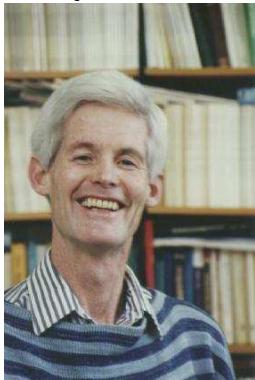


P versus NP I

- Kieliluokat P ja NP näyttävät täysin erilaisilta:
 - a) P on kieliluokka, jossa alkion kuuluminen kieleen voidaan ratkaista polynomisessa ajassa deterministisellä Turingin koneella.
 - b) NP on kieliluokka, jossa alkion kuuluminen kieleen voidaan verifioida polynomisessa ajassa; eli alkion kuuluminen kieleen voidaan ratkaista polynomisessa ajassa epädeterministisellä Turingin koneella.
- Selvästi $P \subset NP$.
- Sen sijaan ei tiedetä, päteekö $P \neq NP$.
- Yleisesti uskotaan, että näin todella on.
- Clay-instituutti on tarjonnut miljoona dollaria sille, joka ratkaisee P vs. NP kysymyksen.
- Probleema on ollut avoin 1970-luvun alusta lähtien.

P versus NP II

- Samoihin aikoihin kysymyksessä saavutettiin edistystä, kun **Stephen Cook** ja **Leonid Levin** esittelivät **NP-täydellisyyden** käsitteen.



Polynominen palautus I

- NP-täydellisuuden määrittelemisen perustuu **polynomisen palautuvuuden käsitteeseen**.
- Se tarkoittaa havainnollisesti, että probleema voidaan kääntää eli palauttaa toiseksi probleemaksi, jonka ratkaisusta saadaan sitten alkuperäisen ratkaisu.
- Joskus käänös on suoraviivainen, joskus hyvinkin mutkikas.

Määritelmä

Funktio $f : \Sigma^ \rightarrow \Sigma^*$ on **polynomisesti laskettavissa**, jos on olemassa polynomisessa ajassa toimiva Turingin kone, joka kirjoittaa pysähtyessään nauhalle arvon $f(w)$, kun alussa nauhalla on $w \in \Sigma^*$.*

Polynominen palautus II

Määritelmä

Kieli A on *polynomisesti palautettavissa* kieleen B , $A \leq_p B$ (myös $A \leq_p^m B$), jos on olemassa polynomisesti laskettavissa oleva funktio $f : \Sigma^* \rightarrow \Sigma^*$, jolle kaikilla $w \in \Sigma^*$ pätee ehto

$$w \in A \iff f(w) \in B.$$

Funktiota f kutsutaan *polynomiseksi palautukseksi* A :sta B :hen.

Palautuksen ei siis tarvitse olla bijektio eikä injektiokaan, mikä korostaa vaihtoehtoinen merkintä \leq_p^m ($m = \text{many-to-one}$).

Lause

Jos $A \leq_p B$ ja $B \in P$, niin $A \in P$.

Todistus. Olkoon M polynomisessa ajassa toimiva Turingin kone, joka ratkaisee B :n, ja olkoon f polynominen palautus A :sta B :hen. Nyt

Polynominen palautus III

seuraava polynominen algoritmi (Turingin kone) ratkaisee A :n, kun syötteenä on w :

- 1 Laske $f(w)$.
- 2 Aja M syötteellä $f(w)$ ja tulosta se, mitä M lopuksi tulostaa.

Kaikki toimii polynomisessa ajassa (koska polynomien yhdiste on polynomi) ja $w \in A$ aina kun $f(w) \in B$. \square

Esimerkki: Nimien järjestelyn palautus lukujen järjestelyyn I

- Ensin hyvin yksinkertainen palautus.
- Oletetaan, että käytössä on kokonaislukuja järjestelevä algoritmi.
- Näytetään, miten nimien eli merkkijonojen järjestely voidaan palauttaa lukujen järjestelemiseen, jos emme aseta mitään rajoja lukujen suuruudelle.
- Merkkijono voidaan kuvata injektiivisesti luvulle siten, että tulkitaan jokainen kirjain bittijonoksi, kuten tietokoneessa tehdään ja laitetaan bittijonot peräjälkeen. Tämä pitempi bittijono tulkitaan luvuksi.
- Selvästi muunnos voidaan tehdä polynomisessa ajassa merkkijonon pituuden suhteen.

Esimerkki: 3SAT:n palautus CLIQUE-ongelmaan I

Seuraavaksi hieman vaikeampi tapaus. Tarvitsemme ensin käsitteitä:

- **Literaali** on Boolean muuttuja tai sellaisen negaatio, x tai \bar{x} .
- **Disjunkttiivinen lause** (engl. clause) koostuu useista literaaleista, jotka on yhdistetty tai-konnektiivilla: $(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4)$.
- Boolean lause on **konjunkttiivisessa normaalimuodossa** eli cnf-muodossa, jos se koostuu useasta disjunkttiivisesta lauseesta, jotka on yhdistetty konjunktioilla:

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6).$$

- Cfn-muoto on **3cfn-muoto**, jos kaikissa disjunkttiivisissa lauseissa on täsmälleen kolme literaalia:

$$(x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_5 \vee x_6) \wedge (x_3 \vee \bar{x}_6 \vee x_4) \wedge (x_4 \vee x_5 \vee x_6).$$

Esimerkki: 3SAT:n palautus CLIQUE-ongelmaan II

- **3SAT** on sellaisten 3cn-muodossa olevien lauseiden joukko, jotka sopivilla muuttujien totuusarvosijoituksilla tulevat todeksi. Siten kaavassa täytyy olla vähintään yksi literaali, jonka totuusarvo on tosi.

Lause

3SAT voidaan palauttaa polynomisesti CLIQUE-ongelmaan.

