töiden suorittaminen vie yhteensä korkeintaan ajan $T^* + T^* \leq 2 \cdot OPT$. \square

Approksimointisuhde 2 on tiukka.

Esimerkki 3.30: Mielivaltaisella m muodostetaan m konetta ja $m^2 - m + 1$ työtä. Ensimmäinen työ vie ajan m kaikilla koneilla ja loput työt kukin ajan 1 kaikilla koneilla.

Optimiratkaisu varaa yhden koneen ensimmäiselle työlle ja jakaa loput $m(m - 1)$ työtä tasan loppuilla $m - 1$ koneelle. Valmistusaika on m .

Selvästi ohjelmalla $LP(T)$ ei ole käypää ratkaisua, jos $T < m$. Oletetaan, että algoritmi valitsee ohjelman $LP(m)$ kantaratkaisun, jossa jokainen kone saa osuuden $1/m$ ensimmäisestä työstä ja $m - 1$ muuta työtä. Pyöristäminen tuottaa valmistusajan $2m - 1$. \square

Monileikkaus ja monihyödykevuoto

Monileikkauksen minimointiongelmassa (minimum multicut) on annettu verkko $G = (V, E)$, jokaiselle kaarelle $e \in E$ kapasiteetti $c_e \geq 0$ sekä joukko solmupareja $\{(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)\}$. Oletamme, että kukin pari esiintyy listassa vain kerran, mutta yksittäinen solmu voi esiintyä useassa parissa.

Monileikkaus on kaarijoukko, jonka poistaminen verkosta erottaa solmut s_i ja t_i toisistaan kaikilla i . Tehtävänä on löytää kapasiteetiltaan pienin monileikkaus.

Tapaus $k = 1$ on aiemmin käsitelty polynomisessa ajassa ratkeava minimileikkausongelma.

Toinen erikoistapaus on monensuuntainen leikkaus (multiway cut), jossa on annettu joukko $\{s_1, \dots, s_k\}$ ja on erotettava kaikki parit (s_i, s_j) , $i \neq j$. Monensuuntaisen leikkauksen minimoiminen on NP-kovaa jo tapauksessa $k = 3$, joten niin on myös monileikkausongelma.

Tarkastelemme tässä monileikkauksen minimointia erikoistapauksessa, että verkko on **puu**. Puussa jokaisen solmuparin (s_i, t_i) välillä on tasan yksi polku, ja tehtävä on siis valita ainakin yksi kaari jokaiselta tällaiselta polulta. Tämänkin ongelman tarkka ratkaisu on NP-kova, vaikka puun korkeus olisi yksi ja kaikilla kaarilla sama kapasiteetti.

Esitämme primaali-duaaliskeemaan perustuvan 2-aproksimointialgoritmin. Duaalin kokonaislukuversio on **monihyödykevuon** (multicommodity flow) maksimointi, ja saamme samalla tälle $\frac{1}{2}$ -aproksimointialgoritmin.

Myöhemmin esitämme $O(\log n)$ -aproksimaation yleisille verkoille.

Olkoon p_i puun yksikäsitteinen polku solmujen s_i ja t_i välillä. Muuttuja d_e esittää, onko kaari e mukana monileikkauksessa. Saamme kokonaislukuohjelman

$$\begin{array}{ll} \text{minimoi} & \sum_{e \in E} c_e d_e \\ \text{ehdolla} & \sum_{e \in p_i} d_e \geq 1 \quad \text{kaikilla } i \in \{1, \dots, k\} \\ & d_e \in \{0, 1\} \quad \text{kaikilla } e \in E. \end{array}$$

Löysennöksestä voidaan taas jättää pois ehdot $d_e \leq 1$:

$$\begin{array}{ll} \text{minimoi} & \sum_{e \in E} c_e d_e \\ \text{ehdolla} & \sum_{e \in p_i} d_e \geq 1 \quad \text{kaikilla } i \in \{1, \dots, k\} \\ & d_e \geq 0 \quad \text{kaikilla } e \in E. \end{array}$$

Löysennöksen käypiä ratkaisuja sanomme **murto**monileikkauksiksi (fractional multicut).

Otamme duaaliin muuttujiksi f_1, \dots, f_k ja saamme

$$\begin{array}{ll} \text{maksimoi} & \sum_{i=1}^k f_i \\ \text{ehdolla} & \sum_{i:e \in p_i} f_i \leq c_e \quad \text{kaikilla } e \in E \\ & f_i \geq 0 \quad \text{kaikilla } i \in \{1, \dots, k\}. \end{array}$$

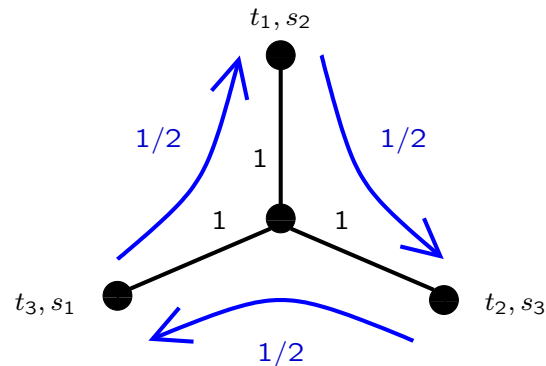
Tämä kuvaa **monihyödykevuon**: Muuttuja f_i on jonkin annetun hyödykkeen vuo solmujen s_i ja t_i välillä. Hyödykkeet ovat erilaisia, joten jokainen niistä erikseen toteuttaa vuon säilymisehdon. Kapasiteettirajoitukset kuitenkin koskevat kaikkien hyödykkeiden yhteisvuota, jossa lisäksi vuon suunnalla ei ole merkitystä.

Kokonaislukuarvoisen monihyödykevuon maksimoinnissa edellinen rajoitetaan muuttujien f_i kokonaislukuarvoihin. Tällöin lisäksi oletetaan, että kaarten kapasiteetit ovat kokonaislukuarvoisia.

Duaalisuuslauseen perusteella

kokonaislukuarvoisen monihyödykevuon maksimi \leq monihyödykevuon maksimi
= murtomonileikkauksen minimi \leq monileikkauksen minimi.

Esimerkki 3.31: Seuraavassa verkossa kunkin kaaren kapasiteetti on 1. Nuolet antavat murtolukuarvoisen vuon $3/2$, mistä $1/2$ kunkin solmuparin välillä. Asettamalla $d_e = 1/2$ jokaiselle kaarelle e saadaan murtolukuleikkaus, jonka kapasiteetti on niinkään $3/2$. Koska arvot ovat yhtäsuuret, kyseessä on optimaalinen primaalin ja duaalin ratkaisupari.



Kokonaislukuarvoiseen leikkaukseen pitää valita ainakin kaksi kaarta, joten kokonaislukuarvoisen monileikkauksen minimi on 2. Monivuo-ongelmassa yksikin yksikkövuoto kyllästää muidenkin hyödykkeiden reitit, joten suurin kokonaislukuvuoto on 1. \square

Approksimointi primaali-duaaliskeemalla

Esitämme algoritmin, jonka tuottamien primaalin ja duaalin arvo ovat kertoimen 2 sisällä toisistaan. Lisäksi kumpikin on kokonaislukuarvoinen. Saamme siis saman tien 2-approksimointialgoritmin monileikkaukselle ja $\frac{1}{2}$ -approksimointialgoritmin monihyödykevuolle.

Saatamme komplementaarisuusehtojen (s. 173) primaaliosan voimaan tarkasti ($\alpha = 1$) ja duaaliosan kertoimella $\beta = 2$. Siis

Primaaliehto: kaikilla $e \in E$, jos $d_e \neq 0$ niin $\sum_{i:e \in p_i} f_i = c_e$.

Duaaliehto: kaikilla $i \in \{1, \dots, k\}$, jos $f_i \neq 0$ niin $\sum_{e \in p_i} d_e \leq 2$.

Siis

- jokainen leikkaukseen valittu kaari on kyllästetty
- jokaiselta vuota kuljettavalta polulta on valittu korkeintaan kaksi kaarta (ja käyppyysehtojen takia vähintään yksi).

Valitsemme puussa G mielivaltaisen solmun r juureksi. Solmun e **syvyys** on polun $r \rightsquigarrow v$ pituus (kaarien lukumäärä).

Solmujen u ja v **alin yhteinen esi-isä** $\text{lca}(u, v)$ (lower common ancestor) on syvin solmu niiden välisellä polulla.

Jos juuresta lähtevällä polulla kaari e_1 on ennen kaarta e_2 , kaari e_2 on **syvemmällä** kuin e_1 .

Algoritmi aloittaa tyhjästä vuosta f ja kaarijoukosta D ja lisää kaaria D :hen, kunnes se on monileikkaus. Strategiana on käsitellä solmut yksi kerrallaan syvyyden mukaan pienenevässä järjestyksessä.

Valitsemme solmun v ja rupeamme lisäämään vuota sellaisten solmuparien välille, joiden alin yhteinen esi-isä v on. Kun tällaisia voita ei enää saada lisätyksi, kyllästyneet kaaret lisätään joukkoon D .

Kun kaikki solmut on käsitelty, D on monileikkaus mutta voi sisältää turhia kaaria. Poistamme nämä [takaperin karsimalla](#), ts. alkaen viimeksi lisätyistä.

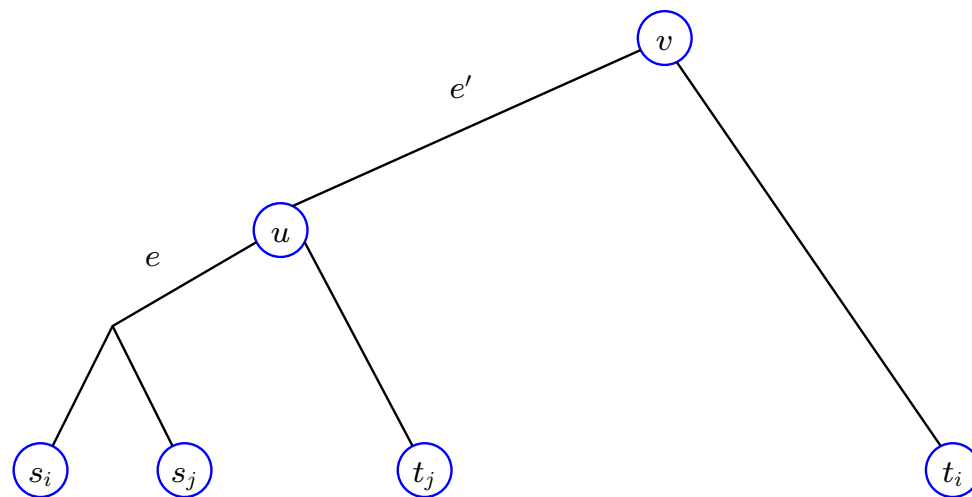
Algoritmi (monileikkaus ja kokonaislukuarvoinen monihyödykevuoto puussa)

1. Alusta $f \leftarrow 0$ ja $D \leftarrow \emptyset$.
2. Jokaiselle solmulle v syvyyden mukaan ei-kasvavassa järjestyksessä:
 - (a) Lisää ahneesti kokonaislukuvuotoa kaikkien solmuparien (s_i, t_i) välillä, joilla $v = \text{lca}(s_i, t_i)$.
 - (b) Lisää joukkoon D kyllästetyksi tulleet kaaret.
3. Kaarille $e \in D$ lisäysjärjestyksen suhteen käänteisessä järjestyksessä: jos $D - \{e\}$ on monileikkaus, aseta $D \leftarrow D - \{e\}$.
4. Tulosta vuo f ja monileikkaus D .

Lemma 3.32: Oletetaan, että ratkaisussa f parin (s_i, t_i) välille tuli nollostapoikkeava vuo. Merkitään $v = \text{lca}(s_i, t_i)$. Nyt monileikkauksessa D on poluilta $s_i \rightsquigarrow v$ ja $t_i \rightsquigarrow v$ kummaltakin korkeintaan yksi kaari

Todistus: Tarkastellaan polkua $s_i \rightsquigarrow v$; polku $t_i \rightsquigarrow v$ käsitellään samaan tapaan. Tehdään vastaoletus, että polulta on valittu kaksi kaarta e ja e' , joista e syvempi.

Tarkastellaan tilannetta, kun karsintavaiheessa e on vuorossa. Koska sitä ei karsita, se on ainoa joukkoon D valittu kaari jollain polulla $s_j \rightsquigarrow t_j$. Olkoon $u = \text{lca}(s_j, t_j)$. Koska e' ei ole polulla $s_j \rightsquigarrow t_j$, solmu u on on syvemmällä kuin e' .



Solmun u läpi menevien voiden kasvatusvaiheessa jokin kaari e'' polulta $s_j \rightsquigarrow t_j$ päättyy joukkoon D .

Toisaalta kaarta e ei ole voitu lisätä ennen solmun v käsittelemistä, sillä muuten vuota $s_i \rightsquigarrow t_i$ ei olisi pystytty kasvattamaan.

Koska solmu v on solmun u esi-isä, kaari e on lisätty kaaren e'' jälkeen. Siis karsintavaiheessa e tulee tarkasteluun ennen kaarta e'' . Tämä on vastoin oletusta, että e on ainoa kaari polulla $s_j \rightsquigarrow t_j$. \square

Lause 3.33: Edellinen algoritmi saavuttaa approksimointisuhteen 2 minimimonileikkaukselle ja approksimointisuhteen $1/2$ kokonaislukuarvoiselle maksimivuolle puussa.

Todistus: Vuo f on konstruktion perusteella käypä kokonaislukuratkaisu primaaliin. Koska vuo on maksimaalinen ja D sisältää aluksi kaikki kyllästetyt kaaret, D on tässä vaiheessa monileikkaus. Karsinta pitää tämän invariantin voimassa.

Koska kaikki leikkaukseen valitut kaaret ovat kyllästettyjä, komplementaarisuusehtojen primaalipuoli on voimassa.

Edellisen lemmän mukaan kullakin vuota sisältävältä polulta $s_i \rightsquigarrow t_i$ valitaan korkeintaan kaksi kaarta, joten duaalipuoli on voimassa kertoimella $\beta = 2$.

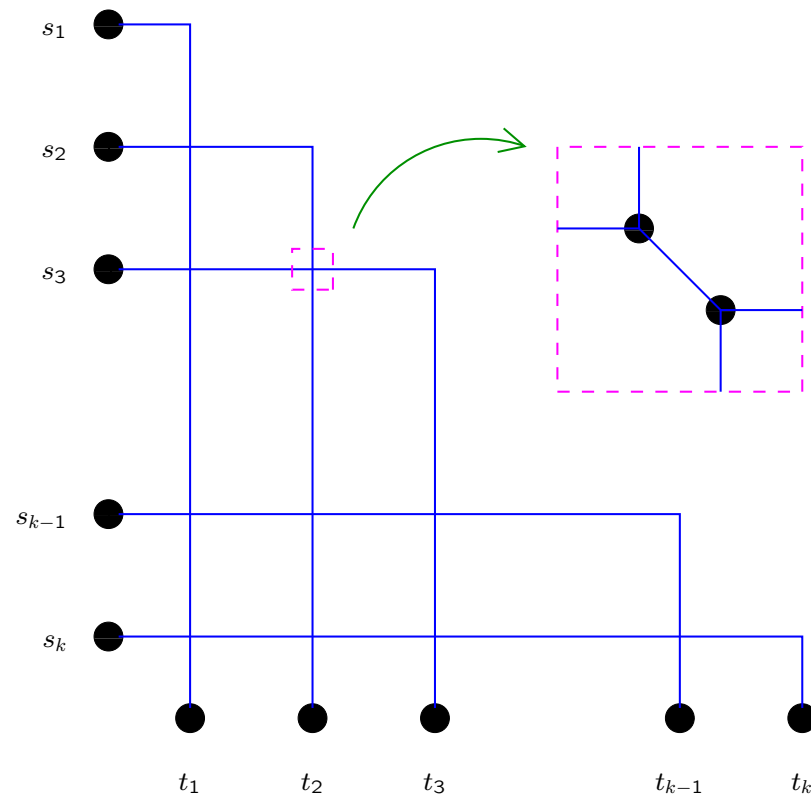
Lemman 3.16 (s. 173) mukaan primaali- ja duaaliratkaisujen arvot ovat siis kertoimen 2 sisällä toisistaan. Primaalin ja duaalin optimiarvot ovat näiden käypien ratkaisujen arvojen välissä, joten väite seuraa. \square

Korollari 3.34: Jos F^* on suurin kokonaislukuarvoinen monihyödykevuoto ja C^* pienimmän monileikkauksen kapasiteetti, niin

$$F^* \leq C^* \leq 2F^*.$$

Esitämme jatkossa $O(\log n)$ -approksimointialgoritmin minimimonileikkaukselle myös yleisissä verkoissa. Kokonaislukuarvoiselle monihyödykevuolle vastaavaa ei tunneta. Seuraava esimerkki osoittaa, että jopa tasoverkossa ongelman kokonaislukurako voi olla $k/2$, missä k on lähde-nieluparien lukumäärä.

Esimerkki 3.35: Allaolevassa tilanteessa voidaan samanaikaisesti lähettää murtolukuvuo $1/2$ jokaisen solmuparin (s_i, t_i) välillä. Sen sijaan yksikin kokonaislukuarvoinen vuo tukkii kaikki muut polut.



Monensuuntaisen leikkauksen minimointi

Tässä siis on annettu verkko $G = (V, E)$, kaarille $e \in E$ kapasiteetti $c(e) \in \mathbb{Q}_+$ ja joukko päätesolmuja $\{s_1, \dots, s_k\}$.

Moninkertainen leikkaus (multiway cut) on kaarijoukko $C \subseteq E$, jonka poistaminen erottaa toisistaan kaikki solmuparit (s_i, s_j) , $i \neq j$. Tehtävänä on minimoida moninkertaisen leikkauksen kapasiteetti $c(C) = \sum_{e \in C} c(e)$.

Esitämme ensin yksinkertaisen algoritmin, joka antaa approksimaatiosuhteen $2 - 2/k$ (Vazirani luku 4.1). Ongelman ilmeisen lineaarisen löysennöksen kokonaislukurako on sama $2 - 2/k$, mutta tekemällä löysennös ei-ilmeisellä tavalla saadaan pyöristyksellä approksimointisuhde $3/2$.

Päättesolmun s_i **eristävä leikkaus** on kaarijoukko, joka erottaa sen kaikista muista päättesolmuista.

Algoritmi $(2 - 2/k)$ -approksimaatio monensuuntaiselle leikkaukselle)

1. Arvoilla $i = 1, \dots, k$ olkoon C_i kapasiteetiltaan pienin solmun s_i eristävä leikkaus. Tämän voi löytää normaalilla minimileikkaus-maksimivuoalgoritmilla yhdistämällä muut solmut s_j vuoverkon toiseksi päättesolmuksi.
2. Heitä menemään kapasiteetiltaan suurin eristävistä leikkauksista C_i . Palauta C , joka on jäljelle jääneiden joukkojen C_i yhdiste.

Lause 3.36: Edellinen algoritmi saavuttaa approksimointisuhteen $2 - 2/k$.

Todistus: Olkoon A optimaalinen monensuuntainen leikkaus. Sen poistaminen jakaa verkon yhtenäisiin komponentteihin V_1, \dots, V_k , missä $s_i \in V_i$.

Otetaan joukkoon $A_i \subset A$ ne kaaret, joiden toinen päätepiste on joukossa V_i . Jokainen kaari tulee siis kahteen joukkoon A_i , joten

$$\sum_{i=1}^k c(A_i) = 2c(A).$$

Koska A_i eristää solmun s_i , pätee $c(C_i) \leq c(A_i)$.

Koska joukoista C_i heitettiin menemään kapasiteetiltaan suurin, saadaan

$$c(C) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(C_i) \leq \left(1 - \frac{1}{k}\right) \sum_{i=1}^k c(A_i) \leq \left(1 - \frac{1}{k}\right) c(A). \quad \square$$

Muodostetaan kokonaislukuohjelma, jossa muuttuja $d(u, v)$ kertoo, tuleeko kaari (u, v) mukaan leikkaukseen:

$$\text{minimoi} \quad \sum_{(u,v) \in E} c(u, v) d(u, v)$$

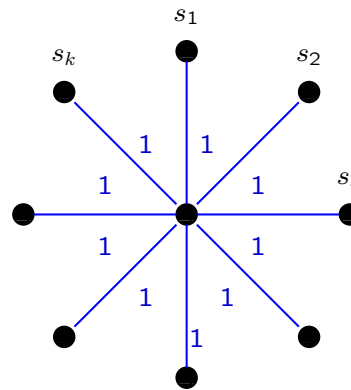
$$\text{ehdolla} \quad \sum_{(u,v) \in p} d(u, v) \geq 1 \quad \text{kaikilla poluilla } p: s_i \rightsquigarrow s_j, i \neq j$$

$$d(u, v) \in \{0, 1\} \quad \text{kaikilla } (u, v) \in E.$$

Löysennöksessä tuttuun tapaan ehto $d(u, v) \in \{0, 1\}$ korvataan ehdolla $d(u, v) \geq 0$.

Rajoitteiden lukumäärä on suurempi kuin polkujen $s_i \rightsquigarrow s_j$ lukumäärä, joka yleisesti ottaen ei ole polynominen syötteen koon suhteen. Ennen tähän puuttumista tarkastellaan ohjelman kokonaislukurakoa.

Esimerkki 3.37: Tarkastellaan tähtiverkkoa, jossa on päätesolmujen lisäksi yksi keskussolmu yhdistettynä kuhunkin päätesolmuun kaarella, jonka kapasiteetti on 1.



Monensuuntaiseen leikkaukseen pitää valita ainakin $k - 1$ kaarta.

Murtolukuarvoisessa ratkaisussa riittää asettaa kaikilla kaarilla $d(u, v) = 1/2$, joten kokonaiskustannukseksi tulee $k/2$.

Kokonaisluku- ja murtolukuratkaisujen suhde on sama kuin esitetyn algoritmin approksimointisuhde $2 - 2/k$. \square

Esitämme nyt vaihtoehtoisen muotoilun, joka johtaa kokonaislukurakoon $3/2 - 1/k$ ja approksimointisuhteeseen $3/2$.

Esitimme aiemmin normaalille minimileikkaukselle ($k = 2$) lineaarisen ohjelman, jossa jokaiseen solmuun v liittyi sen **potentiaalia** esittävä muuttuja $0 \leq p_v \leq 1$:

- $p_s = 1$ ja $p_t = 0$, missä s ja t ovat lähde ja nielu
- $d(u, v) = |p_u - p_v|$
- kokonaislukuratkaisussa jos $p_v = 1$, niin v on leikkauksessa samalla puolella kuin s .

Yleistämme tämän monileikkaukseen ottamalla solmun v "potentiaaliksi" vektorin $\mathbf{x}_v \in \Delta_k$, missä

$$\Delta_k = \left\{ \mathbf{x} \in [0, 1] \mid \sum_{i=1}^k x^i = 1 \right\}$$

on $(k - 1)$ -ulotteinen simpleksi.

Nyt päätesolmun s_i potentiaaliksi tulee yksikkövektori $\mathbf{e}_i \in \Delta_k$, missä $e_i^i = 1$ ja $e_i^j = 0$ jos $j \neq i$.

Saamme kokonaislukuohjelman

$$\text{minimoi} \quad \sum_{(u,v) \in E} c(u,v)d(u,v)$$

$$\text{ehdolla} \quad d(u,v) = \frac{1}{2} \sum_{i=1}^k |x_u^i - x_v^i| \quad \text{kaikilla } (u,v) \in E$$

$$\mathbf{x}_v \in \{0,1\}^k \quad \text{kaikilla } v \in V$$

$$\mathbf{x}_{s_i} = \mathbf{e}_i \quad \text{kaikilla } i \in \{1, \dots, k\}.$$

Löysennöksessä ehdon $\mathbf{x}_v \in \{0,1\}^k$ tilalle tulee $\mathbf{x}_v \in \Delta_k$.

Tämä voidaan tosiaan esittää lineaarisena ohjelmana. Ainoa ongelmallinen kohta on kaariin liittyvät rajoitteet, joita varten jokaiseen kaareen (u,v) liitetään k ylimääräistä muuttujaa x_{uv}^i . Ehdoksi tulee

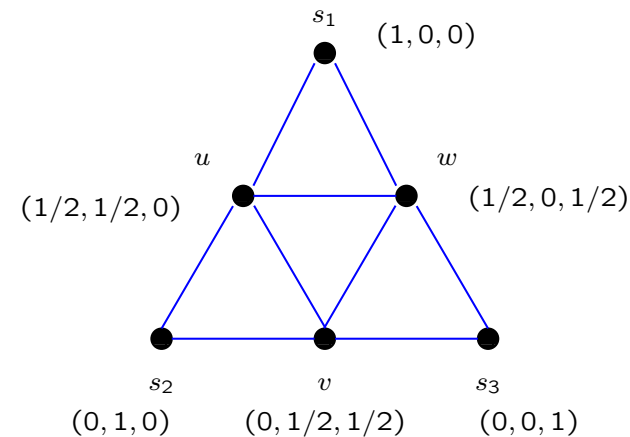
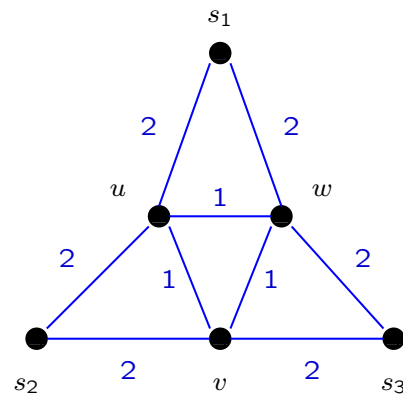
$$x_{uv}^i \geq x_u^i - x_v^i \quad i = 1, \dots, k$$

$$x_{uv}^i \geq x_v^i - x_u^i \quad i = 1, \dots, k$$

$$d(u,v) = \frac{1}{2} \sum_{i=1}^k x_{uv}^i.$$

Saamme ylärajan ongelman kokonaislukuraolle myöhemmin pyöristämiseen perustuvan algoritmin analyysissä. Todetaan tässä, että murtolukuratkaisu voi olla aidosti kokonaislukuratkaisua parempi.

