TEKNILLINEN KORKEAKOULU
Teknillisen fysiikan ja matematiikan osasto
Teknillisen fysiikan koulutusohjelma

Esa Hyytiä

Optimaalinen reititys ja aallonpituuksien jako täysoptisessa WDM-järjestelmässä

Diplomi-insinöörin tutkintoa varten tarkastettavaksi jätetty diplomityö

Työn valvoja professori Olli Simula
Työn ohjaaja professori Jorma Virtamo

Espoo 30. maaliskuuta

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä: Esa Hyytiä

Työn nimi: Optimaalinen reititys ja aallonpituuksien jako täysoptisessa WDM-järjestelmässä

English title: Optimal routing and wavelength assignment in fully photonic networks

Päivämäärä: 30 maaliskuuta 1998 **Sivumäärä:** 66

Osasto: Teknillisen fysiikan ja matematiikan osasto **Professuuri:** Tik-115 Informaatiotekniikka

Valvoja: professori Olli Simula
Ohjaaja: professori Jorma Virtamo

Tässä työssä tarkastellaan reitityksen ja aallonpituusjaon ongelmaa täysoptisessa WDM-järjestelmässä. Mikäli verkon solmupisteissä ei ole mahdollista suorittaa aallonpituuskonversiota, käytetty reititys asettaa lisärajoituksia aallonpituusallokaatiolle. Tällaisen verkon optimaalinen konfigurointi on siten hankalampi tehtävä kuin esimerkiksi perinteisen puhelinverkon, jossa ainoana rajoituksena on jokaisen yhteysvälin maksimikapasiteetti.

Aallonpituuksien jako annetulla reitityksellä on olennaisesti graafin solmujen väritystehtävä, mikä on NP-hankala ongelma. Työssä esitetään joukko heuristisia menetelmiä solmujen värittämiseksi. Menetelmistä laadittiin C-kieliset toteutukset, joilla niiden toimintaa verrattiin. Lisäksi työssä esitetään algoritmi, joka pyrkii hakemaan sekä optimaalisen reitityksen että aallonpituusallokaation annetulle verkolle. Algoritmin toimintaa testattiin esimerkkiverkkojen tapauksissa.

Kokeilluista heuristisista algoritmeista sopivina tähän ongelmaan voidaan pitää ahneita algoritmeja ja tabu-hakua. Ahne algoritmi on hyvin nopea ja robusti menetelmä. Tabu-haku taas antaa parhaita värityksiä, mutta ajoaika on huomattavasti suurempi kuin ahneilla algoritmeilla. Reitityksen ja aallonpituusallokaation ratkaiseva algoritmi toimii hyvin, kunhan verkossa vain on riittävästi fyysisiä yhteyksiä verrattuna haluttuihin päästä päähän -yhteyksiin.

Avainsanat: WDM, reititys, aallonpituusallokointi, graafin väritys

Ei lainata ennen: **Työn sijantipaikka:**

HELSINKI UNIVERSITY OF TECHNOLOGY **ABSTRACT OF THE MASTER'S THESIS**

Author:	Esa Hyytiä		
Title of thesis:	Optimal routing and wavelength assignment in fully photonic networks		
Finnish title:	Optimaalinen reititys ja aallonpituuksien jako täysoptisessa WDM-järjestelmässä		
Date:	30th March 1998	**Pages:**	66
Department:	Department of Engineering Physics and Mathematics	**Chair:**	Tik-115 Information Science
Supervisor:	professor Olli Simula		
Instructor:	professor Jorma Virtamo		

In this work the problem of routing and wavelength assignment in a fully photonic WDM-system is studied. When there is no possibility for wavelength translation in nodes, the used routing sets new constraints to the wavelength assignment. The optimal configuration of such a network is thus clearly a harder problem than routing in traditional networks where the only limitation is the maximum capacity of each link.

The wavelength assignment involves coloring the nodes of a certain graph. The graph coloring is a NP-complete problem. In this work several heuristic graph coloring algorithms are studied. The algorithms were implemented in C-language for test purposes. Furthermore, an algorithm to solve both routing and wavelength assignment iteratively is studied.

The best algorithms for wavelength assignment seems to be greedy algorithms and tabu search. Greedy algorithms are quick and robust methods. Tabu search gives colorings with the fewest number of colors, but the running time is much bigger than with greedy algorithms. The algorithm which solves both routing and wavelength assignment works well as long as the number of physical links is comparable to the number of required point-to-point connections.

Keywords: WDM, routing, wavelength assignment, graph coloring

Not borrowable till: **Library code:**

Alkulause

Tämä diplomityö on tehty TKK:n teletekniikan laboratoriossa osana COST 257 -projektia.

Haluan kiittää työni ohjaajaa professori Jorma Virtamoa lukuisista diplomityötäni koskevista neuvoista ja työni tarkastamisesta. Lisäksi haluan kiittää työni valvojaa professori Olli Simulaa diplomityötä koskevien asioiden joustavasta hoitamisesta.

Lopuksi kiitokset teletekniikan laboratorion henkilökunnalle kaikesta avusta ja viihtyisästä työympäristöstä.

Espoossa, maaliskuun 30. päivänä 1998

Esa Hyytiä

Sisältö

Käytetyt lyhenteet

lyhenne	selitys
AON	all optical network, täysoptinen verkko
EDFA	erbium doped fiber amplifier, erbiumilla rikastettu kuituvahvistin
FDM	frequency division multiplexing, taajuusjakoinen multipleksointi
GA	genetic algorithm, geneettinen algoritmi
ITU	International Telecommunication Union, YK:n alainen kansainvälinen telealan standardoimisjärjestö
NRZ	non-return-to-zero (modulation), modulointitapa, jossa 1 merkitään koko jaksonajan kestävällä loogisella 1:llä ja vastaavasti 0 koko jaksonajan kestävällä 0:lla
PDFA	prasedymium doped fiber amplifier, prasedymium-nimisellä alkuaineella rikastettu kuituvahvistin
RZ	return-to-zero (modulation), modulointitapa, jossa jälkimmäinen puolisko jokaisesta jaksonajasta merkitään loogisella nollalla
SA	simulated annealing, simuloitu jäähdytys
TDM	time division multiplexing, aikajakoinen multipleksointi
WDM	wavelength division multiplexing, aallonpituusjakoinen multipleksointi
NP	nondeterministic polynomial bounded, ei polynomisessa ajassa ratkeavien ongelmien luokka

Käytetyt merkinnät

merkintä	selitys
$\Delta(G)$	graafin G suurimman solmun aste
$\delta(G)$	graafin G pienimmän solmun aste
$\nu(G)$	graafin G solmujen lukumäärä
$\epsilon(G)$	graafin G sivujen lukumäärä.

Luku 1

Johdanto

Tiedonsiirtotarpeiden kasvu tuo mukanaan kasvavia vaatimuksia runkoverkkojen kapasiteetille. Yhtenä ratkaisuna ongelmaan on ehdotettu aallonpituusjakoista multipleksointia (wavelength division multiplexing, WDM). WDM-tekniikassa yhdessä optisessa kuidussa kulkee useampi yhtäaikainen optinen signaali eri aallonpituuksilla. Tällöin jo olemassaolevan optisen verkon kapasiteetti saadaan paremmin hyödynnetyksi. Uusien optisten kuitujen vetäminen saattaa olla hyvinkin kallista (esim. Atlantin ylittävät yhteydet).

Aallonpituusjakoisen järjestelmän pääpiirteitä ovat:

- Täysoptinen verkko, jossa käytetään kuituvahvistimia.
- Jokaisessa optisessa kuidussa siirretään monta aallonpituuskanavaa.
- Tiedonsiirtoverkon kapasiteetti on hyvin suuri (useita Gb/s).
- Tiedonsiirtoverkko muodostaa “laajan” runkoverkon.
- Reititys solmupisteissä tapahtuu aallonpituuden perusteella.

Optista verkkoa suunniteltaessa on ensiksi määriteltävä haluttu fyysinen rakenne. Koska kyseessä on runkoverkko, halutut yhteydet ja liikennemäärät muuttuvat hitaasti ajan funktiona. Eli verkkoa ei tarvitse modifioida dynaamisesti, vaan verkon uudelleenkonfiguroinnin on harvinainen tapahtuma, eikä konfiguroinnin kesto ole kriittinen tekijä järjestelmässä. Myöskään uuden konfiguraation löytämiseen kuluva aika ei ole kriittinen, vaan voidaan käyttää hyvinkin monimutkaisia algoritmeja hyvän lopputuloksen löytämiseksi.

Tällaiset optiset verkot voidaan jakaa kahteen luokkaan solmupisteissä käytetyn tekniikan mukaan:

- Verkko, jonka solmuissa voidaan suorittaa aallonpituuskonversio.
- Verkko, jonka solmuissa aallonpituuskonversiota ei voida tehdä.

Mikäli solmuissa voidaan suorittaa aallonpituuskonversio, ei reititysongelma eroa perinteisestä tilanteesta, jossa käytössä on tietty rajallinen määrä kanavia linkkiä kohden. Optisessa verkossa käytetty tekniikka asettaa sitten rajoitukset käytössä olevien aallonpituuksien lukumäärälle linkkiä kohden.

Aallonpituuskonversio perinteisesti vaatii signaalin muuttamisen sähköiseen muotoon ja siitä takaisin valoksi, mikä lisää ratkaisun kustannuksia ja aiheuttaa vakioviiveen. Toisaalta taas jos aallonpituuskonversiota ei solmupisteissä voida suorittaa, asian käsittely muuttuu hankalammaksi. Tällöin yhteys kahden solmun välillä käyttää samaa aallonpituutta jokaisella linkillä matkallaan kohti määränpäätä. Millään linkillä ei luonnollisesti saa olla sama aallonpituus käytössä kahta tai useampaa kertaa. Kolmannen vaihtoehdon muodostavat ns. hybridiratkaisut, joissa osassa solmuista aallonpituuskonversio on mahdollista suorittaa ja toisissa ei. Käytännössä liikennemäärätkin tietysti vaihtelevat, joten verkon reitityksen muuttaminen dynaamisesti saattaisi olla perusteltua [18].

Tässä työssä on kuitenkin rajoituttu käsittelemään staattista reititystä ja aallonpituusallokaatiota. Täysoptisia aallonpituusjakoisia verkkoja käsitteleviä artikkeleita on olemassa huomattava määrä. Työssä käydään läpi ja vertaillaan kirjallisuudessa esitettyjä algoritmeja verkon konfiguroimiseen. Aallonpituusallokaatiossa joudutaan värittämään graafin solmut siten, että naapurisolmuilla on aina eri värit. Graafin solmujen väritystehtävä on yleisesti NP-hankala ongelma. Työssä käydään läpi erilaisia graafin solmujen väritysmenetelmiä tehden niihin joitakin lisäyksiä ja kokeillaan menetelmien toimivuutta satunnaisilla graafeilla. Lopuksi testataan reitityksen ja aallonpituusallokaation antavan algoritmin toimintaa muutamalla esimerkkiverkolla.

Luku 2

WDM-tekniikka

Täysoptinen aallonpituusjakoinen tiedonsiirto on eräs varteenotettava ratkaisu tulevaisuuden kasvaviin tiedonsiirtotarpeisiin. Valokaapelissa tapahtuva vaimennus on huomattavasti pienempää kuin kuparikoaksiaalikaapelissa [11][12][13]. Dispersiosiirretyn valokaapelin vaimennus on pienimmillään suunnilleen aallonpituudella 1,55 μm, missä se on noin 0.2 dB/km. Vastaavasti koaksiaalikaapelin vaimennus 1 GHz:in kantataajuudella on luokkaa 50 dB/km. Lisäksi kytkettäessä laitteita yhteen optisella kuidulla, haitallisista maatasoeroista ei tarvitse huolestua. Optinen tiedonsiirto ei ole sinänsä uutta tekniikkaa, vaan se on ollut käytössä jo vuosia.

Aallonpituusjakoinen multipleksointi (wavelength division multiplexing) on tekniikka, jossa yhtä optista kuitua pitkin siirretään useampia kanavia käyttäen eri aallonpituuksia. Idea on siis aivan sama kuin radioverkoissa käytettyssä taajuusjakoisessa multipleksoinnissa (frequency division multiplexing). Erona on vain se että WDM-järjestelmissä käytetty kantataajuus on noin miljoona kertaa suurempi, mikä tarjoaa mahdollisuuden selvästi suurempaan tiedonsiirtokapasiteettiin. Koska samassa optisessa kuidussa kulkevia kanavia voidaan käsitellä toisistaan riippumattomasti, eri kanavilla voidaan käyttää eri bittinopeuksia ja koodausta. WDM-kanavat ovat siten ’läpinäkyviä’[1] yhteyden päissä olevien laitteiden kannalta.

Täysoptinen verkko (all optical network, AON) on verkko, missä valosignaalia ei missään vaiheessa muuteta takaisin elektroniseen muotoon signaalin vahvistusta tai kytkemistä varten. Tälläinen verkko pyrkii kuljettamaan modulointitavasta ja -nopeudesta riip-

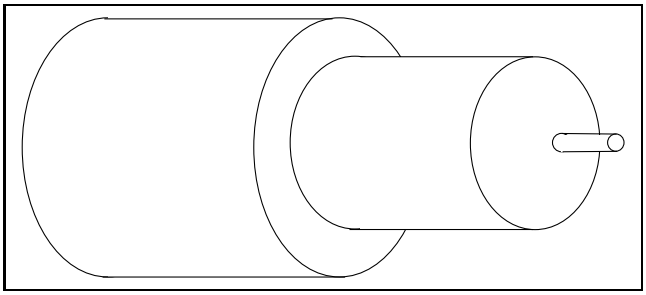
[1] transparency

pumatta valosignaalin muuttumattomana lähtöpisteeestä määränpäähän. Jokaisella reitin varrella olevalla linkillä valosignaalit erottaa toisistaan vain niiden aallonpituus. Verkon solmupisteissä valosignaalit reititetään eri suuntiin aallonpituuden perusteella (vrt. prisma).

2.1 Optiset kuidut

Optisen tiedonsiirron perustana ovat optiset kuidut, joissa valosignaalit kulkevat. Kuidun toiminta perustuu valon taittumis- ja heijastusilmiöön aineiden rajapinnassa. Se, mitä valonsäteelle tapahtuu rajapinnassa, riippuu aineiden taitekertoimista (Snellin laki).

Optinen kuitu on ympyräsymmetrinen ja muodostuu karkeasti ottaen kolmesta osasta (kuva 2.1). Kuidun ydin on rikastettua kvartsilasia (SiO_2), jossa valosignaalin on tarkoitus kulkea. Ydintä ympäröi paksumpi kerros kvartsilasia, jonka taitekerroin on pienempi kuin ytimen. Kun kuidun ytimessä kulkevan valonsäteen tulokulma on riittävän pieni, säde heijastuu rajapinnasta takaisin kohti ytimen keskustaa ja valosignaali pysyy kuidun sisällä. Optisen kuidun uloimpien kerrosten tehtävänä on suojata kaapelia.



Kuva 2.1: Yksinkertaistettu kuva optisesta kuidusta.

Optiset kuidut voidaan luokitella kahteen eri tyyppiin: monimuoto- ja yksimuotokuituihin. Sekä monimuoto- että yksimuotokuiduista on vielä olemassa hieman toisistaan poikkeavia tyyppejä. Yksimuoto- ja monimuotokuidun ominaisuuksien ero johtuu kuitujen taitekerroinprofiilien eroista, joka määrää sen kuinka valo kuidussa etenee [14].

Eräs yleinen monimuotokuitu on asteittaiskertoiminen monimuotokuitu (step index multimode fiber). Siinä taitekerroin muuttuu ytimessä nimensä mukaisesti asteittain ja valonsäde pyrkii pikkuhiljaa taittumaan kohti kuidun keskipistettä. Asteittaiskertoimisessa monimuotokuidussa valo etenee nopeammin ytimen reunoilla, mikä pienentää valon edetessä syntyvää muotodispersiota. Monimuotokuidun ytimen halkaisija on yleensä 50-100 μm.

Yksimuotokuidun ytimen halkaisija on huomattavasti pienempi kuin monimuotokuidussa, esimerkiksi 9 μm. Lisäksi ytimen ja kuoren taitekertoimien ero on sellainen, että kuidussa kulkee vain yksi aaltomuoto. Täten haitallista muotodispersiota ei yksimuotokuidussa esiinny. Koska vaimennus on selvästi pienempi kuin monimuotokuiduissa, yksimuotokuitu sopii paremmin pidempien yhteyksien rakentamiseen.

