Tampereen yliopisto

Aleksi Tastula

FORD–FULKERSON-ALGORITMI

Virtaussysteemin maksimivirtauksen löytäminen

Kandidaatintyö
Tekniikan ja luonnontieteiden tiedekunta
Tarkastaja: Petteri Laakkonen
Tammikuu 2022

TIIVISTELMÄ

Aleksi Tastula: Ford–Fulkerson-algoritmi
Kandidaatintyö
Tampereen yliopisto
Teknis-luonnontieteellinen tutkinto-ohjelma
Tammikuu 2022

Graafiteoriaa voidaan käyttää työkaluna lukuisten reaalimaailman ongelmien ja tilanteiden mallintamisessa, sillä sitä voidaan hyödyntää aina sosiaalisen median analysoinnista logistiikkaketjujen optimoimiseen. Jälkimmäisen kaltaisiin käyttötarkoituksiin sovelletaan graafiteorian virtaussysteemeitä. Tässä työssä tutustutaan graafiteorian virtaussysteemien maksimivirtauksen selvittämiseen Ford–Fulkerson-algoritmin avulla. Tätä algoritmia hyödyntämällä virtaussysteemeistä voidaan selvittää maksimivirtaus ja virtausta rajoittava "pullonkaula". Tämän tiedon avulla saadaan selvitettyä esimerkiksi olemassa olevan logistiikan infrastuktuurin maksimikapasiteetti ja kehitysalueet, joilla maksimikapasiteettia saadaan kasvatettua.

Tämä työ tarjoaa lukijalle kaikki tarvittavat graafiteorian työkalut Ford–Fulkerson-algoritmin ymmärtämistä ja käyttämistä varten. Työn alussa esitellään oleelliset graafiteorian perusteet, minkä jälkeen esitellään graafien erityistapaukset: puut ja niiden kasvatusalgoritmit.

Tässä työssä tarkasteltavat virtaussysteemit on yksinkertaistettu siten, että systeemeissä on vain ja ainoastaan yksi virtauksen lähdesolmu ja yksi nielusolmu. Kaikki systeemin virtaus alkaa lähteestä ja kulkee systeemin muita solmuja ja kaaria pitkin nieluun. Tätä lähteestä nieluun kulkevaa virtausta kutsutaan systeemin kokonaisvirtaukseksi. Systeemin suurinta mahdollista kokonaisvirtausta kutsutaan maksimivirtaukseksi, jota rajoittavat systeemin yksittäisten kaarien kapasiteettiehdot. Maksimivirtauksen löytämisessä kokonaisvirtausta kasvatetaan niin kauan, että systeemin virtaus kohtaa sitä kuljettavien kaarien kapasiteetit eikä kokonaisvirtausta enää voida kasvattaa.

Systeemin maksimivirtaus on siis suurin mahdollinen siinä kulkevan virtauksen määrä systeemin kaarien kapasiteettien sallimissa rajoissa. Kaarijoukkoa, joka erottaa lähteen ja nielun toisistaan, kutsutaan leikkaukseksi. Sitä lähteen ja nielun erottavaa leikkausta, jonka kaarien yhteenlaskettu kapasiteetti on pienin, kutsutaan virtaussysteemin minimileikkaukseksi. Mikäli systeemissä kulkee maksimivirtaus, minimileikkauksen kaarien koko virtauskapasiteetti on käytössä, eli ne ovat saturoituja. Minimileikkaus on systeemin virtauksen "pullonkaula", jota muuttamalla voidaan vaikuttaa systeemin maksimivirtaukseen.

Virtaussysteemien käsittelyn jälkeen käydään läpi perusteellisesti Ford–Fulkerson-algoritmin vaiheet ja kulku. Algoritmin kulkua havainnollistetaan esimerkin avulla. Algoritmin toimivuutta osoitetaan näyttämällä, että kun algoritmi päättyy, se päättyy aina maksimivirtaukseen.

Avainsanat: graafiteoria, max-flow, min-cut, Ford–Fulkerson, algoritmi

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

SISÄLLYSLUETTELO

1. Johdanto 1
2. Graafiteorian perusteet 2
2.1 Graafit, solmut ja viivat 2
2.2 Polut ja komponentit 3
2.3 Suunnatut graafit eli digraafit 6
2.4 Leikkaukset 7
3. Puut ja puiden kasvatusalgoritmit 9
3.1 Leveyssuuntainen läpikäynti 10
3.2 Syvyyssuuntainen läpikäynti 11
4. Virtaussysteemit 13
5. Ford–Fulkerson-algoritmi 19
6. Yhteenveto 29
Lähteet 30

LYHENTEET JA MERKINNÄT

A	kaarijoukko
$c(a)$	kaaren a kapasiteetti
D	digraafi
E	viivajoukko
$\epsilon(a)$	kaaren a maksimikasvatusarvo
$\epsilon(P)$	polun P maksimikasvatusarvo
$\|f_{max}\|$	maksimivirtaus
f	virtaus
$f(a)$	kaaren a virtaus
$f(A)$	kaarijoukon A virtaus
$f^+(X)$	virtaus solmujoukosta X
$f^-(X)$	virtaus solmujoukkoon X
$\|f\|$	systeemin kokonaisvirtaus
G	graafi
I	Virtaussysteemin välittäjäsolmujoukko
K_{min}	minimileikkaus
$N(x, y)$	virtaussysteemi, jossa solmu x on lähde ja y nielu.
$p(v)$	solmun v edeltäjäfunktio
T	puu
V	solmujoukko
$\partial(X)$	solmujoukon X erottava leikkaus
$\partial^+(X)$	leikkauksen $\partial(X)$ solmujoukosta X poispäin suuntautuvien kaarien joukko
$\partial^-(X)$	leikkauksen $\partial(X)$ solmujoukkoon X päin suuntautuvien kaarien joukko
x-puu	Puu, jonka juuri on solmu x
xTy	Polku solmusta x solmuun y puussa T

1. JOHDANTO

Graafiteoria on tehokas matemaattinen tapa kuvata, tulkita ja havainnollistaa monia reaalimaailman systeemejä. Esimerkiksi liikennettä, kuljetusjärjestelmiä, tietoliikenneverkkoja, verkkosivujen yhteyttä ja sosiaalisia verkostoja voidaan kuvata graafien avulla [1]. Graafit ovat usein hyvin visuaalisia, mutta pohjimmiltaan ne ovat kombinatoriikkaa. Graafien visuaalinen puoli on hyvin tärkeä, sillä se auttaa niiden ymmärtämisessä ja tulkinnassa. Visuaalisella tulkinnalla ja päättelyllä monet asiat graafiteoriassa ovat hyvin intuitiivisia, kun taas matemaattiset laskut graafien visuaalisuuden taustalla voivat olla haastavia.

Graafiteoriassa virtaussysteemit ovat verkostojen virtausta kuvaavia graafeja, joilla voidaan kuvata esimerkiksi sähkönsiirtoverkkoja, verkkoliikennettä ja vesijohtoverkoston toimintaa [2]. Tässä työssä tutustutaan yhteen graafiteorian virtausongelmaan: maksimivirtauksen löytämiseen, jossa hyödynnetään Ford–Fulkerson-algoritmia. Tämä työ tarjoaa lukijalle tarvittavat graafiteorian työkalut Ford–Fulkerson-algoritmin ymmärtämiseen ja käyttämiseen virtaussysteemin maksimivirtauksen löytämiseksi. Graafiteorian ei siis tarvitse olla lukijalle ennestään tuttu matematiikan osa-alue.

Virtaussysteemin sellaisten virtausta kantavien viivojen joukkoa, joka rajoittaa kokonaisvirtauksen määrää, kutsutaan minimileikkaukseksi. Minimileikkauksella ja virtaussysteemin suurimmalla mahdollisella virtauksella eli maksimivirtauksella on yhteys. Tämän yhteyden avulla Ford–Fulkerson-algoritmin antama virtaus voidaan osoittaa virtaussysteemin maksimivirtaukseksi.

Työn toisessa luvussa esitellään Ford–Fulkerson-algoritmin kannalta oleelliset graafiteorian perusteet, kuten suunnatut graafit ja leikkaukset. Kolmannessa luvussa käydään läpi puut ja niiden kasvatusalgoritmit polkujen muodostamista varten. Neljännessä luvussa suunnatusta graafista muodostetaan virtaussysteemi ja esitellään virtaussysteemien säännöt ja ominaisuudet. Luvussa käydään läpi myös systeemin maksimivirtaus ja minimileikkaus ja osoitetaan niiden välinen yhteys. Virtaussysteemien esittelyn jälkeen viidennessä luvussa käydään läpi Ford–Fulkerson-algoritmi. Yksityiskohtaisessa esimerkissä selvitetään virtaussysteemin maksimivirtaus Ford–Fulkerson-algoritmin avulla. Lopuksi osoitetaan Ford–Fulkerson-algoritmi toimivaksi virtausta rajoittavien kapasiteettiehtojen ollessa kokonais- ja rationaalilukuja.

2. GRAAFITEORIAN PERUSTEET

Graafiteoria sisältää useita eri termejä ja käsitteitä. Tässä luvussa käsitellään graafiteorian perusteista vain tälle työlle olennaiset termit ja käsitteet. Graafiteorian suomenkieliset termit eivät ole vakiintuneita [3], joten tässä työssä käytetyt termit eivät välttämättä vastaa muuta kirjallisuutta.

2.1 Graafit, solmut ja viivat

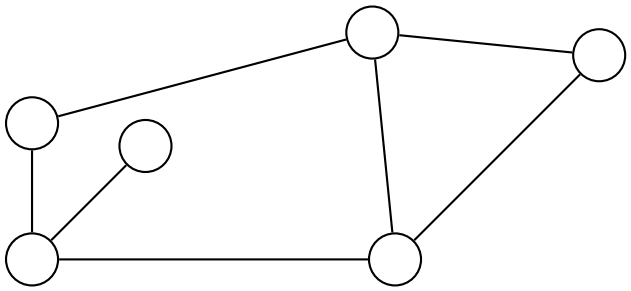
Määritelmä 2.1. *Graafi* on järjestetty pari $G = (V, E)$, joka koostuu *solmujen* (eng. *vertex*) joukosta $V = \{v_1, v_2, \ldots, v_n\}$ ja niitä yhdistävien *viivojen* (eng. *edge*) joukosta E. Viivajoukon E alkiot ovat joukon V järjestämättömiä pareja, ja niitä merkitään $\{v_i, v_j\}$, missä v_i ja v_j ovat joukon V alkioita.

