Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

Verkkojen piirtäminen hierarkkisella ryhmittelyllä

Kandidaatintyö

13.4.2014

Artturi Tilanterä

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä:	Artturi Tilanterä
Työn nimi:	Verkkojen piirtäminen hierarkkisella ryhmittelyllä
Päiväys:	13.4.2014
Sivumäärä:	28
Pääaine:	Ohjelmistotekniikka
Koodi:	T3001
Vastuuopettaja:	Professori Juho Rousu
Työn ohjaaja(t):	Tohtorikoulutettava Tapio Auvinen, tohtorikoulutettava Lasse Hakulinen (Tietotekniikan laitos)

Verkottunutta informaatiota on monilla tieteenaloilla ja eri käyttötarkoituksiin. Informaation havainnollistaminen kuvana on tärkeää ihmisanalyysiä varten. Verkoista voidaan piirtää kuvia automaattisesti useilla menetelmillä.

Suurten verkkojen piirtämisen ongelmana on piirtomenetelmien pitkä laskenta-aika ja ihmisen käsityskyvyn rajallisuus. Tässä kandidaatintyössä ongelmaa lähestytään verkon hierarkkisella ryhmittelyllä ja ryhmittelyä hyödyntävillä piirtomenetelmillä. Hierarkkinen ryhmittely jakaa verkon solmuja ryhmiin, jotka edelleen sisältävät ryhmiä. Näin on mahdollista piirtää jotkin osat verkosta pelkistettyinä, mikä auttaa verkon päärakenteen hahmottamista.

Tutkimuksessa käy ilmi, että on kehitetty useita tehokkaita hierarkkiseen ryhmittelyyn perustuvia piirtomenetelmiä ja vuorovaikutteisia piirto-ohjelmistoja. Menetelmät tuottavat kaksiulotteisen, kiinteän kuvaesityksen suuntaamattomasta verkosta ja eroavat syötteen ja tehokkuuden suhteen. Piirtomenetelmien tehokkuus on polynomiaikainen ja enintään $O(n^3)$. Valmiiksi ryhmitelty, kymmenientuhansien solmujen ja kaarien verkko on mahdollista piirtää sekunneissa ja yleisemmin ajassa $O(n \log n + m)$. Esikäsitellyn verkon osanäkymästä toiseen voi siirtyä sekunnissa, mikä mahdollistaa vuorovaikutteisen havainnollistamisen. Tunnetuimmissa verkonpiirto-ohjelmistoissa hierarkkista ryhmittelyä ei ole vielä toteutettu.

Avainsanat:	suuntaamaton verkko, graafi, piirtäminen, hierarkkinen ryhmittely, informaation visualisointi
Kieli:	Suomi

Sisältö

1 Johdanto

Verkkomuotoista informaatiota on monenlaista: eräitä esimerkkejä ovat ihmisten väliset ystävyyssuhteet, sukupuut, WWW-sivujen hyperlinkit, sähkö-, tie- ja tietoliikenneverkot, aineenvaihdunta, ravintoketjut ja proteiinien vuorovaikutukset (Newman, 2003, s. 174–180).

Verkkomuotoisen informaation esittäminen kuvana on tärkeää, jotta ihminen voi analysoida verkon rakennetta. Verkkojen kuvaesityksiä voidaan tuottaa tietokoneella automaattisesti. Kuvaesityksen tuottamiseen on monia menetelmiä, joita kutsutaan tässä *verkonpiirtoalgoritmeiksi*.

Verkonpiirtoalgoritmien suoria käyttökohteita ovat esimerkiksi navigointi vuorovaikutteisissa järjestelmissä, talouden alojen suuren mittakaavan analyysi ja tyylitellyn joukkoliikennekartan tuottaminen. Teknisempiä sovelluksia ovat puheentunnistusohjelman kehittäminen, suurten järjestelmien valvonta sekä ohjelmisto- ja automaatiojärjestelmien määrittely. (Fleischer ja Hirsch, 2001, s. 3–17; Sugiyama, 2002, s. 147–167; Tuomi, 2012)

Monet todellista maailmaa esittävät verkot ovat suuria, tuhansien tai miljoonien solmujen kokoisia. Ongelmaksi nousee se, että verkon piirtäminen kokonaisuudessaan kaikkine yksityiskohtineen on hidasta tietokoneelle, ja monimutkaisen kaavion ymmärtäminen on vaikeaa ihmiselle. Kun verkko sisältää tuhansia solmuja, esitystä pitää pelkistää, jotta ihmissilmä ymmärtäisi kokonaiskuvan (Batagelj ym., 2011, s. 1587). Tarvitaan siis verkon pelkistämistä siten, että ensin näytetään yleiskuva, sitten lähennetään näkymä johonkin osaan, ja lopuksi yksityiskohtia voidaan näyttää valikoivasti (Shneiderman, 1996, s. 337). Edelleen verkon rakenteen tunnistamisen ja piirtämisen pelkistämisen on hyvä olla automaattista. Tätä ongelmaa lähestytään tässä kandidaatintyössä verkon *hierarkkisella ryhmittelyllä* ja hierarkkisesti ryhmitellyn verkon piirtoalgoritmeilla.

Hierarkkinen ryhmittely sopii todellisen maailman verkoille. Newman (2003, s. 180–186 ja 193–195) esittää, että todellisen maailman verkoilla on muun muassa seuraavat ominaisuudet. Ensiksikin verkot ovat valmiiksi *ryhmittyneitä*: jos solmut a ja b sekä b ja c ovat yhteydessä toisiinsa, todennäköisesti myös solmujen a ja c välillä on yhteys. Toiseksi erityisesti sosiaalisilla ja biologisilla verkoilla on *yhteisörakenne* (community structure): verkossa on solmujen ryhmiä, joiden sisällä on paljon yhteyksiä, kun taas ryhmien välillä on vähemmän yhteyksiä. Myös Didimo ja Montecchiani (2014, s. 185–186) mainitsevat yhteisörakenteen. Verkon yhteisörakenteen tunnistamiseen on muitakin menetelmiä kuin hierarkkinen ryhmittely. Esimerkiksi Girvan ja Newman (2002) esittelevät kaarien ”välissäolevuuteen” (edge ”betweenness”) perustuvan algoritmin. Muita ryhmittelytapoja kuin hierarkkinen ei kuitenkaan tässä työssä käsitellä.

Tämän kandidaatintyön tavoitteena on selvittää, mitä hierarkkisesti ryhmitteleviä ver-

konpiirtoalgoritmeja on olemassa, ja miten ne eroavat toisistaan teoreettisesti. Lisäksi kartoitetaan hierarkkisen ryhmittelyn toteutuksia tunnetuissa verkonpiirto-ohjelmissa.

Tässä kandidaatintyössä keskitytään verkonpiirtoalgoritmeihin seuraavalla tavalla rajatusti. Ensiksi syötteenä ovat suuntaamattomat verkot. Toiseksi algoritmit hyödyntävät verkon solmujen hierarkkista ryhmittelyä. Mukana on myös algoritmeja, jotka tuottavat hierarkkisen ryhmittelyn. Kolmanneksi algoritmit tuottavat kaksiulotteisen, kiinteän kuvaesityksen. Rajaukseen liittyen myös suunnattuja verkkoja voidaan ryhmitellä hierarkkisesti (Schaeffer, 2007, s. 41–47). Brockenauer ja Cornelsen (2001, s. 210–215) mainitsevat erään tavan piirtää ryhmiteltyjä, suunnattuja verkkoja. Jos suunnattu verkko on ryhmitelty hierarkkisesti, myös suuntaamattomien verkkojen hierarkkiseen ryhmittelyyn perustuvia piirtoalgoritmeja voidaan periaatteessa soveltaa jättämällä kaarien suuntaus huomioimatta. Suunnattujen verkkojen ryhmittely on tässä kandidaatintyössä jätetty käsittelemättä aiheen laajuuden takia.

Tutkimuksen aineistona ovat tieteelliset julkaisut, joissa esitellään verkonpiirtoalgoritmeja yksityiskohtaisesti. Algoritmeja vertaillaan tutkimusaineiston pohjalta seuraavien ominaisuuksien mukaan: asymptoottinen tehokkuus, syöte, informaation pelkistäminen ja algoritmin kyky pelkistää verkkoa. Lisäaineistona ovat tunnetuimpien verkonpiirto-ohjelmien käyttöohjeet.

Tässä työssä ei testata algoritmien suoritusta kokeellisesti. Myöskään käytettävyyttä ei tutkita muuten kuin päättelemällä algoritmien ilmoitetuista ominaisuuksista.

Työn tuloksena havaitaan, että hierarkkiseen ryhmittelyyn perustuvia tehokkaita verkonpiirtoalgoritmeja on kehitetty, mutta kunnollista toteutusta ei ole tunnetuissa verkonpiirto-ohjelmissa. Tunnetut verkonpiirto-ohjelmat pystyvät näyttämään tehokkaasti suuren määrän solmuja ja kaaria, ja solmut pystytään jakamaan ryhmiin. Kuitenkin hierarkian luominen on käyttäjän tehtävä, ja hierarkiatasojen välillä navigoiminen on työlästä käyttäjälle.

Tämän kandidaatintyön rakenne on seuraava. Luvussa 2 esitetään taustatietoa, joka liittyy olennaisesti verkkojen piirtämiseen. Siinä määritellään verkko ja verkon hierarkkinen ryhmittely. Lisäksi esitellään lyhyesti verkkojen piirtämisen ongelmaa ja ratkaisumenetelmien jaottelua. Luvussa 3 esitellään tutkittavat piirtomenetelmät yleisesti, ja niitä vertaillaan ominaisuuksiensa mukaan. Luvussa 4 otetaan toiseksi vertailukohteeksi yleisesti tunnetut verkonpiirto-ohjelmistot, ja selvitetään, onko niissä toteutettu verkon piirtämistä hierarkkisella ryhmittelyllä. Luvussa 5 esitetään johtopäätöksiä ja hypoteeseja piirtomenetelmien ominaisuuksista ja soveltuvuudesta.

2 Verkkojen piirtäminen

Tässä luvussa esitellään matemaattinen rakenne nimeltä verkko, sekä verkon kaksi esitystapaa: ihmiselle ja tietokoneelle sopiva. Tämän jälkeen esitellään yleisesti verkkojen piirtämisen ongelmaa ja piirtomenetelmien luokittelua. Lopuksi määritellään verkon hierarkkinen ryhmittely, joka on tämän kandidaatintyön lähestymistapa suurten verkkojen piirtämisen ongelmaan.