Todistus. Olkoon ϕ 3SAT:in kaava, jossa on k disjunkttiivista lausetta:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k).$$

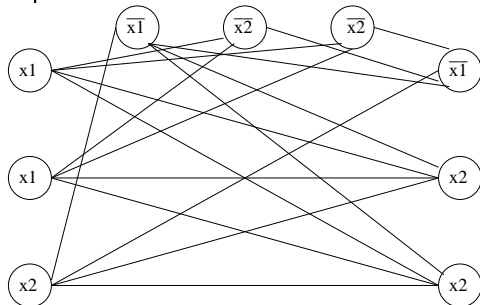
Palautuksessa f kaavasta ϕ generoidaan pari (G, k) , missä G on suuntaamaton verkko. Pari saadaan seuraavasti.

- Verkon solmuiksi otetaan jokaisen disjunkttiivisen lauseen literaalit. Samaa literaalia saattaa siis vastata useampi solmu, jos literaali esiintyy useassa lauseessa.
- Ryhmitellään solmut siten, että lauseen kolme literaalia tulevat aina samaan, kolmen solmun ryhmään.
- Solmun nimi on sama kuin se literaali, jota se edustaa. Siten eri solmuilla voi olla sama nimi.
- Vedetään särmiä solmujen välille seuraavasti:
 - i) Lausetta edustavien kolmen solmun välillä ei ole särmiä.
 - ii) Särmiä ei vedetä myöskään kahden vastakkaisen solmun välille, kuten ei solmujen x_2 ja $\overline{x_2}$ välille.
 - iii) Muiden solmujen välille vedetään särmiä.

Seuraava kuva näyttää, miten konstruktio tapahtuu kaavan

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

tapauksessa.



Seuraavaksi täytyy osoittaa, että muunnoksella on palautuksen ominaisuudet. Eli jos kaava on toteutuva, niin verkossa on k-klikki.

- Oletetaan, että kaava ϕ on toteutuva. Tällöin jokaisessa disjunkttiivisessa lauseessa on ainakin yksi literaali, joka on tosi.
- Valitaan jokaisesta disjunkttiivista lausetta edustavasta solmukolmikosta yksi solmu, joka vastaa totta literaalia. Jos kolmikossa on useampi kuin yksi tällainen solmu, valitaan jokin niistä.
- Nyt näin valitut solmut muodostavat k -klikin.
- Ensinnäkin solmuja valittiin k kpl.
- Valituista solmuista jokaisen kahden välillä on särmä, koska kumpikaan särmän poissulkevista ehdoista ei ole voimassa.
- Siten verkossa on tosiaan klikki. Jää osoitettavaksi, että mikäli verkossa on k -klikki, vastaava kaava on toteutuva.

- Mitkään kaksi klikin solmua eivät sijaitse samassa kolmikossa, koska kolmikon solmuja ei ole yhdistetty toisiinsa särmillä. Siten jokaisessa k kolmikossa on yksi klikin solmu.
- Valitaan totuusarvoasetus siten, että klikin solmuja vastaavat literaalit tulevat todeksi.
- Tämä on mahdollista, koska kahta vastakkaista literaalia ei yhdistetä särmällä, eikä sellaiset solmut voi siten esiintyä yhtäaikaan klikissa.
- Tämä totuusarvoasetus tekee kaavasta ϕ toden, sillä jokainen disjunkttiivinen lause tulee todeksi. \square

NP-täydellisyys I

Määritelmä

Kieli B on **NP-täydellinen**, jos se täyttää seuraavat kaksi ehtoa:

- 1 $B \in \text{NP}$ ja
- 2 jokainen NP:n kieli A on polynomisesti palautettavissa B :hen.

Lause

Jos B on NP-täydellinen ja $B \in P$, niin $P = \text{NP}$.

Todistus. Tämä seuraa suoraan polynomisen palautuksen määritelmästä. \square

Lause

Jos B on NP-täydellinen ja $B \leq_P C$, $C \in \text{NP}$, niin C on myös NP-täydellinen.

Todistus.

NP-täydellisyys II

- Tiedetään, että C on NP:ssä, joten riittää näyttää, että mielivaltainen NP:n kieli A on polynomisesti palautettavissa C :hen.
- Koska B on NP-täydellinen, voidaan A palauttaa polynomisesti B :hen.
- Toisaalta B voidaan palauttaa polynomisesti C :hen oletuksen nojalla.
- Nyt polynomisten palautusten yhdiste on edelleen polynominen palautus. Siten A on polynomisesti palautettavissa C :hen ja C on täten NP-täydellinen. \square

Cookin ja Levinin lause

Määritelmä

SAT on toteutuvien Boolean lauseiden joukko.

Lause

SAT on NP-täydellinen.

Todistus. Näytetään ensin, että SAT kuuluu NP:hen. Tämä on helppo osuus todistuksessa.

- Tehdään epädeterministinen kone, joka arvaa muuttujien totuusarvon.
- Tämän jälkeen on helppo testata polynomisessa ajassa, onko annettu kaava toteutuva noilla asetuksilla.

Seuraavaksi osoitetaan, että jokainen probleema NP:ssä voidaan palauttaa SAT:iin.

- Ainoa tapa tehdä tämä on näyttää, miten mielivaltainen epädeterministinen Turingin kone voidaan esittää kohtuullisen kokoisena Boolean lausekkeena.
- Tarkemmin: Jokaista Turingin koneen syötettä x kohti konstruoidaan Boolean kaava F_x , joka on toteutuva jos ja vain jos kone hyväksyy x :n.
- Tehdään ensin yleinen ratkaisu ja näytetään tämän jälkeen, minkälainen kaava syntyy eräässä suppeassa konkreettisisessä tilanteessa. Päinvastainen järjestys ei oikein kannata, sillä erikoistapauksessakin joudutaan määrittelemään lähes kaikki, mitä yleisessäkin.

- Olkoon siis

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{yes}}, q_{\text{no}})$$

epädeterministinen Turingin kone, ja $p(n) \geq n$ polynomi, jolla

$$\text{time}_M \leq p(n)$$

kaikilla n .

- Voidaan olettaa, että kone on yksinauhainen.
- Oletetaan lisäksi, että

$$Q = \{q_0, q_1, \dots, q_r\},$$

missä $q_{r-1} = q_{\text{yes}}$ ja $q_r = q_{\text{no}}$, ja että

$$\Gamma \cup \{>, <\} = \{s_0, s_1, \dots, s_{m+1}\},$$

missä $s_0 = >$ ja $s_{m+1} = <$.