Esimerkki 3.38: Vasemmalla olevan verkon monensuuntaisen leikkauksen minimikustannus on 8. Oikealla esitetty tapa valita muuttujien x_v arvot simpleksistä Δ_3 antaa kustannuksen $7\frac{1}{2}$.



Algoritmi siis tulee perustumaan löysennökselle saadun optimiratkaisun pyöristämiseen. Pyöristämistä helpottaa, jos löysennöksen ratkaisu x toteuttaa seuraavan ehdon:

Ehto A: Millä tahansa kaarella $(u, v) \in E$ vektorit x_u ja x_v eroavat korkeintaan kahden komponentin osalta. \square

Lisäksi joka tapauksessa $x_u, x_v \in \Delta_k$. Siis jos ehdon A vallitessa $x_u \neq x_v$, niin x_u ja x_v eroavat tasan kahden komponentin osalta. Jos nämä komponentit ovat i ja j , niin lisäksi

$$x_v^i - x_u^i = x_u^j - x_v^j.$$

Yleisyyttä rajoittamatta voimme olettaa, että ehto A pätee. Tarkemmin sanottuna osoittautuu, että voimme menetellä seuraavasti:

1. Etsi löysennökselle tarkka ratkaisu.
2. Lisää verkkoon solmuja ja kaaria ja valitse lisättyjä solmuja v vastaaville muuttujille x_v sellaiset arvot, että ehto A tulee voimaan.
3. Muodosta kokonaislukuratkaisu soveltamalla ehtoon A perustuvaa pyöristysmenetelmää.
4. Muodosta alkuperäisen ongelman kokonaislukuratkaisu jättämällä pois lisättyjä solmuja vastaavat muuttujat.

Oleellista on tehdä solmujen lisääminen niin, että optimiratkaisun arvo pysyy samana sekä murtoluku- että kokonaislukuratkaisulle. Lisäksi meidän pitää pystyä muodostamaan

- alkuperäisen ongelman murtolukuratkaisusta samanarvoinen murtolukuratkaisu muunnettuun (ehdon A toteuttavaan) ongelmaan, ja
- muunnetun ongelman kokonaislukuratkaisusta samanarvoinen kokonaislukuratkaisu alkuperäiseen ongelmaan.

Oletetaan nyt, että $(u, v) \in E$ ja vektorit x_u ja x_v eroavat toisistaan useamman kuin kahden komponentin osalta. Korvaamme kaaren (u, v) polulla (u, w_1, \dots, w_m, v) , missä verkkoon on lisätty solmut w_1, \dots, w_m ja $(u, w_1), (w_1, w_2), \dots, (w_m, v)$ sopivalla m . Tämän polun kaikille uusille kaarille asetetaan kapasiteetiksi $c(u, v)$. Selvästi kokonaislukuratkaisun optimiarvo pysyy samana, ja kaari (u, v) tulee alkuperäisen ongelman optimiratkaisuun, jos ja vain jos tasan yksi polun kaarista tulee muunnetun ongelman optimiratkaisuun.

Meidän pitää vielä osoittaa, miten alkuperäisen ongelman murtolukuratkaisusta saadaan samanarvoinen muunnetun ongelman ratkaisu, joka toteuttaa ehdon A.

Polku muodostetaan iteroimalla seuraavaa menettelyä, joka korvaa kaaren (u, v) kahdella kaarella (u, w) ja (w, v) ja antaa muuttujalle \mathbf{x}_w sellaisen arvon, että

- \mathbf{x}_v ja \mathbf{x}_w eroavat toisistaan tasan kahden komponentin osalta
- \mathbf{x}_w ja \mathbf{x}_u eroavat toisistaan pienemmässä määrässä komponentteja kuin \mathbf{x}_v ja \mathbf{x}_u .

Jotta murtolukuratkaisun arvo pysyisi ennallaan, vektori \mathbf{x}_w on valittava siten, että $d(u, v) = d(u, w) + d(w, v)$ missä koko ajan $d(a, b) = \frac{1}{2} \sum_i |x_a^i - x_b^i|$.

Olkoon i komponentti, jolla $|x_u^i - x_v^i|$ saa pienimmän positiivisen arvonsa. Oletetaan $x_u^i < x_v^i$ ja merkitään $\alpha = x_v^i - x_u^i$. Jollain toisella komponentilla j pätee $x_u^j \geq x_v^j + \alpha$. Vektoriin \mathbf{x}_w komponentille i tulee arvoksi x_u^i ja komponentille j arvoksi $x_v^j + \alpha$. Loput komponentit saavat saman arvon kuin vektorissa \mathbf{x}_v . Selvästi halutut ehdot toteutuvat.

Olkoon nyt x löysennöksen optimiratkaisu, ja edellisen nojalla oletamme, että ehto A pätee. Olkoon OPT_f ratkaisun x kustannus. Olkoon E_i niiden kaarten $(u, v) \in E$ joukko, joilla $x_u^i \neq x_v^i$. Ehdon A nojalla jos $d(u, v) \neq 0$, niin (u, v) kuuluu tasan kahteen joukkoon E_i .

Olkoon $W_i = \sum_{(u,v) \in E_i} c(u, v)d(u, v)$. Numeroidaan päätesolmut uudestaan siten, että W_k on näistä arvoista suurin.

Kun $0 \leq \rho \leq 1$, määritellään

$$B(s_i, \rho) = \{v \in V \mid x_v^i \geq \rho\}.$$

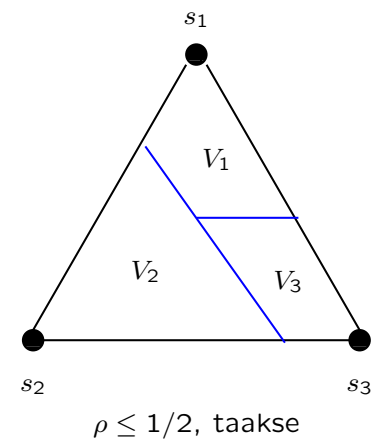
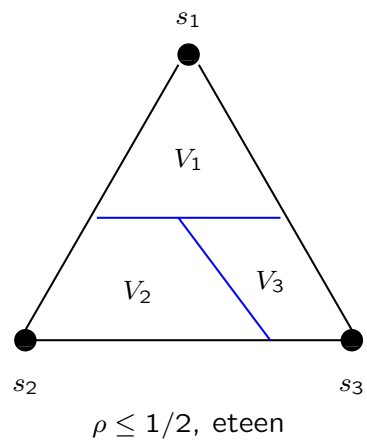
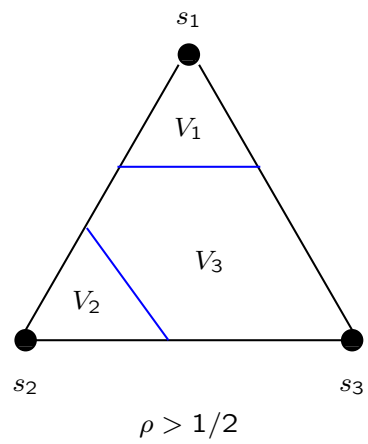
Muodostamme joukon V osituksen V_1, \dots, V_k joukkojen $B(s_i, \rho)$ avulla, kun ensin $0 < \rho < 1$ on valittu satunnaisesti. Näiden joukkojen väliset kaaret muodostavat halutun moninkertaisen leikkauksen.

Jos $\rho > 1/2$, joukot $B(s_i, \rho)$ ovat erillisiä. Asetamme $V_i = B(s_i, \rho)$, kun $i = 1, \dots, k - 1$. Jäljelle jäävät solmut muodostavat joukon V_k .

Jos $\rho \leq 1/2$, muodostamme joukot V_i joko **eteenpäin** tai **taaksepäin**. Valinta näiden kahden vaihtoehdon välillä tehdään satunnaisesti.

Kummassakin tapauksessa jos $x_v^i < \rho$ kaikilla $1 \leq i \leq k - 1$, solmu v tulee joukkoon V_k .

Muuten eteenpäin mennessä solmu v tulee joukkoon V_i **pienimmällä** $i \in \{1, \dots, k - 1\}$, jolla $x_v^i \geq \rho$ eli $v \in B(s_i, \rho)$; taaksepäin mennessä taas **suurimmalla**.



Algoritmi ($\frac{3}{2}$ -approksimaatio monensuuntaiselle leikkaukselle)

1. Muodosta löysennös ja sen tarkka ratkaisu x .
2. Numeroi päätesolmut niin, että $W_k = \max \{ W_1, \dots, W_k \}$.
3. Valitse tasaisen jakauman mukaan satunnainen $0 < \rho < 1$.
Valitse todennäköisyydellä $1/2$ eteen, todennäköisyydellä $1/2$ taakse.
Alusta $V' = V$.
4. Jos eteen, niin toista seuraavaa arvoilla $i = 1, \dots, k - 1$;
jos taakse, niin toista seuraavaa arvoilla $i = k - 1, \dots, 1$:

$$\begin{aligned} V_i &\leftarrow V' \cap B(s_i, \rho) \\ V' &\leftarrow V' - V_i \end{aligned}$$

5. Aseta $V_k \leftarrow V'$.
6. Valitse joukkoon C kaaret, joiden päätepisteet eivät kuulu samaan joukkoon V_i . Palauta C .

Selvästi algoritmin tuottama C on monensuuntainen leikkaus. Osoitamme, että $\mathbf{E}[c(C)] \leq (3/2 - 1/k)\text{OPT}_f$.

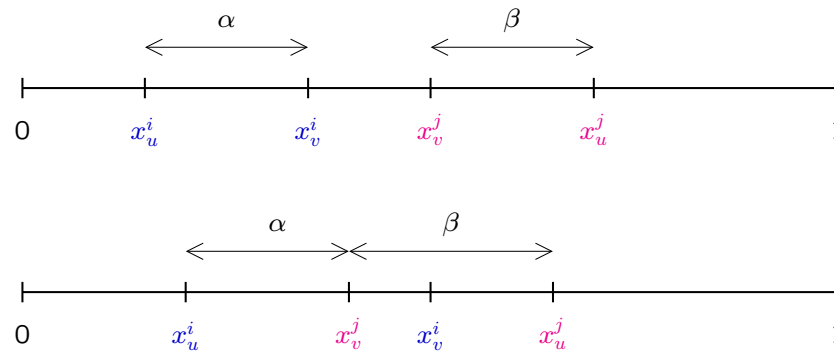
Lemma 3.39: Jos $e \in E - E_k$, niin $\Pr[e \in C] \leq \frac{3}{2}d(e)$. Jos $e \in E_k$, niin $\Pr[e \in C] \leq d(e)$.

Todistus: Olkoon $e = (u, v)$. Jos $d(u, v) = 0$, niin $x_u = x_v$, jolloin u ja v päätyvät aina samaan joukkoon V_i .

Oletetaan siis $d(u, v) > 0$, ja tarkastellaan ensin tapausta $(u, v) \in E - E_k$. Olkoot i ja j ne komponentit, joilla x_u ja x_v poikkeavat toisistaan.

Oletetaan, että x_u^i on pienin arvoista x_u^i , x_u^j , x_v^i ja x_v^j . Siis $d(u, v) = x_u^j - x_v^j = x_v^i - x_u^i$.

Määrittelemme välit $\beta = [x_v^j, x_u^j]$ ja $\alpha = [x_u^i, x_v^i] - \beta$. Kaksi eri tapausta sen mukaan, onko $x_v^i \leq x_v^j$:



Jotta u ja v päätyisivät eri joukkoihin, pitää ensinnäkin olla $\rho \in \alpha \cup \beta$.

Lisäksi tapauksessa $\rho \in \alpha$ kysymyksessä on oltava "eteen"-tapaus ja $j < i$, tai "taakse"-tapaus ja $j > i$. Muuten u ja v päätyvät molemmat joukkoon V_j , elleivät ole aiemmin päätyneet johonkin muuhun joukkoon. Tämän todennäköisyys on $1/2$.

Siis todennäköisyys päätyä eri joukkoihin on korkeintaan

$$|\beta| + \frac{1}{2} |\alpha| \leq \frac{3}{2} d(u, v),$$

missä $|\cdot|$ on osavälin pituus.

Tarkastellaan sitten tapausta $(u, v) \in E_k$. Siis x_u ja x_v poikkeavat komponenteilla k ja jollain $i \neq k$. Koska joukko V_k joka tapauksessa kokoaa kaikki jäljelle jääneet solmut, niin u ja v päätyvät samaan joukkoon, ellei ρ satu arvojen x_u^i ja x_v^i väliin. Tämän todennäköisyys on $d(u, v)$. \square

Lemma 3.40: Algoritmin tuottama monensuuntainen leikkaus C toteuttaa ehdon

$$\mathbf{E}[c(C)] \leq \left(\frac{3}{2} - \frac{1}{k}\right) \text{OPT}_f.$$

Todistus: Siis $\text{OPT}_f = \sum_{e \in E} c(e)d(e)$. Koska jokainen $e \in E$ kuuluu tasan kahteen joukkoon E_i , pätee

$$\sum_{i=1}^k W_i = \sum_{i=1}^k \sum_{e \in E_i} c(e)d(e) = 2 \cdot \text{OPT}_f.$$

Koska numerointi valittiin niin, että W_k on suurin, pätee

$$W_k \geq \frac{2}{k} \cdot \text{OPT}_f.$$

Soveltamalla edellistä lemmaa saamme nyt

$$\begin{aligned}\mathbf{E}[c(C)] &= \sum_{e \in E} c(e) \Pr[e \in C] \\ &= \sum_{e \in E - E_k} c(e) \Pr[e \in C] + \sum_{e \in E_k} c(e) \Pr[e \in C] \\ &\leq \frac{3}{2} \sum_{e \in E - E_k} c(e) d(e) + \sum_{e \in E_k} c(e) d(e) \\ &= \frac{3}{2} \sum_{e \in E} c(e) d(e) - \frac{1}{2} \sum_{e \in E_k} c(e) d(e) \\ &\leq \left(\frac{3}{2} - \frac{1}{k} \right) \text{OPT}_f. \quad \square\end{aligned}$$

Yllä esitetystä seuraa kokonaislukuraolle yläraja $3/2 - 1/k$. Paras tunnettu yläraja on 1,3438. Esimerkin 3.38 perusteella kokonaislukurako voi olla ainakin $16/15$; paras tunnettu alaraja on $8/(7 + 1/(k - 1))$.

Edellinen odotusarvoa koskeva virheraja voidaan standarditekniikoilla muuntaa suurella todennäköisyydellä päteväksi.

Lause 3.41: Olkoon $0 < \delta < 1$. Toistetaan edellistä algoritmia $N = \lceil 2k \ln(1/\delta) \rceil$ kertaa. Olkoon C^* saaduista monensuuntaisista leikkauksista paras. Ainakin todennäköisyydellä $1 - \delta$ pätee $c(C^*) \leq \frac{3}{2} \text{OPT}_f$.

Todistus: Sovelletaan Markovin epäyhtälöä $\Pr[X \geq t] \leq E[X]/t$ ei-negatiiviseen satunnaismuuttujaan $c(C)$, missä C on algoritmin yhden suorituskerran tulos. Saamme

$$\Pr \left[c(C) \geq \frac{3}{2} \text{OPT}_f \right] \leq \frac{(3/2 - 1/k) \text{OPT}_f}{(3/2) \text{OPT}_f} = 1 - \frac{2}{3k} \leq 1 - \frac{1}{2k}.$$

Todennäköisyys, että rajan saavuttaminen epäonnistuu N kertaa peräkkäin, on korkeintaan

$$\left(\left(1 - \frac{1}{2k} \right)^{2k} \right)^{\ln(1/\delta)} \leq (e^{-1})^{\ln(1/\delta)} = \delta,$$

missä on käytetty tietoa $(1 - 1/n)^n \leq e^{-1}$. \square

Monensuuntaisesta leikkauksesta on helposti nähtävä approksimaation säilyttävä palautus seuraavaan **monensuuntaiseen solmuleikkaukseen**:

Tapaus: verkko $G = (V, E)$, solmujen painofunktio $c: V \rightarrow \mathbb{Q}_+$ ja päätesolmujen joukko $S = \{s_1, \dots, s_k\} \subset V$, joka oletetaan riippumattomaksi

Käyvät ratkaisut: solmumonileikkaukset eli sellaiset joukot $C \subseteq V$, että kaikilla $i \neq j$ jokainen polku $s_i \rightsquigarrow s_j$ sisältää ainakin yhden solmun joukosta C

Kustannusfunktio: minimoitava solmumonileikkauksen kustannus $\sum_{v \in C} c_v$.

Osoitamme, että tätä vastaavalla kokonaislukuohjelmalla on aina puolikokonaislukuarvoinen optimiratkaisu. Tästä saadaan pyöristämällä suoraan 2-approksimointialgoritmi. Algoritmia hieman viilaamalla päästään approksimointisuhteeseen $2 - 2/k$.

Kokonaislukuohjelman muuttuja d_v kertoo, onko solmu v mukana monileikkauksessa. Olkoon lisäksi \mathcal{P} kaikkien polkujen joukko, jotka yhdistävät joitain kahta päätesolmua:

$$\begin{array}{ll} \text{minimoi} & \sum_{v \in V-S} c_v d_v \\ \text{ehdolla} & \sum_{v \in p} d_v \geq 1 \quad \text{kaikilla poluilla } p \in \mathcal{P} \\ & d_v \in \{0, 1\} \quad \text{kaikilla } v \in V - S. \end{array}$$

Löysennöksessä taas ehto $d_v \in \{0, 1\}$ korvataan ehdolla $d_v \geq 0$.

Muuttujilla d_v on taas etäisyystulkinta. Polun p pituudeksi tulkitaan $\sum_{v \in p} d_v$, ja kahden solmun etäisyys on lyhimmän niiden välisen polun pituus.

Löysennöksestä on syytä huomata, että rajoitteiden lukumäärä tyypillisesti ei ole polynominen. On kuitenkin helppo muodostaa polynomisessa ajassa toimiva [separointioraakkeli](#). Lineaarisen ohjelman separointioraakkeli saa syötteenä mahdollisen ratkaisun x ja palauttaa

- arvon "käypä", jos x toteuttaa kaikki rajoitteet
- muuten jonkin rajoitteen, jota x rikkoo.

Tässä tapauksessa separointioraakkelin riittää etsiä lyhimmät polut kaikkien päätesolmuparien välillä ja tarkastaa, ovatko ne pituudeltaan ainakin 1.

Separointioraakkeli riittää ohjelman ratkaisemiseen ellipsoidimenetelmällä polynomisessa ajassa (mutta ei välttämättä käytännössä kovin tehokkaasti).

Duaalin voidaan tulkita esittävän monihyödykevuota. Jokaista polkua $p \in \mathcal{P}$ vastaa muuttuja f_p , joka voidaan tulkita polkua pitkin kulkevaksi vuoksi. Polun päätepisteet kertovat, mistä hyödykkeestä on kysymys. Solmun kustannus rajoittaa solmun läpi kulkevaa kokonaisvuota:

$$\begin{array}{ll} \text{maksimoi} & \sum_{p \in \mathcal{P}} f_p \\ \text{ehdolla} & \sum_{p: v \in p} f_p \leq c_v \quad \text{kaikilla } v \in V - S \\ & f_p \geq 0 \quad \text{kaikilla } p \in \mathcal{P}. \end{array}$$

Olkoon d löysennöksen optimiratkaisu. Osoitamme, miten siitä saadaan yhtä hyvä puolikokonaislukuarvoinen ratkaisu. Todistusta varten olkoon lisäksi f duaalin optimiratkaisu. Komplementaarisuusehtojen perusteella

- jos $d_v > 0$, niin solmu v on kyllästetty (kokonaisvuo sen läpi on c_v)
- jos $f_p > 0$, niin polun p pituus on tasan 1.

Tarkastellaan verkossa G ratkaisun d mukaisia etäisyyksiä. Kun $i = 1, \dots, k$, määritellään alue S_i koostumaan solmuista, joiden etäisyys solmusta s_i on nolla (s_i itse mukaanlukien). Alueen reunus on joukko

$$B_i = \{ u \notin S_i \mid (u, v) \in E \text{ jollain } v \in S_i \}.$$

Käyppyysehtojen takia

- $S_i \cap S_j = \emptyset$, kun $i \neq j$
- $s_i \notin B_j$ kaikilla i, j
- jos $v \in B_i \cap B_j$ joillain $i \neq j$, niin $d_v = 1$.

Määritellään reunussolmujen joukko $M = \cup_i B_i$ ja ositetaan se kahteen joukkoon:

- Joukko M^{int} sisältää ne solmut, jotka kuuluvat ainakin kahteen eri reunukseen B_i ja B_j . Näillä solmuilla v siis pätee $d_v = 1$.
- Joukko M^{disj} sisältää ne solmut, jotka kuuluvat tasan yhteen reunukseen B_i .

Voimme nyt todistaa seuraavan:

Lemma 3.42: Olkoon p päätesolmujen s_i ja s_j välinen polku, jolla $f_p > 0$. Polku p sisältää

- joko tasan yhden solmun joukosta M^{int}
- tai tasan kaksi solmua joukosta M^{disj}

eikä muita joukon M solmuja.

Mikä tahansa päätesolmuja s_i ja s_j yhdistävä polku sisältää ainakin yhden solmun joukosta B_i ja yhden joukosta B_j (mutta nämä voivat olla sama solmu).

Komplementaarisuusehtojen takia polun p pituus on tasan 1, joten jos se sisältää yhden solmun joukosta M^{int} , se ei voi sisältää muita solmuja joukosta M .

Tehdään vastaoletus, että polku p sisältää ainakin kolme joukon M^{disj} solmua. Voimme siis valita reunuksen B_k ja sellaiset kolme eri solmua u , v ja w , että $u \in B_i$, $v \in B_k$ ja $w \in B_j$. (Mahdollisesti joko $k = i$ tai $k = j$.)

Tarkastellaan kahta polkua:

- solmusta s_i polkua p solmuun v ja siitä lyhintä reittiä solmuun s_k
- solmusta s_j polkua p solmuun v ja siitä lyhintä reittiä solmuun s_k .

Reunuksen määritelmän perusteella lyhin reitti solmusta v solmuun s_k sisältää solmun v lisäksi vain nolla-arvoisia solmuja. Jos $k \neq i$, niin ensimmäinen polku yhdistää päätesolmut s_i ja s_k . Sen pituus on alle 1, koska polku p sisältää kaikki sen positiivisarvoiset solmut ja lisäksi positiivisarvoisen solmun w . Tämä on vastoin käyppöyseyhteistä; ristiriita. Tapaus $k \neq j$ käsitellään samoin. \square

Lemma 3.43: Muodostetaan ratkaisu h asettamalla $h_v = 1$ jos $v \in M^{\text{int}}$, $h_v = 1/2$ jos $v \in M^{\text{disj}}$ ja $h_v = 0$ muuten. Tämä h on optimiratkaisu.

Todistus: Jokainen kahden päätesolmun välinen polku sisältää ainakin kaksi solmua u ja v , joilla $h_u = h_v = 1/2$, tai yhden solmun u , jolla $h_u = 1$. Siis h on käypä ratkaisu.

Osoitamme, että kohdefunktion arvo ratkaisulla h on sama kuin duaalin arvo optimiratkaisulla f . Tästä seuraa haluttu tulos, että h on optimaalinen.

Edellisen lemmän mukaan polut $p \in \mathcal{P}$, joilla $f_p > 0$, voidaan jakaa kahteen erilliseen luokkaan:

- luokan \mathcal{P}_1 polut sisältävät yhden solmun joukosta M^{int}
- luokan \mathcal{P}_2 polut sisältävät kaksi solmua joukosta M^{disj} .

Komplementaarisuusehtojen takia jokainen joukon M solmu on vuossa f kyllästetty. Siis polkujoukon \mathcal{P}_1 kokonaisvuo on

$$\sum_{v \in M^{\text{int}}} c_v$$

ja polkujoukon \mathcal{P}_2 kokonaisvuo

$$\frac{1}{2} \sum_{v \in M^{\text{disj}}} c_v.$$

Siis kokonaisvuo on

$$\sum_{v \in M^{\text{int}}} c_v + \frac{1}{2} \sum_{v \in M^{\text{disj}}} c_v = \sum_{v \in V-S} h_v c_v,$$

eli h on optimiratkaisu. \square

Korollaari 3.44: Monensuuntaisella solmuleikkauksella on 2-aproksimointialgoritmi.

Todistus: Ratkaistaan löysennös, muunnetaan ratkaisu puolikokonaislukuarvoiseksi edellä esitetyllä menetelmällä ja pyöristetään ylöspäin. \square

Monileikkaus ja monihyödykevuoto

Tarkastelemme nyt yleisessä verkossa samoja ongelmia, joita edellä (s. 208–222) tarkasteltiin puissa. On siis annettu verkko $G = (V, E)$, kaarille e kapasiteetit c_e sekä k solmuparia $\{(s_1, t_1), \dots, (s_k, t_k)\}$.

Monihyödykevuoto-ongelmassa reititetään kustakin lähdesolmusta s_i vastaavaan maalisolmuun t_i vuo f_i hyödykettä i siten, että jokaista kaarta e pitkin kulkee kaikkia hyödykkeitä yhteensä korkeintaan c_e yksikköä. Tehtävänä on maksimoida kokonaisvuo $\sum_{i=1}^k f_i$.

Monileikkausongelmassa valitaan monileikkaus $C \subseteq E$, jonka on sisällettävä kaikilla solmupareilla (s_i, t_i) ainakin yksi kaari jokaiselta polulta $s_i \rightsquigarrow t_i$. Tehtävänä on minimoida monileikkauksen kapasiteetti $c(C) = \sum_{e \in C} c_e$.

Puiden tapauksessa saimme monileikkausongelmalle 2-aproksimointialgoritmin. Nyt esitämme yleisille verkoille $O(\log k)$ -aproksimointialgoritmin.

