

## 6. Approksimointialgoritmit

Tässä luvussa käsitellään lyhyesti approksimointiin liittyvät peruskäsitteet ja joitain keskeisiä approksimoituvuustuloksia.

Tavoitteena on, että opiskelija

- näkee approksimointialgoritmien merkityksen NP-kovien approksimointialgoritmien ratkaisemisessa,
- oppii soveltamaan yleisiä approksimointitarkkuuteen liittyviä käsitteitä,
- tutustuu perusteellisesti joihinkin keskeisiin approksimointialgoritmeihin ja
- saa jonkinlaisen käsityksen approksimointilähestymistavan rajoituksista.

Aihepiiriin voi perehtyä tarkemmin syksyllä 2004 pidettävällä erikoiskurssilla.

## 6.1 Peruskäsitteet

Kurssilta *Laskennan teoria* tiedetään, että

- monet keskeiset ongelmat ovat NP-täydellisiä ja
- nykytietämyksen valossa NP-täydellisiä ongelmia ei osata ratkaista polynomisessa ajassa.

Käytännössä pitää siis tinkiä jostain:

1. käytetään ei-polynomisia algoritmeja; tämä toimii jos ongelmat ovat riittävän pieniä (esim. TSP ja branch-and-bound) tai pahimman tapauksen syötteet ovat harvinaisia (lineaarinen ohjelmointi ja simplex),
2. rajoitutaan sopiviin erikoistapauksiin (esim. SAT-ongelman erikoistapaus 2-SAT, joka kuuluu luokkaan P),
3. käytetään [satunnaisalgoritmeja](#), jotka pienellä todennäköisyydellä antavat väärän vastauksen (NP-kovilla ongelmilla luultavasti toimii vain yhdistettynä approksimointiin) tai
4. käytetään [approksimointialgoritmeja](#), jotka voivat antaa "lievästi virheellisiä" vastauksia.

Johdattelevana esimerkkinä tarkastellaan solmupeiteongelmaa.

Varsinainen solmupeiteongelma (Vertex Cover, VC) on seuraava päätösongelma:

**Annettu:** suuntaamaton verkko  $G = (V, E)$ ,  
luonnollinen luku  $k$

**Kysymys:** onko verkossa  $G$  kokoa  $k$  oleva solmupeite, s.o. joukko  $C \subseteq V$  jolla  $|C| = k$  ja kaikilla  $(u, v) \in E$  pätee  $u \in C$  tai  $v \in C$ .

Solmupeiteongelma on tunnetusti NP-täydellinen.

Vastaava optimointiongelma **VC-opt** on seuraava:

**Annettu:** suuntaamaton verkko  $G$

**Määrättävä:** verkon  $G$  pienimmän solmupeitteen koko.

Jos VC-opt osattaisiin ratkaista polynomisessa ajassa, niin sama pätesi ilmeisesti myös ongelmalle VC, jolloin siis olisi  $P = NP$ . Sanomme siis (hieman epämuodollisesti), että VC-opt on **NP-kova** ongelma.

Yleensä se mistä varsinaisesti ollaan kiinnostuneita on asiaan liittyvä **etsintäongelma**. Siis esim. solmupeitteen tapauksessa halutaan itse pienin solmupeite, ei pelkästään sen kokoa. Tätäkin sanotaan NP-kovaksi, kun päätösongelma on NP-täydellinen.

Tarkastelemme seuraavaa ahnetta heuristiikkaa solmupeitteen minimointiongelmalle:

**GreedySetCover**( $V, E$ ):

$C := \emptyset$

$E' := E$

**while**  $E' \neq \emptyset$  **do**

(1) valitse jokin  $(u, v) \in E'$

$C := C \cup \{u, v\}$

poista joukosta  $E'$  ne kaaret,  
joiden päätepiste on  $u$  tai  $v$

**return**  $C$

Selvästi algoritmi GreedyVertexCover toimii polynomisessa ajassa, ja sen palauttama  $C$  on solmupeite.