Visuaalisesti solmut esitetään graafissa ympyröinä ja viivat ympyröitä yhdistävinä viivoina kuvan 2.1 kaltaisesti. Viiva e *yhdistää* solmut u ja v, mikäli $e = \{u, v\}$. Tällöin solmut u ja v ovat viivan e *päätesolmut* ja näin ollen myös vierekkäiset solmut. Mikäli viivan kumpikin päätesolmu on sama (esim viiva $e = \{v, v\}$), viiva muodostaa *silmukan*. Mikäli viivat yhdistävät saman solmuparin, viivat ovat rinnakkaiset. [4]

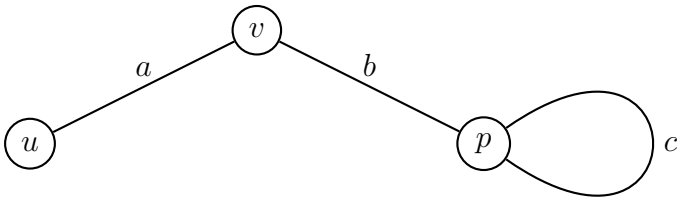
Esimerkki 2.1. *Tutkitaan kuvan 2.2 graafin $G(V, E)$ rakennetta.*

Graafin G

- *solmujoukko $V = \{u, v, p\}$*
- *viivajoukko $E = \{a, b, c\}$*
- *solmut u ja v ovat viivan a päätesolmut*



Kuva 2.1. *Graafi.*



Kuva 2.2. *Graafi* $G(V, E)$.

- *solmut* u *ja* v *ovat vierekkäiset*
- *viivat* a *ja* b *ovat vierekkäiset*
- *viiva* c *muodostaa silmukan.*

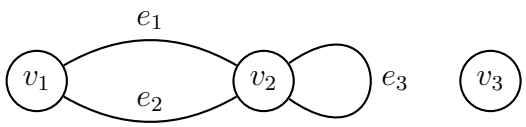
Kaikkien graafin solmujen ei tarvitse liittyä viivaan, mutta jokaisen graafin viivan *tulee* liittyä solmuun. Yhteen solmuun voidaan liittää useita viivoja.

Määritelmä 2.2. Graafi on *yksinkertainen*, jos se ei sisällä rinnakkaisia viivoja eikä silmukoita.

Määritelmä 2.3. Graafi $H(V_H, E_H)$ on graafin $G(V_G, E_G)$ *aligraafi*, mikäli graafin H solmu- ja viivajoukot ovat graafin G solmu- ja viivajoukkojen osajoukko, eli $V_H \subseteq V_G$ ja $E_H \subseteq E_G$.

Esimerkki 2.2. *Kuvan 2.3 graafin* H

- *viivat* e_1 *ja* e_2 *ovat rinnakkaiset, koska* $e_1 = \{v_1, v_2\} = \{v_2, v_1\} = e_2$
- *yksi aligraafi on* $H_1(V_1, E_1)$*, missä* $V_1 = \{v_1, v_2\}$ *ja* $E_1 = \{e_1, e_2\}$.



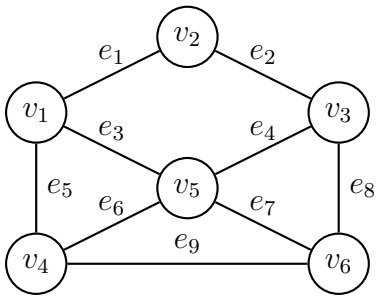
Kuva 2.3. *Graafi* $H(V, E)$.

2.2 Polut ja komponentit

Yleensä graafit muodostavat laajoja solmujen ja viivojen verkostoja. Tällöin pelkkä rinnakkaisten osien tulkinta ei riitä, vaan graafeissa on käytännöllistä muodostaa kulkuja, polkuja ja piirejä. Kulut, polut ja piirit merkitään solmujen ja viivojen jonoina.

Määritelmä 2.4. Kulut, polut ja piirit määritellään seuraavasti:

- Graafissa vierekkäin olevien solmujen ja viivojen jonoa kutsutaan *kuluksi*.



Kuva 2.4. *Graafi G.*

- *Polku* on kulku, jossa kukin solmu ja viiva esiintyvät vain ja ainoastaan kerran alku- ja loppusolmuja lukuunottamatta.
- *Piiri* on kulku, jonka alku- ja loppusolmu on sama ja muut solmut ja viivat esiintyvät kulussa vain ja ainoastaan kerran.

Kulku voi olla solmujen ja viivojen vuorotteleva jono tai vain viivoista muodostuva jono, sillä viivat sisältävät tiedon päätesolmuista [3]. Mikäli graafi on yksinkertainen, voi kulkua merkitä myös vain solmuista muodostuvana jonona, sillä solmuja yhdistävät viivat ovat aina yksiselitteisiä.

Esimerkki 2.3. *Muodostetaan*

(a) *polku graafin G solmusta v_1 solmuun v_6*

(b) *piiri, jonka alku- ja loppusolmu on graafin G solmu v_2,*

kun graafi on kuvan 2.4 mukainen graafi G.

(a) *Polku kahden solmun välillä voidaan usein muodostaa monella eri tavalla. Muodostetaan kuvan 2.4 avulla polku p, joka kulkee solmun v_5 kautta viivoja e_3 ja e_7 pitkin:*

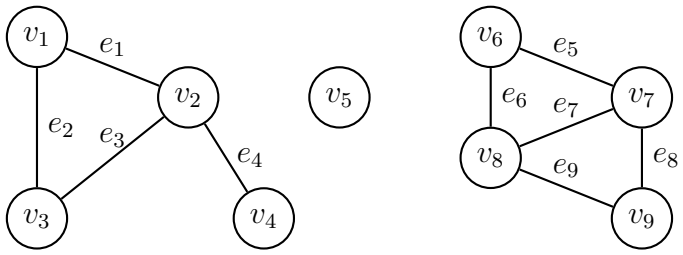
$$p = (v_1, e_3, v_5, e_7, v_6).$$

(b) *Myös piirin muodostamisessa on usein monia vaihtoehtoja. Muodostetaan piiri c, joka kulkee solmujen v_1, v_4, v_6 ja v_3 kautta:*

$$c = (v_2, e_1, v_1, e_5, v_4, e_9, v_6, e_8, v_3, e_2, v_2).$$

Polut ovat tärkeä työkalu graafien tulkitsemisessa. Esimerkiksi sukututkimuksessa kahta eri henkilöä yhdistävästä polusta saadaan selville, miten henkilöt ovat sukua toisilleen. Ford–Fulkerson-algoritmissa etsitään graafin saturoimattomia polkuja, jotka saturoimalla systeemin virtausta saadaan kasvatettua.

Polkuja on graafeista helppo etsiä ja muodostaa, kun voidaan tulkita graafin visuaalista esitystä. Matemaattisesti tai ohjelmallisesti niiden etsiminen on kuitenkin haastavampaa.



Kuva 2.5. *Graafi H.*

Polkujen etsimiseen syvennytään myöhemmin kappaleessa 3, kun käsitellään puita ja niiden kasvatusalgoritmeja. Puut ja kasvatusalgoritmit ovat oleellisiä työkaluja polkujen etsimisessä.

Määritelmä 2.5. Graafi on *yhtenäinen*, kun graafin mitkä tahansa kaksi eri solmua voidaan yhdistää polulla.

Toisin sanoen yhtenäisessä graafissa ei ole täysin toisistaan eristettyjä osia ja kaikki graafin osat ovat jollain tavalla yhdistettynä toisiinsa. Tämän kaltainen ominaisuus on helppo havaita graafin visuaalisen esityksen tulkinnalla. Mikäli graafi ei ole yhtenäinen, se voidaan jakaa *komponentteihin*. Myös graafin komponentit ovat helposti havaittavissa, kun tulkitaan graafin visuaalista esitystä.

Määritelmä 2.6. Graafin $G(V, E)$ yhtenäinen aligraafi $G_1(V_1, E_1)$ on graafin G *komponentti*, mikäli se ei yhdisty viivalla loppuun graafiin. Viivajoukkoon E_1 tulee kuulua myös kaikki ne viivajoukon E viivat, jotka yhdistävät solmujoukon V_1 solmuja.

Esimerkki 2.4. *Jaetaan kuvan 2.5 graafi H komponentteihin. Kuvasta 2.5 nähdään että graafi H ei ole yhtenäinen, sillä esimerkiksi solmuja v_4 ja v_8 ei voida yhdistää minkäänlaisella polulla. Nähdään myös, että solmuun v_5 ei ole yhdistynyt yhtäkään viivaa.*

Valitaan graafin H aligraafi $H_1(V_1, E_1)$ niin, että se sisältää vain solmun v_5. Aligraafi H_1 on yhtenäinen. *Solmu v_5 ei myöskään yhdisty viivalla loppuun graafiin, joten aligraafi H_1 on siis graafin G* komponentti.

Valitaan aligraafi $H_2(V_2, E_2)$ siten, että

$$V_2 = \{v_1, v_2, v_3, v_4\},$$
$$E_2 = \{e_1, e_2, e_3, e_4\}.$$

Aligraafi H_2 on yhtenäinen ja eristetty lopusta graafista, sillä solmujoukkojen V_2 ja $V \setminus V_2$ välillä ei ole yhtäkään viivaa. Kaikki solmujoukon V_2 solmuihin yhdistyvät viivat kuuluvat viivajoukkoon E_2. Näin ollen aligraafi H_2 on graafin H komponentti.

Graafin H kolmas komponentti $H_3(V_3, E_3)$ on jäljelle jäävä aligraafi, missä $V_3 = V \setminus (V_1 \cup V_2)$ ja $E_3 = E \setminus (E_1 \cup E_2)$.

Graafin H komponentit ovat

- $H_1(V_1, E_1)$, *missä*

$$V_1 = \{v_5\}$$
$$E_1 = \emptyset$$

- $H_2(V_2, E_2)$, *missä*

$$V_2 = \{v_1, v_2, v_3, v_4\}$$
$$E_2 = \{e_1, e_2, e_3, e_4\}$$

- $H_3(V_3, E_3)$, *missä*

$$V_3 = \{v_6, v_7, v_8, v_9\}$$
$$E_3 = \{e_5, e_6, e_7, e_8, e_9\}.$$

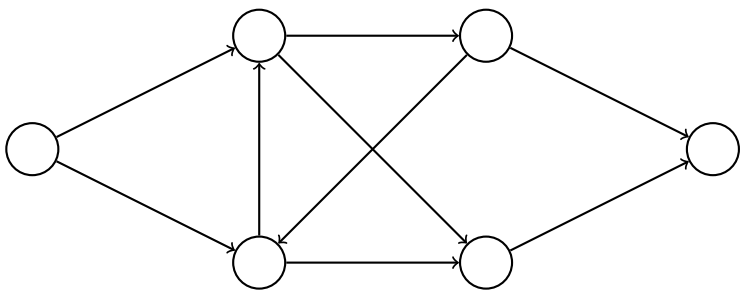
2.3 Suunnatut graafit eli digraafit

Graafiteoriassa pelkillä solmuilla ja viivoilla ei saada sisällytettyä graafiin tarpeeksi tietoa. *Diraafit* ovat graafeja, jotka muodostuvat viivojen sijasta *kaarista*. Digraafissa kaari on suunnattu viiva, jota kuvataan yleensä nuolella. Esimerkki digraafista on kuvattu kuvassa 2.6.

Määritelmä 2.7. Digraafi on järjestetty pari $G = (V, E)$, joka koostuu *solmujen* joukosta $V = \{v_1, v_2, \ldots, v_n\}$ ja niitä yhdistävien *kaarien* (eng. *arc*) joukosta A. Kaarijoukon A alkiot ovat joukon V *järjestettyjä* pareja, ja niitä merkitään (v_i, v_j), missä v_i ja v_j ovat joukon V alkioita.

