NP-täydellisyys

Joonas Järvenpää ja Topi Talvitie

Laskennan teorian opintopiiri
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 23. helmikuuta 2014

Sisältö

1 Johdanto

Polynomisessa aikavaativuushierarkiassa luokkaan P kuuluvat ”yksinkertaiset” ongelmat, joille on aina löydettävissä ratkaisu polynomisessa ajassa. Toisaalta taas NP on niiden ongelmien joukko, joiden ratkaisu pystytään tarkastamaan polynomisessa ajassa. Luokasta NP voidaan myös määritellä erityinen NP-täydellisten ongelmien osajoukko, jotka ovat NP-luokan vaikeimpia ongelmia.

P- ja NP-luokkien välinen suhde on hyvin oleellinen mietittäessä, missä tapauksissa ongelmille ei ole olemassa helppoja ratkaisualgoritmeja. Toinen tässä tutkielmassa tarkasteltava teema on, että onko ratkaiseminen välttämättä vaikeampaa kuin ratkaisun verifiointi.

2 Ongelman määrittely

Määrittelemme teorian ainoastaan *päätösongelmille*, eli kysymyksille, joissa vastauksena on ”kyllä” tai ”ei”. Käytännön ongelmille tarvitaan usein pidempi tuloste, mutta ne voidaan kysellä polynomimäärällä kyllä/ei-kysymyksiä.

Esimerkiksi ongelma

Mikä on lyhin polku verkon solmusta a solmuun b?

ei ole päätösongelma, mutta

Onko lyhimmän polkun a:sta b:hen n:s solmu v?

on päätosongelma. Ratkaisu ensimmäiseen ongelmaan saadaan käymällä tapaukset $n = 1, 2, 3, \ldots$ ja käymällä aina läpi mahdolliset solmut v, kunnes löydetään viimeinen solmu b. Tällä tavalla kutsuja päätösongelmaan tulee tehtyä $O(n^2)$ kappaletta, eli mikäli jälkimmäinen ongelma voidaan ratkoa polynomisessa ajassa, ratkeaa myös ensimmäinen polynomisessa ajassa.

Toinen, yleisempi tapa yleisen ongelman palauttamiseksi päätösongelmaan on päättää tavasta koodata ongelman vastaus binäärijonona, joka sisältää tiedon binäärijonon pituudesta, ja muodostaa päätösongelma

Onko vastauksen n:s bitti 1?

Nyt voidaan helposti selvittää vastaus käymällä läpi bitit $n = 1, 2, 3, \ldots$ kunnes vastaus on luettu.

2.1 Vaativuusluokka P

Nyt kun päätösongelmat on määritelty tarkasti, voidaan määritellä niiden vaativuusluokka P. Päätösongelma kuuluu vaativuusluokkaan P mikäli on olemassa Turingin kone, joka pysähtyy kaikilla syötteillä polynomisessa ajassa syötteen koon suhteen, ja hyväksyy syötteen täsmälleen silloin kun vastaus ongelmaan sillä syötteellä on ”kyllä” [6] [4].

Koska kaikkien käytännöllisten laskennan mallien simuloiminen toisillaan onnistuu polynomiaikaisilla operaatioilla, tämä tarkoittaa käytännössä sitä että P on niiden päätösongelmien luokka, joihin vastaamiseen on polynomiaikainen algoritmi.

2.2 Vaativuusluokka NP

Vaikka päätösongelman vastauksen löytäminen ei onnistuisi monissa tapauksissa, on aina ”kyllä”-tapauksissa olemassa jokin ”kyllä”-tapaukseksi varmentava todistemerkkijono eli ”sertifikaatti”, jotka voidaan polynomiaikaisella algoritmilla tunnistaa. Määrittelemme tällaiset päätösongelmat NP-vaativuusluokaksi.

Eräs esimerkki tällaisesta ongelmasta on selvittää, onko annetussa verkossa Hamiltonin polku [6]. Hamiltonin polku on verkon polku, joka käy kaikissa verkon solmuissa täsmälleen kerran. Hamiltonin polun olemassaolon ratkaisemiseen ei tunneta polynomiaikaista algoritmia, mutta ongelma on NP-ongelma. Siis jos Hamiltonin polku löytyy, vastaus on ”kyllä” ja kaikki verkon Hamiltonin polut voidaan tunnistaa polynomiaikaisella algoritmilla: algoritmin täytyy ainoastaan katsoa, että annettu polku on verkon polku ja se käy kaikissa solmuissa täsmälleen kerran.

