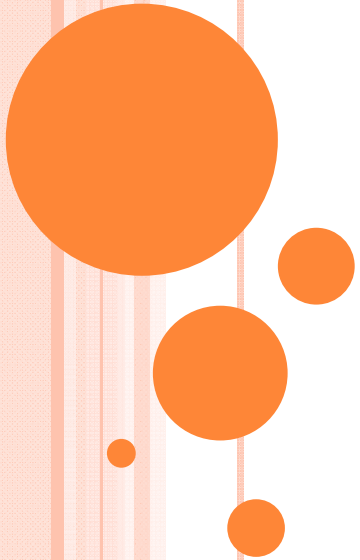


Paritus kaksijakoisessa graafissa

Informaatiotekniikan seminaari

Pekka Rossi 4.3.2008

Sisältö

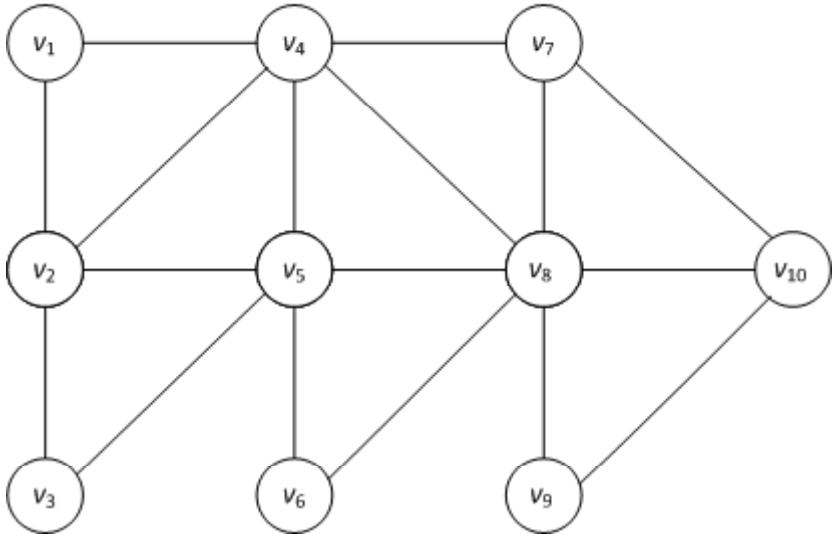
- Johdanto
- Kaksijakoinen graafi
- Sovituksen peruskäsitteet
- Sovitusongelma
- Lisäyspolku
- Bipartite matching-algoritmi
- Kaksijakoisen verkon sovitus ja maksimivuo
- Yhteenveto

JOHDANTO

- *Paritus* tai *sovitus* (matching) on verkon viivojen joukko, jossa kaksi viivaa ei voi liittyä yhteen solmuun
- Ongelmalla monia käytännön sovelluksia
- Tässä käydään läpi *painottoman kaksijakoisen verkon* sovitus
  - Hieman vaativamman ongelman muodostaa *painotetun verkon sovitus* (weighted bipartite matching)

JOHDANTO

- Lähtökohtana sovitusongelmassa on graafi G
  - $G = (V,E)$
  - V *solmujen* (nodes, vertices) joukko
  - E *viivojen* (edges) joukko



KAKSIJAKOINEN GRAAFI

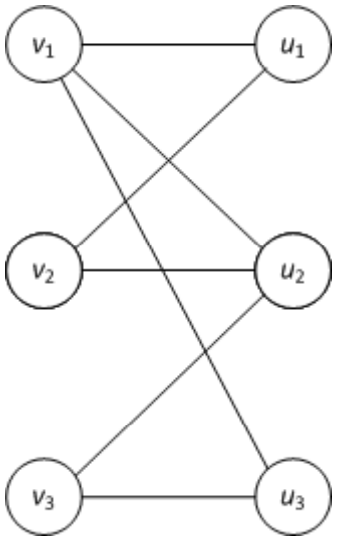
- Olkoon graafi $B = (W,E)$, jossa joukko W sisältää graafin solmut ja E viivat

- Oletetaan lisäksi että W on jaettu kahteen osajoukkoon, V ja U, ja jokaisella joukon E viivalla on toinen solmu joukossa U ja toinen joukossa V