Mitä voidaan sanoa solmupeitteen  $C$  koosta verrattuna pienimpään mahdolliseen?

Olkoon  $A$  niiden kaarten joukko, jotka tulevat valituksi rivillä (1).

Koska lisäyksen  $C := C \cup \{u, v\}$  jälkeen poistetaan kaikki solmuihin  $u$  ja  $v$  liittyvät kaaret, millään kahdella joukon  $A$  kaarella ei voi olla yhteistä päätepistettä. Siis  $|C| = 2|A|$ .

Olkoon  $C'$  mielivaltainen solmupeite. Erityisesti  $C'$  sisältää ainakin toisen päätepisteen jokaisesta joukon  $A$  kaaresta. Koska mikään päätepiste ei kuulu useaan joukon  $A$  kaareen, niin  $|C'| \geq |A|$ .

Siis  $|C| = 2|A| \leq 2|C'|$ . Erityisesti jos  $C'$  on pienin solmupeite, nähdään että  $C$  on korkeintaan kaksi kertaa optimaalisen kokoinen.

Esitetään nyt yleisempiä määritelmiä.

Optimointiongelma on kolmikko  $\Pi = (D, S, c)$  missä

- joukko  $D$  on ongelman **tapausten** joukko,
- kullakin  $x \in D$  joukko  $S(x)$  on tapauksen  $x$  **mahdollisten ratkaisujen** joukko ja
- jokaisella  $x \in D$  ja  $s \in S(x)$  on määritelty **kustannus**  $c(x, s) \geq 0$ .

Esimerkiksi solmupeiteongelman tapauksessa  $D$  on kaikkien verkkojen kokoelma,  $S(G)$  on verkon  $G$  solmupeitteiden joukko ja  $c(G, C) = |C|$ .

Tarkastelemme jatkossa lähinnä minimointiongelmia. Tällöin merkitsemme  $c^*(x) = \min_s c(x, s)$ . Ratkaisu  $s^* \in S(x)$  on tapauksen  $x$  **optimaalinen ratkaisu** jos  $c(x, s^*) = c^*(x)$ . Ratkaisun  $s \in S(x)$  **approksimointisuhde** on

$$r(x, s) = \frac{c(x, s)}{c^*(x)}.$$

Algoritmi  $A$  on  **$\rho$ -approksimointialgoritmi** jos kaikilla  $x \in D$  sen tuloste  $s = A(x)$  toteuttaa ehdon

$$r(x, s) \leq \rho.$$

Siis aina  $\rho \geq 1$ , ja  $\rho = 1$  tarkoittaa tarkkaa ratkaisua.

**Esimerkki** GreedyVertexCover on VC-opt-ongelman 2-approksimointialgoritmi.

Toisaalta samansukuiselle klikin maksimointiongelmalle ei ole olemassa  $\rho$ -approksimointialgoritmia **millään**  $\rho$  ellei sitten päde  $P = NP$  [Arora, Lund, Motwani, Sudan, Szegedy 1992]. Tämä on esimerkki siitä, että yksinkertaisetkaan palautukset eivät yleensä säilytä approksimoituvuutta.

**Huomautus 1** Toisinaan ratkaisun  $s \in S(x)$  tarkkuus ilmoitetaan suhteellisena hyvyytenä

$$\frac{c(x, s) - c^*(x)}{c^*(x)} = r(x, s) - 1.$$

**Huomautus 2** Vastaavasti maksimointiongelmallalla  $c^*(x) = \max_s c(x, s)$  ja

$$r(x, s) = \frac{c(x, s)}{c^*(x)}.$$

Siis tässäkin aina  $r(x, s) \geq 1$ .