- Koska M toimii ajassa $p(n)$, mikään syötteellä mahdollinen laskenta ei voi edetä nauhalla pidemmälle kuin merkkipaikkaan $p(|x|) + 1$.

- Syötettä x , $|x| = n$, vastaavassa kaavassa F_x käytetään seuraavia muuttujaperheitä:
 - 1 $q[t, k]$, $0 \leq t \leq p(n)$, $0 \leq k \leq r$,
eli hetkellä t (so. t :n laskenta-askeleen jälkeen) M on tilassa q_k .
 - 2 $h[t, i]$, $0 \leq t \leq p(n)$, $0 \leq i \leq p(n) + 1$
eli hetkellä t M :n nauhapää on merkkipaikan i kohdalla.
 - 3 $s[t, i, j]$, $0 \leq t \leq p(n)$, $0 \leq i \leq p(n) + 1$, $0 \leq j \leq m + 1$
eli hetkellä t M :n nauhan merkkipaikassa i on merkki s_j .
- Muuttujat on tässä selvyuden vuoksi ryhmitelty ikäänkuin taulukoiksi, mutta ne voitaisiin tietenkin haluttaessa uudelleen nimetä miten tahansa.

- Kukin koneen M mahdollinen laskenta syötteellä x määrää näille muuttujille totuusarvot ilmeisellä tavalla. Jos laskenta pysähtyy enen hetkeä $t = p(n)$, ajatellaan koneen tilanteen pysyvän samana hetkeen $p(n)$ saakka. Toisaalta läheskään kaikki muuttujien totuusarvot eivät vastaa mahdollisia laskentoja.
- Tarkoituksena on muodostaa näistä muuttujista kaava F_x niin, että annettu totuusarvoasetus toteuttaa F_x :n jos ja vain jos **se vastaa jotakin M :n hyväksyvää laskentaa syötteellä x .**
- Kun vielä todetaan, että kaava F_x voidaan muodostaa merkkijonosta x ja koneen M kuvauksesta deterministisesti polynomisessa ajassa, nähdään että kuvaus $f_M : x \mapsto F_x$ on haluttu palautus $f : L(M) \leq_m^P \text{SAT}$.

- Kaava F_x on muodoltaan kuuden alikaavan tai "osaehdon" konjunktio:

$$F_x = G_1 \wedge G_2 \wedge G_3 \wedge G_4 \wedge G_5 \wedge G_6,$$

tai indeksoitua lyhennysmerkintää käyttäen

$$F_x = \bigwedge_{i=1}^6 G_i.$$

- Alikaavojen kuvaamat ehdot ovat:

- G_1 : Kullakin hetkellä t koneen tila on yksikäsitteisesti määrätty.
- G_2 : Kullakin hetkellä t koneen nauhapään osoittama paikka on yksikäsitteisesti määrätty
- G_3 : Kullakin hetkellä t kussakin nauhan merkki-paikassa on yksikäsitteisesti määrätty merkki.
- G_4 : Hetkellä 0 koneen tilanne on alkutilanne syötteellä x .
- G_5 : Hetkellä $p(n)$ kone on hyväksyvässä lopputilassa.
- G_6 : Kullakin hetkellä t , $0 \leq t \leq p(n) - 1$, koneen tilanteen muutos hetkestä t hetkeen $t + 1$ on siirtymäfunktion δ mukainen.

Viisi ensimmäistä ehtoa voidaan toteuttaa seuraavilla kaavoilla:

$$G_1 = \bigwedge_{0 \leq t \leq p(n)} \bigvee_{0 \leq k \leq r} q[t, k] \wedge \bigwedge_{\substack{0 \leq t \leq p(n) \\ 0 \leq k' \leq r, k' \neq k}} \neg(q[t, k] \wedge q[t, k']);$$

$$G_2 = \bigwedge_{0 \leq t \leq p(n)} \bigvee_{1 \leq i \leq p(n)+1} h[t, i] \wedge$$

$$\bigwedge_{\substack{0 \leq t \leq p(n) \\ 0 \leq i < i' \leq p(n)+1}} \neg(h[t, i] \wedge h[t, i']);$$

$$G_3 = \bigwedge_{\substack{0 \leq t \leq p(n) \\ 0 \leq i \leq p(n)+1}} \bigvee_{0 \leq j \leq m+1} s[t, i, j] \wedge \\ \bigwedge_{\substack{0 \leq t \leq p(n) \\ 0 \leq i \leq p(n)+1 \\ 0 \leq j < j' \leq m+1}} \neg(s[t, i, j] \wedge s[t, i, j']);$$

$$\begin{aligned}
G_4 &= q[0, 0] \wedge h[0, 1] \wedge s[0, 0, 0] \wedge \bigwedge_{1 \leq i \leq n} s[0, i, j_i] \wedge \\
&\quad \bigwedge_{n+1 \leq i \leq p(n)+1} s[0, i, m+1], \\
&\quad \text{kun } x = s_{j_1} s_{j_2} \dots s_{j_n}; \\
G_5 &= q[p(n), r-1].
\end{aligned}$$

Alikaava G_6 kuvaa koneen tilaa, nauhapään sijainnin ja kunkin nauhamerkin muuttumista yhdessä laskenta-askeleessa. Se koostuu kolmesta osasta:

$$G_6 = G'_6 \wedge G''_6 \wedge G'''_6.$$

Kaava G'_6 toteaa vain sen, että jos hetkellä t koneen nauhapää ei ole merkkipaikan i kohdalla, niin paikassa i oleva merkki pysyy samana hetkellä $t + 1$:

$$G'_6 = \bigwedge_{\substack{0 \leq t \leq p(n)-1 \\ 0 \leq i \leq p(n)+1 \\ 0 \leq j \leq m+1}} ((s[t, i, j] \wedge \neg h[t, i]) \longrightarrow s[t + 1, i, j]).$$