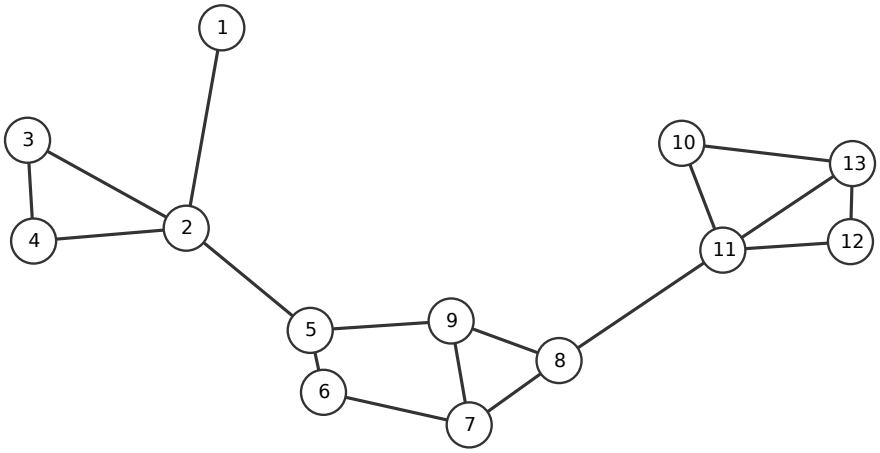
2.1 Verkko matemaattisena rakenteena

Verkko on matemaattinen käsite, jolla esitetään elementtien välisiä suhteita (Sugiyama, 2002, s. 3). Biggs (2002, s. 158) määrittelee verkon seuraavasti:

> *Verkko* G (graph) koostuu äärellisestä joukosta *solmuja*, V (vertices), ja joukosta *kaaria*, E (edges). Jokainen kaari on kahden alkion osajoukko V:stä. Tavallisesti kirjoitetaan $G = (V, E)$.

Toisin sanoen verkossa on solmuja, ja jokaisen solmun välillä joko on tai ei ole yhteys.

Ihminen ymmärtää parhaiten verkon *kuvaesitystä*: solmut ovat pisteitä ja kaaret viivoja pisteiden välillä (Biggs, 2002, s. 158; Sugiyama, 2002, s. 3). Esimerkki verkon kuvaesityksestä on kuvassa 1.



Kuva 1: Esimerkki verkon kuvaesityksestä. Solmut on yksilöity numeroin 1–13.

Tietokoneet käsittelevät verkkoja listoina tai taulukoina. *Vieruslista* tai *naapurilista* (adjacency list, neighbour list) on verkon esitys, jossa jokaisen solmun kohdalla listataan, mihin muihin solmuihin tällä solmulla on yhteys. Toinen esitystapa on *vierusmatriisi* (adjacency matrix): taulukko, jossa kukin solmu on sekä rivinä että sarakkeena. Jos solmujen

välillä on yhteys, taulukon soluun merkitään arvo 1. Muuten taulukkoon merkitään arvo 0. (Biggs, 2002, s. 159; Cormen ym., 2001, s. 527–528) Kuvassa 2 on kuvan 1 verkko naapurilistana. Matriisiesityksestä on esimerkki kuvan 8 keskellä.

Solmu	Naapurit	Solmu	Naapurit
1	2	8	7, 9, 11
2	1, 3, 4, 5	9	5, 7, 8
3	2, 4	10	11, 13
4	2, 3	11	8, 10, 12, 13
5	2, 6, 9	12	11, 13
6	5, 7	13	10, 11, 12
7	6, 8, 9		

Kuva 2: Esimerkki naapurilistaesityksestä. Verkko on sama kuin kuvassa 1.

Lista- ja matriisiesityksen toinen ero verrattuna kuvaesityksen on siinä, että kuvaesitys ei ole yksikäsitteinen. Ensiksikin solmujen sijainnin voi valita vapaasti kaksiulotteisella kuvatasolla, siis tietokoneen näytöllä tai paperilla. Toiseksi kaaret voi esittää suorien viivojen sijasta myös käyrinä. Tästä johtuen yhdellä verkolla on äärettömän monta mahdollista kuvaesitystä.

Verkoilla voi olla lisämääritelmiä käyttötarkoituksesta riippuen. Verkko voi olla *suunnattu* (directed graph, digraph), jolloin kaaret ovat järjestettyjä pareja. *Painotettu verkko* (weighted graph) tarkoittaa, että kullakin kaarella on jokin lukuarvo, *paino*. (Biggs, 2002, s. 225–227) Käytännön esimerkkejä suunnatusta verkosta ovat tieverkko, jossa on yksisuuntaisia katuja, ja talonrakennusprojekti, jossa myöhemmät työvaiheet riippuvat edellisten vaiheiden valmistumisesta. Kaarien paino voi tarkoittaa esimerkiksi etäisyyttä kahden kaupungin välillä tai suurinta mahdollista tiedonsiirtonopeutta tietoverkossa. Verkko voi olla sekä suunnattu että painotettu.

2.2 Verkkojen piirtäminen automaattisesti

Verkon kuvaesityksen piirtäminen on ihmiselle työlästä, ellei verkko ole pieni (Sugiyama, 2002, s. 3). Verkkomuotoinen informaatio voi olla hyvin monimutkaista. Biologisesta datasta esimerkkinä ovat erilaiset proteiinit, joita on ihmisten ja muiden eläinten soluissa. Ihmisen proteiineja tunnetaan 18 255, ja proteiinien välillä on 144 643 vuorovaikutussuhdetta (Biogrid, 2014). Kyseessä on siis verkko, jossa on tuhansia kaaria ja solmuja.

Verkkojen graafisia esityksiä voi luoda tietokoneella täysin automaattisesti. Tarkoitukseen on kehitetty monia menetelmiä, *verkonpiirtoalgoritmeja* (graph drawing algorithm). Näiden algoritmien tehtävänä on määrätä verkon solmuille *asettelu* (layout), siis sijoittelu tasoon. Tavoitteena on löytää asettelu, jonka lopputuloksena saatava esitys on ihmisen

kannalta mahdollisimman ymmärrettävä.

Jotta tietokone saadaan tuottamaan verkosta ihmisen kannalta mahdollisimman ymmärrettävä, siis selkeä tai kaunis kuva, on määriteltävä täsmälliset tavoitteet, joihin verkonpiirtoalgoritmin laskenta pyrkii (Sugiyama, 2002, s. 11). Tavoitteita kutsutaan *esteettisiksi kriteereiksi*. Kriteereitä on useita, ja ne sopivat erilaisiin käyttötarkoituksiin. (Fleischer ja Hirsch, 2001, s. 18–20)

Tarkastellaan esteettisiä kriteereitä tapauksessa, jossa on kuvan 1 kaltainen suuntaamaton, painottamaton verkko. Sugiyama (2002, s. 11–14 ja 42) sekä Fleischer ja Hirsch (2001, s. 19–20) mainitsevat muun muassa seuraavat tavoitteet:

1. *Symmetrian näyttäminen.* Teknisissä piirroksissa on usein piileviä symmetrioita, jotka on hyvä saada näkyviin.

2. *Risteyksien välttäminen.* Mitä vähemmän verkkokaaviossa on risteäviä viivoja, sen helpompaa ihmisen on ymmärtää kaaviota.

3. *Risteyksien kulman maksimointi.* Jos kaaret risteävät, ne risteävät mahdollisimman suorassa kulmassa toisiinsa nähden.

4. *Piirtoalueen koon minimoiminen.* Jos solmut asetellaan mahdollisimman tasaisesti koko piirtoalueelle, kuva näyttää paremmalta. Jos verkko on esimerkiksi paperinen joukkoliikennekartta, on tärkeää, että kartta on käytännöllisen kokoinen.

5. *Ryhmittely.* Monessa tapauksessa solmut on ryhmiteltävä, jotta verkon rakenne saadaan näkyviin.

Esteettisten kriteerien täyttäminen on vaikea ongelma monesta syystä. Ensiksikin kriteerien tuottamat optimointiongelmat ovat yleisesti ottaen NP-kovia, eli absoluuttisesti parhaan lopputuloksen laskeminen on liian hidasta nopeimillakin tietokoneilla. Toiseksi esteettiset kriteerit ovat ristiriidassa keskenään, joten niille pitää antaa tärkeysjärjestys. Samasta verkosta saa hyvin erilaisia asetteluita riippuen siitä, mitä kriteereitä painotetaan. Käytännössä esteettisten kriteerien täyttämisessä käytetään approksimaatiota tai heuristiikkoja, siis menetelmiä, jotka tuottavat kohtuullisessa ajassa riittävän hyvän tuloksen. Eri käyttötarkoituksiin on eri kriteereitä, ja kriteerien käyttökelpoisuus riippuu piirrettävän verkon rakenteesta. (Sugiyama, 2002, s. 42; Fleischer ja Hirsch, 2001, s. 19–20)

Sugiyama (2002, s. 17) esittää viisi pääluokkaa verkonpiirtoalgoritmeille:

1. Algoritmit, jotka hyödyntävät verkkoteoriaa ja verkkoalgoritmeja.

2. Heuristiset algoritmit.

3. Voimaohjatut algoritmit.

4. Hybridialgoritmit, jotka yhdistävät menetelmätyyppejä 1–3.

5. Tekoälyalgoritmit kuten ”layout by example”.

Luokkaan 1 kuuluvat ensimmäisenä *tasoverkon* piirtävät algoritmit. Jotkin verkot voidaan piirtää ilman risteäviä viivoja. Tällaista verkkoa kutsutaan *tasoverkoksi*. On olemassa algoritmeja verkon tunnistamiseksi tasoverkoksi sekä algoritmeja tasoverkon piirtämiseksi. (Weiskircher, 2001) Joskus verkko voidaan muuttaa tasoverkoksi poistamalla muutama kaari. Näin saadaan selkeä perusasettelu, jonka jälkeen voidaan lisätä puuttuvat kaaret. Sitten verkko täydennetään ja kaaret asetetaan kulkemaan ortogonaalisesti pitkin neliöhilaa. Esitys on selkeä, mutta verkon pitää olla melko harva ja suhteellisen pieni. (Sugiyama, 2002, s. 81–87) Kuvassa 8 käytetään muun muassa ortogonaalia kaaria.