Nykyään tärkeimmät Suomessa käytössä olevat kuitutyypit ovat [14]:

- Monimuotokuidut
  - 50/125 μm, GI
  - 62,5/125 μm, GK
  - 100/140 μm, GN
- Yksimuotokuitu:
  - standardiyksimuotokuitu (9/125 μm), SM

GI,GK, GN ja SM ovat kyseisten kuitujen merkinnät.

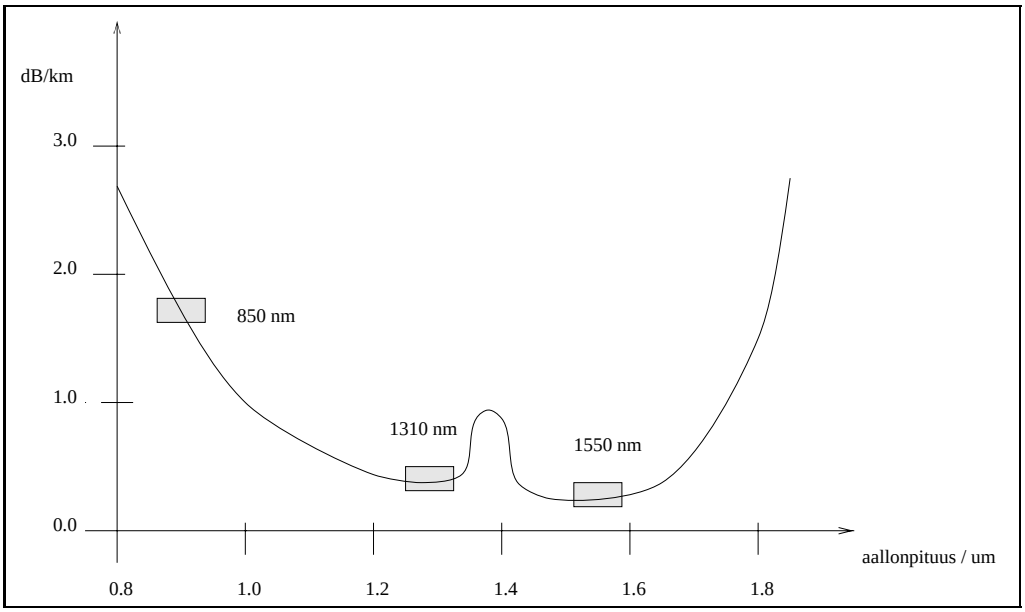
2.1.1 Kuitujen optiset ominaisuudet

Optisten kuitujen tärkeimmät optiset ominaisuudet ovat:

- vaimennus
- kaistanleveys (monimuoto)
- numeerinen aukko (monimuoto)

- dispersio (yksimuoto)
- raja-aallonpituus (yksimuoto)

Vaimennuksella tarkoitetaan kuidussa kulkevan valon intensiteetin pienenemistä. Vaimennuksen yksikkönä käytetään dB/km. Kvartsilasista valmistetun kuidulla vaimennus käyttäytyy kuvan 2.2 mukaisesti. Kuvassa näkyvä vaimennuspiikki on ns. vesipiikki.



Kuva 2.2: Optinen vaimennus kvartsilasista valmistetussa kuidussa.

.

Optisessa tiedonsiirrossa käytetään yleensä kolmea eri aallonpituusaluetta: 850 nm, 1310 nm ja 1550 nm. Kyseiset aallonpituusalueet on merkitty kuvaan 2.2 tummennettuina ruutuina.

Monimuotokuitujen kaistanleveys määrää kuidussa siirrettävän signaalin suurimman mahdollisen taajuuden tietyllä matkalla. Kaistanleveyden yksikkönä käytetään yleensä MHz·km. Rajallinen kaistanleveys on seuraus muotodispersiosta. Monimuotokuitujen numeerinen aukko puolestaan tarkoittaa suurimman sallitun tulokulman siniä. Mikäli valosignaali tulee suuremmassa kulmassa, ei se enää heijastukaan takaisin kohti kuidun keskustaa.

Yksimuotokuiduissa ilmenevä kromaattinen dispersio tarkoittaa ilmiötä, missä lähellä toisiaan olevat aallonpituudet kulkevat hiukan eri nopeuksilla kuidussa. Kromaattisen dispersion yksikkönä käytetään ps/(nm · km). Standardiyksimuotokuidun (SM) minimidispersio

on aallonpituudella 1310 nm ja dispersiosiirretyn yksimuotokuidun (DS) minimidispersio aallonpituudella 1550 nm.

WDM-tekniikka soveltuu suurien runkoverkkojen rakentamiseen ja niissä yhteyksien pituudet ovat usein suuria. Täten yksimuotovalokuitu sopii monesti paremmin WDM-verkkoihin.

WDM-tekniikan yhteydessä käytettyjen kuitujen minimivaimennus sijaitsee yleensä joko 1310 tai 1550 nanometrin paikkeilla. Jälkimmäisen dispersiosiirretyn valokaapelin minimivaimennusalue osuu sopivasti päällekkäin erbiumilla rikastetun kuituvahvistimen (ks. kappale 2.2) kanssa, mikä tekee siitä erittäin lupaavan vaihtoehdon.

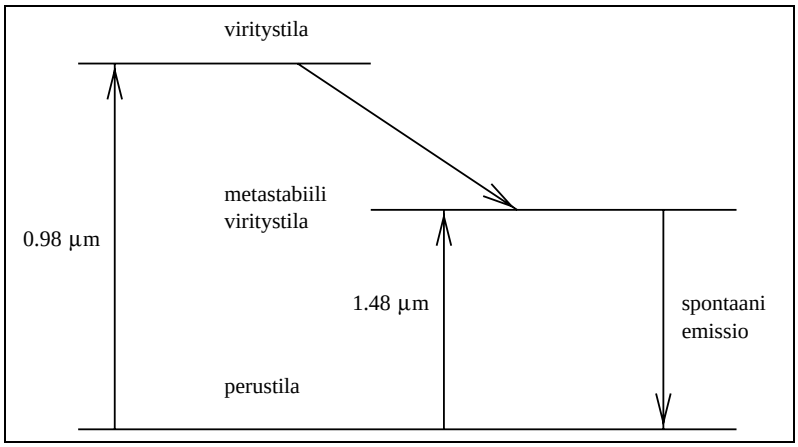
2.2 Kuituvahvistin

Eräs olennainen ongelma optisessa tiedonsiirrossa on ollut signaalin vaimeneminen suurilla siirtoetäisyyksillä. Optisen signaalin vahvistamiseksi se on pitänyt ensiksi muuttaa elektroniseen muotoon ja konstruoida sieltä takaisin optiseksi. Jokainen eri aallonpituus piti siis vahvistaa erikseen. Lisäksi vahvistinyksiköt ovat modulointikohtaisia ja siirtyminen muuhun modulointimenetelmään vaatisi aina vahvistinyksikköjen päivityksen. Tällaisen monimutkaisen vahvistinyksikön hinta nouseekin helposti hyvin korkeaksi.

Vuonna 1987 joukko Southamptonin yliopiston tutkijoita onnistui kehittämään täysoptisen kuituvahvistimen, jossa signaalia ei missään vaiheessa muuteta elektroniseen muotoon. Vahvistin perustuu erbium-nimisen alkuaineen ominaisuuksiin. Erbium emittoi valoa, jonka aallonpituus vastaa suunnilleen dispersiosiirretyn valokuidun optimialuetta (1550 nm).

Kun erbium-ioneilla rikastettua valokuitua pommitetaan fotoneilla, joiden aallonpituus on 980 nm tai 1480 nm, perustilassa oleva erbium-ioni siirtyy korkeampaan viritettyyn tilaan (kuva 2.3). Mikäli laserin aallonpituus on 980 nm, ioni siirtyy hyvin pian metastabiiliin tilaan, mihin 1480 nm:n aallonpituudella olisi suoraan päädytty.

Tässä metastabiilissa tilassa erbium-atomi viipyy normaalisti muutamia mikrosekunteja tai jopa millisekunteja, jonka jälkeen se emittoi fotonin ja palaa perustilaan. Spontaanissa emissiossa vapautuvan fotonin suunta on satunnainen. Tämä aiheuttaa vahvistimessa tietyn additiivisen kohinan, jonka suuruus on 4-5 dB:n luokkaa.



Kuva 2.3: Erbium-vahvistimen toimintaperiaate.

Optisessa tiedonsiirrossa käytetty bittinopeus on varsin suuri verrattuna erbium-ionin metastabiilin tilan elinikään. Kun metastabiilissa tilassa olevan ioniin osuu fotoni, tapahtuu spontaanin emission sijaan stimuloitu emissio, jossa ioni palaa perustilaansa ja emittoi toisen fotonin, jonka aallonpituus ja vaihe ovat samat. Tällöin sisääntuleva optinen signaali vahvistuu.

Optisen vahvistimen kaista on noin 3 THz ja vahvistus 20-40 dB. Lisäksi vahvistimen vahvistus on hyvin vähän riippuvainen polarisaatiosta. Kuituvahvistimista on olemassa jo myös kaupallisia versioita.

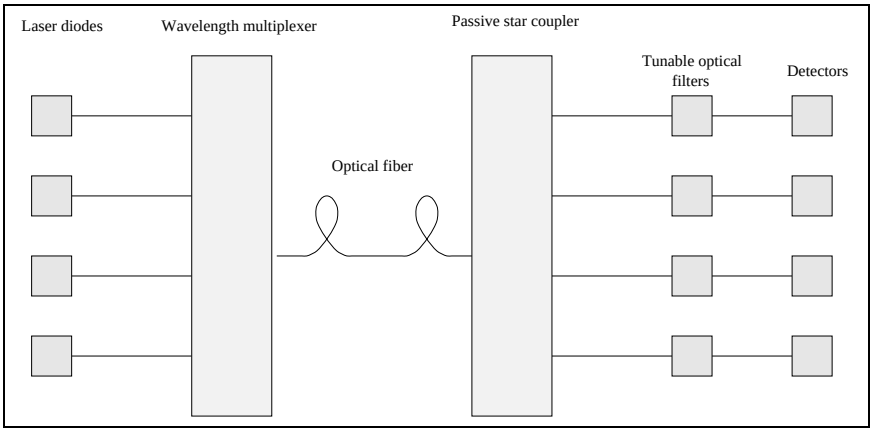
Erbiumpohjainen kuituvahvistin (EDFA, erbium doped fiber amplifier) toimii siis vain 1550 nm alueella. Vastaavanlainen vahvistin on myös onnistuttu rakentamaan 1300 nm taajuusalueelle, jolloin erbium on korvattu praseodyymi nimisellä alkuaineella (Pr).

2.3 Aallonpituusjakoinen multipleksointi

Jotta olemassa olevien optisten kuitujen kapasiteetti saataisiin paremmin hyödynnetyksi, kannattaa siirtyä käyttämään useampaa aallonpituutta. Tällöin eri aallonpituudet tulee pystyä yhdistämään (multipleksointi) lähetyspäässä ja vastaavasti erottelemaan (demul-

tipleksointi) yhteyden toisessa päässä (kuva 2.4). Laserdiodeilla muodostetaan joukko optisia signaaleja, jotka multipleksataan yhteen. Diodien aallonpituudet on sopivasti valittu siten, että kanavien kantataajuudet ovat riittävän etäällä toisistaan (0.4-4 nm).

Tämän jälkeen yhdistetty signaali kulkee optista kuitua pitkin linkin toiseen päähän, jossa se johdetaan useampaan optiseen suodattimeen. Optisista suodattimista kukin päästää vain yhden aallonpituuden läpi.

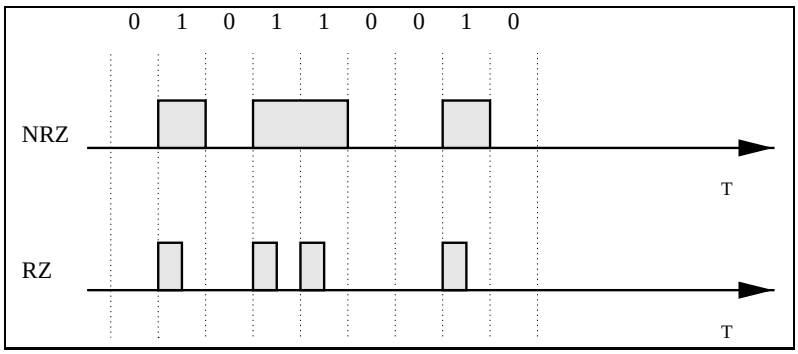


Kuva 2.4: Aallonpituusjakoinen multipleksointi.

Mikäli optinen siirtotie on pitkä, joudutaan käyttämään vahvistimia. Jotta jokainen yksittäinen aallonpituus käytettäisiin mahdollisimman tehokkaasti hyväksi, joudutaan WDM-tekniikan lisäksi käyttämään TDM-tekniikkaa.

Modulaatiot

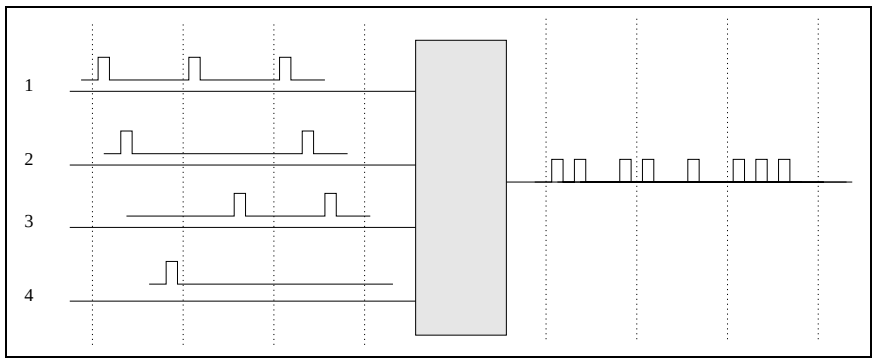
Optisissa järjestelmissä käytetään käytännössä kahta modulaatiota: NRZ (non-return-to-zero) ja RZ (return-to-zero). NRZ-modulointi on yksinkertainen toteuttaa ja se sopii hyvin alhaisille bittinopeuksille. NRZ-modulaatiossa looginen ykkösen aikana valolähde on päällä koko jakson ajan ja vastaavasti loogisen nollan aikana valonlähde on kytketty pois päältä. RZ-modulaatiossa valo on päällä korkeintaan ensimmäisen puoliskon yhden bitin ajasta riippuen siitä, onko siirrettävä bitti yksi vai nolla. Jälkimmäisen puoliskon lähde on aina pimeänä. RZ-modulaatiossa valopulssien pituudet ovat siis kestoltaan puolet NRZ-modulaation tapauksesta, mikäli tiedonsiirtonopeus on sama. Kun bittinopeudet nousevat, kasvaa synkronoinnin merkitys. Täten RZ-modulaation tarjoama parempi synkronointi tekee siitä paremman vaihtoehdon suuremmilla bittinopeuksilla.



Kuva 2.5: Optisen signaalin modulointi.

Pienillä bittinopeuksilla valolähteen modulointi voidaan hoitaa suoraan moduloimalla laserdiodin käyttövirtaa toimintapisteen ympäristössä. Tämä toimii mainiosti 1 Gb/s bittinopeuksille asti. Käytetty modulaatio on yleensä NRZ.

Nopeampiin bittinopeuksiin on päästy käyttämällä ulkoista optista modulaattoria, joka katkoo laserdiodista tulevaa yhtenäistä valoa koodattavan signaalin mukaisesti. Tällä tekniikalla päästään 40 Gb/s nopeuksiin. Tätä nopeammille bittinopeuksille ei ole olemassa lasereita, jotka kykenisivät tuottamaan tarpeeksi lyhyitä pulsseja toistuvasti.



Kuva 2.6: Optinen viivelinja.

Jotta päästäisiin vielä suurempiin bittinopeuksiin kanavaa kohti, pitää avuksi ottaa TDM-tekniikka. Laserdiodilla saadaan aikaan riittävän lyhyitä pulsseja, mutta pulssien välinen aika kasvaa liian suureksi. Käyttämällä optisia viivelinjoja voidaan useampi optinen signaali kytkeä yhteen kuvan 2.6 osoittamalla tavalla. Optisen viivelinjan ansiosta yksittäisen kanavan laserdiodin toistotaajuuden ei tarvitse olla niin suuri, ja laboratoriokokeissa onkin päästy jopa 400 Gb/s bittinopeuksiin kanavaa kohti. Tälläisilla nopeuksilla on syytä käyttää RZ-modulaatiota, jotta voidaan varmistaa lähetys- ja vastaanottopään synkronissa pysyminen. Yhdistetty signaali voidaan vahvistaa välittömästi esimerkiksi edellä esitetyllä optisella kuituvahvistimella.

