Kausaalivaikutusten identifiointi algoritmisesti

Santtu Tikka

Tilastotieteen pro gradu -tutkielma

Jyväskylän yliopisto
Matematiikan ja tilastotieteen laitos
27. helmikuuta 2015

JYVÄSKYLÄN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Tikka, Santtu: Kausaalivaikutusten identifiointi algoritmisesti

Tilastotieteen pro gradu -tutkielma, 43 sivua, 2 liitettä (17 sivua)
27. helmikuuta 2015

Tiivistelmä

Kokeelliset tutkimukset ovat perinteinen lähestymistapa kausaalisuuden tutkimiseen tilastotieteessä. Ideaalisessa tilanteessa kiinnostavat muuttujat voidaan mitata halutulla tarkkuudella ja mahdolliset sekoittavat tekijät voidaan eliminoida hyvin suunnitellulla koeasetelmalla. Tällöin tutkijan on mahdollista sulkea havaittu efekti sattuman ulkopuolelle ja tulkita havainnot kausaalisesta näkökulmasta. Käytännössä tällaista optimaalista tilannetta on usein mahdotonta saavuttaa, eikä moniin tärkeisiin kysymyksiin voida saada vastausta kokeellisella tutkimuksella.

Judea Pearlin kausaalimalli tarjoaa formaalin lähestymistavan kausaalisuuteen, ja mallia voidaan soveltaa niin kokeellisen kuin havainnoivankin tutkimuksen yhteydessä. Tässä tutkielmassa keskitytään erityisesti kausaalimalleihin kohdistuviin interventioihin sekä kausaalilaskentaan, joiden avulla voidaan vastata moniin kausaalisuutta koskeviin kysymyksiin. Kausaalilaskenta rakentuu suunnattujen silmukattomien graafien ympärille, jotka tarjoavat esitystavan muuttujien välisille suhteille.

Kaikkia interventioita ei kuitenkaan ole mahdollista määrittää. Interventioita, jotka voidaan määrittää yksikäsitteisesti riittävillä oletuksilla muuttujien välisistä kausaalisista yhteyksistä, kutsutaan identifioituviksi. Ei ole itsestäänselvyys, mitkä vaikutukset ovat identifioituvia ja mitkä eivät, annetussa graafissa.

Kausaalilaskennan soveltaminen identifioituvuuden määrittämiseksi käytännössä on haastavaa ja työlästä, minkä seurauksena interventioiden käsittelyyn on kehitetty algoritmisia ratkaisuja. Eräs tällainen algoritmi esitellään ja implementoidaan, ja toteutuksen yksityiskohtia käsitellään esimerkkien avulla.

Avainsanat: kausaalimalli, kausaalivaikutus, kausaalilaskenta, identifioituvuus, graafi, C-komponentti, pensasaita, algoritmi, d-separoituvuus

Sisällys

1 Johdanto

Kausaalisuudella tarkoitetaan tapahtumien välisiä suhteita, joissa jokin joukko tapahtumia (syyt) aiheuttaa jonkin toisen joukon tapahtumia (seuraukset). Kausaalipäättelyssä tehdään päätelmiä tällaisista suhteista hyödyntäen esimerkiksi kerättyä aineistoa tai ennakkotietoa. Kausaalipäättelyyn on kehitetty lukuisia tilastotieteellisiä lähestymistapoja, jotka keskittyvät kausaalisuuden eri osa-alueisiin, kuten kausaalisuuden suunnan määrittämiseen tai tapahtumien välisten suhteiden yksityiskohtien selvittämiseen. Kausaalipäättelyä voidaan soveltaa myös johonkin kiinnostavaan muuttujajoukkoon kohdistuvien toimintojen eli interventioiden tarkasteluun, jolloin on kyse kausaalivaikutusten estimoinnista.

Neyman (1923) käsitteli kausaalisuutta potentiaalisten lopputulosten eli kontrafaktuaalien avulla. Kontrafaktuaalien mallissa on äärellinen määrä käsittelyjä, ja jokaisella populaation yksilöllä on potentiaalinen lopputulos jokaista käsittelyä kohden. Yksinkertaisessa tilanteessa lopputuloksia on kaksi, missä yksilö joko altistetaan käsittelylle tai jätetään käsittelemättä. Tällöin kausaalivaikutus määritellään näiden potentiaalisten lopputulosten erotuksena. On mahdotonta havaita molempia lopputuloksia samalta yksilöltä samanaikaisesti, mikä johtaa kausaalipäättelyn perusongelmaan. Neyman käsitteli malliaan vain täysin satunnaistettujen kokeiden yhteydessä, mutta Rubin (1974) yleisti mallin koskemaan sekä havainnoivia että kokeellisia tutkimuksia.

Toinen kausaalipäättelyn haara perustuu rakenneyhtälömalleihin (*Structural Equation Model*). Rakenneyhtälömalli muistuttaa moniulotteista regressiomallia, mutta rakenneyhtälömallissa yhden tai useamman regressioyhtälön vastemuuttuja voi esiintyä selittäjänä jossain toisessa yhtälössä. Nämä yhtälöt kuvaavat mallin muuttujien välisiä kausaalisia yhteyksiä, ja tavoitteena on antaa yhtälöissä esiintyville regressiokertoimille kausaalinen tulkinta (Kline, 1998). Galles ja Pearl (1998) osoittivat, että Neyman–Rubin -kausaalimalli ja rakenneyhtälömalleihin pohjautuva kausaalipäättely ovat keskenään ekvivalentteja.

Neyman–Rubin -kausaalimalli soveltuu kokeellisiin tutkimuksiin, sillä se perustuu tutkijan valitsemaan käsittelyyn, jolloin kokeen lopputuloksen syy on aina konkreettisesti aiheutettu koeasetelmaa manipuloimalla. Tämän oleellisen seikan totesi Holland (1986) fraasillaan: ”No causation without manipulation”. Manipulointiin liittyy kuitenkin lukuisia ongelmia. Jos esimerkiksi halutaan tutkia jonkin taudin vaikutusta, on epäeettistä aiheuttaa kyseinen tauti kiinnostavalle koeryhmälle.

Vaikka asetelmaa pystyttäisiin manipuloimaan, ei intervention määrittäminen ole aina yksinkertaista. Esimerkiksi analysoitaessa jotain teollista prosessia voi kiinnostuksen kohteena olla jokin tietty säätöparametri, jonka arvoa ei voida muuttaa vaikuttamatta samalla muihin parametreihin. Rubinin ja Hollandin ajattelutavassa muuttujan kausaalivaikutus ei ole määriteltävissä

myöskään silloin, kun kyseistä muuttujaa ei voida manipuloida. Tällöin esimerkiksi kysymys siitä, mikä on sukupuolen kausaalivaikutus palkkaeroihin, ei ole mielekäs.

Luvussa 2 käsitellään kausaalimallia, jonka määritteli Pearl (1995). Pearlin kausaalimalli mahdollistaa yleisten interventioiden täsmällisen käsittelyn graafien ja niin sanotun $do(\cdot)$-operaattorin avulla. Pearlin kausaalimallin yhteydessä kausaalivaikutuksella tarkoitetaan juuri tämän operaattorin muodostamia todennäköisyyjakaumia. Erityisen kiinnostavia ovat kausaalivaikutukset, jotka voidaan määrittää yksikäsitteisesti. Tällaisia kausaalivaikutuksia kutsutaan identifioituviksi ja niistä voidaan tehdä päätelmiä pelkästään havaintojen avulla. Pearlin kausaalimallissa periaatteessa mihin tahansa muuttujaan voidaan kohdistaa interventio, jolloin esimerkiksi sukupuolen kausaalivaikutusta palkkaeroihin voidaan käsitellä. Usein tutkimuskysymys onkin, kuinka intervention vaikutus voidaan määrittää tekemättä interventiota.

Luku 3 keskittyy kausaalilaskentaan, jonka Pearl (1995) johti interventioiden vaikutusten määrittämiseksi. Kausaalilaskenta on joukko päättelysääntöjä, joiden avulla kausaalivaikutuksen jakauma pyritään esittämään ainoastaan havaittujen todennäköisyyksien avulla. Päättelysääntöjen soveltamiseen liittyy kuitenkin lukuisia ongelmia, minkä seurauksena identifioituvuuden määrittäminen algoritmisesti on varteenotettava vaihtoehto.

Luvussa 4 esitellään algoritmi, jonka johtivat Shpitser ja Pearl (2006b). Tämän algoritmin avulla identifioituvan kausaalivaikutuksen jakauma voidaan aina määrittää annetussa graafissa. Tapauksissa, joissa kausaalivaikutus ei ole identifioituva, tuottaa algoritmi myös ongelmallisen graafirakenteen, joka aiheuttaa identifioitumattomuuden. Tarkoituksena on toteuttaa kyseinen algoritmi tilastollisella ohjelmointikielellä R (R Core Team, 2014), ja tämän toteutuksen yksityiskohtia käsitellään luvussa 5. Luvussa 6 havainnollistetaan algoritmin sekä sen toteutuksen ominaisuuksia esimerkkien avulla. Vastaavan identifioituvuusalgoritmin määrittelivät Huang ja Valtorta (2006), joka perustui Tianin ja Pearlin (2003) johtamaan algoritmiin. Kumpikaan näistä kahdesta edellä mainituista algoritmeista ei kuitenkaan tuota mitään ylimääräistä informaatiota annetusta graafista identifioitumattoman kausaalivaikutuksen tapauksessa.

2 Kausaalimallit ja identifioituvuus

Graafit koostuvat solmuista ja niiden välisistä särmistä. Solmujen voidaan ajatella vastaavan havaittuja ja havaitsemattomia muuttujia, ja särmien näiden välisiä kausaalisia yhteyksiä. Kausaalisuuden suunta otetaan huomioon tarkastelemalla vain suunnattuja graafeja, jotka eivät sisällä silmukoita. Visuaalisesti solmut ja särmät esitetään pisteinä ja näitä yhdistävinä janoina. Suunnatun graafin tapauksessa särmien suuntia kuvataan nuolilla. Jatkossa toistuvasti käytettävät graafiteoreettiset käsitteet esitellään teoksen Koller ja Friedman (2009) mukaisesti. Myös seuraavat merkinnät ovat jatkuvasti esillä: isoilla kirjaimilla tarkoitetaan muuttujia, pienillä kirjaimilla muuttujien arvoja ja lihavoiduilla kirjaimilla näistä muodostettuja joukkoja.

2.1 Graafiteorian perusteet

Graafi on pari $G = \langle \mathbf{V}, \mathbf{E} \rangle$, missä $\mathbf{V}$ ja $\mathbf{E}$ ovat sellaisia joukkoja, että

$$\mathbf{E} \subset \{\{X, Y\} \mid X \in \mathbf{V}, Y \in \mathbf{V}, X \neq Y\}.$$

Joukon $\mathbf{V}$ alkioita sanotaan graafin G solmuiksi, ja joukon $\mathbf{E}$ alkioita sanotaan graafin G särmiksi. Graafi $F = \langle \mathbf{V}', \mathbf{E}' \rangle$ on graafin G *aligraafi* jos $\mathbf{V}' \subset \mathbf{V}$ ja $\mathbf{E}' \subset \mathbf{E}$. Tällöin merkitään $F \subset G$. Graafi G on *suunnattu* jos sen särmäjoukko $\mathbf{E}$ koostuu suunnatuista pareista (X, Y). Suunnatussa graafissa solmu V_2 on solmun V_1 *lapsi* jos graafi G sisältää särmän solmusta V_1 solmuun V_2, eli jos $(V_1, V_2) \in \mathbf{E}$. Vastaavasti solmu V_2 on solmun V_1 *vanhempi* jos $(V_2, V_1) \in \mathbf{E}$. Kahden solmun välistä suhdetta kuvataan usein merkinnällä $V_1 \to V_2$, jossa solmu V_1 on solmun V_2 vanhempi ja V_2 on solmun V_1 lapsi. Yhtäpitävästi voidaan myös merkitä $V_2 \leftarrow V_1$.

Olkoot $n \geq 1$, $\mathbf{V} = \{V_1, \ldots, V_n\}$ ja $V_i \neq V_j$ kaikilla $i \neq j$. Jos $n > 1$, niin graafi $H = \langle \mathbf{V}, \mathbf{E} \rangle$ on *polku* jos

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}\}$$

tai jos

$$\mathbf{E} = \{\{V_1, V_2\}, \{V_2, V_3\}, \ldots, \{V_{n-1}, V_n\}, \{V_n, V_1\}\}.$$

Ensimmäisessä tapauksessa sanotaan, että H on polku solmusta V_1 solmuun V_n. Jälkimmäisessä tapauksessa sanotaan, että polku H on *silmukka*. Jos $n = 1$, niin graafi $H = \langle \{V_1\}, \emptyset \rangle$ on myös polku. Polku H on *suunnattu polku* jos sen kaikki särmät ovat suunnattuja ja saman suuntaisia, eli jos

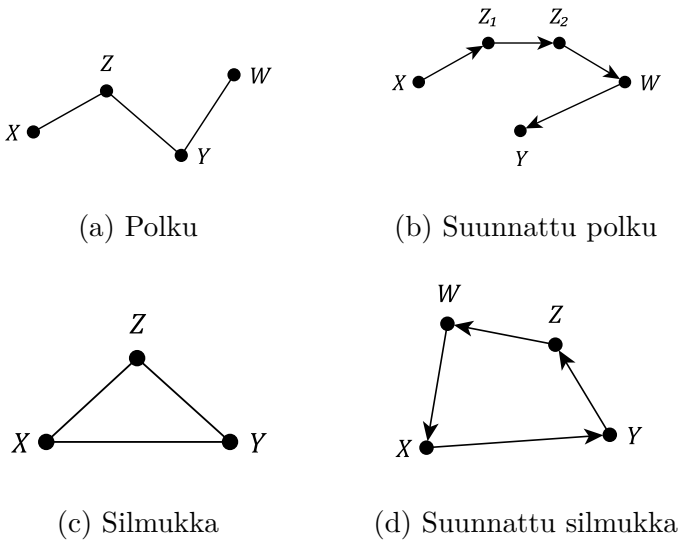
$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n)\}$$

tai jos

$$\mathbf{E} = \{(V_1, V_2), (V_2, V_3), \ldots, (V_{n-1}, V_n), (V_n, V_1)\}.$$

Solmu V_2 on solmun V_1 *jälkeläinen* graafissa G, jos on olemassa suunnattu polku H solmusta V_1 solmuun V_2 ja $H \subset G$. Vastaavasti solmu V_2 on solmun

V_1 *esivanhempi* graafissa G, jos on olemassa suunnattu polku H solmusta V_2 solmuun V_1 ja $H \subset G$. Jos graafi G ei sisällä lainkaan silmukoita, niin se on *silmukaton*. Graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$ on *yhtenäinen* jos jokaisen solmuparin $V_i, V_j \in \mathbf{V}$ välillä on olemassa polku $H \subset G$. Kuvassa 1 on erilaisia polkuja ja silmukoita.

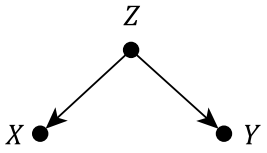


(a) Polku (b) Suunnattu polku

(c) Silmukka (d) Suunnattu silmukka

Kuva 1: Suunnattuja ja suuntaamattomia polkuja sekä silmukoita

Suunnatun graafin tapauksessa voidaan sen aligraafeja tarkastella myös suuntaamattomina, kun kaikki graafin särmät ajatellaan suuntaamattomiksi. Suunnattu graafi voi siis sisältää esimerkiksi polkuja, vaikka se ei sisältäisi yhtään suunnattua polkua. Esimerkiksi kuvan 2 suunnattu graafi sisältää solmuja X ja Y yhdistävän polun, vaikka kyseisten solmujen välillä ei ole suunnattua polkua.



Kuva 2: Suuntaamaton polku suunnatussa graafissa

Olkoot $G = \langle \mathbf{V}, \mathbf{E} \rangle$ graafi ja $\mathbf{Y} \subset \mathbf{V}$. Oletetaan, että joukon $\mathbf{Y}$ solmut vastaavat joitakin havaittuja muuttujia ja että joukko $\mathbf{V}$ voi sisältää solmuja, jotka puolestaan vastaavat joitakin havaitsemattomia muuttujia. Tällöin merkinnöillä $Pa(\mathbf{Y})_G, An(\mathbf{Y})_G,$ ja $De(\mathbf{Y})_G$ tarkoitetaan joukon $\mathbf{Y}$ havaittuja vanhempia, esivanhempia ja jälkeläisiä. Kaikki edellä mainitut joukot sisältävät myös joukon $\mathbf{Y}$.

2.2 Pearlin kausaalimalli

Kausaalimallin avulla voidaan kuvata kiinnostuksen kohteena olevien muuttujien välisiä funktionaalisia suhteita. Lisäksi malli mahdollistaa sen muuttujiin kohdistuvien ulkopuolisten toimintojen eli interventioiden vaikutusten määrittämisen alimallien avulla, joissa intervention kohteena olevat funktiot asetetaan vakioiksi. Probabilistinen kausaalimalli yhdistää kausaaliset oletukset sekä tilastollisen informaation todennäköisyysjakauman muodossa. Judea Pearl määritteli sekä deterministisen kausaalimallin että sen probabilistisen laajennuksen (ks. Pearl, 2009, sivut 203-205), joita käsitellään tässä luvussa.

Määritelmä (Kausaalimalli, Pearl (2009) 7.1.1)**.** *Kausaalimalli* on kolmikko

$$M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F} \rangle,$$

missä:

1. $\mathbf{U}$ on joukko havaitsemattomia taustamuuttujia, jotka määräytyvät mallin ulkopuolisista tekijöistä.

2. $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$ on joukko havaittuja muuttujia, jotka määräytyvät mallin sisältämistä muuttujista, eli joukon $\mathbf{U} \cup \mathbf{V}$ alkioista.

3. $\mathbf{F} = \{f_{V_1}, f_{V_2}, \ldots, f_{V_n}\}$ on sellainen joukko funktioita, että jokainen f_{V_i} on kuvaus joukolta $\mathbf{U} \cup (\mathbf{V} \setminus V_i)$ joukolle V_i, ja joukko $\mathbf{F}$ muodostaa kuvauksen joukolta $\mathbf{U}$ joukkoon $\mathbf{V}$. Toisin sanoen jokainen funktio f_{V_i} määrää muuttujan V_i arvon yksikäsitteisesti ehdolla joukon $\mathbf{U} \cup \mathbf{V}$ muut muuttujat, ja joukolla $\mathbf{F}$ on yksikäsitteinen ratkaisu. Joukko $\mathbf{F}$ voidaan esittää symbolisesti kirjoittamalla $v_i = f_{V_i}(pa_{V_i}, u_{V_i}),\ \ i = 1, \ldots, n$, missä pa_i on jokin realisaatio yksikäsitteisestä minimaalisesta joukosta $\mathbf{PA}_{V_i} \subset \mathbf{V} \setminus V_i$, joka on riittävä funktion f_{V_i} määrittelemiseksi. Vastaavasti u_{V_i} on jokin realisaatio yksikäsitteisestä minimaalisesta joukosta $\mathbf{U}_{V_i} \subset \mathbf{U}$, joka on riittävä funktion f_{V_i} määrittelemiseksi.

Vaatimus siitä, että joukko $\mathbf{F}$ muodostaa kuvauksen joukolta $\mathbf{U}$ joukkoon $\mathbf{V}$ tarkoittaa tässä sitä, että on olemassa yksikäsitteinen joukko funktioita $\mathbf{G} = \{g_{V_1}, \ldots, g_{V_n}\}$, missä jokainen funktio g_{V_i} määrittää muuttujan V_i arvon havaitsemattomien muuttujien avulla. Siis jos u on joukkoon $\mathbf{U}$ kuuluvien muuttujien arvoista muodostettu vektori, niin on oltava

$$v_i = g_{V_i}(u), \quad \text{kaikilla } \ i = 1, \ldots, n.$$

Joukon $\mathbf{F}$ yksikäsitteisellä ratkaisulla tarkoitetaan tällöin juuri joukkoa $\mathbf{G}$.