Heuristiikka on laskentamenetelmä, joka tuottaa nopeasti riittävän tarkan tuloksen verrattuna täydelliseen ratkaisuun. Sugiyama (2002, s. 30, 37) mainitsee heuristisina menetelminä massakeskipistemenetelmän (barycentric method), joka vähentää kaarien risteyksien määrää sekä syvyyshaun (depth first search), jota voi käyttää apuna suunnattuja verkkoja piirtäessä. Suunnattuja verkkoja voidaan piirtää järjestämällä solmut vaakasuorille viivoille kerroksiksi juuri heuristiikkoja hyödyntäen. Suunnatun verkon kerrosasettelusta on esimerkki kuvassa 3. Ideana on järjestää solmut siten, että suunnattuja kaaria esittävät nuolet osoittavat enimmäkseen ylhäältä alas, jolloin verkossa olevat samansuuntaiset polut paljastuvat. (Bastert ja Matuszewski, 2001; Sugiyama, 2002, s. 27–30 ja 77–80)

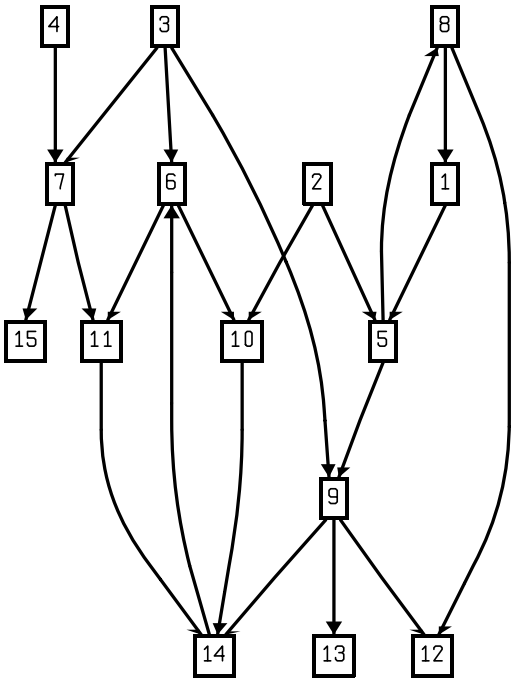
Voimaohjatut algoritmit käyttävät fysikaalista mallinnusta. Kaaria voidaan esimerkiksi mallintaa jousina, jotka vetävät ja työntävät verkon vähitellen muotoon, jossa jousien kokonaisenergia on pienin. Lisäksi kaikki solmut hylkivät toisiaan kuin samanmerkkiset sähkövaraukset. (Brandes, 2001; Sugiyama, 2002, 87–91) Toisaalta tällainen jousialgoritmi (spring embedder) on myös heuristinen (Sugiyama, 2002, s. 98). Sugiyaman luokittelu ei siis ole rajauksissaan täysin ehdoton.

Landgrafin (2001, s. 172) mukaan yleisimmät kaksiulotteisten verkonpiirtoalgoritmien luokat ovat fyysiset mallinnukset, kerrostaminen ja ortogonaalinen piirtäminen.

2.3 Verkon hierarkkinen ryhmittely

Verkon hierarkkinen ryhmittely (hierarchical clustering of a graph) tarkoittaa solmujen jaottelua ryhmiin puumaisesti, kuten Brockenauer ja Cornelsen (2001, s. 195–196) määrittelevät:

> ”Hierarkkisesti ryhmitelty verkko $C = (G, T)$ koostuu verkosta $G = (V, E)$ ja juurellisesta puusta T siten, että T:n lehdet ovat täsmälleen V. Jokainen



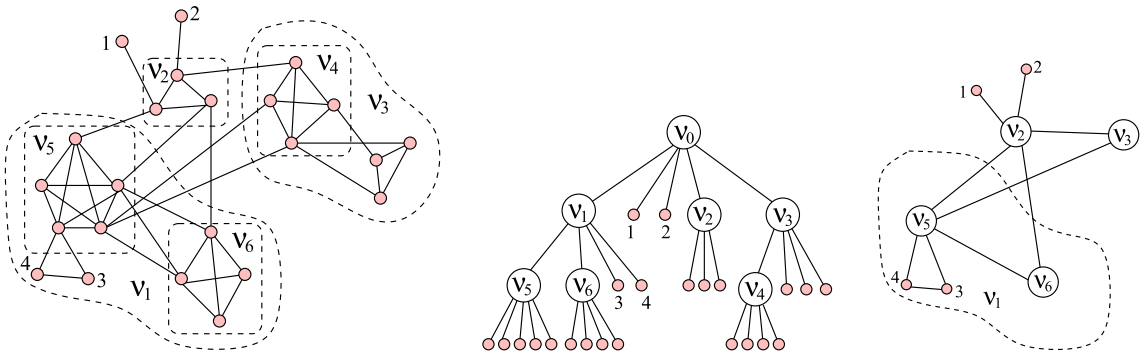
Kuva 3: Suunnattu verkko piirrettynä kerroksittain. Ylimmällä kerroksella ovat solmut 4, 3 ja 8. Seuraavalla kerroksella ovat solmut 7, 6, 2 ja 1. Kuva on artikkelista Bastert ja Matuszewski (2001).

> solmu ν puussa T esittää *ryhmää* $V(\nu)$ lehtiä siinä T:n osapuussa, jonka juuri on ν. T:tä kutsutaan C:n *ryhmittelypuuksi* (inclusion tree, hierarchy tree). Kaaren e sanotaan *liittyvän* ryhmään $V(\nu)$, jos $|e \cap V(\nu)| = 1$. " (Suomennos tekijän)

Toisin sanoen hierarkkisessa ryhmittelyssä on suuria ryhmiä, jotka sisältävät pienempiä osaryhmiä. Pääryhmässä on kaikki verkon solmut, ja tämä pääryhmä on ryhmittelypuun juuri. Puun alimmalla tasolla jokainen verkon solmu on oma ryhmänsä.

Kuvassa 4 on esimerkki verkon hierarkkisesta ryhmittelystä. Ryhmä ν_0 sisältää koko verkon. Ryhmä ν_1 sisältää ryhmät ν_5 ja ν_6 sekä solmut 3 ja 4. *Ositusverkko* (quotient graph) tarkoittaa verkkoa, joka syntyy, kun hierarkkisesti ryhmitellyn verkon C ryhmät pelkistetään solmuiksi.

Mille tahansa verkolle voidaan tuottaa hierarkkinen ryhmittely, sillä solmujen joukko voidaan jakaa osajoukkoihin ja osajoukkojen osajoukkoihin mielivaltaisesti. Verkkojen ryhmittelyalgoritmeja on useita erilaisia, ja ne tuottavat erilaisia ryhmittelyjä samalle verkolle (Brockenauer ja Cornelsen, 2001, s. 197–202). Eräs perusperiaate on laskea ensin jokaisen solmun välinen yhteys jollain tavalla painotetusti, ja sitten yhdistellä solmuja ryhmiksi yksi kaari kerrallaan painotusten mukaan. Näin syntyy ensin pieniä ryhmiä, jotka sitten yhdistyvät suuremmiksi. (Girvan ja Newman, 2002, s. 7821–7822)



(a) G (b) T (c) Q

Kuva 4: (a) verkko G, jossa on hierarkkiset ryhmät $\nu_1 \ldots \nu_6$. (b) Verkon G ryhmittelypuu T. (c) Ositusverkko Q, jossa ryhmät ν_2, ν_3, ν_5 ja ν_6 on pelkistetty omiksi solmuikseen. Kuva on artikkelista Didimo ja Montecchiani (2014).

3 Hierarkkiseen ryhmittelyyn perustuvat verkonpiirtomenetelmät

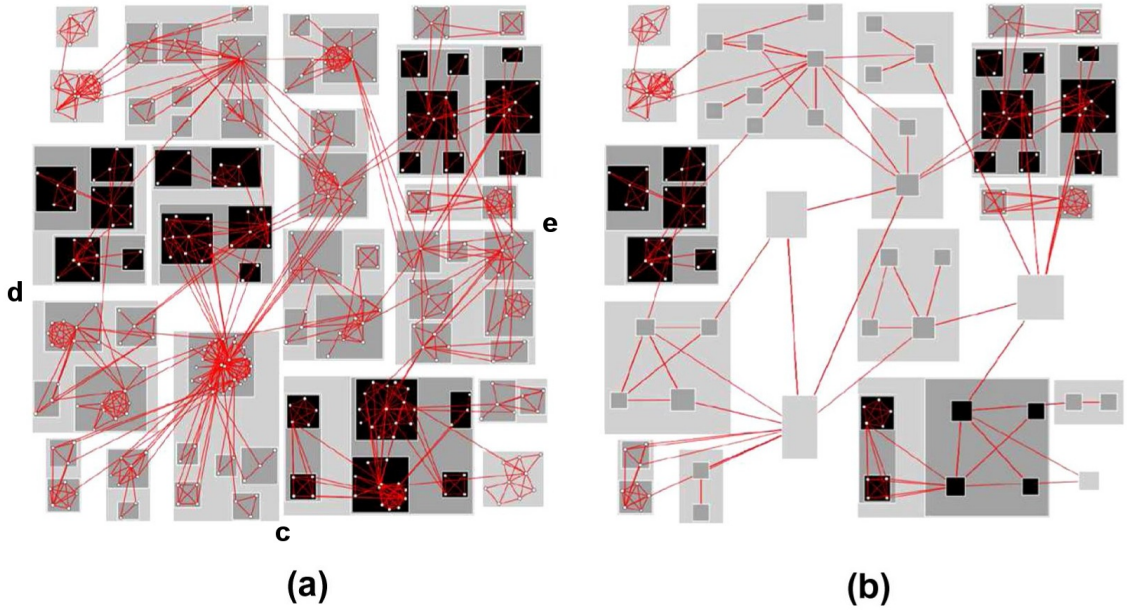
Tässä luvussa esitellään tutkimuskohteiden keskeiset ominaisuudet. Jokaisesta piirtomenetelmästä kerrotaan tiivistäen piirtomenetelmää käsittelevän tieteellisen julkaisun tietoja.

3.1 Algoritmien ja ohjelmien yleispiirteet

Didimon ja Montecchianin (2014) -algoritmi jakaa suorakulmaista piirtoaluetta osiin Treemap-periaatteella: suorakulmioita jaetaan rekursiivisesti pysty- ja vaakasuoraan. Hierarkkiset ryhmät ovat Treemap-jaon tuottamia suorakulmioita, joiden sisälle solmut on aseteltu Hachulin ja Jüngnerin (2005) kehittämällä voimaohjatulla FM^3-algoritmilla. Sisältyvyyspuun ei tarvitse olla binääripuu. Treemap-algoritmin on kehittänyt Shneiderman (1991). Didimon ja Montecchianin algoritmin esimerkkitulos on kuvassa 5.