Olkoon \mathcal{P} kaikkien niiden polkujen joukko, jotka yhdistävät jotain solmuparia (s_i, t_i) . Polun päätepisteet kertovat, mikä hyödyke sitä pitkin kulkee. Voimme siis esittää monihyödykevuon maksimoimisen lineaarisena ohjelmana

$$\begin{array}{ll}
 \text{maksimoi} & \sum_{p \in \mathcal{P}} f_p \\
 \text{ehdolla} & \sum_{p: e \in P} f_p \leq c_e \quad \text{kaikilla } e \in E \\
 & f_p \geq 0 \quad \text{kaikilla } p \in \mathcal{P}.
 \end{array}$$

Muuttujien f_p lukumäärä ei yleensä ole polynominen. Koska tarvitsemme tätä ohjelmaa vain apuna sen duaalin tarkastelemisessa, tämä ei sinänsä ole ongelma.

Duaaliksi saamme

minimoi

$$\sum_{e \in E} c_e d_e$$

ehdolla

$$\sum_{e \in p} d_e \geq 1$$

kaikilla poluilla $p \in \mathcal{P}$

$$d_e \geq 0$$

kaikilla $e \in E$.

Tässä tuttuun tapaan tulkitsemme arvon d_e kaaren e pituudeksi, ja kahden solmun etäisyys on lyhimmän niiden välisen polun pituus. Vaatimus on, että solmujen s_i ja t_i etäisyys on ainakin 1 kaikilla i .

Edellinen lineaarinen ohjelma määrittelee murtolukuarvoisen monileikkauksen minimointiongelman. Varsinainen monileikkausongelma saadaan asettamalla kokonaislukuarvorajoite $d_v \in \{0, 1\}$.

Osoitamme, että murtolukuarvoinen monileikkaus voidaan pyöristää kokonaislukuarvoiseksi siten, että kapasiteetti kasvaa korkeintaan kertoimella $O(\log n)$. Tästä seuraa $O(\log n)$ -approksimointialgoritmi.

Edellisen sivun lineaarisessa ohjelmassa tosin rajoitteiden lukumäärä ei yleensä ole polynominen, mutta voimme

- ratkaista sen polynomisessa ajassa separointioraakkelia ja ellipsoidimenetelmää käyttäen (vrt. s. 247) tai
- muotoilla monihyödykevuon-ongelman uudelleen esittelemällä jokaiselle kaarelle e ja hyödykkeelle i oman muuttujan $f_{e,i}$ (harjoitustehtävä), jolloin sen muuttujien lukumäärä saadaan polynomiseksi.

Olkoon f monihyödykevuon-ohjelman optimiratkaisu ja d optimaalinen murtolukuarvoinen monileikkaus. Olkoon $F = \sum_{e \in E} c_e d_e$ murtolukumonileikkauksen minimiarvo, ja samalla siis monihyödykevuon maksimiarvo.

Olkoon $D = \{e \in E \mid d_e > 0\}$ niiden kaarten joukko, joiden pituus on nolosta poikkeava. Joukko D on selvästi monileikkaus, mutta voi olla kapasiteetiltaan paljon suurempi kuin F . Meidän pitää siis valita sopiva osajoukko joukosta D .

Määritellään kaaren e **painoksi** $c_e d_e$ ja solmujen u ja v etäisyydeksi (kaaripituuksien d_e mielessä) $\text{dist}(u, v)$. Siis erityisesti F on kaikkien kaarten yhteispaino.

Solmujoukolle $S \subseteq V$ olkoon $\delta(C)$ leikkauksen (S, \bar{S}) kaarien joukko ja $c(S) = \sum_{e \in \delta(S)} c_e$ tämän leikkauksen kapasiteetti.

Algoritmin runko on seuraava:

1. Alusta $M \leftarrow \emptyset$.
2. Valitse jokin solmupari (s_i, t_i) , jonka välillä on polku.
3. Alusta alueeseen S yksi solmu $S = \{s_i\}$. Kasvata aluetta sopivasti, mutta pysäytä, ennen kuin t_i tulee liitetyksi siihen.
4. Aseta $M \leftarrow M \cup \delta(S)$. Poista verkosta alue S kaarineen ja myös joukon $\delta(S)$ kaaret.
5. Jos verkossa ei ole jäljellä yhtään solmuparia (s_i, t_i) , palauta M . Muuten palaa kohtaan 2.

Ongelmana on tietysti toteuttaa alueen S kasvattaminen siten, että leikkaukseen tulevien kaarten kapasiteetti $c(S)$ tulee mahdollisimman pieneksi verrattuna niiden kaarten kapasiteettiin, joiden kumpikin päätepiste on joukossa S ja jotka siis jäävät pois.

Esitämme alueen kasvatuksen perusajatuksen käyttäen jatkuva-arvoista prosessia. Muokkaamme tästä sitten laskennallisesti tehokkaamman diskreetin prosessin.

Lähtökohtana on, että valitsemme alueen S juureksi jonkin lähdesolmun s_i . Olkoon $S(r)$ niiden solmujen joukko, joiden etäisyys solmusta s_i on korkeintaan r .

Jos $u, v \in S(r)$, niin $\text{dist}(u, v) \leq 2r$. Käyppyysehtojen takia $\text{dist}(s_j, t_j) \geq 1$ kaikilla j . Siis jos pidämme säteen r pienempänä kuin $1/2$, alueeseen ei tule yhtään lähde-nielusolmuparia.

Perusajatus on nyt määritellä $\widetilde{wt}(S(r))$ niiden kaarten kokonaispainoksi, joiden kumpikin päätepiste on joukossa $S(r)$. Haluaisimme osoittaa, että jollain arvolla $r < 1/2$ pätee $c(S(r)) \leq \varepsilon \widetilde{wt}(S(r))$, missä $\varepsilon = O(\log k)$. Pysäytämme alueen kasvattamisen tähän arvoon r . Tällöin algoritmin jokaisella kierroksella leikkaukseen M lisättävien kaarten paino on korkeintaan ε kertaa poistettavien kaarten paino. Tästä seuraa haluttu tulos, että leikkauksen lopullinen paino on korkeintaan ε kertaa kaikkien kaarten yhteispaino.

Teknisistä syistä on kuitenkin tarkoituksenmukaista

1. ottaa sopiva nollasta poikkeava alkupaino joukolle $S(0) = \{s_i\}$
2. ottaa painossa $\widetilde{\text{wt}}(S(r))$ osittaisina huomioon myös ne kaaret, joista vain toinen päätepiste on joukossa $S(r)$.

Tarkastellaan siis tilannetta säteen ollessa r . Määritellään $q_e = 1$, jos kaaren $e = (u, v)$ kumpikin päätepiste on joukossa $S(r)$. Jos $u \in S(r)$ ja $v \notin S(r)$, asetetaan

$$d_e = \frac{r - \text{dist}(s_i, u)}{\text{dist}(s_i, v) - \text{dist}(s_i, u)}.$$

Jos $u \notin S(r)$ ja $v \notin S(r)$, niin $d_e = 0$. Alueen $S(r)$ painoksi asetetaan nyt

$$\text{wt}(S(r)) = \frac{F}{k} + \sum_e c_e d_e q_e.$$

Lemma 3.45: Jos valitaan $\varepsilon = 2 \ln(k + 1)$, niin ehto

$$c(S(r)) \leq \varepsilon \text{wt}(S(r))$$

pätee jollain $r < 1/2$.

Todistus: Suure $\text{wt}(S(r))$ on parametrin r jatkuva funktio, ja derivoituva lukuunottamatta niitä parametrin r arvoja, joilla jokin solmu on tasan etäisyydellä r solmusta s_i . Derivaataksi saadaan

$$\begin{aligned} \frac{d \text{wt}(S(r))}{d r} &= \frac{d}{d r} \sum_{e \in E} c_e d_e q_e \\ &= \sum_{e \in E} c_e d_e \frac{d q_e}{d r} \\ &= \sum_{u \in S(r), v \notin S(r), (u,v) \in E} c_e d_e \frac{1}{\text{dist}(s_i, v) - \text{dist}(s_i, u)} \\ &\geq \sum_{e \in \delta(S(r))} c_e \\ &= c(S(r)). \end{aligned}$$

Tehdään nyt vastaoletus, että kaikilla $r < 1/2$ pätee $c(S(r)) > \varepsilon \text{wt}(S(r))$.
Siis

$$\frac{d \text{wt}(S(r))}{d r} > \varepsilon \text{wt}(S(r))$$

eli

$$\frac{1}{\text{wt}(S(r))} \cdot \frac{d \text{wt}(S(r))}{d r} = \frac{d}{d r} \ln \text{wt}(S(r)) > \varepsilon.$$

Olkoot $0 = r_0 < r_1 < r_2 < \dots < r_m$ ne etäisyydet $\text{dist}(s_i, v)$, jotka ovat alle $1/2$, ja $r_{m+1} = 1/2$. Integroimalla saadaan

$$\int_{r_j}^{r_{j+1}} \frac{d}{d r} \ln \text{wt}(S(r)) d r > \int_{r_j}^{r_{j+1}} \varepsilon d r$$

eli

$$\ln \text{wt}(S(r_{j+1})) - \ln \text{wt}(S(r_j)) > (r_{j+1} - r_j) \varepsilon$$

ja edelleen laskemalla yhteen arvoilla $j = 0, \dots, m$

$$\ln \text{wt}(S(r_{m+1})) - \ln \text{wt}(S(r_0)) = \ln \frac{\text{wt}(S(1/2))}{\text{wt}(S(0))} > (r_{m+1} - r_0) \varepsilon = \frac{\varepsilon}{2}.$$

Koska $wt(S(0)) = F/k$ ja $wt(S) \leq F/k + F$ kaikilla S , saadaan lopulta

$$\ln \frac{F + F/k}{F/k} = \ln(k + 1) > \frac{\varepsilon}{2}.$$

Tämä on ristiriita, sillä olimme valinneet $\varepsilon = 2 \ln(k + 1)$. \square

Intuitiivisesti alueen painon $wt(S(r))$ kasvunopeus on verrannollinen sen reunalla olevien kaarten painoon $c(S(r))$. Jos reunan paino pysyy vähintään vakiosuhteessa alueen painoon, alueen painon kasvunopeus on eksponentiaalinen. Tätä ei tietenkään voi jatkua kovin pitkään.

Edellä esitetty jatkuva prosessi voidaan tietenkin korvata diskreetillä prosessilla, joka tarkastelee vain niitä säteen r arvoja, joilla joukko S muuttuu. Nämä arvot saadaan laskemalla lyhimmät polut solmusta s_1 lähtien.

Diskreetissä prosessissa tarkastelemme samaa leikkauksen kapasiteettia $c(S)$ kuin aiemminkin. Alueen S painoksi muutetaan

$$wt(S) = \frac{F}{k} + \sum_e c_e d_e,$$

missä summa on yli kaarten, joilla on ainakin toinen päätepiste joukossa S . Paino siis ei ainakaan pienene jatkuvaan prosessiin verrattuna, joten ehto $c(S) \leq \epsilon wt(S)$ saavutetaan ainakin yhtä pienellä säteellä kuin jatkuvassa tapauksessa.

Kaikkiaan algoritmi on seuraava:

Algoritmi ($O(\log k)$ -approksimaation monileikkaukselle)

1. Muodosta lineaarinen löysennös (s. 257). Ratkaise se ellipsoidimenetelmällä, jolloin saat kaarten pituudet d_e .
2. Aseta $\varepsilon = 2 \ln(k + 1)$. Alusta $i \leftarrow 1$, $G_1 \leftarrow G$ ja $M \leftarrow \emptyset$.
3. Jos verkossa G_i ei ole polkua minkään lähde-nielusolmuparin välillä, palauta M .
4. Valitse jokin pari (s_j, t_j) , joiden välillä on polku. Laajenna solmun s_j ympärille sellainen alue S_i , että $c_{G_i}(S_i) \leq \varepsilon \text{wt}_{G_i}(S_i)$, missä c_{G_i} ja wt_{G_i} tarkoittavat verkon G_i mukaan laskettuja suureita c ja wt .
5. Aseta $M \leftarrow M \cup \delta_{G_i}(S)$.
6. Aseta verkoksi G_{i+1} verkko G_i , josta on poistettu alueen S_i solmut.
7. Aseta $i \leftarrow i + 1$ ja palaa kohtaan 3.

Lemma 3.46: Algoritmin palauttama M on monileikkaus.

Todistus: Lemman 3.45 ja sitä edeltävien kommenttien perusteella jatkuva prosessi alkuperäisessä verkossa G ei sisällytä joukkoon S yhtään lähde-nielusolmuparia. Siirtyminen osaverkkoon G_i ei ainakaan lyhennä mitään etäisyyksiä, joten sama pätee osaverkossa. Diskreetti prosessi loppuu vähintään yhtä nopeasti kuin jatkuva ja toteuttaa siis saman ehdon.

Algoritmin tuottaman kaarijoukon M poistaminen jakaa sen komponentteihin S_i ja jäljellejäävään osaan, joista mikään ei siis sisällä lähde-nielusolmuparia. \square

Lemma 3.47: $c(M) \leq 2\varepsilon F = 4 \ln(k + 1)F$.

Todistus: Koska joukon S_i solmut (ja siis niihin liittyvät kaaret) poistetaan aina verkosta, mikään kaari ei tule lisätyksi leikkaukseen M useita kertoja. Samoin jokainen kaari lasketaan korkeintaan yhteen painoon $\text{wt}_{G_i}(S_i)$.

Iteraatioiden lukumäärä on korkeintaan k , sillä jokainen iteraatio katkaisee ainakin yhden lähde-nielusolmuparin yhteyden. Siis

$$c(M) = \sum_i c_{G_i}(S_i) \leq \sum_i \varepsilon \text{wt}_{G_i}(S_i) \leq \varepsilon \left(k \cdot \frac{F}{k} + \sum_{e \in E} c_e d_e \right) = 2\varepsilon F. \quad \square$$

Lause 3.48: Edellinen algoritmi takaa approksimointisuhteen $O(\log k)$ monileikkaukselle.

Todistus: Seuraa suoraan edellisistä lemmoista, sillä murtolukuleikkauksen optimi F on alaraja myös kokonaislukuongelman arvolle. \square

Palauttamalla mieliin alkuperäinen duaalioingelman saadaan

Korollari 3.49: Kun $|F|$ on k -hyödykevuon maksimi ja $|C|$ k -leikkauksen minimi, niin

$$|F| \leq |C| \leq O(\log k) |F|.$$

Edellisestä algoritmista seuraava raja $O(\log k)$ kokonaislukuraolle on tiukka.

Tämän osoittamiseksi käytämme **laajentavia** verkkoja (expander). Verkko $G = (V, E)$ on laajentava, jos solmujoukolla S pätee

$$|\delta(S)| > \min \{ |S|, |\bar{S}| \}$$

aina, kun $S \neq \emptyset$ ja $S \neq V$. Laajentavien verkkojen olemassaolo ei ole aivan ilmeistä, mutta itse asiassa satunnainen tasa-asteinen verkko asteluvulla $d \geq 3$ on suurella todennäköisyydellä laajentava (mitä emme tässä todista).

Esimerkki 3.50: Olkoon H tasa-asteinen k solmun laajentava verkko asteluvulla $d \geq 3$. Muodostetaan monileikkausongelma asettamalla jokaisen kaaren kapasiteetiksi 1 ja valitsemalla päätesolmupareiksi (s_i, t_i) kaikki sellaiset solmuparit, joiden etäisyys on ainakin $\alpha = \lfloor \log_d(k/2) \rfloor$.

Muodostamalla leveyssuuntainen puu nähdään, että solmua s_i lähempänä kuin α on korkeintaan

$$1 + d + d^2 + \dots + d^{\alpha-1} < d^\alpha \leq \frac{k}{2}$$

solmua. Siis päätesolmupareja tulee ainakin $k \cdot (k/2) = \Theta(k^2)$.

Verkossa on kaikkiaan $dk/2$ kaarta ja siis yhteensä kapasiteetti $O(k)$. Yhden yksikön kuljettaminen päätesolmujen s_i ja t_i välillä vie ainakin kapasiteetin $\alpha = \Theta(\log k)$. Siis maksimivuo on $O(k/\log k)$. Tämä on samalla murtolukumonileikkauksen minimi. Osoitamme, että kokonaislukuarvoisen monileikkauksen M kapasiteetti on $\Omega(k)$.

Tarkastellaan jäljelle jääviä yhtenäisiä komponentteja, kun verkosta H poistetaan kaaret M . Nyt missään yhtenäisessä komponentissa ei ole enempää kuin $k/2$ solmua. Tarkastellaan nimittäin mielivaltaisen päätesolmun s_i sisältävää komponenttia. Saman komponentin muiden solmujen etäisyys solmusta s_i on korkeintaan $\alpha - 1$, sillä muuten komponentissa olisi päätesolmupari. Kuten edellä todettiin, etäisyydellä korkeintaan $\alpha - 1$ on korkeintaan $k/2$ solmua.

Edellisen perusteella komponentille S pätee $|S| \leq |\bar{S}|$. Koska H on laajentava verkko, pätee $|\delta(S)| \geq |S|$. Laskemalla yhteen komponenttien yli saadaan $\sum_S |\delta(S)| \geq k$. Monileikkauksen M kaari tulee mukaan korkeintaan kahteen leikkaukseen $\delta(S)$, joten $|M| = \Omega(k)$, kuten väitettiin.

Edellä päätesolmujen lukumäärä oli $\Theta(|V|^2)$. Jos halutaan esimerkki mielivaltaisilla parametreilla k ja $n = |V|$, voidaan korvata edellisessä esimerkissä jokin kaari polulla, jossa on sopiva määrä yksikköpainoisia kaaria. Tämä ei vaikuta vuo- ja leikkausongelmien arvoihin, ja solmujen lukumäärä voidaan valita halutuksi. \square

Esimerkki 3.51: Sovelluksena tarkastelemme $2\text{CNF}\equiv$ -klausuulien poisto-ongelmaa. Annettuna on joukko klausuuleja $p \equiv q$, missä p ja q ovat literaaleja. Klausuuleille on annettu painot. Tehtävänä on poistaa painoltaan pienin määrä klausuuleja siten, että jäljellejäävät toteutuvat jollain muuttujien totuusarvoasetuksella.

Muodostamme n muuttujan ongelmasta $2n$ solmun verkon, jonka solmuja ovat muuttujat x ja niiden negaatiot \bar{x} . Jokaista klausuulia $p \equiv q$ kohti verkkoon tulee kaksi kaarta, (p, q) ja (\bar{p}, \bar{q}) . Kummankin kaaren kapasiteetti on sama kuin klausuulin (p, q) paino.

Klausuulit (p, q) ja (\bar{p}, \bar{q}) ovat ekvivalentit. Jos ongelmassa on tällainen ekvivalentti pari klausuuleja, voimme poistaa toisen ja kasvattaa toisen painoa vastaavasti. Oletamme siis yleisyyttä rajoittamatta, että tällaisia pareja ei ole.

Toteamme nyt, että kaava on toteutuva, jos ja vain jos mikään verkon komponentti ei sisällä muuttujaa ja sen negaatiota.

Selvästi jos muuttujasta on polku sen negaatioon, kaava on ristiriitainen.

Oletetaan toisaalta, että missään komponentissa ei esiinny muuttujaa ja sen negaatiota. Valitaan mielivaltainen komponentti S ja asetetaan sen literaalien arvoksi "tosi"; tästä ei vielä tullut ristiriitaa. Asetetaan nyt arvo "epätosi" kaikille literaaleille p , joilla \bar{p} on komponentissa S , sekä tällaisten literaalien kanssa samassa komponentissa oleville literaaleille. Edelleen saman arvon antaminen samassa komponentissa oleville literaaleille ei aiheuta ristiriitaa. Voimme jatkaa näin, kunnes toteuttava totuusarvoasetus on valmis.

Muodostetaan nyt monileikkausongelma valitsemalla päätesolmupareiksi kaikki n paria (p, \bar{p}) .

Olkoon M monileikkaus ja sen kapasiteetti $c(M)$. Muodostetaan kaava F' poistamalla alkuperäisestä kaavasta F joukon M kaaria vastaavat klausuulit. Kaavaa F' vastaavassa verkossa ei ole polkuja parien (p, \bar{p}) välillä. Edellisen perusteella siis F' on toteutuva. Poistettujen klausuulien joukon C paino $wt(C)$ on korkeintaan $c(M)$.

Olkoon toisaalta C joukko klausuuleja, joiden poistaminen tekee kaavasta toteutuvan. Olkoon M näitä klausuuleja vastaavien kaarten joukko. Klausuulia vastaa kaksi kaarta, joten $c(M) \leq 2 \cdot \text{wt}(C)$. Kaarten M poistamisen jälkeen verkossa mikään pari (p, \bar{p}) ei ole samassa komponentissa, joten C on monileikkaus.

Siis jos C on minimipainoinen poistettava klausuulijoukko ja M minimimonileikkaus, pätee

$$\text{wt}(C) \leq c(M) \leq 2 \cdot \text{wt}(C).$$

Saamme 2CNF \equiv -klausuulien poisto-ongelmalle $O(\log n)$ -approksimointialgoritmin soveltamalla vastaavaa minimimonileikkauksen approksimointialgoritmia. \square

Harvin leikkaus (sparsest cut)

Normaalin yhden solmuparin välisen maksimivuo-ongelman duaali on murtolukuarvoinen minimileikkausongelma. Koska vuo-ongelmalla on aina optimaalinen kokonaislukuarvoinen ratkaisu, itse asiassa maksimivuo ja kokonaislukuarvoisen minimileikkauksen arvot ovat samat.

Monihyödykevuo-ongelman duaali on vastaavasti murtolukuarvoinen minimimonileikkausongelma. Nyt kokonaislukurako on ykkösestä poikkeava, mutta edellä esitetty algoritmi osoittaa erityisesti, että (murtolukuarvoinen) maksimimonihyödykevuo ja (kokonaislukuarvoinen) minimimonileikkaus ovat kertoimen $O(\log k)$ sisällä toisistaan.

Esitämme seuraavaksi monihyödykevuo-ongelmasta kysyntäversion, jonka duaaliksi tulee harvimman leikkauksen ongelma. Tälle versiolle pätee samantyyppinen $O(\log k)$ -suhde kuin edelliselle.

Kysyntäpohjainen monihyödykevuoto

(demands multicommodity flow)

Tässä ongelmassa on annettu normaalin monihyödykevuoto-ongelman tapaan verkko $G = (V, E)$, päätesolmuparit $(s_1, t_1), \dots, (s_k, t_k)$ ja kaarten kapasiteetit c_e . Lisäksi hyödykkeelle i , $i = 1, \dots, k$, on annettu **kysyntä** $\text{dem}(i)$.

Tehtävänä on maksimoida **tuotanto** (throughput) f ehdolla, että hyödykkeen i vuo on ainakin $f \cdot \text{dem}(i)$ kaikilla $1 \leq i \leq k$. Kunkin hyödykkeen vuo toteuttaa normaalit vuon säilymisehdot. Kaaren e yli kulkeva vuo saa olla korkeintaan c_e , kun lasketaan yhteen kaikki hyödykkeet ja kumpikin suunta.

Oletamme, että kahdella eri hyödykkeellä $i \neq j$ pätee $\{s_i, t_i\} \neq \{s_j, t_j\}$. Jos kahdella hyödykkeellä olisi samat päätesolmut, voisimme yhdistää hyödykkeet ja laskea yhteen niiden kysynät.

Käytämme maksimituotannosta merkintää f^* .

Tarkastellaan leikkausta (S, \bar{S}) , missä $S \subseteq V$. Tavalliseen tapaan $c(S)$ on leikkaukseen tulevien kaarten kokonaiskapasiteetti. Määrittelemme tämän leikkauksen leikkaaman kysynnän

$$\text{dem}(S) = \sum_{i \in R} \text{dem}(i),$$

missä $R \subseteq \{1, \dots, k\}$ sisältää ne hyödykkeet i , joiden päätesolmut s_i ja t_i jäävät leikkauksen eri puolille ($s_i \in S$ ja $t_i \in \bar{S}$ tai toisin päin).

Leikkauksen S **harvuus** on nyt

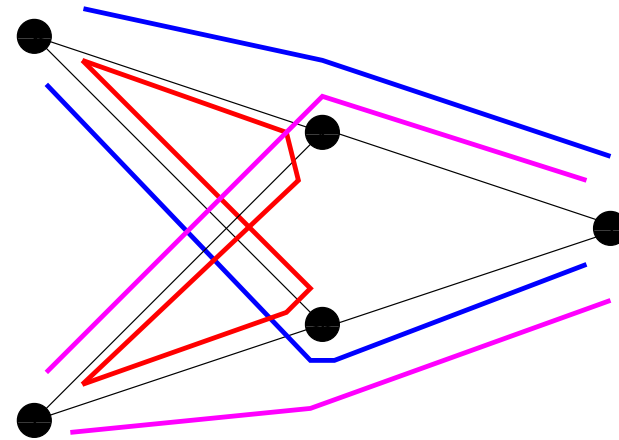
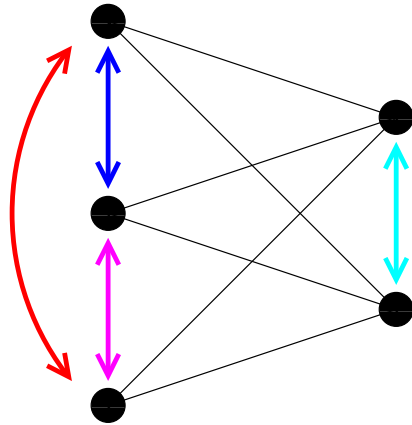
$$\frac{c(S)}{\text{dem}(S)}.$$

Harvimman leikkauksen ongelmassa tehtävänä on löytää S , jolla tämä minimoituu. Käytämme minimiarvosta merkintää α^* .

Selvästi minkä tahansa leikkauksen S harvuus on yläraja tuotannolle. Harvin leikkaus antaa luonnollisesti parhaan tätä muotoa olevista ylärajoista. Voi kuitenkin päteä $f^* < \alpha^*$, kuten seuraavaksi nähdään. Jatkossa esitettävä algoritmi osoittaa, että kuitenkin

$$f^* \geq \frac{\alpha^*}{O(\log k)}.$$

Esimerkki 3.52: Vasemmalla on kaksijakoinen verkko $K_{3,2}$. Kunkin kaaren (ohut musta viiva) kapasiteetti on 1. Kunkin ei-vierekkäisen solmuparin (värilliset nuolet) välinen kysyntä on 1. Harvin leikkaus saavuttaa harvuuden $\alpha^* = 1$.