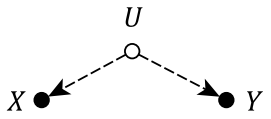
Jokaista kausaalimallia M vastaa suunnattu graafi $G = \langle \mathbf{W}, \mathbf{E} \rangle$, jonka solmujoukko $\mathbf{W}$ sisältää solmun jokaista mallin M havaittua ja havaitsematonta muuttujaa kohden. Graafin G särmäjoukko $\mathbf{E}$ määräytyy kausaalimallin M muuttujien $\mathbf{V}$ ja $\mathbf{U}$ välisistä funktionaalisista suhteista. Joukko $\mathbf{E}$ sisältää

särmän solmusta X solmuun Y jos $X \in \mathbf{PA}_Y$ eli jokaiseen solmuun V_i saapuu särmä kaikista solmuista, jotka tarvitaan tätä vastaavan funktion f_{V_i} määrittämiseen. Joukko $\mathbf{E}$ sisältää särmät solmusta U jokaiseen solmuun V_i, jolle $U \in \mathbf{U}_{V_i}$.

Kausaalimallin määritelmässä havaitsemattomille muuttujille ei aseteta mitään rajoitteita. Havaitsemattomista muuttujista voi siis lähteä mielivaltaisen monta särmää kausaalimallia vastaavassa graafissa. Jos jokaisesta havaitsemattomasta muuttujasta lähtee täsmälleen kaksi särmää, on kausaalimalli *semi-Markov-kausaalimalli*. Verma (1993) osoitti, että mikä tahansa kausaalimalli, joka sisältää havaitsemattomia muuttujia, on muunnettavissa semi-Markov-kausaalimalliksi. Tästä syystä voidaan jatkossa rajoittua tilanteisiin, joissa kaikki havaitsemattomat muuttujat vastaavat joitakin kahden havaitun muuttujan välisiä sekoittavia tekijöitä.

Kun kausaalimallia vastaava graafi on silmukaton, on joukon $\mathbf{G}$ olemassaolo ja yksikäsitteisyys taattu. Funktiot g_{V_i} voidaan tällöin muodostaa funktioiden f_{V_i} avulla, kun jokaisen solmun V_i paikalle sijoitetaan tätä vastaava esitys $f_{V_i}(pa_{V_i}, u_{V_i})$. Havaitut muuttujat saadaan tällä tavalla lopulta esitettyä vain havaitsemattomien muuttujien avulla, kun sijoitukset aloitetaan solmuista, joilla ei ole lainkaan vanhempia. Tällaisilla solmuilla symbolinen esitys saa muodon $f_{V_i}(u_{V_i})$, sillä solmun V_i vanhempia kuvaava joukko $\mathbf{PA}_{V_i}$ on tyhjä joukko. Sijoituksia voidaan nyt jatkaa rekursiivisesti korvaamalla jokaisen muuttujan symbolisessa esityksessä esiintyvät vanhemmat niiden symbolisilla esityksillä. Koska ensimmäisellä tasolla esiintyvät solmut, joilla ei ole lainkaan vanhempia, on saatu esitettyä vain havaitsemattomien muuttujien avulla, välittyy tämä ominaisuus myös kaikille tällaisten solmujen jälkeläisille.

Havaitsemattomaan muuttujaan liittyvät särmät kuvataan graafeissa katkoviivoilla, kuten kuvassa 3.



Kuva 3: Havaitsemattoman muuttujan merkintätapa

Usein havaitsemattomat muuttujat kuitenkin jätetään merkitsemättä graafiin, mikä yksinkertaistaa kausaalimallien esittämistä huomattavasti. Sen sijaan tällaisessa tilanteessa sanotaan, että solmujen X ja Y välillä on *kaksisuuntainen särmä*, joka kuvaa havaitsemattoman muuttujan vaikutusta. Jatkossa kuvan 3 merkintätavan sijaan käytetään siis kaksisuuntaisia särmiä, kuten kuvassa 4.

Kuva 4: Kaksisuuntaisen särmän merkintätapa

Tätä merkintätapaa käyttävät esimerkiksi Huang ja Valtorta (2006), Shpitser ja Pearl (2006b) ja Tian (2002). On syytä huomata, että kahden särmän välinen kaksisuuntainen särmä ei ole sama asia kuin jos graafi sisältäisi solmuja yhdistävät kaksi yksisuuntaista särmää, sillä nämä muodostaisivat graafiin silmukan, mikä ei ole sallittua.

Kaksisuuntaisten särmien yhteydessä on myös tavallista, että havaitsemattomia muuttujia ei sisällytetä suoraan kausaalimallia vastaavan graafin $G = \langle \mathbf{W}, \mathbf{E} \rangle$ solmujoukkoon $\mathbf{W}$. Käytännössä tämä tarkoittaa sitä, että edellä määritellyn kausaalimallia M vastaavan graafin G solmujoukoksi ilmoitetaan ainoastaan joukko, joka sisältää solmun jokaista joukon $\mathbf{V}$ muuttujaa kohden. Todellisuudessa havaitsemattomat muuttujat ovat osa graafin solmujoukkoa, mutta niitä ei määritellä eksplisiittisesti, vaan niiden olemassaolo on implisiittisesti määritelty kaksisuuntaisten särmien avulla. Vastaavasti särmäjoukon E voidaan ajatella sisältävän myös kaksisuuntaisia särmiä, kun ne mielletään kahtena havaitsemattomaan muuttujaan liittyvänä yksisuuntaisena särmänä. Laajennetaan nyt kausaalimallin käsitettä määrittelemällä havaitsemattomille muuttujille yhteisjakauma.

Määritelmä (Probabilistinen kausaalimalli, Pearl (2009) 7.1.6)**.** *Probabilistinen kausaalimalli* on pari

$$M = \langle M_D, P(\mathbf{U}) \rangle,$$

missä M_D on (deterministinen) kausaalimalli ja $P(\mathbf{U})$ on muuttujajoukon $\mathbf{U}$ yhteisjakauma.

Jatkossa kausaalimallilla tarkoitetaan nimenomaan probabilistista semi-Markov-kausaalimallia ilman erillistä mainintaa. Havaitsemattomien muuttujien yhteisjakauma $P(\mathbf{U})$ ja funktiot $\mathbf{F}$ määrittävät luonnollisesti myös havaittujen muuttujien yhteisjakauman $P(\mathbf{V})$. Kausaalimallin M ja sitä vastaavan graafin G välillä on täten yhteys myös todennäköisyysjakauman P kautta, missä $P = P(v_1, \ldots, v_n, u_1, \ldots, u_k)$ määrittää havaittujen ja havaitsemattomien muuttujien yhteisjakauman. Havaitsemattomat muuttujat oletetaan riippumattomiksi, eli $P(\mathbf{U}) = \prod_i P(U_i)$.

Määritelmä (Kausaalinen Markov-ehto)**.** Olkoot graafi G ja todennäköisyysjakauma P. Sanotaan, että G ja P toteuttavat *kausaalisen Markov-ehdon* jos

$$P = \prod_{i=1}^{n} P(v_i | pa^*(V_i)_G) \prod_{j=1}^{k} P(u_j),$$

missä $Pa^*(.)_G$ sisältää myös havaitsemattomat vanhemmat.

Oletus havaitsemattomien muuttujien riippumattomuudesta tarkoittaa käytännössä sitä, että määritelty kausaalimalli on riittävä kuvaamaan havaittujen muuttujien välisiä kausaalisia suhteita. Jos jotkin havaitsemattomat muuttujat $U_1, \ldots, U_l \in \mathbf{U}$ olisivatkin toisistaan riippuvia, olisi niillä oltava jokin yhteinen syy, joka ei ole mukana määritellyssä kausaalimallissa. Mallia olisi nyt laajennettava siten, että riippuvuusrakenteet havaitaan tarpeeksi kattavasti, jotta oletus havaitsemattomien muuttujien riippumattomuudesta on mielekäs. Kun kausaalinen Markov-ehto toteutuu, voidaan graafin G ja todennäköisyysjakauman P riippumattomuusominaisuudet yhdistää toisiinsa seuraavan määritelmän avulla.

Määritelmä (d-separoituvuus, Pearl (2009) 1.2.3)**.** Olkoot polku $H = \langle \mathbf{V}, \mathbf{E} \rangle$ ja solmujoukko $\mathbf{Z} \subset \mathbf{V}$. Polku H on solmujoukon $\mathbf{Z}$ *d-separoima* graafissa G, jos ja vain jos

1. H sisältää ketjun $I \to M \to J$ tai haarukan $I \leftarrow M \to J$, missä $M \in \mathbf{Z}$ ja $I, J \in \mathbf{V}$.

2. H sisältää käänteisen haarukan $I \to M \leftarrow J$, missä $De(M)_G \cap \mathbf{Z} = \emptyset$ eli yksikään solmun M jälkeläisistä ei kuulu joukkoon $\mathbf{Z}$ graafissa G solmu M mukaan lukien, ja $I, J \in \mathbf{V}$.

Erilliset muuttujajoukot $\mathbf{X}$ ja $\mathbf{Y}$ ovat solmujoukon $\mathbf{Z}$ d-separoimia graafissa G, jos kaikki polut joukosta $\mathbf{X}$ joukkoon $\mathbf{Y}$ ovat solmujoukon $\mathbf{Z}$ d-separoimia graafissa G.

Jos erilliset muuttujajoukot $\mathbf{X}$ ja $\mathbf{Y}$ ovat solmujoukon $\mathbf{Z}$ d-separoimia graafissa G, niin muuttujajoukko $\mathbf{X}$ on riippumaton muuttujajoukosta $\mathbf{Y}$ ehdolla $\mathbf{Z}$ graafissa G jokaisen jakauman P suhteen, jolle G ja P toteuttavat kausaalisen Markov-ehdon. Tämä riippumattomuus ja d-separoituvuus graafissa G voidaan esittää merkinnällä $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z})_G$ (Dawid, 1979).

2.3 Kausaalivaikutukset

Kausaalimalliin kohdistuvat interventiot muokkaavat sen kuvaamien muuttujien välisiä funktionaalisia suhteita. Interventiot määritellään $do(\cdot)$-operaattorilla, joka poistaa tiettyjä funktioita mallista ja korvaa ne vakiofunktioilla vaikuttamatta malliin muilla tavoilla. Jokainen kausaalimalliin M kohdistuva interventio $do(\mathbf{X} = \mathbf{x})$ tuottaa siten *alimallin* $M_{\mathbf{x}} = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}_{\mathbf{x}}, P(\mathbf{U}) \rangle$, missä joukko $\mathbf{F}_{\mathbf{x}}$ saadaan korvaamalla funktio $f_X \in \mathbf{F}$ jokaista muuttujaa $X \in \mathbf{X}$ kohden vakiofunktiolla, joka tuottaa aina intervention $do(\mathbf{X} = \mathbf{x})$ määräämän arvon x. Koska $M_{\mathbf{x}}$ on kausaalimalli, on joukolla $F_{\mathbf{x}}$ yksikäsitteinen ratkaisu, mikä takaa yksikäsitteisen ratkaisun olemassaolon myös jokaiselle muuttujalle $V \in \mathbf{V}$. Tämä tarkoittaa sitä, että jokainen havaittu muuttuja on edelleen esitettävissä ainoastaan havaitsemattomien muuttujien avulla alimallissa $M_{\mathbf{x}}$. Tällöin on mielekästä tarkastella, kuinka joukkoa $\mathbf{V}$ koskevat todennäköisyydet muuttuvat intervention seurauksena.

Määritelmä (Kausaalivaikutus, Shpitser ja Pearl (2006b))**.** Olkoot kausaalimalli $M = \langle \mathbf{U}, \mathbf{V}, \mathbf{F}, P(\mathbf{U}) \rangle$ ja muuttujajoukot $\mathbf{Y}, \mathbf{X} \subset \mathbf{V}$. Intervention $do(\mathbf{X} = \mathbf{x})$ *kausaalivaikutus* joukkoon $\mathbf{Y}$ mallissa M on muuttujajoukon $\mathbf{Y}$ yhteisjakauma alimallissa $M_{\mathbf{x}}$, josta käytetään merkintää $P(\mathbf{Y}|do(\mathbf{X} = \mathbf{x})) = P_{\mathbf{x}}(\mathbf{Y})$.

Jokaista interventiota $do(\mathbf{X} = \mathbf{x})$ kohden tulee olla $P(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$. Tämä rajoite takaa, että jakauma $P_{\mathbf{x}}(\mathbf{V})$ ja sen marginaalijakaumat ovat aina hyvin määritellyt. Rajoite on luonnollinen, sillä ei ole mielekästä pakottaa muuttujajoukon $\mathbf{X}$ arvoja sellaisiksi, joita ei voi havaita. Tällaisen intervention jakaumasta ei voida tehdä päätelmiä aineiston perusteella.

Kirjallisuudessa (esim. Holland, 1986 ja Rubin, 1974) kausaalivaikutus määritellään joskus edellisestä määritelmästä poikkeavalla tavalla. Kausaalivaikutus on luonnollista määritellä keskimääräisenä kausaalivaikutuksena dikotomisille muuttujille Y ja X

$$P(Y = 1|do(X = 1)) - P(Y = 1|do(X = 0)).$$

Jos kausaalivaikutuksella tarkoitetaan juuri lauseketta $P_{\mathbf{x}}(\mathbf{Y})$, on keskimääräinen kausaalivaikutus tällöin kahden kausaalivaikutuksen erotus. Reaaliselle satunnaismuuttujalle Y ja dikotomiselle satunnaismuuttujalle X keskimääräinen kausaalivaikutus määritellään odotusarvojen erotuksena

$$\mathbb{E}[Y|do(X = 1)] - \mathbb{E}[Y|do(X = 0)].$$

Jos sekä Y että X ovat reaalisia, niin keskimääräinen kausaalivaikutus voidaan määritellä derivaattana

$$\frac{d}{dx}\mathbb{E}[Y|do(X = x)],$$

mikäli tämä on olemassa. Jatkossa kausaalivaikutuksella kuitenkin tarkoitetaan intervention määräämää yhteisjakaumaa $P_{\mathbf{x}}(\mathbf{Y})$.

Tehtäessä päätelmiä interventioista, ei muuttujien välillä vallitsevia funktionaalisia suhteita tarvitse aina määrittää tarkasti, eikä tämä aina ole edes mahdollista. Usein riittää, että tiedossa on joitakin kausaalisia suhteita ja tilastollista informaatiota tutkittavasta ilmiöstä. Tällöin nämä oletukset voidaan koota taustalla olevaa kausaalimallia vastaavaksi graafiksi sekä havaintoja koskeviksi todennäköisyyksiksi. Vaarana on kuitenkin se, että kiinnostavaa kausaalivaikutusta ei aina voida määrittää yksikäsitteisesti, sillä sama graafi voi vastata useampaa kuin yhtä kausaalimallia, ja on mahdollista, että kaikkia kiinnostavia muuttujia ei ole pystytty mittaamaan. Seuraava määritelmä karakterisoi kausaalimallista tehtyjen oletusten riittävyyden.

Määritelmä (Kausaalivaikutuksen identifioituvuus, Shpitser ja Pearl (2006b) 2)**.** Olkoot graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$ ja muuttujajoukot $\mathbf{Y}$ ja $\mathbf{X}$. Intervention $do(\mathbf{X} = \mathbf{x})$ kausaalivaikutus muuttujajoukkoon $\mathbf{Y}$, jolle $\mathbf{Y} \cap \mathbf{X} = \emptyset$, on *identifioituva* graafissa G jos $P^1_{\mathbf{x}}(\mathbf{Y}) = P^2_{\mathbf{x}}(\mathbf{Y})$ jokaiselle parille kausaalimalleja M^1 ja M^2, joille $P^1(\mathbf{V}) = P^2(\mathbf{V})$ ja $P^1(\mathbf{x}|Pa(\mathbf{X})_G \setminus \mathbf{X}) > 0$.

Määritelmän avulla ei voida yleensä suoraan todeta kausaalivaikutuksen identifioituvuutta, sillä on harvoin mahdollista varmistaa, että $P^1_{\mathbf{x}}(\mathbf{Y}) = P^2_{\mathbf{x}}(\mathbf{Y})$ jokaiselle parille kausaalimalleja M^1 ja M^2, joiden havaittujen muuttujien yhteisjakaumat ovat samat. Määritelmän avulla voidaan kuitenkin todeta identifioitumattomuus määrittelemällä kausaalimallit M^1 ja M^2, joille $P^1(\mathbf{V}) = P^2(\mathbf{V})$, mutta $P^1_{\mathbf{x}}(\mathbf{Y}) \neq P^2_{\mathbf{x}}(\mathbf{Y})$. Identifioituvuus määritellään joskus yllä olevasta määritelmästä poikkeavalla tavalla (esim. Angrist, Imbens ja Rubin, 1996). Jatkossa identifioituvuutta käsitellään kuitenkin määritelmän mukaisesti yleisellä tasolla.

Seuraava esimerkki perustuu konstruktioon, jonka esittivät Shpitser ja Pearl (2006b). Tarkastellaan kuvan 5 graafia G ja muuttujan X kausaalivaikutusta muuttujaan Y. Muuttujan Y voidaan ajatella olevan esimerkiksi jokin kiinnostava vastemuuttuja ja muuttujan X jokin prediktori. Lisäksi vasteeseen ja prediktoriin vaikuttaa jokin havaitsematon satunnaismuuttuja U. Shpitser ja Pearl toteavat, että havaittujen muuttujien reunajakaumien positiivisuus voidaan aina taata esimerkin tilanteessa, mutta eivät osoita tätä suoraan. Laajennetaan konstruktiota siten, että $P(X) > 0$ kaikilla muuttujan X arvoilla ja $P(Y) > 0$ kaikilla muuttujan Y arvoilla kummassakin mallissa.

Kuva 5: Yksinkertainen graafi, jossa kausaalivaikutus $P_x(y)$ ei identifioidu

Määritellään kausaalimallit M^1 ja M^2 seuraavasti: kummassakin mallissa asetetaan $U \sim \text{Tas}(\{0, 1\})$ ja $f_X(u) = u$, joten $P^1(X = 1) = P^2(X = 1) = 0.5$. Mallissa M^1 funktio $f_Y(u, x)$ määritellään siten, että

$$f_Y(u, x) = \begin{cases} (u + 2x) \bmod 2, & \text{kun } u = 1 \\ (u + x) \bmod 2, & \text{kun } u = 0. \end{cases}$$

Koska muuttujan X arvo määräytyy deterministisesti satunnaismuuttujan U arvosta, niin muuttujan Y arvot määräytyvät mallissa M^1 seuraavasti: jos $u = 1$, niin

$$f_Y(u, x) = (u + 2x) \bmod 2 = (u + 2f_X(u)) \bmod 2 = (u + 2u) \bmod 2 = 1.$$

Jos $u = 0$, niin

$$f_Y(u, x) = (u + x) \bmod 2 = (u + f_X(u)) \bmod 2 = (u + u) \bmod 2 = 0.$$

Siis $P^1(Y = 1) = 0.5$. Mallissa M^2 vastaava funktio $f_Y(u)$ määritellään siten, että $f_Y(u) = u$. Tällöin $P^2(Y = 1) = 0.5$.

Johdetaan seuraavaksi havaittujen muuttujien yhteisjakaumat kummassakin mallissa. Mallissa M^2 muuttujien X ja Y arvot on asetettu suoraan

satunnaismuuttujan U arvoksi, joten $P^2(X, Y) = 0.5$ jos $X = Y$. Muussa tapauksessa $P^2(X, Y) = 0$. Edellä osoitettiin, että muuttuja Y saa myös mallissa M^1 satunnaismuuttujan U arvon. Koska muuttujan X arvo on asetettu satunnaismuuttujan U arvoksi molemmissa malleissa, on $P^1(X, Y) = 0.5$ jos $X = Y$. Muussa tapauksessa $P^1(X, Y) = 0$.