2.3.1 Demultipleksointi

Elektroninen demultipleksointi onnistuu noin 40 Gb/s bittinopeuksille asti. Tämän jälkeen joudutaan keksimään uusia keinoja lähetys- ja vastaanottopään kellojen synkronissa pitämisen lisäksi. Eräs lupaavalta tuntuva ratkaisu on epälineaarinen optinen “loop” peili, jonka avulla laboratoriokokeissa on onnistuttu demultipleksaamaan jopa 100 Gb/s kanavanopeuksia.

2.4 Epäideaalisuudet ja muut ongelmat

Vaikka optisen valokaapelin vaimennus on pieni verrattuna koaksiaalikaapelin vaimennukseen, tarvitaan silti vahvistimia. Kuituvahvistimet (EDFA) eivät kuitenkaan ole lineaarisia, ja kahden aallonpituuden summasignaalista muodostuu kaksi uutta häiriösignaalia taajuuksille, jotka ovat alkuperäisten signaalien taajuuksien erotus ja summa. Ilmiötä kutsutaan neliaaltosekoitukseksi (four-wave mixing).

Toinen merkittävä rajoite optiselle tiedonsiirrolle on valokaapeleissa ilmenevä dispersio eli laajakaistaisen pulssin leviäminen, kun valopulssi liikkuu pitkin valokaapelia. Ilmiö johtuu siitä, että pidempien aallonpituuksien etenemisnopeus on joko suurempi tai pienempi kuin lyhyempien. Dispersion aiheuttamia haittoja voidaan pienentää käyttämällä dispersiosiirrettyä kuitua. Toinen tapa, jolla dispersion haittojen lisäksi kuidun epälineaarisuudesta johtuvia häiriöitä voidaan pienentää, on käyttää kuidussa määrävälein dispersion kompensoivia suotimia. Suotimet toimivat päinvastoin kuin optinen kuitu ja hidastavat kuidussa nopeammin kulkevia aallonpituuksia.

2.5 Nykytilanne

Tämän hetken kaupalliset WDM-ratkaisut pystyvät tyypillisesti käsittelemään 4, 8, 16 tai 32 eri aallonpituutta. Kansainvälinen telealan standardoimisjärjestö ITU on ehdottanut aallonpituusjakoisen optisen verkon perusaallonpituudeksi 1552,5 nm ja kanavaväliksi n. 0,8 nm. Tämän hetken kaupalliset tuotteet eivät tähän vielä kuitenkaan yllä, vaan käyttävät esim. 1,6 nm:n kanavaväliä.

Useammassa laboratoriossa on jo päästy yli terabitin nopeuksiin muutamien kymmenien kilometrien matkoilla. Toisaalta myös pidemmillä matkoilla ollaan edistytty ja 100 Gb/s on onnistuttu siirtämään yli 9000 kilometrin päähän dispersiosiirretyssä optisessa kuidussa. Tällaisten tuotteiden kaupallisten versioiden voidaan olettaa saapuvan markkinoilla muutamien vuosien sisällä.

Luku 3

Reitityksen ja aallonpituuksien valinnan ongelma WDM-verkossa

Kuten edellä kävi ilmi, WDM-verkon suunnittelu vaatii ottamaan huomioon myös yhteyksien käyttämän aallonpituuden reitityksen yhteydessä [16][15][19]. Jokaiselle yhteydelle pitää siis löytää sopiva reitti ja aallonpituus. Jos jokaisen linkin kapasiteetti oletetaan rajattomaksi, edullinen reititys on helppo löytää esimerkiksi Dijkstran (ks. kappale 4.2.1) tai Floydin (ks. kappale 4.2.2) algoritmeilla, joilla löydetään lyhimmät reitit jokaisen pisteparin välille. Toisin sanoen signaalien etenemisviive tulisi tällöin minimoiduksi.

Todellisuudessa linkkien kapasiteetit ovat kuitenkin rajallisia. Esimerkiksi televerkossa voidaan ajatella, että jokaisella linkillä on tietty maksimi kapasiteetti, tietty määrä linjoja. Joten lisäehtona on, että linkkien maksimikapasiteettia ei saa ylittää. Lyhimmän polun algoritmilla saadut reitit eivät siten välttämättä kelpaa.

Kun kyseessä on aallonpituusjakoinen täysoptinen verkko, jossa solmupisteissä ei voida suorittaa aallonpituuskonversiota, tulee reititysongelmasta vielä monimutkaisempi. Tällöin eri linkkien väliset riippuvuudet nousevat hyvinkin monimutkaisiksi ja ongelman täydellinen ratkaisu tuntuu mahdottomalta.

Mikäli ensin kiinnitetään yhteyksien käyttämät aallonpituudet, tehtäväksi jää yhteyksien reititys siten, että aallonpituuskonfliktia ei syntyisi verkon millään yhteysvälillä. Intuitiivisesti ajatellen on hyvin epävarmaa, että tällöin kaikille yhteyksille löydettäisiin aina kelvollinen reitti .

Olettaen, että värien lukumäärää ei ole rajoitettu, voidaan mille tahansa reititykselle löytää aallonpituusallokaatio, jossa aallonpituuskonflikteja ei esiinny. Näin ollen on luultavasti parempi ensin pyrkiä löytämään lupaava reititys, jonka jälkeen kiinnitetyille reiteille pyritään antamaan optimaalinen sallittu aallonpituusallokaatio.

Kolmas mahdollinen vaihtoehto on, että sekä reititystä että väritystä vaihdetaan ja muokataan yhtäaikaa, kunnes mahdollisesti löydetään hyvän tuloksen antava kombinaatio. Tällainen menettely ei tunnu kovin lupaavalta johtuen ongelman monimutkaisuudesta.

Edullisen WDM-verkon konfiguraation etsiminen on siten syytä aloittaa reitityksestä. Reitityksen jälkeen on vuorossa reittien väritys, joka palautuu graafin solmujen väritysongelmaksi. Tämä olisi siten syytä ottaa jotenkin huomioon eri reittivaihtoehtoja verrattaessa.

3.1 Huomioita reitityksestä ja tarvittavista aallonpituuksista

Tarvittavien värien määrä riippuu olennaisesti valitusta reitityksestä (ks. kappale 4.3). Kiinnittämättä sen enempää huomiota graafin solmujen värityksen ongelmaan voidaan kuitenkin tehdä seuraavat huomiot reitityksestä:

- Selvästi aallonpituuksia allokoitaessa voidaan unohtaa ne reitit, joiden pituus on yksi, sillä tällaisen yhteyden väriksi voidaan valita mikä tahansa vapaista väreistä eikä valinnalla ole vaikutusta kyseisen linkin ulkopuolle. Ts. kyseisen reitin väri on “jokeri”.
- Yleensä pitkät reitit ovat huonoja, koska ne kuormittavat useampaa linkkiä ja täten varaavat värin useammalta yhteydeltä.
- Kannattaa ehkä pyrkiä minimoimaan yhdelle linkille tulevien yhteyksien lukumäärää, sillä se on ehdoton alaraja tarvittavien aallonpituuksien lukumäärälle.
- Kannattaa pyrkiä minimoimaan päästä päähän -yhteyden kanssa yhteisiä linkkejä omaavien yhteyksien lukumäärää (4.1) (ehdoton yläraja).

- Kun verkko jaetaan kahteen joukkoon, menee tämän rajapinnan läpi tietty määrä yhteyksiä. Nyt näiden yhteyksien lukumäärä jaettuna rajapinnan linkkien lukumäärällä antaa selvästi alarajan värien lukumäärälle [17]. Tämä on oleellisesti edellisen yleistys ja pätee siis myös muunlaisille verkoille, joissa linkeillä on kapasiteettirajoitukset.
- Ehdottomana ylärajana tarvittavien aallonpituuksien lukumäärälle on tietenkin haluttujen yhteyksien lukumäärä, sillä voidaanhan jokaiselle yhteydelle antaa oma aallonpituus. Tällainen menettely ei selvästikään voi johtaa hyvään lopputulokseen, ja käytännössä selvitäänkin aina selvästi pienemmällä määrällä aallonpituuksia.

Verkon kahden solmun välinen lyhin reitti on perinteisesti ratkaistu esimerkiksi Dijkstran (kappale 4.2.1) tai Floydin (kappale 4.2.2) algoritmilla. Dijkstran algoritmi hakee lyhimmän reitin annetusta solmusta kaikkiin muihin solmuihin, kun taas Floydin algoritmin tuloksena saadaan kaikkien solmuparien väliset lyhimmät reitit. Molempien algoritmien kompleksisuus on sama $O(v^3)$, kun haetaan yhteydet kaikkien solmuparien välille. Floydin algoritmi on kuitenkin yleensä nopeampi.

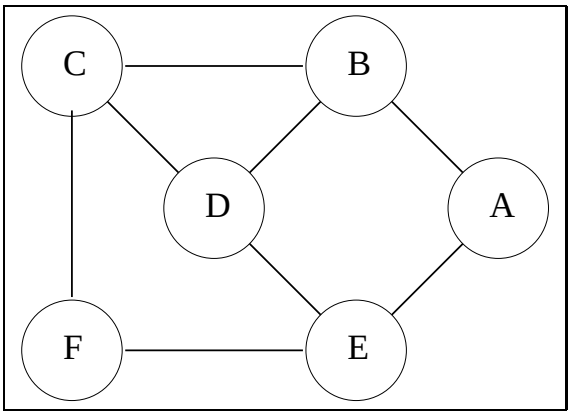
Kun annetulle reititykselle on löydetty joku kohtuullinen väritys, voidaan väritystä tutkimalla löytää ne linkit, joilla on “ruuhkaa”. Painottamalla tällaisia ruuhkaisia linkkejä voidaan iteroimalla pyrkiä löytämään parempi reititys [15]. Tällainen lähestymistapa vaatii, että värityksen hakevan algoritmin suoritusaika ei ole suuri.

Toinen aallonpituusallokaatiota helpottava tekijä olisi useamman optisen kuidun käyttö ruuhkaisilla linkeillä. Tällöin aallonpituuskonfliktien välttäminen olisi merkittävästi helpompaa, sillä jokaista väriä olisi käytössä useampi kappale verkon kahden solmun välillä. Esimerkiksi lisäämällä muutamalle ruuhkaisemmalle yhteysvälille toinen optinen kuitu avuksi, saattaa tarvittavien aallonpituuksien lukumäärä pudota merkittävästi. Verkon suunnittelussa ja reitityksessä olisikin hyvä, jos tällaiset pullonkaulana toimivat yhteysvälit pystyttäisiin löytämään ja siten korjaamaan helposti.

3.2 Aallonpituusallokaatio

Uutena piirteenä reititykseen aallonpituusjakoinen multipleksointi tuo aallonpituusallokaation. Oletetaan, että verkon topologinen rakenne on annettu ja kiinteä. Samoin käytet-

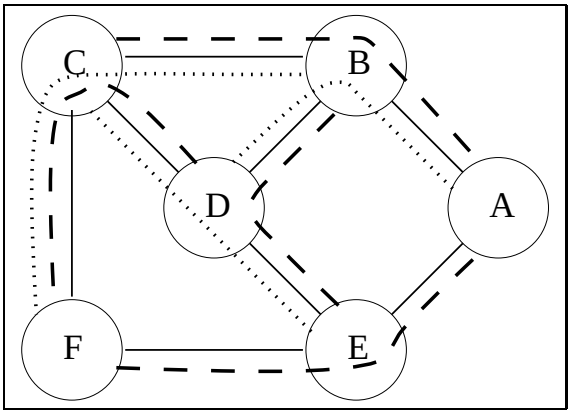
ty reitti verkon kahden solmun välillä on määrätty etukäteen jollakin menetelmällä. Joten verkon täydellisestä konfiguroimistehtävästä jäljellä on vain sopivien aallonpituuksien allokointi yhteyksille mahdollisimman pienellä lukumäärällä eri aallonpituuksia. Jatkossa väreistä ja värityksestä puhuttaessa tarkoitetaan käytettyjä aallonpituuksia.



Kuva 3.1: Esimerkkiverkko.

Huomioitavaa on, että tässä ei oleteta jokaisen solmuparin välillä olevan reittiä. Lisäksi sallitaan myös mahdolliset useammat reitit samojen solmujen välille. Esimerkiksi joidenkin solmupisteiden välillä saattaa olla niin suuri kapasiteetin tarve, että yhden yhteyden sijasta tarvitaan kaksi yhteyttä. Vastaavasti joidenkin solmupisteiden väliset liikennemäärät saattavat olla niin pieniä, että kokonaisen yhteyden varaaminen olisi suoranaista kapasiteetin tuhlausta. Tällöin kyseisen solmuparin välinen liikenne voidaan hyvin kierrättää kolmannen solmupisteen kautta.

Käytettävissä olevat aallonpituudet pyritään nyt allokoimaan siten, että eri aallonpituuksia olisi käytössä mahdollisimman vähän. Käytännön toteutuksissa aallonpituuksien maksimilukumäärän rajoittaa kulloinkin käytetty tekniikka. Tyypillisesti tämä lukumäärä on muutamia kymmeniä, mutta laboratoriokokeissa on päästy jo huomattavasti suurempiin lukumääriin.



Kuva 3.2: Esimerkkiverkko, reitit valittu ja värit allokoitu.

Eräs optimaalinen konfiguraatio kuvan 3.1 yksinkertaiselle verkolle on esitetty kuvassa 3.2. Kyseisen kuusi solmua sisältävän verkon tapauksessa siis tarvittaisiin vähintään kolme aallonpituutta, jotta saataisiin aikaan täysin kytketty verkko[1]. Esimerkkiverkon tapauksessa linkkien kapasiteetit ovat myös hyvin optimaalisesti käytetyt, sillä vain kahdella linkillä on yksi väri käyttämättä.

3.2.1 Matriisimuoto

Yhteyksien väritysongelma voidaan esittää myös yhteysmatriisien avulla seuraavasti. Olkoon verkossa solmut $a \ldots e$. Allaolevassa binäärimatriisissa on x, mikäli reiteillä on yksi tai useampi yhteinen linkki. Ts. x tarkoittaa, että kyseisillä yhteyksillä tulee käyttää eri värejä.

reitit	ab	ac	ad	ae	bc	bd	be	cd	ce	de
ab	x	x							x	
ac	x	x		x	x					
ad			x						x	
ae		x		x		x	x			
bc		x			x					
bd				x		x				
be				x			x			
cd								x		x
ce	x		x						x	
de								x		x

Esimerkiksi solmujen b ja d välinen yhteys sisältää ainakin yhden yhteisen linkin solmujen a ja e välisen yhteyden kanssa jne. Nyt tehtävänä on valita kullekin reitille oma väri

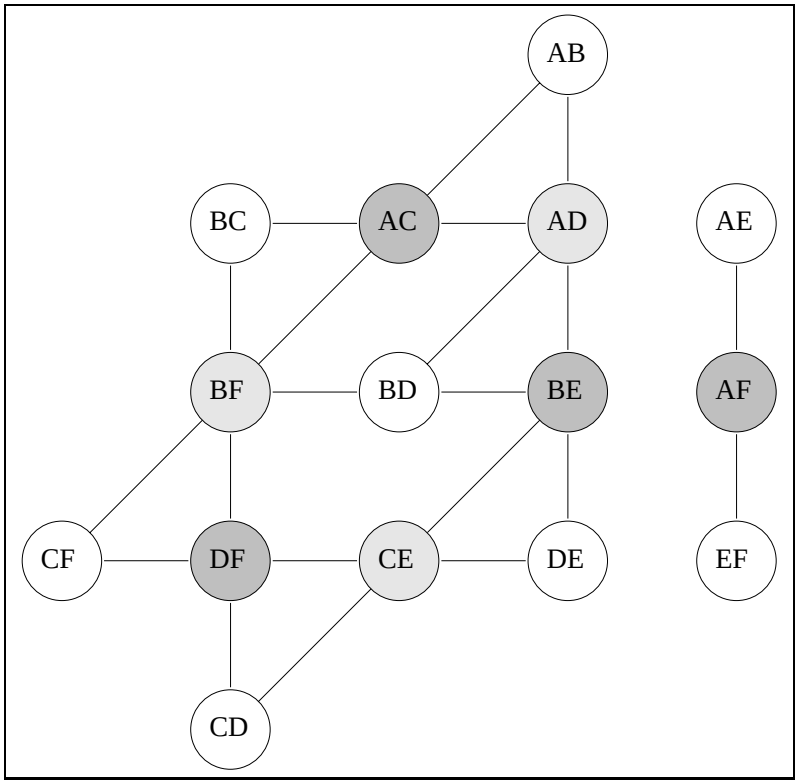
[1] fully connected network, verkon jokaisen solmuparin välillä on yhteys

siten, että reiteillä, joilla on yhteisiä linkkejä, on eri värit ja samalla käytettyjen värien lukumäärä on mahdollisimman pieni. Yhteysmatriisi on luonnollisesti symmetrinen.