Kaava G_6'' toteaa sen, että jos kone hetkellä t on lopputilassa 'yes' tai 'no', niin sen tilanne hetkellä $t + 1$ on sama kuin hetkellä t :

$$G_6'' = \bigwedge_{\substack{0 \leq t \leq p(n)-1 \\ k=r-1, r \\ 0 \leq i \leq p(n)+1 \\ 0 \leq j \leq m+1}} [(q[t, k] \wedge h[t, i] \wedge s[t, i, j]) \longrightarrow (q[t + 1, k] \wedge h[t + 1, i] \wedge s[t + 1, i, j])]$$

Kaava G_6''' lopulta formalisoi sen keskeisen vaatimuksen, että ellei kone ole ajanhetkeen t mennessä pysähtynyt, niin hetkestä t hetkeen $t + 1$ sen tila, nauhapään sijainti ja nauhapään kohdalla oleva merkki muuttuvat siirtymäfunktion δ mukaisella tavalla:

$$G_6''' = \bigwedge_{\substack{0 \leq t \leq p(n) - 1 \\ 0 \leq k \leq r - 2 \\ 0 \leq i \leq p(n) + 1 \\ 0 \leq j \leq m + 1}} [A_6''' \longrightarrow B_6'''],$$

missä

$$A_6''' = q[t, k] \wedge h[t, i] \wedge s[t, i, j]$$

$$B_6''' = \bigvee_{(q_{k'}, s_{j'}, \Delta) \in \delta(q_k, s_j)} (q[t + 1, k'] \wedge h[t + 1, i + \Delta] \wedge s[t + 1, i, j'])$$

Nauhapään siirtosuunta on tässä ajateltu koodatuksi siten, että suuntaa L vastaa arvo $\Delta = -1$ ja suuntaa R arvo $\Delta = 1$.

Suoraviivainen tarkastus osoittaa, että merkkijonosta x , $|x| = n$, näin muodostettu kaava F_x on

- polynomista kokoa n :n suhteen, itse asiassa $\mathcal{O}(p(n)^3)$,
- se voidaan muodostaa deterministisesti polynomisessa ajassa,
- ja se on toteutuva jos ja vain jos $x \in L(M)$.

Siten kuvaus $f : x \mapsto F_x$ on palautus kielestä $A = L(M)$ kieleen SAT. \square

- Hieman myöhemmin James Karp esitteli toistakymmentä NP-täydellistä problemaa.



- Jos yhdellä NP-täydellisellä probleemalla on polynominen ratkaisu, niin kaikilla muillakin on!
- Uusien NP-täydellisten ongelmien löytäminen ei edellytä yhtä mutkikasta todistamista kuin SAT-ongelman yhteydessä.

- Nimittäin jatkossa riittää antaa vain polynominen palautus SAT:iin tai johonkin muuhun, jo NP-täydelliseksi tiedettyyn ongelmaan.
- Tässä yhteydessä erityisen käyttökelpoinen on aikaisemmin esitelty 3SAT-ongelma, joka osoitetaan NP-täydelliseksi CSAT-ongelman avulla.

Määritelmä

Lausekalkyylin kaava F on *konjunkttiivisessa normaalimuodossa, cnf*, jos se on muotoa

$$F = C_1 \wedge C_2 \wedge \dots \wedge C_m,$$

missä kukin *tekijä (clause)* C_i on disjunktio

$$C_i = \alpha_{i1} \vee \alpha_{i2} \vee \dots \vee \alpha_{ir_i}.$$

Termit α_{ij} ovat *literaaleja*, so. muuttujia tai niiden negaatioita.

CSAT = $\{F \mid F \text{ on cnf - muotoinen toteutuva lausekalkyylin kaava}\}.$

Lause

Kieli CSAT on NP-täydellinen.

Todistus.

- CSAT kuuluu NP:hen, koska se on SAT:in erikoistapaus.
- Osoitetaan siten vain, että $A \in \text{NP} \implies A \leq_m^p \text{CSAT}$.
- Tutkimalla Cookin ja Levinin lauseen todistusta nähdään, että siinä muodostettavat kaavat

$$F_x = G_1 \wedge G_2 \wedge G_3 \wedge G_4 \wedge G_5 \wedge (G'_6 \wedge G''_6 \wedge G'''_6)$$

ovat melkein cnf-muotoisia, kun

- ▶ lyhennysmerkinnät $\phi \longrightarrow \psi$ kirjoitetaan auki muotoon $\neg\phi \vee \psi$, ja
- ▶ tarvittaessa sovelletaan de Morganin sääntöä $\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi)$.

- Ainoan poikkeuksen muodostavat alikaavat G_6'' ja G_6''' .
- G_6''' :

$$G_6''' = \bigwedge_t \left[(q \wedge h \wedge s) \longrightarrow \bigvee_{1 \leq i \leq k} (q' \wedge h' \wedge s') \right] \quad \text{eli}$$

$$G_6''' = \bigwedge_t \left[\neg q \vee \neg h \vee \neg s \vee \bigvee_{1 \leq i \leq k} (q' \wedge h' \wedge s') \right],$$

missä k on n :stä riippumaton vakio.

- Lausekalkyylin osittelulakien nojalla saadaan tästä myös kaavalle G_6''' konjunkttiivinen normaalimuoto:

$$G_6''' = \bigwedge_t \bigwedge_{\alpha \in \{q,h,s\}^k} \left[\neg q \vee \neg h \vee \neg s \vee \bigvee_{1 \leq i \leq k} \alpha(i)_{ti} \right].$$

- Tämä normaalimuoto on tosin noin 3^k kertaa alkuperäisen kaavan kokoinen, mutta koska k on vakio, kasvu ei haittaa.
- \square

3SAT:n täydellisyys I

Lause

3SAT on NP-täydellinen.