NP-vaativuusluokka voidaan myös määritellä tarkemmin niiden ongelmien joukkona, joille on olemassa polynomiajassa pysähtyvä deterministinen Turingin kone M ja kiinteät luvut $p, q > 0$, että vastaus syötteeseen x on ”kyllä” jos ja vain jos kone M hyväksyy jonkin parin (x, y), missä $|y| \leq p|x|^q$. Tässä y vastaa edellisen määritelmän ”sertifikaattia”.

Yhtäpitävä määritelmä NP-vaativuusluokalle on epädeterministisellä Turingin koneella polynomiajassa ratkeavat ongelmat. NP-ongelmat ratkeavat epädeterministisellä Turingin koneella polynomiajassa, sillä kone voi ensin epädeterministisesti arvata sertifikaattiehdokkaan ja sitten ajaa deterministisen sertifikaatit tunnistavan Turingin koneen. Toisaalta epädeterministisellä

Turingin koneella polynomiajassa ratkeavat ongelmat ovat NP-ongelmia, koska sertifikaatteihin voidaan koodata koneen tekemät epädeterministiset valinnat.

Helposti nähdään, että kaikki P-ongelmat ovat myös NP-ongelmia, koska sertifikaatit tunnistava algoritmi voi jättää sertifikaatin huomiotta.

3 Reduktiot

Ongelmien vaativuutta tutkiessa on hyödyllistä, jos pystyy osoittamaan ongelman A olevan redusoitavissa ongelmaan B. Mikäli itse reduktio ei vie liikaa aikaa, A on tällöin enintään yhtä vaikea kuin B.

Polynomiaikaisella reduktiolla (polynomial-time many-one reduction) muunnetaan ongelman A syöte x polynomiaikaisella algoritmilla ongelman B:n syötteeksi x' siten, että ongelmien ratkaisijoiden tulokset ovat samat [2]. Tällaisen reduktion olemassaoloa merkitään

$$A \leq_m^P B.$$

Muunnosta kutsutaan myös polynomiseksi transformaatioksi tai Karp-reduktioksi.

Reduktion merkintä on helppo muistaa siitä, että jos $A \leq_m^P B$ niin intuitiivisesti

$$A\text{:n vaikeus} \leq B\text{:n vaikeus},$$

mikäli polynomiaikainen reduktiovaihe katsotaan merkityksettömäksi. Reduktiorelaatiolle pätee myös transitiivisuusominaisuus, eli jos $A \leq_m^P B \leq_m^P C$ niin $A \leq_m^P C$, sillä A voidaan redusoida C:ksi redusoimalla se ensin B:ksi ja sitten C:ksi.

3.1 NP-täydellisyys

NP-ongelmaa kutsutaan NP-täydelliseksi, mikäli mikä tahansa NP-ongelma voidaan redusoida polynomiajassa siihen.

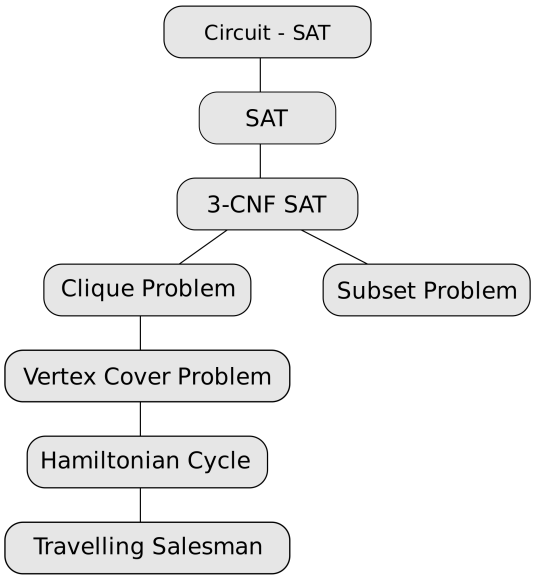
$$\text{NP-Complete} = \{B \in \text{NP} \mid \text{kaikilla } A \in \text{NP} \text{ pätee } A \leq_m^P B\}.$$

Intuitiivisesti tämä tarkoittaa, että NP-täydelliset ongelmat ovat NP-ongelmien vaikeimmat ongelmat.