Digraafi eroaa siis graafista vain siten, että digraafin kaaret ovat *järjestettyjä* solmupareja. Kun kaari lähtee solmusta x ja päättyy solmuun y, sanotaan, että kaari on suunnattu pois päin solmusta x ja kohti solmua y. Tällöin kaaren häntä on solmussa x ja pää solmussa

Kuva 2.6. *Digraafi.*



y. Digraafin kaaret ovat rinnakkaiset, mikäli ne lähtevät samasta solmusta ja päättyvät samaan solmuun.

Virtausongelmia tutkittaessa graafit ovat digraafeja, sillä virtauksella on suunta. Esimerkiksi tieliikennettä voidaan kuvata digraafeilla, joissa kaaret kuvaavat kaistoja. Nämä kaaret on suunnattu kaistojen ajosuunnan mukaisesti. Kaksisuuntaisia teitä kuvattaisiin vastakkaissuuntaisina kaarina.

2.4 Leikkaukset

Määritelmä 2.8. *Leikkaus* (eng. *edge cut*) on niiden viivojen joukko, jotka poistamalla yhtenäinen graafi jakautuu kahteen komponenttiin. Leikkausta, joka erottaa solmujoukon X lopusta graafista, merkitään $\partial(X)$.

Määritelmä 2.9. Digraafin leikkauksissa kaarien suunnilla ei ole väliä. Digraafin leikkaus $\partial(X)$ voidaan kuitenkin jakaa kahteen osajoukkoon kaarien suuntien perusteella. Osaleikkaus $\partial^+(X)$ on niiden kaarien joukko leikkauksessa $\partial(X)$, jotka on suunnattu pois päin solmujoukosta X, eli ne kaaret, joiden hännät ovat solmujoukossa X ja päät solmujoukossa $V \setminus X$. Osaleikkaus $\partial^-(X)$ on niiden kaarien joukko, jotka on suunnattu kohti solmujoukkoa X, eli ne kaaret, joiden päät ovat solmujoukossa X ja hännät solmujoukossa $V \setminus X$.

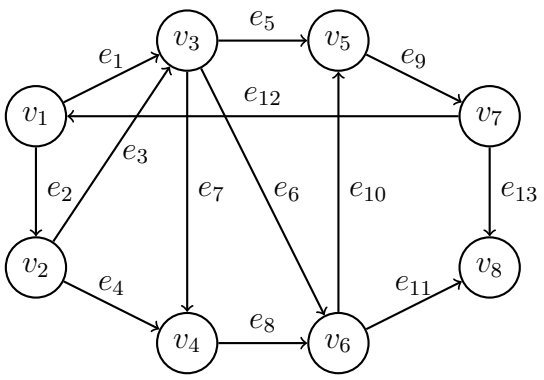
Esimerkki 2.5. *Selvitetään* ***a)*** $\partial(X)$ ***b)*** $\partial(Y)$ ***c)*** $\partial^+(X)$ ***d)*** $\partial^-(X)$ *kuvan 2.7 digraafista* D*, kun* $X = \{v_1, v_2, v_4\}$ *ja* $Y = \{v_3, v_5\}$.

a) *Leikkaus* $\partial(X)$ *on se kaarijoukko, jonka poistamalla digraafi* D *jakautuu komponentteihin, joiden solmujoukot ovat* X *ja* $V \setminus X$*. Leikkaus*

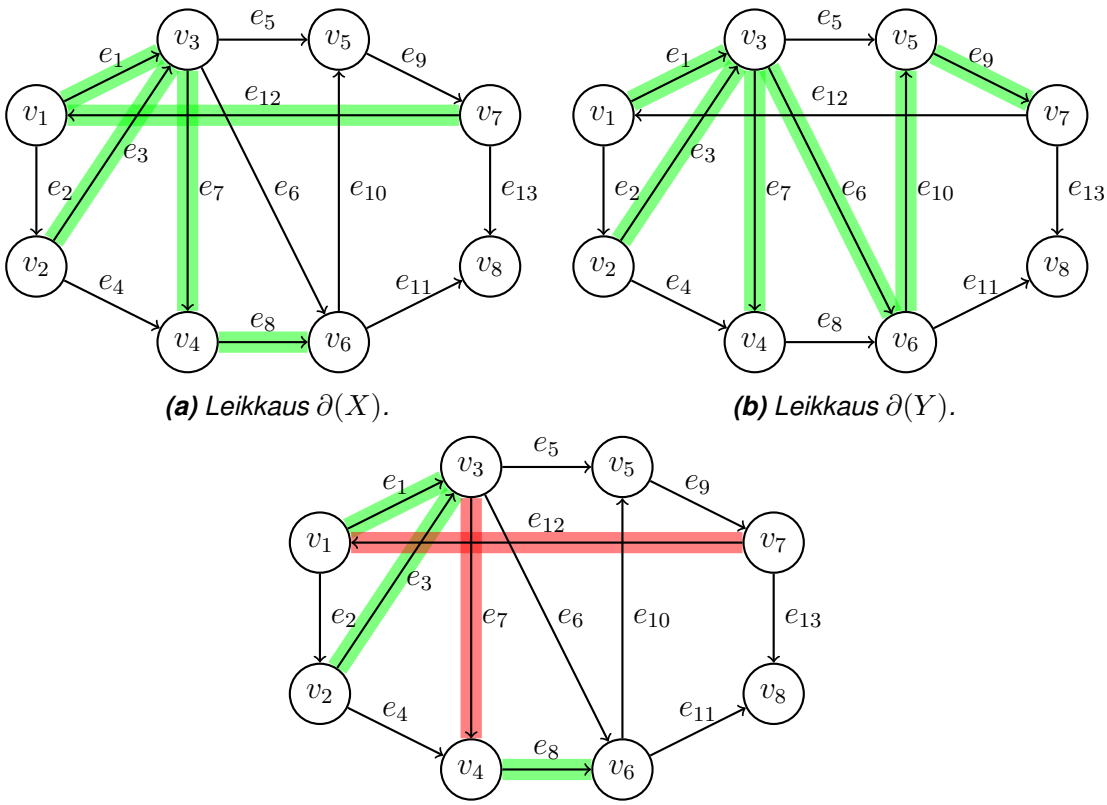
$$\partial(X) = \{e_1, e_3, e_7, e_8, e_{12}\}.$$

Leikkaus on kuvattu vihreällä kuvassa 2.8a.

b) *Poistamalla leikkaus* $\partial(Y)$ *digraafi* D *jakautuu komponentteihin, joiden solmujoukot*



Kuva 2.7. *Digraafi* $D(V, A)$.



(a) *Leikkaus* $\partial(X)$.

(b) *Leikkaus* $\partial(Y)$.

(c) *Osaleikkaukset* $\partial^+(X)$ *ja* $\partial^-(X)$.

Kuva 2.8. *Digraafin* $D(V, A)$ *leikkaukset.*

ovat Y *ja* $V \setminus Y$. *Leikkaus*

$$\partial(Y) = \{e_1, e_3, e_6, e_7, e_9, e_{10}\}.$$

Leikkaus on kuvattu kuvassa 2.8b.

c) *Osaleikkaus* $\partial^+(X)$ *on leikkauksen* $\partial(X)$ *ne kaaret, jotka ovat suunnattu pois päin solmujoukosta* X. *Osaleikkaus* $\partial^+(X) = \{e_1, e_3, e_8\}$ *on kuvattu kuvassa 2.8c vihreällä.*

d) *Osaleikkaus* $\partial^-(X)$ *on leikkauksen* $\partial(X)$ *solmujoukkoa* X *kohti suunnatut kaaret. Osaleikkaus* $\partial^-(X) = \{e_7, e_{12}\}$ *on kuvattu kuvassa 2.8c punaisella.*

3. PUUT JA PUIDEN KASVATUSALGORITMIT

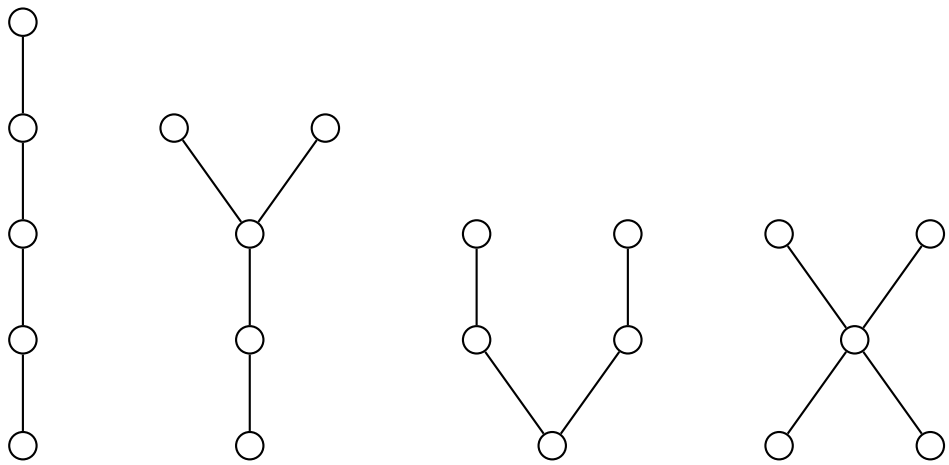
Määritelmä 3.1. *Puu* on yhtenäinen graafi, jonka mitä tahansa kahta solmua yhdistää vain ja ainoastaan yksi polku. Toisin sanoen puu on graafi, jossa ei ole piirejä.

Kaikki viisisolmuiset puut on kuvattu kuvassa 3.1. Polkua, joka yhdistää solmut x ja y puussa T merkitään polkuna xTy . Puiden kasvatuksessa on oleellista valita solmu, josta sitä kasvatetaan. Tätä solmua kutsutaan puun *juureksi*. Puuta, jonka juuri on solmu r kutsutaan r-puuksi [5]. Juurellisessa puussa vierekkäiset solmut ovat toistensa vanhempia tai lapsia. Jatkossa tässä työssä puulla tarkoitetaan aina juurellista puuta.

Puun T ollessa x-puu, solmu u on solmun v vanhempi, mikäli solmut ovat vierekkäiset ja $u \in xTv$. Solmu v on tällöin solmun u lapsi. Puun juuri x on puun ainoa solmu, jolla ei ole vanhempaa. Edeltäjäfunktio p (eng. *predecessor function*) palauttaa solmun vanhemman $p(v) = u$ ja on määritelty, kun $v \in V \setminus \{x\}$. [5]

Graafeja tulkittaessa visuaalisesti on helppo muodostaa haluttu polku puussa. Tämä polku on puun määritelmän mukaan *aina* yksiselitteinen. Mikäli puun polku halutaan muodostaa matemaattisesti tai ohjelmallisesti, on edeltäjäfunktio hyödyllinen, koska se palauttaa solmun vanhemman. Edeltäjäfunktio johtaa rekursiivisesti aina puun juureen, joten rekursiivisen edeltäjäfunktion avulla saadaan helposti muodostettua polku puun mistä tahansa pisteestä juureen.