Abello ym. (2006) ovat kehittäneet ohjelmiston *ASK-GraphView*, joka soveltaa asiakas–palvelinarkkitehtuuria ja piirtää osaverkkoja nopeasti ja vuorovaikutteisesti. Palvelimen kiintolevylle on tallennettu koko verkko, joka on ryhmitellään aluksi automaattisesti, karkeasti ja hierarkkisesti. Asiakasohjelma piirtää tietokoneen näytölle verkkonäkymän, jossa esitettävien solmujen ja kaarien määrää on rajoitettu. Asiakasohjelma myös tekee hienojakoisemman ryhmittelyn ja asettelun. ASK-GraphViewin käyttöliittymä on kuvassa 6.

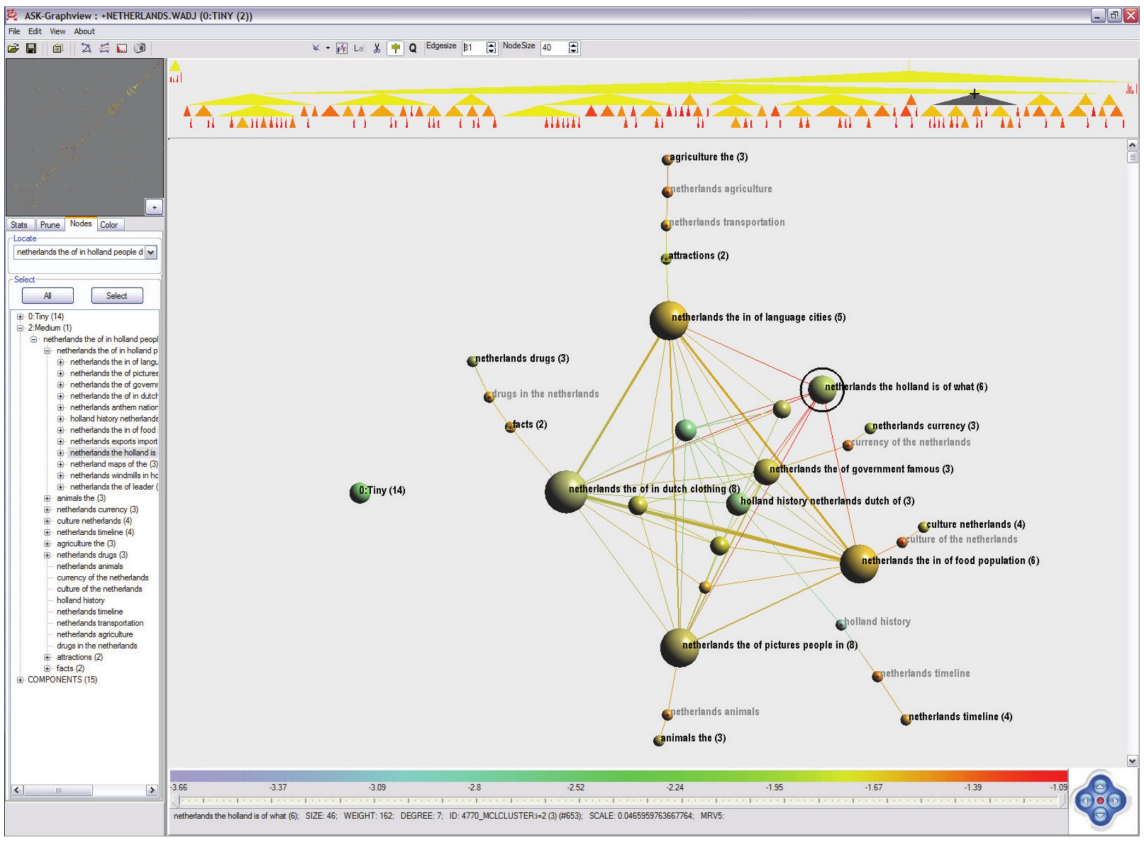
Archambault ym. (2008) esittelevät GrouseFlocks-ohjelmiston, jolla voi tutkia suuria verkkoja vuorovaikutteisesti kuten ASK-GraphViewissä. GrouseFlocksissa käyttäjä voi luoda erilaisia hierarkioita automaattisella avustuksella käyttäen ryhmittelyperusteena



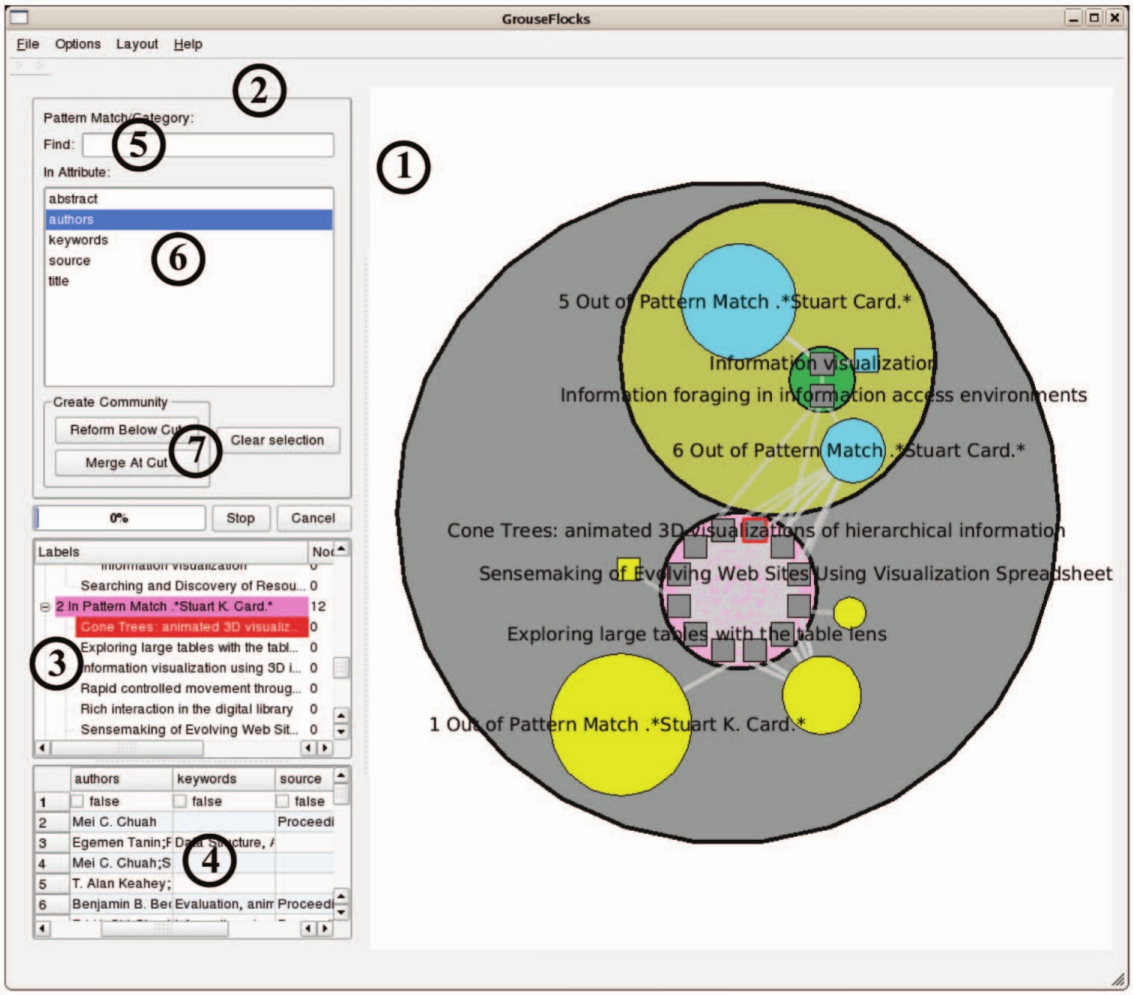
Kuva 5: Esimerkki Didimon ja Montecchianin piirtoalgoritmin tuloksesta. Kuvassa (a) on koko verkko, ja kuvassa (b) ryhmiä on pelkistetty. Kuvassa (a) neliönmuotoinen piirtoalue on ensin jaettu pystysuoraan kahtia keskeltä kohdasta c. Syntyvät kaksi suorakulmiota on edelleen jaettu vaakasuoraan kohdista d ja e. Kuva on artikkelista Didimo ja Montecchiani (2014), mutta siihen on lisätty jakokohdat c–e.

solmuihin liitettyjä arvoja (attribute). Kuva 7 selventää asiaa.