Oikealla verkko on piirretty toisin. Esitetyssä vuossa jokaista värillistä reittiä kulkee vuo $1/2$. Tämä on ainoa tapa saada kolmelle ensimmäiselle hyödykkeelle vuo 1. Kaikki kaaret on kyllästetty, joten viimeiselle hyödykkeelle ei enää saada vuota. Siis f^* on aidosti pienempi kuin 1. \square

Lineaarisen ohjelman muodostamista varten olkoon $\mathcal{P}_i = \{q_j^i\}$ solmujen s_i ja t_i välisten polkujen joukko. Muuttuja f_j^i esittää polkua q_j^i pitkin kulkevaa vuota ja muuttuja f tuotantoa. Vuo-ongelman lineaarisiksi ohjelmaksi tulee

maksimoi	f	
ehdolla	$\sum_j f_j^i \geq f \cdot \text{dem}(i)$	kaikilla $1 \leq i \leq k$
	$\sum_{i,j:e \in q_j^i} f_j^i \leq c_e$	kaikilla kaarilla $e \in E$
	$f \geq 0$	
	$f_j^i \geq 0$	kaikilla i, j .

Määrittelemme **kysyntäverkon** $H = (V_H, E_H)$, missä $V_H = \{s_1, t_1, \dots, s_k, t_k\}$ ja $E_H = \{(s_i, t_i) \mid i = 1, \dots, k\}$. Kaarelle $e = (s_i, t_i) \in E_H$ määritellään $\text{dem}(e) = \text{dem}(i)$.

Osoitamme jatkossa, että maksimituotanto f^* toteuttaa yhtälön

$$f^* = \min_d \frac{\sum_{e \in E} c_e d_e}{\sum_{e \in E_H} \text{dem}(e) d_e},$$

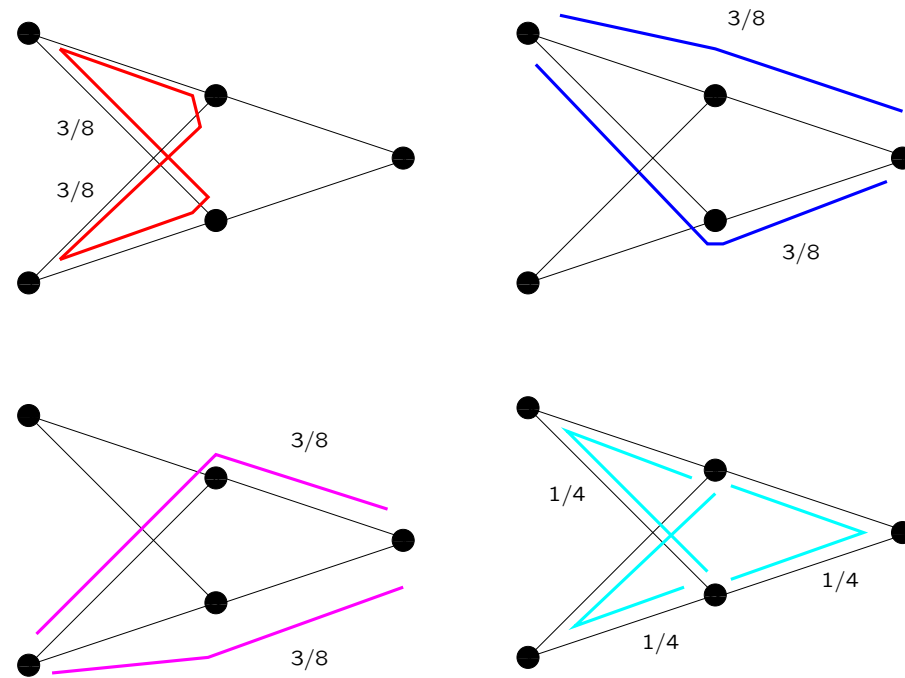
missä minimi on yli **metriikoiden** d . Metriikalla tarkoitetaan tässä kolmioepäyhtälön toteuttavia pituuksia d_e kaarille $e \in E$. Jos $e \notin E$, merkintä d_e tarkoittaa vastaavaa lyhintä polunpituutta.

Lisäksi näemme, että minimoiva d saadaan duaaliongelman ratkaisuna.

Duaaliongelmaan tulee tuttuun tapaan muuttuja d_e kaaren e kapasiteettirajoitusta varten. Muuttujat l_i , $i = 1, \dots, k$, liittyvät hyödykkeen i vuon ja tuotannon f väliseen rajoitteeseen:

minimoi	$\sum_{e \in E} c_e d_e$	
ehdolla	$\sum_{e \in q_j^i} d_e \geq l_i$	kaikilla poluilla $q_j^i \in \mathcal{P}_i$
	$\sum_{i=1}^k l_i \text{dem}(i) \geq 1$	
	$f \geq 0$	
	$d_e \geq 0$	kaikilla $e \in E$
	$l_i \geq 0$	kaikilla $1 \leq i \leq k$.

Esimerkki 3.53: Edellisessä esimerkissä saavutetaan maksimituotto $f^* = 3/4$ ohjaamalla kunkin päätesolmuparin välillä vuot seuraavasti:



Duaalin optimiratkaisussa $d_e = 1/8$ kaikilla e ja $l_i = 1/4$ kaikilla i . \square

Lemma 3.54: Duaalin optimiratkaisu voidaan valita siten, että

- d toteuttaa kolmioepäyhtälön
- $l_i = d_{(s_i, t_i)}$ kaikilla hyödykkeillä i ja
- $\sum_i d_{(s_i, t_i)} \text{dem}(i) = 1$.

Todistus: Jos jollain u, v, w pätee $d_{uw} > d_{uv} + d_{vw}$, ratkaisua voidaan parantaa asettamalla $d_{uw} \leftarrow d_{uv} + d_{vw}$: koska lyhimmät polut eivät muutu, ratkaisun käyppyyскään ei muutu, ja kohdefunktion arvo ei ainakaan kasva. Toistamalla tällaisia muutoksia saadaan kolmioepäyhtälö voimaan ratkaisun optimaalisuutta menettämättä.

Nyt $d_{(s_i, t_i)}$ on lyhimmän polun pituus $s_i \rightsquigarrow t_i$. Voimme asettaa $l_i \leftarrow d_{(s_i, t_i)}$ muuttamatta ratkaisun käyppyyttä tai kohdefunktion arvoa.

Lopulta jos $\sum_i d_{(s_i, t_i)} \text{dem}(i) = \beta > 1$, voimme jakaa kaikki muuttujat vakiolla β pitäen ratkaisun käypänä. Kohdefunktion arvo vain paranee. \square

Yhtälössä

$$f^* = \min_d \frac{\sum_{e \in E} c_e d_e}{\sum_{e \in E_H} \text{dem}(e) d_e},$$

voidaan kertoa kaikki etäisyydet d_e millä tahansa positiivisella vakiolla muuttamatta oikean puolen arvoa. Voimme siis yhtäpitävästi kirjoittaa sen muotoon

$$f^* = \min_d \sum_{e \in E} c_e d_e,$$

missä nyt minimointi on yli metriikoiden \mathbf{d} , jotka lisäksi toteuttavat ehdon $\sum_{e \in E_H} \text{dem}(e) d_e = 1$. Edellisen lemmän mukaan tällainen \mathbf{d} saadaan duaalin optimiratkaisusta. Siis oikealla puolella oleva minimi on duaaliohjelman arvo. Duaalisuuden perusteella se on myös primaalin arvo eli f^* , kuten väitettiin.

Leikkauspakkaukset (cut packing)

Tarkastellaan täydellistä n solmun verkkoa $G = (V, E_n)$ ja sen kaarenpituuksia d , jotka toteuttavat kolmioepäyhtälön.

Tarkastellaan funktiota y , joka liittyy jokaiseen solmujoukkoon S arvon $y(S) \geq 0$. Kaaren $e \in E$ leikkausmäärä on

$$\sum_{S:e \in \delta(S)} y(S),$$

missä taas $\delta(S)$ on joukkojen S ja \bar{S} välisten kaarten joukko.

Funktio y on **leikkauspakkaus**, jos minkään kaaren leikkausmäärä ei ylitä sen pituutta. Jos tämä pätee yhtälönä eli

$$\sum_{S:e \in \delta(S)} y(S) = d_e \quad \text{kaikilla } e \in E,$$

kysymyksessä on tarkka leikkauspakkaus.

Annetulle metriikalle d ei välttämättä ole olemassa tarkkaa leikkauspakkausta. Tarkastelemme jatkossa β -approksimatiivisia leikkauspakkauksia, joilla

$$\frac{d_e}{\beta} \leq \sum_{S:e \in \delta(S)} y(S) \leq d_e \quad \text{kaikilla } e \in E.$$

Tässä siis $\beta \geq 1$, ja $\beta = 1$ vastaa tarkkaa leikkauspakkausta.

Lause 3.55: Olkoon d_e edellä esitetty metriikka, jolla

$$f^* = \min_d \frac{\sum_{e \in E} c_e d_e}{\sum_{e \in E_H} \text{dem}(e) d_e}.$$

Oletetaan, että y on β -approksimatiivinen leikkauspakkaus ja $(S', \overline{S'})$ harvin niistä leikkauksista, joilla $y(S) \neq 0$. Tällöin leikkauksen $(S', \overline{S'})$ harvuus on korkeintaan βf^* .

Todistus: Soveltamalla approksimatiivisen leikkauspakkauksen määritelmän kumpaakin suuntaa saadaan

$$f^* \geq \frac{\sum_{e \in E} c_e \sum_{S: e \in \delta(S)} y(S)}{\sum_{e \in E_H} \text{dem}(e) \sum_{S: e \in \delta(S)} \beta y(S)} = \frac{\sum_S y(S) c(S)}{\beta \sum_S y(S) \text{dem}(S)}.$$

Soveltamalla arviota

$$\frac{\sum_i \alpha_i a_i}{\sum_i \alpha_i b_i} \geq \min_i \frac{a_i}{b_i},$$

joka pätee kaikilla $a_i \geq 0$, $\alpha_i > 0$ ja $b_i > 0$, saadaan

$$f^* \geq \frac{1}{\beta} \cdot \frac{c(S')}{\text{dem}(S')}. \quad \square$$

Korollaari 3.56: Olkoon d kuten edellä ja oletetaan, että y on tarkka leikkauspakkaus. Millä tahansa sellaisella S , että $y(S) \neq 0$, leikkaus (S, \overline{S}) on harvin leikkaus.

Todistus: Pitää siis osoittaa, että leikkauksen S harvuus on f^* , kun $y(S) \neq 0$.

Soveltamalla edellistä lausetta arvolla $\beta = 1$ nähdään heti, että harvin leikkaus, jolla $y(S) \neq 0$, on harvuudeltaan korkeintaan f^* . Minkä tahansa leikkauksen harvuus on myös yläraja tuotolle, joten f^* on tasan tämän leikkauksen harvuus. Siis edellisen lauseen todistuksessa kaikki pätee yhtäsuuruutena. Todistuksessa käytetty arvio

$$\frac{\sum_i \alpha_i a_i}{\sum_i \alpha_i b_i} \geq \min_i \frac{a_i}{b_i}$$

pätee yhtäsuuruutena vain, jos kaikki arvot a_i/b_i ovat samoja. Siis kaikki harvuudet leikkauksille (S, \overline{S}) , joilla $y(S) \neq 0$, ovat samoja ja siis yhtäsuuria kuin f^* . \square

Edellisessä esimerkissä pienin leikkauksen harvuus oli aidosti suurempi kuin f^* . Kyseisessä esimerkissä siis ei ole tarkkaa leikkauspakkausta. Osoitamme, että joka tapauksessa kuitenkin löytyy $O(\log n)$ -approksimatiivinen leikkauspakkaus.

Kun $p \geq 1$, määrittelemme avaruuden \mathbb{R}^m vektoreille ℓ_p -normin

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^m |x_i|^p \right)^{1/p}.$$

Siis tapaus $p = 2$ on euklidinen normi; tässä olemme kiinnostuneita tapauksesta $p = 1$. Joka tapauksessa ℓ_p -normista saadaan edelleen metriikka

$$d_p(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p.$$

Verkon $G = (V, E)$ solmujen kuvaus $\sigma: V \rightarrow \mathbb{R}^m$ on isometrinen ℓ_1 -upotus (isometric ℓ_1 embedding), jos

$$d(u, v) = \|\sigma(u) - \sigma(v)\|_1 \quad \text{kaikilla } u, v \in V.$$

Jos vakiolla $\beta > 1$ kaikilla u, v pätee

$$\frac{d(u, v)}{\beta} \leq \|\sigma(u) - \sigma(v)\|_1 \leq d(u, v),$$

kuvaus σ on β -vääristynyt ℓ_1 -upotus.

Lemma 3.57: Olkoon $\sigma: V \rightarrow \mathbb{R}^m$ kuvaus. On olemassa leikkauspakkaus y , jolla jokaisen kaaren (u, v) leikkausmäärä on $\|\sigma(u) - \sigma(v)\|_1$. Lisäksi $y(S) \neq 0$ korkeintaan $m(n - 1)$ joukolla S .

Todistus: Tarkastellaan ensin tapausta $m = 1$. Numeroidaan solmut v_i siten, että $\sigma(v_1) \leq \sigma(v_2) \leq \dots \leq \sigma(v_n)$. Haluttu ehto pätee, kun asetetaan $y(\{v_1, \dots, v_i\}) = \sigma(v_{i+1}) - \sigma(v_i)$ kaikilla $1 \leq i \leq n - 1$, ja $y(S) = 0$ muilla S .

Olkoon nyt $m > 1$. Koska ℓ_1 -normi laskee suoraan yhteen eri koordinaattien osuudet, voimme määritellä leikkauspakkauksen y_i erikseen joka koordinaatille $i \in \{1, \dots, m\}$ edellä esitettyyn tapaan ja määritellä $y(S) = \sum_{i=1}^m y_i(S)$. \square

Lemma 3.58: Olkoon y leikkauspakkaus, jossa tasan m eri joukolle S pätee $y(S) \neq 0$. On olemassa kuvaus $\sigma: V \rightarrow \mathbb{R}^m$, jolla $\|\sigma(u) - \sigma(v)\|_1$ on kaikilla $(u, v) \in E$ sama kuin kaaren (u, v) leikkausmäärä.

Todistus: Otamme käyttöön oman dimension jokaiselle S , jolla $y(S) \neq 0$. Olkoot S_1, \dots, S_m nämä joukot. Määrittelemme $\sigma_i(u) = 0$, jos $u \in S_i$, ja $\sigma_i(u) = y(S_i)$, jos $u \in \overline{S_i}$. Siis dimensio i lisää kaaren kuvan ℓ_1 -pituutta tasan saman verran, kuin joukko S_i lisää kaaren leikkausmäärää. \square

Edellisistä kahdesta lemmasta saadaan suoraan

Lause 3.59: Metriikalla d on β -vääristynyt ℓ_1 -upotus, jos ja vain jos sillä on β -approksimatiivinen leikkauspakkaus. Lisäksi nolosta poikkeavien leikkausten lukumäärä ja ℓ_1 -upotuksen dimensio ovat polynomisessa suhteessa. \square

Korollari 3.60: Metriikalla d on isometrinen ℓ_1 -upotus, jos ja vain jos sillä on tarkka leikkauspakkaus. \square

Muodostamme ensin upotuksen, joka ei ainakaan kasvata solmujen etäisyyksiä. Tähän riittää yksi ulottuvuus.

Oletetaan siis annetuksi metriikka (V, d) . Valitaan mielivaltainen joukko $S \subseteq V$ ja asetetaan

$$\sigma(v) = \min_{s \in S} d(s, v).$$

Lemma 3.61: Kaikilla u, v pätee

$$|\sigma(u) - \sigma(v)| \leq d(u, v).$$

Todistus: Olkoot s_1 ja s_2 solmuja u ja v lähimmät solmut joukossa S . Voidaan olettaa $d(s_1, u) \leq d(s_2, v)$. Nyt

$$\begin{aligned} |\sigma(u) - \sigma(v)| &= d(s_2, v) - d(s_1, u) \\ &\leq d(s_1, v) - d(s_1, u) \\ &\leq d(u, v), \end{aligned}$$

missä viimeinen askel perustuu kolmioepäyhtälöön. \square

Koska metriikka ℓ_1 on additiivinen, voimme lisätä ulottuvuuksia toisistaan riippumatta. Valitaan siis mielivaltaiset m osajoukkoa S_1, \dots, S_m ja määritellään

$$\sigma_i(v) = \frac{1}{m} \min_{s \in S_i} d(s, v).$$

Toistamalla edellisen lemmän todistus erikseen jokaiselle ulottuvuudelle ja laskemalla yhteen saadaan taas

$$\|\sigma(u) - \sigma(v)\|_1 \leq d(u, v).$$

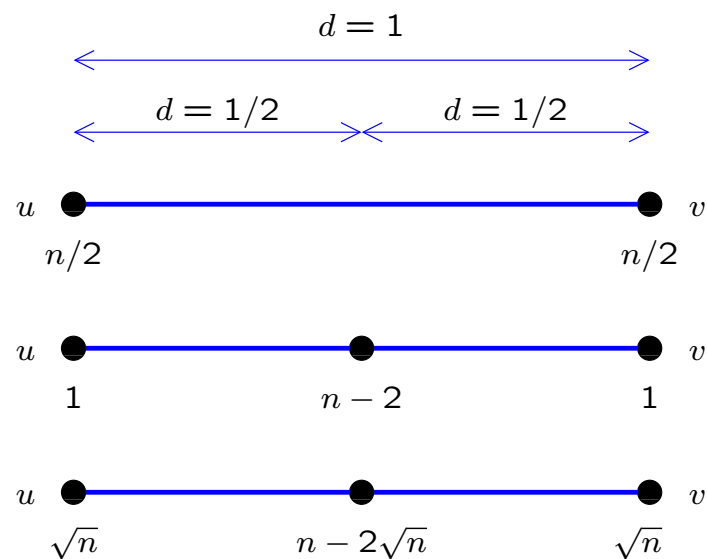
Ongelmana on enää valita sopivat joukot S_i siten, että etäisyydet eivät myöskään pienene liikaa. Esitämme satunnaisen tavan valita tällaiset joukot.

Tarkastelemme etäisyyksien pienenemistä ensin yksittäisen kaaren (u, v) osalta. Kun joukko S_i valitaan satunnaisesti, sen **odotusarvoinen osuus** kaarta (u, v) vastaavaan ℓ_1 -etäisyyteen on

$$\mathbf{E} [|\sigma_i(u) - \sigma_i(v)|].$$

Oletetaan yksinkertaisuuden vuoksi, että $n = 2^l$ jollain $l \in \mathbb{N}$. Määrittelemme satunnaiset joukot S_2, \dots, S_{l+1} valitsemalla joukkoon S_i kunkin solmut v toisista riippumatta todennäköisyydellä $1/2^i$. Perusideana on, että annetulla i joukko S_i voi joillain metriikoilla tuottaa pienen osuuden ja joillain suuremman, mutta kaikilla metriikoilla ainakin jokin S_i tuottaa suuren osuuden.

Esimerkki 3.62: Tarkastelemme seuraavia kolmea metriikkaa, joissa kaikissa $d(u, v) = 1$ ja solmuja on kaikkiaan n .



Kussakin tapauksessa voimme löytää sellaisen i , että joukon S_i osuus on $\Omega(d(u, v)/l)$.

Ensimmäisen metriikan tapauksessa valitsemme $i = l$. Joukkoon S_l tulevien solmujen lukumäärän odotusarvo on $n \cdot (1/2^l) = 1$. Tästä voidaan edelleen osoittaa, että todennäköisyys saada tasan yksi solmu on ainakin vakio (ei mene nollaan, kun n kasvaa; sivuutamme yksityiskohdat). Tapauksessa $|S_l| = 1$ toinen solmuista u ja v on joukosta S_l etäisyydellä nolla ja toinen etäisyydellä 1, joten $|\sigma_l(u) - \sigma_l(v)| = d(u, v)/l$.

Toisen metriikan tapauksessa valitsemme $i = 2$. Vakiotodennäköisyydellä tasan yksi solmuista u ja v tulee joukkoon S_2 , ja tällöin taas $|\sigma_l(u) - \sigma_l(v)| = d(u, v)/(2l)$ (olettaen, että ainakin yksi keskellä olevista $n - 2$ solmusta valittiin).

Kolmannen metriikan tapauksessa valitsemme $i = \lceil l/2 \rceil$. Vakiotodennäköisyydellä joukkoon S_i tulee tasan yksi niistä $2\sqrt{n}$ solmusta, jotka ovat päällekkäin solmun u tai v kanssa. Tällöin $|\sigma_l(u) - \sigma_l(v)| = d(u, v)/(2l)$. \square

Määritellään solmulla $u \in V$ ja säteellä $r \geq 0$ pallo

$$B(u, r) = \{v \in V \mid d(u, v) \leq r\}.$$

Lemma 3.63: Jos joillain $r_1 \geq r_2 \geq 0$ ja $c > 0$ ehto

$$S_i \cap B(u, r_1) = \emptyset \quad \text{ja} \quad S_i \cap B(v, r_2) \neq \emptyset$$

toteutuu ainakin todennäköisyydellä c , niin joukon S_i odotusarvoinen osuus on ainakin $c(r_1 - r_2)/l$.

Todistus: Ehdon toteutuessa $d(u, S_i) > r_1$ ja $d(v, S_i) \leq r_2$. Siis $\sigma_i(u) > r_1/l$ ja $\sigma_i(v) \leq r_2/l$, joten $|\sigma_i(u) - \sigma_i(v)| > (r_1 - r_2)/l$. Jos tämä tapahtuu ainakin todennäköisyydellä c , niin odotusarvoinen osuus on halutun suuruinen. \square

Sopivien r_1 ja r_2 löytämisessä edellisen soveltamiseksi on hyötyä seuraavasta:

Lemma 3.64: Jollain $1 \leq t \leq l - 1$ olkoot A ja B erilliset solmujoukot, joilla $|A| < 2^t$ ja $|B| \geq 2^{t-1}$. Valitaan joukkoon S kukin solmu toisista riippumatta todennäköisyydellä $p = 1/2^{t+1}$. Nyt ainakin todennäköisyydellä $(1/2) \cdot (1 - e^{-1/4})$ pätee

$$S \cap A = \emptyset \quad \text{ja} \quad S \cap B \neq \emptyset.$$

Todistus: Soveltamalla arviota $(1 - x)^n \geq (1 - nx)$ saadaan

$$\Pr[S \cap A = \emptyset] = (1 - p)^{|A|} \geq (1 - p|A|) \geq 1/2.$$

Soveltamalla arviota $1 - x \leq e^{-x}$ saadaan

$$\Pr[S \cap B = \emptyset] = (1 - p)^{|B|} \leq e^{-p|B|} \leq e^{-1/4},$$

joten

$$\Pr[S \cap B \neq \emptyset] \geq 1 - e^{-1/4}.$$

Koska A ja B ovat erilliset, nämä tapahtumat ovat toisistaan riippumattomat, joten haluttu tulos seuraa. \square

Merkitään $c = (1/2) \cdot (1 - e^{-1/4})$. Kun $0 \leq t \leq l$, olkoon ρ_t pienin sellainen säde ρ , että pallot $B(u, \rho)$ ja $B(v, \rho)$ kumpikin sisältävät ainakin 2^t solmua. Erityisesti siis $\rho_0 = 0$ ja $\rho_l \geq d(u, v)$.

Määritellään

$$\hat{t} = \max \{ t \mid \rho_t \leq d(u, v)/2 \}.$$

Lisäksi käytetään u -keskisestä r -säteisestä **avoimesta** pallosta merkintää

$$B^\circ(u, r) = \{ v \in V \mid d(u, v) < r \}.$$

Lemma 3.65: Kun $1 \leq t \leq \hat{t}$, joukon S_{t+1} odotusarvoinen osuus on ainakin $(c/l) \cdot (\rho_t - \rho_{t-1})$. Kun $t = \hat{t} + 1$, joukon S_{t+1} odotusarvoinen osuus on ainakin $(c/l) \cdot (d(u, v)/2 - \rho_{t-1})$.

Todistus: Tarkastellaan ensin tapausta $1 \leq t \leq \hat{t}$. Määritelmän mukaan ainakin toinen avoimista palloista $B^\circ(u, \rho_t)$ ja $B^\circ(v, \rho_t)$ sisältää alle 2^t solmua. Voidaan olettaa, että se on $B^\circ(u, \rho_t)$. Joka tapauksessa $|B(v, \rho_{t-1})| \geq 2^{t-1}$. Koska $\rho_{t-1} \leq \rho_t < d(u, v)/2$, joukot $B^\circ(u, \rho_t)$ ja $B(v, \rho_{t-1})$ ovat erilliset. Edellisen lemmän perusteella ainakin todennäköisyydellä c pätee

$$S_{t+1} \cap A = \emptyset \quad \text{ja} \quad S_{t+1} \cap B \neq \emptyset.$$

Nyt lemma 3.63 antaa halutun rajan joukon S_{t+1} odotusarvoiselle osuudelle.

Tarkastellaan nyt tapausta $t = \hat{t} + 1$. Ainakin toinen avoimista palloista $B^\circ(u, d(u, v)/2)$ ja $B^\circ(v, d(u, v)/2)$ sisältää alle 2^t solmua. Oletetaan taas, että tämä tapahtuu keskipisteellä u . Siis $|B^\circ(u, d(u, v)/2)| < 2^t$. Joka tapauksessa $|B(v, \rho_{t-1})| \geq 2^{t-1}$. Koska $\rho_{t-1} < d(u, v)/2$, joukot $B^\circ(u, d(u, v)/2)$ ja $B(v, \rho_{t-1})$ ovat erilliset. Loppu päätellään kuten edellisessä tapauksessa. \square

Lemma 3.66: Joukkojen S_2, \dots, S_{l+1} odotusarvoinen osuus on yhteensä ainakin $cd(u, v)/(2l)$.