Kun halutaan selvittää graafin ominaisuuksia, on puun kasvattaminen monesti hyödyl-



Kuva 3.1. *Puut, jotka koostuvat 5:stä solmusta.*

linen työkalu. Kasvatettava puu T on graafin G aligraafi. Kun puuta T aletaan kasvattamaan graafissa G, valitaan graafin G jokin solmu kasvatettavan puun juureksi. Puuta kasvatetaan lisäämällä puuhun graafin G solmuja, jotka ovat vierekkäisiä puun T solmujen kanssa, mutta eivät ole puussa T.

Puiden kasvatuksessa solmuja tarkastellaan tietyssä järjestyksessä, jonka määrää niiden läpikäymisjärjestys. Kasvatusalgoritmit eroavat toisistaan solmujen läpikäyntijärjestyksiltä, jotka voidaan jakaa kahteen pääasialliseen osa-alueeseen: leveys- ja syvyyssuuntaisiin läpikäynteihin. Leveyssuuntaisissa läpikäynneissä puuta kasvatetaan mahdollisimman läheltä juurta, kun taas syvyyssuuntaisissa läpikäynneissä kasvatus tapahtuu mahdollisimman kaukana juuresta. Leveyssuuntaisia läpikäyntejä hyödynnetään esimerkiksi silloin kun etsitään lyhintä polkua solmujen välillä. Syvyyssuuntaisia läpikäyntejä puolestaan käytetään alialgoritmeissa ja suurissa graafeissa niiden pienen muistijalanjäljen takia.

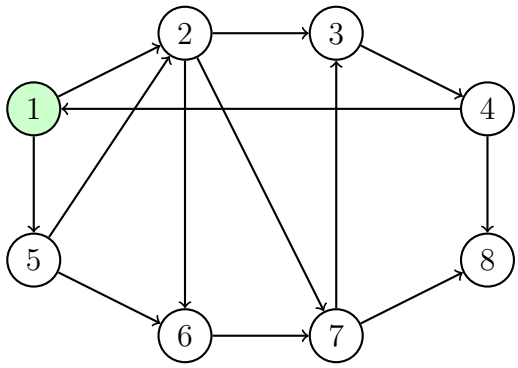
Kasvatusalgoritmin käyttökohde määrää ehdot solmujen lisäämiselle puuhun. Mikäli graafin viivoilla tai kaarilla on myös ominaisuuksia, voidaan niitä sisällyttää solmujen lisäämisehtoihin. Joissakin tapauksissa, kuten Ford–Fulkerson-algoritmissa, kaikki puun kasvatuksen ehdot liittyvät solmuja yhdistävien kaarien ominaisuuksiin.

Tässä työssä esitellään tiivistetysti perustapaukset molemmista läpikäyntijärjestyksistä, sillä Ford–Fulkerson-algoritmissa voidaan käyttää mitä tahansa läpikäymisjärjestystä. Tässä työssä virtausongelmien esimerkeissä käytetään leveyssuuntaista läpikäyntiä. Puiden kasvatusalgoritmeihin voi tutustua tarkemmin Bundyn ja Murtyn kirjan luvussa 6 [5].

3.1 Leveyssuuntainen läpikäynti

Leveyssuuntaisessa läpikäynnissä solmuja, joista puuta mahdollisesti voidaan kasvattaa, kerätään jonoon. Tätä jonoa voidaan kutsua *läpikäyntijonoksi*. Läpikäyntijonon ensimmäinen solmu otetaan *käsiteltäväksi*. Kun solmua käsitellään, tarkastellaan solmun jokaista viereistä solmua, joka ei kuulu kasvatettavaan puuhun. Solmua *tarkasteltaessa* katsotaan täyttääkö solmu annetut ehdot. Jos solmu täyttää ehdot, se lisätään puuhun ja läpikäyntijonon perälle. Solmu *poistetaan* läpikäyntijonosta kun se on käyty läpi, ja jonon seuraava solmu otetaan läpikäytäväksi. [5, s. 137]

Esimerkki 3.1. *Kasvatetaan kuvan 3.2 digraafin solmusta 1 (merkitty vihreällä) puu leveysetsinnällä. Ehto: puuhun lisättävien kaarien tulee olla suunnattuna pois päin juuresta.*

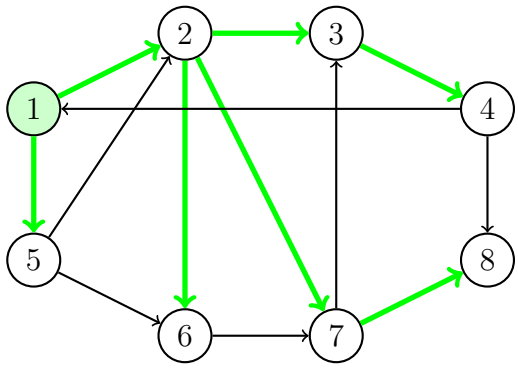


Kuva 3.2. *Digraafi D.*

Leveyssuuntaisen läpikäynnin läpikäyntijonon kehitysvaiheet:

$$\begin{aligned}
&\emptyset \to 1 \to 1\,2 \to 1\,2\,5 \to 2\,5 \to 2\,5\,3 \to 2\,5\,3\,6 \\
&\quad \to 2\,5\,3\,6\,7 \to 5\,3\,6\,7 \to 3\,6\,7 \to 3\,6\,7\,4 \to 6\,7\,4 \\
&\quad \to 7\,4 \to 7\,4\,8 \to 4\,8 \to 8 \to \emptyset.
\end{aligned}$$

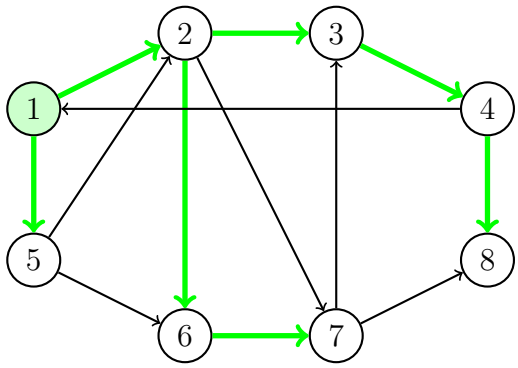
Leveyssuuntaisella läpikäynnillä kasvatettu puu on kuvattu kuvassa 3.3.



Kuva 3.3. *Digraafin D yksi leveyssetsinnällä löydetty puu.*

3.2 Syvyyssuuntainen läpikäynti

Syvyyssuuntaisessa läpikäynnissä solmuja, joista puuta voi olla mahdollista kasvattaa kerätään pinoon. Tätä pinoa kutsutaan *läpikäyntipinoksi*. Läpikäyntipinon päällimmäinen solmu otetaan *käsittelyyn*. Käsittelyssä olevan solmun yksi viereinen puuhun kuulumaton solmu otetaan *tarkasteltavaksi*. Käsittelyssä oleva solmu *poistetaan* läpikäyntipinosta, mikäli kaikkia solmun viereisiä solmuja on tarkasteltu. Solmua tarkasteltaessa katsotaan, täyttääkö solmu annetut ehdot. Jos solmu täyttää ehdot, niin solmu lisätään puuhun ja läpikäyntipinon päällimmäiseksi. Näin juuri puuhun lisätty solmu on pinon päälimmäisenä ja otetaan seuraavaksi käsittelyyn. [5, s. 139]



Kuva 3.4. *Digraafin D yksi syvyysetsinnällä löydetty puu.*

Esimerkki 3.2. *Kasvatetaan kuvan 3.2 digraafin solmusta 1 (merkitty vihreällä) puu syvyysetsinnällä. Ehto: puuhun lisättävät kaaret tulevat olla suunnattuna pois päin juuresta.*

Syvyyssuuntaisen läpikäynnin läpikäyntipinon kehitysvaiheet:

$$\begin{aligned}
&\emptyset \to 1 \to 12 \to 123 \to 1234 \to 12348 \\
&\quad \to 1234 \to 123 \to 12 \to 126 \to 1267 \\
&\quad \to 126 \to 12 \to 1 \to 15 \to 1 \to \emptyset.
\end{aligned}$$

Leveyssuuntaisella läpikäynnillä kasvatettu puu on kuvattu kuvassa 3.4.

4. VIRTAUSSYSTEEMIT

Suunnattuja graafeja voidaan soveltaa virtauksien tutkimiseen. Jotta näin voidaan tehdä, tulee määritellä *virtaussysteemi* (eng. *network*). Virtaussysteemi on suunnattu graafi, joka sisältää virtauksille oleellista tietoa, kuten mistä ja mihin virtaus tapahtuu sekä kuinka paljon virtausta kulkee ja voi kulkea missäkin kohdassa systeemiä. Tässä työssä virtaussysteemit oletetaan myös *aina* yksinkertaisiksi, sillä yksinkertainen graafi voidaan muodostaa helposti yhdistämällä rinnakkaiset kaaret ja poistamalla silmukat. Tämä luku mukailee pitkälti Bundyn ja Murtyn kirjan lukua 7.1 [5].

Määritelmä 4.1. *Virtaussysteemi* $N(x, y)$ on suunnattu graafi, jossa solmu x toimii virtauksen *lähteenä* ja solmu y *nieluna*. Yksikään virtaussysteemin kaarista ei pääty lähteeseen x tai lähde nielusta y. Virtaussysteemin kaikki muut solmut ovat välittäjäsolmuja. Tälle solmujoukolle käytetään merkintää I. Systeemin virtaus syntyy lähteestä x ja matkaa välittäjäsolmuja pitkin nieluun y. Virtaussysteemin kaarijoukon A jokaisella kaarella on *kapasiteetti* $c(a) \in \mathbb{N}$, joka määrittää kaaren läpi kulkevan maksimivirtauksen.

Tässä työssä käsitellään virtaussysteemeitä, joilla on ainoastaan yksi lähde ja nielu. Virtaussysteemi, jolla on useampi lähde voidaan helposti yksinkertaistaa yksilähteiseksi. Tämä tapahtuu siten, että lisätään uusi lähde, muutetaan vanhat lähteet välittäjäsolmuiksi ja yhdistetään ne kaarilla uuteen lähteeseen. Samaa lähestymistapaa voidaan käyttää nieluihin.[6, s. 15]

Määritelmä 4.2. *Virtaus* on virtaussysteemin N kaarissa määritelty funktio f, joka kuvaa niissä kulkevaa virtausta. Virtaussysteemissä solmuun v saapuvasta virtauksesta käytetään merkintää $f^+(v)$ ja solmusta v lähtevästä virtauksesta käytetään merkintää $f^-(v)$. Kaikille solmuille $v \in I$ pätee yhtäsuuruus $f^-(v) = f^+(v)$.