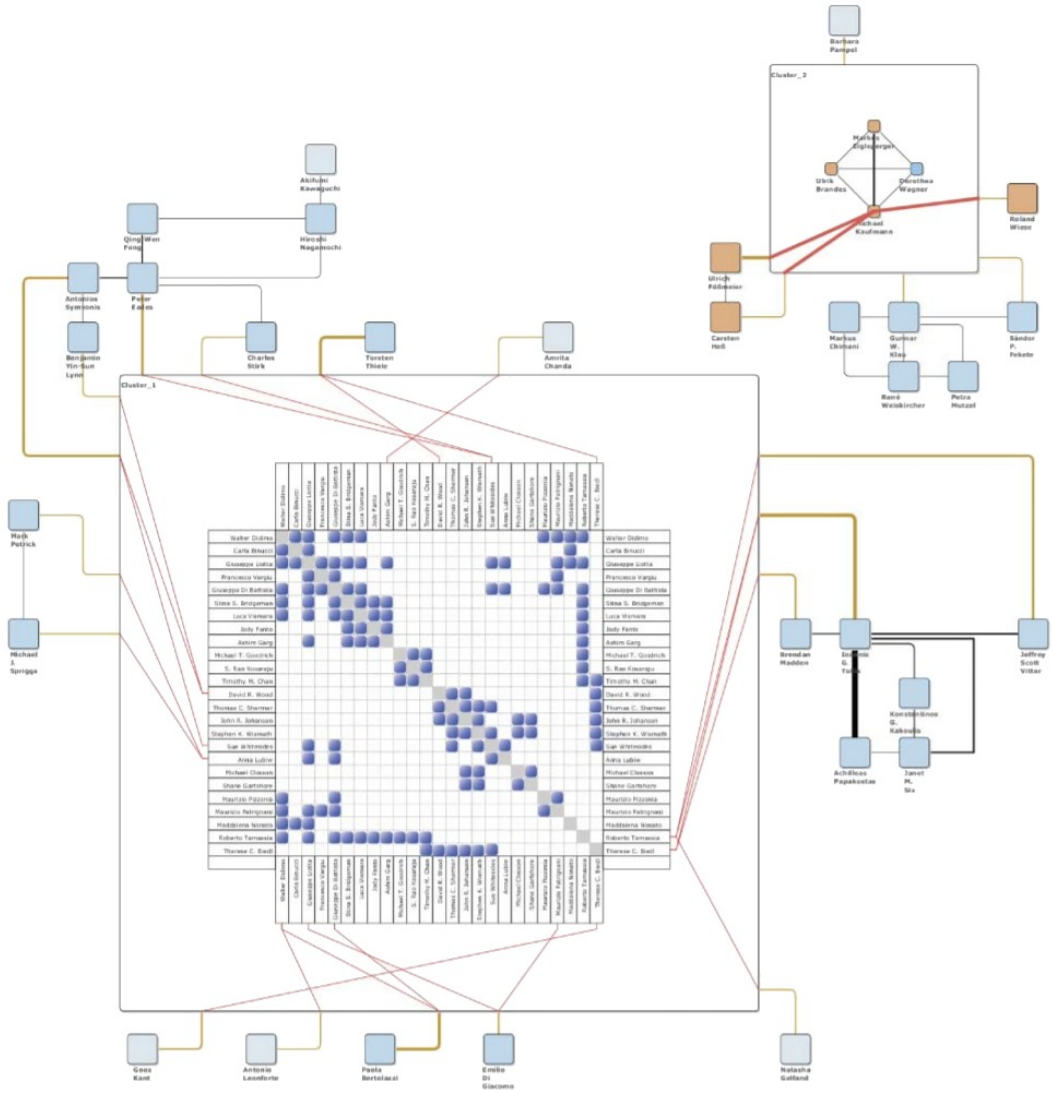
Batagelj ym. (2011) esittelevät kaksi käyttökelpoista (polynomiaikaista) ryhmittelyalgoritmia ja hybridipiirtomenetelmiä hyödyntävän ohjelmiston VHyXY, joka on kuvassa 8. Mukana on vertailun vuoksi myös yksi NP-kova ryhmittelyalgoritmi. Ryhmittely tehdään kahdella ehdolla: ensiksi on vaatimus kunkin ryhmän osaverkon topologiasta, ja toiseksi on vaatimus ositusverkon topologiasta. Ensimmäinen ryhmittelyalgoritmeista tuottaa tasomaisen, k-ydinkomponenttinen ryhmittelyn. Verkon G k-ydin (k-core) on osaverkko, johon kuuluvat kaikki ne solmut, joiden aste on vähintään k. K-ytiminen komponentti (k-core component) on yksi k-ytimen yhtenäinen osaverkko. Tekijät esittelevät algoritmin, joka etsii k-ydinkomponentit, tuottaa jokaisesta komponentista oman ryhmänsä ja lisäksi tasomaisen ositusverkon. Verkon rakenteesta riippuu, mikä on pienin k:n arvo, jolla ositusverkosta saadaan tasomainen. Toinen Batageljin et al. ryhmittelyalgoritmi on d-tiheyksinen ja k-ydinkomponenttinen. Se on samankaltainen kuin edellämainittu algoritmi, mutta parametri d on osaverkkojen tiheys eli kaarien määrä suhteessa solmujen määrään. Tämä ryhmittely estää liian suurien tai tiheiden ryhmien muodostumisen. Ryhmittelyalgoritmeja voidaan käyttää hierarkkiseen ryhmittelyyn rekursiivisesti.



Kuva 6: ASK-GraphViewin käyttöliittymä. Esillä on 489 solmun verkko, joka esittää tietoa Alankomaista. Vasemmalla on puumainen lista hierarkkisesta ryhmittelystä. Ikkunan yläreunassa on hierarkiapuu, jossa tämänhetkinen verkkonäkymä on merkitty tummanharmaana kolmiona. Ruudun alareunassa on liukusäädin, jolla käyttäjä voi hallita solmujen suodattamista halutun attribuutin mukaan. Kuva on artikkelista Abello ym. (2006).



Kuva 7: GrouseFlocksin käyttöliittymä. Verkkonäkymä (1) on tuotettu hierarkiapuun poikkileikkauksen mukaan. Hakutoiminnolla (2) voi valita osan verkosta. Puunäkymä (3) näyttää hierarkiapuun ja nykyisen verkkonäkymän sijainnin puussa. Attribuuttilistasta (4) käyttäjä voi valita, mitkä solmujen attribuutit näytetään verkkonäkymässä. Hakukenttään (5) käyttäjä voi antaa säännöllisen lausekkeen, jonka mukaista tekstiä etsitään valitusta attribuutista (6). Painikkeilla (7) muokataan ryhmittelyä. Kuva on artikkelista Archambault ym. (2008).



Kuva 8: VHyXY-ohjelmiston tuottama verkkonäkymä. Pääverkon ryhmät näkyvät vaaleansinisinä, pyöristettyinä neliöinä. Näitä ryhmiä yhdistävät kaaret ovat ortogonaalisia, siis mutkittelevat vaaka- ja pystysuoraan. Nämä kaaret eivät myöskään risteä. Kaksi ryhmäsolmua on laajennettu: ne näkyvät suurina neliöinä, joiden sisällä ovat ryhmän solmut. Suurempi ryhmä on matriisiesityksenä kuvan keskellä: on neliöruudukko, jossa jotkin ruudut on väritetty sinisellä. Vaaka- ja pystyriveillä ovat ryhmän solmut, joita on yhteensä 25. Lisäksi matriisin reunoista on merkitty punaisilla viivoilla ryhmän solmujen yhteydet naapuriryhmiin. Toinen ryhmä on oikeassa yläkulmassa. Siinä on neljä solmua oletettavasti voimaohjatusti aseteltuna kuvaesityksenä. Kuva on artikkelista Batagelj ym. (2011).

3.2 Syöte

Algoritmin *syöte* vaikuttaa algoritmin käyttökelpoisuuteen. Tavoitteena on tutkia, sopiiko algoritmi tarvittaessa painotetulle verkolle, tai vaatiiko se valmiin hierarkkisen ryhmittelyn. Jos piirtoalgoritmi ei sisällä hierarkkista ryhmittelytoimintoa, periaatteessa mitä tahansa hierarkkista ryhmittelyalgoritmia voi käyttää piirtoalgoritmin kanssa.

Didimon ja Montecchianin (2014) algoritmin syötteenä on hierarkkisesti ryhmitelty verkko ja suorakulmaisen piirtoalueen sivujen suhde. Solmujen painot vaikuttavat asetteluun, mutta kaarien painot eivät.

Kirjoittaessaan ASK-GraphViewin resurssien käytöstä tekijät toteavat, että heidän järjestelmänsä ei aseta vaatimuksia verkon rakenteelle. Kuitenkin verkon pitää olla painotettu, koska painotusta hyödynnetään hierarkkisen ryhmittelyn tuottamisessa. (Abello, 2004, s. 350; Abello ym., 2006, s. 669 ja 671)

GrouseFlocks vaatii painottamattoman verkon suurten osaverkkojen pelkistämisen (coarsening) takia. Verkon hierarkian voi antaa lisäsyötteenä. Archambault ym. (2008, s. 901, 907)

Batageljin ym. (2011) ryhmittelyalgoritmit eivät aseta vaatimuksia verkon rakenteelle, mutta eivät myöskään hyödynnä painotusta tai attribuutteja ryhmittelyssä. VHyXY voi hyödyntää piirtämisessä solmujen ja kaarien painoja: käyttäjä voi suodattaa verkon osia pois näkymästä verkon painoihin perustuen. (Batagelj ym., 2011, s. 1590–1592 ja 1595)

3.3 Asymptoottinen tehokkuus

Asymptoottinen tehokkuus (asymptotic efficiency) ajan suhteen on tekninen ominaisuus, joka kuvaa karkeasti algoritmin tarvitsemaa laskenta-aikaa suurilla syötteillä. Asymptoottinen tehokkuus ilmoitetaan tässä työssä iso-O-merkinnällä. (Cormen ym., 2001, s. 41–56) Verkonpiirtoalgoritmeissa laskenta-aikaan vaikuttavat solmujen ja kaarien määrä ja joskus verkon rakenne. Jos laskenta-aika kasvaa voimakkaasti suhteessa syötteeseen, tiettyä kokoa suuremmilla verkoilla algoritmi on käyttökelvoton, koska käyttäjä joutuu odottamaan liian kauan tuloksen valmistumista.

Didimon ja Montecchianin (2014) algoritmi on tehokkuudeltaan $O(n \log n + m)$, missä n on solmujen ja m kaarien määrä. Algoritmi sopii rinnakkaislaskentaan.

ASK-GraphViewin ideana on taata, että solmujen ja kaarien määrältä rajoitettu verkon osanäkymä saadaan näkyviin esimerkiksi muutamassa sekunnissa. Hierarkiapuun tuottaminen palvelimella tehdään muunnetulla Boruvkan supistamisalgoritmilla (Boruvka's contraction algorithm). Tämä algoritmi vaatii datan läpikäymistä $O(\log(n/r))$ kertaa, missä r on käytettävissä olevan työmuistin (RAM) määrä. (Abello, 2004; Abello ym.,

2006, s. 670–671) Esikäsittelyvaihe on siis logaritminen, mutta sen jälkeen verkon osanäkymien piirtäminen saadaan tehtyä vakioajassa sopivista rajoituksista johtuen.

GrouseFlocks käyttää osaverkkojen piirtämiseen eri algoritmeja riippuen osaverkon topologiasta: puut piirretään ajassa $O(n)$ (Buchheim ym., 2002, s. 350; Grivet ym., 2006), kalaverkkomaiset verkot (mesh) ajassa $O(n + m)$ (Harel ja Koren, 2002, s. 207), ja täydelliset verkot (complete graph) ympyräasettelulla (circular layout). Tarvittaessa käytetään voimaohjattua algoritmia, joka on arviolta $O(m^3)$ (Frick ym., 1995, s. 397). Solmujen päällekkäisyydet poistetaan algoritmilla, joka on $O(n \log n)$ (Dwyer ym., 2006, s. 154). Jos käyttäjä yrittää näyttää sellaisen osaverkon, joka on liian suuri, GrouseFlocks tekee verkkonäkymästä karkeamman (coarsen) ajassa $O(n \log^2 n + m \log n)$. Erityisesti GrouseFlocks piirtää verkkoa tarpeen mukaan, joten ASK-GraphViewin tapaista koko verkon alkuanalyysia ei tarvita. (Archambault ym., 2008, s. 905–908, 912) Tehokkuuksia verratessa on huomioitava, että karkeistusvaiheen jälkeen osaverkko sisältää olennaisesti vähemmän solmuja ja kaaria. Näin ollen voidaan olettaa, että suurille osaverkoille laskenta-aikaan vaikuttaa eniten karkeistamisvaihe, joten yleistetään pahimmaksi tapaukseksi $O(n \log^2 n + m \log n)$.