  - Tällöin graafi $B=(W,E)=(V,U,E)$ on *kaksijakoinen* (bipartite graph)

KAKSIJAKOINEN GRAAFI

- Graafi on kaksijakoinen jos ja vain jos se ei sisällä parittoman pituisia kierroksia

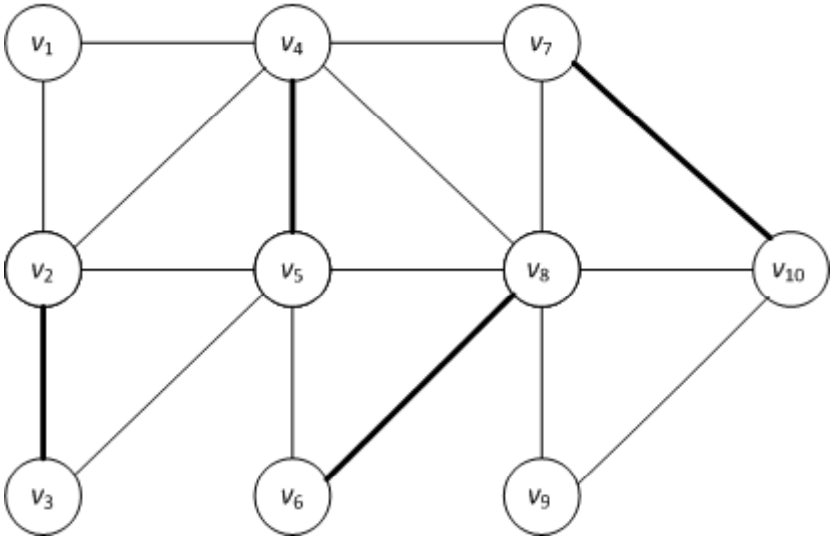


Sovituksen Peruskäsitteet

- *Sovitus M*:
  - Viivojen joukko, jossa kaksi viivaa ei liity samaan solmuun
    - Joukkoon *M* kuuluvia viivoja kutsutaan *sovitetuiksi viivoiksi* (matching edges)
    - Muut verkon viivat ovat *vapaita* (free)
    - Jos solmu ei kuulu mihinkään sovitettuun viivaan, on se *sovittamaton* (exposed)
  - Sovituksen koko $|M|$ on sovitteessa *M* olevien viivojen lukumäärä
  - Jos sovitteen *M* kokoa ei voida kasvattaa graafissa *G*, on se *maksimisovitus*.

Sovituksen Peruskäsitteet

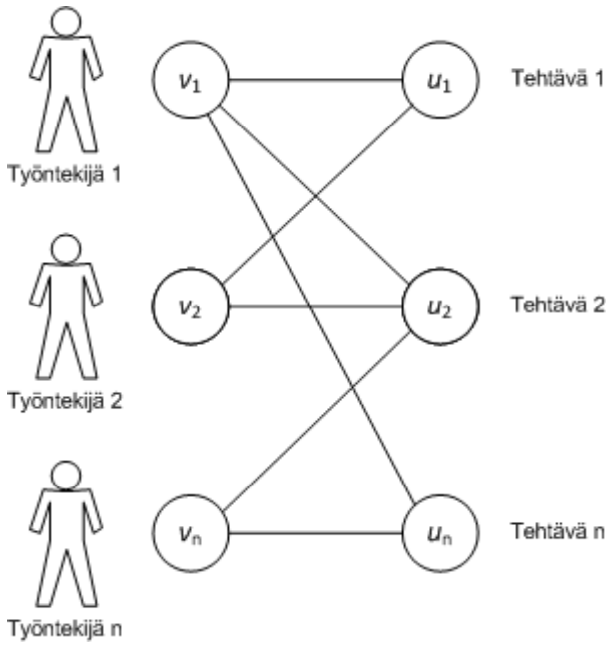
- Esimerkki:



- Viivat $M = \{[v_2,v_3],[v_4,v_5],[v_6,v_8],[v_7,v_{10}]\}$ muodostavat sovitteen
- Muut viivat ovat vapaita
- Solmut v_1 ja v_9 ovat sovittamattomia