Olisi sinänsä mahdollista, että näin määritelty joukko NP-Complete olisi tyhjä. Kuitenkin voidaan todistaa, että CIRCUIT-SAT-ongelma on NP-täydellinen. CIRCUIT-SAT kysyy annetusta loogisten porttien muodostamasta piireistä, voidaanko syötteet valita siten että ulostulo on tosi. Todistuksessa muunnetaan NP-ongelman sertifikaatin tunnistava kone loogiseksi piiriksi, ja CIRCUIT-SATin avulla selvitetään löytyykö jokin sertifikaatti jonka kone hyväksyy, jolloin siis jokainen NP-ongelma redusoituu CIRCUIT-SATiin. Todistus esitetään tarkemmin lähteessä [4].

Aina kun tunnetaan jokin NP-täydellinen ongelma B, niin kaikki NP-ongelmat C, joihin B redusoituu polynomiajassa, ovat myös NP-täydellisiä, koska NP-täydellisyyden määritelmän nojalla kaikilla $A \in \text{NP}$ pätee $A \leq_m^P B$ ja koska $B \leq_m^P C$, niin transitiivisuuden nojalla kaikilla $A \in \text{NP}$ pätee $A \leq_m^P C$, eli C on NP-täydellinen.

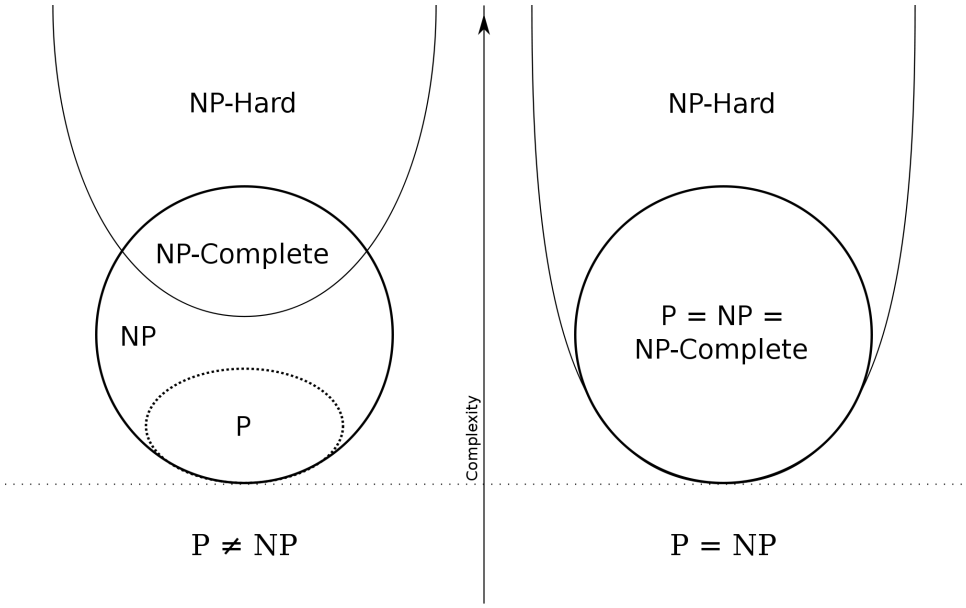
Mitä enemmän NP-täydellisiä ongelmia löydetään, sitä helpompi on löytää reduktiolla uusia. Näin NP-täydellisiksi todistetut ongelmat muodostavat puumuotoisen ”reduktiokaavion”, jossa on aina kaari jokaista todistettua redusoituvuutta kohden, ja juurena on ensimmäinen NP-täydelliseksi todistettu ongelma, esimerkiksi CIRCUIT-SAT.



Kuva 1: Eräs NP-täydellisyysluokan reduktiokaavio [1].