Havaittujen muuttujien yhteisjakaumille pätee siis $P^1(X, Y) = P^2(X, Y)$. Jos satunnaismuuttujaan X kohdistetaan nyt interventio $do(X = x)$, niin se muuntaa funktion $f_X(u) = u$ vakiofunktioksi $f_x(u) = x$ kummassakin mallissa. Interventio ei vaikuta satunnaismuuttujan Y jakaumaan mallissa M^2, sillä muuttujan Y arvot määräytyvät suoraan muuttujan U arvoista, jolloin $P^2_x(Y = 1) = 0.5$. Mallissa M^1 interventio puolestaan muuntaa muuttujan Y jakaumaa. Jos tehty interventio oli $do(X = 1)$, niin

$$f_Y(u, 1) = \begin{cases} (u + 2) \bmod 2, & \text{kun } u = 1 \\ (u + 1) \bmod 2, & \text{kun } u = 0. \end{cases}$$

Siis Y saa arvon 1 riippumatta muuttujan U arvosta. Jos tehty interventio oli $do(X = 0)$, niin

$$f_Y(u, 0) = \begin{cases} u \bmod 2, & \text{kun } u = 1 \\ u \bmod 2, & \text{kun } u = 0, \end{cases}$$

eli $f_Y(u, 0) = u$. Interventiot $do(X = 1)$ ja $do(X = 0)$ ovat hyvin määritellyt kummassakin kausaalimallissa, sillä sekä $P^1(X = x) > 0$ että $P^2(X = x) > 0$ kaikilla muuttujan X arvoilla x, ja havaittujen muuttujien yhteisjakaumat ovat samat. Kausaalivaikutukset eivät kuitenkaan ole samat, sillä

$$P^1_1(Y = 1) = 1 \neq 0.5 = P^2_1(Y = 1),$$

jolloin kausaalivaikutus $P_x(Y)$ ei ole identifioituva graafissa G.

3 Kausaalilaskenta

Kausaalivaikutuksista on haastavaa tehdä päätelmiä käyttäen ainoastaan identifioituvuuden määritelmää. Kausaalivaikutusten identifioituvuutta voidaan kuitenkin lähestyä myös interventioiden kautta. Judea Pearl määritteli edellisessä luvussa esitellyn $do(\cdot)$-operaattorin tulkinnan pohjalta päättelysäännöstön, jota kutsutaan kausaalilaskennaksi (*do calculus*, ks. Pearl, 2009, sivut 85-86 ja Pearl, 1995).

Kausaalilaskennan tavoitteena on esittää kausaalivaikutuksen $P_{\mathbf{x}}(\mathbf{y})$ lauseke ainoastaan havaittuja muuttujia koskevien todennäköisyyksien avulla. Kausaalivaikutus on identifioituva, mikäli tällainen esitystapa on mahdollista saavuttaa päättelysääntöjen iteratiivisella soveltamisella. Tämä tulos on suora seuraus identifioituvuuden määritelmästä, sillä havaittujen muuttujien jakaumat oletetaan yhtäsuuriksi kaikille tarkasteltaville kausaalimalleille.

3.1 Laskusäännöt

Olkoot $\mathbf{X}$, $\mathbf{Y}$ ja $\mathbf{Z}$ mielivaltaisia pareittain erillisiä solmujoukkoja kausaalimallia M vastaavassa graafissa G. Merkinnällä $G_{\overline{\mathbf{X}}}$ tarkoitetaan graafia, joka saadaan graafista G poistamalla kaikki solmujoukkoon $\mathbf{X}$ saapuvat särmät. Merkinnällä $G_{\underline{\mathbf{X}}}$ tarkoitetaan graafia, joka saadaan graafista G poistamalla kaikki solmujoukosta $\mathbf{X}$ lähtevät särmät. Merkinnällä $G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}$ tarkoitetaan graafia, joka saadaan graafista G poistamalla kaikki solmujoukkoon $\mathbf{X}$ saapuvat särmät ja kaikki solmujoukosta $\mathbf{Z}$ lähtevät särmät. Olkoon nyt P kausaalimallin M havaittujen ja havaitsemattomien muuttujien yhteisjakauma. Tällöin seuraavat säännöt ovat voimassa (Pearl, 1995):

1. Havaintojen lisääminen ja poistaminen

$$P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z}),$$

   jos $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}$ eli jos muuttujat $\mathbf{Y}$ ovat riippumattomia muuttujista $\mathbf{Z}$ ehdolla $\mathbf{X}$ ja $\mathbf{W}$ graafissa G, josta on poistettu solmujoukkoon $\mathbf{X}$ saapuvat särmät.

2. Toiminnan ja havainnon vaihtaminen

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{z},\mathbf{w}),$$

   jos $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{\mathbf{Z}}}}$ eli jos muuttujat $\mathbf{Y}$ ovat riippumattomia muuttujista $\mathbf{Z}$ ehdolla $\mathbf{X}$ ja $\mathbf{W}$ graafissa G, josta on poistettu solmujoukkoon $\mathbf{X}$ saapuvat särmät ja solmujoukosta $\mathbf{Z}$ lähtevät särmät.

3. Toiminnan lisääminen ja poistaminen

$$P_{\mathbf{x},\mathbf{z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{x}}(\mathbf{y}|\mathbf{w}),$$

jos $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X}, \mathbf{W})_{G_{\overline{\mathbf{X}}, Z(\overline{\mathbf{W}})}}$ eli jos muuttujat $\mathbf{Y}$ ovat riippumattomia muuttujista $\mathbf{Z}$ ehdolla $\mathbf{X}$ ja $\mathbf{W}$ graafissa G, josta on poistettu solmujoukkoon $\mathbf{X}$ saapuvat särmät ja solmujoukkoon $Z(\mathbf{W})$ saapuvat särmät, missä

$$Z(\mathbf{W}) = \mathbf{Z} \setminus An(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$$

eli $Z(\mathbf{W})$ sisältää joukon $\mathbf{Z}$ ne solmut, jotka eivät kuulu joukkoon $\mathbf{W}$ ja eivät ole minkään joukon $\mathbf{W}$ solmun havaittuja esivanhempia graafissa G, josta on poistettu solmujoukkoon $\mathbf{X}$ saapuvat särmät.

Sääntö 1 yhdistää d-separoituvuuden ja intervention $do(\mathbf{X} = \mathbf{x})$ synnyttämän todennäköisyysjakauman toisiinsa. Graafissa G vallitsevat ehdolliset riippumattomuudet säilyvät, sillä muuttujajoukkoon $\mathbf{X}$ liittyvien funktioiden korvaaminen vakiofunktioilla ei synnytä uusia riippuvuuksia muiden muuttujien välille. Kun sääntö 2 on voimassa, niin interventiolla $do(\mathbf{Z} = \mathbf{z})$ on sama vaikutus joukkoon $\mathbf{Y}$ kuin passiivisella havainnolla $\mathbf{Z} = \mathbf{z}$. Säännön 3 avulla voidaan lisätä tai poistaa interventioita vaikuttamatta muuttujaa $\mathbf{Y}$ koskeviin todennäköisyyksiin.

Kausaalilaskennan säännöt voidaan johtaa hyödyntäen d-separoituvuutta edellä esiintyvissä graafeissa sekä soveltamalla $do(\cdot)$-operaattorin määritelmää. Pearl esitti todistukset näille kolmelle säännölle (ks. Pearl, 1995). Kausaalilaskenta on myös osoitettu täydelliseksi, eli sen avulla voidaan johtaa kaikkien identifioituvien kausaalivaikutusten jakaumat (Shpitser ja Pearl, 2006b, Huang ja Valtorta, 2006).

3.2 Esimerkki kausaalilaskennasta

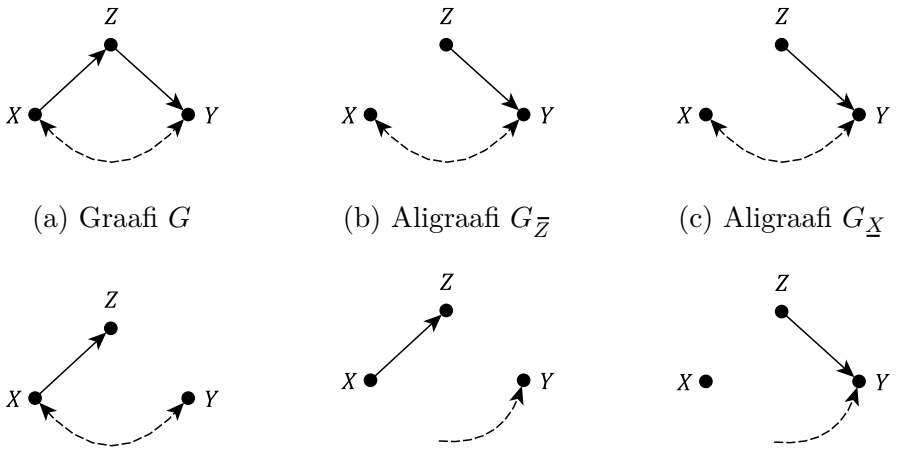
Tarkastellaan kausaalilaskennan laskusääntöjen soveltamista käytännössä yksinkertaisen esimerkin avulla. Olkoot graafi G kuten kuvassa 6(a) ja kiinnostuksen kohteena kausaalivaikutus $P_x(y)$.

Solmu Y toimii käänteisenä haarukkana solmuja X ja Z yhdistävällä polulla kuvan 6(c) graafissa $G_{\underline{X}}$, jolloin muuttujat X ja Z ovat d-separoituneita kyseisessä graafissa. Siis $(X \perp\!\!\!\perp Z)_{G_{\underline{X}}}$, jolloin säännön 2 nojalla

$$P_x(z) = P(z|x). \tag{1}$$

Koska kuvan 6(b) graafi $G_{\overline{Z}}$ on sama kuin kuvan 6(c) graafi $G_{\underline{X}}$, niin $(X \perp\!\!\!\perp Z)_{G_{\overline{Z}}}$ ja säännön 3 nojalla $P_z(x) = P(x)$. Edelleen koska solmu X d-separoi kaikki polut solmusta Y solmuun Z kuvan 6(d) graafissa $G_{\underline{Z}}$, niin $(Y \perp\!\!\!\perp Z|X)_{G_{\underline{\mathbf{Z}}}}$ ja säännön 2 nojalla $P_z(y|x) = P(y|z, x)$. Tällöin pätee

$$P_z(y) = \sum_x P_z(y|x)P_z(x) = \sum_x P(y|z, x)P(x). \tag{2}$$



(a) Graafi G (b) Aligraafi $G_{\overline{Z}}$ (c) Aligraafi $G_{\underline{X}}$

(d) Aligraafi $G_{\underline{Z}}$ (e) Aligraafi $G_{\overline{X},\underline{Z}}$ (f) Aligraafi $G_{\overline{X},\overline{Z}}$

Kuva 6: Esimerkki kausaalilaskennassa käytettävistä graafioperaatioista

Tarkastellaan seuraavaksi kuvan 6(e) graafia $G_{\overline{X},\underline{Z}}$. Tässä graafissa solmujen Z ja Y välillä ei ole polkuja, jolloin voidaan todeta, että solmu X d-separoi kaikki polut kyseisten solmujen välillä. Vastaavasti solmujen Y ja X välillä ei ole polkuja kuvan 6(f) graafissa $G_{\overline{X},\overline{Z}}$, jolloin solmun Z voidaan todeta d-separoivan kyseiset solmut. Siis $(Y \perp\!\!\!\perp Z|X)_{G_{\overline{X},\underline{Z}}}$ ja $(Y \perp\!\!\!\perp X|Z)_{G_{\overline{X},\overline{Z}}}$, jolloin sääntöjen 2 ja 3 nojalla

$$P_x(y|z) = P_{x,z}(y) = P_z(y). \tag{3}$$

Yhdistämällä kohdat (2) ja (3) saadaan

$$P_x(y|z) = \sum_x P(y|z,x)P(x). \tag{4}$$

Sijoittamalla kohdat (1) ja (4) kausaalivaikutuksen $P_x(y)$ lausekkeeseen saadaan

$$P_x(y) = \sum_z P_x(y|z)P_x(z) = \sum_z \Big[\sum_x P(y|z,x)P(x)\Big]P(z|x).$$

On syytä huomata, että muuttuja x esiintyy lausekkeessa useammassa kuin yhdessä roolissa. Muuttuja x toimii summamuuttujana sulkulausekkeessa sekä ehdollistavana muuttujana lausekkeessa $P(z|x)$. Kyseessä on kuitenkin kaksi eri muuttujaa.

4 Kausaalilaskennan algoritmi

Vaikka kausaalivaikutus olisi identifioituva, eivät kausaalilaskennan säännöt itsessään takaa, että niiden avulla pystyttäisiin esittämään tämän vaikutuksen jakauma pelkästään havaittujen todennäköisyyksien avulla. Identifioituvan kausaalivaikutuksen tapauksessa ei ole myöskään itsestään selvää missä järjestyksessä laskusääntöjä tulisi soveltaa kausaalivaikutuksen jakauman johtamiseksi kyseessä olevan kausaalimallin havaittujen muuttujien yhteisjakaumasta $P(\mathbf{V})$.

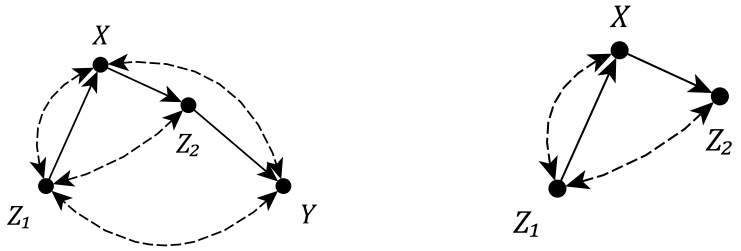
Näistä rajoitteista huolimatta on identifioituvuuden määrittämiseksi johdettu lukuisia tuloksia, joista tässä luvussa keskitytään Shpitserin ja Pearlin (2006) kehittämään kausaalilaskennan algoritmiin. Algoritmin avulla voidaan todeta minkä tahansa kausaalivaikutuksen identifioituvuus, minkä lisäksi algoritmi myös tuottaa vaikutuksen jakauman lausekkeen tilanteessa, jossa vaikutus on identifioituva.

4.1 Määritelmiä

Algoritmin käsittelemiseksi tarvitaan lukuisia kausaalimalleihin ja suunnattuihin silmukattomiin graafeihin liittyviä määritelmiä, joiden avulla vaikutusten identifioituvuus voidaan todeta tietyissä erityistilanteissa.

Määritelmä (Indusoitu aligraafi)**.** Olkoot graafit $H = \langle \mathbf{W}, \mathbf{F} \rangle$ ja $G = \langle \mathbf{V}, \mathbf{E} \rangle$ sellaisia, että $\mathbf{W} \subset \mathbf{V}$. Jos jokaisen solmuparin $X, Y \in \mathbf{W}$ välillä on särmä graafissa H täsmälleen silloin kun niiden välillä on saman suuntainen särmä graafissa G, niin H on *solmujoukon* $\mathbf{W}$ *indusoima aligraafi* ja merkitään $H = G[\mathbf{W}]$.

Indusoitujen aligraafien avulla voidaan helposti määrittää uusia graafeja pelkästään tietyn solmujoukon perusteella. Esimerkiksi kuvan 7(a) graafin G solmuista X, Z_1 ja Z_2 on muodostettu indusoitu aligraafi kuvassa 7(b).



(a) Graafi G

(b) Solmujoukon $\{X, Z_1, Z_2\}$ indusoima aligraafi $G[\{X, Z_1, Z_2\}]$

Kuva 7: Indusoidun aligraafin määritelmää havainnollistava esimerkki

Tärkein määritelmistä on kuitenkin C-komponentti (*confounded component*).

Määritelmä (C-komponentti, Shpitser ja Pearl (2006b) 3)**.** Olkoon graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$. Jos on olemassa sellainen joukko $\mathbf{B} \subset \mathbf{E}$, että $\mathbf{B}$ sisältää vain kaksisuuntaisia särmiä, ja graafi $\langle \mathbf{V}, \mathbf{B} \rangle$ on yhtenäinen, niin G on *C-komponentti*.

C-komponentteja ovat esimerkiksi molemmat kuvan 7 graafeista, mutta kuvan 6(a) graafi ei ole. Vaikka graafi ei olisikaan C-komponentti, voidaan sen aligraafeista aina löytää ainakin yksi C-komponentti. Esimerkiksi kaikista yksittäisistä solmuista muodostetut aligraafit ovat aina C-komponentteja. Usein on kuitenkin mielenkiintoisempaa selvittää, kuinka annettu graafi voidaan jakaa mahdollisimman suuriin C-komponentteihin.

Määritelmä (Maksimaalinen C-komponentti)**.** Olkoot graafi G ja C-komponentti $C = \langle \mathbf{V}, \mathbf{E} \rangle$, $C \subset G$. C-komponentti C on *maksimaalinen* (suhteessa graafiin G) jos kaikille graafin G kaksisuuntaisista särmistä muodostuneille poluille H, jotka sisältävät ainakin yhden joukon $\mathbf{V}$ solmun, pätee $H \subset C$.

Tian (2002) osoitti, että maksimaalisten C-komponenttien avulla voidaan aina faktoroida graafin G muuttujien yhteisjakauma $P(\mathbf{V})$, jolloin jokainen tulon termeistä vastaa yhtä maksimaalista C-komponenttia. Tämä ominaisuus osoittautui merkittäväksi algoritmin kannalta, sillä kausaalivaikutuksen jakauma voidaan jakaa rekursiivisesti yhä yksinkertaisemmiksi lausekkeiksi.

Jos graafi G ei ole C-komponentti, niin se voidaan aina jakaa yksikäsitteiseksi joukoksi $C(G)$ aligraafeja, joista jokainen on maksimaalinen C-komponentti. Tämä tulos seuraa siitä, että kahden solmun välillä on kaksisuuntaisista särmistä muodostunut polku graafissa G jos ja vain jos solmut kuuluvat samaan maksimaaliseen C-komponenttiin, mikä puolestaan on suora seuraus maksimaalisen C-komponentin määritelmästä. Graafin G kaksisuuntaisista särmistä muodostuneet polut määrittävät siten myös graafin maksimaaliset C-komponentit.

Erikoistapaus C-komponentista on C-puu. C-puut liittyvät läheisesti suoriin vaikutuksiin eli kausaalivaikutuksiin, jotka ovat muotoa $P_{Pa(Y)}(Y)$.

Määritelmä (C-puu, Shpitser ja Pearl (2006b) 4)**.** Olkoon graafi G sellainen C-komponentti, että sen jokaisella havaitulla solmulla on korkeintaan yksi lapsi. Jos on olemassa solmu Y, jolle $G[An(Y)_G] = G$, niin G on *Y-juurtunut C-puu*.

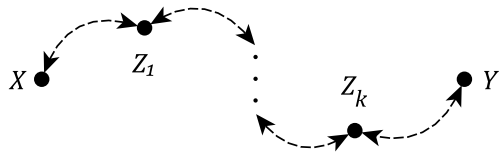
C-puiden ja C-komponenttien avulla on mahdollista määrittää lukuisia kausaalivaikutuksia, jotka kohdistuvat yhteen muuttujaan. C-metsä yleistää C-puun useamman kuin yhden muuttujan tilanteeseen, jossa graafin G *juurijoukko*, eli joukko $\{X \in G \mid De(X)_G \setminus \{X\} = \emptyset\}$ koostuu useammasta kuin yhdestä solmusta.

Määritelmä (C-metsä, Shpitser ja Pearl (2006b) 5)**.** Olkoot G graafi ja $\mathbf{Y}$ sen juurijoukko. Jos graafi G on C-komponentti, jonka jokaisella havaitulla solmulla on korkeintaan yksi lapsi, niin G on *$\mathbf{Y}$-juurtunut C-metsä*.

Molemmat kuvan 7 C-komponenteista ovat myös C-metsiä, sillä jokaisella niiden havaituista solmuista on täsmälleen yksi lapsi. Lisäksi C-komponenttien juurijoukot muodostuvat solmusta Y. C-metsät ovat sidoksissa yleisten kausaalivaikutusten laskentaan eli vaikutuksiin, jotka ovat muotoa $P_{\mathbf{x}}(\mathbf{Y})$. Tällaisten kausaalivaikutusten identifioituvuutta voidaan tarkastella erityisen kahdesta C-metsästä muodostuvan graafiparin avulla.

Määritelmä (Pensasaita, Shpitser ja Pearl (2006b) 6)**.** Olkoot $G = \langle \mathbf{V}, \mathbf{E} \rangle$ graafi ja $\mathbf{X}, \mathbf{Y} \subset \mathbf{V}$ erilliset muuttujajoukot. Jos on olemassa kaksi $\mathbf{R}$-juurtunutta C-metsää $F = \langle \mathbf{V}_F, \mathbf{E}_F \rangle$ ja $F' = \langle \mathbf{V}_{F'}, \mathbf{E}_{F'} \rangle$ siten, että $\mathbf{V}_F \cap \mathbf{X} \neq \emptyset$, $\mathbf{V}_{F'} \cap \mathbf{X} = \emptyset$, $F' \subset F$, ja $\mathbf{R} \subset G[An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}]$, niin C-metsät F ja F' muodostavat *pensasaidan* kausaalivaikutukselle $P_{\mathbf{x}}(\mathbf{y})$ graafissa G.