Todistus:

- Selvästi 3SAT on NP:ssä.
- Riittää osoittaa, että $CSAT \leq_p 3SAT$.
- Olkoon $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$, missä
- $C_k = \alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_r$, $r \geq 3$.
- Korvataan C_k :t 3-cnf -muotoisella kaavalla

$$C'_k = (\alpha_1 \vee \alpha_2 \vee t_1) \wedge (\neg t_1 \vee \alpha_3 \vee t_2) \wedge (\neg t_2 \vee \alpha_4 \vee t_3) \wedge \dots \wedge (\neg t_{r-3} \vee \alpha_{r-1} \vee \alpha_r),$$

missä t_1, t_2, \dots, t_{r-3} ovat uusia muuttujia.

3SAT:n täydellisyys II

- Kukin kaava C'_k voidaan selvästi muodostaa vastaavasta kaavasta C_k polynomisessa ajassa.
- Tarkastetaan vielä, että muunnos $F \mapsto F'$ säilyttää kaavojen toteutuvuuden ja toteutumattomuuden, so. että muunnos täyttää palautusehdon $F \in \text{CSAT}$ jos ja vain jos $F' \in \text{3SAT}$:
- **Jos F toteutuva, niin F' on myös toteutuva:** Tarkastellaan jotain kaavan F tekijää C_k ja vastaavaa F' :n tekijää C'_k .
- Minkä tahansa F :n toteuttavan totuusarvoasetuksen täytyy asettaa jonkin C_k :ssa esiintyvän literaalin α_i arvoksi 1.
- Tästä saadaan C'_k :n toteuttava totuusarvoasetus asettamalla literaalien α_i arvot samoin, ja uusien muuttujien arvot seuraavasti:

$$t_j = \begin{cases} 1, & \text{jos } j \leq i - 2; \\ 0, & \text{jos } j > i - 2. \end{cases}$$

3SAT:n täydellisyys III

- Jos F' toteutuva, niin myös F on toteutuva: Mikä tahansa kaavan F' toteuttava totuusarvoasetus toteuttaa erityisesti kutakin F :n tekijää C_k vastaavan alikaavan C'_k .
- Tällöin jokin alikaavassa esiintyvistä literaaleista $\alpha_1, \dots, \alpha_r$ saa arvon 1 ja tekijä C_k toteutuu. Jos nimittäin jokainen α on epätosi, ei C_k :sta saada totta, sillä t :t on valittu siten, että aina joudutaan ristiriitaan.
- Edellä on tarkasteltu vain vähintään neljä literaalia sisältävien tekijöiden muuntamista 3-cnf -muotoon. Yksinkertaisemmat tekijät käsitellään seuraavasti:

$$C_k = \alpha_1 \vee \alpha_2 \vee \alpha_3 \implies C'_k = C_k,$$

$$C_k = \alpha_1 \vee \alpha_2 \implies C'_k = (\alpha_1 \vee \alpha_2 \vee t) \wedge (\alpha_1 \vee \alpha_2 \vee \neg t),$$

$$C_k = \alpha \implies C'_k = (\alpha \vee t_1 \vee t_2) \wedge (\alpha \vee t_1 \vee \neg t_2) \wedge (\alpha \vee \neg t_1 \vee \neg t_2).$$

3SAT:n täydellisyys IV

- selvästi myös nämä muunnokset säilyttävät kaavojen toteutuvuusominaisuudet. \square

Lause

Klikki-ongelma on NP-täydellinen.

- On jo todistettu, että 3SAT voidaan palauttaa polynomisesti klikki-ongelmaan.
- Samoin on jo näytetty, että klikki-ongelma kuuluu NP:hen.
- Siten lause on tosi. \square

Solmupeiteongelma I

Määritelmä

- G suuntaamaton verkko.
- G :n solmupeite on sellainen solmujoukon osajoukko, että jokainen särmä koskettaa jotain solmupeitteen solmua.
- Solmupeiteongelmassa kysytään, sisältääkö verkko tietyn kokoisen solmupeitteen.

Solmupeiteongelma II

Lause

Solmupeiteongelma on NP-täydellinen.

Todistus.

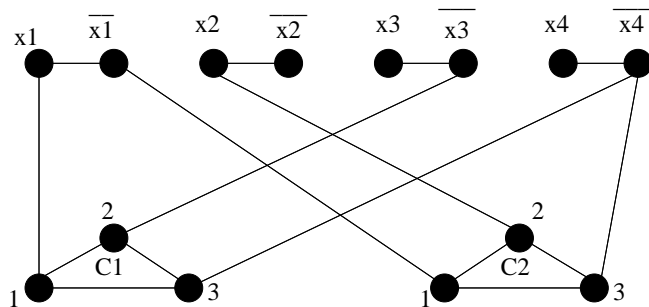
- Pitää jälleen osoittaa, että ongelma kuuluu NP:hen ja että jokainen NP:n ongelma voidaan palauttaa solmupeiteongelmaan.
- NP:hen kuuluminen on helppo osoittaa. Todistus eli sertifikaatti on yksinkertaisesti k :n kokoinen solmupeite. Sen testaaminen sujuu helposti polynomisessa ajassa.
- Palautetaan seuraavaksi 3SAT solupeiteongelmaan. Tämä vaatii hieman kekseliäisyyttä.

Solmupeiteongelma III

Seuraava kuva näyttää, miten kaavaa

$$\phi = (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_4)$$

vastaava verkko G ja solmupeitteen koko k konstruoidaan:



Lisäksi $k = 8$.

Solmupeiteongelma IV

Konstruktion kuvaus:

- Olkoon

$$F = C_1 \wedge \dots \wedge C_m$$

3-cnf -muotoinen lausekalkyylin kaava, jossa esiintyvät muuttujat x_1, \dots, x_n .
Muodostetaan verkko G .

- G :ssä on solmu kullekin literaalille x_i ja \bar{x}_i , missä $i = 1, \dots, n$, sekä kutakin F :n tekijää C_j kohden kolme solmua C_j^1 , C_j^2 ja C_j^3 , missä $j = 1, \dots, m$.

- Verkossa on seuraavat särmät:

- ▶ (x_i, \bar{x}_i) , $i = 1, \dots, n$;
- ▶ (C_j^1, C_j^2) , (C_j^2, C_j^3) , (C_j^3, C_j^1) , $j = 1, \dots, m$,
- ▶ jos $C_j = (\alpha_1 \vee \alpha_2 \vee \alpha_3)$, niin
 (C_j^1, α_1) , (C_j^2, α_2) , (C_j^3, α_3) , $j = 1, \dots, m$.