Batagelj ym. (2011) ryhmittelyalgoritmeista tasomainen, k-ydinkomponenttinen versio vaatii ajan $O((n + m) \log \Delta)$, missä Δ on suurin asteluku verkon solmuille, siis suurin määrä kaaria solmussa. D-tiheyksisen ja k-ydinkomponenttisen ryhmittelyn tuottava algoritmi on hitaampi: $O((n + m)\Delta)$. Tekijät huomauttavat, että usein kuitenkin $\Delta < n$. Artikkelissa esitellään myös k-klikkisen (k-clique) versio. Verkon *klikki* (clique) tarkoittaa osaverkkoa, jossa kaikki solmut ovat yhteydessä toisiinsa. Kuitenkin tekijät toteavat k-klikkisen version olevan NP-kova, siis käyttökelvoton suurille verkoille. (Batagelj ym., 2011, s. 1592) VHyXY-ohjelmiston ositusverkko piirretään algoritmilla, jonka ovat kehittäneet Battista ym. (1999), ja jonka asymptoottista tehokkuutta ei ole mainittu (Batagelj ym., 2011, s. 1593).

3.4 Pelkistäminen

Informaation pelkistäminen on olennaista, koska ihminen pystyy ymmärtämään vain rajallisen monimutkaista verkkoa. Shneiderman (1996) esittää, että informaation kuvaamisessa on paras esittää ensin yleisnäkymä, ja sitten antaa käyttäjän valita jokin osa kuvasta, suodattaa pois epäkiinnostava osa informaatiosta ja lopuksi näyttää yksityiskohtia tarpeen mukaan. Verkkojen piirtämisessä informaation pelkistäminen tarkoittaa esimerkiksi solmuryhmän näyttämistä yhtenä solmuna tai kahta solmuryhmää yhdistävien kaarien pelkistämistä yhdeksi kaareksi.

Didimon ja Montecchianin (2014) menetelmässä lasketaan ensin ositusverkot, minkä takia jotkin verkon osat voi piirtää pelkistettynä. Asettelun valmistuttua ryhmien välisiä kaaria

voidaan erillisellä algoritmilla *niputtaa* (edge bundling): jos kahden ryhmän välillä on useita kaaria, nämä kaaret voidaan piirtää yhtenä viivana, joka haarautuu viuhkamaisesti vasta lähellä kumpaakin ryhmää.

ASK-GraphViewistä kertovassa artikkelissa on määritelmä *antiketju* (antichain), joka vastaa Didimon ja Montecchianin (2014, s. 670) määritelmää ositusverkosta. ASK-GraphView käyttää hierarkkista ryhmittelyä ja abstrakteja solmuja: käyttäjä voi valita verkkonäkymästä yhden solmun ja tarkentaa näkymää siten, että näytetään valitun solmun sisältämä osaverkko. Edelleen tämä osaverkko voi sisältää abstrakteja solmuja, joita voi laajentaa. Verkkonäkymästä voi suodattaa pois solmuja ja kaaria. (Abello ym., 2006)

GrouseFlocksissa on samanlainen vuorovaikutteinen, pelkistetty verkkonäkymä kuin ASK-GraphViewissä. Suuren osaverkon pelkistäminen (coarsening) jättää osaverkon suuren mittakaavan rakenteen ja karsii pienen mittakaavan rakennetta. (Archambault ym., 2008, s. 904–905, 907)

VHyXY-ohjelmiston ryhmittelyalgoritmit tuottavat tasomaisen ositusverkon, joka piirretään ortogonaalisesti: kaaret koostuvat vaaka- ja pystysuorista janoista, jotka eivät risteä. K-ytimiset ryhmät näkyvät ositusverkossa suorakulmion sisässä, ja ne voidaan piirtää ympyräasettelulla (circular layout), voimaohjatulla asettelulla tai vierusmatriisina. Ryhmät voi pelkistää yhdeksi solmuksi. (Batagelj ym., 2011, s. 1592–1595)

3.5 Vakaus

Kuvaesityksen vakaudella on merkitystä käytettävyyden kannalta. Verkko on *dynaaminen*, jos sen rakenne muuttuu ajan suhteen. Olennaista on, muuttuuko kuvaesitys merkittävästi, jos verkkoon lisää tai siitä poistaa yhden solmun tai kaaren. Tyypillisesti verkonpiirto-ohjelmat sallivat verkon muokkaamisen. Jos käyttäjä on jo ehtinyt tottua yhteen näkymään verkosta, muuttuneeseen näkymään tottuminen voi vaatia käyttäjältä merkittävää työtä. On toivottavaa, että verkon kuvaesitys muuttuu vain siltä osin kuin verkon rakenne muuttuu. (Branke, 2001, s. 228)

Eades ym. (1991) käyttävät nimitystä *sisäinen kartta* (mental map) kuvaamaan käyttäjän oppimaa verkkokaavion rakennetta ja analysoivat keinoja sisäisen kartan säilyttämiseksi. Sellaiset kaaviomuunnokset, jotka heidän mielestään säilyttävät käyttäjän sisäisen kartan, ovat kaavion osien pysty- ja vaakasuoran järjestyksen säilyttäminen (orthogonal ordering), ryhmien säilyttäminen ja topologian säilyttäminen (jokin asia, joka on suljetun, jatkuvan käyrän sisällä, myös pysyy tämän käyrän sisällä muunnoksessa). Branke (2001, s. 230–235) listaa joitakin täsmällisiä mittoja vakaudelle: muuttumattomien solmujen pysyminen paikallaan kuvaesityksessä, summa jokaisen solmun siirtymästä euklidisena etäisyytenä ja solmujen pysty- ja vaakasuoran järjestyksen säilyminen.

Tässä kandidaatintyössä tutkittavien piirtomenetelmien vakautta eivät niiden tekijät ole ilmoittaneet. Piirtomenetelmien vakauden tutkiminen kokeellisesti tai vakauden todistaminen matemaattisesti on työlästä, joten luotettavia tuloksia vakaudesta tässä kandidaatintyössä ei esitetä. Kuitenkin luvussa 5.1 esitetään havaintoja ja hypoteeseja ryhmittely- ja piirtomenetelmien vakaudesta.

4 Tunnetut verkonpiirto-ohjelmistot

Tässä luvussa esitellään neljä tunnettua ohjelmistoa, joissa on verkonpiirto-ominaisuuksia. Tunnettuudella tarkoitetaan sitä, että ohjelmisto on ollut useamman vuoden saatavilla, sitä kehittää jatkuvasti ryhmä ihmisiä, ja ohjelmistolla on muitakin käyttökohteita kuin verkkojen piirtämisen tutkimus. Tarkoitus on verrata näiden tunnettujen ohjelmistojen ominaisuuksia edellisessä luvussa esiteltyihin piirtomenetelmiin ja kokeellisiin ohjelmistoihin.

Cytoscape on vuorovaikutteinen ohjelmisto verkkojen analysoimiseen ja piirtämiseen. Se on suunniteltu erityisesti molekyylibiologian tarpeisiin. Ohjelmisto kykenee lukemaan useita verkkotiedostomuotoja ja sisältää useita verkonpiirtoalgoritmeja. Käyttäjä voi portaattomasti lähentää ja loitontaa kuvaa verkosta. Solmujen kokoa ja väritystä voidaan muuttaa perustuen automaattiseen analyysiin. Cytoscapen versiossa 3.1.0 ei ole automaattista hierarkkista ryhmittelyä eikä hierarkkista ryhmittelyä hyödyntävää piirtoalgoritmia. Käyttäjän on kuitenkin mahdollista valita joukko solmuja, merkitä ne ryhmäksi, ja pelkistää ryhmä yhdeksi solmuksi tai solmuksi, jonka sisällä on toinen verkko. Myös ryhmä kaaria voidaan niputtaa tiiviiksi ryhmäksi, kunhan kaaret on ensin valittu. (The Cytoscape Consortium, 2014) On ilmeistä, että Cytoscape lukee koko verkkotiedoston keskusmuistiin, mikä rajoittaa käsiteltävien verkkojen kokoa.

Gephi on samantapainen ohjelmisto kuin Cytoscape. Siinä on vakiotoimintona grafiikan laitteistokiihdytys. Käsiteltävän verkon suurimmaksi kooksi on ilmoitettu miljoona solmua ja kaarta. (Bastian ym., 2009; The Gephi consortium, 2013) Gephin versiossa 0.8.2 on MCL-ryhmittelyalgoritmi, joka osaa etsiä ryhmiä. Gephiin saa asennettua myös Chinese Whispers -ryhmittelyliitännäisen (plugin). Käyttäjän pitää itse järjestää ryhmät hierarkkisesti. Kun ryhmittely on tehty, Gephi pystyy näyttämään ryhmittelypuun ja valitun osaverkon. Ryhmittelypuuta ei voi pelkistää.

Mathematica on yleiskäyttöinen, laaja matematiikkaohjelmisto, joka perustuu symboliseen laskentaan. Mathematican versiossa 9 on monia toimintoja verkkojen analysoimiseksi ja piirtämiseksi. Verkkotiedostoja voi tuoda muun muassa GraphML-muodossa. Komento `FindGraphCommunities` etsii verkoista yhteisömäisiä ryhmiä: ryhmien sisällä solmuilla on paljon kaaria toistensa välillä, ja ryhmien välillä on vähän kaaria. Komennolle voi antaa

lisämääreen `Method->"Hierarchical"`, joka tuottaa hierarkkisen ryhmittelyn perustuen solmujen samankaltaisuuteen. Tarkempaa kuvausta algoritmista ei ole. Komento `CommunityGraphPlot` piirtää kuvan, jossa näkyy verkon ryhmärakenne. (Wolfram Research, 2014)

Mathematicaa käyttämällä paljastuu, etteivät sen verkkotoiminnot ole reaaliaikaisesti tai graafisesti vuorovaikutteisia. Käyttäjä kirjoittaa komentoja, ja ohjelma vastaa tuottamalla kiinteän kuvan tai tekstimuotoisen vastauksen. Mathematica-komennoilla voi ohjelmoida, joten edistynyt käyttäjä pystynee tuottamaan komentosarjoja, jotka esimerkiksi jakavat verkon hierarkkisesti ryhmiin ja piirtävät kuvan jokaisesta ryhmästä. Tämä on kuitenkin kaukana Shneidermanin (1996) ideasta visuaaliselle tiedonhaulle (visual information seeking mantra): ensin yleisnäkymä, sitten lähennys ja suodatus, ja lopuksi yksityiskohtia tarpeen mukaan. Siis Mathematican käytettävyys suurten verkkojen analysoimiseen vuorovaikutteisesti ja ihmissilmin on kyseenalainen.