Kuvan 5 graafi G sisältää pensasaidan kausaalivaikutukselle $P_x(y)$. Määritelmän mukaiset $\mathbf{R}$-juurtuneet C-metsät ovat $F = G[\{X, Y\}]$ ja $F' = G[Y]$, missä $\mathbf{R} = \{Y\}$. Pensasaidat ovat huomattava rakenne, sillä ne yleistävät tiettyjä erikoistapauksia koskevia tuloksia identifioituvuudelle. Eräs esimerkki tällaisesta erikoistapauksesta on kahden muuttujan välisiä kausaalivaikutuksia $P_x(y)$ koskeva tulos, jonka johtivat Tian ja Pearl (2002). He osoittivat, että $P_x(y)$ ei ole identifioituva jos ja vain jos solmu Y on solmun X lapsi, ja on olemassa kaksisuuntaisista särmistä muodostunut polku solmusta X solmuun Y. Tarkastellaan kuvan 8 graafia $H = \langle \mathbf{V}, \mathbf{E} \rangle$, joka sisältää solmut X ja Y sekä näiden välisen kaksisuuntaisista särmistä muodostuvan polun, joka koostuu lisäksi solmuista $\{Z_1, \ldots, Z_k\}$. Tällöin C-metsät H ja $H[\mathbf{V} \setminus \{X\}]$ muodostavat pensasaidan kausaalivaikutukselle $P_x(Y, Z_1, \ldots, Z_k)$.



Kuva 8: Polku H

Shpitser ja Pearl (2006b) osoittivat, että jos graafi G sisältää pensasaidan kausaalivaikutukselle $P_{\mathbf{x}}(\mathbf{y})$, niin kyseinen vaikutus ei ole identifioituva.

4.2 Algoritmi

Edellä esiteltyjä määritelmiä hyödyntäen voidaan nyt määritellä algoritmi 1, joka karakterisoi täydellisesti yleisten kausaalivaikutusten identifioituvuuden. Shpitser ja Pearl (2006b) osoittivat, että algoritmin 1 palauttama lauseke kausaalivaikutukselle $P_{\mathbf{x}}(\mathbf{y})$ on aina oikein, jos kyseinen vaikutus on identifioituva. He osoittivat myös, että jos algoritmin toiminta keskeytyy rivillä viisi, niin alkuperäinen graafi G sisältää pensasaidan, joka estää kausaalivaiku-

tuksen identifioituvuuden. Pensasaidan olemassaolo on siis yhtäpitävää identifioitumattomuuden kanssa. Koska algoritmi 1 koostuu ainoastaan tunnetuista todennäköisyys- ja kausaalilaskennan sääntöihin pohjautuvista toimenpiteistä, niin edellinen tulos osoittaa myös, että kausaalilaskennan säännöillä voidaan johtaa kaikkien identifioituvien kausaalivaikutusten jakaumat. Kaikki esiintyvät muuttujat oletetaan diskreeteiksi, mutta algoritmia voidaan soveltaa myös tilanteessa, jossa osa muuttujista on jatkuvia, kun niihin liittyvät summat korvataan integraaleilla.

Algoritmin on kyettävä käsittelemään syötteenä annetun graafin solmuja iteratiivisesti, joten solmuille on asetettava jokin mielekäs järjestys. Esimerkiksi graafin särmien suunta on otettava tässä järjestyksessä huomioon, ja järjestys on aina pystyttävä muodostamaan mille tahansa annetulle graafille. Topologisella järjestyksellä on nämä ominaisuudet.

Määritelmä (Topologinen järjestys)**.** Suunnatun silmukattoman graafin $G = \langle \mathbf{V}, \mathbf{E} \rangle$ *topologinen järjestys* π on sen solmujen järjestys, jossa joko $X > Y$ tai $Y > X$ kaikille pareille solmuja $X, Y \in \mathbf{V}$, $X \neq Y$. Lisäksi yksikään solmu ei voi olla suurempi kuin jälkeläisensä. Toisin sanoen $X < Y$ järjestyksessä π jos ja vain jos solmu X on solmun Y esivanhempi graafissa G.

Jokaisella suunnatulla silmukattomalla graafilla on aina olemassa ainakin yksi topologinen järjestys, mutta järjestyksiä voi olla myös useita. Topologinen järjestys voidaan muodostaa esimerkiksi etsimällä graafista ensin kaikki solmut, joilla ei ole lainkaan esivanhempia, ja asettamalla nämä johonkin mielivaltaiseen järjestykseen. Tämän jälkeen etsitään kaikki solmut, joilla ei ole lainkaan esivanhempia aikaisemmin järjestettyjä solmuja lukuun ottamatta, ja järjestetään nämä taas keskenään mielivaltaisesti. Lisäksi asetetaan, että edellisen kohdan suurin solmu on pienempi kuin viimeisimmän kohdan pienin solmu. Näin jatketaan, kunnes kaikki solmut on järjestetty.

Topologisella järjestyksellä voidaan indeksoida graafin G ja sen aligraafien solmut, mitä hyödynnetään algoritmin 1 riveillä neljä, kuusi ja seitsemän. Merkinnällä $V_\pi^{(i-1)}$ tarkoitetaan niitä graafin G havaittuja solmuja, jotka ovat pienempiä kuin solmu V_i topologisessa järjestyksessä π. Mikä tahansa graafin G topologinen järjestys on myös topologinen järjestys mille tahansa sen aligraafille. Uutta järjestystä ei siis tarvitse etsiä jokaiselle muodostetulle aligraafille erikseen, vaan järjestys voidaan kiinnittää ennen algoritmin soveltamista.

Algoritmi 1 Intervention $do(\mathbf{X} = \mathbf{x})$ kausaalivaikutus muuttujajoukkoon $\mathbf{Y}$.

SYÖTE: Arvojoukot $\mathbf{x}$ ja $\mathbf{y}$, yhteisjakauma $P(\mathbf{v})$ ja suunnattu silmukaton graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$. G ja P toteuttavat kausaalisen Markov-ehdon.
TULOSTE: Lauseke jakaumalle $P_{\mathbf{x}}(\mathbf{y})$ tai **HYLKÄÄ**(F, F').

funktio ID$(\mathbf{y}, \mathbf{x}, P, G)$
1: **jos** $\mathbf{x} = \emptyset$, **niin**
  palauta $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.
2: **jos** $\mathbf{V} \neq An(\mathbf{Y})_G$, **niin**
  palauta ID$(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G])$.
3: **olkoon** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
  jos $\mathbf{W} \neq \emptyset$, **niin**
  palauta ID$(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.
4: **jos** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, **niin**
  palauta $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k}$ **ID**$(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.
  jos $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$, **niin**
5: **jos** $C(G) = \{G\}$, **niin**
    aiheuta HYLKÄÄ$(G, G[\mathbf{S}])$.
6: **jos** $G[\mathbf{S}] \in C(G)$, **niin**
    palauta $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.
7: **jos** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ siten, että $G[\mathbf{S}'] \in C(G)$, **niin**
    palauta ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

Algoritmi 1 on siinä mielessä yksinkertainen, että jokaisella rekursiokierroksella suoritus etenee tasan yhdelle riville. Tämä on helppo nähdä siitä, että kun riviä koskeva ehto on tarkastettu, niin suoritetaan joko **palauta**- tai **HYLKÄÄ**-komento. Jos $\mathbf{x} = \emptyset$ rivillä yksi, niin kausaalivaikutuksen sijaan lasketaankin reunajakaumaa $P(\mathbf{y})$, joka saadaan tunnetusti marginalisoimalla kaikkien muuttujien yhteisjakaumasta $P(\mathbf{V})$. Rivillä kaksi rajoitutaan tarkastelemaan solmujoukon $\mathbf{Y}$ esivanhempia graafissa G, sillä kausaalinen Markov-ehto takaa tähän tarvittavat ehdolliset riippumattomuudet. Rivillä kolme lisätään interventioita alkuperäiseen vaikutukseen, mikä on sallittua kausaalilaskennan kolmannen säännön nojalla, sillä $(\mathbf{Y} \perp\!\!\!\perp \mathbf{W} | \mathbf{X})_{G_{\overline{\mathbf{X}}, \overline{\mathbf{W}}}}$.

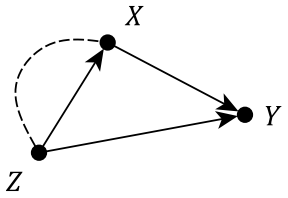
Rivillä neljä määritetään graafin $G[\mathbf{V} \setminus \mathbf{X}]$ maksimaaliset C-komponentit ja hyödynnetään niiden faktorointiominaisuutta. Jos C-komponentteja on enemmän kuin yksi, niin alkuperäisen vaikutuksen määrittämiseksi on laskettava uusi kausaalivaikutus jokaista C-komponenttia kohden. Jos C-komponentteja on vain yksi, niin edetään jollekin riveistä viisi, kuusi tai seitsemän.

Jos algoritmi 1 aiheuttaa **HYLKÄÄ**-komennon rivillä viisi, niin alkuperäinen graafi G sisältää viimeisimmän rekursiotason graafien G ja $G[\mathbf{S}]$ muodostaman ongelmallisen pensasaidan, minkä seurauksena kausaalivaikutus ei ole identifioituva, ja suoritus keskeytetään. Jos suoritusta ei keskeytetä, niin

määritetään onko $G[\mathbf{S}]$ jokin graafin G maksimaalisista C-komponenteista. Jos on, niin suoritus etenee alimmalle rekursiotasolle rivillä kuusi. Muussa tapauksessa interventio voidaan rajata solmujen $\mathbf{X}$ ja $\mathbf{S}'$ leikkausjoukkoon rivillä seitsemän.

4.3 Algoritmin toiminta

Tarkastellaan algoritmin 1 toimintaa aluksi yksinkertaisen esimerkin kautta. Olkoot graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$ kuten kuvassa 9 ja kiinnostuksen kohteena kausaalivaikutus $P_x(y)$, joka pyritään identifioimaan yhteisjakaumasta $P(X, Y, Z)$. Graafin G solmuille on olemassa vain yksi mahdollinen topologinen järjestys, joka on $Z < X < Y$.



Kuva 9: Yksinkertainen graafi, jossa kausaalivaikutus $P_x(y)$ on identifioituva

Nähdään, että $\mathbf{x} \neq \emptyset, \mathbf{V} = An(Y)_G$ ja $\mathbf{W} = \emptyset$, joten kolme ensimmäistä riviä sivuutetaan ja päädytään riville neljä, sillä

$$C(G[\mathbf{V} \setminus \mathbf{X}]) = C(G[\{Z, Y\}]) = \{G[Z], G[Y]\}.$$

Koska $\mathbf{v} \setminus (\{y\} \cup \{x\}) = \{z\}$, niin alkuperäisen vaikutuksen määrittämiseksi on nyt identifioitava kaksi uutta kausaalivaikutusta seuraavassa lausekkeessa:

$$\sum_z P_{x,y}(z) P_{x,z}(y).$$

Tarkastellaan aluksi termiä $P_{x,y}(z)$. Koska $\mathbf{V} \neq An(Z)_G$, niin päädytään riville kaksi, jonka mukaan solmut, jotka eivät ole solmun Z esivanhempia, voidaan sivuuttaa. Koska $\{X, Y\} \cap An(Z)_G = \{X, Y\} \cap \{Z\} = \emptyset$, niin saadaan $P_{x,y}(z) = P(z)$. Termiä $P_{x,z}(y)$ määritettäessä päädytään riville kuusi, sillä

$$C(G[\mathbf{V} \setminus \{X, Z\}]) = C(G[Y]) = \{G[Y]\},$$

ja $G[Y]$ on yksi graafin G maksimaalisista C-komponenteista. Kausaalivaikutus $P_{x,z}(y)$ saa ehdollisen jakauman muodon:

$$P_{x,z}(y) = P(y|x, z).$$

Yhdistämällä tulokset saadaan lauseke alkuperäiselle kausaalivaikutukselle:

$$P_x(y) = \sum_z P(y|x, z) P(z).$$

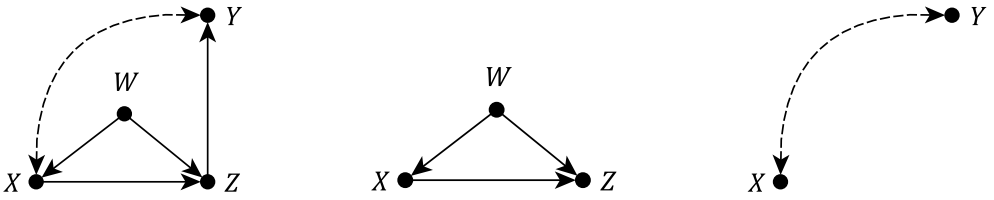
Tarkastellaan algoritmin 1 toimintaa edellistä monimutkaisemmassa tilanteessa. Olkoot graafi $G = \langle \mathbf{V}, \mathbf{E} \rangle$ kuten kuvassa 10(a) ja kiinnostuksen kohteena kausaalivaikutus $P_x(y)$, joka pyritään identifioimaan yhteisjakaumasta $P(X, Y, Z, W)$. Graafin G solmuille on olemassa vain yksi mahdollinen topologinen järjestys, joka on $W < X < Z < Y$. Selvästi $\mathbf{x} \neq \emptyset, \mathbf{V} = An(Y)_G$ ja $\mathbf{W} = \emptyset$, joten kolme ensimmäistä riviä sivuutetaan ja päädytään lopulta riville neljä, sillä

$$C(G[\mathbf{V} \setminus \{X\}]) = \{G[W], G[Z], G[Y]\}.$$

Koska $\mathbf{v} \setminus (\{y\} \cup \{x\}) = \{w, z\}$, niin alkuperäisen vaikutuksen määrittämiseksi on nyt identifioitava kolme uutta kausaalivaikutusta seuraavassa lausekkeessa:

$$\sum_{w,z} P_{x,z,y}(w) P_{w,x,y}(z) P_{w,x,z}(y).$$

Tarkastellaan tulon ensimmäistä termiä. Koska $\mathbf{V} \neq An(W)_G$, niin päädytään riville kaksi, jonka mukaan solmut, jotka eivät ole solmun W esivanhempia, voidaan sivuuttaa.



(a) Graafi G (b) Aligraafi $G[An(Z)_G]$ (c) Aligraafi $G[\mathbf{S}']$

Kuva 10: Graafi G ja sen aligraafeja

Tämän seurauksena ensimmäinen termi sievenee muotoon $P(w)$. Myös tulon toista termiä laskettaessa päädytään riville kaksi, jonka mukaan

$$P_{w,x,y}(z) = P_{w,x}(z)$$

aligraafissa, joka muodostuu solmun Z esivanhemmista kuvassa 10(b). Koska

$$C(G[An(Z)_G \setminus \{W, X\}]) = \{G[Z]\}$$

ja koska

$$G[Z] \in C(G[An(Z)_G]) = \{G[X], G[W], G[Z]\},$$

niin päädytään edelleen riville kuusi, jonka mukaan

$$P_{w,x}(z) = P(z|w, x).$$

Tulon viimeinen termi $P_{w,x,z}(y)$ toteuttaa rivin neljä ehdon, sillä

$$C(G[\mathbf{V} \setminus \{W, X, Z\}]) = \{G[Y]\}.$$

Graafi $G[Y]$ ei ole graafin G maksimaalinen C-komponentti, mutta solmu Y kuuluu erääseen graafin G maksimaalisista C-komponenteista, sillä $\{Y\} \subset \{X, Y\} = \mathbf{S}'$. Solmujoukolle $\mathbf{S}'$ pätee

$$G[\mathbf{S}'] \in C(G) = \{G[\{X, Y\}], G[W], G[Z]\}.$$

On siis laskettava kausaalivaikutus $P_x(y)$ jakaumasta $P(X|w)P(Y|X, w, z)$ graafissa 10(c). On syytä huomata, että tämä kausaalivaikutus ei ole sama kuin alkuperäinen kausaalivaikutus $P_x(y)$, sillä graafin G muuttujien yhteisjakauma $P(\mathbf{V})$ ei ole sama kuin tämän rekursiotason aligraafin muuttujien yhteisjakauma $P(X|w)P(Y|X, w, z)$.

Seuraavaksi päädytään jälleen riville kaksi, ja koska solmulla Y ei ole havaittuja esivanhempia graafissa 10(c), niin saadaan

$$P_x(y) = \sum_x P(x|w)P(y|x, w, z).$$

Yhdistämällä edellisten kohtien tulokset, ja järjestelemällä termejä, saadaan lopulta lauseke alkuperäiselle kausaalivaikutukselle:

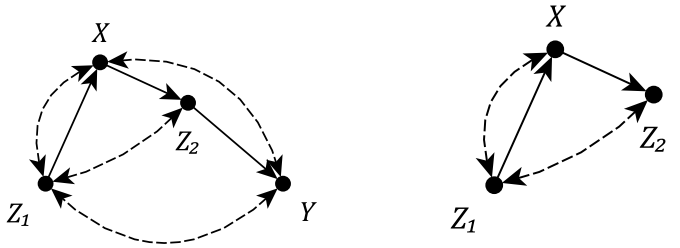
$$P_x(y) = \sum_{w,z} P(z|w, x)P(w) \sum_x P(y|w, x, z)P(x|w).$$

Luvussa 2.3 osoitettiin, että kausaalivaikutus $P_x(y)$ ei ole identifioituva kuvan 5 graafissa G. Todetaan tulos myös algoritmin 1 avulla. Koska $\mathbf{x} \neq \emptyset, \mathbf{V} = An(Y)_G$ ja $\mathbf{W} = \emptyset$, niin kolme ensimmäistä riviä sivuutetaan. Koska

$$C(G[\mathbf{V} \setminus \mathbf{X}]) = C(G[Y]) = \{G[Y]\},$$

niin myös rivi neljä sivuutetaan. Edelleen $C(G) = \{G\}$, joten algoritmi etenee riville viisi ja aiheuttaa **HYLKÄÄ**-komennon. Graafi G sisältää siis pensasaidan kausaalivaikutukselle $P_x(y)$, joka muodostuu C-metsistä G ja $G[Y]$.

Tarkastellaan vielä algoritmin 1 toimintaa edellistä monimutkaisemmassa tilanteessa, jossa pyritään määrittämään kausaalivaikutus, joka ei ole identifioituva. Olkoot graafi $F = \langle \mathbf{V}, \mathbf{E} \rangle$ kuten kuvassa 11(a) ja kiinnostuksen kohteena kausaalivaikutus $P_x(y)$, joka pyritään identifioimaan yhteisjakaumasta $P(X, Y, Z_1, Z_2)$. Asetetaan graafin F topologiseksi järjestykseksi $Z_1 < X < Z_2 < Y$.



(a) Graafi F (b) Aligraafi $An(Z_2)_F$

Kuva 11: Graafi F ja sen aligraafi $F[An(Z_2)_F]$

Aluksi päädytään riville kolme, sillä

$$\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{F_{\overline{\mathbf{X}}}} = (\{X, Y, Z_1, Z_2\} \setminus \{X\}) \setminus \{X, Z_2, Y\} = \{Z_1\} \neq \emptyset.$$

Alkuperäiseen interventioon lisätään muuttuja Z_1, joten määritettävä kausaalivaikutus on nyt $P_{z_1,x}(y)$. Seuraavaksi edetään riville neljä, sillä

$$C(F[\mathbf{V} \setminus \{Z_1, X\}]) = \{F[Z_2], F[Y]\}.$$

Koska $\mathbf{v}\setminus(\{y\}\cup\{z_1, x\}) = \{z_2\}$, niin alkuperäisen vaikutuksen määrittämiseksi on nyt määritettävä kaksi uutta kausaalivaikutusta seuraavassa lausekkeessa:

$$\sum_{z_2} P_{z_1,x,y}(z_2) P_{z_1,x,z_2}(y).$$

Tarkastellaan tulon ensimmäistä termiä. Koska $\mathbf{V} \neq An(Z_2)_F$, niin päädytään riville kaksi, jonka mukaan

$$P_{z_1,x,y}(z_2) = P_{z_1,x}(z_2)$$

aligraafissa, joka muodostuu solmun Z_2 esivanhemmista kuvassa 11(b). Kausaalivaikutus $P_{z_1,x}(z_2)$ ei kuitenkaan ole identifioituva, sillä algoritmi etenee seuraavaksi riville viisi, koska

$$C(F[An(Z_2)_F \setminus \{Z_1, X\}]) = \{F[Z_2]\} \text{ ja } C(F[An(Z_2)_F]) = \{F[An(Z_2)_F]\}.$$

Graafi F sisältää siis pensasaidan kausaalivaikutukselle $P_{z_1,x}(z_2)$, joka muodostuu C-metsistä $F[Z_2]$ ja $F[\{Z_1, Z_2, X\}]$. Tämän seurauksena alkuperäinen kausaalivaikutus $P_x(y)$ ei ole identifioituva.