- Arvoksi k valitaan luku $n + 2m$.

Solmupeiteongelma V

Verkko G voidaan selvästi muodostaa kaavasta F polynomisessa ajassa. Osoitetaan, että G :llä on enintään k solmun solmupeite, jos ja vain jos F on toteutuva.

- Olkoon $t : \{x_1, \dots, x_n\} \rightarrow \{0, 1\}$ jokin kaavan F toteuttava totuusarvoasetus.
- Vastaavaan verkon G solmupeitteeseen V' otetaan ensin kutakin literaaliparia x_i, \bar{x}_i kohden se solmu, jota vastaava literaali saa arvon 1.
- Tämän jälkeen on kustakin C_j -kolmiosta ainakin yhdestä kulmasta alkava kaari C_j^f, α_r jo peitetty, ja peitteeseen lisätään kolmion kaksi muuta kulmasolmua.
- Näin saatu solmujoukko V' selvästi peittää kaikki G :n kaaret ja $|V'| = n + 2m = k$.

Solmupeiteongelma VI

Olkoon toisaalta V' jokin G :n solmupeite, jolla $|V'| \leq k$.

- Koska V' :n on sisällettävä vähintään yksi solmu kustakin literaaliparista x_i , \bar{x}_i , ja vähintään kaksi solmua kustakin C_j -kolmiosta, on oltava myös $|V'| \geq n + 2m = k$.
- Siten $|V'| = k$, ja V' sisältää täsmälleen yhden solmun kustakin literaaliparista ja täsmälleen kaksi solmua kustakin C_j -kolmiosta.
- Saadaan muuttujien totuusarvoasetus

$$t(x_i) = \begin{cases} 1 & \text{jos } x_i \in V'; \\ 0 & \text{jos } \bar{x}_i \in V'. \end{cases}$$

Solmupeiteongelma VII

- Nyt kunkin kolmion kärjistä alkavista kaarista vain kaksi voi olla kolmiosta peitteeseen valittujen solmujen peittämiä; kolmannen peittää jokin literaalisolmu $\alpha \in V'$.
- Mutta tällöin $t(\alpha) = 1$, ja totuusarvoasetus t toteuttaa tekijän C_j . \square

Hamiltonin kehä -ongelma

Määritelmä

Olkoon G suunnattu verkko. Solmuja s ja t yhdistävä G :n polku on **Hamiltonin kehä**, jos polku kulkee kaikkien G :n solmujen kautta täsmälleen kerran.

Lause

Hamiltonin kehä -ongelma on NP-täydellinen.

Hamiltonin kehän NP-täydellisyystodistus

- Aikaisemmin on jo näytetty, että Hamiltonin kehä kuuluu NP:hen.
- Jää näytettäväksi, että jokainen NP:n kieli voidaan palauttaa Hamiltonin kehään.
- Tämä osoitetaan konstruoimalla palautus

$$3\text{SAT} \leq_p \text{HAMPATH.}$$

- Lähtökohtana siis on lausekalkyylin kaava muotoa

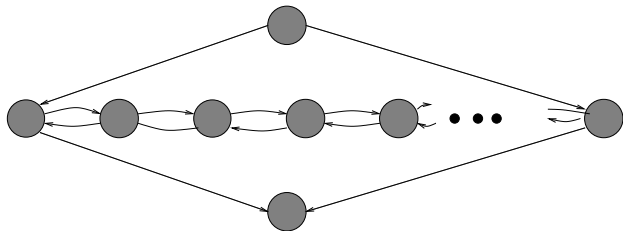
$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k),$$


missä jokainen a , b ja c on literaali x_i tai \bar{x}_i .

- Olkoot x_1, \dots, x_m kaavan ϕ m muuttujaa.
- Pitää konstruoida verkko G , jossa on Hamiltonin kehä silloin ja vain silloin, kun kaava on toteutuva.

Hamiltonin kehän NP-täydellisyystodistus II

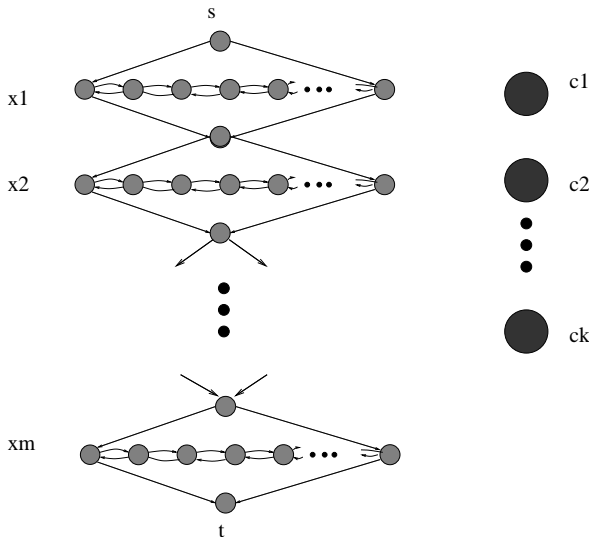
- Asetetaan jokaista muuttujaa x_i vastaamaan timantin muotoinen solmujoukko:



- Timantin keskellä olevien solmujen lukumäärä määrätään myöhemmin.
- Jokaista ϕ :n tekijää (clause) vastaa yksi solmu, 

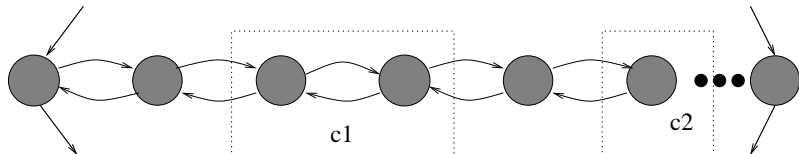
Hamiltonin kehän NP-täydellisyytodistus III

- Seuraavassa kuviossa nähdään verkon globaali rakenne:



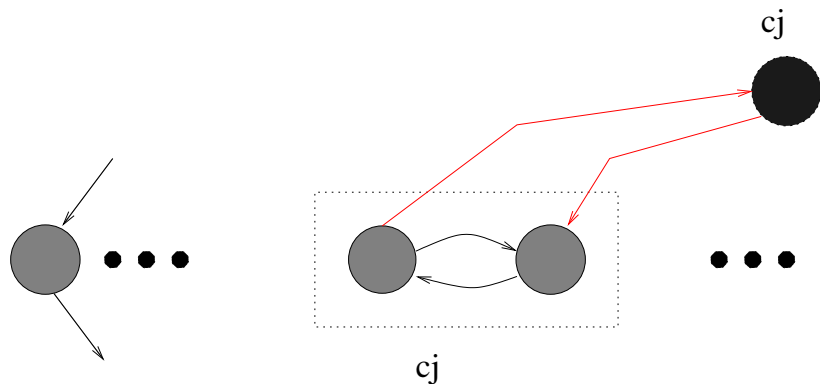
Hamiltonin kehän NP-täydellisyystodistus IV

- Kuviossa ei ole vielä piirretty muuttujien ja tekijöiden välisiä kaaria.
- Jokaisessa muuttujaa vastaavassa timantissa on keskellä solmujen ketju, joka on kaksisuuntainen.
- Ketju sisältää $3k + 1$ solmua loppusolmujen lisäksi. Nämä $3k + 1$ solmua ryhmitellään pareiksi siten, että kukin pari vastaa yhtä tekijää ja parien välissä on yksi solmu. Seuraava kuvio valaisee asiaa:



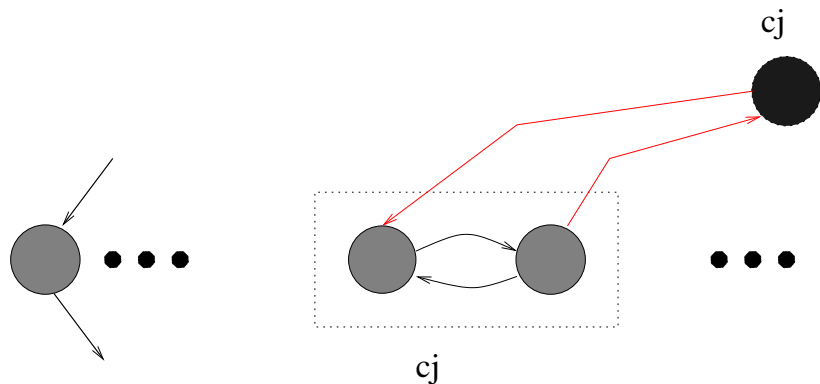
Hamiltonin kehän NP-täydellisyystodistus V

- Jos muuttuja x_i esiintyy tekijässä c_j , niin lisätään seuraavat kaaret i. timantin j.parista j. tekijän solmuun:



Hamiltonin kehän NP-täydellisyystodistus VI

- Jos \bar{x}_i esiintyy tekijässä c_j , lisätään samat kaaret kuin edellä, mutta kaarien suunta on päinvastainen.



Hamiltonin kehän NP-täydellisyystodistus VII

- Pitää osoittaa, että verkossa on Hamiltonin kehä s :stä t :hen, jos ja vain jos kaava on toteutuva. Oletetaan ensiksi, että kaava on toteutuva.
- onstruoidaan ensin polku, joka kulkee kaikkien solmujen paitsi solmujen c_j kautta. Polku lähtee s :stä ja kulkee jokaisen timantin läpi seuraavasti.
- Ensin tullaan timantin yläsolmuun. Vastatkoon timantti muuttujaa x_i .
- Jos x_i :n arvo on tosi, siirrytään vasemmalle ja jatketaan ketjun läpi oikealle ja lopuksi alas.
- Jos x_i :n arvo on epätosi, siirrytään oikealle ja jatketaan timantin ketjun läpi vasemmalle ja alas.

Hamiltonin kehän NP-täydellisyytodistus VIII

- Seuraavaksi täytyy ratkaista, miten tekijäsolmut c_j otetaan huomioon.
- Käydään läpi jokainen tekijä c_j ja valitaan niistä yksi literaali, x_i tai \bar{x}_i , joka on sijoituksessa saanut arvon tosi.
- Jos on valittu x_i tekijässä c_j , voimme "koukata" c_j :n kautta, kun kuljetaan pitkin timantin i ketjua vasemmalta oikealle. Tämä onnistuu, sillä x_i on saanut arvon tosi ja polku kulkee timantissa vasemmalta oikealle ja koukkaus c_j :n kautta ei ole ristiriidassa tämän kulkusuunnan kanssa.
- Jos on valittu \bar{x}_i , voimme taas kulkea c_j :n kautta, sillä nyt polku timantin i läpi kulkee oikealta vasemmalle ja koukkaus sopii yhteen tämän suunnan kanssa.
- Jos tekijässä on useita tosia literaaleja, valitaan vain yksi.
- Näin saadaan aikaan Hamiltonin polku.

Hamiltonin kehän NP-täydellisyytodistus IX

- On vielä osoitettava, että mikäli verkossa on Hamiltonin polku, kaava on toteutuva.
- Jos polku on **normaali**, eli kulkee timantit järjestyksessä läpi yhdestä m :ään, saadaan totuuarvoasetus katsomalla, mihin suuntaan polku kulkee muuttujaa x_i vastaavassa timantissa. Jos se kulkee vasemmalta oikealle, otetaan muuttujalle x_i arvoksi tosi, muuten epätosi.
- On siten vain osoitettava, että jokainen Hamiltonin polku on normaali.
- Normaalisuus katoaa vain, jos polku tekijäsolmuun yhdestä timantista ja palaa takaisin toiseen timanttiin. Seuraava kuva valaiseen tilannetta:

Hamiltonin kehän NP-täydellisyystodistus XI

- Polku menee solmusta a_1 solmuun c , mutta a_2 :n sijasta palaa solmuun b_2 toisessa timantissa.
- Tällöin joko a_2 tai a_3 :n täytyy olla erottava solmu parien välissä.
- Jos a_2 on erottava solmu, ainoat solmuun a_2 tulevat kaaret tulevat joko a_1 :stä tai a_3 :sta.
- Jos a_3 on erottava solmu, a_1 ja a_2 ovat samassa parissa, jolloin ainoat a_2 :een tulevat kaaret ovat peräisin a_1 :stä, a_3 :sta tai c :stä.
- Kummassakaan tapauksessa polku ei voi sisältää a_2 :ta.