Tulip on Cytoscapen ja Gephin kaltainen vuorovaikutteinen, suurten verkkojen analysoimiseen ja piirtämiseen kehitetty ohjelmisto (Auber ym., 2012, 2013). Tulipin versio 4.4.0 sisältää vielä suuremman kokoelman analyysi- ja piirtoalgoritmeja kuin Cytoscape ja Gephi. Mukana on hierarkkinen ryhmittelyalgoritmi, mutta se ei toiminut lyhyessä käytännön kokeilussa.

5 Johtopäätökset

Tässä luvussa pohditaan, mikä verkonpiirtomenetelmä sopii mihinkin käyttötarkoitukseen. Ensin esitetään havaintoja ja hypoteeseja liittyen piirtomenetelmien vakauteen. Vakausominaisuus on esitelty luvussa 3.5.

5.1 Vakaus

Didimo ja Montecchiani toteavat, että heidän tekniikkansa mahdollistaa verkon eri osien piirtämisen toisistaan riippumattomasti, minkä takia algoritmilla voisi pystyä piirtämään tehokkaasti dynaamisia verkkoja, joiden rakenne muuttuu paikallisesti. Artikkelin mukaan algoritmi jakaa piirtoaluetta pyrkien siihen, että suorakulmioiden pinta-ala vertautuu kussakin ryhmässä olevien solmujen määrään tai kokonaispainoon. (Didimo ja Montecchiani, 2014) Tästä voi päätellä, että solmujen lisääminen ja poistaminen vaikuttaa piirroksessa paikallisesti. Edelleen voi päätellä, että jos ryhmittelyhierarkiaan ei tule uusia ryhmiä, piirroskaan ei muutu merkittävästi. Myös muutokset kaarissa vaikuttavat vain niihin ryhmiin, joihin kaaret liittyvät. Algoritmin kerrotaan hienosäätävän asettelua niillä solmuilla, joilla on ryhmien välisiä kaaria. Tässä yhteydessä kaarten lisääminen ja poistaminen

muuttaa ryhmien sisäistä asettelua kaarimuutoksiin liittyvien solmujen osalta, siis vain vähän. Suuremmassa mittakaavassa vierekkäiset ryhmät saattavat vaihtaa paikkaa, jos kaaria naapuriryhmiin lisätään ja poistetaan riittävän paljon. Tämä johtuu ositusverkkojen laskemisesta ja asettelemisesta voimaohjatulla algoritmilla.

ASK-GraphViewissä käytetty muunnettu Boruvkan supistamisalgoritmi sulauttaa yhteen solmuja, joita yhdistävillä kaarilla on pieni paino, ja tekee tämän rekursiivisesti (Abello, 2004, s. 350–351; Abello ym., 2006, s. 671). Jos verkkoa muokataan, muunnettu Boruvkan algoritmi voidaan aina suorittaa uudestaan koko verkolle. Jos yksi kaari lisätään, poistetaan tai sen painoa muutetaan, vaikutus verkon suuren mittakaavan ryhmittelyyn on sitä suurempi, mitä suurempi kaaren paino on: jos lisätään painava kaari, se muuttaa solmujen ryhmittelyä niissä ryhmissä, joihin kaari liittyy. Mitä painavampi kaari on, sitä ylemmäksi ryhmittelyhierarkiaan muutokset heijastuvat.

ASK-GraphViewin lopullinen asettelualgoritmi on voimaohjattu ja samanlainen kuin Eadesin ja Lin Huangin (2000) kuvaama: ryhmien sisällä kaaret aiheuttavat vahvoja jousivoimia, ryhmien väliset kaaret ovat heikompia jousivoimia, ja kaikki solmut hylkivät toisiaan. Koska ryhmien sisällä on vahvempia voimia, on helppo päätellä, että muutokset yhden ryhmän sisällä eivät merkittävästi vaikuta vierekkäisten ryhmien asetteluun. Toisin sanoen paikalliset muutokset näkyvät asettelussakin paikallisesti.

GrouseFlocksin sovelutuvuutta dynaamisten verkkojen piirtämiseen ei ole kommentoitu (Archambault ym., 2008). GrouseFlocksin käyttämä voimaohjattu algoritmi hyödyntää satunnaisuutta, eivätkä algoritmin tekijät ole todistaneet vakautta (Frick ym., 1995). Samoin VHyXY-ohjelmiston käyttäytymistä verkon muuttuessa ei ole kommentoitu (Batagelj ym., 2011). Tasomainen, k-ydinkomponenttinen ryhmittely heijastaa verkon muutokset ositusverkon piirrokseen, jos muuttuneiden solmujen aste on alle k, ja vastaavasti k-ytimen komponentteihin, jos solmujen aste on k tai suurempi.

5.2 Piirtomenetelmien yhteenveto

Taulukko kuvassa 9 vertailee lukujen 3.1–3.5 piirtomenetelmiä ominaisuuksittain.

Syöteominaisuus jakaa piirtomenetelmiä eniten. Kaarien suhteen painotettu verkko sopii ASK-GraphViewiin. GrouseFlocks vastaavasti vaatii painottamattoman verkon. Muut menetelmät eivät huomioi painotusta. Didimon ja Montecchianin piirtoalgoritmi on monikäyttöinen, koska sen kanssa voi käyttää erilaisia hierarkkisia ryhmittelyalgoritmeja.

Kaikki piirtomenetelmiin liittyvät ryhmittely- ja piirtoalgoritmit ovat tehokkaita polynomiaikaisia, monet jopa luokkaa $O(n \log n)$. GrouseFlocks ja VHyXY käyttävät verkon eri osien piirtämiseen eri algoritmeja. Vaikka GrouseFlocks käyttää yleisessä tapauksessa voimaohjattua $O(m^3)$ algoritmia, tämän voisi ehkä korvata tehokkaammalla Hachulin ja

Jüngnerin (2005) FM^3-algoritmilla. Olennaisinta on kuitenkin, että piirtoalgoritmit eivät piirrä verkon kaikkia yksityiskohtia kerralla, joten yksityiskohtien määrää rajoittamalla voidaan taata verkkonäkymän päivittäminen muutamassa sekunnissa.

Algoritmi tai ohjelma	Syöte	Asymp. tehokkuus	Pelkistäminen	Vakaus
Didimo ja Montecchiani (2014)	painotetut solmut, hierarkkinen ryhmittely	$O(n \log n + m)$	osaverkot kyllä, kaarten niputus lisäalgoritmilla	luultavasti kyllä
ASK-GraphView	painotetut kaaret	$O(\log(n/r))$ esikäsittely, $O(1)$ piirtäminen	osaverkot kyllä	luultavasti kyllä, riippuu kaarien painoista
GrouseFlocks	painottamaton, hierarkia tarvittaessa	$O(n \log^2 n + m \log n)$ pahin tapaus, riippuu osaverkon rakenteesta	osaverkot kyllä	tuntematon
VHyXY: (tasoverkko, k-ydinkomponentit)	Painotus ei vaikuta ryhmittelyyn.	ryhmittely $O((n + m) \log \Delta)$, piirtäminen ei yksiselitteistä	osaverkot kyllä	tuntematon
VHyXY: (d-tiheyksinen, k-ydinkomponentit)	Painotus ei vaikuta ryhmittelyyn.	ryhmittely $O((n + m)\Delta)$, piirtäminen ei yksiselitteistä	osaverkot kyllä	tuntematon

n solmujen määrä
m kaarien määrä
r keskusmuistin määrä
k solmun aste (kaarien määrä solmussa)
d m/n jollekin osaverkolle
Δ korkein asteluku (suurin määrä kaaria solmussa)

Kuva 9: Yhteenveto lukujen 3.1–3.5 tuloksista

5.3 Yhteenveto kaikista tuloksista

Tämän kandidaatintyön tavoitteena oli selvittää, millaisia hierarkkisesti ryhmitteleviä verkonpiirtoalgoritmeja on olemassa ja miten ne eroavat toisistaan teoreettisesti. Tutkimuksessa selvisi, että on kehitetty useita tehokkaita algoritmeja ja ohjelmistoarkkiteh-

tuureja suurten verkkojen piirtämiseen hierarkkisen ryhmittelyn periaatteella. Nämä menetelmät ovat kuitenkin vielä tutkimusvaiheen prototyyppejä. Yleisesti saatavilla olevat tunnetut ohjelmistot, kuten Cytoscape, Gephi ja Mathematica, eivät tällä hetkellä sisällä automaattista hierarkkista ryhmittelyä tai suurten verkkojen automaattista pelkistämistä Shneidermanin (1996) ideaa mukaillen.

5.4 Kriittistä tarkastelua

Tämä kandidaatintyö ei ole täysin kattava kaikkien hierarkkiseen ryhmittelyyn perustuvien verkonpiirtomenetelmien suhteen. Myöskään dynaamisten verkkojen piirtämisen vakautta ei ole selvitetty täydellisesti. Käytettävyys on mukana rajatusti lähinnä piirtoalgoritmien tehokkuutena ja hierarkkisten ryhmien pelkistämiskykynä. Kuitenkin se, miten ymmärrettäviä tuloksia ihmisen kannalta ryhmittely ja piirtäminen tuottavat, on myös tärkeää.

5.5 Suositus ja jatkotutkimus

Cytoscapeen, Gephiin tai Tulipiin olisi ehkä mahdollista rakentaa ASK-GraphViewin tai GrouseFlocksin tapaisia ominaisuuksia. Ensiksikin tarvittaisiin hierarkkinen ryhmittelytoiminto, joka kykenee tuottamaan valmiin ryhmittelypuun sellaisesta verkkotiedostosta, jota ei kokonsa puolesta voi käsitellä keskusmuistissa. Toiseksi olisi hyvä, jos hierarkkisia ryhmittelyalgoritmeja olisi monia erilaisia, jotta käyttäjä pystyisi luomaan erilaisia ryhmittelypuita. Luodut ryhmittelyt voisi myös tallentaa verkkotiedostoon. Kolmanneksi ASK-GraphViewin kaltainen ryhmittelypuun navigointinäkymä on käyttökelpoinen, mutta siihenkin pitäisi käytettävyyden takia soveltaa Shneidermanin (1996) ideaa. Hierarkiapuu olisi aluksi pelkistetty näyttäen vain juurisolmut; puuta voisi laajentaa ja supistaa, kuten ASK-GraphViewissä laajennetaan ja supistetaan abstrakteja solmuja; ryhmittelypuunäkymää voisi lähentää, loitontaa ja selata kuten verkkonäkymää.