5 Algoritmin toteutus R-kielellä

Algoritmin 1 implementoimiseksi on valittu tilastolliseen laskentaan suunnattu ohjelmointikieli R (R Core Team, 2014). Toteutuksessa hyödynnetään toistuvasti myös tälle tehtyjä paketteja XML (Temple Lang, 2013) ja igraph (Csardi ja Nepusz, 2006). Implementaatio on julkaistu CRAN-sivustolla (*The Comprehensive R Archive Network*) nimellä *causaleffect* ja sen dokumentaatio on liitteessä B.

5.1 Graafitiedostot

Kausaalimallin indusoima graafi G on algoritmin 1 oleellinen parametri. Graafien kuvaamiseksi on kehitetty lukuisia tiedostoformaatteja moniin eri tarkoituksiin, joista osa on hyvinkin yksinkertaisia, eivätkä tee eroa suunnattujen tai suuntaamattomien graafien välille. Osa formaateista taas tarjoaa kausaalimallien yhteydessä tarpeettomia ominaisuuksia tai edellyttää vaikeaselkoista syntaksia, jonka käsittely voi olla aikaavievää.

GraphML on helppokäyttöinen tiedostomuoto graafien esittämiseksi. Sen ominaisuuksiin kuuluvat muun muassa tuki suunnatuille graafeille ja graafiset kuvaukset. GraphML pohjautuu XML-kieleen (*Extensible Markup Language*), mikä tekee tiedostojen käsittelystä yksinkertaista. GraphML-tiedostot voivat esimerkiksi pitää sisällään solmujen nimet, jolloin niitä ei enää tarvitse määrittää uudelleen itse R-ympäristössä. Graafitiedostoja voidaan luoda helposti esimerkiksi yWorksin kehittämällä yEd-ohjelmistolla, joka tukee kaikkia merkittävimpiä käyttöjärjestelmiä ja on vapaasti saatavilla. Tässä toteutuksessa on yEd-ohjelmistolla tuotettuja GraphML-tiedostojen käsittelyä varten kehitetty oma funktio `parse.graphml`. Yleisesti algoritmin 1 toteutusta voidaan kuitenkin soveltaa mihin tahansa GraphML-tiedostoon igraph-paketin avulla, kunhan se vain toteuttaa jonkin jatkossa käsiteltävistä kaksisuuntaisten särmien esitystavoista.

Graafisten parametrien avulla voidaan erottaa kaksi- ja yksisuuntaiset särmät toisistaan. Tätä varten on kiinnitetty kolme tapaa, joilla havaitsemattomia muuttujia vastaavat kaksisuuntaiset särmät voidaan esittää.

(a) Tapa 1 (b) Tapa 2 (c) Tapa 3

Kuva 12: Kaksisuuntaisten särmien merkintätavat

Kuvissa 12(a) ja 12(b) esiintyvät merkintätavat ovat lähes identtiset. Erityisesti merkintätapa 1 on yleinen kirjallisuudessa. Vastaavuutensa vuoksi kummallekin merkintätavalle on annettu nimi *standard*. Kolmas merkintätapa,

joka esiintyy kuvassa 12(c) on edellisistä selvästi poikkeava. On selvää, että tätä merkintää ei voida käyttää sellaisenaan, sillä se aiheuttaa graafiin silmukan, mikä ei ole sallittua. GraphML-formaatti kuitenkin tarjoaa mahdollisuuden asettaa särmille parametreja, mitä hyödyntäen nämä särmät voidaan erottaa muista yksisuuntaisista särmistä. Merkintätapaa 3 käytettäessä asetetaan kaksisuuntaisia särmiä vastaaville yksisuuntaisille särmille parametri `description` ja sen arvoksi kirjain `U` (*unobserved*). Merkintätapa 3 vastaa tässä toteutuksessa käytettyä sisäistä formaattia, joten sen nimeksi on annettu *internal*.

yEd-editorilla tuotettujen GraphML-tiedostojen käsittelyyn käytetään R-pakettia XML, jonka sisältämän funktion `xmlParse` avulla tiedostot luetaan suoraan R-ympäristössä käsiteltäviksi olioiksi. On kuitenkin huomioitava, että XML-paketin muodostamat R-oliot ovat ainoastaan sen sisäisen C-kielen toteutuksen esiintymiä ja poikkeavat siten useimpien R-pakettien luomista olioista. Tämä on otettava huomioon vapauttamalla XML-olioiden varaama muisti, kun olioita ei enää tarvita. Tavallisesti R tekee tämän automaattisesti.

Algoritmi 1 tarvitsee toimiakseen XML-sisällöstä vain pienen osan, ja ylimääräinen sisältö karsitaan pois hakemalla XML-rakenteesta vain tarpeelliset solmut. Oleellista tietoa ovat solmujen nimet, solmujen lukumäärä, särmien lukumäärä ja särmien `description`-parametrien argumentit. Jos annetun GraphML-tiedoston kaksisuuntaisille särmille on käytetty kuvan 12(a) tai 12(b) merkintätapaa, niin ne muunnetaan vastaamaan sisäistä formaattia, kuten kuvassa 12(c). Tietojen haku suoritetaan funktiolla `getNodeSet`. Tämä funktio hyödyntää XPath-kieltä, joka on erityinen XML-tiedonhakuun kehitetty prosessointikieli (Simpson, 2002).

Kun tarpeeton informaatio on saatu karsittua, muodostetaan jäljelle jääneestä XML-sisällöstä igraph-graafi. igraph on graafien visualisointiin ja prosessointiin erikoistunut R-paketti, joka kykenee käsittelemään jopa miljoonia solmuja sisältäviä graafeja, mikä on sen C-kielisen toteutuksen ansiota. Paketti tarjoaa myös algoritmiin 1 liittyviä hyödyllisiä ominaisuuksia, kuten solmun esivanhempien määrittämisen, topologisen järjestyksen etsimisen ja aligraafien muodostamisen solmu- tai särmäjoukon perusteella. Yksi igraph-paketin päätarkoituksista onkin juuri graafialgoritmien vaivaton implementointi.

5.2 Jakaumaoliot

Tärkeä kysymys edellisen luvun 4.2 algoritmin toteutuksen kannalta on, kuinka suorituksen edetessä muuttuva todennäköisyysjakauma tulisi määritellä. Luonnollinen ratkaisu tähän on toteuttaa jakaumaolio, joka ylläpitää jakaumassa kullakin hetkellä esiintyviä termejä. Jakaumaolio on rakenteeltaan rekursiivinen kuten itse algoritmikin. Tämä tarkoittaa käytännössä sitä, että suorituksen edetessä riville neljä, viisi tai seitsemän, joilla todennäköisyysjakauma faktoroituu useammaksi termiksi, muodostetaan jakauma-aliolioita, joista jokainen vastaa yhtä tulon tekijöistä. Nämä alioliot voivat edelleen tar-

vittaessa haarautua aliolioiden aliolioiksi ja niin edelleen.

Jakaumaoliot pitävät sisällään lukuisia attribuutteja, joita tarvitaan vastaavan todennäköisyysjakauman täsmälliseen esittämiseen. Merkkijonovektorit `var` ja `cond` ovat attribuuteista tyypillisimpiä, sillä niiden avulla voidaan määrittää yksinkertainen ehdollinen jakauma. Jakauma muodostuu vektorin `var` muuttujista ehdolla vektorin `cond` muuttujat. Olkoon esimerkiksi jakaumaolio `p` attribuutteinaan `var = "Y"` ja `cond = "X"`. Kyseinen olio `p` esittää siten ehdollista jakaumaa $P(Y|X)$.

Jos jakauma muodostuu useamman termin tulosta, niin tulon tekijät ilmoitetaan listana `children`, ja loogiselle muuttujalle `recursive` asetetaan totuusarvo `TRUE`. Esimerkiksi jakaumaa $P^* = P(Z|X)P(X|Y)P(Y)$ esittävälle oliolle pätee `children = list(a,b,c)`, missä jakaumaoliot `a,b` ja `c` kuvaavat jakaumia $P(Z|X), P(X|Y)$ ja $P(Y)$ vastaavasti.

Reunajakaumien esittämiseksi on määritetty merkkijonovektori `sumset`, jonka sisältämän muuttujajoukon yli jakaumaa integroidaan jatkuvassa tapauksessa tai summataan diskreetissä tapauksessa. Yksinkertaisessa tilanteessa tätä attribuuttia ei tarvita, mutta astetta monimutkaisempien graafien yhteydessä tulee vastaan tilanteita, joissa reunajakauman lausekkeen sieventäminen ei ole aivan yksinkertaista. Jos olisi esimerkiksi laskettava satunnaismuuttujan X reunajakauma $P^*(X)$ edellisen esimerkin yhteisjakaumasta $P^*(X,Y,Z)$, niin tätä esittävälle jakaumaoliolle tulisi asettaa `sumset = c("Y","Z")`, sillä $P^*(X) = \sum_{Y,Z} P(Z|X)P(X|Y)P(Y)$.

Jakaumaoliot monimutkaistuvat entisestään, kun halutaan määrittää ehdollisia jakaumia tulomuotoisista yhteisjakaumista, jolloin tähän asti esitetyt attribuutit eivät välttämättä riitä sopivan jakaumaolion muodostamiseksi. Tarkastellaan jälleen jakaumaa P^*. Jos nyt reunajakauman $P^*(X)$ sijaan haluttaisiin määrittää ehdollinen jakauma $P^*(X|Y)$, niin saadaan

$$P^*(X|Y) = \frac{P^*(X,Y)}{P^*(Y)} = \frac{\sum_Z P(Z|X)P(X|Y)P(Y)}{\sum_{X,Z} P(Z|X)P(X|Y)P(Y)} = \frac{P(X|Y)\sum_Z P(Z|X)}{\sum_X [P(X|Y)\sum_Z P(Z|X)]} = P(X|Y).$$

Vastaavat yksinkertaiset tilanteet on otettu toteutuksessa huomioon. Summalausekkeiden sisällä olevat termit, jotka eivät sisällä yhtäkään niistä muuttujista, joiden yli summaa lasketaan, siirretään summalausekkeen ulkopuolelle. Tämän jälkeen tarkistetaan, voidaanko lausekkeita sieventää edelleen summausjärjestystä vaihtamalla. Lopuksi osoittajan ja nimittäjän yhteiset termit supistetaan pois.

Edellä esitetyt sievennyssäännöt eivät riitä läheskään kaikkien tilanteiden ratkaisemiseen. Esimerkiksi lauseke $\sum_X P(Y|X)P(X)$ ei sievene tällä menettelyllä, sillä summalausekkeen sisältä ei voida tuoda termejä sen ulkopuolelle, eikä summausjärjestystä voida vaihtaa. Tilanteita, joissa nimittäjä ei supistu kokonaan lausekkeesta, kuvataan jakaumaolioiden attribuuteilla `fraction` ja

`divisor`. Kuten attribuutti `recursive`, on myös attribuutti `fraction` looginen muuttuja, jolle asetetaan totuusarvo `TRUE`, kun jakauman lausekkeelle on tarpeellista määritellä nimittäjä. Listan `divisor` alkiot ovat nimittäjän lauseketta kuvaavia jakaumaolioita.

5.3 Maksimaaliset C-komponentit

Luvussa 4.1 osoitettiin, että jokaista suunnattua silmukatonta graafia G kohden on aina olemassa yksikäsitteinen joukko $C(G)$, joka sisältää graafin G maksimaaliset C-komponentit. Tämän joukon määrittämiseksi haetaan graafista ensin kaikki kaksisuuntaiset särmät ja muodostetaan aligraafi, josta kaikki muut särmät on poistettu. Jäljelle jääneeseen graafiin syntyy yksi tai useampi *komponentti* eli aligraafi, jonka jokaista solmuparia yhdistää polku. Koska komponentit ovat erillisiä, ja jokaiseen komponenttiin kuuluvat solmut ovat yhteydessä toisiinsa vain kaksisuuntaisista särmistä koostuvien polkujen kautta, on näiden solmujoukkojen oltava graafin G maksimaalisia C-komponentteja. Graafin $G = \langle \mathbf{V}, \mathbf{E} \rangle$ kaksisuuntaiset särmät etsitään käyttäen sen vieruspistematriisia (*adjacency matrix*).

Määritelmä (vieruspistematriisi)**.** Suunnatun graafin G *vieruspistematriisi* on $n \times n$ -matriisi $A = [a_{ij}]$, missä n on graafin G solmujen lukumäärä, $\mathbf{V} = \{V_1, V_2, \ldots, V_n\}$ ja a_{ij} on särmien lukumäärä solmusta V_i solmuun V_j.

Koska graafi G on suunnattu, ei sen vieruspistematriisi ole välttämättä symmetrinen. On helppo todeta, että kahden solmun V_i ja V_j välillä on ainakin yksi kaksisuuntainen särmä jos ja vain jos $a_{ij} \geq 1$ ja $a_{ji} \geq 1$. Kaikki kaksisuuntaiset särmät voidaan siten määrittää yksinkertaisesti vertaamalla matriisia A sen transpoosiin A^T, ja valitsemalla vain niitä indeksejä vastaavat särmät, joille sekä $a_{ij} \geq 1$ että $a_{ij}^\mathsf{T} \geq 1$.

Vain kaksisuuntaisia särmiä sisältävä aligraafi muodostetaan igraph-paketin funktiolla `subgraph.edges`. Funktio säilyttää kaikki alkuperäisen graafin solmut, mutta poistaa kaikki ne särmät, joita ei ole annettu funktiolle argumentteina. Funktion palauttama aligraafi jaetaan komponentteihin käyttäen igraph-paketin funktiota `decompose.graph`.

5.4 Implementaatio

Edellä on tehty kaikki tarvittavat valmistelut algoritmin 1 implementoimiseksi. Todennäköisyysjakaumia voidaan käsitellä jakaumaolioiden avulla ja vieruspistematriisi antaa keinon määrittää graafin G maksimaaliset C-komponentit. Muut graafiteoreettiset operaatiot, kuten aligraafien muodostaminen ja esivanhempien etsiminen, on puolestaan toteutettu igraph-paketissa. Tässä implementaatiossa algoritmi 1 on ymmärretty siten, että se ottaa syötteenään joukot $\mathbf{x}$ ja $\mathbf{y}$ sekä graafin G ja palauttaa merkkijonon, joka on kausaalijakauman $P_\mathbf{x}(\mathbf{y})$ lauseke graafin G muuttujien yhteisjakauman $P(\mathbf{V})$ avulla ilmaistuna. Merkkijonon esittämiseksi on valittu merkintäkieli LaTeX.

Algoritmia 1 vastaava R-funktio on nimeltään `id`. Funktiolla on viisi parametria: merkkitietovektori `y`, merkkitietovektori `x`, jakaumaolio `P`, igraph-graafi `G` ja merkkitietovektori `to`. Neljä ensimmäistä parametria vastaavat algoritmin 1 matemaattisia vektoreita $\mathbf{x}$ ja $\mathbf{y}$, jakaumaa P ja graafia G. Viimeinen parametri `to` on graafin `G` solmujen topologinen järjestys. Tarvittavat joukko-operaatiot on toteutettu R-kielessä funktioina `intersect`, `setdiff` ja `union`, jotka vastaavat joukkojen leikkausta, erotusta ja yhdistettä. Funktion `id` ja sen käyttämien apufunktioiden R-koodit ovat liitteessä A.

Jokaisella rekursiotasolla tallennetaan havaittu osa graafista `G` graafiksi `G.obs`. Kyseinen graafi sisältää siis kaikki alkuperäisen graafin havaitut solmut ja niiden väliset särmät. Lisäksi graafin `G` havaitut solmut ja esivanhemmat talletetaan merkkitietovektoreiksi `v` ja `anc`. Käsitellään seuraavaksi algoritmin 1 toteutusta rivi kerrallaan.

1: **jos** $\mathbf{x} = \emptyset$, **niin**
   palauta $\sum_{v \in \mathbf{v} \setminus \mathbf{y}} P(\mathbf{v})$.

Rivillä 1 määritetään ehdon $\mathbf{x} = \emptyset$ totuusarvo. Tämä tarkastetaan laskemalla vektorin `x` pituus. Jos pituus on nolla, niin funktio `id` palauttaa jakaumaolion `P`, jonka attribuutin `sumset` arvoihin lisätään vektoreiden `v` ja `y` erotus.

2: **jos** $\mathbf{V} \neq An(\mathbf{Y})_G$, **niin**
   palauta ID$(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, P(An(\mathbf{Y})_G), G[An(\mathbf{Y})_G])$.

Rivin 2 ehdon totuusarvo määritetään laskemalla vektoreiden `v` ja `anc` erotuksen pituus. Jos pituus ei ole nolla, niin funktiota id kutsutaan seuraavilla määritteillä: vektorin `y` alkiot pysyvät muuttumattomina, ja vektorin `x` alkioista valitaan vain vektorin `anc` alkiot. Lisäksi on määritettävä reunajakauma $P(An(\mathbf{Y})_G)$, joten jakaumaolion `P` attribuutin `sumset` arvoksi asetetaan vektoreiden `v` ja `anc` erotus. Lopuksi lasketaan indusoitua aligraafia $G[An(\mathbf{Y})_G]$ vastaava igraph-aligraafi solmuista `anc` ja niitä yhdistävistä särmistä igraph-paketin funktiolla `induced.subgraph`. Funktio muodostaa aligraafin syötteenä annetusta graafista sekä solmujoukosta säilyttäen kaikki solmujoukon keskinäiset särmät alkuperäisessä graafissa.

3: **olkoon** $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
   jos $\mathbf{W} \neq \emptyset$, **niin**
   palauta ID$(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.

Jotta solmujoukkoa $\mathbf{W}$ vastaava vektori `w` voidaan määrittää, on aluksi muodostettava aligraafi $G_{\overline{\mathbf{X}}}$ ja tätä varten etsittävä solmujoukkoon $\mathbf{X}$ saapuvat särmät. igraph-paketti tarjoaa tähän tarkoitukseen soveltuvan operaattorin `%->%`, jonka avulla voidaan etsiä tietystä solmusta tai tiettyyn solmuun suuntautuvia särmiä. Tarvittavat särmät saadaan tässä tapauksessa komennolla `E(G) [1:length(E(G)) %->% x]`, missä funktio `E` palauttaa argumenttina annetun graafin kaikki särmät.

Kun aligraafi on muodostettu, voidaan vektori `w` muodostaa tavallisilla joukko-operaatioilla. Jos vektorin `w` pituus ei ole nolla, niin funktiota `id` kut-

sutaan seuraavilla määritteillä: vektorin `y` alkiot pysyvät muuttumattomina, ja vektorin `x` alkioihin yhdistetään vektorin `w` alkiot. Jakaumaolio `P` ja graafi `G` pysyvät muuttumattomina.

4: **jos** $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}_1], \ldots, G[\mathbf{S}_k]\}$, **niin**
palauta $\sum_{v \in \mathbf{v} \setminus (\mathbf{y} \cup \mathbf{x})} \prod_{i=1}^{k} \mathbf{ID}(\mathbf{s}_i, \mathbf{v} \setminus \mathbf{s}_i, P, G)$.

