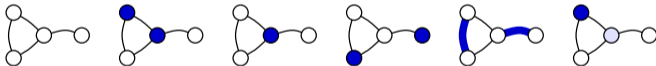


□ Seminaari: Hajautetut algoritmit syksy 2009

<http://www.cs.helsinki.fi/u/josuomel/sem-2009s/>



Jukka Suomela
17.9.2009

□ **Seminaari: Hajautetut algoritmit syksy 2009**

- Optimointiongelmien peruskäsitteistöä

□ Maksimaalinen vs. maksimi

Maksimointiongelman	Minimointiongelman
optimaalinen	optimaalinen
optimiratkaisu	optimiratkaisu
maksimikokoinen	minimikokoinen
maksimipainoinen	minimipainoinen
suurin, painavin	pienin, kevyin
<i>maximum</i>	<i>minimum</i>
maksimaalinen	minimaalinen
<i>maximal</i>	<i>minimal</i>

$$\alpha \geq 1$$

Minimointiongelma:

- α -approksimaatio = kelvollinen ratkaisu, jonka koko on enintään $\alpha \cdot \text{OPT}$

Maksimointiongelma:

- α -approksimaatio = kelvollinen ratkaisu, jonka koko on vähintään $\frac{1}{\alpha} \cdot \text{OPT}$
- Joissain lähteissä $\frac{1}{\alpha}$ -approksimaatio

$$\alpha \geq 1$$

α -approksimointialgoritmi = algoritmi, joka tuottaa α -approksimaation

- Tuttuja NP-kovien optimointiongelmiä yhteydestä: jos ei polynomisessa ajassa voida tai osata löytää optimia, onko mahdollista löytää hyvää approksimaatiota (pieni α)?
- Vastaavasti esim. hajautettujen algoritmien yhteydessä

□ **Approksimointi**

Maksimaalinen ratkaisu saattaa olla approksimaatio:

- Maksimaalinen pariutus on 2-approksimaatio maksimikokoisesta pariutuksesta

Saattaa myös olla olematta:

- Esim. maksimaalinen riippumaton joukko

□ **Seminaari: Hajautetut algoritmit syksy 2009**

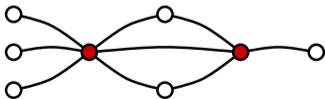
- Esimerkki optimointiongelmasta hajautettujen algoritmien näkökulmasta

□ Solmupeite

Minimointiongelma

Etsittävä solmujoukko, joka peittää kaikki kaaret
(joka kaaresta mukana ainakin yksi pää)

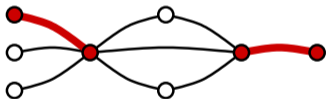
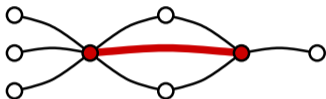
Klassinen NP-kova ongelma, ei tunneta edes
polynomiaikaista 1.99-approksimointialgoritmia



□ Solmupeite

Mutta 2-approksimointi on hyvin helppoa: etsi *maksimaalinen pariutus* ja ota kaarien päät

- Miksi solmupeite?
- Miksi 2-approksimaatio?

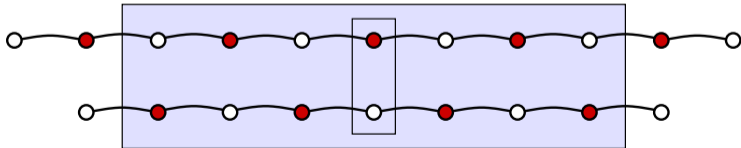


□ Solmupeite hajautetusti

Helppo näyttää, että millään hajautetulla algoritmilla ei pääse optimiin ajassa $o(n)$

- Ei tarvita mitään oletuksia tyyliin $P \neq NP$

Entä tehokas hajautettu approksimointialgoritmi?