Virtaussysteemin $N(x, y)$ kolme solmutyyppiä ovat:

1. *Lähde* x luo systeemin virtauksen. Se on virtaussysteemin ainoa solmu, jonka nettovirtaus poispäin itsestään on positiivinen, eli

$$f^+(x) - f^-(x) \geq 0.$$

2. *Nielu* y vastaanottaa kaiken systeemin virtauksen. Se on systeemin ainoa solmu,

jonka nettovirtaus itseensä on positiivinen, eli

$$f^-(y) - f^+(y) \geq 0.$$

3. Virtaussysteemin loput solmut ovat *välittäjäsolmuja*. Välittäjäsolmuissa pätee Kirchhoffin virtauslaki, jonka mukaan välittäjäsolmujen nettovirtaus on *aina* nolla. Toisin sanoen

$$f^-(v) = f^+(v), \quad \text{kaikilla } v \in I. \tag{4.1}$$

Virtaussysteemissä lähde on siis ainoa solmu, joka luo systeemiin virtausta. Nielu on systeemin ainoa solmu, joka syö virtausta. Välittäjäsolmut eivät vaikuta systeemin kokonaisvirtaukseen, vaan nimensä mukaisesti välittävät virtausta eteenpäin lähteestä kohti nielua. Kaikki virtaus alkaa lähteestä ja päätyy nieluun. Esimerkiksi tuotteen kuljetusketjussa tehdas voidaan mieltää systeemin lähteeksi, kauppa nieluksi, varastot välittäjäsolmuiksi ja rahtiliikenne virtaussysteemin kaariksi.

Määritelmä 4.3. Virtaussysteemin kokonaisvirtaus $|x|$ on nettovirtaus lähteestä x, eli

$$|x| = f^+(x) - f^-(x).$$

Virtausta voidaan tarkastella erikseen yksittäisissä kaarissa tai kaarien joukoissa. Solmujen ja solmujoukkojen virtauksen tarkastelu tapahtuu tarkastelemalla niiden leikkauksien virtausta. Kaikki virtaussysteemien $N(x, y)$ leikkaukset oletetaan tästä eteenpäin leikkauksiksi, jotka erottavat virtaussysteemin lähteen ja nielun toisistaan, siten että osaleikkaus

$$K = \partial^+(X),$$

missä lähde x kuuluu solmujoukkoon X ja nielu y solmujoukkoon $V \setminus X$.

Määritelmä 4.4. Kaarijoukon $S \subseteq A$ virtaus lasketaan joukon kaarien virtauksien summana

$$f(S) = \sum_{a \in S} f(a).$$

Solmujoukon $X \subseteq V$ virtaus määritellään sen osaleikkauksien $\partial^+(X)$ ja $\partial^-(X)$ avulla asettamalla

$$f^+(X) = f(\partial^+(X)),$$
$$f^-(X) = f(\partial^-(X)).$$

Edellisen määritelmän mukaan solmujoukoista *lähtevää virtausta* kuvataan positiivisella

yläindeksillä $f^+(X)$ ja *saapuvaa virtausta* negatiivisella yläindeksillä $f^-(X)$. Virtaussysteemin kokonaisvirtauksen arvoa merkitään $|f|$.

Määritelmä 4.5. Nettovirtausta $f^+(X) - f^-(X)$ kutsutaan nettovirtaukseksi solmujoukosta X ja nettovirtausta $f^-(X) - f^+(X)$ nettovirtaukseksi solmujoukkoon X.

Lause 4.1. *Olkoon virtaussysteemin solmujoukot $X \subset V$ ja $Y = V \setminus X$. Nettovirtaus solmujoukosta X on yhtä suuri kuin nettovirtaus solmujoukkoon Y.*

Todistus. Nettovirtaus joukosta X on

$$f^+(X) - f^-(X) = f(\partial^+(X)) - f(\partial^-(X)),$$

ja nettovirtaus joukkoon Y on

$$f^-(Y) - f^+(Y) = f(\partial^-(Y)) - f(\partial^+(Y)).$$

Koska $Y = V \setminus X$, niin

$$\partial^-(Y) = \partial^+(X) \quad \text{ja} \quad \partial^+(Y) = \partial^-(X).$$

Täten voidaan kirjoittaa

$$\begin{aligned} f^-(Y) - f^+(Y) &= f(\partial^-(Y)) - f(\partial^+(Y)) = f(\partial^+(X)) - f(\partial^-(X)) \\ &= f^+(X) - f^-(X). \end{aligned}$$

□

Lause 4.2. *Virtaussysteemissä $N(x, y)$ nettovirtaus lähteestä x on yhtä suuri kuin nettovirtaus systeemin nieluun y.*

Todistus. Valitaan solmujoukot X ja Y siten, että $x \in X$ ja $y \in Y = V \setminus X$. Koska Kirchoffin virtalain (4.1) mukaan virtaussysteemissä välittäjäsolmujen nettovirtaukset ovat nolla, voidaan solmujoukkojen X ja Y nettovirtaukset laskea helposti. Nettovirtaus solmujoukosta X on siis

$$\begin{aligned} f^+(X) - f^-(X) &= \sum_{v \in X} f^+(v) - f^-(v) \\ &= \left[\sum_{v \in X \setminus \{x\}} f^+(v) - f^-(v) \right] + f^+(x) - f^-(x) \\ &= f^+(x) - f^-(x). \end{aligned}$$

Samoin nettovirtaus solmujoukkoon Y on

$$\begin{aligned} f^-(Y) - f^+(Y) &= \sum_{v \in Y} f^-(v) - f^+(v) \\ &= \left[\sum_{v \in Y \setminus \{y\}} f^-(v) - f^+(v) \right] + f^-(y) - f^+(y) \\ &= f^-(y) - f^+(y). \end{aligned}$$

Nyt lauseen 4.1 avulla voidaan kirjoittaa

$$\begin{aligned} f^+(x) - f^-(x) = f^+(X) - f^-(X) &= f^-(Y) - f^+(Y) \\ &= f^-(y) - f^+(y). \end{aligned}$$

□

Lause 4.3. *Jos virtaussysteemin lähde x kuuluu solmujoukkoon $X \subset V$ ja nielu y kuuluu solmujoukkoon $Y = V \setminus X$, nettovirtaus solmujoukosta X on systeemin kokonaisvirtaus $|f|$.[5, s. 159]*

Todistus. Kirchoffin virtauslain (4.1) mukaan välittäjäsolmuilla ei ole nettovirtausta ja määritelmän 4.3 nojalla nettovirtaus lähteestä on yhtä suuri kuin systeemin kokonaisvirtaus, eli

$$\begin{aligned} f^+(v) - f^-(v) &= 0, \quad \text{kun } v \in X \setminus \{x\} \\ f^+(x) - f^-(x) &= |f|. \end{aligned}$$

Nyt nettovirtaus solmujoukosta X voidaan laskea

$$\begin{aligned} f^+(X) - f^-(X) &= \sum_{v \in X} (f^+(v) - f^-(v)) \\ &= \sum_{v \in X \setminus \{x\}} (f^+(v) - f^-(v)) + f^+(x) - f^-(x) \\ &= |f|. \end{aligned}$$

□

Kaaren *kapasiteetti* määrittää suurimman mahdollisen kaaressa kulkevan virtausmäärän. Virtaussysteemin kaarien kapasiteetit vaikuttavat systeemin maksimivirtaukseen.

Määritelmä 4.6. Virtaus kaaressa a on *mahdollinen* jos

$$0 \leq f(a) \leq c(a).$$

Tämän rajoituksen yläraja kuvaa materiaalivirtauksen maksimimäärää esimerkiksi putken läpi. Rajoituksen alaraja on nolla, koska negatiivinen virtaus kulkisi kaaressa vastakkaiseen suuntaan. Tätä tilannetta kuvataan vastakkaiseen suuntaan suunnatulla kaarella, jossa kulkeva virtaus olisi tällöin positiivinen. Kaaressa kulkevan virtauksen perusteella kaari on:

- *virtaukseton* jos $f(a) = 0$
- *virtaava* jos $f(a) > 0$
- *saturoimaton* jos $f(a) < c(a)$
- *saturoitu* jos $f(a) = c(a)$.

Kaarijoukon K kapasiteetti on yksittäisten kaarien kapasiteettien summa

$$\text{cap}(K) = \sum_{a \in K} c(a).$$

Virtaussysteemin osaleikkauksien kapasiteetit ovat

$$\text{cap}(\partial^+(X)) = \sum_{a \in \partial^+(X)} c(a) \quad \text{ja} \quad \text{cap}(\partial^-(X)) = \sum_{a \in \partial^-(X)} c(a).$$

Mikään virtaussysteemin kokonaisvirtauksen arvo ei voi ylittää minkään $\partial^+(X)$ osaleikkauksen kapasiteettia. Mikäli systeemin kokonaisvirtaus olisi suurempi kuin jokin systeemin osaleikkauksen $\partial^+(X)$ kapasiteetti, virtaus rikkoisi kapasiteettisääntöä eikä näin olisi mahdollinen. Mille tahansa virtaussysteemin virtaukselle ja osaleikkaukselle siis pätee $|f| \leq \text{cap}(\partial^+(X))$.

Lause 4.4. *Mille tahansa systeemin $N(x, y)$ virtaukselle f ja osaleikkaukselle $K = \partial^+(X)$ pätee [5, s. 160]*

$$|f| \leq \textit{cap}(K).$$

Todistus. Määritelmien 4.4 ja 4.6 avulla voidaan kirjoittaa:

$$f^+(X) \leq \text{cap}(K) \tag{4.2}$$

$$f^-(X) \geq 0. \tag{4.3}$$

Lauseen 4.3 perusteella voidaan kirjoittaa

$$|f| = f^+(X) - f^-(X) \leq \text{cap}(K).$$

□

Korollaari 4.5. *Jokainen osaleikkauksen $K = \partial^+(X)$ kaari on* saturoitu *ja jokainen osaleikkauksen $\partial^-(X)$ kaari on* virtaukseton *jos ja vain jos [5, s. 160]*

$$|f| = \textit{cap}(K).$$

Todistus. Epäyhtälö (4.2) voidaan kirjoittaa yhtälönä jos ja vain jos kaikki osaleikkauksen K kaaret on saturoitu. Samoin epäyhtälö (4.3) voidaan kirjoittaa yhtälönä jos ja vain jos kaikki osaleikkauksen $\partial^-(X)$ kaaret ovat virtauksettomia. Tällöin pätee

$$|f| = \text{cap}(K).$$

□

Systeemin maksimivirtaus on virtaussysteemin kapasiteettirajoituksien rajaama, suurin mahdollinen kokonaisvirtaus. Systeemin maksimivirtauksen selvittämisessä oleellinen työkalu on minimileikkaus. Minimileikkaus $K = \partial^+(X)$, missä $x \in X$ on virtaussysteemin $N(x, y)$ se leikkaus, jonka kapasiteetti cap(K) on pienin. Minimileikkaus voidaan mieltää systeemin virtauksen "pullonkaulaksi" eli leikkaukseksi, joka on suurin virtausta rajoittava tekijä. Minimileikkauksen ja maksimivirtauksen yhteys on varsin intuitiivinen. Systeemin kaikki virtaus kulkee minimileikkauksen läpi, joten systeemin maksimivirtaus ei voi ylittää minimileikkauksen kapasiteettia.