**Huomautus 3** Approksimaatiotarkkuus voi myös riippua syötteen koosta. Esim.  $\log n$ -approksimointialgoritmin tulosteella  $s$  pätee

$$c(x, s) \leq (\log n)c^*(x)$$

kaikilla  $x$  joiden koko on korkeintaan  $n$ .

Jos ongelmalle  $\Pi$  on jokaisella  $\rho > 1$  olemassa polynomisessa ajassa toimiva approksimointialgoritmi  $A_\rho$ , sanomme että ongelmalla  $\Pi$  on **polynominen approksimointiskeema**.

Tässä  $\rho$  ei ole osa algoritmien syötettä, vaan kullakin  $\rho$  on oma algoritminsä  $A_\rho$ . Erityisesti algoritmien  $A_\rho$  aikavaativuudet saavat kasvaa mielivaltaisen nopeasti, kun  $\rho$  lähestyy ykköstä.

**Esimerkki** Jos kauppamatkustajan ongelmassa oletetaan, että verkon solmut ovat tason pisteitä ja kustannusfunktio niiden välinen euklidinen etäisyys, niin ongelmalle on ajassa  $O(n(\log n)^{c/\varepsilon})$  ( $c$  vakio) toimiva  $(1 + \varepsilon)$ -approksimointialgoritmi millä tahansa  $\varepsilon > 0$  [Arora 1996].  $\square$

Algoritmi  $A$  on **täysin polynominen approksimointiskeema** ongelmalle  $\Pi$ , jos jokaisella syötteellä  $(x, \varepsilon)$  algoritmi

- pysähtyy ajassa  $p(|x|, 1/\varepsilon)$ , missä  $p$  on polynomi ja  $|x|$  syötteen  $x$  koko, ja
- tulostaa ratkaisun  $s \in S(x)$  jolla  $r(x, s) \leq (1 + \varepsilon)$ .

”Pelkkään” polynomiseen approksimointiskeemaan verrattuna lisävaatimus on siis, että suoritus aika on polynominen tarkkuusparametrin  $1/\varepsilon = 1/(1 - \rho)$  suhteen.

## 6.2 Kauppamatkustajan ongelma

Kauppamatkustajan ongelman (Travelling Salesman Problem, TSP) optimointiversio on seuraava:

- $D$  koostuu kaikista painotetuista suuntaamattomista verkoista  $(V, E, d)$ ,  $d: E \rightarrow \mathbf{R}_+$
- $S(V, E, d)$  koostuu verkon  $(V, E)$  Hamiltonin kehistä ja
- $c((V, E, d), p)$  on kehän  $p$  kaarien painojen summa.

Yleensä oletetaan että verkko on täydellinen ( $E = V \times V$ ). Tällöin kaaren  $(u, v)$  "puuttuminen" voidaan esittää asettamalla etäisyydeksi  $d(u, v)$  jokin suuri luku.

Tarkastelemme lähinnä ongelman **metristä** erikoistapausta  $\Delta$ TSP, jossa oletetaan lisäksi kolmioepäyhtälö

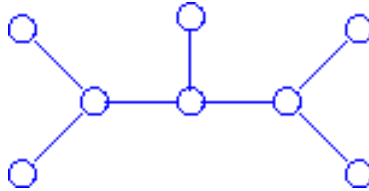
$$d(x, z) \leq d(x, y) + d(y, z)$$

kaikilla  $x, y, z \in V$ . Siis  $d$  on pseudometriikka joukossa  $V$ . (Koska verkko on suuntaamaton, on aina  $d(x, y) = d(y, x)$ .)

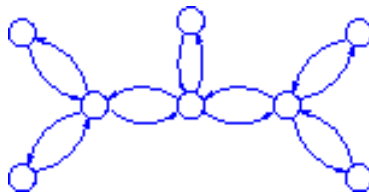
Tarkastellaan approksimointia pienimmän virittävän puun (MST) avulla.