Todistus: Laskemalla yhteen edellisestä lemmasta tulevat rajat ja muistamalla $\rho_0 = 0$ saamme teleskooppisumman

$$\frac{c}{l} \left((\rho_1 - \rho_0) + (\rho_2 - \rho_1) + \dots + (\rho_{\hat{t}} - \rho_{\hat{t}-1}) + \left(\frac{d(u, v)}{2} - \rho_{\hat{t}} \right) \right) = \frac{cd(u, v)}{2l}. \quad \square$$

Lemma 3.67: Olkoon p todennäköisyys, että kaikkien joukkojen yhteenlaskettu osuus on ainakin

$$\frac{cd(u, v)}{4l}.$$

Tällöin $p \geq c/(4 - c)$.

Todistus: Yksittäisen joukon S_i osuudella on selvästi yläraja $d(u, v)/l$. Siis odotusarvoisten osuuksien summa on korkeintaan

$$(1 - p) \cdot \frac{cd(u, v)}{4l} + p \cdot \frac{d(u, v)}{l}.$$

Edellisen lemmän mukaan toisaalta tämä on ainakin

$$\frac{cd(u, v)}{2l}.$$

Ratkaisemalla p saadaan haluttu tulos. \square

Edellä osoitettiin, että mikä tahansa yksittäinen kaari saa suurella todennäköisyydellä riittävästi osuuksia. Tekemällä riippumattomia toistoja ja lisäämällä vastaavasti ulottuvuuksia saadaan haluttu tulos kaikille kaarille yhtäaikaan.

Olkoon $X = \sum_{i=1}^n X_i$, missä satunnaismuuttujat X_i ovat riippumattomia ja saavat arvon 1 todennäköisyydellä p ja 0 todennäköisyydellä $1 - p$. Siis $E[X] = np$, ja kaikilla $0 < \delta \leq 1$ pätee tunnettu Chernoffin raja

$$\Pr[X < (1 - \delta)np] < \exp(-\delta^2 np/2).$$

Valitsemme nyt joukot S_2, \dots, S_{l+1} satunnaisesti edellä esitettyyn tapaan riippumattomasti $N = O(\log n)$ kertaa. Merkitään saatuja joukkoja S_i^j , $i = 2, \dots, l + 1$, $j = 1, \dots, N$. Muodostamme näitä käyttäen metriikalle (V, \mathbf{d}) upotuksen Nl -ulotteiseen avaruuteen edellä esitettyyn tapaan.

Tarkastellaan ensin yksittäistä kaarta (u, v) .

Lemma 3.68: Olkoon $p = c/(4 - c)$. Valitsemalla $N = O(\log n)$ saadaan ehto

$$\|\sigma(u) - \sigma(v)\|_1 \geq \frac{pcd(u, v)}{8l}$$

pätemään ainakin todennäköisyydellä $1 - 1/(2n^2)$.

Todistus: Tarkastelemme kullakin j joukkojen S_i^j valintaa. Tätä satunnaiskoetta sitten toistetaan riippumattomasti N kertaa.

Yksittäinen toisto j **onnistuu**, jos kaikkien joukkojen S_i^j yhteenlaskettu osuus kaareen (u, v) on ainakin $cd(u, v)/(4Nl)$. Edellä esitetyn mukaan onnistumistodennäköisyys on ainakin p .

Chernoffin raja arvolla $\delta = 1/2$ antaa nyt ylärajan $\exp(-Np/8)$ todennäköisyydelle, että onnistumisia tulee alle $Np/2$. Tämä todennäköisyys on korkeintaan $1/(2n^2)$ sopivalla $N = O(\log n)$.

Jos onnistumisia on ainakin $Np/2$, saamme

$$\begin{aligned}
 \|\sigma(u) - \sigma(v)\|_1 &= \sum_{j=1}^N \sum_{i=2}^{l+1} \left| \sigma_i^j(u) - \sigma_i^j(v) \right| \\
 &\geq \frac{Np}{2} \cdot \frac{cd(u, v)}{4Nl} \\
 &= \frac{pcd(u, v)}{8l}. \quad \square
 \end{aligned}$$

Lause 3.69: Edellä esitetty upotus, jossa ulottuvuuksia on siis $Nl = O((\log n)^2)$, on korkeintaan $O(\log n)$ -vääristynyt ainakin todennäköisyydellä $1/2$.

Todistus: Ehto $\|\sigma(u) - \sigma(v)\|_1 \leq d(u, v)$ siis pätee aina. Todennäköisyys, että ehto $d(u, v)/\beta \leq \|\sigma(u) - \sigma(v)\|_1$ ei päde, on edellisen mukaan korkeintaan $1/(2n^2)$ millä tahansa kiinteällä (u, v) . Laskemalla yhteen nämä virhetodennäköisyydet kaikille $n(n-1)/2$ solmuparille saadaan haluttu tulos. \square

Saamme nyt $O(\log n)$ -approksimointialgoritmin harvimmalle leikkaukselle yhdistämällä seuraavat palat:

- Ratkaisun d voidaan olettaa toteuttavan kolmioepäyhtälö (lemma 3.54, s. 287).
- Jos saadulla metriikalla d on β -approksimatiivinen leikkauspakkaus y , harvin leikkaus löytyy tarkastelemalla niitä S , joilla $y(S) > 0$ (lause 3.55, s. 291).
- Jos metriikka d voidaan upottaa m -ulotteiseen ℓ_1 -avaruuteen vääristymällä β , niin tästä upotuksesta saadaan β -approksimatiivinen leikkauspakkaus, jonka nolasta poikkeavien arvojen lukumäärä on polynominen parametrien m ja n suhteen (lemma 3.59, s. 296).
- Tämä onnistuu (suurella todennäköisyydellä) arvoilla $m = O((\log n)^2)$ ja $\beta = O(\log n)$. (lemma 3.69, s. 312)

Voimme parantaa algoritmin approksimointisuhteen arvosta $O(\log n)$ arvoon $O(\log k)$ parilla yksinkertaisella havainnolla.

Lauseessa 3.55 oletettiin, että y on β -approksimatiivinen leikkauspakkaus eli

$$\frac{d_e}{\beta} \leq \sum_{S:e \in \delta(S)} y(S) \leq d_e \quad \text{kaikilla } e \in E.$$

Itse asiassa todistus kuitenkin käytti suuntaa

$$d_e \leq \beta \sum_{S:e \in \delta(S)} y(S) \quad (*)$$

vain niille e , jotka kuuluvat kysyntäverkkoon E_H , ts. kaarille (s_i, t_i) .

Ehto $(*)$ puolestaan seurasi siitä, että upotus σ ei lyhentänyt kaarta e enemmän kuin kertoimella β . Meidän siis riittää muodostaa upotus, joka

1. ei pidennä yhtään kaarta $e \in E$ ja
2. ei lyhennä yhtään kaarta (s_i, t_i) , $1 \leq i \leq k$, enempää kuin kertoimella β .

Esittämässämme upotuksessa σ ehto 1 eli

$$\|\sigma(u) - \sigma(v)\|_1 \leq d_{(u,v)}$$

saatiin aina helposti voimaan kaikilla $u, v \in V$.

Ehto 2 eli

$$\|\sigma(u) - \sigma(v)\|_1 \geq \frac{d_{(u,v)}}{\beta}$$

saavuttamiseksi arvolla $\beta = O(\log n)$ kiinteällä (u, v) valittiin n solmun joukosta osajoukkoja S_i arvoilla $i = 1, \dots, l$, missä $l = O(\log n)$ (lemma 3.68). Tämä piti toistaa $N = O(\log n)$ kertaa kaikkien mahdollisten $u, v \in V$ käsittelemiseksi suurella todennäköisyydellä.

Todistuksesta nähdään helposti, että valitsemalla $O(\log k)$ osajoukkoa $O(k)$ solmun joukosta $\{s_1, t_1, \dots, s_k, t_k\}$ ja toistamalla $O(\log k)$ kertaa saadaan $\beta = O(\log k)$.

Saamme seuraavan algoritmin, jossa yksinkertaisuuden vuoksi on oletettu päätesolmujen joukossa $V' = \{s_1, t_1, \dots, s_k, t_k\}$ olevan tasan 2^l alkia; yleisemmin siis $l = O(\log k)$.

Algoritmi ($O(\log k)$ -approksimaatio harvimmalle leikkaukselle)

1. Muodosta metriikka (V, d) ratkaisemalla lineaarinen ohjelma.
2. Sopivalla $N = O(\log k)$ muodosta Nl joukkoa S_i^j , $i = 1, \dots, l$, $j = 1, \dots, N$, valitsemalla aina kukin joukon V' solmu riippumattomasti todennäköisyydellä $1/2^i$.
3. Muodosta joukkojen S_i^j perusteella upotus σ avaruuteen \mathbb{R}^{Nl} .
4. Muodosta upotuksen σ perusteella leikkauspeite y .
5. Tulosta harvin leikkaus (S, \bar{S}) , jolla $y(S) > 0$.

Lause 3.70: Edellinen algoritmi on $O(\log k)$ -approksimointialgoritmi.

Todistus: Seuraa edellä esitetyistä huomautuksista. \square

Korollari 3.71: Olkoon f^* maksimituotanto ja α^* harvimman leikkauksen harvuus. Tällöin

$$\frac{\alpha^*}{O(\log k)} \leq f^* \leq \alpha^* \quad \square.$$

Minimilaajennus

Määrittelemme leikkaukselle (S, \bar{S}) , missä $|S| \leq n/2$, **kaarilaajentuman** (edge expansion) $|\delta(S)| / |S|$. Esitämme algoritmin approksimatiivisen pienimmän laajentuman löytämiseksi harvimman leikkauksen avulla.

Muodostetaan verkosta G **tasainen monihyödykevuoto**: Päätesolmupareja ovat kaikki $n(n-1)/2$ solmuparia. Kunkin hyödykkeen kysyntä on 1, samoin jokaisen kaaren kapasiteetti. Leikkauksen (S, \bar{S}) harvuus on siis

$$\frac{c(S)}{|S| \cdot |\bar{S}|}.$$

Etsitään $O(\log k)$ -approksimatiivisesti harvin leikkaus (S, \bar{S}) edellisellä algoritmilla. Voidaan olettaa $|S| \leq |\bar{S}|$, jolloin

$$\frac{1}{n} \cdot \frac{c(S)}{|S|} \leq \frac{c(S)}{|S| \cdot |\bar{S}|} \leq \frac{2}{n} \cdot \frac{c(S)}{|S|}.$$

Siis S on myös $O(\log k)$ -approksimatiivisesti pienilaajentumaisin leikkaus. Tässä $k = O(n^2)$, joten $O(\log k) = O(\log n)$. \square

Konduktanssi [Ei kuulu koealueeseen]

Tarkastellaan **kääntyvää** (reversible) Markovin ketjua, jolla siis tasapainojakauma π toteuttaa ehdon

$$\pi(x)P(x, y) = \pi(y)P(y, x) \quad \text{kaikilla tiloilla } x \text{ ja } y.$$

Määritellään suuntaamaton verkko $G = (X, E)$, missä X on Markovin ketjun tilajoukko ja $(x, y) \in E$, jos ja vain jos $\pi(x)P(x, y) > 0$. Määritellään lisäksi kaarelle (x, y) paino $w(x, y) = \pi(x)P(x, y)$. Ketjun **konduktanssi** on

$$\Phi = \min_{0 < \pi(S) \leq 1/2} \frac{w(S, \bar{S})}{\pi(S)},$$

missä minimointi on yli tilajoukkojen $S \subset X$ ja totuttuun tapaan $w(S, \bar{S}) = \sum_{x \in S, y \in \bar{S}} w(x, y)$ ja $\pi(S) = \sum_{x \in S} \pi(x)$. Pieni konduktanssi tarkoittaa, että ketjulla on taipumus jäädä pitkäksi aikaa pyörimään johonkin tilajoukkoon $S \subset X$.

Esitämme konduktanssille $O(\log n)$ -approksimointialgoritmin.

Tarkastelemme konduktanssin symmetrisoitua versiota

$$\Phi' = \min_{0 < \pi(S) \leq 1} \frac{w(S, \bar{S})}{\pi(S)\pi(\bar{S})}.$$

Kun $\pi(S) \leq 1/2$, niin $1/2 \leq \pi(\bar{S}) \leq 1$, joten Φ ja Φ' ovat kertoimen 2 sisällä toisistaan.

Otamme taas jokaiselle solmuparille oman hyödykkeensä, jonka kysyntä on 1. Nyt Φ' on suoraan verkon G leikkauksen S minimiharvuus, jota voidaan approksimoida tarkkuudella $O(\log n)$ edellä esitetyllä algoritmilla. \square

Tasapainoinen leikkaus

Tehtävänä on löytää pienimmän kapasiteetin leikkaus (S, \bar{S}) , kun sen lisäksi on toteutettava ehto

$$bn \leq |S| \leq (1 - b)n$$

annetulla $0 < b \leq 1/2$.

Tarkastelemme tapausta $b = 1/2$ eli **puolitusleikkausta** (bisection cut). Esitämme **pseudoapproksimointialgoritmin**, jonka tuottama leikkaus (S, \bar{S}) on kapasiteetiltaan kertoimen $O(\log n)$ sisällä pienimmästä puolitusleikkauksesta ja joka toteuttaa hieman lievennetyn ehdon $n/3 \leq |S| < 2n/3$.

Algoritmi (puolitusleikkauksen pseudoapproksimointi)

1. Alusta $U \leftarrow \emptyset$ ja $V' \leftarrow V$.
2. Jos $|U| \geq n/3$, aseta $S = U$ ja tulosta (S, \bar{S}) .
3. Muodosta G' rajoittamalla G joukon V' kaariin. Etsi verkosta G' solmujoukko W , joka approksimatiivisesti minimoi kaarilaajentuman (s. 318).
4. Aseta $U \leftarrow U \cup W$ ja $V' \leftarrow V' - W$. Jatka kohdasta 2.

Lause 3.72: Ylläoleva algoritmi tuottaa $(1/3)$ -tasapainoisen leikkauksen, jonka kapasiteetti on kertoimen $O(\log n)$ sisällä puolitusleikkauksen minimistä.

Todistus: Toiseksi viimeisen kierroksen lopulla vielä $|U| < n/3$. Siis ennen viimeistä kierrosta $|V'| \geq 2n/3$. Korkeintaan puolet joukon V' solmuista lisätään joukkoon U viimeisellä kierroksella, joten $|\bar{S}| \geq n/3$ ja $n/3 \leq |S| \leq 2n/3$. Siis leikkaus (S, \bar{S}) on $(1/3)$ -tasapainoinen.

Olkoon (T, \bar{T}) minimipuolitusleikkaus. Koska koko ajan pysyy voimassa $|V'| \geq 2n/3$, sekä $V' \cap T$ että $V' \cap \bar{T}$ sisältävät ainakin $n/6$ solmua. Eräs yläraja pienimmälle kaarilaajentumalle verkossa G' saadaan joukosta $V' \cap T$; tästä edelleen saadaan yläraja $c(T)/(n/6)$.

Algoritmin löytämän joukon W kaarilaajentuma on kertoimen $O(\log n)$ sisällä optimaalisesta puolitusleikkauksesta, joten

$$c(W) \leq O(\log n) \frac{c(T)}{n/6} |W|.$$

Koska $c(A \cup B) \leq c(A) + c(B)$, voidaan laskea yhteen iteraatioiden yli, ja saadaan

$$c(U) \leq O(\log n) \frac{c(T)}{n/6} |U|.$$

Algoritmin loppuessa $S = U$ ja $|S| \leq 2n/3$, joten

$$c(S) \leq O(\log n) \cdot \frac{c(T)}{n/6} \cdot \frac{2n}{3} = O(\log n)c(T). \quad \square$$

Pienileikkauksisin numerointi

Tarkastellaan verkkoa $G = (V, E)$, missä $|V| = n$. Kaarella e on kapasiteetti $c(e)$. Verkon solmujen **numerointi** (linear arrangement) on mikä tahansa bijektio $\sigma: V \rightarrow \{1, \dots, n\}$. Annetulla σ määrittelemme $n - 1$ leikkausta (S_i, \overline{S}_i) , missä $S_i = \{v \in V \mid \sigma(v) \leq i\}$. Tehtävänä on löytää numerointi σ , joka minimoi maksimileikkauksen

$$\max_{1 \leq i \leq n-1} c(S_i) = \max_{1 \leq i \leq n-1} \sum_{u, v: \sigma(u) \leq i < \sigma(v)} c(u, v).$$

Lause 3.73: Algoritmi $P(\{1, \dots, n\}, V)$ (seuraava sivu) muodostaa numeroinnin, jonka maksimileikkaus on kertoimen $O((\log n)^2)$ sisällä minimistä.

Seuraava rekursiivinen algoritmi $P(\{a, a + 1, \dots, b\}, U)$ jakaa numerot $a, a + 1, \dots, b$ solmujoukolle U , missä $|U| = b - a + 1$:

1. Jos $a = b$ ja $U = \{v\}$, niin aseta $\sigma(v) = a$.
2. Muuten muodosta verkko G_U rajaamalla verkko G joukon U solmuihin. Muodosta verkossa G_U pseudoapproksimatiivinen puolitusleikkaus (S, \bar{S}) sivun 322 algoritmillä. Siis $|U|/3 \leq |S| < 2|U|/3$.
3. Numeroi solmujoukot S ja \bar{S} rekursiivisilla kutsuilla $P(\{a, \dots, a + |S| - 1\}, S)$ ja $P(\{a + |S|, \dots, b\}, \bar{S})$.

Algoritmin analysoimiseksi muodostetaan sen rekursiivisia kutsuja kuvaava binääripuu $T = T_V$.

Jokaista algoritmin suorittamaa kutsua $P(\{a, \dots, b\}, U)$ kohti puussa T on alipuu T_U , jonka lehtiin on talletettu joukon U solmut, yksi solmu lehteä kohti:

- Jos $|U| = 1$, puu T_U muodostuu yhdestä lehdestä.
- Jos $|U| > 1$ ja kutsussa $P(\{a, \dots, b\}, U)$ muodostetaan leikkaus (S, \bar{S}) , puun T_U vasemmaksi alipuuksi tulee T_S ja oikeaksi $T_{\bar{S}}$. Lisäksi puun juureen talletetaan leikkauksen (S, \bar{S}) kaaret.

Jokainen solmu tulee tasan yhteen lehteen ja jokainen kaari tasan yhteen sisäsolmuun:

- Algoritmin palauttama numerointi vastaa puun solmujen järjestystä siten, että $\sigma(u) < \sigma(v)$, jos ja vain jos solmun u lehti puussa T on solmun v lehdestä vasemmalle.
- Kaari (u, v) talletetaan sisäsolmuun, joka on solmut u ja v sisältävien lehtien alin yhteinen esi-isä.

Havainto: Olkoon σ pienileikkauksisin järjestys ja β vastaava kohdefunktion arvo. Millä tahansa verkon G osaverkolla G' on puolitusleikkaus, jonka kapasiteetti on korkeintaan β .

Olkoon nimittäin verkon G' solmujoukko V' , missä $|V'| = m$. Valitaan joukkoon S joukosta V' numeroinnin σ mukaan $m/2$ ensimmäistä solmua ja joukkoon \bar{S} loput. Määritelmän mukaan leikkauksen (S, \bar{S}) kapasiteetti on korkeintaan β . \square

Koska rekursiivisissa kutsuissa solmujoukko pienenee ainakin kertoimella $2/3$, puun syvyys on $O(\log n)$. Erityisesti siis algoritmin suoritus-aika on polynominen.

Olkoon α puun sisäsolmu, joka vastaa solmujoukolla U suoritettua kutsua. Siis solmuun α talletettujen kaarten kapasiteetti on korkeintaan $O(\log n)$ kertaa verkon G_U pienimmän puolitusleikkauksen kapasiteetti. Edellä esitetyn havainnon nojalla tämä puolestaan on korkeintaan pienileikkauksisimman numeroinnin optimiarvo β .

Tarkastellaan nyt leikkausta (S_i, \bar{S}_i) . Olkoon α niiden kahden lehden alin yhteinen esi-isä, joiden numerot ovat i ja $i + 1$. Nyt kaikki leikkaukseen (S_i, \bar{S}_i) kuuluvat kaaret on talletettu solmuun α tai sen esi-isiin. Koska esi-isiä on $O(\log n)$, näiden kaarten yhteiskapasiteetti on korkeintaan $O(\log n) \cdot O(\log n) \cdot \beta = O((\log n)^2)\beta$. Tämä oli haluttu approksimaatiotulos.

Tuotantolaitosten sijoittelu

Tarkastelemme tuotantolaitosten sijoitteluongelman **metristä** versiota **ilman kapasiteetteja**.

Annettuna on joukko F **tuotantolaitoksia** (facilities) ja joukko C **kaupunkeja** (cities).

Tuotantolaitoksen $i \in F$ **avaamiskustannus** on f_i .

Kuljetuskustannus tuotantolaitoksesta $i \in F$ kaupunkiin $j \in C$ on c_{ij} . Kuljetuskustannusten oletetaan noudattavan kolmioepäyhtälöä.

Kelvollinen ratkaisu koostuu avointen laitosten joukosta $I \subseteq F$ ja kuvauksesta $\phi: C \rightarrow I$, joka liittää jokaiseen kaupunkiin avoimen tuotantolaitoksen. Saman avoimen tuotantolaitoksen saa liittää useaan kaupunkiin.

Tavoitteena on minimoida ratkaisun (I, ϕ) kustannus

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j), j}.$$

Esitämme ongelmalle 3-aproksimointialgoritmin.

Ratkaisua (I, ϕ) vastaa arvoasetus binäärisille muuttujille x_{ij} , missä $i \in F$ ja $j \in C$, ja y_i , missä $i \in F$: asetetaan $x_{ij} = 1$, jos $i = \phi(j)$, ja $y_i = 1$, jos $i \in I$. Samastamme jatkossa binääriarvoiset vektorit (x, y) ja sijoitusongelman ratkaisut (I, ϕ) .

Muuttujien (x, y) avulla ongelma voidaan muotoilla kokonaislukuohjelmana

minimoi
$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i$$

ehdolla
$$\sum_{i \in F} x_{ij} \geq 1$$

kaikilla $j \in C$

$$y_i - x_{ij} \geq 0$$

kaikilla $i \in F$ ja $j \in C$

$$x_{ij} \in \{0, 1\}$$

kaikilla $i \in F$ ja $j \in C$

$$y_i \in \{0, 1\}$$

kaikilla $i \in F$.

Löysennöksessä taas kokonaislukuehdot korvautuvat ehdoilla $x_{ij} \geq 0$ ja $y_i \geq 0$. Duaaliksi saadaan

maksimoi	$\sum_{j \in C} \alpha_j$	
ehdolla	$\alpha_j - \beta_{ij} \leq c_{ij}$	kaikilla $i \in F$ ja $j \in C$
	$\sum_{j \in C} \beta_{ij} \leq f_i$	kaikilla $i \in F$
	$\alpha_j \geq 0$	kaikilla $j \in C$
	$\beta_{ij} \geq 0$	kaikilla $i \in F$ ja $j \in C$.

Duaalin intuitiivisen merkityksen ymmärtämiseksi oletetaan, että (löysennetyllä) primaalilla sattuu olemaan kokonaislukuarvoinen optimiratkaisu (\mathbf{x}, \mathbf{y}) (jota siis vastaa sijoittelu (I, ϕ)). Olkoon $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ puolestaan duaalin optimi.

Primaalisten komplementaarisuusehtojen nojalla

1. jos $x_{ij} > 0$ niin $\alpha_j - \beta_{ij} = c_{ij}$
2. jos $y_i > 0$ niin $\sum_{j \in C} \beta_{ij} = f_i$

ja duaalisten komplementaarisuusehtojen nojalla

3. jos $\alpha_j > 0$ niin $\sum_{i \in F} x_{ij} = 1$
4. jos $\beta_{ij} > 0$ niin $y_i = x_{ij}$.

Perustulkinta on, että kaupungilla j on rahasumma α_j , josta se maksaa valitsemalleen tuotantolaitokselle $i = \phi(j)$ kuljetuskustannuksen c_{ij} sekä osuuden β_{ij} avaamiskustannuksesta.

Ehdon 2 mukaan jos laitos i on avoin, sen saamat osuudet β_{ij} maksavat koko kustannuksen f_i .

Ehdon 4 mukaan kaupunki j maksaa nolasta poikkeavan osuuden β_{ij} laitokselle i vain, jos tosiaan $i = \phi(j)$.

Ehdon 3 mukaan jokainen kaupunki, joka ylipäänsä on otettu mukaan ratkaisuun, valitsee tasan yhden tuotantolaitoksen.

Ratkaisemme ongelman primaali-duaaliskeemalla. Palautetaan mieleen approksimatiiviset komplementaarisuusehdot (s. 173).

Oletetaan, että primaaliratkaisu (\mathbf{x}, \mathbf{y}) ja duaaliratkaisu $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ toteuttavat duaaliset komplementaarisuusehdot tarkasti ja primaaliset approksimatiivisesti siten, että

1. jos $x_{ij} > 0$ niin $\frac{1}{3}c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij}$

2. jos $y_i > 0$ niin $\frac{1}{3}f_i \leq \sum_{j \in C} \beta_{ij} \leq f_i$.

Tällöin kyseiset ratkaisut ovat kertoimen 3 sisällä optimaalisesta.

Jatkossa muodostettava ratkaisu (\mathbf{x}, \mathbf{y}) on kokonaislukuarvoinen, joten siitä saatava sijoittelu (I, ϕ) antaa ylläolevien ehtojen vallitessa 3-approksimaation.

Itse asiassa todistamme hieman voimakkaamman ehdon. Algoritmin kuluessa kaupungit jakautuvat **suoraan** ja **epäsuorasti yhdistettyihin**. Vain suoraan yhdistetty kaupunki osallistuu avausmaksuun; ts. jos kaupunki j on epäsuorasti yhdistetty, niin $\beta_{ij} = 0$ kaikilla j .