3.2.2 Graafimuoto

Sama ongelma voidaan esittää graafien avulla. Tehtävä palautuu graafin solmujen väritysongelmaksi (ks. kappale 4.3). Graafin solmujen väritys yleisesti on NP-hankala ongelma. Sitä on yritetty ratkaista mm. erilaisilla heuristisilla menetelmillä kuten geneettisillä algoritmeilla ja simuloidulla jäähdytyksellä [1].

Olkoon jokainen päästä päähän -yhteys graafin solmu. Yhdistetään toisiinsa ne solmut (=reitit), joilla on yhteisiä linkkejä, jolloin niistä tulee toistensa naapurisolmuja. Kuvassa 3.3 on esimerkkiverkkoa vastaava graafin väritysongelma[2].



Kuva 3.3: Esimerkkiverkkoa vastaava väritysongelma.

[2]Kyseessä on tasograafi, joten neliväriteoreeman mukaan se voidaan värittää ainakin neljällä värillä

Saadun graafin solmut tulee värittää siten, että vierekkäiset solmut ovat aina erivärisiä. Tällöin reitit, jotka kulkevat jonkun yhteisen linkin kautta, käyttävät eri aallonpituuksia. Väritettävä graafi on olennaisesti edelläesitetty 'yhteysmatriisi'. Esimerkkigraafin kuvasta nähdään, että suurin klikki[3] ko. graafissa sisältää kolme solmua. Selvästi klikin kaikki solmut pitää värittää eri väreillä. Kolme onkin kyseisen graafin kromaattinen luku eli graafille löytyy sallittu väritys kolmella värillä mutta ei enää kahdella.

Klikki väritysgraafissa tarkoittaa sitä, että kyseiset yhteydet käyttävät kaikki samaa linkkiä jossakin vaiheessa. Suurin klikki antaakin heti ehdottoman alarajan tarvittavien värien lukumäärälle. Suurimman klikin löytäminen on myös NP-hankala ongelma, joten se ei tarjoa oikotietä laadukkaaseen väritykseen. Klikkien avulla voi kyllä yrittää ratkaista graafin solmujen väritysongelmaa, ja siihen perustuvia algoritmeja onkin olemassa paljon.

3.3 Reititys ja aallonpituusallokaatio

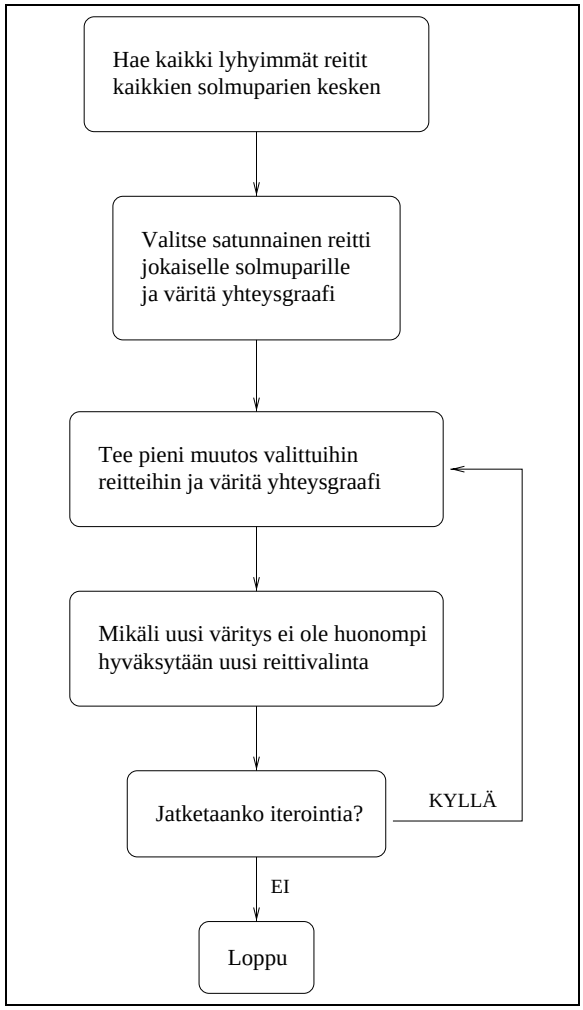
Oletetaan, että annettuna on verkon topologinen rakenne eli käytettävissä olevat linkit. Verkon reititys on kuitenkin jätetty vielä avoimeksi. Nyt pitäisi siis keksiä jokaiselle halutulle yhteydelle (=solmuparille) reitti ja aallonpituus siten, että valitut aallonpituudet eivät törmää millään linkillä. Kuten edellä kävi ilmi, jo aallonpituusallokaatio on NP-hankala ongelma. Joten kun reititys otetaan myös vapaaksi parametriksi, optimointitehtävä on vielä vaikeampi.

3.3.1 Ratkaisu kahdessa vaiheessa

Annettu optimointiongelma voidaan ratkaista kahdessa vaiheessa: keksitään ensin edullinen reititys, jonka jälkeen allokoidaan yhteyksien aallonpituudet kappaleen 3.2 mukaisesti. Tämä menettely ei luonnollisestikaan takaa mitään optimaalista ratkaisua.

[3] Solmut muodostavat klikin, mikäli ne ovat kaikki toistensa naapureita, ks. kappale 4.1

Kun ensimmäisessä vaiheessa unohdetaan tuleva värien allokointi, tehtävä palautuu perinteiseksi reititysongelmaksi: kuinka monta johtoa tarvitaan, jotta halutut yhteydet voidaan tarjota. Toisin sanoen minimoidaan yhdelle linkille tulevien yhteyksien määrä. Mikäli solmupisteissä voidaan suorittaa aallonpituuskonversio, värien allokointi jää pois, ja tämä onkin optimaalinen ratkaisu. Tässä on kuitenkin päädytty hieman erilaiseen menettelyyn johtuen reititysvaihetta seuraavasta aallonpituusallokaatiosta.



Kuva 3.4: Reitityksen ja aallonpituusallokaation iteratiivisesti hakeva algoritmi.

Kuvassa 3.4 on esitetty algoritmi, jolla iteratiivisesti pyritään löytämään sellainen reititys, jolle on olemassa mahdollisimman vähän aallonpituuksia tarvitseva aallonpituusallokaatio. Huomattavaa on, että algoritmi kokeilee vain sellaisia reittivaihtoehtoja, joiden pituus on lyhin mahdollinen. Reitin pituudella ei tässä tarkoiteta käytettyjen linkkien yhteenlaskettua fyysistä pituutta, vaan reitin käyttämien linkkien lukumäärää. Yleensä tämä onkin perusteltua, sillä tällöin jokainen yhteys vie mahdollisimman vähän yhteisiä resursseja eli

kaistaa käytettävissä olevilta linkeillä.

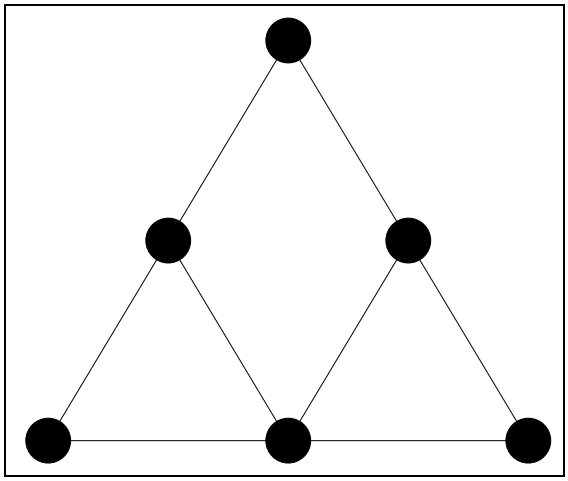
Koska kyseessä on iteratiivinen algoritmi[4], on toivottavaa, että jokaisen kierroksen ajoaika olisi kohtuullisen pieni. Täten graafin väritys on syytä suorittaa jollakin ahneella algoritmilla (ks. kappale 5.1). Tämä tietenkin ohjaa reitityksen ahneelle algoritmille sopivaksi. Mikäli lopullinen väritys haetaan jollakin raskaammalla algoritmilla, tämä ei ehkä ole hyvä asia. Toisaalta raskaampien väritysalgoritmien käyttö iteraatiossa saattaa hyvin olla mahdotonta niiden kompleksisuuden vuoksi.

[4] oikeastaan kyseessä on *local search*-menetelmä

Luku 4

Graafiteoriaa

Graafi eli verkko muodostuu solmuista (kärjistä) ja niitä yhdistävistä sivuista [4][5][3]. Kuvan 4.1 esimerkkigraafissa on kuusi solmua ja kahdeksan sivua. Graafit ovat luonnollinen tapa esittää asioita, joilla on riippuvuussuhteita. Solmuihin ja sivuihin voidaan sitoa tarpeen mukaan ylimääräistä tietoa. Esimerkiksi sivun pituutta kuvaava paino voisi olla tälläinen.



Kuva 4.1: Tasograafi

Moni käytännön ongelma on helppo esittää graafien avulla. Esimerkiksi kauppamatkustajan ongelma (traveling salesman problem, TSP) on tyypillinen graafiteoreettinen tehtävä. Kauppamatkustajan ongelmassa tehtävänä on löytää lyhin mahdollinen reitti, joka kulkee kaikkien annettujen kaupunkien kautta. Kaupungit ovat verkon solmuja ja solmut yhdistävillä sivuilla on painot, jotka kuvaavat kaupunkien välistä etäisyyttä. Myös tiedonsiirtoverkkojen käsittely ja kuvaus graafien avulla on helppoa.

4.1 Määritelmiä

Muodollisesti graafi G määritellään $G = (V, E)$, missä V on solmujen joukko ja E sivujen joukko.

Merkitään $\nu(G)$:llä graafin solmujen lukumäärää eli joukon V alkioiden lukumäärää. Vastaavasti $\epsilon(G)$ olkoon graafin sivujen lukumäärää eli joukon E alkioiden lukumäärä.

- Suunnatussa graafissa yhteyksillä on nimensä mukaisesti myös suunta. Jostain solmusta on yhteys toiseen solmuun mutta ei välttämättä toisinpäin.
- Graafi on yhtenäinen, mikäli kaikki $v_i, v_j \in V$ ovat yhdistettyjä ts. on olemassa reitti solmusta v_i solmuun v_j.
- Graafi on yksinkertainen, jos jokaisella solmuparilla on korkeintaan yksi sivu.
- Graafi on täydellinen, mikäli kaikki $v_i, v_j \in V$ ovat toistensa naapureita.
- Solmun v_i aste on siihen tulevien sivujen lukumäärä ja sitä merkitään d_i:llä.
- Graafi on tasograafi, mikäli siinä olevat yhteydet voidaan piirtää siten, että ne eivät leikkaa toisiaan.
- Graafin ositus eli partitio tarkoittaa graafin solmujen jakoa kahteen tai useampaan joukkoon.

Määritellään graafin solmujen asteille

$$\begin{aligned} \delta(G) &:= \min_{v \in V} \{ \text{ solmun } v \text{ aste } \}, \\ \Delta(G) &:= \max_{v \in V} \{ \text{ solmun } v \text{ aste } \}. \end{aligned}$$

Klikki (clique) on osajoukko graafin solmuja, jotka ovat kaikki toistensa naapureita, eli täydellinen osagraafi. Maksimaalinen klikki on suurin mahdollinen graafista löytyvä klikki. Täydellisen graafin maksimaalinen klikki on täten koko graafi.

Naapurimatriisi on $n \times n$-matriisi, joka ilmaisee graafin naapuruussuhteet. Naapurimatriisi määritellään usein tilanteen mukaan.

1. Binäärisen naapurimatriisin alkio $a_{i,j} = 1$, mikäli solmut v_i ja v_j ovat naapureita, ja muulloin nolla.

2. Mikäli kyseessä on painotettu graafi, voidaan yhteyksien painot myös tallettaa naapurimatriisiin:

$$d_{i,j} = \begin{cases} \text{solmuja } i \text{ ja } j \text{ yhdistävän sivun paino, tai} \\ \infty, \text{ mikäli solmut } i \text{ ja } j \text{ eivät ole keskenään naapureita} \end{cases}$$

4.2 Lyhimmän polun algoritmit

Eräs graafien yhteydessä monesti esille tuleva ongelma on tarve löytää lyhin mahdollinen reitti kahden solmun välillä. Oletetaan, että on annettu graafi $G = (V, E)$ ja jokaisella sivulla $e \in E$ on paino w_e. Paino voi kuvata esimerkiksi pisteiden välistä etäisyyttä tai linkin hitautta. Tehtävänä on siis löytää mahdollisimman edullinen reitti graafin kahden solmun välille. Määritellään lisäksi graafin naapurimatriisi siten, että sen alkio on ääretön, mikäli vastaavat solmut eivät ole toistensa naapureita (ks. kappale 4.1).

Lyhimmän polun algoritmeja on olemassa useita ja niillä on omat erikoispiirteensä. Toiset soveltuvat paremmin yhden polun hakemiseen ja toiset kaikkien pisteparien välisten polkujen hakemiseen. Seuraavassa käydään läpi kaksi ehkäpä yleisimmin käytettyä algoritmia.

4.2.1 Dijkstran lyhimmän polun algoritmi

Dijkstran algoritmi [21] on ehkä kuuluisin lyhimmän polun algoritmi. Se hakee lyhimmät polut[1] verkon yhdestä solmusta kaikkiin muihin solmuihin. Algoritmi olettaa, että kaikkien sivujen pituudet ovat positiivisia. Algoritmi on seuraava:

1. Annettuna on painotettu graafi $G = (V, E)$, lähtösolmu v ja naapurimatriisi, jonka alkiot ovat $d_{i,j}$.

2. Aseta $S = v$.

3. Aseta $D_v = 0$ ja $D_j = d_{j,v}$ kun $j \neq v$.

[1] lyhin polku kahden solmun välillä ei välttämättä ole yksikäsitteinen

4. Aseta $pred(j) = v$ kaikille solmuille j.
5. Valitse solmu $i \in V - S$, jonka etäisyys D_i on pienin.
6. Lisää solmu i S:ään. Mikäli $S = V$, niin lopetetaan algoritmin suoritus.
7. Jokaiselle $j \in V - S$ päivitä etäisyys: $D_j = \min\{D_j, d_{j,i} + D_i\}$. Mikäli solmun i kautta löytyi lyhyempi reitti, päivitä samalla solmun j edeltäjä: $pred(j) = i$.
8. Palaa kohtaan 5.

Jokaisella kierroksella joukkoon S lisätään yksi alkio, joten koko graafi on valmis $|V|$:n kierroksen jälkeen. Lyhimmän etäisyyden etsintä tarkoittaa ensimmäisellä kierroksella hakua $|V| - 1$:stä solmusta, seuraavalla $|V| - 2$:sta ja niin edelleen. Joten algoritmin kompleksisuus on $O(|V|^2)$. Taulukkoa $pred(i)$ lukemalla löydetään kysytty lyhin reitti mielivaltaisesta solmusta i pisteeseen v.

4.2.2 Floyd-Warshall algoritmi

Toinen hyvä lyhimmän polun algoritmi on Floyd-Warshall algoritmi [21]. Sen avulla löydetään lyhimmät reitit graafin kaikkien solmuparien välillä yhdellä suorituskerralla.

Menetelmän olennainen osa on etäisyysmatriisin D^k rekursiivinen päivitys. Yläindeksi k tarkoittaa algoritmin kierrosta. Matriisin D^k alkio (i, j) on lyhin etäisyys solmusta i solmuun j. Etäisyys on ääretön, mikäli solmujen välillä ei vielä ole löydetty reittiä. Jokaisella kierroksella k päivitetään etäisyysmatriisia D^k siten, että mikäli solmun k kautta löytyy nopeampi reitti kyseisen solmuparin välille, niin uusi reitti otetaan käyttöön.

Floyd-Marshall algoritmi on siis seuraava:

1. Aseta $D^0 = d_{i,j}$.
2. Aseta $k = 1$.
3. Aseta $D^k_{i,j} = \min\left\{D^{k-1}_{i,j}, D^{k-1}_{i,k} + D^{k-1}_{k,j}\right\}$ kaikilla i ja j.
4. Aseta $k = k + 1$.
5. Mikäli $k < |V|$ hyppää kohtaan 3.

