Kotitentti kurssille ***Verkkoteoria tietojenkäsittelytieteissä***

Tekijöinä Annemari Auvinen ja Mikko Vapa

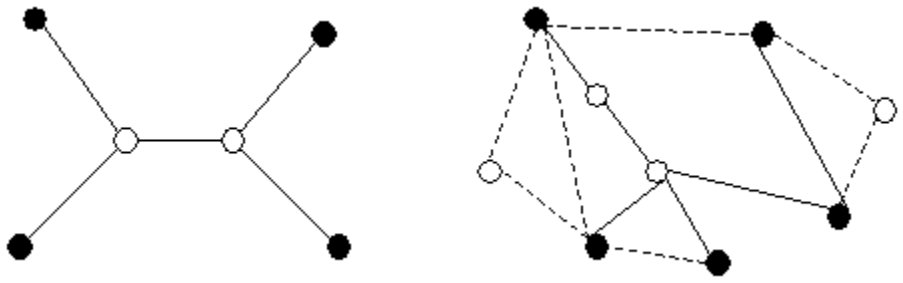
Minimaaliset Steiner-puut ja vertaisverkot

Minimaaliset Steiner-puut [1]

Normaali virittävä puu virittää kaikki verkon solmut, Steiner-puu sen sijaan virittää jonkun määrätyn solmujen osajoukon. Steinerin pienimmän virittävän puun ongelmassa solmut jaetaan kahteen osaan: päätesolmuihin ja ei-päätesolmuihin. Päätesolmut kuuluvat siihen annettuun solmujen joukkoon, joiden pitää kuulua ratkaisuun. Steiner-puun kustannukset määritellään välien painojen kokonaisarvoilla. Steiner-puu voi sisältää joitakin ei-päätesolmuja kustannusten vähentämiseksi. Olkoon V solmujen joukko. Yleisesti ottaen solmuista annetaan päätesolmujen joukko $L \subset V$ ja metriikka, joka määrittelee kahden solmun välisen etäisyyden. Tavoitteena on löytää yhtenäinen aliverkko, joka virittää kaikki päätesolmut minimaalisella yhteiskustannuksella. Koska etäisyydet ovat positiivisia, ratkaisu on puun mallinen. Annetusta metriikasta riippuen Steiner-puu-ongelmasta on tutkittu kahta muunnelmaa.

- Steinerin minimaaliset puut (Steiner minimal trees, SMT): Muunnelmassa solmujen joukko ja metriikka on annettu äärellisessä verkossa.
- Euklidisissa Steinerin minimaalisissa puissa (Euclidean SMT): Solmujen joukko on kokonainen Euklidinen avaruus ja siten ääretön.

Kuvassa 1 on esimerkki molemmista verkoista.



Kuva 1: Vasemmalla on euklidinen Steinerin minimaalinen puu ja oikealla verkko Steinerin minimaalinen puu. Kuvissa päätesolmut ovat mustia ja ei-päätesolmut valkoisia.

Minimaalisen Steiner-puun arviointi MST:llä [1]

Olkoon $G = (V,E,w)$ suuntaamaton verkko positiivisilla välien painoilla. Annettuna on päätesolmujen joukko $L \subset V$, Steinerin minimaalinen puu on puujoukko $T \subset G$ s.e. T sisältää kaikki solmut joukosta L.

Ongelma: Steinerin minimaalinen puu verkolle

Ilmentymä: Verkko $G = (V,E,W)$ ja päätesolmujen joukko $L \subset V$

Tavoite: Löytää puu T päätesolmujen joukolla $L \subset V(T)$ s.e. $w(T)$ minimoituu

Ongelma on NP-täydellinen [2], joten tyydymme likiarvoiseen tulokseen.

Tunnettu metodi SMT:n arviointiin on käyttää minimaalista virittävää puuta (minimal spanning tree, MST). Ensin muodostetaan täydellinen verkko solmuilla L ja välien painoilla, jotka ovat yhtä suuria kuin lyhimmät polkujen pituudet. Sen jälkeen etsitään minimaalinen virittävä puu, jossa jokainen väli vastaa yhtä lyhintä polkua alkuperäisessä verkossa. Lopulta minimaalinen virittävä puu muunnetaan takaisin Steiner-puuksi korvaamalla jokainen väli lyhimmällä polulla ja poistamalla mahdolliset syklit.