MST-TSP( $V, E, d$ ):

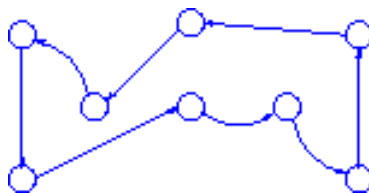
1. Muodosta verkon  $(V, E, d)$  pienin virittävä puu  $T \subseteq E$ .
2. Muodosta puusta  $T$  suunnattu verkko  $G' = (V, T')$  ottamalla jokainen kaari kumpaankin suuntaan. (Huom. verkossa  $G'$  jokaisen kaaren tuloaste = lähtöaste)
3. Muodosta verkossa  $G'$  Eulerin kehä  $p'$
4. Muodosta Eulerin kehästä  $p'$  Hamiltonin kehä  $p$  oikaisemalla niiden solmujen ohi, joissa on ko käyty.
5. Palauta  $p$ .



Pienin virittävä puu  $T$ .



Suunnattu verkko jossa Eulerin kehä



Eräs tapa oikaista polku

Aiemmin esitettyjen verkkoalgoritmien avulla on selvää, että MST-TSP voidaan toteuttaa polynomisessa ajassa. Tarkastellaan sen approksimointitarkkuutta.

Mistä tahansa Hamiltonin kehästä saadaan virittävä puu jättämällä pois yksi kaari. Siis jos  $c(p^*)$  on lyhimmän TSP-reitin kustannus ja  $c(T)$  pienimmän virittävän puun kustannus, pätee

$$c(T) \leq c(p^*).$$

Eulerin kehälle  $p'$  kukin puun  $T$  kaari tulee kahdesti, joten

$$c(p') = 2c(T).$$

Kolmioepäyhtälön nojalla

$$c(p) \leq c(p').$$

Yhdistämällä edelliset saadaan

$$c(p) \leq c(p') \leq 2c(T) \leq 2c(p^*),$$

joten MST-TSP on 1-approksimointialgoritmi.

Algoritmia voidaan parantaa valitsemalla verkko  $G'$  huolellisemmin.

Tarvittava perusominaisuus on, että verkossa  $G'$  voidaan muodostaa Eulerin kehä. Muodostetaan siis *suuntaamaton* verkko  $G'$ , jonka kaikki solmut ovat parillisasteisia.

Olkoon  $U \subseteq V$  niiden solmujen joukko, joiden aste verkossa  $(V, T)$  on pariton. Huomaa, että  $|U|$  on parillinen.

Tunnetuilla tekniikoilla (ks. esim. Papadimitriou ja Steiglitz) voidaan muodostaa sellainen solmujoukon  $U$  pariutus  $M$ , jonka **paino**  $c(M) = \sum_{(u,v) \in M} d(u,v)$  on pienin mahdollinen. Asettamalla nyt algoritmin MST-TSP askelessa 2

$$G' := (V, T \cup M)$$

saadaan **Christofidesin algoritmi**.

Huomaa, että verkossa  $G'$  jokainen solmu on parillisasteinen, joten Eulerin kehä  $p'$  todella voidaan muodostaa.

Algoritmin MST-TSP analyysin tapaan Christofidesin algoritmille saadaan

$$c(p) \leq c(T) + c(M) \leq c(p^*) + c(M).$$

Kustannuksen  $c(M)$  arvioimiseksi ajatellaan ensin muodostetuksi TSP-reitti  $\hat{p}$  pelkästään joukon  $U$  solmuille. Kun oletetaan  $\hat{p}$  valitun optimaalisesti, niin kolmioepäyhtälöstä seuraa

$$c(\hat{p}) \leq c(p^*).$$

Polusta  $\hat{p}$  saadaan kaksi joukon  $U$  pariutusta, joista ainakin toinen on kustannukseltaan korkeintaan puolet koko polun kustannuksesta. Siis

$$c(M) \leq \frac{1}{2}c(p^*).$$

Kaikkiaan siis

$$c(p) \leq \frac{3}{2}c(p^*)$$

eli Christofidesin algoritmi on 3/2-approksimointialgoritmi.