Ylläolevasta algoritmista puuttuu kirjanpito varsinaisista reiteistä. Tämä on kuitenkin hyvin helppo lisätä. Jokaista matriisin solmuparia vastaa reitti $r_{i,j}$. Algoritmin alussa reitit asetetaan suoriksi yhteyksiksi: $r^0_{i,j} = \{i\ j\}$. Algoritmin kuluessa uudet reittitiedot päivitetään samalla, kun etäisyysmatriisin alkiot saavat uusia arvoja. Toisin sanoen, mikäli etäisyyksien päivityksessä $D^{k-1}_{i,j} > D^{k-1}_{i,k} + D^{k-1}_{k,j}$, niin asetetaan

$$r_{i,j} = \left\{r_{i,k}\ {}'+'\ r_{k,j}\right\},$$

eli otetaan solmujen i ja j välillä käyttöön reitti, joka muodostuu reiteistä r_{i_k} ja r_{k_j}.

Floyd-Marshall algoritmin kompleksisuus on $O(|V|^3)$, mikä on sama kuin jos Dijkstran algoritmi suoritettaisiin erikseen jokaiselle solmulle. Yleensä kuitenkin Floyd-Marshall algoritmi suoriutuu nopeammin kaikkien solmuparien lyhimmän reitin ongelmasta. Kovin harvoilla graafeilla (vähän yhteyksiä) Dijkstran algoritmi saattaa kuitenkin olla nopeampi.

4.3 Graafin solmujen väritys

Olkoon graafi G yhtenäinen ja yksinkertainen[2]. Mikäli annettu graafi ei ole yhtenäinen, jokainen yhtenäinen osagraafia voidaan käsitellä erikseen.

Annetaan graafin G solmuille värit siten, että naapurisolmut ovat aina erivärisiä. Graafin kromaattinen luku on pienin tarvittava värien lukumäärä, ja sitä merkitään χ:llä. Graafin solmujen optimaalinen väritys on NP-hankala ongelma, joten lähes aina joudutaan käyttämään heuristisia algoritmeja kohtuullisen värityksen löytämiseen.

[2] Aallonpituusallokoinnissa näin on käytännössä aina

Graafin G kromaattiselle luvulle pätevät seuraavat epäyhtälöt [4][5][9].

$$\chi(G) \leq \Delta + 1, \tag{4.1}$$

$$\chi(G) \leq 1 + \lambda_1, \tag{4.2}$$

$$\lambda_1 \text{ on graafin } G \text{ naapurimatriisin suurin ominaisarvo}, \tag{4.3}$$

$$\chi(G) \leq \max_i \min\{d_i + 1, i\}, \tag{4.4}$$

$$\text{missä } d_1 \geq d_2 \geq \ldots \geq d_\nu \text{ ovat graafin } G \text{ solmujen asteet}, \tag{4.5}$$

$$\chi(G) \leq 1 + \max\{\delta(H) | H \in G\}, \tag{4.6}$$

$$\chi(G) \geq \frac{\nu^2}{\nu^2 - 2\epsilon}. \tag{4.7}$$

Kaava (4.6) on Szekeresin ja Wilfin tulos vuodelta 1968. Sen avulla nähdään helposti, että mikä tahansa tasograafi voidaan värittää 6:lla värillä. Kuuluisan neliväriteoreeman (käsitellään kappaleessa 4.3.2) mukaan kaikki tasograafit voidaan värittää neljällä värillä.

Esitetyillä epäyhtälöillä ei tosin ole suoranaisia sovelluksia: niiden antamat ala- ja ylärajat ovat usein heikkoja tai sitten itse epäyhtälön arvon laskeminen on kovin työlästä. Sen sijaan käytännön ongelmissa joudutaan turvautumaan erilaisiin heuristisiin ratkaisuihin.

Todistetaan kaava (4.4) (muiden kaavojen todistukset löytyvät mainituista lähteistä).

Väite:

$$\chi \leq \max_i \min\{d_i + 1, i\}.$$

Todistus:

1. Olkoon $p = \max_i\{i \mid d_i \geq i - 1\}$
2. Annetaan solmuille $v_1, \ldots, v_p$ omat värit (p väriä siis)
3. Nyt selvästi pätee $d_\nu \leq \ldots \leq d_{p+1} < p$. Toisin sanoen kaikilla jäljellä olevilla solmuilla on vähemmän kuin p naapuria, joten ne voidaan värittää jo olemassa olevilla p:llä värillä (mikäli $p = \nu$ on kyseessä täydellinen graafi ja väite on triviaali).
4. Saadaan graafi väritetyksi p:llä värillä.
5. Nyt $p = \max_i \min\{d_i + 1, i\}$ sillä
   - kun $i < p$ on $\min\{d_i + 1, i\} = i \leq p$,

- kun $i = p$ on $\min\{d_i + 1, i\} = i = p$,
- kun $i > p$ on $\min\{d_i + 1, i\} = d_i + 1 \leq d_p + 1 \leq p$.

Joten väite on tosi.

4.3.1 Eräitä graafin solmujen väritysongelmia

Graafin solmujen väritysongelma ei ole vain matemaattinen hauska ongelma, vaan sillä on paljon tärkeitä reaalimaailman sovelluksia. Ongelman muotoilu ja esittäminen graafien avulla antaa myös paremman käsityksen ongelman luonteesta ja siinä olevista riippuvuussuhteita. Alla on joukko esimerkkejä joissa (osa)ongelmana on graafin solmujen väritys.

- **Aikataulujen suunnittelu**, esimerkiksi oppilaitoksen kurssien lukujärjestykset. Opettajalla ei voi olla kahta tuntia samaan aikaan, luokassa voi olla vain yksi tunti kerrallaan. Samoin oppilaat eivät voi olla kahdessa paikassa samaan aikaan. Opettajien ja oppilaiden hyppytunnit pitäisi samalla minimoida.
- **Taajuusallokointi** matkapuhelinjärjestelmissä. Lähellä toisiaan olevien tukiasemien pitää käyttää eri taajuuksia, etteivät ne häiritse toisiaan. Mutta kun etäisyys tukiasemien välillä kasvaa riittävästi, sama taajuus voidaan käyttää uudelleen. Taajuusallokoinnissa solmujen naapuruussuhteet eivät aina ole yhtä selvät kuin ideaalisessa tapauksessa.
- **Aallonpituusallokaatio** WDM-verkoissa. Samassa optisessa kuidussa kulkevien yhteyksien tulee aina käyttää eri aallonpituuksia. Yhteys tarkoittaa väritysgraafissa solmua, ja naapurisolmuja ovat sellaiset yhteydet, jotka kulkevat jossain vaiheessa samassa optisessa kuidussa.
- **Rekisteri-allokointi** ohjelmointikielten kääntäjissä. Prosessoreissa on äärellinen määrä sisäisiä rekistereitä ja ne tulisi käyttää mahdollisimman tehokkaasti hyväksi. Mikäli rekisterit loppuvat kesken, jonkun rekisterin arvo pitää väliaikaisesti tallettaa pinoon, mikä on “hidas” operaatio. Kääntäjän tulee pyrkiä minimoimaan pinon käyttö eli allokoida rekisterien käyttö siten, että ohjelman suorituksen aikana syntyy mahdollisimman vähän tilanteita, joissa rekisterin arvo joudutaan joko tallettamaan tai lataamaan muistista.

Eri ongelmilla on luonnollisesti omia erikoispiirteitään, jotka näkyvät sitten väritysgraafin ominaisuuksissa.

4.3.2 Neliväriteoreema

Yksi kuuluisimpia graafiteoreettisia tuloksia on neliväriteoreema. Neliväriteoreeman mukaan mielivaltainen tasokartta voidaan värittää neljällä eri värillä siten, että naapurimaat ovat aina erivärisiä. Jotta maat luokiteltaisiin naapureiksi, niillä pitää olla yhteistä rajaa enemmän kuin piste.

Neliväriteoreeman historia on pitkä. Vuonna 1852 pian valmistumisensa jälkeen Francis Guthrien kirjoitti kirjeen veljelleen, joka oli edelleen kuuluisan matemaatikon De Morganin oppilaana. Kirjeessään Francis esitti, että jokainen tasokartta näyttäisi olevan väritettävissä neljällä värillä, ja kysyi onko olemassa jokin tapa todistaa hypoteesi. Frederick esitti ongelman edelleen De Morganille. Konjektuuri pysyi todistamattomana yli sata vuotta, jona aikana nähtiin monta yritystä todistaa väite todeksi. Asiaa pohtivat mm. sellaiset henkilöt kuten Cayley, Kempe ja Heawood onnistumatta todistamaan väitettä.

Vasta vuonna 1976 Kenneth Appel ja Wolfgang Haken [10] pystyivät osoittamaan konjektuurin todeksi. Todistuksessa käytettiin hyväksi tietokoneen laskentakapasiteettia ja aikaisemmin todistettuja tuloksia. Tietokoneella käytiin läpi suuri määrä alkeistapauksia. Tämä herättikin paljon kritiikkiä matemaatikoiden keskuudessa, joista osa oli sitä mieltä että kyseessä ei ole “hyväksyttävä“ todistus.

Neliväriteoreema voidaan myös muotoilla graafin solmujen väritysongelmana, jolloin sen sisältönä on, että kaikki tasograafit voidaan värittää neljällä värillä. Sallittua väritystä neljällä värillä ei kuitenkaan aina ole helppoa löytää. Sen todistaminen, että viisi väriä riittää värittämään mielivaltaisen graafin on jo helpompaa [9].

4.3.3 Graafin sivujen väritysongelma

Joskus ongelma voidaan muotoilla graafin sivujen väritysongelmaksi, jossa vaaditaan, että solmuun tulevat sivut ovat aina erivärisiä. Tämä ongelma voidaan kuitenkin esittää helposti myös solmujen väritysongelmana. Jokaista alkuperäisen graafin sivua vastaa yksi

solmu uudessa graafissa. Nyt ne uuden graafin solmut yhdistetään toisiinsa, joita vastaavat alkuperäisen graafin sivujen päätepisteet ovat samat.

Kaikki solmunväritysongelmat eivät kuitenkaan ole muunnettavissa sivujen väritysongelmaksi, joten solmujen väritysongelma on yleisempi tapaus. Aallonpituusallokoinnissa syntyvät väritysgraafit eivät esimerkiksi yleensä ole muunnettavissa sivujenväritysongelmaksi.

Luku 5

Heuristisia graafin solmujen väritysalgoritmeja

Kuten edellä kävi ilmi, graafin solmujen väritystehtävä on NP-hankala ongelma. Tässä kappaleessa käydään läpi joukko heuristisia algoritmeja, joilla väritysongelma voidaan ratkaista. Ratkaisut eivät luonnollisesti yleisesti ole optimaalisia vaan lähinnä “hyviä“. Algoritmien suoritusajoissa on suuria eroja, jotkut suoriutuvat väritystehtävästä hyvinkin nopeasti, kun taas monimutkaisemmilta algoritmeilta aikaa kuluu tuntuvasti enemmän. Vastaavasti värityksen laadukkuudessa alkaa olla eroja monimutkaisempien algoritmien hyväksi graafien ollessa kohtalaisen kokoisia (esim. suurempia kuin 100 solmua).

5.1 Ahne algoritmi

Toista ääripäätä graafin solmujen väritysalgoritmeissa edustavat ahneet algoritmit [7]. Ahneille algoritmeille on tyypillistä hyvin nopea suoritusaika. Yleensä kohtalainen väritys saadaan seuraavalla ahneella algoritmilla:

1. Järjestetään solmut asteidensa mukaiseen järjestykseen: $d_1 \geq \ldots \geq d_\nu$.
2. Aloitetaan väritys solmusta 1 ja annetaan sille väri 1.
3. Loput solmut käydään läpi asteiden mukaisessa järjestyksessä.
4. Annetaan aina seuraavalla solmulle ensimmäinen vapaa (sallittu) väri tai uusi väri,

mikäli vapaita ei ole.

Tällä algoritmilla väritetyt graafit toteuttavat aina epäyhtälön $\hat{\chi} \leq \max_i \min\{d_i + 1, i\}$, ks. (4.4).

Tästä algoritmista löytyy myös muita versioita. Kaikille näille on kuitenkin ominaista se, että sopivasti valituilla graafeilla ahne algoritmi päätyy huonohkoon lopputulokseen.

5.1.1 DSATUR

Eräs variaatio yksinkertaisesta ahneesta algoritmista on Daniel Brélazin DSATUR [8]. Erona tavalliseen ahneeseen algoritmiin on se, että väritysjärjestys on dynaaminen. Järjestystä päivitetään joka kierroksella siten, että seuraavaksi väritettäväksi solmuksi valitaan aina se, jolla on vähiten vapaita värejä jäljellä sillä hetkellä. Intuitiivisesti tämä kuulostaa aivan järkevältä. Mikäli väritettävä graafi voidaan jakaa kahteen sellaiseen osajoukkoon, missä samaan osajoukkoon kuuluvat solmut eivät ole naapureita[1], antaa DSATUR-menetelmä sille optimaalisen värityksen.

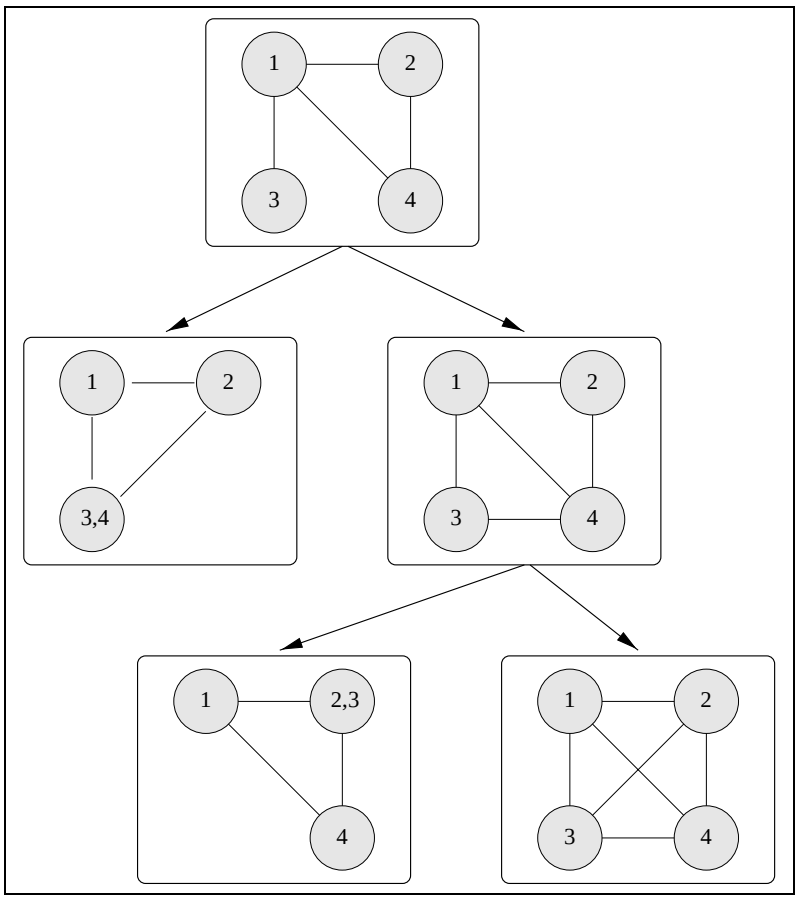
Satunnaisilla graafeilla DSATUR tosin tuntuu toimivan keskimäärin hieman huonommin kuin tavallinen ahne algoritmi. Algoritmien suoritusajoissa ei luonnollisesti ole vakiotekijää suurempaa eroa.

5.2 Täydellinen haku

Annetun graafin solmut voidaan aina värittää optimaalisesti seuraavalla algoritmilla. Algoritmi käy järjestelmällisesti läpi kaikki mahdolliset väritykset (=täydellinen haku). Jokaisella askeleella graafista muodostetaan kaksi uutta graafia, kunnes päädytään täydelliseen graafiin, jonka optimaalinen väritys on triviaali (kuva 5.1).

Algoritmi on hyvin yksinkertainen. Mikäli graafi ei ole täydellinen, niin on olemassa ainakin yksi pari solmuja (i, j), jotka eivät ole toistensa naapureita. Nyt nämä kaksi solmua voidaan värittää:

[1] engl. bipartite graph



Kuva 5.1: Graafin solmujen väritys täydellisellä haulla.

- samalla värillä
- eri väreillä

Nämä kaksi alitapausta voidaan käsitellä erikseen:

- **Väritetään solmut samalla värillä.** Koska solmujen väri on sama, ne voidaan yhdistää. Kummankin solmun vanhoista naapureista tulee uuden solmun naapureita.
- **Väritetään solmut eri väreillä.** Koska solmut väritetään eri väreillä, niiden väliin voidaan piirtää sivu, jolloin niistä tulee naapureita.