MST-Steiner –algoritmi

Syöte: Verkko $G = (V,E,w)$ ja päätesolmujen joukko $L \subset V$

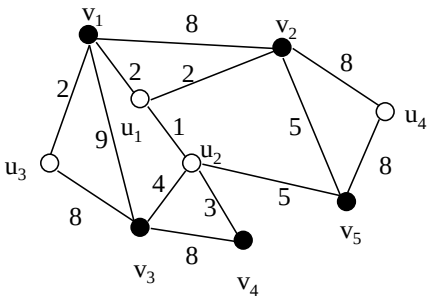
Tulos: Steiner-puu T

1. Muodosta täydellinen verkko G_L päätesolmujen joukosta L
2. Etsi minimaalinen virittävä puu T_L verkosta G_L
3. $T \leftarrow 0$
4. Tee kaikille väleille $e = (u,v)$, jotka kuuluvat $E(T_L)$
    a. Etsi lyhin polku P solmusta u solmuun v verkossa G
    b. Jos P sisältää vähemmän kuin kaksi verkon T solmua niin lisää P T:hen. Muuten oletetaan p_i:n ja p_j:n olevan ensimmäinen ja viimeinen solmu, jotka löytyvät T:stä. Lisää alipolku solmusta u solmuun p_i ja solmusta p_j solmuun v T:hen
5. Tulos Steiner-puu T

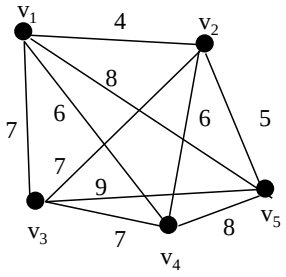
Algoritmissa siis korvataan vaiheessa neljä jokainen väli T_L:ssä vastaavalla lyhimmällä polulla. Mikäli kaksi solmua ovat jo puussa, polun lisääminen aiheuttaa syklin. Tässä tapauksessa lisätään vain alipolku päätesolmuista puussa oleviin solmuihin. Näin vältetään syklit ja taataan, että päätesolmut kuuluvat ratkaisuun. Algoritmi palauttaa tuloksena Steiner-puun.

MST-Steiner –algoritmi löytää SMT:n 2-likiarvon yleiselle verkolla ajassa $O(|V||L|^2)$, missä V on solmujen ja L päätesolmujen joukko. Tämä tarkoittaa sitä, että algoritmi voi tehdä enintään kaksinkertaisen virheen verrattuna minimaaliseen Steiner-puuhun.

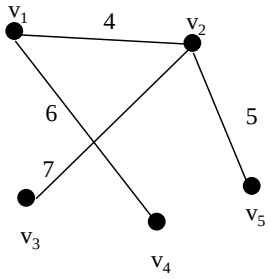
Esimerkki MST-Steiner –algoritmin toiminnasta