SOVITUSONGELMA

- *Sovitusongelma*:
  - Löydä maksimisovitus graafissa $G = (E,V)$
- Esim: Assignment Problem
  - n miestä ja n työtehtävää. Miten jakaa työtehtävät miehille, kun kaikki miehet eivät voi tehdä kaikkia työtehtäviä?
    - Muodostetaan ongelmasta kaksijakoinen graafi
    - Ratkaisu saadan graafin maksimisovituksesta

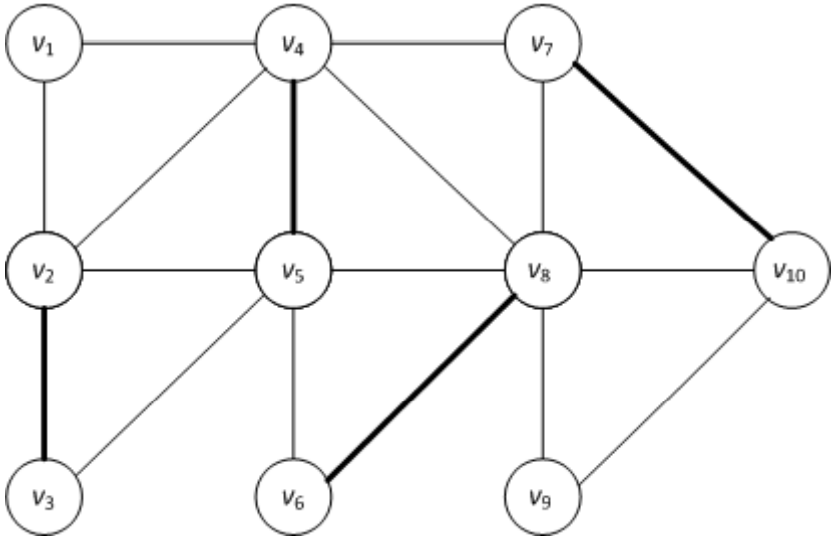


LISÄYSPOLKU

- Verkon G polku $p = [u_1,u_2,..,u_k]$ on *vuorotteleva* (alternating path), jos
  - Ensimmäinen polun alkio on sovittamaton
  - Joka toinen viiva on vapaa ja joka toinen sovitettu

- Jos myös vaihtelevan polun p viimeinen alkio on sovittamaton, p on *lisäyspolku* (augmented path)
  - Tämä on tärkeä käsite, jota sovelletaan kaksijakoisen graafin sovitusongelman ratkaisemiseen

LISÄYSPOLKU



- Esimerkiksi polku $p_1 = [v_1,v_2,v_3,v_5,v_4,v_8]$ on vuorotteleva polku
- Polku $p_2 = [v_1,v_4,v_5,v_6,v_8,v_9]$ on lisäksi myös lisäyspolku

SOVITTEEN KASVATTAMINEN

- Sovitteen kokoa voidaan kasvattaa lisäyspolun avulla

- **Lemma 1:**

Olkoon M verkon sovitus ja P lisäyspolun p kaaret.

Tällöin $M' = M \otimes P = XOR(M, P)$ on sovite kooltaan $|M|+1$

  - Todistuksen idea:
    - Näytetään ensin, että M' = XOR(M,P) on sovite
    - Näytetään että, uuden sovitteen koko on $|M|+1$
      - Toteutuu, jos joukossa P on joukkoon M kuulumattomia viivoja

Sovitteen kasvattaminen

- **Teoreema 1:**
  - *Verkon G sovite M on maksimaalinen jos ja vain jos verkossa G ei ole lisäyspolkua*
    - Heuristinen perustelu:
      - Jos verkolla on maksimaalinen sovitus, joukossa V tai joukossa U kaikki solmut ovat sovitettuja. Tällöin verkossa ei ole myöskään lisäyspolkua.