Lause 4.6. *Jos on olemassa virtaus f ja leikkaus K, ja*

$$|f| = \textit{cap}(K),$$

niin $|f|$ on systeemin maksimivirtaus ja K systeemin minimileikkaus. [5, s. 160]

Todistus. Merkitään maksimivirtausta $|f_{max}|$ ja minimileikkausta K_{min}. Virtaussysteemin mikä tahansa mahdollinen virtaus on aina pienempi tai yhtäsuuri kuin systeemin maksimivirtaus. Samoin mikä tahansa systeemin leikkauksen kapasiteetti on suurempi tai yhtä suuri kuin systeemin minimileikkaus. Systeemin maksimivirtaus on pienempi tai yhtä suuri kuin systeemin minimileikkaus. Voidaan siis kirjoittaa epäyhtälö

$$|f| \leq |f_{max}| \leq \text{cap}(K_{min}) \leq \text{cap}(K).$$

Koska $|f| = \text{cap}(K)$, epäyhtälö voidaan kirjoittaa yhtälöksi ja kirjoittaa

$$|f| = |f_{max}| = \text{cap}(K) = \text{cap}(K_{min}).$$

□

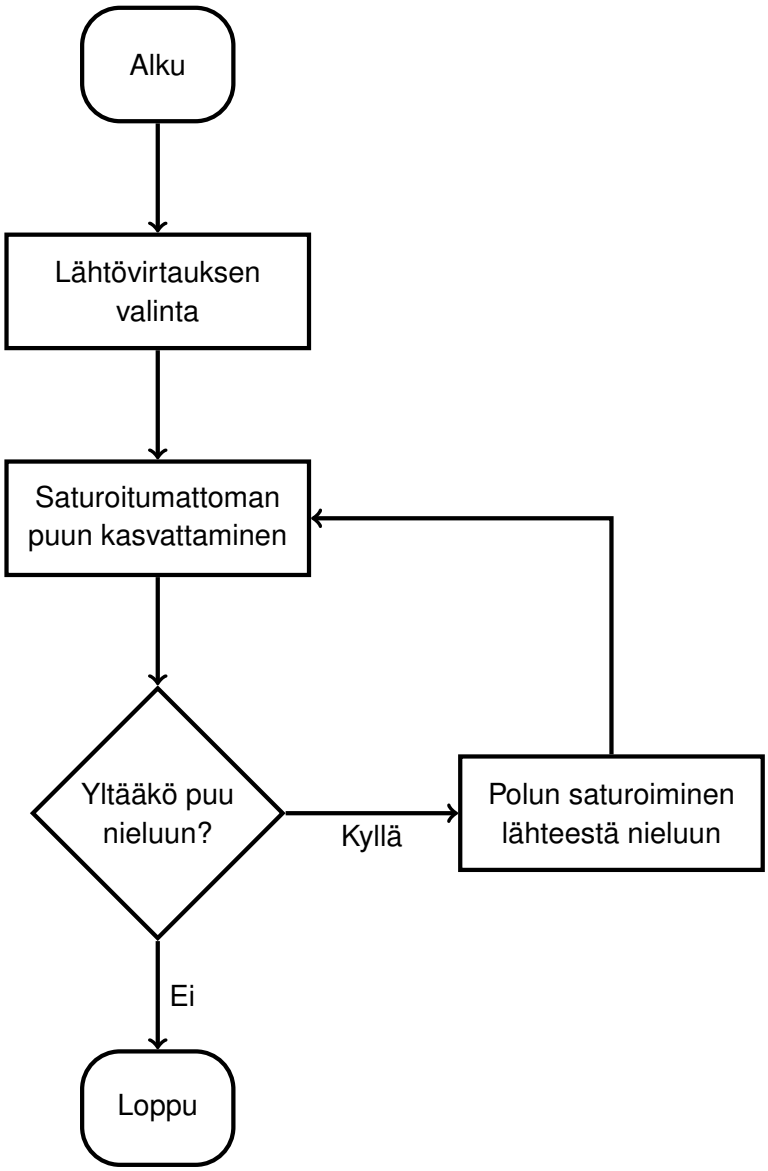
5. FORD–FULKERSON-ALGORITMI

Ford–Fulkerson-algoritmi on tehokas tapa selvittää virtausgraafin maksimivirtaus. Algoritmi toimii yksinkertaisuudessaan siten, että se etsii ja saturoi systeemissä polkuja lähteestä nieluun niin kauan, kun vajaita polkuja on olemassa. Algoritmi päättyy, kun kaikki systeemin polut lähteestä nieluun on saturoitu. Tällöin systeemin kokonaisvirtaus systeemin maksimivirtaus. Tässä työssä esitelty Ford–Fulkerson-algoritmi mukailee Bundyn ja Murtyn kirjan algoritmia [5, s. 163]. Ford–Fulkerson-algoritmin kulku on kuvattu kaaviossa 5.1.

Ford–Fulkerson-algoritmin avulla löydetään myös systeemin minimileikkaus, jonka voi kuvitella virtaussysteemin "pullonkaulaksi". Mikäli minimileikkaukseen sisältyvien kaarien kapasiteettia kasvatetaan, saadaan systeemin maksimivirtaustakin kasvatettua. Lisäksi maksimivirtausta voidaan kasvattaa muuttamalla systeemin minimileikkausta esimerkiksi lisäämällä systeemiin kaaria. Ford–Fulkerson-algoritmin vaiheet ovat:

- **Lähtövirtauksen valinta.** Ford–Fulkerson-algoritmin alussa valitaan systeemin lähtövirtaus. Yleensä lähtövirtaukseksi asetetaan nollavirtaus ($|f| = 0$) eli systeemin tilanne, jossa virtausta ei ole.
- **Saturoitumattoman puun kasvattaminen.** Systeemin lähteestä kasvatetaan puu seuraavia ehtoja noudattaen:
  (a) Puuta X kasvatetaan oksien päistä pois päin suuntautuvilla kaarilla $a = (u, v) : u \in X$ ja $v \in V \setminus X$, joiden virtaus on pienempi kuin niiden kapastiteetti, eli $f(a) < c(a)$. Nämä ovat kaaria, joiden virtauksen kasvattaminen voi johtaa systeemin kokonaisvirtauksen kasvamiseen.
  (b) Puuta kasvatetaan myös oksien päihin suuntautuvilla kaarilla $a = (u, v) : u \notin X$ ja $v \in X$, joissa on virtausta eli $f(a) > 0$. Nämä ovat kaaria, joiden virtausta pienentämällä voidaan ohjata kaareen tullut virtaus kohti nielua toista kaarta pitkin. Virtauksen reitin muuttaminen voi antaa tilaa suuremmalle kokonaisvirtaukselle.

  Puuta kasvatetaan kunnes se saavutta nielun tai kunnes puuta ei enää voi edellisten sääntöjen mukaan kasvattaa. Huomaa, että se järjestys missä puuta kasvatetaan vaikuttaa puun muodostumiseen.



Kuva 5.1. *Vuokaavio algoritmin kulusta.*

- **Yltääkö puu nieluun?** Mikäli kasvatettu puu yltää nieluun asti, voidaan systeemin virtausta lisätä. Jos puu ei yllä nieluun asti, on systeemin virtaus maksimoitu ja algoritmi päättyy.
- **Virtausta kasvattavan polun saturoiminen.** Tässä vaiheessa otetaan käsittelyyn se edellisessä vaiheessa kasvatetun puun kaarijoukko, joka ylettyy lähteestä nieluun asti. Tämä kaarijoukko muodostaa polun P, joka saadaan muodostettua funktion p avulla. Polku P on saturoimaton, eli polun virtausta voidaan kasvattaa, jolloin systeemin kokonaisvirtaus kasvaa. Systeemin maksimivirtauksen saavuttamiseksi polku täytetään virtauksella eli saturoidaan. Määrää, jolla polun virtausta voidaan kasvattaa kutsutaan polun maksimikasvatusarvoksi $\epsilon(P)$, ja se on määritelty seuraavasti:

Määritelmä 5.1. Polun maksimikasvatusarvo

$$\epsilon(P) = \min\{\epsilon(a) : a \in A(P)\},$$

jossa

$$\epsilon(a) = \begin{cases} c(a) - f(a), & \text{mikäli } a \text{ on suunnattu kohti nielua polussa } P \\ f(a), & \text{mikäli } a \text{ on suunnattu kohti lähdettä polussa } P. \end{cases}$$

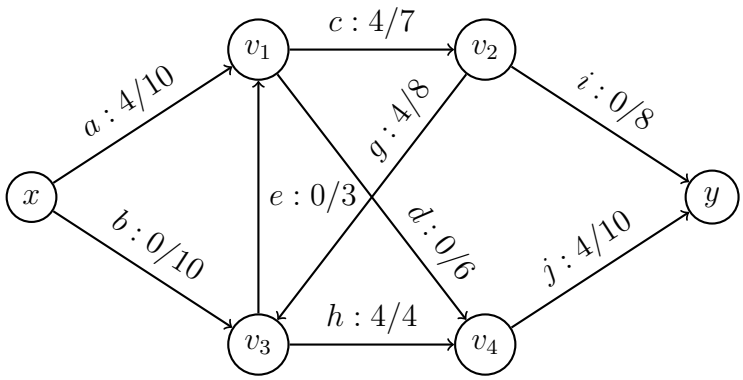
Koska polun P kaikki nielua kohti suunnatut kaaret ovat saturoimattomia ja lähdettä kohti suunnatut kaaret ovat virtaavia, niin $\epsilon(a) > 0$, eli $\epsilon(P) > 0$. Polku saadaan saturoitua kasvattamalla tai vähentämällä polun jokaisen kaaren virtausta polun maksimikasvatusarvon verran, jolloin saadaan kaaren uusi virtaus

$$f'(a) = \begin{cases} f(a) + \epsilon(P), & \text{mikäli } a \text{ on suunnattu kohti nielua polussa } P \\ f(a) - \epsilon(P), & \text{mikäli } a \text{ on suunnattu kohti lähdettä polussa } P. \end{cases}$$

Algoritmin Päättyminen Kun systeemin virtausta on kasvatettu niin paljon, että viimeisin kasvatettu saturoitumaton puu ei yllä enää nieluun asti, on systeemin maksimivirtaus saavutettu. Systeemin maksimivirtaus on nyt nieluun päätyvä kokonaisvirtaus.

Systeemin minimileikkaus saadaan selville siitä algoritmin kasvattamasta saturoitumattomasta puusta, joka ei yllä nieluun asti. Tätä puuta kutsutaan IPS-puuksi (*Incrementing Path Search*), ja systeemin minimileikkaus on se joukko kaaria, joiden hännät koskettavat IPS-puuta. [5] [4]

Esimerkki 5.1. *Selvitetään kuvan 5.2 virtaussysteemin $N(x, y)$ maksimivirtaus ja minimileikkaus Ford–Fulkerson-algoritmin avulla. Kuvassa 5.2 kaariin on merkitty niiden ka-*



Kuva 5.2. *Systeemi $N(x, y)$.*

pasiteetti ja niissä kulkeva virtaus. Esimerkiksi kaaren a virtaus $f(a) = 4$ ja kapasiteetti $c(a) = 10$.