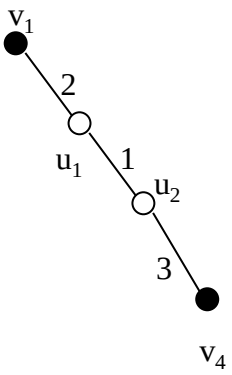
Kuva 2. Verkko G



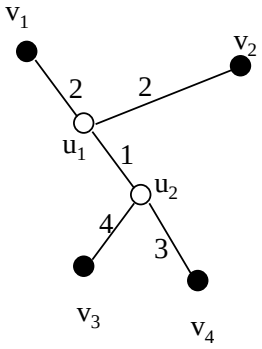
Kuva 3. Verkko G_L



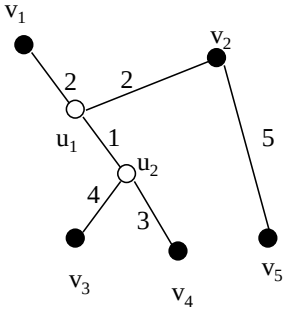
Kuva 4. Puu T_L



Kuva 5. Puu T_L ensimmäisellä iteraatiolla



Kuva 6. Puu T_L toisella iteraatiolla



Kuva 7. Puu T_L viimeisellä iteraatiolla

Kuvassa 2 on annettu verkko *G*, missä päätesolmujen joukko on $L = \{v_i | 1 \le i \le 5\}$ ja $u_i, 1 \le i \le 4$ ovat ei-päätesolmuja. Kuvassa 3 on muodostettu täydellinen verkko G_L ja pienin päätöspuu T_L vastaavasti kuvassa 4. Oletuksena *T* on tyhjä. Oletetaan, että väli $(v_1, v_4) \in E(T_L)$ on ensimmäinen valittu väli. Vastaava lyhin polku *G*:ssä on (v_1, u_1, u_2, v_4). Koska kaikki polun solmut eivät ole *T*:ssä, koko polku lisätään *T*:hen (kuva 5). Toisella iteraatiolla valitaan väli $(v_2, v_3) \in E(T_L)$. Vastaava lyhin polku on (v_2, u_1, u_2, v_3). Kuitenkin u_1 ja u_2 ovat jo *T*:ssä. Siksi vain (v_2, u_1) ja (u_2, v_3) lisätään (kuva 6). Kolmannella iteraatiolla valitaan väli (v_2, v_5) ja lisätään sama väli (kuva 6). Viimeisenä välinä T_L:ssä on (v_1, v_2). Koska sekä v_1 että v_2 ovat jo *T*:ssä, ei yhtään väliä lisätä ja puu kuvassa 7 on tulos.

Vertaisverkot

Vertaisverkot ovat hajautettuja järjestelmiä, jotka koostuvat keskenään resursseja jakavista prosesseista. Tyypillinen käyttökohde vertaisverkoilla on esim. tiedostojen jakaminen. Yksi käyttäjä voi jakaa oman levynsä tiedostot muiden vertaisten saataville ja myös samalla etsiä ja käyttää muiden levyillä olevia tiedostoja. Näin ollen järjestelmään liittyneet prosessit toimivat sekä palvelimena että asiakkaana tarjoten ja kuluttaen resursseja.

Vertaisverkkojen keskeinen ongelma on resurssien etsintä, koska järjestelmässä ei ole keskitettyä pistettä tai hakemistoa, josta resurssien tiedot voitaisiin löytää. Ongelma voidaan ratkaista käyttämällä hajautettua hakualgoritmia, jossa kyselevä prosessi lähettää kyselyn naapureilleen ja jotka taas lähettävät kyselyään eteenpäin heidän naapureilleen. Lopulta kun kyselyt ovat edenneet sellaisille solmuille, joilta haluttu resurssi löytyy he palauttavat tiedot resursseista, jonka jälkeen kyselyn aloittanut solmu voi käyttää resurssia esim. hakemalla tiedoston lähimmältä vertaiselta.

Hakualgoritmi toimii optimaalisesti, jos kysely välitetään vain niille naapureille, jotka joko tarjoavat haluttua resurssia tai jotka voivat välittää kyselyn lyhintä tietä resurssin omaaville prosesseille. Ongelmana tämä voidaan mallintaa minimaalisena Steiner-puuna, jossa pääteterminaaleina on resursseja omistavat prosessit ja koko verkon solmujoukkona kaikki vertaisverkon prosessit.

Kuitenkin vertaisverkon resurssien hakuongelma on hieman erilainen kuin minimaalisen Steiner-puun löytäminen, koska hakualgoritmin ei yleensäkään tarvitse löytää kaikkia resursseja vaan esimerkiksi kriteerinä voi olla löytää puolet tarjolla olevista resursseista. Tällöin hakualgoritmin aiheuttamaa kuormaa voidaan verkossa vähentää huomattavasti, mikä on tärkeää esim. mobiililaitteiden muodostamassa vertaisverkoissa, jossa akunkulutus ja näin ollen kyselypakettien lähetysten määrä pitää minimoida.

Steiner-puu ongelmasta on siis muokattava *k*-Steiner-puu ongelma, jossa riittää valita pelkästään *k*-kappaletta päätesolmuja niin, että yksi solmuista (kyselyn aloittaja) on puun juuri ja loput *k-1* solmua ovat löydetyt resurssit. Alustavan kirjallisuuskatsauksen perusteella näyttäisi siltä, että kyseistä Steiner-puu ongelman erityistapausta ei ole kirjallisuudessa tutkittu, joten ratkaisualgoritmia ei ole suoraan saatavilla.

Yksi tapa ratkaista ongelma on muuttaa likiarvoista MST-Steiner ratkaisualgoritmia, niin että algoritmin kohdassa 2 etsitään minimaalinen virittävä puu T_L lähtien juurisolmusta ja lopettaen silloin kun virittävässä puussa on yhteensä *k*-solmua. Tällöin tuloksena on *k*-Steiner-puun likiarvoinen ratkaisu.

Lähteet

[1] B. Y. Wu ja K.-M. Chao, ”Spanning Trees and Optimization Problems”, julkaistu kirjasarjassa Discrete Mathematics and Its Applications, 184 sivua, Chapman & Hall/CRC, 2004.

[2] R. M. Karp, ”Reducibility among combinatorial problems”, julkaistu kirjassa Complexity of Computer Computations, sivut 85-103, Plenum Press, New York, 1975.