Luvuissa 3.1–3.5 mainittujen algoritmien ja ohjelmistojen käyttökelpoisuutta dynaamisille verkoille olisi hyvä tutkia: joko kokeellisesti ja tilastollisesti tai matemaattisella todistamisella. Näiden algoritmien ja ohjelmistojen käytettävyyttä voisi myös tutkia enemmän esimerkiksi toteuttamalla niiden kaltaisia ominaisuuksia tunnettuihin verkonpiirtoohjelmiin ja tekemällä käytettävyystestejä.

Lähteet

James Abello. Hierarchical graph maps. *Computers & Graphics*, 28(3):345–359, 2004. ISSN 0097-8493. doi: 10.1016/j.cag.2004.03.012.

James Abello, Frank van Ham ja Neeraj Krishnan. Ask-graphview : A large scale graph visualization system. *IEEE transactions on visualization and computer graphics*, 12(5): 669–676, 2006. ISSN 1077-2626. doi: 10.1109/TVCG.2006.120.

Daniel Archambault, Tamara Munzner ja David Auber. Grouseflocks: steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008. doi: 10.1109/TVCG.2008.34.

David Auber, David Archambault, Romain Bourqui, Antoine Lambert, Morgan Mathiaut, Patrick Mary, Maylis Delest, Jonathan Dubois ja Guy Melançon. The Tulip 3 framework: A scalable software library for information visualization applications based on relational data. Tutkimusraportti 7860, Research centre Bordeaux – sud-ouest, Grenoble, Ranska, 2012. URL `http://hal.inria.fr/hal-00659880`.

David Auber, Sophie Bardet, Gerald Gainant, Chrys Myers, Bertrand Mathieu, Sebastien Grivet, David Duke, Loïc Jézéquel, Daniel Archambault, Maylis Delest, Jean Philippe Domenger, Patrick Mary, Charles Huet, Jonathan Dubois, Morgan Mathiaut ja Ludwig Fiolka. Tulip, 2013. URL `http://tulip.labri.fr/`. Tietokoneohjelma ja verkkosivu. Viitattu 6.3.2014.

Oliver Bastert ja Christian Matuszewski. Layered drawings of digraphs. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 87–120. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Mathieu Bastian, Sebastien Heymann ja Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. *Proceedings of the third international ICWSM conference*, sivut 361–362. Association for the Advancement of Articifial Intelligence, 2009. ISBN 978-1-57735-421-5 (painettu) 978-1-57735-422-2 (sähköinen). URL `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154/1009`.

Vladimir Batagelj, Franz J. Brandenburg, Walter Didimo, Giuseppe Liotta, Pietro Palladino ja Maurizio Patrignani. Visual analysis of large graphs using (x, y)-clustering and hybrid visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 17 (11):1587–1598, 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2010.265.

Giuseppe Battista, Walter Didimo, Maurizio Patrignani ja Maurizio Pizzonia. Orthogonal and quasi-upward drawings with vertices of prescribed size. *Graph Drawing : 7th International Symposium, GD'99 Štiřín Castle, Czech Republic September 15–19, 1999 Proceedings*, osa 1731 sarjasta *Lecture Notes in Computer Science*, sivut 297–310, Berlin Heidelberg, Saksa, 1999. Springer-Verlag. ISBN 978-3-540-66904-3 (painettu) 978-3-540-46648-2 (sähköinen). doi: 10.1007/3-540-46648-7.

Norman L. Biggs. *Discrete mathematics*. Oxford university press, Oxford, Yhdistynyt kuningaskunta, toinen painos, 2002. ISBN 978-0-19-850717-8.

Biogrid. Biogrid database statistics, 2014. URL `http://wiki.thebiogrid.org/doku.php/statistics`. Viitattu 24.2.2014.

Ulrik Brandes. Drawing on physical analogies. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 71–86. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Jürgen Branke. Dynamic graph drawing. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 228–246. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Ralf Brockenauer ja Sabine Cornelsen. Drawing clusters and hierarchies. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 193–227. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Christoph Buchheim, Michael Jünger ja Sebastian Leipert. Improving walker's algorithm to run in linear time. Teoksessa *Graph Drawing : 10th International Symposium, GD 2002 Irvine, CA, USA, August 26–28, 2002 Revised Papers*, osa 2528 sarjasta *Lecture Notes in Computer Science*, sivut 344–353. Springer-Verlag, Berlin Heidelberg, Saksa, 2002. ISBN 978-3-540-00158-4 (painettu) 978-3-540-36151-0 (sähköinen). doi: 10.1007/3-540-36151-0.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest ja Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, Massachusetts, Yhdysvallat, toinen painos, 2001. ISBN 0-262-53196-8.

Walter Didimo ja Fabrizio Montecchiani. Fast layout computation of clustered networks: Algorithmic advances and experimental analysis. *Information Sciences*, 260:185–199, 2014. ISSN 0020-0255. doi: 10.1016/j.ins.2013.09.048.

Tim Dwyer, Kim Marriott ja Peter J. Stuckey. Fast node overlap removal. *Graph Drawing : 13th International Symposium, GD 2005, Limerick, Ireland, September 12-14, 2005. Revised Papers*, osa 3843 sarjasta *Lecture Notes in Computer Science*, sivut 153–164, Berlin Heidelberg, Saksa, 2006. Springer-Verlag. ISBN 978-3-540-31425-7 (painettu) 978-3-540-31667-1 (sähköinen). doi: 10.1007/11618058.

Peter Eades ja Mao Lin Huang. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4(3):157–181, 2000. doi: 10.7155/jgaa.00029.

Peter Eades, Wei Lai, Kazuo Misue ja Kozo Sugiyama. Preserving the mental map of a diagram. Teoksessa *Proceedings of compugraphics 91*, sivut 24–33. Association for Computing Machinery, 1991. URL `http://www.cs.tsukuba.ac.jp/~misue/publications/techreport/iias-rr-91-16e.pdf`.

Rudolf Fleischer ja Colin Hirsch. Graph drawing and its applications. Teoksessa *Drawing Graphs : Methods and models*, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 1–22. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Arne Frick, Andreas Ludwig ja Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. Teoksessa *Graph Drawing*, osa 894 sarjasta *Lecture Notes in Computer Science*, sivut 388–403. Springer-Verlag, Berlin Heidelberg, Saksa, 1995. ISBN 978-3-540-58950-1 (painettu) 978-3-540-49155-2 (sähköinen). doi: 10.1007/3-540-58950-3.

M. Girvan ja M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. doi: 10.1073/pnas.122653799.

Sebastien Grivet, David Auber, Jean-Philippe Domenger ja Guy Melancon. Bubble tree drawing algorithm. *Computer Vision and Graphics : International Conference, ICCVG 2004, Warsaw, Poland, September 2004, Proceedings*, osa 32 sarjasta *Computational Imaging and Vision*, sivut 633–641. Springer Netherlands, 2006. ISBN 978-1-4020-4178-5 (painettu) 978-1-4020-4179-2 (sähköinen). doi: 10.1007/1-4020-4179-9.

Stefan Hachul ja Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. *Graph Drawing : 12th International Symposium, GD 2004, New York, NY, USA, September 29-October 2, 2004, Revised Selected Papers*, osa 3383 sarjasta *Lecture notes in computer science*, sivu 285–295, Berlin Heidelberg, Saksa, 2005.

Springer-Verlag. ISBN 978-3-540-24528-5 (painettu) 978-3-540-31843-9 (sähköinen). doi: 10.1007/978-3-540-31843-9_29.

David Harel ja Yehuda Koren. Graph drawing by high-dimensional embedding. Teoksessa *Graph Drawing*, osa 2528 sarjasta *Lecture Notes in Computer Science*, sivut 207–219. Springer-Verlag, Berlin Heidelberg, Saksa, 2002. ISBN 978-3-540-00158-4. doi: 10.1007/3-540-36151-0_20.

Britta Landgraf. 3d graph drawing. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 172–192. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

M. Newman. The structure and function of complex networks. *SIAM Review*, 45 (2):167–256, 2003. ISSN 0036-1445 (painettu) 1095-7200 (sähköinen). doi: 10.1137/S003614450342480.

Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007. ISSN 1574-0137. doi: 10.1016/j.cosrev.2007.05.001.

Ben Shneiderman. Tree visualization with tree-maps: a 2-d space-filling approach. Tekninen raportti, Human-Computer Interaction Lab, University of Maryland Institute for Advanced Computer Studies, 1991. URL `http://hcil2.cs.umd.edu/trs/91-03/91-03.html`.

Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, 1996. Proceedings., IEEE Symposium on*, sivut 336–343. Springer, Berlin Heidelberg, Saksa, 1996. doi: 10.1109/VL.1996.545307.

Kozo Sugiyama. *Graph drawing and applications for software and knowledge engineers*. World scientific, River Edge, N.J., USA, 2002. ISBN 978-981-02-4879-6 (painettu) 978-981-4489-24-9 (sähköinen). doi: 10.1142/9789812777898.

The Cytoscape Consortium. *Cytoscape 3.1.0 User Manual*, 2014. URL `http://www.cytoscape.org/manual/Cytoscape3_1_0Manual.pdf`. Viitattu 22.2.2014.

The Gephi consortium. Gephi, open source graph visualization software, 2013. URL `https://gephi.org/`. Tietokoneohjelma ja verkkosivu. Viitattu 23.4.2014.

Ian Tuomi. Logiikkakaavioiden generointi tehdassuunnittelussa. Kandidaatintyö, Sähkötekniikan korkeakoulu, Aalto-yliopisto, Espoo, 2012.

René Weiskircher. Drawing planar graphs. Teoksessa *Drawing Graphs : Methods and models*, Michael Kaufmann ja Dorothea Wagner, toimittajat, osa 2025 2001 sarjasta *Lecture notes in computer science*, sivut 23–45. Springer-Verlag, Berlin Heidelberg, Saksa, 2001. ISBN 978-3-540-42062-0 (painettu) 978-3-540-44969-0 (sähköinen). doi: 10.1007/3-540-44969-8.

Wolfram Research. *Graphs & networks–Wolfram Mathematica 9 Documentation*, 2014. URL `http://reference.wolfram.com/mathematica/guide/GraphsAndNetworks.html`. Viitattu 28.2.2014.