Epäsuoraan yhdistetyn kaupungin j ehto approksimoituu siis muotoon

$$\frac{1}{3}c_{ij} \leq \alpha_j \leq c_{ij} \quad \text{kun } x_{ij} > 0.$$

(Ja muistetaan, että $x_{ij} > 0$ tarkoittaa samaa kuin $i = \phi(j)$.)

Muita ehtoja ei approksimoida, vaan jokaisella suoraan yhdistetyllä kaupungilla j pätee

$$\alpha_j - \beta_{ij} = c_{ij} \quad \text{kun } i = \phi(j),$$

ja avoimella laitoksella $i \in I$ pätee

$$\sum_{j:\phi(j)=i} \beta_{ij} = f_i.$$

Algoritmissa on kaksi vaihetta.

- Vaihe 1 löytää duaalille käyvän ratkaisun ja primaalia varten joukon $F_t \subseteq F$ tilapäisesti avattuja laitoksia. Kaaret (i, j) , jolla $\beta_{ij} > 0$, julistetaan erityisiksi.
- Vaihe 2 valitsee tilapäisesti avattujen laitosten joukosta lopullisesti auki jäävän joukon $I \subseteq F_t$. Jokaiselle kaupungille j valitaan avoin laitos $\phi(j)$ niiden laitosten i joukosta, joilla kaari (i, j) on erityinen.

Vaiheessa 1 muuttujia α_i ja β_{ij} kasvatetaan nolosta lähtien pitäen duaalin käyppyysehdot koko ajan voimassa. Ajattelemme, että arvot kasvavat jatkuvasti ajan kuluessa. Algoritmin analyysissä on merkitystä sillä, missä järjestyksessä eri rajoitteet tulevat tiukoiksi.

Vaihe 1

Aluksi kaikki kaupungit merkitään yhdistämättömiksi.

Kaikilla yhdistämättömillä kaupungeilla j muuttujaa α_j kasvatetaan yhdellä yksiköllä per aikayksikkö. Siis ajanhetkellä t muuttujalla α_j on arvo t , ellei kaupunki j ole jo tullut yhdistetyksi.

Kun jollain (i, j) muuttuja α_j saavuttaa arvon c_{ij} , muuttujaa β_{ij} ruvetaan kasvattamaan tasatahtiin muuttujan α_j kanssa, jotta ehto $\alpha_j - \beta_{ij} \leq c_{ij}$ pysyy voimassa. Kaaria, joilla tämä pätee yhtälönä, sanotaan tiukoiksi. Kun muuttujan β_{ij} saa aidosti positiivisen arvon, kaari (i, j) julistetaan erityiseksi.

Jos jollain laitoksella i sen saama kokonaismaksu $\sum_{j \in C} \beta_{ij}$ saavuttaa arvon f_i , laitos i merkitään tilapäisesti avoimeksi. Kaikki laitokset j , joilla kaari (i, j) on tiukka, julistetaan yhdistetyiksi ja laitos i niiden yhdistystodisteeksi.

Yhdistettyjen kaupunkien j muuttujia α_j ei enää kasvateta.

Jatkossa jos i on jo tilapäisesti avoin ja (i, j) tulee tiukaksi, taas julistetaan j yhdistetyksi ja i sen yhdistystodisteeksi. Kaupungilla j ei voi ennestään olla yhdistystodistetta, koska tällöin α_j olisi lakannut kasvamasta. Huomaa, että tässä tapauksessa β_{ij} jää arvoon 0 eikä kaaresta (i, j) tule erityistä.

Jos useita asioita tapahtuu yhtäikaa, vastaavat toimet valitaan suoritettavaksi mielivaltaisessa järjestyksessä.

Vaihe 1 päättyy, kun kaikki kaupungit on yhdistetty.

Vaihe 2

Vaiheessa 1 kaupunki j saattoi päätyä antamaan maksun $\beta_{ij} > 0$ usealle laitokselle i . Nyt pitää valita, mikä näistä on $\phi(j)$.

Olkoon F_t vaiheessa 1 tilapäisesti avattujen laitosten joukko. Muodostetaan verkko H , jonka solmujoukko on F_t . Laitosten i ja i' välillä on kaari verkossa H , jos jollain $j \in C$ kaaret (i, j) ja (i', j) ovat kumpikin erityisiä.

Algoritmi valitsee nyt lopulliseksi avointen laitosten joukoksi I minkä tahansa maksimaalisen riippumattoman joukon verkossa H .

Kaupungille j olkoon

$$\mathcal{F}_j = \{i \in F_t \mid (i, j) \text{ on erityinen.}\}$$

Koska I on riippumaton joukko, korkeintaan yksi joukon \mathcal{F}_j laitoksista on avoin.

Jos joukossa \mathcal{F}_j on tasan yksi laitos i , asetetaan $\phi(j) = i$ ja julistetaan kaupunki j **suoraan yhdistetyksi**.

Muuten olkoon i' kaupungin j yhdistystodiste. Jos $i' \in I$, asetetaan taas $\phi(j) = i$ ja julistetaan kaupunki j suoraan yhdistetyksi.

Jos $i' \notin I$, niin jollain solmun i' naapurilla i verkossa H pätee $i \in I$, sillä I valittiin maksimaaliseksi riippumattomaksi joukoksi. Asetetaan $\phi(j) = i$ jollain tällaisella i ja julistetaan j **epäsuoraan yhdistetyksi**.

Vaihe 2 päättyy tähän.

Algoritmin tuottamasta sijoittelusta (I, ϕ) saadaan primaalin käypä ratkaisu (\mathbf{x}, \mathbf{y}) . Tarkastelemme nyt approksimatiivisten komplementaarisuusehtojen voimassaoloa.

Kaupungin j rahamääräksi tulkittu α_j jaetaan kahteen osaan: α_j^f maksaa laitoksen $\phi(j)$ avaamisesta ja α_j^e kaaren $(\phi(j), j)$ kuljetuskustannuksesta.

Joka tapauksessa $\alpha_j = \alpha_j^f + \alpha_j^e$. Jos j on epäsuorasti yhdistetty, se ei maksa laitoksen avaamisesta. Asetamme

$$\alpha_j^f = 0 \quad \text{ja} \quad \alpha_j^e = \alpha_j$$

Jos j on suoraan yhdistetty ja $i = \phi(j)$, kaari (i, j) on tiukka eli $\alpha_j = c_{ij} + \beta_{ij}$. Asetamme

$$\alpha_j^f = \beta_{ij} \quad \text{ja} \quad \alpha_j^e = c_{ij}.$$

Avoimet laitokset on täysin maksettu näillä osuuksilla:

Lemma 3.74: Jos $i \in I$, niin

$$\sum_{j:\phi(j)=i} \alpha_j^f = f_i.$$

Todistus: Kun laitos i on tilapäisesti auki vaiheen 1 lopussa, se on täysin maksettu. Siis

$$\sum_{j:(i,j) \text{ erityinen}} \beta_{ij} = f_i.$$

Edellä todettiin, että joukossa \mathcal{F}_j voi olla korkeintaan yksi avoin laitos $i \in I$. Siis kaikki kaupungit j , joiden β_{ij} osallistuu avoimen laitoksen i maksuun, tulevat suoraan yhdistetyiksi laitokseen i .

Siis näillä j pätee $\alpha_j^f = \beta_{ij}$. Muilla kaupungeilla j pätee $\alpha_j^f = 0$. \square

Korollaari 3.75:

$$\sum_{j \in C} \alpha_j^f = \sum_{i \in I} f_i. \quad \square$$

Koska $\alpha_j^f = 0$ epäsuorasti yhdistetyillä kaupungeilla j , edellä riittää summata yli suoraan yhdistettyjen kaupunkien.

Lemma 3.76: Jos kaupunki j on epäsuorasti yhdistetty ja $i = \phi(j)$, niin $c_{ij} \leq 3\alpha_j^e$.

Todistus: Epäsuorasti yhdistetylle j pätee $\alpha_j^e = \alpha_j$. Olkoon i' kaupungin j yhdistystodiste. Koska j on epäsuorasti yhdistetty laitokseen i , verkossa H on kaari (i', i) . Siis jollain j' sekä (i, j') että (i', j') ovat erityisiä.

Osoitamme, että

$$\begin{aligned} \alpha_j &\geq c_{i'j} \\ \alpha_j &\geq c_{ij'} \\ \alpha_j &\geq c_{i'j'}. \end{aligned}$$

Laskemalla yhteen ja soveltamalla kolmioepäyhtälöä saamme $c_{ij} \leq 3\alpha_j^e$.

Ensimmäinen epäyhtälö $\alpha_j \geq c_{i'j}$ seuraa suoraan siitä, että kaari (i', j) on tiukka.

Kaaret (i', j') ja (i, j') ovat kumpikin tiukkoja, joten $\alpha_j \geq c_{i'j'}$ ja $\alpha_j \geq c_{ij'}$. Koska ne ovat lisäksi erityisiä, ne ovat kumpikin tulleet tiukiksi ennen kuin kumpikaan solmuista i ja i' on tilapäisesti avattu.

Olkoot t_1 ja t_2 ajat, joina i ja i' tilapäisesti avattiin, ja $t = \min\{t_1, t_2\}$. Muuttujan $\alpha_{j'}$ kasvaminen loppuu viimeistään ajanhetkellä t , joten $\alpha_{j'} \leq t$. Toisaalta i' on kaupungin j yhdistystodiste, joten $\alpha_j \geq t_2$. Näistä seuraa halutut epäyhtälöt. \square

Lause 3.77: Algoritmin tuottamille primaali- ja duaaliratkaisulle pätee

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + 3 \sum_{i \in F} f_i y_i \leq 3 \sum_{j \in C} \alpha_j.$$

Huomautus: tämä on siis hieman tiukempi kuin 3-aproksimaation takaava

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \leq 3 \sum_{j \in C} \alpha_j.$$

Todistus: Suoraan yhdistetylle kaupungille j ja laitokselle $i = \phi(j)$ pätee $c_{ij} = \alpha_j^e$. Yhdistämällä edelliseen lemmaan nähdään, että kaikilla kaupungeilla j pätee $c_{ij} \leq 3\alpha_j^e$. Laskemalla yhteen yli kaupunkien saadaan

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq 3 \sum_{j \in C} \alpha_j^e.$$

Yhdistämällä tämä edelliseen korollaariin saadaan väite. \square

Kuten primaali-duaaliskeemaan perustuvat algoritmit yleensäkin, algoritmi on laskennallisesti tehokas. Hahmottelemme ajassa $O(m \log m)$ toimivan toteutuksen, missä $m = |F| \cdot |C|$ on kaarten lukumäärä.

Aluksi laitamme kaaret listaan painon mukaan kasvavaan järjestykseen. Tämä antaa meille samalla järjestyksen, jossa kaaret voivat tulla tiukoiksi.

Saamme myös jokaiselle laitokselle i arvioidun avaamisajan t_i , joka kertoo, milloin laitos tulisi täysin maksetuksi, jos siitä maksavien kaupunkien joukko ei muuttuisi. Aluksi kaikilla laitoksilla maksavien laitosten joukko on tyhjä ja t_i on ääretön. Pidämme laitokset minimikeossa ajan t_i mukaan.

Otamme aina keosta tai listasta seuraavan tapahtuma-ajan ja päivitämme rakenteita seuraavaksi esitettävään tapaan.

Kaari (i, j) tulee tiukaksi:

- Jos i ei vielä ole tilapäisesti auki, se saa yhden maksajan lisää, ja sen arvioitu aukeaminen nopeutuu.
- Jos i on jo tilapäisesti auki, kaupunki j julistetaan yhdistetyksi. Siis jatkossa α_j ei enää kasva. Laitoksilla i' , joiden maksamiseen j osallistuu, maksajien joukkoa pitää pienentää ja arvioitua avaamisaikaa pitää vastaavasti lykätä.

Laitos i tulee täysin maksetuksi (ja tilapäisesti avatuksi):

- Kaikki laitosta i maksavat kaupungit j julistetaan yhdistetyiksi. Kuten edellä, tämä pitää ottaa huomioon niissä muissa laitoksissa, joita ne ovat maksamassa.

Kaareen (i, j) voi liittyä kaksi tapahtumaa:

- kaari tulee tiukaksi
- laitos i avataan tilapäisesti.

Kummassakin tapahtumassa päivitysten aikavaativuutta dominoi keon päivityksen $O(\log |F|)$. Siis vaihe 1 kaikkiaan vie ajan $O(m \log m)$.

Tiukka esimerkki

Ongelmassa on kaupungit a_1, \dots, a_n ja laitokset f_1 ja f_2 .

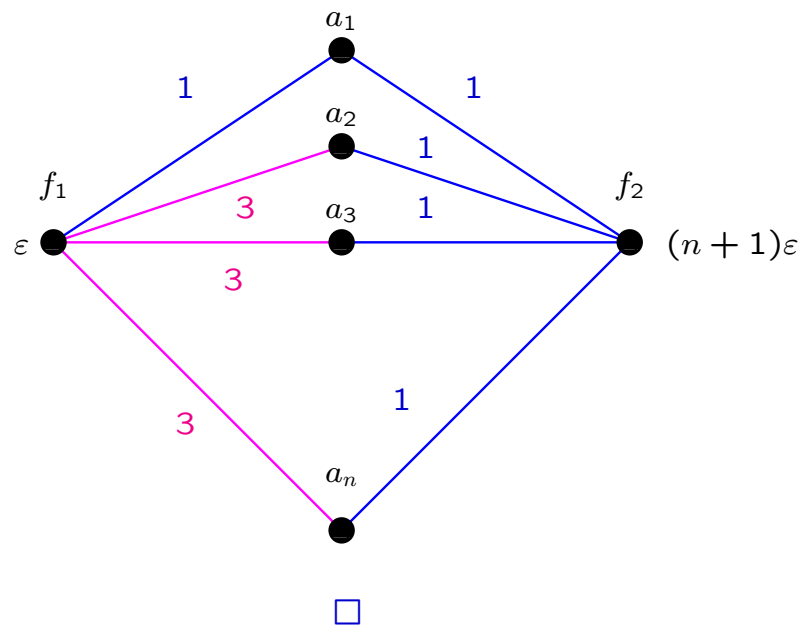
Etäisyys laitoksesta f_2 on 1 jokaiseen kaupunkiin.

Etäisyys laitoksesta f_1 kaupunkiin a_1 on 1 ja muihin kaupunkeihin 3 (suurin joka toteuttaa kolmioepäyhtälön).

Avaamiskustannus on ε laitokselle f_1 ja $(n + 1)\varepsilon$ laitokselle f_2 , missä ε on pieni positiivinen luku.

Optimiratkaisu on avata laitos f_2 ja yhdistää kaikki kaupungit siihen, kustannuksena $(n + 1)\varepsilon + n$.

Algoritmin vaiheessa 1 kumpikin laitos tulee alustavasti avatuksi. Laitos f_1 on yhdistystodiste kaupungille a_1 ja f_2 muille kaupungeille. Jos riippumattomaksi joukoksi I valitaan $\{f_1\}$, kustannukseksi tulee $\varepsilon + 1 + 3(n - 1)$.



Metrinen k -mediaaniongelma

Tämä muistuttaa tuotantolaitosten sijoittelua. Nyt kuitenkin yhden tuotantolaitoksen avaamisella ei ole kustannusta, vaan avattavien tuotantolaitosten kokonaismäärälle on asetettu kiinteä yläraja. Ongelma liittyy läheisesti myös ryvästämiseen (clustering).

On siis annettu joukko F tuotantolaitoksia ja C kaupunkeja sekä kustannus c_{ij} laitoksen $i \in F$ yhdistämiselle kaupunkiin $j \in C$. Kustannukset toteuttavat kolmioepäyhtälön. Lisäksi on annettu avattavien tuotantolaitosten lukumäärän yläraja $k \in \mathbb{N}_+$.

Tehtävänä on löytää avattavien tuotantolaitosten joukko $I \subseteq F$, jolla $|I| \leq k$, ja kaupungit avoimiin tuotantolaitoksiin yhdistävä kuvaus $\phi: C \rightarrow I$ siten, että kustannus $\sum_{j \in C} c_{\phi(j),j}$ minimoituu.

Ongelma voidaan esittää kokonaislukuohjelmana, missä muuttujien x_{ij} ja y_i tulkinta on sama kuin tuotantolaitosten sijoittelussa:

$$\begin{array}{ll}
 \text{minimoi} & \sum_{i \in F, j \in C} c_{ij} x_{ij} \\
 \text{ehdolla} & \sum_{i \in F} x_{ij} \geq 1 \quad \text{kaikilla } j \in C \\
 & y_i - x_{ij} \geq 0 \quad \text{kaikilla } i \in F \text{ ja } j \in C \\
 & \sum_{i \in F} (-y_i) \geq -k \\
 & x_{ij} \in \{0, 1\} \quad \text{kaikilla } i \in F \text{ ja } j \in C \\
 & y_i \in \{0, 1\} \quad \text{kaikilla } i \in F.
 \end{array}$$

Löysennöksessä taas kokonaislukurajoitteet korvautuvat rajoitteilla $x_{ij} \geq 0$ ja $y_i \geq 0$.

Duaaliksi tulee

maksimoi

$$\sum_{j \in C} \alpha_j - zk$$

ehdolla

$$\alpha_j - \beta_{ij} \leq c_{ij}$$

kaikilla $i \in F$ ja $j \in C$

$$\sum_{j \in C} \beta_{ij} \leq z$$

kaikilla $i \in F$

$$\alpha_j \geq 0$$

kaikilla $j \in C$

$$\beta_{ij} \geq 0$$

kaikilla $i \in F$ ja $j \in C$

$$z \geq 0.$$

Käytämme hyväksi tuotantolaitoksen sijoitteluongelmaa.

Valitaan jokin arvo $z \geq 0$. Metrinen k -keskusongelman tapauksesta voidaan muodostaa tuotantolaitosten sijoitteluongelman tapaus antamalla jokaisen tuotantolaitoksen perustamiselle sama kustannus z .

Olkoot (\mathbf{x}, \mathbf{y}) ja $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ optimaalinen primaali- ja duaaliratkaisu näin saadulle lineaariselle ohjelmalle (s. 331–332) Vahvan dualisuuden perusteella

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} z y_i = \sum_{j \in C} \alpha_j.$$

Oletetaan nyt, että tämä ratkaisu sattuu avaamaan (murtolukumääräisesti) tasan k tuotantolaitosta, ts. $\sum_{i \in F} y_i = k$. Väitämme, että (\mathbf{x}, \mathbf{y}) ja $(\boldsymbol{\alpha}, \boldsymbol{\beta}, z)$ ovat optimaalinen primaali- ja duaaliratkaisu k -keskusongelman lineaariselle löysennökselle.

Ratkaisun käyppyyys seuraa suoraan tuotantolaitosongelman käyppyysehdoista ja oletuksesta $\sum_{i \in F} y_i = k$.

Optimaalisuus seuraa edellisen sivun yhtälöstä

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} z y_i = \sum_{j \in C} \alpha_j.$$

sijoittamalla $\sum_{i \in F} y_i = k$ ja järjestämällä termit uudelleen:

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} = \sum_{j \in C} \alpha_j - z k.$$

Siis primaalin ja duaalin kohdefunktiolla on sama arvo, joten ratkaisupari on optimaalinen.

Edellä tarkasteltiin murtolukuarvoisia ratkaisuja. Oletetaan nyt, että tuotantolaitosten sijoittelun 3-approksimaatioalgoritmi (s. 337–341) tuottaa ratkaisut (x, y) ja (α, β) , joissa taas tasan k tuotantolaitosta on avattu. Lauseen 3.77 nojalla

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} + 3zk \leq 3 \sum_{j \in C} \alpha_j.$$

Nyt (x, y) ja (α, β, z) ovat k -keskusongelman käypä kokonaislukuarvoinen primaaliratkaisu ja murtolukuarvoinen duaaliratkaisu. Koska ne toteuttavat ehdon

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq 3 \left(\sum_{j \in C} \alpha_j - zk \right),$$

ratkaisun kustannus poikkeaa optimista korkeintaan kertoimella 3.

Jos siis osaisimme löytää arvon z , joka tuottaa ominaisuuden $\sum_{i \in F} y_i = k$, saisimme edellisen nojalla 3-aproksimaatioalgoritmin. Menetelmää tällaisen arvon z löytämiseksi ei kuitenkaan tunneta. Esitämme hieman toisenlaisen menettelyn, joka johtaa approksimaatiosuhteeseen 6.

Selvästi mitä suurempi z , sen vähemmän laitoksia avataan. Erityisesti jos $z = 0$, kaikki laitokset avataan. Jos taas $z \geq nc_{\max}$, missä c_{\max} on suurin kustannus c_{ij} ja $n = |F| + |C|$, niin vain yksi laitos avataan. Binäärihaulla välin $[0, nc_{\max}]$ yli voidaan löytää arvot z_1 ja z_2 , joilla

- arvolla $z = z_1$ algoritmi avaa $k_1 < k$ laitosta (kokonaislukuarvoisesti)
- arvolla $z = z_2$ algoritmi avaa $k_2 > k$ laitosta (kokonaislukuarvoisesti)
- $z_1 - z_2 \leq c_{\min}/(12|C|^2)$.

(Jos sattuu löytymään ratkaisu, jossa avataan tasan k laitosta, voidaan tämä proseduuri keskeyttää ja siirtyä edellä esitettyyn.)

Olkoot nyt $(\mathbf{x}^s, \mathbf{y}^s)$ ja $(\mathbf{x}^l, \mathbf{y}^l)$ arvoja z_1 ja z_2 vastaavat primaalin kokonaislukuratkaisut. Siis

$$\sum_{i \in F} y_i^s = k_1 < k < k_2 = \sum_{i \in F} y_i^l.$$

Olkoot (α^s, β^s) ja (α^l, β^l) vastaavat algoritmin tuottamat duaaliratkaisut.

Valitaan nyt kertoimet $0 \leq a, b \leq 1$, joilla

$$\begin{aligned} a + b &= 1 \\ ak_1 + bk_2 &= k. \end{aligned}$$

(Siis $a = (k_2 - k)/(k_2 - k_1)$ ja $b = 1 - a$.) Muodostetaan ratkaisujen $(\mathbf{x}^s, \mathbf{y}^s)$ ja $(\mathbf{x}^l, \mathbf{y}^l)$ konvekssi kombinaatio

$$(\mathbf{x}, \mathbf{y}) = a(\mathbf{x}^s, \mathbf{y}^s) + b(\mathbf{x}^l, \mathbf{y}^l).$$

Nyt (\mathbf{x}, \mathbf{y}) on primaalin käypä murtolukuratkaisu ja $\sum_{i \in F} y_i = ak_1 + bk_2 = k$. Lisäksi ratkaisussa \mathbf{x} kukin kaupunki on yhdistetty korkeintaan kahteen laitokseen.

Lemma 3.78: Ratkaisun (x, y) kustannus on kertoimen $(3 + 1/|C|)$ sisällä optimaalisen murtolukuratkaisun kustannuksesta.

Todistus: Lauseen 3.77 nojalla

$$\sum_{i \in F, j \in C} c_{ij} x_{ij}^s \leq 3 \left(\sum_{j \in C} \alpha_j^s - z_1 k_1 \right) \quad (*)$$

$$\sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq 3 \left(\sum_{j \in C} \alpha_j^l - z_2 k_2 \right). \quad (**)$$

Osoitamme jatkossa, että

$$\sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq \left(3 + \frac{1}{|C|} \right) \cdot \left(\sum_{j \in C} \alpha_j^l - z_1 k_2 \right). \quad (***)$$

Kertomalla (*) vakiolla a ja (***) vakiolla b ja laskemalla yhteen saadaan

$$\sum_{i \in F, j \in C} c_{ij} x_{ij} \leq \left(3 + \frac{1}{|C|}\right) \cdot \left(\sum_{j \in C} \alpha_j - z_1 k\right),$$

missä $\alpha = a\alpha^s + b\alpha^l$.

Merkitään samoin $\beta = a\beta^s + b\beta^l$. Koska $z_1 > z_2$, ratkaisu (α^l, β^l) on käypä duaaliratkaisu myös arvolle $z = z_1$. Siis myös konvekssi kombinaatio (α, β) on käypä duaaliratkaisu arvolle $z = z_1$. Tästä seuraa, että (α, β, z_1) on käypä duaaliratkaisu k -mediaaniongelmaan.

Siis ratkaisun (x, y) kustannus on kertoimen $(3 + 1/|C|)$ sisällä duaalin optimista, joten väite seuraa.

Pitää vielä osoittaa (***) eli

$$\sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq \left(3 + \frac{1}{|C|}\right) \cdot \left(\sum_{j \in C} \alpha_j^l - z_1 k_2\right).$$

Koska $z_1 - z_2 \leq c_{\min}/12 |C|^2$, epäyhtälöstä (**) seuraa

$$\sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq 3 \left(\sum_{j \in C} \alpha_j^l - z_1 k_2 + \frac{c_{\min} k_2}{12 |C|^2}\right).$$

Kirjoitetaan (***) muotoon

$$\frac{3}{3 + 1/|C|} \sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq 3 \left(\sum_{j \in C} \alpha_j^l - z_1 k_2\right)$$

eli

$$\left(1 - \frac{1/|C|}{3 + 1/|C|}\right) \sum_{i \in F, j \in C} c_{ij} x_{ij}^l \leq 3 \left(\sum_{j \in C} \alpha_j^l - z_1 k_2\right).$$

Riittää siis osoittaa, että

$$\frac{1/|C|}{3 + 1/|C|} \sum_{i \in F, j \in C} c_{ij} x_{ij}^l \geq \frac{c_{\min} k_2}{4|C|^2}$$

eli

$$\frac{|C|}{3 + 1/|C|} \sum_{i \in F, j \in C} c_{ij} x_{ij}^l \geq \frac{c_{\min} k_2}{4}.$$

Tämä seuraa, koska $\sum_{i \in F, j \in C} c_{ij} x_{ij}^l \geq c_{\min}$ ja $|C|/(3 + 1/|C|) \geq |C|/(3 + 1) \geq k_2/4$. \square

Oletetaan nyt, että olemme saaneet kaksi laitostensijoitteluongelman kokonaislukuratkaisua, (x_1, y_1) ja (x_2, y_2) .