BIPARTITE MATCHING-ALGORITMI

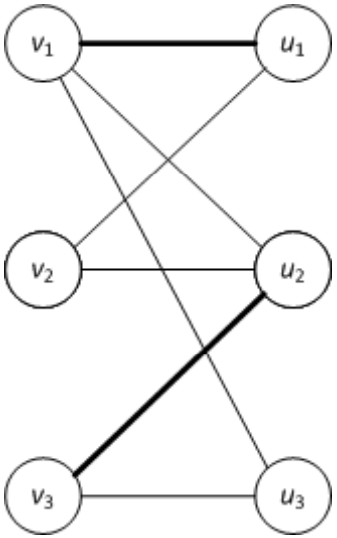
- Bipartite matching-algoritmi nojaa voimakkaasti Lemmaan 1 ja teoreemaan 1

- Algoritmin idea yksinkertaisesti:
  1. Etsi jokin lisäyspolku P
  2. Muodosta $M' = M \otimes P$
  3. Mene takaisin kohtaan 1. ja toista kunnes verkosta ei löydy enää lisäyspolkua

BIPARTITE MATCHING-ALGORITMI

- Oletataan esimerkiksi, että meillä on kuvan a) sovite $M = [(v_3,u_2),(v_1,u_1)]$
  - Tällöin laajennuspolkua voidaan etsiä lähtemällä liikkeelle sovittamattomasta solmusta v_2
    - Tämä voidaan tehdä bfs-hakuna, kunnes löydetään toinen sovittamaton solmu v_3 (kuva b)
    - Saadaan laajennuspolun viivat $P = [(v_2,u_2),(u_2,v_3),(v_3,u_3)]$

a)

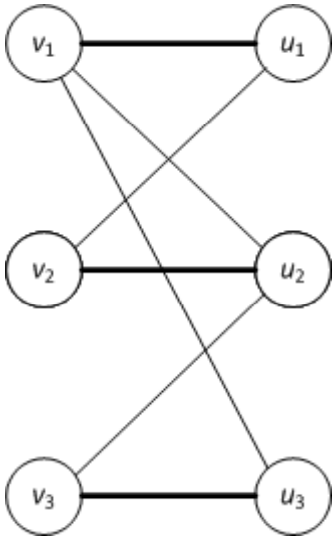


b)

BIPARTITE MATCHING-ALGORITMI

- Nyt $M' = M \otimes P =$
  $[(v_3,u_2),(v_1,u_1)] \otimes [(v_2,u_2),(v_3,u_2),(v_3,u_3)] =$
  $[(v_1,u_1),(v_2,u_2),(v_3,u_3)]$

- Kuvan verkolle tämä on myös maksimisovitus



KAKSIJAKOISEN VERKON SOVITUS JA MAKSIMIVUO

- Kaksijakoisen verkon sovitusongelma voidaan muuttaa max-flow-ongelmaksi

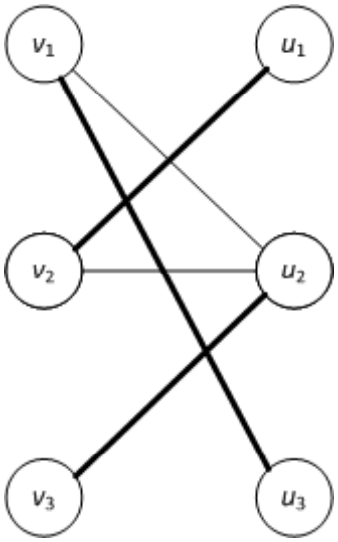
- Olkoon $B = (V,U,E)$ kaksijakoinen verkko. Määritellään verkon B avulla seuraava flow-verkko $N(s,t,W,A)$, jossa kunkin kaaren kapasiteetti on yksi
  1. Kaari lähteestä (source) s jokaiseen joukon V alkioon
  2. Kaari jokaisesta joukon U alkosta nieluun (sink) t
  3. Korvataan verkon B viivat joukosta V joukkoon U suuntautuvilla kaarilla