Yleisen TSP:n approksimoiminen näyttää paljon vaikeammalta.

**Lause** Jos  $P \neq NP$ , niin kauppamatkustajan ongelmalle ei ole polynomisessa ajassa toimivaa  $\rho$ -approksimointialgoritmiä millään vakiolla  $\rho$ .

**Todistus** Oletetaan, että  $A$  on  $\rho$ -approksimointialgoritmi. Ratkaistaan Hamiltonin kehä -ongelma (HC) algoritmin  $A$  avulla.

Kun on annettu suuntaamaton verkko  $G = (V, E)$ , muodostetaan TSP-tapaus  $(V, E', d)$  missä  $E' = V \times V$  ja

$$d(u, v) = \begin{cases} 1 & \text{jos } (u, v) \in E \\ \lceil \rho \rceil |V| + 1 & \text{muuten} \end{cases}$$

Väitetään, että verkossa  $G$  on Hamiltonin kehä, jos ja vain jos algoritmi  $A$  syötteellä  $(V, E', d)$  tulostaa reitin, jonka kustannus on korkeintaan  $\lceil \rho \rceil |V|$ .

Jos verkossa  $G$  ei ole Hamiltonin kehää, niin selvästi verkon  $(V, E', d)$  jokaisen TSP-reitin kustannus on vähintään  $\lceil \rho \rceil |V| + 1$ .

Oletetaan toisaalta, että verkossa  $G$  on Hamiltonin kehä. Siis verkon  $(V, E', d)$  optimaalisen TSP-reitin kustannus on  $|V|$ . Oletuksen mukaan  $A$  tulostaa reitin, jonka kustannus on korkeintaan  $\rho|V|$ .

Siis HC voidaan ratkaista seuraavasti:

1. Muodosta  $G' = (V, E', d)$  kuten edellä.
2. Ratkaise TSP-tapaus  $G'$  algoritmilla  $A$ .
3. Jos tuloksen kustannus on korkeintaan  $\rho|V|$ , vastaa "kyllä", muuten "ei".

Koska HC on NP-täydellinen, väite seuraa.  $\square$

## 6.3 Joukkopeiteongelma (Set Cover, SC)

Käytetään joukon  $X$  potenssijoukosta (kaikkien osajoukkojen joukosta) merkintää  $\mathcal{P}(X)$ :

$$\mathcal{P}(X) = \{U \mid U \subseteq X\}.$$

Osajoukkokokoelma  $F \subseteq \mathcal{P}(X)$  on perusjoukon  $X$  **joukkopeite** jos  $\cup C = X$ , ts. kaikilla  $x \in X$  pätee  $x \in U$  jollakin  $U \in C$ .

Tästä saadaan päätösongelma Set Cover:

**Annettu:** perusjoukko  $X$ , osajoukkokokoelma  $F \subseteq \mathcal{P}(X)$

**Kysymys:** onko olemassa  $C \subseteq F$  jolla  $|C| = k$  ja  $C$  on joukkopeite

Set Cover on NP-täydellinen (palautus ongelmasta Vertex Cover).

Tarkastellaan ahnetta approksimointialgoritmia joukkopeitteen minimointiongelmalle.

**GreedySetCover**( $X, F$ ):

$U := X$

$C := \emptyset$

**while**  $U \neq \emptyset$  **do**

    valitse  $S \in F$  jolla  $|S \cap U|$  on suurin

$U := U - S$

$C := C \cup \{S\}$

**return**  $C$

Ilmeisesti GreedySetCover toimii polynomisessa ajassa ja sen palauttama  $C$  on joukkopeite. Arvioidaan joukkopeitteen  $C$  kokoa.

Merkitään  $H_n = \sum_{i=1}^n (1/i)$ . Muistetaan  $H_n \sim \ln n$ .