Määritelmän nojalla kaikki NP-täydelliset ongelmat redusoituvat toisik-

seen. Siis joko kaikki NP-Complete-ongelmat ovat P-ongelmia tai yksikään ei ole. Erityisesti jos yksikin NP-Complete-ongelma ratkeaa polynomiajassa, niin P = NP.



Kuva 2: P- ja NP-luokkien koostumus kahdessa eri tapauksessa [1].

3.2 NP-kovat ongelmat

NP-kovat ongelmat ovat yleisesti ne ongelmat, joihin voidaan polynomiajassa redusoida kaikki NP-ongelmat, mutta NP-kovien ongelmien ei tarvitse itse kuulua NP:hen. Kaikki NP-täydelliset ongelmat kuuluvat siis myös NP-kovien ongelmien joukkoon. NP-kovat ongelmat ovat siis *vähintään yhtä vaikeita* kuin mikä tahansa NP-ongelma. On myös olemassa sellaisia NP-kovia ongelmia, jotka eivät kuulu NP-täydellisten joukkoon, kuten esimerkiksi Turingin koneen pysähtymisongelma.

4 P vs. NP -ongelman pohdintaa

Emme kuitenkaan tiedä, ovatko P ja NP samoja luokkia. Kukaan ei ole myöskään osoittanut, etteikö tätä pystyisi todistamaan jollain aksiomaattisella järjestelmällä. Aaronson on päätynyt pohdinnassaan arvioon, että P $\neq$ NP täytyisi olla joko totta tai epätotta, eli vastaus ei voi olla määrittelemätön,

mutta todistuksen löytäminen on hyvin epävarmaa [3].

P vs. NP onkin selkeästi merkittävin avoin kysymys laskennan teoriassa ja koko matematiikassa [5]. Kysymys voidaan kiteyttää muotoon ”Onko vaikeampaa löytää ratkaisuja kuin tarkistaa niitä?” Intuition mukaan väite voisi pitää paikkansa, mutta kysymyksen ratkaiseminen formaalisti olisi jo filosofiselta kannalta merkittävää. Tapauksessa P = NP koko polynominen vaativuushierarkia romahtaisi yhteen luokkaan, ja tällöin olisi esimerkiksi mahdollista rakentaa automaattinen teoreemantodistusohjelma, mikä pakottaisi miettimään uudelleen luovuuden merkitystä matematiikassa.

Tähän asti aiheesta on lähinnä saavutettu negatiivisia tuloksia, eli että asiaa ei voi todistaa joidenkin tiettyjen menetelmien puitteissa. Näköpiirissä ei ole juurikaan realistisia menetelmiä kysymyksen ratkaisemiseen [3].

5 NP-täydellisyyden todistaminen

Ongelman todistaminen NP-täydelliseksi on vahva todiste siihen suuntaan, että ongelma ei ratkea polynomiajassa, koska todennäköisimmin P $\neq$ NP [4].

Tietorakenteet-kurssilta tutuista aikavaativuuksien ylärajatodistuksista poiketen NP-täydellisyyden todistaminen etenee eri tavalla, koska tarkoituksena on osoittaa, *kuinka vaikea ongelma on vähintään*, eikä osoittaa sen helppoutta pahimmassa tapauksessa.

5.1 Esimerkkitodistus

Näytetään esimerkkinä todistus sille, että annetun kokoisen klikin etsiminen on NP-täydellinen ongelma olettaen, että 3-SAT on NP-täydellinen. Siis täytyy todistaa, että 3-SAT voidaan redusoida annetun kokoisen klikin olemassaoloon, ja että klikin olemassaolo on NP-ongelma.

3-SAT eli 3-CNF-SAT-ongelmassa selvitetään, onko olemassa totuusarvot $x, y, z, w, \ldots$ siten, että kaikki annetut kolmesta symbolista koostuvat disjunktiolausekkeet toteutuvat. Esimerkiksi

$$\begin{aligned} &(x \vee y \vee z) \\ \wedge\ &(\neg x \vee y \vee z) \\ \wedge\ &(\neg y \vee \neg z \vee x) \end{aligned}$$

Verkon klikillä tarkoitetaan sellaista solmujoukkoa, että jokaisen solmu-

parin välillä on kaari. Todistamme nyt, että sen selvittäminen, että onko verkossa klikkiä jossa on k solmua, on NP-täydellinen ongelma. Selvästi se on ainakin NP-ongelma, koska klikin tunnistaminen onnistuu käymällä kaikki solmuparit läpi.