Ensimmäisessä avointen laitosten joukko on A , ja $|A| = k_1 < k$.

Toisessa avointen laitosten joukko on B , ja $|B| = k_2 > k$.

Kuten edellä, olkoot a ja $b = 1 - a$ sellaiset, että $ak_1 + bk_2 = k$, ja $(x, y) = a(x_1, y_1) + b(x_2, y_2)$. Edellisen perusteella voimme tiettyjen ehtojen vallitessa olettaa, että ratkaisun (x, y) kustannus on kertoimen $3 + 1/|C|$ sisällä k -mediaaniongelman optimikustannuksesta.

Muodostamme murtolukuratkaisusta (x, y) satunnaisesti pyöristämällä kokonaislukuratkaisun, jossa on tasan k avattua laitosta. Kustannus kasvaa odotusarvoisesti korkeintaan kertoimella $1 + \max\{a, b\}$.

Myöhemmin osoitamme, miten satunnaisuus voidaan poistaa.

Muodostetaan joukko $B' \subset B$ valitsemalla siihen jokaiselle joukon A laitokselle sitä lähin joukon B laitos. Jos näin valittuja laitoksia on alle k_1 (ts. usealla joukon A laitoksella on sama lähin joukon B laitos), joukkoon B' otetaan lisäksi mielivaltaisia muita joukon B alkioita niin, että lopulta $|B'| = k_1$.

Avaamme ensinnäkin k_1 laitosta seuraavasti:

- todennäköisyydellä a kaikki joukon A laitokset
- muuten (todennäköisyydellä $b = 1 - a$) kaikki joukon B' laitokset.

Lisäksi avaamme

- satunnaisesti valitut $k - k_1$ laitosta joukosta $B - B'$.

Yksittäisen joukon $B - B'$ laitoksen todennäköisyys tulla avatuksi on $(k - k_1)/(k_2 - k_1) = b$. Joka tapauksessa avattavien laitosten joukossa I on tasan k laitosta.

Kaupungit laitoksiin yhdistävä kuvaus $\phi: C \rightarrow F$ muodostetaan seuraavasti:

Olkoot $i_1 \in A$ ja $i_2 \in B$ laitokset, joihin kaupunki j on yhdistetty ratkaisuihin x_1 ja x_2 .

Jos $i_2 \in B'$, niin joko i_1 tai i_2 on joukossa I . Asetetaan vastaavasti $\phi(j) = i_1$ tai $\phi(j) = i_2$.

Olkoon nyt $i_2 \in B - B'$.

- Jos $i_2 \in I$, valitaan $\phi(j) = i_2$.
- Muuten jos $i_1 \in I$, valitaan $\phi(j) = i_1$.
- Muuten valitaan $\phi(j) = i_3$, missä $i_3 \in B'$ on laitosta i_1 lähinnä oleva joukon B laitos.

Huomaa, että jos $i_1 \notin I$, niin $B' \subset I$, joten todella $i_3 \in I$.

Olkoon

$$\text{cost}(j) = ac_{i_1,j} + bc_{i_2,j}$$

kaupungin j yhdistämiskustannus ratkaisussa (\mathbf{x}, \mathbf{y}) .

Lemma 3.79: Kaupungin j yhdistämiskustannuksen odotusarvo edellä kuvatus pyörityksen jälkeen on

$$\mathbf{E}[c_{\phi(j),j}] \leq (1 + \max\{a, b\})\text{cost}(j).$$

Huom. todistuksesta näemme myös, miten odotusarvon $\mathbf{E}[c_{\phi(j),j}]$ tarkka arvo voidaan laskea. Tarvitsemme tätä satunnaisuuden poistamiseen ehdollisen odotusarvon menetelmällä.

Todistus: Jos $i_2 \in B'$, niin $\phi(j) = i_1$ todennäköisyydellä a ja $\phi(j) = i_2$ todennäköisyydellä b , joten $\mathbf{E}[c_{\phi(j),j}] = \text{cost}(j)$. Oletetaan siis $i_2 \in B - B'$.

Kun $i_2 \in B - B'$, niin

- todennäköisyydellä b laitos i_2 tulee avatuksi ja $\phi(j) = i_2$
- todennäköisyydellä $(1 - b)a = a^2$ laitos i_1 tulee avatuksi ja $\phi(j) = i_1$
- todennäköisyydellä $(1 - b)(1 - a) = ab$ laitos i_3 tulee avatuksi ja $\phi(j) = i_3$.

Siis

$$\mathbf{E}[c_{\phi(j),j}] = bc_{i_2,j} + a^2c_{i_1,j} + abc_{i_3,j}.$$

Koska i_3 on laitosta i_1 lähin joukon B alkio,

$$c_{i_1, i_3} \leq c_{i_1, i_2} \leq c_{i_1, j} + c_{i_2, j},$$

missä on käytetty kolmioepäyhtälöä. Edelleen kolmioepäyhtälöstä seuraa

$$c_{i_3, j} \leq c_{i_1, j} + c_{i_1, i_3} \leq 2c_{i_1, j} + c_{i_2, j}.$$

Siis

$$\begin{aligned} \mathbf{E}[c_{\phi(j), j}] &= bc_{i_2, j} + a^2c_{i_1, j} + abc_{i_3, j} \\ &\leq bc_{i_2, j} + a^2c_{i_1, j} + ab(2c_{i_1, j} + c_{i_2, j}) \\ &= (ac_{i_1, j} + bc_{i_2, j}) + ab(c_{i_1, j} + c_{i_2, j}) \\ &\leq (ac_{i_1, j} + bc_{i_2, j})(1 + \max\{a, b\}), \end{aligned}$$

missä toinen yhtäsuuruus seuraa yhtälöstä $a^2 + ab = a$. \square

Korollaari 3.80: Edellä esitetyn pyöristysmenetelmän tuottamalle kokonaislukuratkaisulle $(\mathbf{x}^k, \mathbf{y}^k)$ pätee

$$\mathbf{E} \left[\sum_{i \in F, j \in C} c_{ij} x_{ij}^k \right] \leq (1 + \max \{ a, b \}) \left(\sum_{i \in F, j \in C} c_{ij} x_{ij} \right).$$

Poistetaan nyt satunnaisuus ehdollisen odotusarvon menetelmällä.

Ensimmäinen satunnaisvalinta on joko joukon A tai joukon B' avaaminen. Kokeillaan kumpaakin valintaa ja lasketaan, mikä on odotusarvoinen kustannus, kun lisäksi avataan satunnaiset $k - k_1$ laitosta joukosta $B - B'$. Tehdään pienemmän odotusarvon antava valinta.

Toinen satunnaisvalinta on $k - k_1$ laitoksen valitseminen joukosta $B - B'$.

Jos on jo valittu $D \subset B - B'$, missä $|D| \leq k - k_1$, olkoon

$$\mathbf{E}[D, B - (B' \cup D)]$$

odotusarvoinen kustannus, jos avataan kaikki joukon D laitokset ja lisäksi satunnaiset $k - k_1 - |D|$ joukon $B - (B' \cup D)$ laitosta. Koska jokainen joukon $B - (B' \cup D)$ laitos on yhtä todennäköinen,

$$\mathbf{E}[D, B - (B' \cup D)] = \frac{1}{|B - (B' \cup D)|} \sum_{i \in B - (B' \cup D)} \mathbf{E}[D \cup \{i\}, B - (B' \cup D \cup \{i\})].$$

Siis jollain $i \in B - (B' \cup D)$ pätee

$$\mathbf{E}[D \cup \{i\}, B - (B' \cup D \cup \{i\})] \leq \mathbf{E}[D, B - (B' \cup D)].$$

Liitetään tällainen i joukkoon D . Toistetaan, kunnes $|D| = k - k_1$.

Odotusarvot voidaan laskea lemmän 3.79 todistuksesta ilmenevällä tavalla.

Muistetaan, että $n = |F| + |C|$ oli verkon solmujen ja $m = |F| \cdot |C|$ kaarten lukumäärä. Yhteenvetona saamme nyt seuraavan:

Lause 3.81: Edellä esitetty algoritmi tuottaa k -mediaaniongelmalle 6-approksimaation ajassa $O((m \log m)(L + \log n))$, missä $L = \log_2(c_{\max}/c_{\min})$.

Todistus: Aluksi tarvittiin siis arvot z_1 ja z_2 , joilla arvo $z = z_1$ laitostensijoitteluongelmassa avaa $k_1 < k$ laitosta, arvo $z = z_2$ laitostensijoitteluongelmassa avaa $k_2 > k$ laitosta ja $z_1 - z_2 \leq c_{\min}/(12n^2)$.

Lähtemällä alkuarvoista $z_2 = 0$ ja $z_1 = nc_{\max}$ väli saadaan kavennetuksi binäärihaulla, joka vaatii $O(\log(n^3 c_{\max}/c_{\min})) = O(L + \log n)$ iteraatiota.

Yhden iteraation aikavaativuus on oleellisesti laitostensijoittelualgoritmin aikavaativuus $O(m \log m)$.

Lemman 3.78 mukaisesti (s. 360) arvoja $z = z_1$ ja $z = z_2$ vastaavien laitostensijoitteluratkaisujen konvekssi yhdistelmä (x, y) on kustannukseltaan kertoimen $3 + 1/|C|$ sisällä ongelman optimiarvosta. Korollaan 3.80 mukaan pyöristäminen kasvattaa kustannusta odotusarvoisesti korkeintaan kertoimella $1 + \max\{a, b\}$. Satunnaisuuden poistamisen jälkeenkin pyöristäminen sujuu ajassa $O(m)$, eli binäärihaun aikavaativuus dominoi.

Todetaan vielä, että

- $a = (k_2 - k)/(k_2 - k_1) \leq 1 - 1/|C|$ (pahin tapaus $k_1 = k - 1$ ja $k_2 = |C|$)
- $b = (k - k_1)/(k_2 - k_1) \leq 1 - 1/k$ (pahin tapaus $k_1 = 1$ ja $k_2 = k + 1$).

Siis pyöristämisestä tuleva lisäkerroin on siis korkeintaan $1 + \max\{a, b\} \leq 2 - 1/|C|$. Yhteensä approksimaatiosuhteen ylärajaksi tulee

$$\left(3 + \frac{1}{|C|}\right) \cdot \left(2 - \frac{1}{|C|}\right) < 6. \quad \square$$

Algoritmin approksimaatiosuhteelle 6 ei tunneta tiukkaa esimerkkiä. Seuraava osoittaa, että pyöristysvaiheessa tosiaan kustannus voi kasvaa kertoimella $1 + \max\{a, b\}$.

Esimerkki 3.82: Laitokset ovat $F = \{f_0, \dots, f_{k+1}\}$ ja kaupungit $C = \{c_1, \dots, c_n\}$. Etäisyys on laitoksesta f_0 kuhunkin kaupunkiin 1 ja laitoksesta f_{k+1} kuhunkin kaupunkiin ε . Etäisyys laitoksesta f_0 muihin laitoksiin on 1, ja etäisyys laitoksesta f_i laitokseen f_{i+1} on 2 kaikilla $1 \leq i \leq k$. Muut etäisyydet määräytyvät näistä lyhimpinä polunpituuksina.

Lähtökohdiksi otetaan ratkaisut (x^s, y^s) , joka avaa tuotantolaitoksen f_0 (siis $k_1 = 1$), ja (x^l, y^l) , joka avaa tuotantolaitokset f_1, \dots, f_{k+1} (siis $k_2 = k + 1$). Arvoilla $a = 1/k$ ja $b = 1 - 1/k$ pätee nyt $ak_1 + bk_2 = k$.

Konveksin yhdistelmän $a(\mathbf{x}^s, \mathbf{y}^s) + b(\mathbf{x}^l, \mathbf{y}^l)$ kustannus on $an + b\epsilon n$.

Pyörästysvaiheessa oletetaan, että laitoksen f_0 lähimmäksi naapuriksi on valittu f_1 . Siis

- todennäköisyydellä $(k-1)/k = b$ laitos f_{k+1} avataan ja yhdistetään kaikkiin kaupunkeihin;
kustannus kaupunkia kohti tässä tapauksessa ϵ
- todennäköisyydellä $(1-b)a = a^2$ avataan laitos f_0 mutta ei laitosta f_{k+1} ;
kustannus kaupunkia kohti tässä tapauksessa 1
- todennäköisyydellä $(1-b)(1-a) = ab$ avataan laitos f_1 mutta ei laitosta f_0 eikä f_{k+1} ;
kustannus kaupunkia kohti tässä tapauksessa 2

Odotusarvoinen kustannus on siis $n(b\epsilon + a^2 + 2ab)$. Rajalla $\epsilon \rightarrow 0$ nähdään, että pyörästys lisää kustannusta kertoimella

$$\frac{n(a^2 + 2ab)}{an} = a + 2b = 1 + b. \quad \square$$

Algoritmi osoittaa kokonaislukuraolle ylärajan 6. Seuraava esimerkki osoittaa alarajaksi oleellisesti 2.

Esimerkki 3.83: Verkon solmujoukko on $V = \{a_0, \dots, a_n\}$. Etäisyys solmuista a_1, \dots, a_n solmuun a_0 on 1 ja muihin solmuihin 2. Tuotantolaitosten joukko on $F = V$ ja kaupunkien joukko $C = \{a_1, \dots, a_n\}$. Tuotantolaitoksia avataan $k = n - 1$ kappaletta.

Eräs optimaalinen kokonaislukuratkaisu on avata tuotantolaitokset a_1, \dots, a_{n-1} , kustannuksena 2.

Muodostamme murtolukuratkaisun avaamalla laitoksen a_0 osuudella $1/(n - 1)$ ja loput laitokset kukin osuudella $(n - 2)/(n - 1)$. Kustannus on $n/(n - 1)$.

Siis kokonaislukurako on $2(n - 1)/n$. \square

4. PCP-teoreema ja approksimoitumattomuus

PCP-teoreema (probabilistic checking of proofs) (Arora ja Safra 1992; Arora, Lund, Motwani, Sudan, Szegezy 1992) on merkittävä vaihtoehtoinen karakterisointi luokalle NP. Tätä tulosta ja raon tuottavia palautuksia käytten voidaan tulostaa esim. että

- jos $P \neq NP$, niin on olemassa vakio $\varepsilon > 0$, jolla lukumääräistä klikkiongelmaa on mahdotonta approksimoida suhteella $1/n^\varepsilon$
- jos lukumääräiselle joukkopeiteongelmalla on $((1 - \varepsilon) \ln n)$ -approksimaatioalgoritmi jollain $\varepsilon > 0$, niin $NP \subseteq DTIME(n^{O(\log \log n)})$.

Käymme tässä vain hyvin pintapuolisesti läpi perusesimerkkejä palautustekniikasta.

Satunnaiset todistuksen tarkastajat

Tunnetusti kieli L on luokassa NP, jos ja vain jos on olemassa polynomi p ja sellainen polynomisessa ajassa toimiva Turingin kone A , että

- jos $x \in L$, niin on olemassa merkkijono y , jolla $|y| \leq p(|x|)$ ja kone A hyväksyy syötteen $\langle x, y \rangle$
- jos $x \notin L$, niin kaikilla sellaisilla merkkijonoilla y , joilla $|y| \leq p(|x|)$, kone A hylkää syötteen $\langle x, y \rangle$.

Jos $A(x, y)$ hyväksyy, niin y on todistus (proof) sille, että $x \in L$. Algoritmi A on tarkastaja (verifier) tällaisille todistuksille.

Esittelemme nyt mallin, jossa tarkastaja tarkastaa vain satunnaisesti valitun pienen osan sille tarjotusta todistusehdokkaasta, mutta silti suurella todennäköisyydellä saa kiinni tapaukset $x \notin L$.

Oletetaan annetuksi kaksi kokonaislukuarvoista funktiota, satunnaisbittien määrä $r(\cdot)$ ja tarkastettavien bittien määrä $q(\cdot)$.

Näillä parametreilla varustettu satunnainen todistuksen tarkastaja on algoritmi, joka saa syötteenä

- varsinaisen syötemerkkijonon x
- todistuksen y , jolla $|y| \leq p(|x|)$ jollain polynomilla p
- jonon satunnaisbittejä, joita on $r(|x|)$.

Tarkastaja toimii polynomisessa ajassa ja

lukee todistuksesta y vain $q|x|$ bittiä.

Oletamme tässä, että tarkastaja pystyy jollain suorasaantimekanismilla vakioajassa lukemaan minkä tahansa osoittamansa yksittäisen bitin todistuksesta y . Sen valitsemat bitit voivat siis riippua syötteestä x ja satunnaisbiteistä.

Satunnainen todistuksen tarkastaja tunnistaa kielen L , jos

- jos $x \in L$, niin on olemassa sellainen todistus y , että tarkastaja hyväksyy todennäköisyydellä 1
- jos $x \notin L$, niin millä tahansa todistuksella y tarkastaja hyväksyy todennäköisyydellä, joka on alle $1/2$.

Jos kieli L voidaan tunnistaa satunnaisella todistuksen tarkastajalla, joka käyttää $O(r(\cdot))$ satunnaisbittiä ja lukee vain $O(q(\cdot))$ bittiä todistuksesta, kirjoitamme

$$L \in \text{PCP}(r(n), q(n)).$$

Määritelmästä seuraa suoraan, että $\text{NP} = \text{PCP}(0, \text{poly}(n))$.
Mielenkiintoisemmin

Lause 4.1 (PCP-teoreema): $\text{NP} = \text{PCP}(\log n, 1)$.

Suunta $\text{PCP}(\log n, 1) \subseteq \text{NP}$ on helppo, suunta $\text{NP} \subseteq \text{PCP}(\log n, 1)$ erittäin vaikea todistaa.

Intuition saamiseksi tarkastellaan monien NP-täydellisyystodistusten lähtökohtaa 3SAT-ongelmaa ja sen todistuksia y , jotka ovat yksinkertaisesti syötekaavan φ toteuttavia muuttujien totuusarvoasetuksia.

Oletetaan, että syötekaavassa φ on m klausuulia; siis $|\phi| = O(m)$. Käyttämällä $O(\log |\varphi|)$ satunnaisbittiä voidaan valita satunnainen klausuuli c .

Tarkastamalla nyt todistuksesta y niiden kolmen muuttujan arvo, jotka esiintyvät klausuulissa c , nähdään, toteutuiko klausuuli c . Jos φ ei ole toteutuva, niin millä tahansa y on ainakin yksi klausuuli, jonka valitseminen johtaa kaavan epätotuuden havaitsemiseen.

Ongelma on, että jos ei-toteutuvia klausuuleja on vain yksi, niin todennäköisyys osua siihen on $1/m$, ei $1/2$ niin kuin määritelmä edellyttäisi. Sen osoittaminen, että $3SAT \in PCP(\log n, 1)$, edellyttää jotain yllättävää konstruktiota.

PCP-teoreemasta saadaan suoraan ensimmäinen, ehkä ei kovin luonnollinen, approksimoitumattomuustulos.

Hyväksymistodennäköisyyden maksimointi Tarkastellaan jotain kiinteää PCP($\log n, 1$)-tarkastajaa V ongelmalle SAT.

Tapaus: propositiokaava φ

Käypä ratkaisu: totuusarvoasetus y kaavan φ muuttujille

Kohdefunktio: todennäköisyys, että V hyväksyy kaavan φ todisteen y (maksimoitava).

Lause 4.2: Jos $P \neq NP$, niin hyväksymistodennäköisyyden maksimoinnille ei ole $(1/2)$ -approksimointialgoritmia.

Todistus: Jos φ on toteutuva, niin valitsemalla toteuttava y saadaan hyväksymistodennäköisyyden maksimi 1. Jos φ ei ole toteutuva, niin määritelmän mukaan hyväksymistodennäköisyys millä tahansa y on alle $1/2$.

Oletetaan, että algoritmi A antaa approksimaatiosuhteen $1/2$. Olkoon y algoritmin A tuottama todiste kaavalle φ . Siis

1. jos φ on toteutuva, hyväksymistodennäköisyys todistuksella y on siis ainakin $1/2$
2. jos φ ei ole toteutuva, hyväksymistodennäköisyys on alle $1/2$ millä tahansa todistuksella ja siis erityisesti todistuksella y .

Jos tarkastaja V käyttää $c \log n$ satunnaisbittiä, mahdollisia satunnaisbittijonoja on $2^{c \log n} = n^c$ eli polynominen määrä. Voimme siis polynomisessa ajassa laskea hyväksymistodennäköisyyden simuloimalla V :n toimintaa kaikilla mahdollisilla satunnaisjonoilla ja laskemalla hyväksymiset ja hylkäämiset. Tämä kertoo polynomisessa ajassa, ollaanko tapauksessa (1) vai (2). \square

MAX-3SAT-ongelman approksimoinnin vaikeus

Tarkastelemme siis seuraavaa ongelmaa:

Tapaus: 3-CNF-kaava φ (konjunktio korkeintaan kolmen literaalin disjunktioista)

Käypä ratkaisu: totuusarvoasetus y kaavan φ muuttujille

Kohdefunktio: niiden klausuulien (eli disjunktoiden) lukumäärä kaavassa φ , jotka toteutuvat totuusarvoasetuksella y (**maksimoitava**).

Lause 4.3: Jos $P \neq NP$, niin on olemassa sellainen vakio $\varepsilon_M > 0$, että MAX-3SAT-ongelmalla ei ole $(1 - \varepsilon_M)$ -approksimointialgoritmia.

Korollari 4.4: Jos $P \neq NP$, niin MAX-3SAT-ongelmalla ei ole polynomista approksimointiskeemaa. \square

Todistus: Riittää osoittaa, että on olemassa polynomisessa ajassa laskettava kuvaus päätösongelman SAT tapauksista φ optimointiongelman MAX-3SAT tapaukseksi ψ (jotka kumpikin ovat propositiokaavoja), että kun m on kaavan φ klausuulien lukumäärä, niin

- jos φ on toteutuva, niin $\text{OPT}(\psi) = m$
- jos φ ei ole toteutuva, niin $\text{OPT}(\psi) < (1 - \varepsilon_M)m$.

Jos nyt MAX-3SAT-ongelmalla olisi $(1 - \varepsilon_M)$ -approksimointialgoritmi, saisimme siitä helposti polynomisessa ajassa toimivan ratkaisualgoritmin SAT-ongelmalle.

Sanomme tällaisia palautuksia päätösongelmasta optimointiongelmaan [raon tuottaviksi palautuksiksi](#).

Käytämme välitavoitteena seuraavaa ongelmaa **MAX k -FUNCTION SAT**:

Tapaus: n propositiomuuttujaa x_1, \dots, x_n ja m funktiota f_1, \dots, f_m , joista kukin riippuu vain k muuttujasta x_i

Käypä ratkaisu: totuusarvoasetukset y_i muuttujille x_i

Kohdefunktio: niiden funktioiden f_j lukumäärä, jotka toteutuvat sijoituksella $x = y$ (**maksimoitava**).

Lemma 4.5: Jollain vakiolla k on olemassa polynomisessa ajassa laskettava kuvaus SAT-tapauksilta φ MAX k -FUNCTION SAT -tapauksille I siten, että

- jos φ on toteutuva, niin $\text{OPT}(I) = m$
- jos φ ei ole toteutuva, niin $\text{OPT}(I) < m/2$.

Todistus: Olkoon SAT-ongelmalla PCP($\log n, 1$)-tarkastaja V , joka käyttää $c \log n$ satunnaisbittiä ja lukee q bittiä todistuksesta. Tarkastellaan SAT-tapausta φ , jolla $|\varphi| = n$. Kaikilla satunnaisvalinnoilla yhteensä V siis voi lukea korkeintaan qn^c eri bittiä todistuksesta. Otetaan käyttöön oma totuusarvomuuttuja jokaiselle näistä biteistä. Käytetään näiden muuttujien joukosta merkintää B .

Valitsemme $k = q$ ja $m = n^c$. Muodostamme jokaiselle $c \log n$ bitin jonolle r funktion f_r , joka riippuu q :sta joukon B muuttujasta.

Funktio f_r argumenteilla z_1, \dots, z_q palauttaa tosi, jos V hyväksyy, kun syöte on φ , satunnaisbittijono r ja todistuksesta luettavat bitit z_i . Nämä funktiot jollain järkevällä tavalla koodattuina on selvästi mahdollista muodostaa polynomisessa ajassa syötteestä φ .

Jos φ on toteutuva, jokin todistus saa tarkastajan V hyväksymään todennäköisyydellä 1. Vastaavat joukon B muuttujien arvot toteuttavat siis kaikki m funktiota f_r .

Jos φ ei ole toteutuva, millä tahansa todistuksella V hyväksyy alle todennäköisyydellä $1/2$, ja siis millä tahansa joukon B muuttujien arvoilla alle puolen funktioista f_r toteutuu. Tämä päättää lemmän todistuksen. \square

Olkoon nyt φ SAT-tapaus. Muodostamme siitä ensin edellä esitettyyn tapaan MAX k -FUNCTION SAT -tapauksen I , jossa on $m = n^c$ funktiota f_r . Seuraavaksi muodostamme funktiojoukosta I MAX-3SAT-tapauksen ψ .

Esitetään ensin kukin funktio f_r konjunkttiivisessa normaalimuodossa. Siis funktiota f_r tulee esittämään kaava ψ_r , joka on konjunktio korkeintaan 2^q disjunktioista, kussakin disjunktiossa korkeintaan q muuttujaa.

Muodostetaan näiden CNF-kaavojen konjunktio $\psi = \bigwedge_r \psi_r$.

- Jos φ on toteutuva, jokin muuttujien arvoasetus toteuttaa kaikki funktiot f_r ja siis kaavan ψ kaikki klausuulit.
- Jos φ ei ole toteutuva, millä tahansa arvoasetuksella yli puolet funktioista f_r ja siis ainakin $n^c/2$ kaavan ψ klausuulia jää toteutumatta.