Kummassakin tapauksessa uusi graafi on askeleen lähempänä täydellistä graafia. Kun tätä prosessia jatketaan rekursiivisesti, päädytään lopulta tilanteeseen, jossa saadun puun lehdet ovat täydellisiä graafeja. Saadun puun lehdistä poimitaan sitten se, jossa solmuja on

jäljellä vähiten. Optimaalinen väritys ei välttämättä ole yksikäsitteinen kuten kuvasta 5.1 nähdään.

Täydelliselle graafille pätee tunnetusti

$$\frac{v^2 - v}{2} = e.$$

Määritellään etäisyydeksi täydellisestä graafista funktio

$$D_n = \frac{v_n^2 - v_n}{2} - e_n,$$

joka on selvästi ei-negatiivinen ja nolla ainoastaan, kun graafi on täydellinen.

Yhdistettäessä solmut (i, j) on uudessa graafissa yksi solmu vähemmän. Samalla graafin sivujen lukumäärä pienenee tapauksesta riippuen jollakin luvulla väliltä $0 \ldots \min\{d_i, d_j\}$.

Olkoon d_i solmun i aste. Selvästi kaikille i pätee $d_i \leq v - 1$. Koska solmuja i ja j ei ole vielä yhdistetty, on niistä ainakin toisen asteen oltava pienempi kuin $v - 2$. Joten yhdistettäessä solmut saadaan solmujen ja sivujen lukumäärien muutoksiksi

$$\begin{aligned} v_{n+1} &= v_n - 1, \\ e_{n+1} &= e_n - x \text{ missä } 0 \leq x \leq v - 2\ . \end{aligned}$$

Joten etäisyysfunktiolle pätee

$$D_{n+1} \leq D_n - 1.$$

Mikäli solmut väritetään eri väreillä, on uudessa graafissa sama määrä solmuja ja yksi sivu enemmän kuin alkuperäisessä:

$$\begin{aligned} v_{n+1} &= v_n, \\ e_{n+1} &= e_n - 1, \end{aligned}$$

jolloin etäisyysfunktion uudeksi arvoksi saadaan

$$D_{n+1} = D_n - 1.$$

Siten puun korkeus on D_0, ja tarvittavien iteraatioaskelten lukumäärä on pienempi kuin $2^{D_0} - 1$.

Käytännössä suurilla ja harvoilla graafeilla (D_0 on suuri) algoritmin suoritus onkin mahdotonta, koska suoritusaika kasvaa nopeasti järjettömän suureksi[2].

Koska jokaisessa vaiheessa kyseisen graafin väritystehtävä jaetaan kahdeksi aliongelmaksi, ei aliongelmia tarvitse tutkia, mikäli tiedetään, että kyseisen graafin väritys ei voi onnistua pienemmällä määrällä värejä kuin paras tähän mennässä löydetty väritys sisältää. Tässä voidaan käyttää hyväksi kaavaa (4.7), joka antaa alarajan tarvittavien värien lukumäärälle. Alarajan mukaanotto pudotti algoritmin ajoajan noin viidennekseen, kun graafissa oli solmuja 13 kappaletta. Kun puu käydään läpi siten, että jokaisessa jaossa tutkitaan ensiksi tilanne, jossa solmuille on annettu sama väri, on luultavaa, että saadaan kohtalainen väritys aikaan hyvin nopeasti. Tällöin saadaan jonkinlainen arvio graafin kromaattisesta luvusta ja todennäköisesti monen turhan aliongelman tutkiminen jää pois.

Kuvattu algoritmi on helposti rinnakkaistuva, koska jokaisella askeleella puolitetaan ongelma kahteen täysin **riippumattomaan** osaongelmaan. Täten algoritmin suoritus voidaan helposti jakaa mielivaltaiselle määrälle koneita. Tosin ongelman luonteesta johtuen tarvittavien koneiden lukumäärä kasvaa nopeasti graafin asteen funktiona.

5.3 Karsittu täydellinen haku

Kuten kohdassa 5.2 kävi ilmi, aivan pieniä graafeja lukuunottamatta kaikkia mahdollisia värityksiä ei voi käydä järjestelmällisesti läpi. Nyt yksi mahdollisuus on pyrkiä karsimaan täydellistä hakupuuta, sillä optimaalinen värityshän on periaatteessa polku juuresta yhteen lehteen. Karsinta voidaan hoitaa seuraavalla tavalla:

- Lähdetään rakentamaan samalla tavalla puuta kuin järjestelmällisessä haussa (kuva 5.1).
- Jokaisella tasolla suoritetaan karsinta pudottaen pois sellaiset haarat, joiden arvioidaan johtavan huonoon lopputulokseen, ts. estimoidaan annettujen graafien kromaattista lukua.
- Karsinnassa estimaattorina graafin kromaattiselle luvulle voidaan hyvin käyttää ahnetta algoritmia, sillä sen suoritusaika on huomattavan lyhyt. Toisin sanoen värite-

[2] Kyseessähän oli NP-hankala ongelma

tään jäljellä olevat aligraafit ahneella algoritmilla ja pudotetaan pois huonoimmat.

Jokaisesta ei-täydellisestä hakupuun solmusta syntyy jokaisella kierroksella aina kaksi uutta solmua. Nyt algoritmin ainut parametri on käytetyn hyvyysmitan (estimaattorin) lisäksi säilytettävien ehdokkaiden lukumäärä per kierros. Säilytettävien ehdokkaiden lukumäärä voisi jopa muuttua algoritmin ajon aikana.

Tämä algoritmi kuitenkin osoittautui graafin solmujen lukumäärän kasvaessa kehnoksi. Järjestyksellä, jolla solmupareja käsiteltiin, on vaikutus algoritmin toimintaan. Ahneen algoritmin väritysjärjestys saattaa “vinoutua” yhdisteltäessä solmuja, joiden aste on pieni. Lisäksi jokaisella kierroksella suoritettava väritys ahneella algoritmilla alkaa olla suuritöinen kun graafin koko kasvaa.

Algoritmi luonnollisesti löytää aina optimin, kunhan vain säilytetään riittävän monta ehdokasta joka kierroksella. Kokeissa tämä säilytettävien ehdokkaiden lukumäärä oli 20, mikä alkoi olla riittämätön, kun graafin koko ylitti sata solmua.

Kuten täydellinen haku myös karsittu versio on helposti rinnakkaistettavissa useammalla eri koneella ajettavaksi algoritmiksi.

5.4 Geneettinen algoritmi

Geneettinen algoritmi (GA) voidaan ajatella tavaksi suorittaa “älykkäästi” satunnaishakua. Algoritmi pyrkii käyttämään hyväksi tietämystä, jonka se on kerännyt valitessaan satunnaisesti seuraavaksi kokeiltavia ratkaisuvaihtoehtoja. Tällöin toivottavasti ei tutkittaisi turhaan sellaisia ratkaisuavaruuden alueita, joissa hyvyysfunktio saa pieniä arvoja.

Geneettisten algoritmien ideana on pyrkiä mallintamaan evoluutiota. Jokaisella kierroksella suositaan lupaavia ratkaisuvaihtoehtoja ja huonot vaihtoehdot kuolevat. Ratkaistavan kombinatorisen ongelman parametreista muodostetaan vektori eli kromosomi. Analogia genetiikan kanssa on erittäin selvä. Kromosomit sisältävät tietyn määrän geenejä (vektorin alkiot). Kun kromosomin geenit tunnetaan, määräävät nämä yksilön genotyypin[3] ja fenotyypin[4]. Geneettisten algoritmien yhteydessä genotyypin voidaan ymmärtää olevan

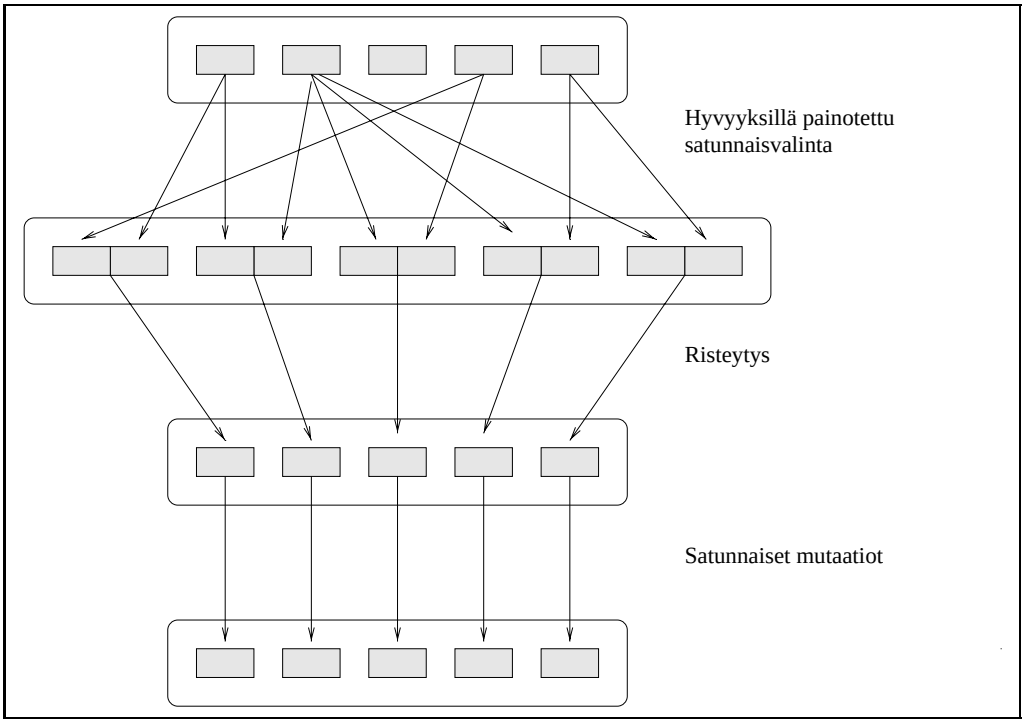
[3] “geenien järjestys”
[4] “fyysinen ilmenemismuoto”

koodattu jono ja fenotyyppi vastaa purettua jonoa eli ratkaisua alkuperäiseen ongelmaan.

Jokaisella iteraatiokierroksella vanhasta populaatiosta muodostetaan uusi. Geneettisten algoritmien yhteydessä puhutaankin sukupolvista. GA:n tärkeimmät operaatiot ovat

- **Risteytys**, kaksi kromosomia yhdistetään (crossover) jollakin tavalla toivoen, että hyvät ominaisuudet periytyisivät jossain määrin
- **Mutaatio**, kromosomin geenejä muokataan hieman satunnaisesti.

Risteytyksessä suositaan sellaisia yksilöitä, joiden hyvyysfunktio on suuri. Risteytyksen jälkeen satunnaisille yksilöille suoritetaan mutaatio-operaatio.



Kuva 5.2: Geneettisen algoritmin yksi iteraatiokierros.

Ensimmäinen tehtävä, kun geneettisiä algoritmeja halutaan soveltaa, on alkuperäisen tehtävän parametrien koodaaminen jollakin algoritmin kannalta edullisella tavalla geeneiksi. Monesti geenien alkiot ovat olleet binääriarvoisia, jolloin *risteytys*- ja *mutataatio*-operaatiot ovat helppoja toteuttaa.

Yksinkertainen *risteytys*-operaatio on esimerkiksi seuraavanlainen:

```
P1  1 0 1 1 0 1 1 0          O1  1 0 1 0 0 1 0 0
        |              ==>
P2  1 1 0 0 0 1 0 0          O2  1 1 0 1 0 1 1 0
```

Ensiksi katkaisukohta arvotaan satunnaisesti. Tämän jälkeen uuden jälkeläisten alkuosaksi kopioidaan ensimmäisen vanhemman alkuosa katkaisukohtaan asti ja loppuosa toiselta vanhemmalta. Toinen jälkeläinen $O2$ saadaan vaihtamalla vanhempien järjestys kopioinnissa keskenään.

5.4.1 Solmujen väritys geneettisellä algoritmilla

Geneettistä algoritmia voidaan käyttää hyväksi etsittäessä hyvää väritysjärjestystä ahneelle algoritmille [1]. Hyvän järjestyksen järjestelmällinen hakeminen on ylivoimainen tehtävä, sillä mahdollisia väritysjärjestyksiä on solmujen lukumäärän kertoman verran. Järjestyksiä voisi generoida myös satunnaisesti, mutta seuraava geneettinen algoritmi on todettu käytännössä hyväksi vaihtoehdoksi.

Tässä sovelluksessa geenit eivät ole binääriarvoisia, vaan kromosomi esittää jotain solmujenväritysjärjestystä. Esimerkiksi mikäli väritettävässä graafissa olisi solmut $1 \ldots 8$, niin $\{4\ 3\ 7\ 1\ 5\ 6\ 8\ 2\}$ olisi kelvollinen kromosomi. Selvästi jokainen solmu tulee esiintyä tasan yhden kerran kromosomeissa. Johtuen tästä koodauksesta, edellä esitetty risteytysoperaatio ei enää kelpaa. Algoritmi on seuraava:

1. Generoidaan N kappaletta satunnaisia järjestyksiä $P_i(0)$ ja väritetään graafit niillä.
2. Generoidaan uusi sukupolvi $P(n+1)$ yhdistelemällä nykyisiä järjestyksiä (risteytys) ja lisäämällä hieman satunnaisuutta (mutaatio).
3. Yhdistettävien parien valinnassa suositaan hyviä järjestyksiä (luonnollinen valinta). Näin toivotaan, että hyvät järjestykset dominoisivat ja heikot kuolisivat pois.
4. Uudelle sukupolven yksilöille lasketaan hyvyysluku (=tarvittavien värien lukumäärä), ts. väritetään graafi ko. järjestyksillä.
5. Mikäli toivottua lopputulosta ei ole vielä löydetty, hypätään kohtaan 1.

Vapaita parametreja algoritmissa ovat siis

- populaation koko N
- risteytys-operaatio toteutus
- mutaatioiden toteutus ja todennäköisyys
- iteroinnin lopetushetki

Yleensä jopa 30 yksilön populaatio on riittävä. Crossover-operaatioksi valittiin seuraava:

1. Olkoon vanhemmat A ja B (vektoreita). A on valittu kaikista vanhemmista suosien hyvän värityksen omaavia ehdokkaita. B on valittu satunnaisesti koko populaatiosta. Kummankin vektorin pituus on siis N.
2. Asetetaan $i_A = i_B = 1$ ja generoitava jälkeläinen C asetetaan tyhjäksi.
3. Valitaan todennäköisyydellä 0.75 vektori A ja todennäköisyydellä 0.25 vektori B.
4. Valitun vektorin seuraava alkio (i_A tai i_B) lisätään generoitavaan vektoriin C, mikäli sitä ei siellä vielä ole.
5. Tätä toistetaan kunnes jälkeläinen C sisältää kaikki alkiot $1 \ldots N$.

Mutaatio-operaatiossa yksinkertaisesti vaihdetaan kahden satunnaisen alkion paikka.

Geneettisen algoritmin käyttö sopivan järjestyksen haussa toimii varsin hyvin. Suhteellisen suuretkaan ($\sim$400 solmua) graafit eivät ole ylivoimaisia väritettäviä ja saatu lopputulos on melko hyvä verrattuna tavalliseen ahneeseen algoritmiin. Algoritmin vaatima ajoaika taas on selvästi pienempi kuin simuloidulla jäähdytyksellä, mikäli pyritään vastaaviin tuloksiin tarvittavien värien osalta.

Huomattavaa on, että yleensä jo ensimmäisessä populaatiossa on varsin hyviä yksilöitä ja iteroinnin aikana saavutettu parannus on vain muutamia värejä vähemmän. Tämä kertonee siitä, että käytetty koodaus saattaa olla liian “tiukka”.

Menetelmä perustuu oleellisesti hyvän väritysjärjestyksen hakemiseen ahneelle algoritmille. C-kielellä toteutetun algoritmin suoritusajasta menee noin 75% graafin värityksen

laskemiseen eri permutaatioilla. Mielenkiintoista oli tietää, millaisilla graafeilla on olemassa väritysjärjestys, joka antaisi optimaalisen värityksen.

5.5 Simuloitu jäähdytys

Simuloitu jäähdytys (simulated annealing, SA) on hyvin paljon käytetty menetelmä, kun pitää ratkaista hankalia kombinatorisia tehtäviä [1][2]. Ensimmäiset kokeilut simuloidun jäähdytyksen menetelmällä ajoittuvat 1980-luvun alkuun, jolloin se sai osakseen suurta huomiota. Suosion syynä oli menetelmän yksinkertaisuus ja helppo sovellettavuus erilaisiin optimointiongelmiin.