Joukko $C(G[\mathbf{V} \setminus \mathbf{X}])$ määritetään apufunktiolla `c.components`. Funktio määrittää jokaisen syötteenä annetun graafin maksimaalisen C-komponentin solmujoukon ja palauttaa nämä listana, joka tallennetaan muuttujaan `s`. Jos palautetun listan pituus on suurempi kuin yksi, niin funktio `id` palauttaa uuden jakaumaolion, jonka attribuutin `sumset` arvoksi asetetaan vektorin `v` ja vektoreiden `y` ja `x` yhdisteen erotus. Lisäksi olion attribuutti `recursive` saa totuusarvon `TRUE`, eli kyseessä on tulomuotoa oleva jakaumaolio. Tämän jakaumaolion listassa `children` olevat alioliot määräytyvät uusista rekursiivisista funktiokutsuista. Funktiota `id` kutsutaan jokaista C-komponenttia $G[\mathbf{S}_i]$, $i = 1, \ldots, k$ kohden uudelleen seuraavilla määritteillä: vektoriksi `y` asetetaan kyseisen C-komponentin solmujoukkoa vastaava listan `s` alkio `s[[i]]`, kun taas vektoriksi `x` asetetaan vektorin `v` ja solmujoukon `s[[i]]` erotus. Näiden funktiokutsujen jakaumaolio `P` ja graafi `G` pysyvät jälleen muuttumattomina.

Jos algoritmi ei ole tähän mennessä edennyt yhdellekään edellisistä riveistä, on ehdon $C(G[\mathbf{V} \setminus \mathbf{X}]) = \{G[\mathbf{S}]\}$ täytynyt toteutua. Ainoan löytyneen C-komponentin $G[\mathbf{S}]$ solmujoukkoa $\mathbf{S}$ vastaa nyt merkkitietovektori `s`. Toisin sanoen edellisessä kohdassa määritetty C-komponenttien lista `s` korvataan sen ensimmäisellä alkiolla `s[[1]]`.

5: **jos** $C(G) = \{G\}$, **niin**
aiheuta HYLKÄÄ$(G, G[\mathbf{S}])$.

Jos apufunktion `c.components` palauttaman listan pituus olikin yksi, niin edetään viimeisille riveille. Jos lisäksi tämän listan ainoan C-komponentin $G[\mathbf{S}]$ solmujoukko on sama kuin graafin G solmujoukko, niin suoritus keskeytetään R-ohjelmiston `stop` funktiolla. Virheilmoituksena tulostetaan merkkitietovektorit `v` ja `s`, joita vastaavat solmujoukot $\mathbf{V}$ ja $\mathbf{S}$. Näitä solmujoukkoja vastaavat indusoidut aligraafit muodostavat edelleen pensasaidan kyseisen rekursiotason kausaalivaikutukselle.

6: **jos** $G[\mathbf{S}] \in C(G)$, **niin**
palauta $\sum_{v \in \mathbf{s} \setminus \mathbf{y}} \prod_{V_i \in \mathbf{S}} P(v_i | v_\pi^{(i-1)})$.

On hyödynnettävä jälleen apufunktiota `c.components` graafin G maksimaalisten C-komponenttien määräämiseksi. Jos rivillä neljä löydetty C-komponentti $G[\mathbf{S}]$ on yksi graafin G maksimaalisista C-komponenteista, niin funktio `id` palauttaa uuden jakaumaolion, jonka attribuutin `sumset` arvoksi asetetaan C-komponentin $G[\mathbf{S}]$ solmujen `s` ja graafin G solmujen `v` erotus. Jakauma on jälleen tulomuotoa, eli attribuutti `recursive` saa tässäkin tapauksessa totuusarvon `TRUE`. Uuden jakaumaolion alioliot listassa `children` määräytyvät

ehdollisia jakaumia esittävistä jakaumaolioista, jotka on laskettava rekursiokutsun jakaumaoliosta `P` jokaista joukon $\mathbf{S}$ solmua V_i kohden. Ehdollistavat solmut määräytyvät solmua V_i edeltävistä solmuista topologisessa järjestyksessä `to`.

7: **jos** $(\exists \mathbf{S}')\mathbf{S} \subset \mathbf{S}'$ siten että $G[\mathbf{S}'] \in C(G)$, **niin**
palauta ID$(\mathbf{y}, \mathbf{x} \cap \mathbf{s}', \prod_{V_i \in \mathbf{S}'} P(V_i | V_\pi^{(i-1)} \cap \mathbf{S}', v_\pi^{(i-1)} \setminus \mathbf{s}'), G[\mathbf{S}'])$.

Jos C-komponentti $G[\mathbf{S}]$ ei ole yksi graafin G maksimaalisista C-komponenteista, on sen oltava jonkin graafin G maksimaalisen C-komponentin $G[\mathbf{S}']$ osajoukko. Selkeyden vuoksi merkkitietovektori `s` korvataan solmujoukon $\mathbf{S}$ muuttujien nimillä. Funktiota `id` kutsutaan seuraavilla määritteillä: vektori `y` pysyy muuttumattomana, ja vektorin `x` arvoksi asetetaan C-komponentin $G[\mathbf{S}']$ solmujen `s` ja rekursiokutsun vektorin `x` leikkaus. Funktiokutsun jakaumaolio on jälleen tulomuotoa, joten attribuutti `recursive` saa totuusarvon `TRUE`. Jakaumaolion alioliot listassa `children` määräytyvät ehdollisia jakaumia esittävistä jakaumaolioista, jotka on laskettava nykyisen rekursiokutsun jakaumaoliosta `P` jokaista C-komponentin $G[\mathbf{S}']$ solmua eli vektorin `s` alkiota kohden.

Tämän toteutuksen kannalta ei erottelulla muuttujiin ja muuttujien arvoihin ole tässä yhteydessä merkitystä, joten ehdollistavina muuttujina ovat yksinkertaisesti solmua V_i edeltävät muuttujat järjestyksessä `to`, sillä

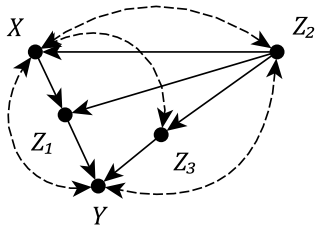
$$(V_\pi^{(i-1)} \cap \mathbf{S}') \cup (V_\pi^{(i-1)} \setminus \mathbf{S}') = V_\pi^{(i-1)}.$$

Lisäksi on muodostettava aligraafi C-komponentin $G[\mathbf{S}']$ solmuista $\mathbf{S}'$ ja niiden välisistä särmistä, jolloin hyödynnetään jälleen funktiota `induced.subgraph`. Tällä kertaa argumenttina annetaan vektori `s`.

6 Esimerkkejä

6.1 Monimutkainen kausaalivaikutus

Algoritmin 1 rivillä 6 laskettavat ehdolliset jakaumat $P(v_i|v_\pi^{(i-1)})$ voivat tuottaa hankalia lausekkeita määritettäessä kausaalivaikutuksia monimutkaisista graafeista, koska implementaation sievennyssäännöillä ei pystytä käsittelemään kaikkia mahdollisia tapauksia. Tällaista erikoistilannetta voidaan havainnollistaa esimerkiksi kuvan 13 graafin G avulla, josta pyritään määrittämään kausaalivaikutus $P_x(z_1, z_2, z_3, y)$.



Kuva 13: Esimerkki graafista, josta identifioitava kausaalivaikutus tuottaa monimutkaisen lausekkeen

Tian (2002) osoitti väitöskirjassaan tämän vaikutuksen olevan identifioituva ja määritti sen lausekkeen, joka on

$$P_x(z_1, z_2, z_3, y) = P(z_1|x, z_2) \sum_x \Big[P(y, z_3|x, z_1, z_2)P(x, z_2)\Big].$$

Sovellettaessa algoritmia 1 tähän kausaalivaikutukseen, on eräänä välivaiheena määrittää ehdollinen jakauma $P^*(Y|Z_2, Z_3)$, missä

$$P^*(Y, Z_2, Z_3) = \sum_X P(Y|Z_2, X, Z_3, Z_1)P(Z_3|Z_2, X)P(X|Z_2)P(Z_2)$$

ja P on graafin G havaittujen muuttujien yhteisjakauma. Sovelletaan nyt algoritmin 1 implementaatiota kyseiseen graafiin kausaalivaikutuksen määrittämiseksi, jolloin saadaan seuraava lauseke

$$P_x(z_1, z_2, z_3, y) = \\ P(z_1|z_2, x)P(z_3|z_2)P(z_2)\frac{\sum_x \Big[P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)\Big]}{\sum_{x,y} \Big[P(y|z_2, x, z_3, z_1)P(z_3|z_2, x)P(x|z_2)\Big]}.$$

Tulos on huomattavasti vaikeaselkoisempi kuin Tianin johtama lauseke. Osoitetaan seuraavaksi kausaalilaskentaa hyödyntäen, että lausekkeet yhtenevät.

Koska joukko $\{Z_2, X\}$ d-separoi kaikki polut solmusta Z_1 solmuun Z_3, niin $(Z_3 \perp\!\!\!\perp Z_1 | Z_2, X)_{G_{\overline{X},\overline{Z}_2}}$, jolloin

$$\begin{aligned}
&P(z_1|z_2,x)P(z_2)\sum_x\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x|z_2)\Big] \\
&= P(z_1|z_2,x)\sum_x\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x,z_2)\Big] \\
&= P(z_1|z_2,x)\sum_x\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x,z_1)P(x,z_2)\Big] \\
&= P(z_1|z_2,x)\sum_x\Big[P(y,z_3|z_2,x,z_1)P(x,z_2)\Big],
\end{aligned}$$

missä toinen yhtäsuuruus seuraa kausaalilaskennan säännöstä 1. Viimeinen rivi vastaa Tianin johtamaa lauseketta termien järjestystä lukuun ottamatta. Osoitetaan vielä, että loput termit supistuvat lausekkeesta.

$$\begin{aligned}
&\frac{P(z_3|z_2)}{\sum_{x,y}\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x|z_2)\Big]} \\
&= \frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x,z_2)\Big]}.
\end{aligned}$$

Soveltamalla nimittäjään edellä tehtyä päättelyä saadaan

$$\frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}\Big[P(y|z_2,x,z_3,z_1)P(z_3|z_2,x)P(x,z_2)\Big]} = \frac{P(z_3|z_2)P(z_2)}{\sum_{x,y}\Big[P(y,z_3|z_2,x,z_1)P(x,z_2)\Big]}.$$

Käyttämällä kausaalilaskennan sääntöä 1 kuten edellä, mutta käänteiseen suuntaan, saadaan edelleen

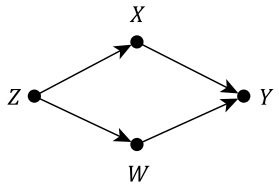
$$\begin{aligned}
&\frac{P(z_3,z_2)}{\sum_x\Big[P(z_3|z_2,x,z_1)P(x,z_2)\Big]} = \frac{P(z_3,z_2)}{\sum_x\Big[P(z_3|z_2,x)P(x,z_2)\Big]} \\
&= \frac{P(z_3,z_2)}{\sum_x\Big[P(z_3|z_2,x)P(x,z_2)\Big]} = \frac{P(z_3,z_2)}{\sum_x P(z_3,z_2,x)} = \frac{P(z_3,z_2)}{P(z_3,z_2)} = 1.
\end{aligned}$$

Implementaatio tuottaa siis oikean tuloksen lausekkeen monimutkaisuudesta huolimatta.

6.2 d-separoituneisuus

Algoritmi 1 ei hyödynnä kaikkia mahdollisia graafien riippumattomuusominaisuuksia. Esimerkiksi rivillä kuusi muodostettavaa jakaumaa ehdollistetaan kaikilla solmua V_i edeltävillä solmuilla topologisessa järjestyksessä π, vaikka usein ainakin jotkin polut solmun V_i ja tätä edeltävien solmujen välillä ovat d-separoituneita. Tällöin solmun V_i kanssa d-separoituneet muuttujat voitaisiin

jättää pois ehtolausekkeesta, sillä ne ovat ehdollisesti riippumattomia muuttujasta V_i kyseisessä graafissa. Tilannetta havainnollistaa seuraava esimerkki, jossa määritetään kausaalivaikutus $P_{x,w}(y)$ alla olevassa kuvan 14 graafissa G.



Kuva 14: Esimerkki graafista, joka synnyttää kausaalisen Markov-ehdon ulkopuolisia riippumattomuuksia

Algoritmi 1 antaa kausaalivaikutuksen lausekkeeksi

$$P(y|z, x, w),$$

vaikka $(Y \perp\!\!\!\perp Z|X, W)_G$. Kausaalivaikutuksen lauseke voitaisiin siis edelleen sieventää muotoon $P(y|x, w)$.

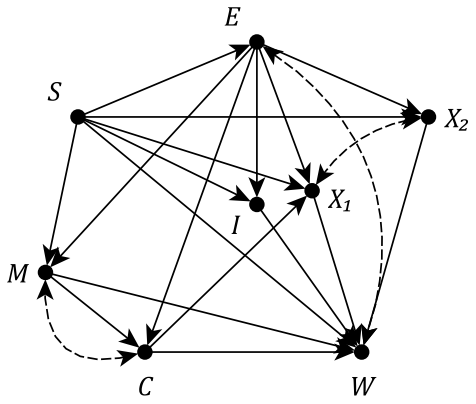
6.3 Palkkaerot

Käytännön esimerkkinä kausaalilaskennan sovelluksesta käsitellään kysymystä siitä, miten eri tekijät vaikuttavat palkan määräytymiseen. Tällaisista tekijöistä kenties mielenkiintoisin on sukupuoli. Sukupuolen kausaalivaikutus palkkaan on helppo mieltää kysymyksenä siitä, minkälaisia eroja henkilön palkassa havaittaisiin jos hän olisikin vastakkaista sukupuolta. Sukupuoli ei luonnollisesti ole ainoa tekijä palkkatason määräytymisessä, vaan siihen liittyy lukuisia työntekijää, työnantajaa ja työtehtäviä koskevia muuttujia. On selvää, että kaikkia mahdollisia tekijöitä ei voida ottaa huomioon, eivätkä kaikki kausaaliset suhteet ole itsestäänselvyyksiä. Empiiristen tutkimusten valossa voidaan kuitenkin valita malliin keskeisimpiä muuttujia.

Yksilötasolla korkeasti koulutetut ja runsaasti työkokemusta hankkineet henkilöt saavat yleensä korkeampaa palkkaa (Becker, 1962 ja Mincer, 1974). Asplund (2000) kuitenkin toteaa, että Suomessa nämä tekijät vaikuttavat molempien sukupuolten palkkoihin lähes samalla tavalla. Sen sijaan potentiaalisesta työkokemuksesta eli ajasta, joka on kulunut henkilön valmistumisesta, näyttäisi olevan enemmän hyötyä miehille kuin naisille (Asplund, 2000 ja Napari, 2008). Myös henkilön siviilisääty ja perheen koko vaikuttavat oleellisesti palkan määräytymiseen (Pollman-Schult, 2011 sekä Davies ja Pierre, 2005).

Palkkaerojen analyyseissä hyödynnetään usein myös työnantajien toimialoihin liittyviä piirteitä, jotka usein selittävät havaittua eroa kattavasti (Fields ja Wolff, 1995). Napari (2008) kuitenkin havaitsi, että lukuisista tekijöistä ainoastaan yrityksen koolla on vaikutusta Suomessa.

Seuraavaksi tarkastellaan kahta potentiaalista kausaalimallia palkkaeroille, jotka perustuvat esiteltyihin empiirisiin tuloksiin. Ensimmäistä mallia vastaava graafi on kuvassa 15, jossa esiintyvät muuttujat määritellään seuraavasti:



Kuva 15: Ensimmäinen esimerkki palkkaeroihin liittyvästä kausaalirakenteesta

$$
\begin{aligned}
S &= \text{työntekijän sukupuoli} \\
E &= \text{työntekijän koulutus} \\
X_1 &= \text{työntekijän potentiaalinen työkokemus} \\
X_2 &= \text{työntekijän realisoitunut työkokemus} \\
I &= \text{yrityksen toimiala} \\
M &= \text{työntekijän siviilisääty} \\
C &= \text{työntekijän lasten lukumäärä} \\
W &= \text{työntekijän palkka}
\end{aligned}
$$

Sukupuolella on suora vaikutus useimpiin mallissa esiintyviin muuttujiin. Vaikutusta siviilisäätyyn voidaan perustella esimerkiksi sillä, että Suomessa naiset avioituvat keskimäärin nuoremmassa iässä kuin miehet. Miehet ja naiset myös hakeutuvat koulutuksessaan tyypillisesti eri aloille, ja hankittu koulutus edistää edelleen koulutuksen mukaisille toimialoille työllistymistä. Vaikutus potentiaaliseen työkokemukseen voidaan perustella naisten keskimääräisesti nopeammalla valmistumisella. Korkea koulutus rajoittaa henkilön potentiaalista työkokemusta, mutta voi edistää konkreettisen työkokemuksen hankkimista. Keskimääräinen avioitumisikä on myös korkeampi korkeasti koulutetuilla.

Malliin liittyviä havaitsemattomia muuttujia on myös syytä perustella. Potentiaalisen ja realisoituneen työkokemuksen yhteyttä voidaan selittää esimerkiksi henkilön aktiivisuudella työmarkkinoilla. Koulutuksen ja palkan yh-

teyteen liittyy esimerkiksi henkilön motivaatio, joka puolestaan voi edelleen vaikuttaa tehdyn työn laatuun.

Soveltamalla algoritmin 1 implementaatiota saadaan sukupuolen kausaalivaikutukselle palkkaan seuraava lauseke:

$$P_s(w) = \sum_{e,m,x_2,i,c,x_1} \Big[P(i|s,e)P(c|s,e,m)P(m|s,e)P(w|s,e,m,x_2,i,c,x_1) \\ P(x_1|s,e,m,x_2,i,c)P(x_2|s,e,m)P(e|s)\Big].$$

On kuitenkin syytä huomata, että kausaalivaikutus voidaan tässä tapauksessa määrittää yksinkertaisesti kausaalilaskennan säännön 2 avulla. Koska $(S \perp\!\!\!\perp W)_{G_{\underline{S}}}$, niin $P_s(w) = P(w|s)$ eli sukupuolen kausaalivaikutus palkkaan on sama kuin palkan ehdollinen jakauma ehdolla sukupuoli. Kuten luvun 6.1 esimerkin tapauksessa, tuottaa algoritmi 1 jälleen tarpeettoman monimutkaisen lausekkeen kausaalivaikutukselle. Usein onkin syytä tarkastaa, onko kiinnostava kausaalivaikutus määritettävissä helposti ennen algoritmin 1 soveltamista.

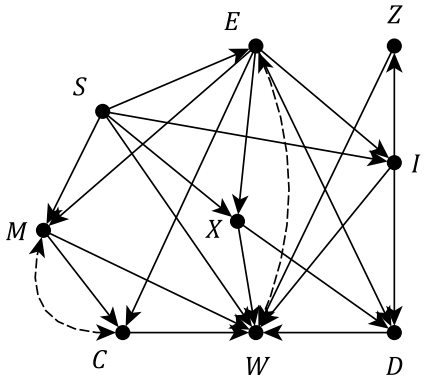
Mielenkiintoista voisi olla myös selvittää, mikä on sukupuolen ja koulutuksen kausaalivaikutus palkkaan. Vaikutuksen määrittäminen suoraan ei nyt ole yhtä suoraviivaista kuin pelkän sukupuolen tapauksessa. Soveltamalla algoritmin 1 implementaatiota saadaan sukupuolen ja koulutuksen kausaalivaikutukselle palkkaan seuraava lauseke:

$$P_{s,e}(w) = \sum_{m,x_2,i,c,x_1} \Big[P(i|s,e)P(c|s,e,m)P(m|s,e)P(x_1|s,e,m,x_2,c) \\ P(x_2|s,e,m)\sum_e \Big[P(w|s,e,m,x_2,i,c,x_1)P(e|s)\Big]\Big]$$

Kausaalivaikutus on siis identifioituva ja vaikutuksen analysoimiseksi olisi esimerkin tilanteessa mallinnettava lukuisia ehdollisia jakaumia. Osa termeistä sisältää ainoastaan demografisia tekijöitä, kuten eri sukupuolten koulutusvuosien jakauma $P(e|s)$, lasten lukumäärän jakauma $P(c|s,e,m)$ ehdolla sukupuoli, koulutus ja siviilisääty sekä siviilisäädyn jakauma $P(m|s,e)$ ehdolla sukupuoli ja koulutus. Näitä termejä mallintamalla olisi myös mahdollista kuvailla töissä käyvää osuutta väestöstä.