**Lause** [Johnson 1979] GreedySetCover on  $H_m$ -approksimointialgoritmi, missä  $m = \max \{ |S| \mid S \in F \}$ .

**Huom.** tässä siis approksimointitarkkuus ei ole vakio vaan riippuu syötteen koosta.

**Todistus** Olkoot joukkopeitteeseen  $C$  valitut joukot valitsemisjärjestyksessä  $S_1, \dots, S_{|C|}$ . Merkitään

$U_i = X - \bigcup_{j=1}^{i-1} S_j$ . Siis iteraatiokierroksella  $i$  joukon  $S_i \cap U_i$  alkioit tulevat ensimmäistä kertaa peitetyksi.

Jokainen  $x \in X$  kuuluu joukkoon  $S_i \cap U_i$  tasan yhdellä  $i$ . Liitetään alkioon  $x$  paino  $c_x = 1/|S_i \cap U_i|$ . Siis joukon  $S_i \cap U_i$  kokonaispaino on

$$\sum_{x \in S_i \cap U_i} c_x = |S_i \cap U_i| \cdot 1/|S_i \cap U_i| = 1.$$

Koko perusjoukon paino on siis lopullisen peitteen koko:

$$\sum_{x \in X} c_x = |C|.$$

Olkoon  $C^*$  pienin joukkopeite. Voimme arvioida

$$|C| = \sum_{x \in X} c_x = \sum_{x \in C^*} c_x \leq \sum_{S \in C^*} \sum_{x \in S} c_x.$$

Osoitamme seuraavaksi, että kaikilla  $S \in F$  pätee

$$\sum_{x \in S} c_x \leq H_{|S|}.$$

Kiinnitetään mielivaltainen  $S \in F$  ja merkitään

$$v_i = |S - (S_1 \cup \dots \cup S_i)| = |S \cap U_{i+1}|.$$

Eryteisesti  $v_0 = S$ . Olkoon  $k$  pienin, jolla  $v_k = 0$  eli  $S \subseteq S_1 \cup \dots \cup S_k$ . Koska iteraatiokierroksella  $i$  peitetyksi tulee  $v_{i-1} - v_i$  joukon  $S$  alkioita,

$$\sum_{x \in S} c_x = \sum_{i=1}^k (v_{i-1} - v_i) |S_i \cap U_i|^{-1}.$$

Koska  $S_i$  valittiin ahneesti,

$$|S \cap U_i| \leq |S_i \cap U_i|,$$

joten

$$\begin{aligned} \sum_{x \in S} c_x &\leq \sum_{i=1}^k (v_{i-1} - v_i) |S \cap U_i|^{-1} \\ &= \sum_{i=1}^k (v_{i-1} - v_i) / v_{i-1}. \end{aligned}$$

Koska kaikilla  $a \leq b$  pätee

$$(b - a)/b \leq \frac{1}{b} + \frac{1}{b-1} + \frac{1}{b-2} + \dots + \frac{1}{a+1} = H_b - H_a,$$

saadaan

$$\begin{aligned} \sum_{x \in S} c_x &\leq \sum_{i=1}^k (H_{v_{i-1}} - H_{v_i}) \\ &= H_{v_0} - H_{v_k} \\ &= H_{|S|}. \end{aligned}$$

Saadaan siis

$$|C| \leq \sum_{S \in C^*} \sum_{x \in S} c_x \leq |C^*| H_{|S|}.$$

□

Ei pidetä luultavana, että joukkopeiteongelmalla olisi oleellisesti tätä parempaa approksimointialgoritmia.

Tiedetään, että jos ongelmalla olisi  $\rho$ -approksimointialgoritmi missä  $\rho < (1 - o(1)) \ln n$ , niin pätsi

$$\text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)})$$

eli NP-ongelmat olisivat vain "hieman" eksponentiaalisia [Feige 1996].