Ideana simuloidussa jäähdytyksessä on nimen mukaisesti simuloida kappaleen jäähtymistä. Aluksi kappale on hyvin suuressa lämpötilassa ja systeemin osat voivat helposti siirtyä tiloihin, joissa systeemin kokonaisenergia kasvaa. Pikkuhiljaa systeemin lämpötilaa pudotetaan ja epäedulliset siirtymät tulevat aina vain harvinaisemmiksi. Lopulta systeemi on "jäähdytetty" ja sen toivotaan olevan minimienergiatilassa. Kyseessä on siis eräänlainen naapurustohaku, jossa siirrytään aina minimiä kohti sillä poikkeuksella, että tietyllä todennäköisyydellä hyväksytään myös muutokset, jotka kasvattavat kustannusfunktion arvoa. Tällä pyritään välttämään tilanteet, joissa juututaan paikalliseen minimiin. Yleensä kustannusfunktiota kasvattava muutos hyväksytään todennäköisyydellä

$$P(\Delta E) = e^{-\Delta E/T},$$

missä T on systeemin "lämpötila". Kun T lähestyy nollaa, tulevat kustannusfunktiota kasvattavat siirrot aina vain epätodennäköisemmiksi. Todennäköisyysfunktion muoto on peräisin tilastollisesta mekaniikasta.

Toinen olennaisesti menetelmän hyvyyteen vaikuttava tekijä on tapa, jolla systeemin jäähdytys suoritetaan. Jäähdytys ei saa tapahtua liian nopeasti mutta toisaalta aikaakaan ei käytännön sovelluksia ratkaistaessa ole rajattomasti. Joten joudutaan tekemään kompromissi ratkaisun laadun ja ratkaisuun käytetyn ajan välillä. Eräs tapa suorittaa jäähdytys on seuraava. Lämpötila pidetään vakiona, kunnes

1. on suoritettu N kappaletta muutoksia systeemiin,
2. on yritetty M kappaletta muutosehdotuksia.

Tämän jälkeen systeemin lämpötilaa lasketaan jonkin verran. Yleensä käytettyjä tapoja ovat kaksi seuraavaa:

$$\begin{aligned} T &\leftarrow \alpha T, \\ T &\leftarrow \frac{T}{1+\beta T}. \end{aligned}$$

Yhtälössä esiintyvä vakio α on tyypillisesti hieman alle yksi ($0.9 \ldots 0.99$) ja vastaavasti β:n arvo on hyvin lähellä nollaa.

Simuloidun jäähdytyksen menetelmä löytää teoriassa aina globaalin minimin, mikäli jäähdytys tehdään riittävän hitaasti. Tämä ei käytännössä taas ole mahdollista, joten hankalien ongelmien yhteydessä ei yleensä saavuteta optimia, vaan joku “hyvä” ratkaisu. Yleisesti ottaen menetelmä on hyvä mutta monesti hidas. Hyvänä puolena sille voidaan laskea sen erittäin laaja sovellettavuus. Esimerkiksi klassinen kauppamatkustajan ongelma ja graafin kahtia jako ovat helposti formuloitavissa simuloidulle jäähdytykselle sopivissa muodoissa.

5.5.1 Värien allokointi simuloidun jäähdytyksen avulla

Värien allokointiongelma ratkaistiin simuloidun jäähdytyksen menetelmällä seuraavalla tavalla:

1. Annetaan jokaiselle solmulle oma väri (huono mutta sallittu allokaatio).
2. Asetetaan systeemin alkulämpötilaksi $T = T_0$ sopiva (riittävän suuri) arvo.
3. Systeemin energiaksi valitaan käytettyjen värien lukumäärä (alussa solmujen lukumäärä).
4. Valitaan satunnainen solmu ja sille uusi väri. Uusi väri on joku vanhoista tai kokonaan uusi väri.
5. Mikäli tehtävä muutos johtaisi laittomaan lopputilaan, palataan kohtaan 4.
6. Lasketaan systeemiin energian muutos ΔE.
7. Mikäli $\Delta E < 0$ tai $e^{-\Delta E/T} > \mathrm{rnd}(0,1)$ hyväksytään ehdotettu muutos. Funktio $\mathrm{rnd}(0,1)$ on välille $(0,1)$ tasanjakautunut satunnaismuuttuja.
8. Mikäli systeemiin on tehty M muutosta tai N muutosehdotusta, niin $T = \alpha \cdot T$, missä α on sopiva hieman ykköstä pienempi vakio.
9. Mikäli T on suurempi kuin T_1, niin palataan kohtaan 4.

Vapaita parametreja algoritmissa ovat siis α, T_0, T_1, N ja M. Systeemin alkulämpötilan on oltava riittävän suuri, jotta mikä tahansa muutos voi tapahtua. Tässä tapauksessa, kun systeemin energiaksi on valittu käytettyjen värien lukumäärä, energian muutos voi olla vain -1, 0 tai 1. Siten esimerkiksi $T_0 = 1$ kelpaa hyvin alkulämpötilaksi, jolloin epäedulliset muutokset hyväksytään aluksi noin 37% todennäköisyydellä.

Systeemin energiafunktion määritteleminen ei ole yksikäsitteistä. Luonnollinen valinta "$E =$tarvittavien värien lukumäärä" johtaa energiafunktioon, joka on vakio lähes kaikkialla. Tämä ei ole edullista algoritmin suppenemisen kannalta, koska jokaisella iteraatiokierroksella päätökset perustuvat energiafunktion muutokseen nykyisen ratkaisun ympäristössä.

Vaihtoehtoinen muoto energiafunktiolle on seuraavanlainen [1].

$$E = -\sum_i |C_i| + \sum_i 2|C_i||E_i|,$$

missä $|C_i|$ on partitiossa i olevien solmujen lukumäärä ja $|E_i|$ siinä olevien sivujen lukumäärä. Jälkimmäinen termi on sakkotermi, jonka tulisi jäähdytyksen aikana saavuttaa arvo nolla. Muuten kyseessä on väritys, jossa on aallonpituuskonflikti. Energiafunktio suosii siis suuria osajoukkoja, joten todennäköisyys, että päästään ylimääräisistä väreistä eroon, kasvaa.

5.6 Tabu-haku

Tabu-haku on varsin uusi heuristinen menetelmä kombinatoristen ongelmien ratkaisemiseksi [1][2]. Toisin kuin simuloidulla jäähdytyksellä, tabu-haulla ei ole suoranaista analogiaa reaalimaailmaan. Menetelmässä jokaisella askeleella siirrytään nykyisestä pisteestä siihen naapuriin, jossa kustannusfunktio saa pienimmän arvon. Naapurustosta suljetaan pois “tabut siirtymät”, jotka tyypillisesti ovat aikaisempien siirroksien käänteissiirtymiä. Tällä pyritään välttämään lokaaliin minimiin kiinnijääminen. Tabu-listaan lisätään aina viimeisin siirto ja vanhin poistetaan. Tabu-lista toimii tavallaan menetelmän muistina siitä, mitä on jo kokeiltu. Mikäli siirtymä johtaisi huomattavaan parannukseen, se hyväksytään vaikka se olisikin luokiteltu tabuksi.

Iterointia jatketaan, kunnes hyväksyttävä kustannusfunktion arvo on saavutettu tai tietty määrä iteraatioita on suoritettu.

Tässäkin menetelmässä on useita vapaita parametreja, joiden sopivalla valinnalla voidaan vaikuttaa paljon menetelmän hyvyyteen. Nykyisen ratkaisun[5] naapuroston tulisi olla sellainen, että kustannusfunktio olisi mahdollisimman “jatkuva”. Jos taas naapuruston koko on todella suuri, ei koko naapurostoa ehkä kannata käydä läpi vaan voidaan tyytyä ensimmäiseen hyvään siirtymään. Tabu-listan pituus ja tabun siirtymän määrittelyt voivat myös olla hankalia.

Tabu-haku on hyvin läheistä sukua simuloidun jäähdytyksen menetelmälle. Molemmat ovat naapurustohakuja, joissa tietyillä lisäehdoilla pyritään välttämään lokaalit minimit.

[5] piste ratkaisu-avaruudessa

5.6.1 Graafin solmujen väritys tabu-haun menetelmällä

Tabu-hakua on sovellettu menestyksekkäästi myös graafin solmujen väritysongelmaan [6]. Tavoitteena on löytää jokin k-väritys. Olkoon $s = (V_1, \ldots, V_k)$ joku graafin G partitio. Jokainen partitio edustaa yhtä väriä. Määritellään kustannusfunktioksi

$$f(s) = \sum_i |E(V_i)|,$$

eli kustannusfunktion arvo on yhtäsuuri kuin väritysrikkeiden lukumäärä. Joten kun $f(s) = 0$, on löydetty laillinen partitio eli graafi on tullut väritetyksi k:lla värillä.

Määritellään lisäksi partition s naapureiksi sellaiset partitiot, joissa yksi solmu on siirretty jostain partitiosta toiseen. Toisin sanoen jollekin solmulle annetaan uusi väri. Algoritmi on nyt seuraava:

1. Olkoon annettuna
   - väritettävä graafi G
   - tavoiteltu värien lukumäärä k
   - tabu-listan pituus $|T|$
   - naapurien lukumäärä rep
   - iteraatioiden maksimilukumäärä $nbmax$
2. Keksi alkutilanne $s = (V_1, \ldots, V_2)$.
3. Aseta $nbiter = 0$ (kierroslaskuri).
4. Keksi satunnainen tabu-lista T.
5. Niin kauan kun $f(s) > 0$ ja $nbiter < nbmax$
   (a) Keksi $nrep$ naapuria s_i joille pätee $s \to s_i \notin T$ tai $f(s_i) \leq Af(s)$.
   (b) Valitaan näistä paras (tai ensimmäinen jolle $f(s_i) < f(s)$).
   (c) Päivitä tabu-lista T.
   (d) Aseta $s = s'$ ja $nbiter = nbiter + 1$.
6. Mikäli $f(s) = 0$, niin on löydetty kyseiselle graafille laillinen k-väritys.

Ero simuloidun jäähdytyksen menetelmällä tehtyyn väritykseen on se, että tässä pyrittiin vain löytämään sallittu k väritys. Mikäli halutaan löytää mahdollisimman pieni k, jolle väritys vielä löytyy, joudutaan sama algoritmi suorittamaan useampaan kertaan ja iteroimaan, kunnes algoritmi ei enää löydä sallittua solmujen jakoa. Sekä tabu-hakuun että simuloidun jäähdytyksen menetelmään perustuvissa algoritmeissa alkeissiirtymät ovat samoja: vaihdetaan graafin jonkun solmun väri.

Mikäli tabu-hakuun perustuva algoritmi ei onnistu värityksessä, nähdään lopputuloksesta kuitenkin ne solmut, jotka aiheuttavat ongelmia. Tämän jälkeen voidaan esimerkiksi reititystä muuttamalla päästä parempaan lopputulokseen[6].

[6] tämä tietysti pätee myös muille algoritmeille

Luku 6

Reitityksen ja aallonpituusallokoinnin testaus

Edellä esiteltyjä algoritmeja testattiin kahdella tavalla. Graafin väritysalgoritmeja testattiin värittämällä satunnaisia graafeja, joissa solmut ovat naapureita todennäköisyydellä 0.5. Tämä vastaa siis aallonpituusallokointia WDM-järjestelmässä. Tosin ei ole täysin selvää, kuinka hyvin tällaiset satunnaiset graafit vastaavat aallonpituusallokaatiossa syntyviä väritysgraafeja.

Kun myös WDM-järjestelmän reititys ajatellaan vapaaksi, se pitää ratkaista ennen aallonpituuksien allokointia. Reitityksen ja aallonpituusallokoinnin ratkaisevaa algoritmia (kohta 3.3.1) testattiin erikseen muutamalla esimerkkiverkolla.

6.1 Testausympäristö

Algoritmit ohjelmoitiin C-kielellä [20], jotta toteutus oli suhteellisen tehokas. Jokainen algoritmi sijoitettiin omaan tiedostoonsa. Lähdekoodia syntyi yhteensä yllättävän paljon, yli 4000 riviä.

Graafin solmujen väritysalgoritmien kutsurajapinta määriteltiin seuraavanlaiseksi:

```
int xxx_assign( char **C, int color[], int n, int goal);
```

missä *C* on väritettävän graafin naapurimatriisi, *color* on taulukko, jossa väritys palautetaan ja *n* on väritettävän graafin solmujen lukumäärä. Parametri *goal* on tavoite johon pyritään. Normaalisti sen arvo on -1, jolloin algoritmin on määrä hakea paras mahdollinen väritys. Mikäli *goal* on suurempi kuin nolla, osa algoritmeista pyrkii löytämään värityksen, joka vaatii *goal* väriä.

Testit ajettiin Intel Pentium 120 MHz pohjaisessa mikrotietokoneessa, jossa käyttöjärjestelmänä oli Linux. Nykyäänhän on käytettävissä huomattavasti nopeampiakin tietokoneita, mutta koska tarkoituksena oli vain vertailla menetelmien keskinäistä paremmuutta, ei tällä ole merkitystä.

Suoritusajan mittaaminen tapahtui systeemikutsulla *times()*, joka palauttaa prosessin viemän ajan. Mittauksen resoluutio on periaatteessa mikrosekunti, mutta saatuihin mittausaikoihin sisältyy epävarmuustekijöitä johtuen mm. muista samanaikaisesti ajettavista prosesseista. Lisäksi satunnaisten graafien värityksessä algoritmien suoritusajat luonnollisesti vaihtelivat riippuen siitä, millainen satunnainen graafi sattui olemaan.

6.2 Satunnaisten graafien väritys

Satunnaisia graafeja väritettäessä päästiin kokeilemaan jopa täydellistä hakua, kunhan solmujen määrä pidettiin tarpeeksi pienenä. Täydellinen haku kuitenkin kestää jo muutaman kymmenen solmun verkossa selvästi kauemmin kuin heuristiset menetelmät, joten sen soveltaminen aallonpituusallokaatioon on käytännössä mahdotonta.

Satunnaisten graafien yhteysmatriisin alkiot olivat ykkösiä todennäköisyydellä 0.5 ja vastaavasti nollia samalla todennäköisyydellä. Toisin sanoen mitkä tahansa kaksi solmua olivat naapureita todennäköisyydellä 0.5. Tällöin sivujen lukumäärän odotusarvo on puolet täydellisen graafin sivujen lukumäärästä. Täydellisessä haussa syntyvän hakupuun korkeuden odotusarvo on täten

$$\overline{D}_0 = \frac{v^2 - v}{4}.$$

solmujen	Täydellinen haku				Karsittu haku			
lkm.	$\overline{N}$	V(N)	$\overline{T}$	V(T)	$\overline{N}$	V(N)	$\overline{T}$	V(T)
15	4.9	0.32	3.83	2.01	5.1	0.32	0.07	0
18	5.3	0.23	174	4044	5.6	0.27	0.13	0.01
20	5.7	0.23	3684	6132046	5.9	0.32	0.20	0.03
30	–	–	–	–	7.6	0.27	1.15	1.37
40	–	–	–	–	9.2	0.62	3.31	12.1
50	–	–	–	–	10.9	0.32	16.0	286
60	–	–	–	–	12.1	0.77	53.5	804
70	–	–	–	–	13.9	0.10	135	6243
80	–	–	–	–	15.4	0.49	231	26470
90	–	–	–	–	16.7	0.46	528	87022

Taulukko 6.1: Täydellisen ja karsitun haun tulokset

Jokaisella algoritmilla väritettiin samat kymmenen satunnaista graafia solmujen lukumäärää kohti. Tarvittavien värien lukumääristä ja suoritusajoista laskettiin keskiarvot ja varianssit. Taulukoissa ilmoitetut suoritusajat ovat sekunneissa.

6.2.1 Täydellinen ja karsittu haku

Väritettäessä satunnaisten graafien solmuja täydellisellä haulla ja karsitulla haulla saatiin taulukon 6.1 mukaiset tulokset. Täydellisen haun ajoaika kasvoi aivan liian nopeasti, jotta sillä olisi voinut mitata suurempia graafeja ja verrata tuloksia heuristisilla menetelmillä saavutettuihin värityksiin. Karsittu haku sen sijaan antaa hieman täydellistä hakua huonompia tuloksia jo pienilläkin graafeille. Karsitun haun suoritusaika näyttää kasvavan suunnilleen eksponentiaalisesti.