Yritykseen ja työkokemukseen liittyviä termejä ovat toimialan jakauma $P(i|s,e)$ ehdolla sukupuoli ja koulutus, realisoituneen työkokemuksen jakauma $P(x_2|s,e,m)$ ehdolla sukupuoli, koulutus ja siviilisääty sekä potentiaalisen työkokemuksen jakauma $P(x_1|s,e,m,x_2,c)$ ehdolla kaikki havaitut muuttujat palkkaa ja toimialaa lukuun ottamatta. Lopuksi tarvitaan vielä palkan ehdollinen jakauma $P(w|s,e,m,x_2,i,c,x_1)$ ehdolla kaikki muut havaitut muuttujat. Näiden termien avulla voidaan tarkastella kausaalivaikutuksen lisäksi esimerkiksi sitä, miten koulutus ja työkokemus vaikuttavat palkan määräytymiseen muiden tekijöiden ohella.

Käsitellään seuraavaksi toista kausaalimallia palkkaeroille. Tätä mallia vastaava graafi on kuvassa 16, jossa esiintyvät muuttujat määritellään seuraavasti:



Kuva 16: Toinen esimerkki palkkaeroihin liittyvästä kausaalirakenteesta

$$
\begin{aligned}
S &= \text{työntekijän sukupuoli} \\
E &= \text{työntekijän koulutus} \\
X &= \text{työntekijän realisoitunut työkokemus} \\
Z &= \text{yrityksen koko} \\
I &= \text{yrityksen toimiala} \\
M &= \text{työntekijän siviilisääty} \\
C &= \text{työntekijän lasten lukumäärä} \\
D &= \text{työtehtävän haastavuus} \\
W &= \text{työntekijän palkka}
\end{aligned}
$$

Malli vastaa ensimmäistä mallia siviilisäädyn, lasten lukumäärän sekä koulutuksen osalta. Uusia muuttujia ovat yrityksen koko sekä tehtävän haastavuus, ja potentiaalinen työkokemus on jätetty kokonaan pois mallista. Yrityksen koko tai mahdollisuus laajentua riippuu pitkälti sen toimialasta. Työtehtävän haastavuuteen puolestaan vaikuttaa henkilön omaama työkokemus sekä tämän saama koulutus. Jotkin työtehtävät ovat myös muita haastavampia, mikä puolestaan riippuu toimialasta.

Sukupuolen kausaalivaikutus palkkaan saadaan jälleen suoraan kausaalilaskennan toisen säännön avulla, jolloin $P_s(w) = P(w|s)$. Edelleen voidaan kuitenkin selvittää miten tilanne muuttuu, jos interventio kohdistetaan sukupuolen lisäksi koulutukseen. Soveltamalla algoritmin 1 implementaatiota saadaan sukupuolen ja koulutuksen kausaalivaikutukselle palkkaan seuraava lauseke:

$$
\begin{aligned}
P_{s,e}(w) = \sum_{m,x,i,c,d,z} \Big[& P(d|s,e,x,i)P(z|s,e,i)P(x|s,e)P(i|s,e)P(c|s,e,m) \\
& P(m|s,e) \sum_{e} \Big[P(w|s,e,m,x,i,c,d,z)P(e|s) \Big] \Big].
\end{aligned}
$$

Saatu lauseke sisältää osittain samoja termejä, kuin edellisen esimerkin yhteydessä saatu lauseke. Demografisiin tekijöihin liittyvät jälleen sukupuolten koulutusvuosien jakauma $P(e|s)$, lasten lukumäärän jakauma $P(c|s, e, m)$ ehdolla sukupuoli, koulutus ja siviilisääty, siviilisäädyn jakauma $P(m|s, e)$ ehdolla sukupuoli ja koulutus. Lisäksi samoja termejä ovat realisoituneen työkokemuksen jakauma $P(x|s, e)$ ehdolla sukupuoli ja koulutus sekä toimialan jakauma $P(i|s, e)$ ehdolla sukupuoli ja koulutus. Uusia mallinnettavia termejä ovat työtehtävän haastavuuden jakauma $P(d|s, e, x, i)$ ehdolla sukupuoli, koulutus, työkokemus ja toimiala sekä yrityksen koon jakauma $P(z|s, e, i)$ ehdolla sukupuoli, koulutus ja toimiala. Uudet termit voisivat antaa mahdollisesti edellistä mallia tarkemman kuvan siitä, miten yritysten toimialat ja työntekijöiden työtehtävät vaikuttavat palkkaan.

On syytä huomata, että edellä esitellyt kaksi kausaalimallia ovat vain esimerkkejä mahdollisista tavoista mallintaa sukupuolten välistä palkkaeroa. Vaikka malleissa on pyritty realistisuuteen, eivät ne välttämättä vastaa todellisuutta.

6.4 Kausaalivaikutusten lausekkeiden muodostuminen

Lukujen 6.1, 6.2 ja 6.3 esimerkkejä yhdistää algoritmin 1 tuottaman kausaalivaikutuksen lausekkeen monimutkaisuus. Implementaation hyödyntämisen kannalta on epäkäytännöllistä, että tuotettu lauseke sisältää aina jokaista havaittua muuttujaa jossain roolissa, sillä usein lauseke on todellisuudessa huomattavasti yksinkertaisempi. Ei ole selvää, milloin algoritmin tuottama lauseke on mahdollisimman yksinkertainen, mutta lausekkeen muodostumista voidaan tarkastella sekä algoritmin ominaisuuksien että graafin rakenteen kannalta.

Lausekkeen muodostuminen liittyy läheisesti C-komponentteihin, sillä niiden avulla kausaalivaikutus voidaan jakaa osiin algoritmin 1 riveillä neljä, kuusi ja seitsemän. Lisäksi sekoittavat tekijät vaikuttavat lausekkeen muodostumiseen. Havainnollistetaan tätä esimerkin avulla, jossa pyritään määrittämään kahden muuttujan välinen kausaalivaikutus, johon liittyy lisäksi sekoittava muuttuja.

Tarkastellaan aluksi kuvan 17(a) graafia G, jossa muuttujiin X ja Y liittyvä sekoittava muuttuja Z on havaittu, ja halutaan määrittää kausaalivaikutus $P_x(y)$. Algoritmin 1 implementaatio tuotaa kausaalivaikutukselle seuraavan lausekkeen:

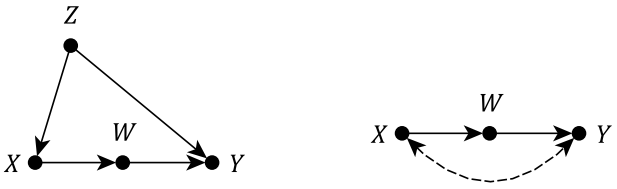
$$P_x(y) = \sum_{z,w} \Big[P(y|z, x, w) P(w|z, x) P(z) \Big].$$

Graafin $G[\mathbf{V} \setminus \{X\}]$ maksimaaliset C-komponentit $C(G[\mathbf{V} \setminus \{X\}])$, jotka muodostuvat solmujoukoista $\{Y\}$, $\{W\}$ ja $\{Z\}$, tuottavat summalausekkeen sisällä olevan tulon. Lisäksi lauseke sisältää jokaiseen havaittuun muuttujaan liitty-

viä termejä. Lauseketta voidaan kuitenkin muokata yksinkertaisemmaksi, sillä

$$\sum_{z,w}\Big[P(y|z,x,w)P(w|z,x)P(z)\Big] = \sum_{z,w}\Big[P(y,w|z,x)P(z)\Big] = \sum_{z}\Big[P(y|z,x)P(z)\Big].$$

Muuttuja W ei siis ollut tarpeellinen vaikutuksen lausekkeessa.



(a) Sekoittava havaittu muuttuja Z

(b) Sekoittava havaitsematon muuttuja

Kuva 17: Kausaalivaikutuksen lausekkeen muodostumiseen liittyvä esimerkki

Tilanne muuttuu, jos sekoittava tekijä onkin havaitsematon muuttuja, kuten kuvan 17(b) graafissa. Kun implementaatiota sovelletaan tässä graafissa kausaalivaikutuksen $P_x(y)$ identifioimiseksi, saadaan

$$P_x(y) = \sum_{w}\Big[P(w|x)\sum_{x}\Big[P(y|x,w)P(x)\Big]\Big].$$

Lauseketta ei voida tässä tapauksessa enää sieventää. Kaikki havaitut muuttujat olivat siis tarpeellisia vaikutuksen esittämiseksi.

On mahdollista, että algoritmia 1 tai sen implementaatiota voitaisiin kehittää siten, että tuotettu lauseke olisi aina minimaalinen kausaalivaikutuksen esittämiseen tarvittavien muuttujien lukumäärän suhteen.

7 Johtopäätökset

Tässä tutkielmassa esiteltiin Pearlin kausaalimalliin pohjautuva kausaalipäättelyn teoreettinen perusta ja kausaalilaskenta, jonka avulla on mahdollista tehdä päätelmiä kausaalimalleihin kohdistuvista interventioista. Lisäksi määriteltiin kausaalivaikutusten identifioituvuus ja lukuisia graafiteoreettisia käsitteitä algoritmisen lähestymistavan esittelemiseksi. Kausaalivaikutusten identifioituvuutta lähestyttiin kausaalilaskennan algoritmin avulla, jonka johtivat Shpitser ja Pearl (2006b). Algoritmi implementoitiin R-ohjelmointikielellä ja tämän toteutuksen yksityiskohtia tarkasteltiin eri näkökulmista.

Yhteyttä graafien ja niiden solmujoukkojen osajoukkojen välillä tarkennettiin käyttäen indusoituja aligraafeja, joita Shpitser ja Pearl (2006b) eivät hyödyntäneet. He käsittelivät graafeja ja solmujoukkoja tietyssä mielessä ekvivalentteina. Esimerkiksi tilanteessa, jossa $G = \langle \mathbf{V}, \mathbf{E} \rangle$ ja $\mathbf{W} \subset \mathbf{V}$, piti joukko $\mathbf{W}$ ymmärtää myös graafina, joka sisältää kaikki ne särmät joukosta $\mathbf{E}$, jotka yhdistävät joukon $\mathbf{W}$ solmuja. Tämä ajatusmalli johtaa yksinkertaisempiin merkintöihin, mutta algoritmin 1 toteutuksen kannalta solmujoukkojen ja graafien välinen ekvivalenssi on ohjelmoitava täsmällisesti.

Algoritmi 1 toteutettiin hyödyntäen olio-ohjelmointia. Kausaalivaikutuksen jakaumaa päivitetään rekursiivisesti, joten oli luonnollista määritellä jakaumaolio, joka ylläpitää kausaalijakauman lauseketta laskennan edetessä. Algoritmin 1 rivi kuusi on ongelmallinen, sillä se on kolmesta algoritmin terminoivasta rivistä selvästi monimutkaisin. Usean ehdollisista jakaumista muodostuvan tulotermin määrittäminen on haastavaa, sillä R-kieli ei ole erikoistunut symboliseen laskentaan. Nykyiset riviä kuusi koskevat sievennyssäännöt eivät ole optimaalisia, mutta tuottavat aina oikean lausekkeen kausaalivaikutukselle, mikäli se on identifioituva. Vastaavia ongelmia aiheutuu C-komponentteihin perustuvasta jakauman faktoroinnista algoritmin riveillä neljä ja seitsemän, joka voi yksinkertaisessakin tilanteessa tehdä kausaalivaikutuksen lausekkeesta vaikeasti ymmärrettävän.

Kausaalimalleja vastaavat graafit synnyttävät monissa tilanteissa kausaalisen Markov-ehdon ulkopuolisia ehdollisia riippumattomuuksia, joita Shpitser ja Pearl (2006b) eivät ottaneet huomioon määrittäessään algoritminsa. Kausaalivaikutusten lausekkeet sievenisivät monissa tilanteissa huomattavasti jos graafien riippumattomuusominaisuudet otettaisiin aina huomioon. Tämä on kuitenkin ongelmallista, sillä kaikkien mahdollisten graafissa vallitsevien ehdollisten riippumattomuuksien generointi on laskennallisesti raskas toimenpide.

Esitellyn implementaation tehokkuutta voitaisiin vielä analysoida esimerkiksi simulointikokein generoimalla satunnaisia suunnattuja silmukattomia graafeja, joiden solmujen ja särmien lukumäärät vaihtelevat. Voisi olla myös mielenkiintoista verrata suorituskykyä identifioituvien ja identifioitumattomien kausaalivaikutusten välillä. Nykyisessä implementaatiossa on pyritty käyttämään mahdollisimman tehokkaita R-paketteja graafitiedostojen ja niitä

vastaavien objektien käsittelyssä.

Kirjallisuudessa on esitetty myös muita graafiteoriaan pohjautuvia algoritmeja erilaisten kausaalimalleihin liittyvien kysymysten ratkaisemiseksi. Näitä ovat esimerkiksi:

- Ehdollisten kausaalivaikutusten identifioituvuusalgoritmi IDC (Shpitser ja Pearl, 2006a). Ehdolliset kausaalivaikutukset ovat muotoa $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$.

- Kausaalivaikutusten z-identifioituvuusalgoritmi ID^{Z} (Bareinboim ja Pearl, 2012). z-identifioituvuus liittyy tilanteeseen, jossa on mahdollista hyödyntää alkuperäisestä interventiojoukosta $\mathbf{X}$ erillistä joukkoa $\mathbf{Z}$ apuna identifioituvuuden määrittämisessä.

- Kausaalivaikutusten kuljetettavuusalgoritmi sID (Bareinboim ja Pearl, 2013a). Kuljetettavuus tarkoittaa sitä, että kokeellisessa tutkimuksessa hankittu informaatio voidaan yleistää koskemaan myös jotain populaatiota, jossa vain havainnoivat tutkimukset ovat mahdollisia.

- Kausaalivaikutusten metakuljetettavuusalgoritmi μsID (Bareinboim ja Pearl, 2013b). Metakuljetettavuus on kuljetettavuuden laajennus, jossa halutaan yleistää tuloksia useammasta kuin yhdestä kokeellisesta tutkimuksesta samanaikaisesti.

- Kontrafaktuaalien ja ehdollisten kontrafaktuaalien identifioituvuusalgoritmit ID* ja IDC* (Shpitser ja Pearl, 2007).

Näiden algoritmien implementoinnissa olisi myös mahdollista käyttää tässä työssä kehitettyjä ohjelmointiratkaisuja.

Lähteet

Angrist, J. D., Imbens, G. W. ja Rubin, D. B. (1996). "Identification of Causal Effects Using Instrumental Variables". *Journal of the American Statistical Association* 91 (434), s. 444–455.

Asplund, R. (2000). *Private returns to education in Finland: Back to basics.* The Research Institute of the Finnish Economy, Discussion papers no. 720. Helsinki.

Bareinboim, E. ja Pearl, J. (2013a). "A General Algorithm for Deciding Transportability of Experimental Results". *Journal of Causal Inference* 1 (1), s. 107–134.

— (2012). "Causal inference by surrogate experiments: z-identifiability". Teoksessa: *Proceedings of the Twenty-Eight Conference on Uncertainty in Artificial Intelligence.* Toim. N. de Freitas ja K. Murphy. AUAI Press, s. 113–120.

— (2013b). "Meta-Transportability of Causal Effects: A Formal Approach". Teoksessa: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, s. 135–143.

Becker, G. (1962). "Investment in human capital: A theoretical analysis". *Journal of Political Economy* 70 (5), s. 9–49.

Csardi, G. ja Nepusz, T. (2006). "The igraph software package for complex network research". *InterJournal* Complex Systems, s. 1695. URL: `http://igraph.org`.

Dawid, A. P. (1979). "Conditional independence in statistical theory". *Journal of the Royal Statistical Society. Series B (Methodological)* 41 (1), s. 1–31.

Davies, R. ja Pierre, R. (2005). "The family gap in pay in Europe: A cross-country study". *Labour Economics* 12 (4), s. 469–486.

Fields, J. ja Wolff, E. (1995). "Interindustry wage differentials and the gender wage gap". *Industrial and Labor Relations Review* 49 (1), s. 105–120.

Galles, D. ja Pearl, J. (1998). "An axiomatic characterization of causal counterfactuals". *Foundation of Science* 3 (1), s. 151–182.

Holland, P. W. (1986). "Statistics and causal inference". *Journal of the American Statistical Association* 81 (396), s. 945–960.

Huang, Y. ja Valtorta, M. (2006). "Pearl's calculus of intervention is complete". Teoksessa: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence.* AUAI Press, s. 217–224.

Kline, R. B. (1998). *Principles and Practice of Structural Equation Modeling.* New York: Guilford.

Koller, D. ja Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

Mincer, J. (1974). *Schooling, experience and earnings*. New York: NBER and Columbia University Press.

Napari, S. (2008). "The early-career gender wage gap among university graduates in the Finnish private sector". *Labour* 22 (4), s. 697–733.

Neyman, J. (1923). "Sur les applications de la théorie des probabilités aux experiences agricoles: Essai des principes". *Roczniki Nauk Rolniczych* 10, s. 1–51.

Pearl, J. (1995). "Causal diagrams for empirical research". *Biometrika* 82 (4), s. 669–688.

— (2009). *Causality: Models, Reasoning and Inference*. 2nd edition. New York: Cambridge University Press.

Pollman-Schult, M. (2011). "Marriage and earnings: Why do married men earn more than single men?" *European Sociological Review* 27 (2), s. 147–163.

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: `http://www.R-project.org/`.

Rubin, D. B. (1974). "Estimating causal effects of treatments in randomized and nonrandomized studies". *Journal of Educational Psychology* 66 (5), s. 688–701.

Shpitser, I. ja Pearl, J. (2006a). "Identification of conditional interventional distributions". Teoksessa: *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI2006)*. AUAI Press, s. 437–444.

— (2006b). "Identification of Joint Interventional Distributions in Recursive semi-Markovian Causal Models". Teoksessa: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*. Boston, Massachusetts: AAAI Press, s. 1219–1226.

— (2007). "What counterfactuals can be tested". Teoksessa: *Proceedings of Twenty Third Conference on Uncertainty in Artificial Intelligence*. Vancouver, Canada, s. 352–359.

Simpson, J. E. (2002). *XPath and XPointer: Locating Content in XML Documents*. Sebastopol, CA, USA: O'Reilly & Associates, Inc.

Temple Lang, D. (2013). *XML: Tools for parsing and generating XML within R and S-Plus.* R package version 3.98-1.1. URL: `http://CRAN.R-project.org/package=XML`.

Tian, J. (2002). ”Studies in Causal Reasoning and Learning”. Väitöskirja. Department of Computer Science, Univerisity of California, Los Angeles.

Tian, J. ja Pearl, J. (2002). ”A general identification condition for causal effects”. Teoksessa: *Proceedings of the 18th National Conference on Artificial Intelligence*. AAAI/The MIT Press, s. 567–573.

— (2003). *On the identification of causal effects*. Tekninen raportti. R-290-L. Department of Computer Science, University of California, Los Angeles.

Verma, T. S. (1993). *Graphical aspects of causal models*. Tekninen raportti. R-191. Department of Computer Science, Univerisity of California, Los Angeles.