- ***Lähtövirtauksen valinta***
  Valitaan systeemin lähtövirtaukseksi kuvan 5.2 mukainen kokonaisvirtaus $|f| = 4$.
- ***Saturoitumattoman puun kasvattaminen***
  Valitaan $X_1 = \{x\}$, eli aletaan kasvattamaan puuta X_1 lähteestä x.
  - *Otetaan käsiteltäväksi solmu x, ja tarkastellaan solmua v_1 ja kaarta $a = (x, v_1)$. Kaaren a häntä x kuuluu puuhun X_1 eikä kaaren pää kuulu puuhun X_1. Kaari a on myös saturoimaton, sillä $f(a) < c(a)$. Ehdot täyttyvät, eli solmu v_1 voidaan lisätä puuhun X_1. Kaari a lisätään puuhun merkitsemällä solmun v_1 vanhemmaksi solmu x. Solmun ja kaaren lisääminen puuhun tapahtuu seuraavasti:*

$$X_1 = X_1 \cup \{v_1\},$$
$$p(v_1) = x.$$

  - *Tarkastellaan solmua v_3 ja kaarta $b = (x, v_3)$. Ehdot täyttyvät kuten kaarella a, joten kaari b ja solmu v_3 voidaan lisätä puuhun seuraavasti:*

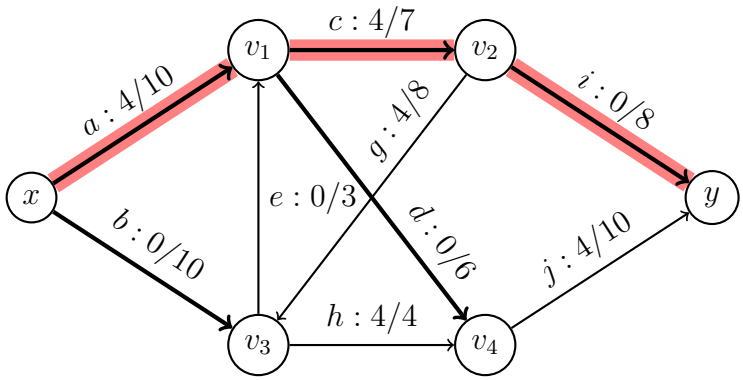
$$X_1 = X_1 \cup \{v_3\},$$
$$p(v_3) = x.$$

  - *Otetaan käsittelyyn solmu v_1, ja tarkastellaan kaaria $c = (v_1, v_2)$ ja $d = (v_1, v_4)$. Kaarta $e = (v_3, v_1)$ ei tarkastella, sillä sekä kaaren pää että häntä kuuluvat jo puuhun. Kaaret c ja d voidaan lisätä puuhun X_1 seuraavasti:*

$$X_1 = X_1 \cup \{v_2, v_4\},$$
$$p(v_2) = v_1,$$
$$p(v_4) = v_1.$$

  - *Otetaan käsittelyyn solmu v_3. Kaaria $e = (v_3, v_1)$, $g = (v_2, v_3)$ ja $h = (v_3, v_4)$ ei tarkastella, sillä kaarien hännät ja päät kuuluvat jo puuhun*
  - *Otetaan käsittelyyn solmu v_2, ja valitaan kaari $i = (v_2, y)$. Kaarta $g = (v_2, v_3)$ ei tarkastella. Ehdot täyttyvät, joten kaari i lisätään puuhun X_1 asettamalla*

$$X_1 = X_1 \cup \{y\},$$
$$p(y) = v_2.$$



Kuva 5.3. *Ensimmäinen saturoitumaton puu X_1 ja polku P_1.*

Puuta X_1 on nyt kasvatettu niin, että se yltää lähteestä x nieluun y. Näin ollen siirrytään algoritmin seuraavaan vaiheeseen, jossa muodostetaan polku lähteestä nieluun ja saturoidaan se. Puu X_1 kattaa systeemin N kaikki solmut $X_1 = \{x, v_1, v_2, v_3, v_4, y\}$, ja se on kuvattu kuvassa 5.3 vahvistetuilla nuolilla.

- ***Virtausta kasvattavan polun saturoiminen***
  Funktion p avulla saadaan muodostettua saturoitumaton polku $P_1 = \{x, v_1, v_2, y\}$. Polku P_1 on kuvattu kuvassa 5.3 punaisella.

  Systeemin kokonaisvirtausta kasvatetaan saturoimalla polku P_1. Polun virtausta voidaan kasvattaa enintään polun yksittäisten kaarien pienimmän maksimikasvatusarvon verran.

  Lasketaan polun P_1 kaarien $A(P_1) = \{a, c, i\}$ maksimikasvatusarvot. Kaikki kaaret on suunnattu kohti nielua. Maksimikasvatusarvot ovat

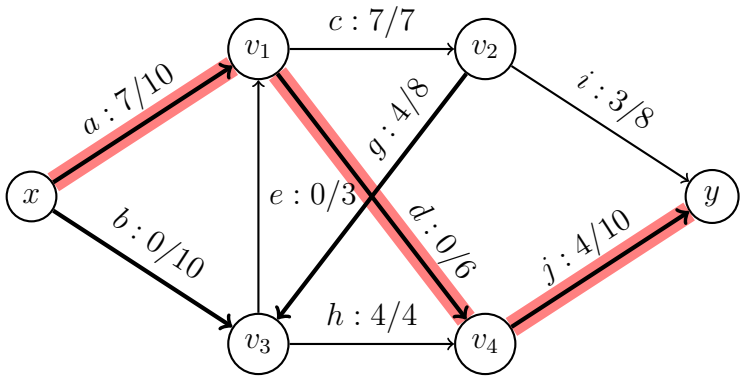
$$\epsilon(a) = c(a) - f(a) = 10 - 4 = 6$$
$$\epsilon(c) = c(c) - f(c) = 7 - 4 = 3$$
$$\epsilon(i) = c(i) - f(i) = 8 - 0 = 8.$$

  Polun P_1 maksimikasvatusarvo on siis

$$\epsilon(P_1) = min\{\epsilon(a) : a \in A(P_1)\} = min\{6, 3, 8\} = 3.$$

  Lasketaan polun kaarien uudet virtausarvot. Kaikki kaaret on suunnattu kohti nielua, joten kaikkien kaarien virtausarvoa kasvatetaan polun P_1 maksimikasvatusarvon verran. Uudet virtaukset ovat

$$f'(a) = f(a) + \epsilon(P_1) = 4 + 3 = 7$$
$$f'(c) = f(c) + \epsilon(P_1) = 4 + 3 = 7$$
$$f'(i) = f(i) + \epsilon(P_1) = 0 + 3 = 3.$$



Kuva 5.4. *Puu X_2 ja polku P_2.*

Nyt systeemin N kokonaisvirtausta on korotettu saturoimalla ensimmäisen saturoitumattoman puun avulla löydetty polku.

- ***Uuden saturoitumattoman puun kasvattaminen ja polun saturoiminen***
  Seuraavaksi palataan algoritmissa takaisin saturoitumattoman puun kasvattamiseen. Uusi saturoitumaton puu X_2 kasvaa eri tavalla, koska edellinen löydetty saturoimaton polku on nyt saturoitu. Tässä tapauksessa puuta X_2 kasvattaessa kaari c ei enää täytä puun kasvatusehtoja. Näin ollen solmua v_3 käsiteltäessä voidaan kaari g nyt lisätä puuhun X_2, koska kaaressa on virtausta ja se on suunnattu kohti puuta X_2

  Puusta X_2 saadaan muodostettua polku $P_2 = \{a, d, j\}$, joka kulkee lähteestä nieluun. Polun kaikki kaaret on suunnattu kohti nielua, joten polun maksimikasvatusarvo $\epsilon(P_2)$ lasketaan kuten edellisessä vaiheessa:

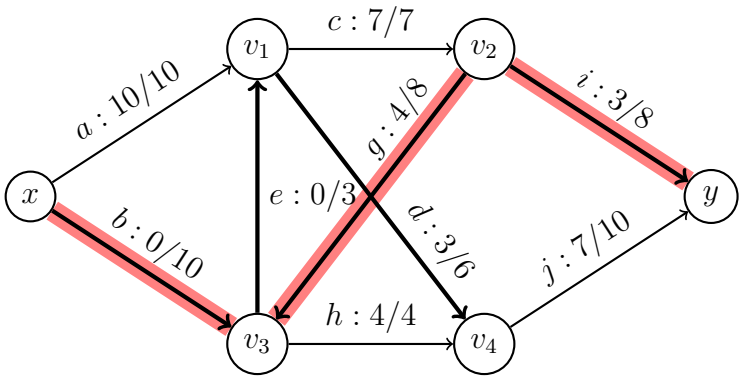
$$\epsilon(P_2) = min\{\epsilon(a) : a \in A(P_2)\} = min\{3, 6, 6\} = 3.$$

  Kaarien uudet virtausarvot ovat siis:

$$f'(a) = f(a) + \epsilon(P_2) = 10$$
$$f'(d) = f(d) + \epsilon(P_2) = 3$$
$$f'(j) = f(j) + \epsilon(P_2) = 7.$$

  Kuvassa 5.4 uusi kasvatettu puu X_2 on kuvattu vahvistetuilla nuolilla ja saturoitumaton polku P_2 on korostettu punaisella.

- ***Kolmannen saturoitumattoman puun kasvattaminen ja polun saturoiminen***
  Kolmas saturoitumaton puu X_3 saadaan kasvatettua kuten edellisissä vaiheissa, ja puu on kuvattu kuvassa 5.5. Polkua $P_3 = \{b, g, i\}$ saturoidessa tulee kuitenkin huomata, että kaari g on nyt suunnattu kohti lähdettä, jolloin sen maksimikasvatusarvo on kaaressa sillä hetkellä kulkeva virtaus, eli $\epsilon(g) = f(g) = 4$, jota vähennetään. *Maksimikasvatusarvot ovat*



Kuva 5.5. *Puu X_3 ja polku P_3.*

$$\epsilon(b) = c(b) - f(b) = 10 - 0 = 10$$
$$\epsilon(g) = f(g) = 4$$
$$\epsilon(i) = c(i) - f(i) = 8 - 3 = 5.$$

Polun maksimikasvatusarvo

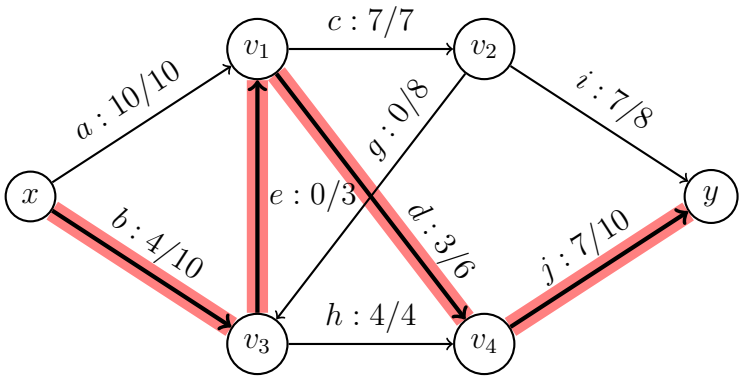
$$\epsilon(P_3) = min\{\epsilon(a) : a \in A(P_3)\} = min\{10, 4, 5\} = 4.$$

Lasketaan kaarien uudet virtausarvot. Kaari g on suunnattu kohti lähdettä, joten kaaren virtausta vähennetään polun maksimikasvatusarvon $\epsilon(P_3)$ verran. Tällä solmuun v_2 saapuvaa virtausta ohjataan kaaren g sijaan kaareen i, ja solmu v_3 saa virtauksen kaaresta b. Kaarien uudet virtausarvot ovat

$$f'(b) = f(b) + \epsilon(P_3) = 0 + 4 = 4$$
$$f'(g) = f(g) - \epsilon(P_3) = 4 - 4 = 0$$
$$f'(i) = f(i) + \epsilon(P_3) = 3 + 4 = 7.$$

- ***Neljännen saturoitumattoman puun kasvattaminen ja polun saturoiminen***
  X_4 saadaan vielä kasvatettua nieluun asti, jolloin systeemin virtausta voidaan vielä kasvattaa. Puusta X_4 muodostetaan polku $P_4 = \{b, e, d, j\}$, jonka maksimikasvatusarvo on

$$\epsilon(P_4) = min\{\epsilon(a) : a \in A(P_4)\} = min\{6, 3, 3, 3\} = 3.$$



Kuva 5.6. *Puu X_4 ja polku P_4.*

Kaarien uudet virtausarvot ovat

$$\begin{aligned} f'(b) &= f(b) + \epsilon(P_4) = 4 + 3 = 4 \\ f'(e) &= f(e) - \epsilon(P_4) = 0 + 3 = 3 \\ f'(d) &= f(d) - \epsilon(P_4) = 3 + 3 = 6 \\ f'(j) &= f(j) + \epsilon(P_4) = 7 + 3 = 10. \end{aligned}$$

X_4 ja $P_4 = \{b, e, d, j\}$ on kuvattu kuvassa 5.6.