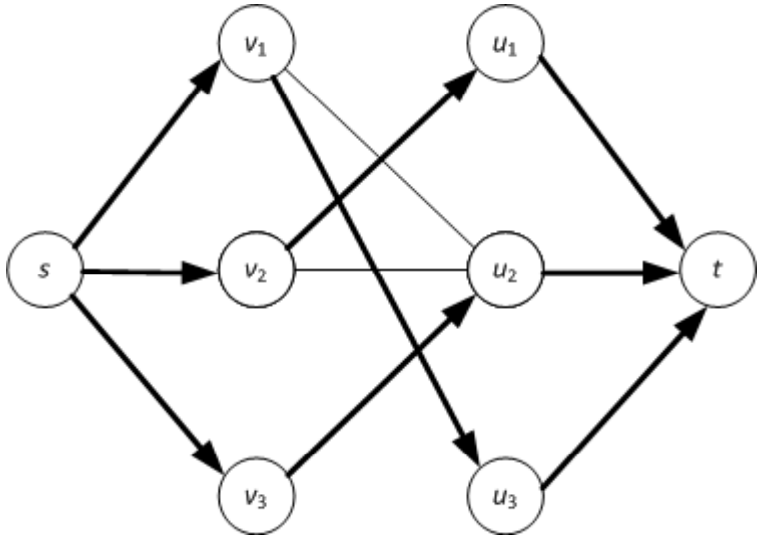
KAKSIJAKOISEN VERKON SOVITUS JA MAKSIMIVUO

- **Teoreema 2**

  Kaksijakoisen verkon maksimisovitus on yhtä suuri kuin verkon N(B) s-t-maksimivuo

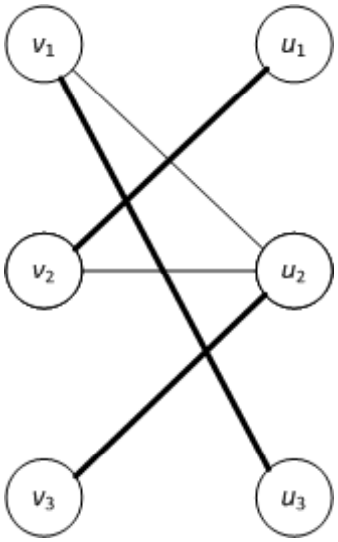
  - Kaksijakoisen verkon sovitus voidaan ratkaista max-flow-ongelman menetelmillä

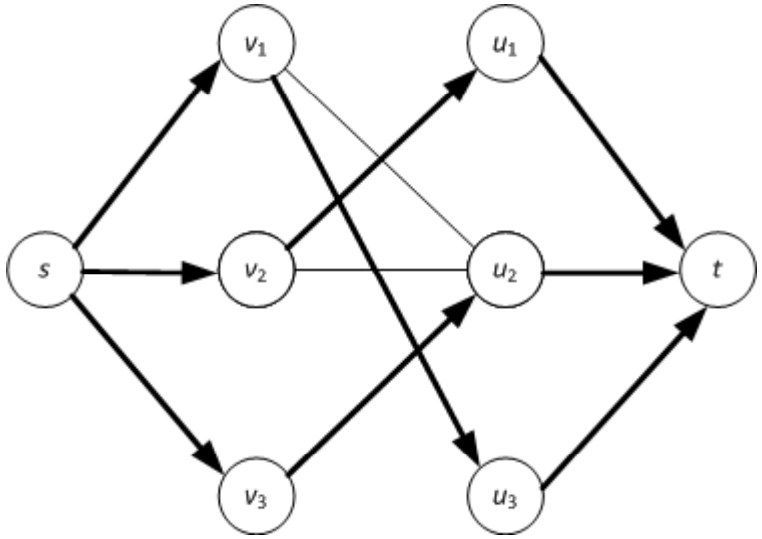




KAKSIJAKOISEN VERKON SOVITUS JA MAKSIMIVUO

- Perustelu maksimivuolle:
  - Jokaisesta joukon V solmusta lähtevän kaaren vuo on yksi
  - Jos solmuun tulee kaksi kaarta ja toisen kaaren vuo 1, niin toisen kaaren vuon täytyy olla 0, koska muuten vuon säilyminen rikkoutuu





YHTEENVETO

- Verkon sovitusongelma voidaan ratkaista lisäyspolkujen avulla
  - Tällöin maksimisovitus löytyy kasvattamalla sovituksen kokoa iteratiivisesti lisäyspoluilla
  - Analogia muiden LP-ongelmien iteratiivisten ratkaisumenetelmien kanssa
- Sovitusongelma voidaan muuttaa max-flow-tehtäväksi
  - Tällöin maksimisovitus voidaan ratkaista max-flow-menetelmien avulla