Liite A: R-koodi

```
1  # Santtu Tikka
2  # 17.9.2014
3
4  library(igraph)
5  library(XML)
6
7  probability <- function(var = c(), cond = c(), sumset
       = c(), recursive = FALSE, children = list(),
     fraction = FALSE, divisor = list()) {
8    p <- list(var = var, cond = cond, sumset = sumset,
       recursive = recursive, children = children,
       fraction = fraction, divisor = divisor)
9    class(p) = "probability"
10   return(p)
11 }
12
13 causal.effect <- function(y, x, G) {
14   if (!is.dag(observed.graph(G))) stop("Graph 'G' is
       not a DAG")
15   to <- topological.sort(observed.graph(G))
16   to <- get.vertex.attribute(G, "name")[to]
17   if (length(setdiff(y, to)) > 0) stop("Set 'y'
       contains variables not present in the graph.")
18   if (length(setdiff(x, to)) > 0) stop("Set 'x'
       contains variables not present in the graph.")
19   res <- id(y, x, probability(), G, to)
20   res <- organizeTerms(res)
21   res <- getExpression(res)
22   return(res)
23 }
24
25 id <- function(y, x, P, G, to) {
26   G.obs <- observed.graph(G)
27   e <- get.edge.attribute(G, "name")
28   v <- get.vertex.attribute(G, "name")
29   v <- to[which(to %in% v)]
30   anc <- ancestors(y, G.obs, to)
31
32   # line 1
33   if (length(x) == 0) {
34     P$sumset <- union(setdiff(v, y), P$sumset)
35     P$var <- v
```

```
36      return(P)
37    }
38
39    # line 2
40    if (length(setdiff(v, anc)) != 0) {
41      anc.graph <- induced.subgraph(G, anc)
42      var.nonign <- getNonIgnorableNodes(P)
43      P$sumset <- intersect(setdiff(v, anc), var.nonign
          )
44      return(id(y, intersect(x, anc), P, anc.graph, to)
          )
45    }
46
47    # line 3
48    edges.to.x <- E(G) [1:length(E(G)) %->% x]
49    G.x.overbar <- subgraph.edges(G, E(G)[setdiff(E(G),
        edges.to.x)], delete.vertices = FALSE)
50    w <- setdiff(setdiff(v, x), ancestors(y, observed.
       graph(G.x.overbar), to))
51    if (length(w) != 0) return(id(y, union(x, w), P, G,
        to))
52
53
54    # line 4
55    G.remove.x <- induced.subgraph(G, v[!(v %in% x)])
56    s <- c.components(G.remove.x, to)
57    if (length(s) > 1) {
58      productlist <- list()
59      for (i in 1:length(s)) productlist[[i]] <- id(s[[
          i]], setdiff(v, s[[i]]), P, G, to)
60      return(probability(sumset = setdiff(v, union(y, x
          )), recursive = TRUE, children = productlist))
61    } else {
62      s <- s[[1]]
63
64      # line 5
65      cG <- c.components(G, to)
66      if (identical(cG[[1]], v)) {
67        v.string <- paste(v, sep = "", collapse = ",")
68        s.string <- paste(s, sep = "", collapse = ",")
69        stop("Graph contains a hedge formed by C-
            forests of nodes: \n {", v.string , "} and {
            ", s.string , "}.", call. = FALSE)
70      }
```

```
71
72      # line 6
73      is.element <- FALSE
74      for (i in 1:length(cG)) {
75        if(identical(s, cG[[i]])) {
76          is.element <- TRUE
77          break
78        }
79      }
80      if (is.element) {
81        ind <- which(v %in% s)
82        if (length(s) > 1) {
83          productlist <- list()
84          for (i in 1:length(s)) {
85            if (P$recursive) {
86              productlist[[i]] <- parse.joint(P, s[i],
                    v[0:(ind[i]-1)], v)
87            } else {
88              P.prod <- P
89              P.prod$var <- s[i]
90              P.prod$cond <- v[0:(ind[i]-1)]
91              productlist[[i]] <- P.prod
92            }
93          }
94          return(probability(sumset = setdiff(s, y),
                recursive = TRUE, children = productlist))
95        } else {
96          if (P$recursive) {
97            P.prod <- parse.joint(P, s[1], v[0:(ind
                  [1]-1)], v)
98            P.prod$sumset <- union(P.prod$sumset,
                  setdiff(s, y))
99            return(P.prod)
100         } else {
101           P.prod <- P
102           P.prod$var <- s[1]
103           P.prod$cond <- v[0:(ind[1]-1)]
104           P.prod$sumset <- union(P.prod$sumset,
                  setdiff(s, y))
105           return(P.prod)
106         }
107       }
108     }
109
```

```
110     # line 7
111     set.contain <- unlist(lapply(cG, FUN = function(x
            ) setequal(intersect(x, s), s)))
112     set.contain <- which(set.contain)[1]
113     s <- cG[[set.contain]]
114     productlist <- list()
115     ind <- which(v %in% s)
116     s.graph <- induced.subgraph(G, s)
117     if (length(s) > 1) {
118       for (i in 1:length(s)) {
119         P.prod <- P
120         P.prod$var <- s[i]
121         P.prod$cond <- v[0:(ind[i]-1)]
122         productlist[[i]] <- P.prod
123       }
124       return(id(y, intersect(x, s), probability(
              recursive = TRUE, children = productlist), s
              .graph, to))
125     } else {
126       P.prod <- P
127       P.prod$var <- s[1]
128       P.prod$cond <- v[0:(ind[1]-1)]
129       return(id(y, intersect(x, s), P.prod, s.graph,
              to))
130     }
131   }
132 }
133
134 ancestors <- function(y, G, to) {
135   neighbors <- unique(unlist(neighborhood(G, order =
         vcount(G), nodes = y, mode = "in")))
136   v <- V(G)[neighbors]$name
137   v <- to[which(to %in% v)]
138   return(v)
139 }
140
141 c.components <- function(G, to) {
142   A <- as.matrix(get.adjacency(G))
143   v <- get.vertex.attribute(G, "name")
144   e <- E(G)
145   bidirected <- c()
146   indices <- which(A >= 1 & t(A) >= 1, arr.ind = TRUE
        )
147   c.comp <- list()
```

```
148   if (nrow(indices) > 0) {
149     for (i in 1:nrow(indices)) {
150       bidirected <- c(bidirected, e[v[indices[i,1]]
            %->% v[indices[i,2]]])
151     }
152   }
153   G.bidirected <- subgraph.edges(G, bidirected,
        delete.vertices = FALSE)
154   subgraphs <- decompose.graph(G.bidirected)
155   for (i in 1:length(subgraphs)) {
156     v.sub <- get.vertex.attribute(subgraphs[[i]], "
          name")
157     v.sub <- to[which(to %in% v.sub)]
158     c.comp[[i]] <- v.sub
159   }
160   return(c.comp)
161 }
162
163 observed.graph <- function(G) {
164   obs.edges <- setdiff(1:length(E(G)), which(edge.
        attributes(G)$description == "U"))
165   G.obs <- subgraph.edges(G, E(G)[obs.edges], delete.
        vertices = FALSE)
166   return(G.obs)
167 }
168
169 parse.joint <- function(P, v, cond, var) {
170   P.num <- P
171   P.num$sumset <- c(union(P$sumset, setdiff(var,
        union(v, cond))))
172   P.num <- parse.expression(P.num)
173   if (length(cond) > 0) {
174     P.den <- P
175     P.den$sumset <- c(union(P$sumset, setdiff(var,
          cond)))
176     P.den <- parse.expression(P.den)
177     i <- 1
178     k <- 0
179     while (i <= length(P.num$children) & length(P.num
          $children) > 0 & length(P.den$children) > 0) {
180       is.element <- FALSE
181       for (j in 1:length(P.den$children)) {
182         if (identical(P.num$children[[i]], P.den$
              children[[j]])) {
```

```
183             is.element <- TRUE
184             k <- j
185             break
186           }
187         }
188         if (is.element) {
189           P.den$children[[k]] <- NULL
190           P.num$children[[i]] <- NULL
191           i <- 0
192         }
193         i <- i + 1
194       }
195       if (length(P.den$children) > 0) {
196         P.num$fraction <- TRUE
197         P.num$divisor <- P.den
198       }
199     }
200     return(P.num)
201   }
202
203   parse.expression <- function(P) {
204     P.parse <- probability(recursive = TRUE, children =
          list())
205     j <- 1
206     remove <- c()
207     for (i in 1:length(P$children)) {
208       if (length(intersect(P$children[[i]]$var, P$
            sumset)) == 0 & length(intersect(P$children[[i
            ]]$cond, P$sumset)) == 0) {
209         P.parse$children[[j]] <- P$children[[i]]
210         remove <- c(remove, i)
211         j <- j + 1
212       }
213     }
214     P$children[remove] <- NULL
215     if (length(P$children) == 0) return(P.parse)
216     k <- 0
217     j <- 0
218     while (k <= length(P$sumset) & length(P$sumset) > 0
          & length(P$children) > 0) {
219       k <- k + 1
220       count <- 0
221       for (i in 1:length(P$children)) {
```

```
222       if (length(intersect(P$children[[i]]$var, P$
              sumset[k])) == 0 & length(intersect(P$
              children[[i]]$cond, P$sumset[k])) == 0) {
223         count <- count + 1
224         j <- i
225       }
226     }
227     if (count == 1) {
228       P$sumset <- P$sumset[-k]
229       P$children[[j]] <- NULL
230       k <- 0
231     }
232   }
233   if (length(P$children) == 0) return(P.parse)
234   else P.parse$children[[length(P.parse$children)+1]]
          <- P
235   return(P.parse)
236 }
237
238 getExpression <- function(x) {
239   UseMethod("getExpression", x)
240 }
241
242 getExpression.probability <- function(x) {
243   P <- ""
244   s.print <- length(x$sumset) > 0
245   sum.string <- ""
246   cond.string <- ")"
247   if (x$fraction) P <- "\\frac{"
248   if (s.print) {
249     sum.string <- paste(tolower(x$sumset), sep = "",
          collapse = ",")
250     P <- paste(P, "\\sum_{", sum.string, "}\\bigl[",
          sep = "", collapse = "")
251   }
252   if (x$recursive) {
253     for (i in 1:length(x$children)) P <- paste(P,
          getExpression(x$children[[i]]), sep = "",
          collapse = ",")
254   } else {
255     var.string <- paste(tolower(x$var), sep = "",
          collapse = ",")
256     P <- paste(P, "P(", var.string, sep = "",
          collapse = "")
```

```
257     if (length(x$cond) > 0) {
258       cond.string <- paste(tolower(x$cond), sep = "",
              collapse = ",")
259       cond.string <- paste("\\vert ", cond.string, ")
            ", sep = "", collapse = ",")
260     }
261     P <- paste(P, cond.string, sep = "", collapse = "
          ,")
262   }
263   if (s.print) P <- paste(P, "\\bigr]", sep = "",
        collapse = ",")
264   if (x$fraction) {
265     P <- paste0(P, "}{")
266     P <- paste(P, getExpression(x$divisor), sep = "",
          collapse = ",")
267     P <- paste0(P, "}")
268   }
269   return(P)
270 }
271
272 getNonIgnorableNodes <- function(x) {
273   UseMethod("getNonIgnorableNodes", x)
274 }
275
276 getNonIgnorableNodes.probability <- function(x) {
277   var <- c()
278   if (x$recursive) {
279     for (i in 1:length(x$children)) var <- c(var,
          getNonIgnorableNodes(x$children[[i]]))
280   } else {
281     var <- c(var, x$var, x$cond)
282   }
283   return(var)
284 }
285
286 organizeTerms <- function(x) {
287   UseMethod("organizeTerms", x)
288 }
289
290 organizeTerms.probability <- function(obj) {
291   if (obj$recursive) {
292     children.copy <- obj$children
293     obj$children <- list()
```

```
294     rec <- unlist(lapply(children.copy, FUN =
          function(x) x$recursive))
295     children.rec <- children.copy[rec]
296     if (length(children.rec) > 0) for(i in 1:length(
          children.rec)) obj$children[[i]] <-
          organizeTerms(children.rec[[i]])
297     children.nonrec <- children.copy[!rec]
298     if (length(children.nonrec) > 0) {
299       ord <- order(unlist(lapply(children.nonrec, FUN
            = function(x) length(x$cond))), decreasing
          = TRUE)
300       obj$children <- c(children.nonrec[ord], obj$
          children)
301     }
302     if (obj$fraction) obj$divisor <- organizeTerms(
          obj$divisor)
303   }
304   return(obj)
305 }
306
307 parse.graphml <- function(file, format = c("standard"
    , "internal"), nodes = c(), use.names = TRUE) {
308   format <- match.arg(format)
309   res <- switch(format, standard = parse.graphml.
       standard(file, nodes, use.names),
310                 internal = parse.graphml.internal(
                    file, nodes, use.names),
311                 stop(paste("Unknown file format:",
                    format)))
312   return(res)
313 }
314
315 parse.graphml.standard <- function(file, nodes, use.
    names) {
316   doc <- xmlParse(file, useInternalNodes = TRUE)
317   top <- xmlRoot(doc)
318   graph <- top[["graph"]]
319   if (!use.names) {
320     if (xmlSize(which(xmlSApply(graph, xmlName) == "
          node")) != length(nodes)) stop("Incorrect
          number of node names")
321   }
322   ns <- c(ns = "http://graphml.graphdrawing.org/xmlns
       ")
```

```
323   edges <- which(xmlSApply(graph, xmlName) == "edge")
324   remove.id <- 0
325   removals <- list()
326   for (i in 1:length(edges)) {
327     current.edge <- graph[[edges[i]]]
328     edge.attr <- xmlAttrs(current.edge)
329     datas <- which(xmlSApply(current.edge, xmlName)
          == "data")
330     for (j in 1:length(datas)) {
331       current.data <- current.edge[[datas[j]]]
332       if (xmlAttrs(current.data) == "d10") {
333         arc <- current.data[[1]]
334         arw.attr <- xmlAttrs(arc[["Arrows"]])
335         src <- edge.attr[["source"]]
336         trgt <- edge.attr[["target"]]
337         two.arrows <- !identical(arw.attr[["source"
              ]], "none") & !identical(arw.attr[["target
              "]], "none")
338         no.arrows <- identical(arw.attr[["source"]],
              "none") & identical(arw.attr[["target"]],
              "none")
339         if (two.arrows | no.arrows) {
340           remove.id <- remove.id + 1
341           removals[[remove.id]] <- current.edge
342           e1 <- newXMLNode("edge", parent = doc,
                attrs = c(id = "e", source = src, target
                 = trgt),
343                     newXMLNode("data", attrs = c(key =
                        "d9"), cdata = TRUE, "U" ))
344           e2 <- newXMLNode("edge", parent = doc,
                attrs = c(id = "e", source = trgt,
                target = src),
345                     newXMLNode("data", attrs = c(key =
                        "d9"), cdata = TRUE, "U" ))
346           addChildren(graph, kids = list(e1, e2))
347         }
348       }
349     }
350   }
351   removeNodes(removals)
352   if (use.names) {
353     node.data <- getNodeSet(doc, "//ns:data[contains(
          @key,'d6')]/*", ns)
```

```
354     for (i in 1:length(node.data)) nodes[i] <-
          xmlValue(node.data[[i]]["NodeLabel"]$NodeLabel
          [1]$text)
355   }
356   all <- getNodeSet(doc, "//*[position() > 1]", ns)
357   keep <- getNodeSet(doc, "//ns:edge | //ns:node | //
        ns:graph | //ns:key[@attr.name] | //ns:data[
        contains(@key,'d9')]", ns)
358   removeNodes(setdiff(all, keep))
359   temp <- tempfile(fileext = ".graphml")
360   temp.xml <- saveXML(doc, file = temp)
361   free(doc)
362   igrph <- read.graph(temp.xml, format = "graphml")
363   igrph <- set.vertex.attribute(igrph, "name", value
        = nodes)
364   return(igrph)
365 }
366
367 parse.graphml.internal <- function(file, nodes, use.
      names) {
368   doc <- xmlParse(file, useInternalNodes = TRUE)
369   top <- xmlRoot(doc)
370   graph <- top[["graph"]]
371   if (!use.names) {
372     if (xmlSize(which(xmlSApply(graph, xmlName) == "
          node")) != length(nodes)) stop("Incorrect
          number of node names")
373   }
374   ns <- c(ns = "http://graphml.graphdrawing.org/xmlns
        ")
375   if (use.names) {
376     node.data <- getNodeSet(doc, "//ns:data[contains(
          @key,'d6')]/*", ns)
377     for (i in 1:length(node.data)) nodes[i] <-
          xmlValue(node.data[[i]]["NodeLabel"]$NodeLabel
          [1]$text)
378   }
379   all <- getNodeSet(doc, "//*[position() > 1]", ns)
380   keep <- getNodeSet(doc, "//ns:edge | //ns:node | //
        ns:graph | //ns:key[@attr.name] | //ns:data[
        contains(@key,'d9')]", ns)
381   removeNodes(setdiff(all, keep))
382   temp <- tempfile(fileext = ".graphml")
383   temp.xml <- saveXML(doc, file = temp)
```

```
384   free(doc)
385   igrph <- read.graph(temp.xml, format = "graphml")
386   igrph <- set.vertex.attribute(igrph, "name", value
        = nodes)
387   return(igrph)
388 }
```

identify.R

Liite B: causaleffect-paketin dokumentaatio

Package 'causaleffect'

September 25, 2014

Version 1.0

Date 2014-09-17

Title Deriving Expressions of Joint Interventional Distributions in Causal Models

Author Santtu Tikka

Maintainer Santtu Tikka <santtuth@gmail.com>

Imports igraph, XML

Description An implementation of the complete identification algorithm constructed by Ilya Shpitser and Judea Pearl (2006) for deriving expressions of joint interventional distributions in causal models, which contain unobserved variables and induce directed acyclic graphs.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-25 22:42:50

R topics documented:

`causaleffect-package` *Deriving Expressions of Joint Interventional Distributions in Causal Models*

Description

Causal calculus is concerned with estimating the interventional distribution of some action from the observed joint probability distribution of the variables in a given causal structure. All identifiable causal effects can be derived using the rules of do-calculus, but the rules themselves do not give any direct indication whether the effect in question is identifiable or not. Ilya Shpitser and Judea Pearl (2006) constructed an algorithm for identifying joint interventional distributions in causal models, which contain unobserved variables and induce directed acyclic graphs. This algorithm can be seen as a repeated application of the rules of do-calculus and known properties of probabilities, that ultimately either derives an expression for the causal distribution, or fails to identify the effect, in which case the effect is non-identifiable. causaleffect provides an implementation of this algorithm.

Details

Package:	causaleffect
Type:	Package
Version:	1.0
Date:	2014-09-17
License:	GPL-2

The function `causal.effect` receives two character vectors describing the variables of interest and an `igraph` (package `igraph`) object as arguments and returns a string describing the interventional distribution in LaTeX syntax if the effect is identifiable.

Author(s)

Santtu Tikka <santtuth@gmail.com>

References

Pearl J. 2009 *Causality: Models, Reasoning and Inference*, New York: Cambridge University Press.

Shpitser I., Pearl J. 2006 Identification of Joint Interventional Distributions in Recursive semi-Markovian Causal Models. *Proceedings of the 21st National Conference on Artificial Intelligence*, **2**, 1219–1226.

`causal.effect` *Identify a causal effect*

Description

This function returns an expression for the joint distribution of the set of variables y given the intervention on the set of variables x if the effect is identifiable. Otherwise an error is thrown describing the graphical structure that witnesses non-identifiability.

Usage

```
causal.effect(y, x, G)
```

Arguments

y	A character vector of variables of interest given the intervention.
x	A character vector of the variables that are acted upon.
G	An igraph object created by the function parse.graphml that describes the directed acyclic graph induced by the causal model.

Value

A character string that describes the interventional distribution in LaTeX syntax.

Author(s)

Santtu Tikka

References

Shpitser I., Pearl J. 2006 Identification of Joint Interventional Distributions in Recursive semi-Markovian Causal Models. *Proceedings of the 21st National Conference on Artificial Intelligence*, **2**, 1219–1226.

See Also

parse.graphml

Examples

```
library(igraph)

# simplify = FALSE to allow multiple edges
g <- graph.formula(X -+ Y, Z -+ X, Z -+ Y , X -+ Z, Z -+ X, simplify = FALSE)

# Here the bidirected edge between X and Z is set to be unobserved in graph g
# This is denoted by giving them a description attribute with the value "U"
# The edges in question are the fourth and the fifth edge
g <- set.edge.attribute(graph = g, name = "description", index = c(4,5), value = "U")

res <- causal.effect("Y", "X", g)
```

`parse.graphml` *Prepare graphml files for internal use*

Description

This function reads graphml files created by the yEd graph editor, which describe directed acyclic graphs. The R-package XML is utilized to parse the contents of the files to suit the internal format used by `causal.effect`. Bidirected arcs are replaced by two unobserved unidirected arcs, and the resulting XML file is coerced into an igraph object. This function also serves as a wrapper for files that already correspond to the internal format. Names for the nodes of the graph can be supplied or read directly from the input file.

Usage

```
parse.graphml(file, format = c("standard", "internal"),
    nodes = c(), use.names = TRUE)
```

Arguments

`file`	The connection to read from.
`format`	A character constant describing how bidirected arcs are denoted in the graphml file. Option `standard` corresponds to bidirected arcs that are notated with a graphical parameter describing an arrow at each end of the arc or no arrows at all. Option `internal` matches the format that `standard` graphs are coerced into. This option should be used only if all bidirected arcs in the graph are denoted by two unidirected arcs which have a description parameter of a single character "U" (shorthand for "unobserved").
`nodes`	A character vector that describes the names of the nodes in the graph. This is ignored if `use.names` is `TRUE`.
`use.names`	A logical value indicating whether the names of the nodes should be read from the file or not.

Value

An object of class igraph that describes the causal diagram. The parsed graph can now be used by `causal.effect`.

Author(s)

Santtu Tikka

Index