Kaava ψ on konjunkttiivisessa normaalimuodossa, mutta yksittäisessä klausuulissa voi olla q literaalia. Muodostamme ekvivalentin 3CNF-kaavan ψ' korvaamalla aina klausuulin

$$x_1 \vee x_2 \vee \dots \vee x_q$$

3CNF-kaavalla

$$(x_1 \vee x_2 \vee y_1) \wedge (\bar{y}_1 \vee x_3 \vee y_2) \wedge (\bar{y}_2 \vee x_4 \vee y_3) \wedge \dots \wedge (\bar{y}_{q-2} \vee x_{q-1} \vee x_q),$$

missä y_i ovat uusia muuttujia.

- Jos φ on toteutuva, jokin muuttujien arvoasetus toteuttaa kaavan ψ ja siis kaavan ψ' kaikki klausuulit.
- Jos φ ei ole toteutuva, millä tahansa arvoasetuksella ainakin $n^c/2$ kaavan ψ klausuulia ja siis myös ainakin $n^c/2$ kaavan ψ' klausuulia jää toteutumatta.

Kaavassa ψ' on $m \leq n^c 2^q (q - 2)$ klausuulia. Siis $\varepsilon_M = 1/(2^{q+1}(q - 2))$ toteuttaa lauseen vaatimuksen. \square

MAX-3SAT rajoitetuilla muuttujien esiintymillä

Ongelma MAX-3SAT(k) on kuten MAX-3SAT, mutta kukin muuttuja saa esiintyä korkeintaan k kertaa.

Lause 4.6: Olkoon ε_M kuten lauseessa 4.3 ja $\varepsilon_b = \varepsilon_M/85$. On olemassa polynomisessa ajassa laskettava muunnos yleisen MAX-3SAT-ongelman tapauksesta φ MAX-3SAT(29)-ongelman tapaukseksi ψ siten, että

- jos $\text{OPT}(\varphi) = m$, niin $\text{OPT}(\psi) = m'$
- jos $\text{OPT}(\varphi) < (1 - \varepsilon_M)m$, niin $\text{OPT}(\psi) < (1 - \varepsilon_b)m'$,

missä m ja m' ovat kaavojen φ ja ψ klausuulien lukumäärät.

Korollaaari 4.7: Jos $P \neq NP$, niin ongelmalla MAX-3SAT(29) ei ole $(1 - \varepsilon_b)$ -approksimointialgoritmia.

Todistus: Olkoon θ mielivaltainen SAT-tapaus, φ siitä lauseen 4.3 todistuksen mukaisesti muodostettu MAX-3SAT-tapaus ja ψ edelleen lauseen 4.6 mukaisesti muodostettu MAX-3SAT(29)-tapaus.

Lauseen 4.3 mukaan

- jos θ on toteutuva, niin $\text{OPT}(\varphi) = m$
- jos θ ei ole toteutuva, niin $\text{OPT}(\varphi) < (1 - \varepsilon_M)m$.

Lauseen 4.6 mukaan tästä seuraa, että

- jos θ on toteutuva, niin $\text{OPT}(\psi) = m'$
- jos θ ei ole toteutuva, niin $\text{OPT}(\psi) < (1 - \varepsilon_b)m'$.

Jos MAX-3SAT(29)-ongelmalla olisi $(1 - \varepsilon_b)$ -approksimointialgoritmi, niin nämä kaksi tapausta pystyttäisiin tunnistamaan polynomisessa ajassa. \square

Lauseen 4.6 todistus: Tarvitsemme todistuksessa laajentajaverkkoja (expander graph). Tunnetun tuloksen mukaan on olemassa sellaiset polynomisessa ajassa toimiva algoritmi A ja vakio N_0 , että millä tahansa $N \geq N_0$ algoritmi A tulostaa laajentajaverkon $G = (V, E)$, jolla

- jokaisen solmun asteluku on 14
- $|V| = k$, missä $N \leq k \leq 2N$
- kaikilla epätyhjillä $S \subset V$ pätee

$$|E(S, \bar{S})| > \min \{ |S|, |\bar{S}| \},$$

missä $E(S, \bar{S})$ on solmujoukkojen S ja \bar{S} välisten kaarten joukko.

Oletamme jatkossa, että kaavan φ jokainen muuttuja esiintyy ainakin N_0 kertaa. Jos näin ei ole, voimme ottaa kaavasta N_0 -kertaisen konjunktion itsensä kanssa muuttamatta mitään oleellista.

Olkoon B kaavan φ muuttujien joukko.

Tarkastellaan muuttujaa $x \in B$, joka esiintyy kaavassa φ yhteensä $N \geq N_0$ kertaa. Olkoon k kuten edellisellä kalvolla. Otetaan käyttöön joukko uusia muuttujia $V_x = \{x_1, \dots, x_k\}$.

Toistetaan tämä jokaisella muuttujalla $x \in B$. Muodostetaan kaava φ' korvaamalla jokaisella $x \in B$ jokainen muuttujan x esiintymä omalla muuttujallaan joukosta V_x .

Lauseen mukainen kaava ψ saadaan nyt muodossa

$$\psi = \varphi' \wedge \left(\bigwedge_{x \in B} \psi_x \right),$$

missä kaava ψ_x pakottaa hyvissä ratkaisuissa kaikki joukon V_x muuttujat samaan arvoon. Tarkemmin sanoen jos joukossa V_x ainakin yksi muuttuja on tosi ja ainakin yksi epätosi, niin kaavassa ψ_x on runsaasti epätosia klausuuleja.

Kaavan ψ_x muodostamiseen tarvitaan edellä esitetyn mukaista laajentajaverkkoa $G = (V_x, E)$, jonka solmuja siis ovat joukon V_x muuttujat. Kaava ψ_x on konjunktio, johon tulee jokaisella kaarella $(x_i, x_j) \in E$ klausuulit $x_i \vee \bar{x}_j$ ja $\bar{x}_i \vee x_j$.

Tarkastellaan nyt jotain muuttujien totuusarvoasetusta. Jaetaan joukon V_x muuttujat totuusarvonsa mukaisesti joukkoihin S ja \bar{S} . Olkoon S näistä pienempi. Verkon G laajennusominaisuuden nojalla jos $S \neq \emptyset$ ja $\bar{S} \neq \emptyset$, niin $E(S, \bar{S}) > |S|$. Siis kaavan ψ_x klausuuleista ainakin $|S| + 1$ on epätosia, jos kaikki joukon V_x muuttujat eivät saa samaa arvoa.

Tarkastellaan kaavaa

$$\psi = \varphi' \wedge \left(\bigwedge_{x \in B} \psi_x \right).$$

Sanotaan kaavan φ' klausuulien olevan tyyppiä I ja kaavojen ψ_x klausuulien tyyppiä II.

Kaava ψ on MAX-3SAT(29)-tapaus, sillä jokainen joukon V_x muuttuja esiintyy korkeintaan yhdessä tyyppin I klausuulissa ja tasan 28:ssa tyyppin II klausuulissa.

Olkoon τ arvoasetus MAX-3SAT(29)-tapauksen ψ muuttujille. Oletetaan, että jollain $x \in B$ arvoasetus τ antaa joukon V_x muuttujille sekä arvoa tosi että arvoa epätosi. Muodostetaan τ' asettamalla kaikki joukon V_x muuttujat siihen arvoon, joka on yleisempi asetuksessa τ . Olkoon S niiden muuttujien joukko, joiden arvo vaihtui. Kaavan ψ_x konstruktion perusteella ainakin $|S| + 1$ sen klausuulia muuttui todeksi, ja mikään ei muuttunut epätodeksi. Toisaalta kukin muuttuja esiintyy korkeintaan yhdessä tyyppin I klausuulissa, joten näistä korkeintaan $|S|$ muuttui epätodeksi. Siis ratkaisu parani.

Optimiratkaisussa siis jokaisella x kaikki joukon V_x muuttujat saavat saman arvon.

Kaavassa ψ on m tyyppin I klausuulia. Jokainen joukon V_x muuttuja esiintyy tasan 28:ssa tyyppin II klausuulissa, ja jokaisessa klausuulissa on kaksi muuttujaa. Koska kaavan φ muuttujaesiintymiä on korkeintaan $3m$, joukoissa V_x yhteensä on korkeintaan $6m$ muuttujaa, ja

$$m' \leq m + 6m \cdot \frac{28}{2} = 85m.$$

Jos φ on toteutuva, niin on myös ψ .

Olkoon toisaalta $\text{OPT}(\varphi) < (1 + \varepsilon_M)m$. Siis jos τ on kaavan ψ muuttujien arvoasetus, joka jokaisella x antaa joukon V_x muuttujille kaikille saman arvon, niin τ tekee yli $\varepsilon_M m \geq \varepsilon_M m' / 85$ tyyppin I klausuulia epätodeksi. Koska optimiratkaisulla todettiin olevan tämä ominaisuus, pätee $\text{OPT}(\psi) < (1 - \varepsilon_b)m'$. \square

Solmupeiteongelman approksimoinnin vaikeus

Olkoon $VC(d)$ solmupeiteongelma rajoitettuna verkkoihin, joissa kunkin solmun asteluku on korkeintaan d .

Lause 4.8: Olkoon ε_b kuten lauseessa 4.6 ja $\varepsilon_v = \varepsilon_b/2$. On olemassa polynomisessa ajassa laskettava muunnos MAX-3SAT(29)-ongelman tapauksesta φ VC(30)-ongelman tapaukseksi $G = (V, E)$ siten, että

- jos $\text{OPT}(\varphi) = m$, niin $\text{OPT}(G) \leq \frac{2}{3}|V|$
- jos $\text{OPT}(\varphi) < (1 - \varepsilon_b)m$, niin $\text{OPT}(G) > (1 + \varepsilon_v)\frac{2}{3}|V|$,

missä m on kaavan φ klausuulien lukumäärä.

Yhdistämällä tämä edellisiin tuloksiin saadaan

Korollaari 4.9: Jos $P \neq NP$, niin ongelmalla VC(30) ei ole $(1 + \varepsilon_v)$ -approksimointialgoritmia. \square

Todistus: Voimme olettaa, että jokaisessa klausuulissa on tasan kolme literaalia (muuten toistetaan jotain literaalia).

Palautus on sama kuin todistettaessa solmupeiteongelman NP-täydelliseksi. Jokaista kaavan φ klausuulia kohti verkkoon G tulee kolme solmua, jotka merkitään klausuulissa esiintyvillä literaaleilla ja yhdistetään toisiinsa kaarilla. Lisäksi kaikilla muuttujilla x yhdistetään kaikki sellaiset solmuparit, joista toinen on merkitty literaalilla x ja toinen literaalilla \bar{x} .

Verkon G suurimman riippumattoman joukon koko on $\text{OPT}(\varphi)$.

Olkoon nimittäin S riippumaton joukko. Voimme asettaa todeksi kaikki joukon S solmuihin liittyvät literaalit, jolloin saatu muuttujien arvoasetus toteuttaa ainakin $|S|$ klausuulia.

Toisaalta jos τ on muuttujien arvoasetus, joka toteuttaa r klausuulia, voimme valita jokaista toteutuvaa klausuulia vastaavista solmuista yhden, jota vastaava muuttuja on tosi. Näistä muodostuu r -solmuinen riippumaton joukko.

Verkon G suurimman riippumattoman joukon koko on $\text{OPT}(\varphi)$. Suurimman riippumattoman joukon komplementti on pienin solmupeite. Siis $\text{OPT}(G) = |V| - \text{OPT}(\varphi)$.

Jos $\text{OPT}(\varphi) = m$, niin $\text{OPT}(G) = 2m = \frac{2}{3}|V|$.

Jos $\text{OPT}(\varphi) < (1 - \varepsilon_b)m$, niin $\text{OPT}(G) > 3m - (1 - \varepsilon_b)m = (1 + \varepsilon_v) \cdot \frac{2}{3}|V|$. \square

Klikkiongelman approksimoimisen vaikeus

Tarkastelemme lukumääräistä klikkiongelmaa. Siis tehtävänä on löytää verkosta G mahdollisimman suuri täydellinen aliverkko. Jos $P \neq NP$, niin jollain $\varepsilon > 0$ klikkiongelmalle ei ole $(1/n^\varepsilon)$ -approksimointialgoritmia.

Osoitamme ensin helpomman tuloksen, josta seuraa, että klikkiongelmallalla ei ole $(1/2)$ -approksimointialgoritmia.

Lemma 4.10: On olemassa sellaiset vakiot b ja q , että SAT-tapaus φ voidaan polynomisessa ajassa muuntaa sellaiseksi klikkitapaukseksi $G = (V, E)$, että $|V| = 2^q n^b$ ja

- jos φ on toteutuva, niin $\text{OPT}(G) \geq n^b$
- jos φ ei ole toteutuva, niin $\text{OPT}(G) < n^b/2$.

Todistus: Olkoon F ongelman SAT PCP($\log n, 1$)-tarkastaja, joka käyttää $b \log_2 n$ satunnaisbittiä ja lukee q bittiä todistuksesta. Verkkoon G tulee solmu $v_{r,\tau}$ jokaiselle satunnaisbittijonolle $r \in \{0, 1\}^{b \log_2 n}$ ja vastausjonolle $\tau \in \{0, 1\}^q$.

Bittijonolle r olkoot $Q(r)$ ne q positiota, jotka F lukee todistuksesta, jos r on satunnaisjono.

Solmu $v_{r,\tau}$ on **hyväksyvä**, jos F hyväksyy satunnaisjonolla r , kun todistuksesta luettavien bittien arvot ovat τ . Muuten $v_{r,\tau}$ on **hylkäävä**.

Solmut v_{r_1,τ_1} ja v_{r_2,τ_2} ovat **yhteensopivat**, jos τ_1 ja τ_2 antavat samat arvot niille biteille, jotka ovat mukana sekä $Q(r_1)$:ssä että $Q(r_2)$:ssa. Tällöin jos $v_{r_1,\tau_1} \neq v_{r_2,\tau_2}$, niin $r_1 \neq r_2$.

Solmujen v_{r_1,τ_1} ja v_{r_2,τ_2} välille tulee kaari, jos ne ovat yhteensopivat ja kumpikin hyväksyviä.

Solmu $v_{r,\tau}$ on **yhteensopiva** todistuksen p kanssa, jos todistuksen p positiot $Q(r)$ sisältävät bitit τ .

Jos φ on toteutuva, jollain todistuksella p tarkastaja F aina hyväksyy. Olkoon $p(r)$ todistuksen p bitit positioissa $Q(r)$. Nyt kaikki solmut $v_{r,p(r)}$ ovat hyväksyviä ja pareittain yhteensopivia. Ne siis muodostavat klikin, ja niitä on yhtä monta kuin satunnaisjonoja eli $2^{b \log_2 n} = n^b$.

Oletetaan nyt, että φ ei ole toteutuva. Olkoon C klikki. Siis sen solmut ovat hyväksyviä ja pareittain yhteensopivia. Pareittaisen yhteensopivuuden perusteella voidaan muodostaa todistus p , joka on yhteensopiva kaikkien klikin C solmujen kanssa.

Jos r on satunnaisjono, jolla $v_{r,\tau}$ on klikissä C jollain τ , niin todistuksella p tarkastaja F hyväksyy, jos satunnaisjono on r . Tällaisia satunnaisjonoja on ainakin $|C|$ kappaletta. Koska hyväksymistodennäköisyys on alle $1/2$, pätee $|C| < n^b/2$. \square

Haluaisimme nyt tiukentaa tuloksen osoittamaan, että klikkiongelmalla ei ole $(1/n^\varepsilon)$ -approksimointialgoritmia jollain vakiolla ε .

Approksimointisuhteen raja $1/2$ tuli edellä suoraan siitä, että määritelmän mukaan SAT-tarkastajan hyväksymistodennäköisyys ei-toteutuville kaavoille oli alle $1/2$.

Hyväksymistodennäköisyys $1/n^\varepsilon$ saataisiin toistamalla todennäköisyydellä alle $1/2$ hyväksyvää tarkastajaa $\log_2(n^\varepsilon) = \varepsilon \log_2 n$ kertaa. Tähän tarvittaisiin $O((\log n)^2)$ satunnaisbittiä ja $O(\log n)$ bittiä todistuksesta.

Jos tarkastaja käyttää $r(n)$ satunnaisbittiä ja $q(n)$ bittiä todistuksesta, edellisessä palautuksessa rakennettuun verkkoon tulee $2^{r(n)+q(n)}$ solmua. Tältä kannalta siis $q(n) = O(\log n)$ olisi vielä hyväksyttävää, mutta $r(n) = \Omega((\log n)^2)$ johtaa ei-polynomiseen kokoon.

Meidän pitää siis hieman viritellä satunnaisbittien, todistuksesta luettujen bittien ja hyväksymistodennäköisyyden välistä hyötysuhdetta.

Olkoon $\text{PCP}_{c,s}(r(n), q(n))$ niiden kielten L joukko, joille on seuraavat ehdot täyttävä tarkastaja V :

- tarkastaja V toimii polynomisessa ajassa, käyttää $O(r(n))$ satunnaisbittiä ja lukee $O(q(n))$ bittiä todistuksesta
- jos $x \in L$, niin V hyväksyy ainakin todennäköisyydellä c
- jos $x \notin L$, niin V hyväksyy alle todennäköisyydellä s .

Siis aiemmin määritelty $\text{PCP}(r(n), q(n))$ on sama kuin $\text{PCP}_{1,1/2}(r(n), q(n))$.

Lause 4.11: $\text{NP} = \text{PCP}_{1,1/n}(\log n, \log n)$.

Ennen lauseen todistusta katsomme, miten siitä seuraa haluttu approksimoitumattomuustulos.

Lause 4.12: Joillain vakioilla b ja q on olemassa sellainen polynomisessa ajassa laskettava kuvaus SAT-tapauksesta φ klikkitapaukseen $G = (V, E)$, että kun $|\varphi| = n$, niin $|V| = n^{b+q}$ ja

- jos φ on toteutuva, niin $\text{OPT}(G) \geq n^b$
- jos φ ei ole toteutuva, niin $\text{OPT}(G) < n^{b-1}$.

Korollaari 4.13: Jos $P \neq NP$, niin lukumääräisellä klikkiongelmalla ei ole $(1/n^{\varepsilon_q})$ -approksimointiongelmaa, missä $\varepsilon_q = 1/(b+q)$ edellisen lauseen vakioille b ja q . \square

Lauseen 4.12 todistus: Kiinnitetään SAT-ongelman $\text{PCP}_{1,1/n}(\log n, \log n)$ -tarkastaja F , joka käyttää $b \log_2 n$ satunnaisbittiä ja lukee $q \log_2 n$ bittiä todistuksesta. Kaavasta φ muodostetaan verkko G aivan kuten lemmän 4.10 todistuksessa. Siihen tulee siis $2^{b \log_2 n + q \log_2 n} = n^{b+q}$ solmua.

Jos φ on hyväksyvä, jokin todistus p johtaa aina hyväksymiseen ja vastaavat solmut $v_{r,p(r)}$ muodostavat n^b -solmun klikin, kuten lemmassa 4.10.

Jos φ ei ole hyväksyvä, jokaisesta klikistä C saadaan $|C|$ satunnaisjonoa ja yhteensopivat todistuksen bitit, joilla F hyväksyy. Siis $|C|/n^b < 1/n$. \square

Palataan nyt PCP($\log n, 1$)-tarkastajan hyväksymistodennäköisyyden pienentämiseen. Olkoon kuten edellä $b \log_2 n$ tarkastajan tarvitsemien satunnaisbittien lukumäärä hyväksymistodennäköisyyden saamiseksi alle $1/2$:n lukien vakiomäärä todistuksen bittejä.

Konstruoidaan verkko H , jossa on n^b solmua ja jokaisen solmun asteluku on jokin vakio d . Nimetään kukin solmu omalla $b \log_2 n$ bitin jonollaan.

Suoritetaan tässä verkossa satunnaiskulku siten, että lähdetään satunnaisesta alkusolmusta ja aina siirrytään johonkin satunnaiseen naapuriin, kumpikin tasaisen jakauman mukaan. Tehdään $k \log_2 n$ siirtymää jollain k ja laitetaan vierailtujen solmujen nimet peräkkäin.

Saamme $bk(\log_2 n)^2$ bitin pseudosatunnaisen jonon. Koska d on vakio, tarvitsemme $O(\log n)$ aitoa satunnaisbittiä tämän jono tuottamiseen. Saadun jonon bitit eivät ole riippumattomia. Osoittautuu kuitenkin, että jos H on [laajentajaverkko](#), bitit ovat riittävän satunnaisia tarpeisiimme.

Jos $H = (V, E)$ on tasa-asteinen, niin satunnaiskulun tasapainojakauma on tasainen. Jos $H = (V, E)$ on lisäksi laajentajaverkko, niin pätee

Lause 4.14: Olkoon $S \subset V$ sellainen, että $|S| < n^b/2$. Jollain vakiolla k todennäköisyys, että $k \log n$ askelen satunnaiskulussa kaikki solmut olisivat joukossa S , on alle $1/n$. \square

Tässä siis edelleen $|V| = n^b$. Tämä tulos (jota emme todista) on juuri riittävää tarpeisiimme.

Lauseen 4.11 todistus: Todistamme vain vaikeamman suunnan

$$\text{PCP}_{1,1/2}(\log n, 1) \subseteq \text{PCP}_{1,1/n}(\log n, \log n),$$

joka on samalla se suunta, jota edellä tarvitsimme.

Olkoon siis $L \in \text{PCP}_{1,1/2}(\log n, 1)$ ja F tämän mukainen tarkastaja. Muodostamme $\text{PCP}_{1,1/n}(\log n, \log n)$ -tarkastajan F' .

Tarkastaja F' muodostaa ensin vakioasteisen laajentajaverkon H kuten edellä. Verkon H avulla F' tuottaa $O(\log n)$ alkuperäisestä satunnaisbitistään $k \log n$ tilan pituisen satunnaiskulun. Satunnaiskulun solmujen nimistä saadaan $k \log n$ pseudosatunnaista bittijonoa, joissa kussakin on $b \log n$ bittiä. Nyt F' simuloi tarkastajaa F kaikilla näillä bittijonoilla ja hyväksyy, jos kaikki simulaatiot hyväksyivät.

Jos $x \in L$ ja p on todistus, jolla tarkastaja F aina hyväksyy, niin selvästi p kelpaa myös tarkastajalle F' .

Olkoon $x \notin L$ ja p tarkastajalle F' tarjottu todistus. Olkoon S niiden verkon H solmujen joukko, joita vastaavalla satunnaisjonolla F hyväksyy. Koska $|S| < n^b/2$, niin satunnaiskulun todennäköisyys pysyä koko ajan tässä joukossa on alle $1/n$. Tämä kuitenkin tarvittaisiin, että F' hyväksyisi. \square

Todetaan vielä lopuksi ilman todistusta pari muuta keskeistä PCP-tulosta:

Lause 4.15: Jos jollain $\varepsilon > 0$ lukumääräisellä joukkopeiteongelmalla on $((1 - \varepsilon) \ln n)$ -approksimointialgoritmi, niin

$$\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)}).$$

Lause 4.16: SAT-ongelmalle on PCP-tarkastaja, joka $O(\log n)$ satunnaisbitin perusteella laskee kolme osoitetta todistukseen, olkoot ne i , j ja k , sekä satunnaisbitin b , ja hyväksyy jos

$$y(i) + y(j) + y(k) \equiv b \pmod{2}.$$

5. Lopuksi

Esimerkkinä kurssin ulkopuolelle jääneestä tärkeästä ongelmaluokasta mainitaan tässä **#P-täydelliset lukumääräongelmat** (counting problems). Kokonaislukuarvoinen funktio f kuuluu luokkaan #P (luetaan "number P" tai "sharp P"), jos jollain epädeterministisellä polynomisessa ajassa toimivalla Turingin koneella M pätee, että $f(x)$ on koneen M hyväksyvien laskentojen lukumäärä syötteellä x .

CNF-kaavan toteuttavien totuusarvoasetusten lukumäärän laskeminen on esimerkki luokan #P täydellisestä ongelmasta (luonnollisesti määritellyn palautuksen suhteen).

Selvästi jos $P \neq NP$, niin #P-täydellistä funktiota ei voi laskea tarkasti polynomisessa ajassa. Myös monella luokkaan P kuuluvalla päätösongelmalla vastaava lukumääräongelma on #P-täydellinen.

CNF-kaavojen toteuttavia totuusarvoasetuksia ei voi edes approksimoida millään ei-triviaalilla approksimointisuhteella, sillä vaikea tapaus on erottaa tilanteet $f(\varphi) = 0$ ja $f(\varphi) \geq 1$, missä funktion f maksimiarvo on 2^n .

Merkittävä käänne oli Jerrumin ja Sinclairin (1989) satunnainen täysin polynominen approksimointiskeema (FPRAS) tiheän kaksijakoisen verkon täydellisten pariutusten laskemiselle (ja joukolle muita ongelmia). Tulos perustuu nopeasti sekoittuviin Markovin ketjuihin.

Itse asiassa osoittautuu, että yleisemmin jos $\#P$ -täydellinen ongelma on approksimoitavissa jollain ei-triviaalilla tarkkuudella, niin sillä on FPRAS.

Yhteenveto

Kurssilla käsiteltiin (enimmäkseen deterministisiä) **polynomisessa** ajassa toimivia algoritmeja, joiden tulos on **todistettavasti** jonkin kertoimen sisällä optimaalisesta.

Käytännössä ehkä tärkeämpiä ovat

- heuristiikat, jotka ovat nopeita mutta joiden tuloksille ei ole hyvyystakuuta
- ei-polynomiset algoritmit (branch-and-bound, dynaaminen ohjelmointi).

Approksimointitulokset antavat kuitenkin intuitiota ongelman luonteesta ja siitä, mitkä ovat vaikeita tapauksia.

Approksimointisuhde α saadaan osoittamalla jokaiselle syötteelle (olettaen minimointiongelma)

- alaraja a optimiratkaisulle
- yläraja b algoritmin tuottamalle ratkaisulla
- toteamalla $b \leq \alpha a$.

Siis keskeinen piirre nimenomaan approksimointialgoritmien analyysissä on alarajatekniikat.

Eryteisesti lineaarisena kokonaislukuohjelmana esitetyille ongelmille saatiin alarajoja

- murtolukuratkaisusta (joka sitten pyöristettiin)
- duaaliratkaisusta (joka jollain tapaa konstruointiin rinnan primaaliratkaisun kanssa).