- ***Viidennen saturoitumattoman puun kasvattaminen***
  Viidettä puuta X_5 saadaan kasvatettua vain niukasti, sillä kaaret a, e ja h ovat saturoituja oksista pois päin suuntautuvia kaaria. Kaari g on puun oksiin päin suuntautunut virtaukseton kaari. Puuta X_5 ei saada kasvatettua nieluun y asti, joten systeemin maksimivirtaus on saavutettu ja algoritmi päättyy.

- ***Algoritmin päättyminen***
  Puu $X_5 = \{x, v_3\}$ ei yllä nieluun asti, joten algoritmi päättyy tähän. Algoritmi on kasvattanut systeemin virtausta niin paljon kuin mahdollista. Viimeisin kokonaisvirtaus on systeemin maksimivirtaus

$$|f_{max}| = f(x) = f(a) + f(b) = 10 + 7 = 17.$$

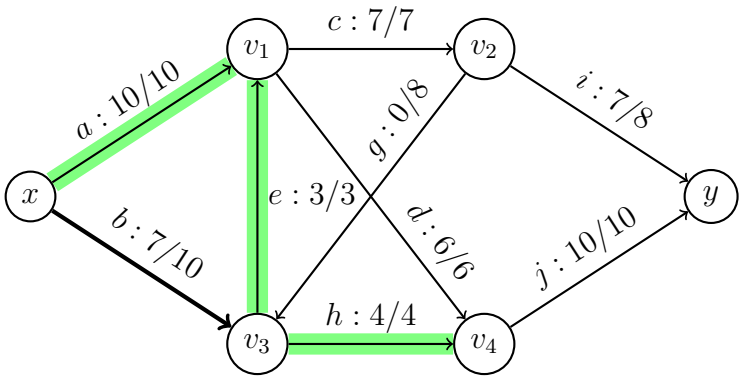
Puu X_5 on systeemin N IPS-puu. Systeemin N minimileikkaus on siis

$$K_{min} = \partial^+(X_5) = \{a, e, h\}.$$

Saadun minimileikkauksen kapasiteetti on yhtä suuri kuin saatu maksimivirtaus

$$K_{min} = |f_{max}| = 17.$$

Lauseen 4.6 nojalla K_{min} on systeemin minimileikkaus ja $|f_{max}|$ on systeemin mak-



Kuva 5.7. *IPS-puu X_5.*

simivirtaus. Kuvassa 5.7 systeemin N IPS-puu X_5 on kuvattu vahvistettuna ja minimileikkaus K_{min} vihreänä.

Algoritmin päättyessä päädytään tilanteeseen, jossa systeemin maksimivirtaus voidaan todistaa lauseella 4.6. Algoritmin toimivuus todetaan osoittamalla, että algoritmi päättyy *aina* maksimivirtaukseen.

Lause 5.2. *Olkoon virtaussysteemillä $N(x, y)$ kokonaisvirtaus $|f|$. Oletetaan myös, että systeemissä ei ole olemassa virtausta kasvattavaa polkua. Olkoon solmujoukko X systeemin IPS-puun solmujoukko. Kokonaisvirtaus $|f|$ on systeemin N maksimivirtaus ja $\partial^+(X)$ minimileikkaus.*

Todistus. Virtaussysteemin lähde $x \in X$. Systeemissä ei ole virtausta kasvattavaa polkua, joten IPS-puu ei yllä systeemin nieluun $y \in V \setminus X$.

Olkoon kaari $a(u, v) \in \partial^+(X)$. On olemassa osittainen virtausta kasvattava polku $p = (x, \ldots, u)$, koska solmu $u \in X$. Mikäli kaari a olisi saturoimaton, olisi olemassa osittain virtausta kasvattava polku $p = (x, \ldots, u, v)$, mutta koska $v \in V \setminus X$, polkua ei ole. Näin kaaren a on oltava saturoitu. Samankaltaisella päättelyllä voidaan osoittaa, että jos $a \in \partial^-(X)$, niin kaaren a on oltava virtaukseton.

Kaikki osaleikkauksen $\partial^+(X)$ kaaret ovat siis saturoituja ja osaleikkauksen $\partial^-(X)$ kaaret virtauksettomia. Nyt korollaarin 4.5 nojalla

$$|f| = \text{cap}(\partial^+(X)).$$

Lauseen 4.6 nojalla kokonaisvirtaus $|f|$ on maksimivirtaus ja osaleikkaus $\partial^+(X)$ minimileukkaus. □

Lause 5.3. *Ford–Fulkerson-algoritmi päättyy aina, kun virtaussysteemin $N(x, y)$ kaarien kapasiteetit ovat positiivisia kokonaislukuja.*

Todistus. Koska virtaussysteemin $N(x, y)$ kaikkien kaarien kapasiteetit ovat positiivisia kokonaislukuja, lauseen 4.6 nojalla myös systeemin maksimivirtaus on myös positiivinen kokonaisluku ja näin ollen äärellinen. Virtausta kasvattavaa polkua saturoidessa algoritmi kasvattaa systeemin kokonaisvirtausta vähintään yhden yksikön verran. Näin ollen algoritmi kasvattaa systeemin kokonaisvirtausta äärellisen monta kertaa, kunnes algoritmi saavuttaa maksimivirtauksen ja päättyy. [7, s. 192] □

Ford–Fulkerson-algoritmi voidaan helposti laajentaa toimimaan rationaaliluvuilla siten, että skaalataan virtaussysteemin kapasiteettien arvot pienimmällä yhteisellä jaettavalla [7, s. 191]. Näin virtaussysteemin kapasiteetit saadaan muutettua kokonaisluvuiksi, jolloin lauseen 5.3 nojalla algoritmi toimii. Mikäli virtaussysteemin kapasiteeteiksi sallitaan irrationaaliluvut, on mahdollista muodostaa virtaussysteemi, jossa algoritmi ei pääty koskaan. Tämän osoittivat Ford ja Fulkerson kirjassaan [6, s. 21]. Yksinkertaisempia esimerkkejä virtaussysteemeistä, joissa Ford–Fulkerson-algoritmi ei pääty esitteli Zwick myöhemmin vuonna 1995 [8].

6. YHTEENVETO

Tässä työssä esiteltiin Ford-Fulkerson-algoritmi sekä sen ymmärtämiseen vaadittavat graafiteorian käsitteet ja työkalut, kuten suunnatut graafit, leikkaukset ja puut. Työn oleellisin käsite, virtaussysteemi, on suunnattu graafi, jossa lähteen luoma virtaus matkaa välittäjäsolmuja pitkin nielusolmuun virtaussysteemin kaarien kapasiteettiehtoja noudattaen. Lähde on systeemin ainoa virtausta tuottava solmu, ja nielu ainoa virtausta syövä solmu. Virtaussysteemin kokonaisvirtaus on se nettovirtauksen määrä, joka syntyy lähteessä ja saapuu nieluun. Maksimivirtaus on suurin mahdollinen virtaussysteemin kaarien kapasiteettiehtoja noudattava kokonaisvirtaus. Maksimivirtaus on yhtä suuri kuin systeemin minimileikkaus, joka on se systeemin lähteen nielusta erottava leikkaus, jonka kapasiteetti on pienin. Minimileikkauksen ollessa täysin saturoitu on kokonaisvirtaus maksimivirtaus-minimileikkaus-lauseen mukaan suurin mahdollinen.

Ford-Fulkerson-algoritmin avulla voidaan selvittää systeemin maksimivirtaus ja minimileikkaus. Algoritmi etsii virtaussysteemistä lähteestä nieluun kulkevia saturoimattomia polkuja ja täyttää niitä virtauksella, kunnes saturoimattomia polkuja ei enää ole. Tällöin virtaussysteemin jokin leikkaus on täysin saturoitu, jolloin se todetaan minimileikkaukseksi ja senhetkinen kokonaisvirtaus maksimivirtaukseksi.

Ford-Fulkerson-algoritmin palauttama kokonaisvirtaus todettiin maksimivirtaukseksi maksimivirtaus-minimileikkaus-lauseella, ja algoritmin todettiin päättyvän luotettavasti kapasiteettien ollessa kokonais- tai rationaalilukuja. Tietyillä irrationaaliluvuilla algoritmi ei pääty.

LÄHTEET

[1] Riaz, F. ja Ali, K. M. Applications of Graph Theory in Computer Science. *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*. 2011, s. 142–145. DOI: 10.1109/CICSyN.2011.40.

[2] Kyi, M. ja Naing, L. Application of Ford-Fulkerson Algorithm to Maximum Flow in Water Distribution Pipeline Network. *International Journal of Scientific and Research Publications (IJSRP)* 8 (joulukuu 2018). DOI: 10.29322/IJSRP.8.12.2018.p8441.

[3] Koivisto, P. ja Niemistö, R. *Graafiteoriaa*. Tampereen yliopisto, 2018. URL: https://urn.fi/URN:ISBN:978-952-03-0681-6.

[4] Ruohonen, K. *Graafiteoria*. 2013. URL: http://math.tut.fi/~ruohonen/GT.pdf (viitattu 13. 02. 2020).

[5] Bondy, J. A. ja Murty, U. S. R. *Graph Theory*. 3. painos. Vol. 244. Graduate texts in mathematics. Springer, 2008. ISBN: 9781846289699.

[6] Ford Jr, L. R. ja Fulkerson, D. R. *Flows in Networks.* Princeton University Press, 1974. ISBN: 9780691079622.

[7] Wai-kai, C. *Net Theory And Its Applications: Flows In Networks.* Series in Electrical and Computer Engineering. Imperial College Press, 2003. ISBN: 9781860942266.

[8] Zwick, U. The smallest networks on which the Ford-Fulkerson maximum flow procedure may fail to terminate. *Theoretical Computer Science* 148.1 (1995), s. 165–170. ISSN: 0304-3975. DOI: 10.1016/0304-3975(95)00022-O.