□ Solmupeite hajautetusti

Kyllä, voimme käyttää esim. samaa tekniikkaa kuin polynomiaikaisessa 2-aproksimointialgoritmissa:

- Etsitään maksimaalinen pariutus (aihe B4)
- Deterministinen hajautettu algoritmi
- Ajoaika esim. $O(\log^4 n)$ tai $O(\Delta + \log^* n)$

Δ = yläraja verkon solmujen asteluvulle

□ Solmupeite hajautetusti

Deterministisissä pariutusalgoritmeissa kuitenkin rajoitteita:

- Joudutaan olettamaan, että solmuilla yksilölliset tunnisteet (symmetrian rikkominen)
- Ajoaika $\Omega(\log^* n)$ kierrosta (aihe A4)

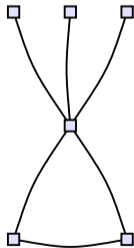
Tutustutaan approksimointialgoritmiin, jolla ei näitä rajoitteita

□ **Seminaari: Hajautetut algoritmit syksy 2009**

- Esimerkki hajautetusta approksimointialgoritmista:
solmupeitteen 3-approksimointi

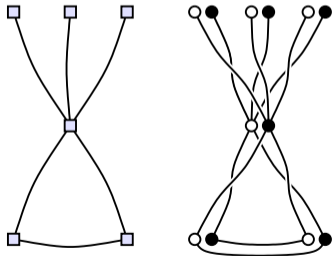
□ Solmupeitteen 3-approksimointi

Syöte \mathcal{G}



□ Solmupeitteen 3-approksimointi

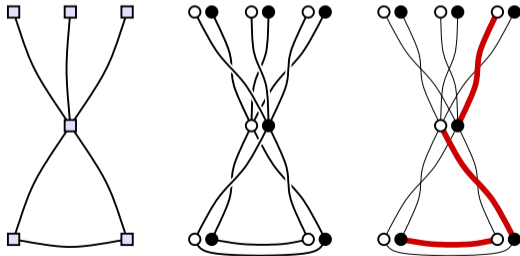
- Kahdennetaan solmut: musta ja valkoinen kopio
- Yhdistetään kaaret ristiin
- Saadaan kaksijakoinen, *2-väritetty* verkko



□ Solmupeitteen 3-aproksimointi

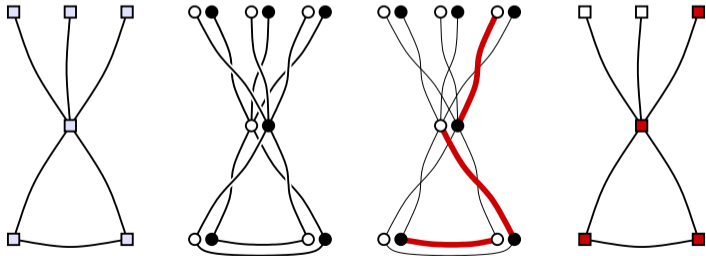
Etsitään verkossa *maksimaalinen pariutus*:

- mustat lähettävät ehdotuksia valkoisille naapureille
- valkoiset vastaavat myöntävästi ensimmäiselle



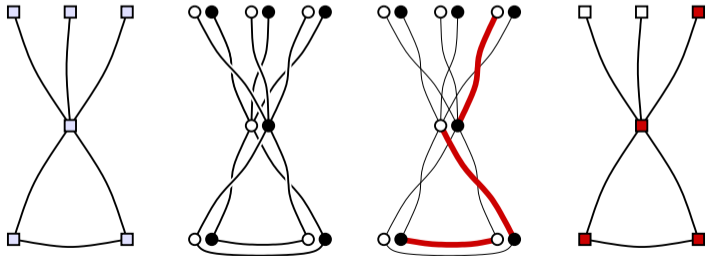
□ Solmupeitteen 3-aproksimointi

Solmupeitteeseen mukaan alkuperäisen verkon solmut, joihin liittyy ainakin yksi pariutuksen kaari



□ Solmupeitteen 3-approksimointi

- Ajoaika 2Δ kierrosta
- Miksi solmupeite?
- Miksi enintään 3 kertaa optimin kokoinen?



□ Solmupeitteen 3-approksimointi

- Symmetrian rikkominen: ei tarvittu – miksi?
- Rinnakkaisuus: vuorotellen aktiivisina kaikki mustat kopiot tai kaikki valkoiset kopiot