Hamiltonin kehän NP-täydellisyystodistus XII

- Nimittäin polku ei voi tulla a_2 :een c :stä tai a_1 :stä, koska polku menee näistä solmuista muualle.
- Polku ei saapua a_2 :een a_3 :sta, koska a_3 on ainoa solmu, johon a_2 :sta voi vielä siirtyä.
- Siten polun täytyy tulla a_2 :sta a_3 :een ja Hamiltonin polku on normaali.
- Lopuksi todetaan, että palautus toimii polynomisessa ajassa. \square

Osasummaprobleema

Lause

Osasummaprobleema SUBSET-SUM on NP-täydellinen.

Todistus. Tiedetään, että SUBSET-SUM kuuluu NP:hen. Näytetään, että 3SAT voidaan palauttaa polynomisessa ajassa SUBSET-SUM -ongelmaan.

- Olkoon ϕ Boolean kaava, jossa muuttujat x_1, \dots, x_n ja tekijät c_1, \dots, c_k .
- Muodostetaan taulukko, jossa on jokaista muuttujaa x_i kohti kaksi riviä, joita kutsutaan y_i -riviksi ja z_i -riviksi.
- Lisäksi jokaista tekijää c_i kohti on kaksi riviä, joita kutsutaan g_i -riviksi ja h_i -riviksi.
- Viimeisenä on vielä rivi, jota kutsutaan t -riviksi.
- Jokainen rivi koostuu biteistä 0 ja 1 ja rivi tulkitaan desimaaliluvuksi.
- Seuraavassa on kuva taulukosta.

	1	2	3	4	...	n	c_1	c_2	...	c_k
y_1	1	0	0	0	...	0	1	0	...	0
z_1	1	0	0	0	...	0	0	0	...	0
y_2		1	0	0	...	0	0	1	...	0
z_2		1	0	0	...	0	1	0	...	0
.				
.				
.				
y_n						1	0	0	...	0
z_n						1	0	0	...	0
g_1							1	0	...	0
h_1							1	0	...	0
.										.
.										.
.										.
g_k										1
h_k										1
t	1	1	1	1	...	1	3	3	...	3

- Siis lukua x_i vastaa desimaaliluku, jonka kokonaisosassa on ykkönen ja $n - i$ nollaa.
- Desimaaliosa eli oikeanpuoleinen osa sisältää yhden bitin jokaista termiä kohti. Tarkemmin sanottuna y_j :n j . bitti on yksi, jos termi c_j sisältää literaalin x_j . Vastaavasti z_j :n j . bitti on yksi, jos c_j sisältää literaalin \bar{x}_j .
- Jos bittiä ei ole määritetty yhdeksi, se on nolla.
- Taulukossa näkyy kaavaa

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \dots) \wedge \dots \wedge (\bar{x}_3 \vee \dots \vee \dots)$$

vastaavia arvoja.

- Lisäksi jokaista termiä c_j vastaa pari g_j, h_j . Nämä kaksi lukua ovat identtiset, alkavat ykkösellä, jota seuraa $k - j$ nollaa.
- t :n arvo näkyy taulukosta; se on aina saman muotoinen.

- Osoitetaan, että konstruktio toimii. Oletetaan ensin, että ϕ on toteutuva.
- Muodostetaan luvuista y_i ja z_i sellainen joukko S , että joukon lukujen summa on t .
- Valitaan y_i , jos x_i :n arvo on tosi totuusarvoasetuksessa, ja z_i , jos \bar{x}_i on tosi.
- Huomataan, että näillä valinnoilla lukujen summan kokonaisosa on yksi, sillä aina valitaan joko y_i tai z_i kultakin riviltä.
- Toisaalta desimaaliosan luvut ovat väliltä 1 ja 3, sillä jokainen termi on toteutuva ja käsittää korkeintaan kolme eri literaalia.
- Täydennetään S :ää tarvittaessa ottamalla tarpeellinen määrä g_i ja h_i lukuja.
- Näin siis saadaan aikaan osasummaprobleema (S, t) , joka on ratkeava.

- Oletetaan nyt, että S :llä on osajoukko, jonka alkioiden summa on t .
- Konstruoidaan ϕ :n totuusarvoasetus.
- Ensinnäkin kaikki numerot luvuissa ovat joko nollia tai ykkösiä.
- Jokainen sarake käsittää korkeintaan viisi ykköstä.
- Siten sarakkeella ei milloinkaan tapahdu yhteenlaskun ylivuotoa muistinumeroineen.
- Jotta summaksi kokonaisuudessa tulisi 1, joko y_i :n tai z_i :n täytyy olla 1, muttei molempien.
- Nyt voidaan konstruoida totuusarvoasetus.

- Jos osajoukko sisältää y_i :n, asetetaan x_i todeksi, muuten epätodeksi.
- Tämän totuusarvoasetuksen täytyy tehdä ϕ todeksi, sillä desimaaliosassa jokainen luku on 3.
- Tästä kolmosesta vain kaksi tulee g_i :stä ja h_i :stä, joten viimeisen yhden täytyy tulla y_i :stä tai z_i :stä.
- Jos tämä on y_i , niin x_i esiintyy c_j :ssä ja omaa arvon tosi, joten c_j on tosi. Jos tämä on z_i , niin \bar{x}_i esiintyy c_j :ssä ja x_i on epätosi, joten c_j on tosi.
- Siten ϕ on toteutuva.
- Taulukon koko on noin $(n + k)^2$ ja jokainen paikka taulukossa on helposti laskettavissa kaavasta ϕ . Siten kokonaisaika on neliöllinen kaavan kokoon nähden. \square