Olkoon annettu 3-SAT-ongelma jossa on muuttujat ovat $x_1, x_2, \ldots, x_n$, ja lausekkeita on k kappaletta.

Muodostetaan verkko, jossa on k-klikki täsmälleen silloin, kun muuttujat $x_1, x_2, \ldots, x_n$ voidaan asettaa sillä tavoin, että kaikki lausekkeet toteutuvat. Ideana on luoda jokaista lauseketta kohti kolme solmua, yksi jokaista disjunktion tekijää kohden. Verkon kaaret valitaan siten, että jokaisen lausekkeen solmuista voidaan valita vain yksi, ja ei voida valita kahta ristiriidassa toistensa kanssa olevaa tekijää, esimerkiksi sekä x_5 että $\neg x_5$. Tällöin k-klikin olemassaolo tarkoittaa että on ristiriidaton valinta muuttujien arvoiksi siten, että jokainen lauseke pätee.

Kaariksi valitaan siis kaikki, paitsi kahden saman lausekkeen solmun väliset (jotta ei voitaisi valita montaa solmua samasta lauseesta), ja ristiriidassa olevia lauseita vastaavien solmujen väliset. Tämä muunnos voidaan helposti tehdä polynomiajassa, eli klikin olemassaolo on NP-täydellinen ongelma.

6 Joitain esimerkkiongelmia

6.1 P

- Kokonaislukutaulukon järjestäminen
- Lyhyimmän polun löytäminen verkossa
- Lineaarinen ohjelmointi
- Alkuluvun tunnistus

6.2 NP

- Verkkojen isomorfismi
- Kokonaisluvun tekijöihin jakaminen
- (Kaikki NP-Complete- ja P-ongelmat)

6.3 NP-täydellisyys

- Hamiltonin polku / Pisin polku verkossa
- Maximum clique

- Circuit SAT
- Subset sum

7 Yhteenveto

Tekstissä tutustuimme aluksi polynomiaikaisesti ratkeaviin ”helppoihin” ongelmiin, jotka kuuluvat luokkaan P.

Kuitenkin löytyy monia ongelmia, joiden ratkaiseminen näyttäisi olevan polynomiaikaista vaikeampaa, mutta tarkastaminen yhtä helppoa kuin luokassa P. Sanomme näiden ongelmien kuuluvan luokkaan NP. Luokalla NP on olemassa myös erityinen NP-täydellisten ongelmien osajoukko, jonka ominaisuutena on, että jos yhdelläkin on ”helppo” ratkaisualgoritmi, tällöin kaikilla muillakin NP-ongelmilla on sellainen. Voidaan myös sanoa, että NP-täydelliset ongelmat ovat *vähintään yhtä vaikeita* kuin mikä tahansa ongelma NP:ssä.

Jos tunnetaan yksi NP-täydellinen ongelma, reduktioiden avulla voidaan pyrkiä osoittaa jonkin toisen ongelman olevan vähintään yhtä vaikea kuin NP-täydelliset ongelmat yleensä. Näin voidaan saada vahva todiste kyseisen ongelman vaikeudesta.

Lähteet

[1] *Wikipedia: NP-complete.* `http://en.wikipedia.org/w/index.php?title=NP-complete&oldid=596284955`.

[2] *Wikipedia: Polynomial-time reduction.* `http://en.wikipedia.org/w/index.php?title=Polynomial-time_reduction&oldid=594462975`.

[3] Aaronson, Scott: *Is P versus NP formally independent?* Bulletin of the EATCS, 81:109–136, 2003.

[4] Cormen, T.H., Leiserson, C.E., Rivest, R.L. ja Stein, C.: *Introduction To Algorithms.* MIT Press, 2001, ISBN 9780262032933.

[5] Moore, C. ja Mertens, S.: *The Nature of Computation.* Oxford University Press, 2011, ISBN 9780199233212.

[6] Sipser, M.: *Introduction To The Theory Of Computation.* Advanced Topics Series. Thomson Course Technology, 2006, ISBN 9780534950972.