Karsitussa haussa säilytettiin jokaisella askeleella korkeintaan 20 erilaista väritystä. Tämä on tietenkin kriittinen parametri ja määrää kuinka hyvin algoritmi toimii. Mitä suurempi graafi sitä leveämpi pitäisi “haravan” olla. Karsinnassa käytetty hyvyysmitta voisi toki olla jokin muukin kuin ahneen algoritmin antama väritys.

6.2.2 Ahne algoritmi ja DSATUR

Taulukossa 6.2 on esitetty ahneella ja DSATUR-algoritmeilla saadut mittaustulokset. Algoritmien antamat tulokset ja ajoajat ovat hyvin lähellä toisiaan, kuten arvata saattaakin.

solmujen	ahne algoritmi				DSATUR			
lkm.	$\overline{N}$	V(N)	$\overline{T}$	V(T)	$\overline{N}$	V(N)	$\overline{T}$	V(T)
15	5.2	0.62	0	0	4.9	0.32	0	0
18	5.9	0.32	0	0	6.0	0.22	0	0
20	6.6	0.49	0	0	6.5	0.94	0	0
30	8.0	0.44	0	0	8.0	0.89	0	0
40	10.1	0.10	0	0	10.4	0.71	0	0
50	12.3	0.46	0	0	12.3	0.46	0	0
60	13.3	0.90	0	0	14.0	1.11	0	0
70	15.6	0.71	0	0	15.5	0.72	0	0
80	16.7	0.68	0	0	17.5	1.61	0	0
90	18.6	0.27	0.01	0	18.7	0.46	0.01	0
100	20.3	2.23	0.01	0	20.4	0.49	0.01	0
125	23.5	0.94	0.01	0	23.9	0.32	0.01	0
150	27.9	1.88	0.01	0	27.7	0.23	0.02	0
175	30.7	0.68	0.02	0	30.9	0.99	0.02	0
200	33.5	0.50	0.02	0	33.9	0.77	0.03	0
400	57.9	1.21	0.11	0	59.1	1.43	0.16	0
600	80.8	1.29	0.32	0	82.8	3.51	0.46	0
800	102.1	1.66	0.70	0	104.2	0.62	0.96	0
1000	122.1	3.21	1.23	0	124.6	1.38	1.67	0

Taulukko 6.2: Ahneen algoritmin ja DSATUR-menetelmän tulokset.

Suurilla solmujen lukumäärillä DSATUR menetelmä näyttää antavan pari väriä huonompia tuloksia kuin tavallinen ahne algoritmi.

Huomattavaa on molempien algoritmien ajoaika. Esimerkiksi 200 solmua sisältävän graafin väritys vei aikaa vain sekunnin sadasosia, kun vastaavaan graafin väritys muilla menetelmillä kestää useita sekunteja.

6.2.3 Tabu-haku, simuloitu jäähdytys ja geneettinen algoritmi

Monimutkaisempien graafin solmujen väritysalgoritmien tulokset on esitetty taulukossa 6.3. Pienimmillä solmujen lukumäärillä kaikki algoritmit (GA, SA ja tabu) löytävät optimaalisen värityksen. Tämän jälkeen simuloidun jäähdytyksen menetelmä antaa parhaita tuloksia, kunnes 60-70 solmun kokoluokassa tabu-haku ohittaa sen. Siitä eteenpäin tabu-haku antaa parhaita tuloksia; 200 solmun kohdalla ero on jo huomattava, 4-6 väriä verrattuna SA:n ja GA:n menetelmiin.

	SA				GA				TABU			
solmut	$\overline{N}$	**V(**N**)**	$\overline{T}$	**V(**T**)**	$\overline{N}$	**V(**N**)**	$\overline{T}$	**V(**T**)**	$\overline{N}$	**V(**N**)**	$\overline{T}$	**V(**T**)**
15	4.9	0.32	4.34	2.19	4.9	0.32	0.20	0	4.9	0.32	0.85	0
18	5.3	0.23	5.81	3.69	5.3	0.23	0.27	0	5.3	0.23	0.94	0
20	5.7	0.23	6.99	3.34	5.7	0.23	0.33	0	5.8	0.18	1.04	0
30	7.1	0.32	13.0	19.2	7.1	0.32	0.64	0	7.4	0.27	1.44	0.03
40	8.6	0.27	22.0	20.3	9.1	0.10	1.14	0	8.7	0.46	2.29	0.20
50	10.3	0.23	28.1	86.4	10.5	0.28	1.81	0.01	10.4	0.27	3.05	0.65
60	11.9	0.32	35.3	47.3	12.3	0.46	2.63	0.02	11.5	0.28	4.31	0.79
70	13.9	0.32	36.5	38.7	14.0	0.22	3.72	0.02	12.8	0.18	7.05	1.02
80	15.3	0.46	44.6	58.1	15.3	0.23	5.12	0.02	14.0	0.00	9.71	3.53
90	17.1	0.32	54.3	23.0	17.2	0.18	6.81	0.04	15.4	0.27	14.6	6.45
100	18.9	0.77	60.0	62.9	18.2	0.18	8.71	0.07	16.8	0.18	16.5	14.5
125	23.4	0.71	83.3	47.9	22.1	0.32	15.1	0.85	19.9	0.32	33.4	48.8
150	28.3	0.46	110	53.9	25.6	0.27	23.5	0.42	22.5	0.50	63.5	270
175	32.2	1.07	139	123	29.0	0.67	35.3	1.15	25.5	0.72	88.7	207
200	36.8	0.62	177	148	32.0	0.22	49.0	1.82	28.2	0.84	139	783

Taulukko 6.3: Simuloidun jäähdytyksen, geneettisen algoritmin ja tabu-haun tulokset

vakio	arvo	vakio	arvo
α	0.995	N	$5 \cdot n$
T_0	1.0	M	$1 + n/4$
T_1	0.005		

Taulukko 6.4: Simuloidun jäähdytyksen testauksessa käytetyt parametrien arvot

Simuloidun jäähdytyksen menetelmän “huonot” tulokset suurilla solmujen lukumäärillä selittyvät tietenkin parametrien valinnoilla. Jäähdytyksessä alkulämpötila ja loppulämpötila olivat kiinteitä. Myös jäähdytysnopeuden määrävä parametri α oli kiinteä. Iteraatiokierrosten lukumääräparametrit N ja M taas olivat väritettävän graafin solmujen lukumäärän funktioita. Testeissä käytetyt parametrien arvot ovat taulukossa 6.4 Jos jäähdytys tehtäisiin riittävän hitaasti, tulokset olisivat varmasti parempia. Testissä pyrittiin pitämään menetelmän ajoaika suunnilleen samassa luokassa tabu-haun kanssa.

Geneettisen algoritmin tulokset ovat varsin hyviä. Testeissä käytetty populaation koko oli 30. Geneettisen algoritmin ajoaika (annetuilla parametreilla) kasvaa selvästi hitaammin kuin tabu-hakuun tai simuloituun jäähdytykseen perustuvilla algoritmeilla. Silti suurilla solmujen lukumäärillä sen antamat väritykset ovat parempia kuin simuloidun jäähdytyksen menetelmällä saadut. Geneettinen algoritmihan perustui ahneeseen algoritmiin, jossa

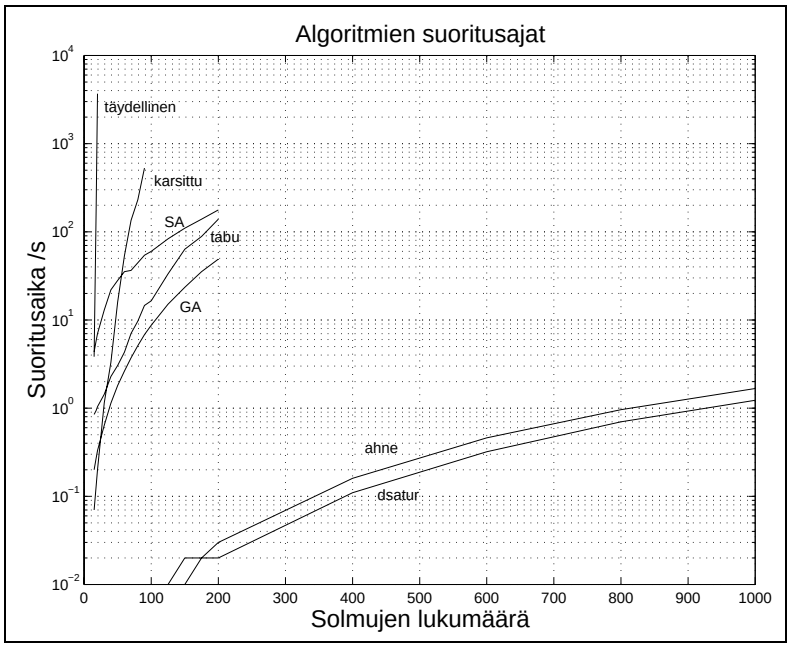
väritysjärjestystä muuttamalla pyrittiin parhaaseen mahdolliseen väritykseen. Täten sen toiminta on rajoitettu ahneen algoritmin sallimissa rajoissa.

Suuremmilla solmujen lukumäärillä parhaimmat väritykset löysi tabu-haun menetelmä. Tabu-haun menetelmässä oli käytössä seuraavat vakioiden arvot:

$$\begin{cases} |T| &= 7 \\ nbmax &= 10000 \\ nrep &= 80 \end{cases}$$

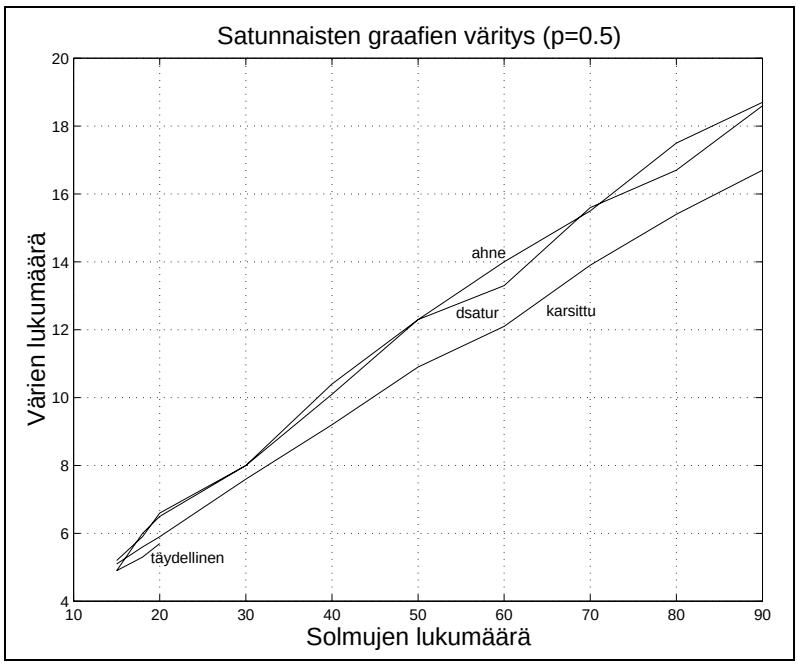
6.2.4 Johtopäätöksiä graafien väritysalgoritmeista

Kuvassa 6.1 on esitetty algoritmien vaatimat ajoajat. Algoritmit voidaan karkeasti jakaa kolmeen luokkaan ajoajan perusteella. Ahneet algoritmit ovat selvästi muita nopeampia, seuraavaan luokkaan kuuluvat heuristiset algoritmit SA, GA ja tabu-haku, ja kolmanteen täydellinen ja karsittu haku. Täydellisen haun ajoaika kasvoi aivan liian nopeasti niin suureksi, että testien tekeminen oli sillä mahdotonta. Tämä on sinänsä harmi, sillä täydellisen haun antama paras mahdollinen väritys olisi hyvä referenssitaso muille algoritmeille.



Kuva 6.1: Algoritmien ajoajat.

Kuvassa 6.2 on esitetty täydellisen ja karsitun haun antamien värien lukumäärien kehitys. Karsitun haun ajoaika kasvoi myös sen verran nopeasti, että 90 solmua suurempia graafeja

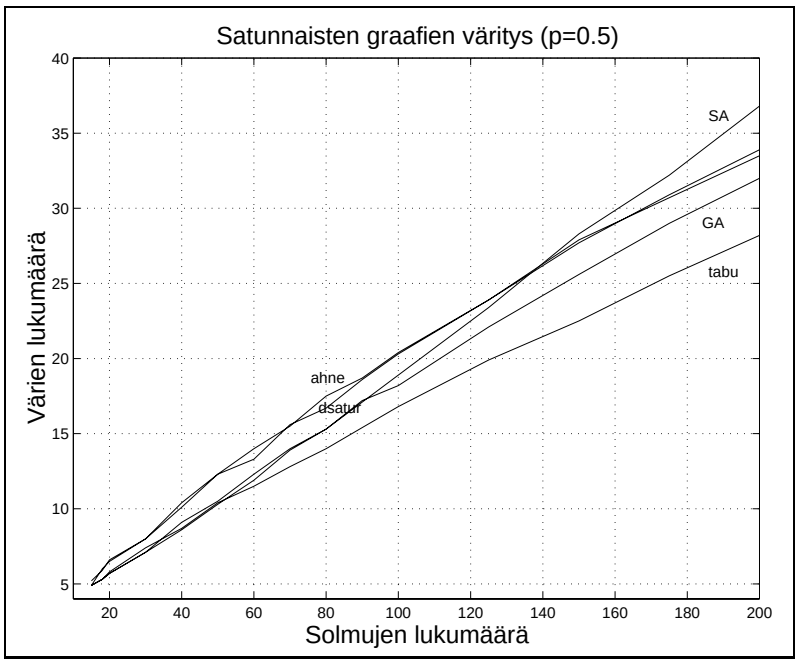


Kuva 6.2: Täydellisen ja karsitun haun käyttämien värien lukumäärien kehitys.

ei testiin mukaan otettu. Vertailun vuoksi kuvaan on piirretty myös ahneiden algoritmien antamat värien lukumäärät. Karsitussa haussa kriteerinä käytettiin ahnetta algoritmia, joten on selvää, että lopputulos ei voi olla huonompi kuin ahneella algoritmilla. Saavutettu parannus näyttää olevan kahden värin luokkaa kokoluokassa 50-90 solmua.

Kuvaan 6.3 on mukaan otettu ahneiden algoritmien lisäksi simuloidun jäähdytyksen, geneettisen algoritmin ja tabu-haun tulokset. Kuvasta nähdään, että tabu-haku suoriutuu väritystehtävästä parhaiten (lukuunottamatta aivan alkupäätä). Seuraavaksi paras on geneettiseen algoritmiin perustuva menetelmä. Koska GA:n menetelmä antaa säännöllisesti parempia tuloksia kuin kumpikaan ahneista algoritmeista, voidaan päätellä, että GA on keskimäärin onnistunut löytämään paremman väritysjärjestyksen. Simuloitu jäähdytys käytetyillä parametreilla toimii kohtuullisesti, kun solmujen lukumäärä on alle sata, mutta tämän jälkeen käytettyjen värien lukumäärä kasvaa nopeammin kuin muilla kuvan 6.3 algoritmeilla.

Olettaen, että väritysgraafeissa on muutama sata solmua, ovat tabu-haku ja ahneet algoritmit luultavasti parhaimmat vaihtoehdot. Mikäli suoritusaika on kriittinen tekijä, kääntyy valinta ehdottomasti ahneisiin algoritmeihin (tavallinen tai DSATUR), muuten tabu-haku lienee paras vaihtoehto.



Kuva 6.3: SA, GA ja tabu -menetelmien tarvitsemat värien lukumäärät.

6.3 Esimerkkiverkkojen reititys ja aallonpituusallokointi

Kohdassa 3.3.1 esitettyä reitityksen ja aallonpituusallokaation ratkaisevaa algoritmia testattiin muutamalla kuvitteellisella verkolla. Algoritmihan pyrkii hakemaan hyvän reitityksen iteratiivisesti värittämällä eri reititysehdokkaita ahneella algoritmilla. Testeillä pyrittiin saamaan jonkinlainen kuva konkreettisista verkoista ja siitä, kuinka tarvittavien värien määrä käyttäytyy.

Reititystä haettaessa mukaan otettiin vain ne reitit, jotka olivat yhteysvälien lukumäärän mielessä lyhimpiä; tällaisia reittejä on yleensä useita. Täten jokainen yhteyspari vie kapasiteettia linkeiltä mahdollisimman vähän ja keskimääräinen kuorma linkkiä kohden tulee minimoiduksi. Tämä ei kuitenkaan tarkoita, ettei sellaisella reitityksellä, jossa yksi tai useampi yhteyspari käyttäisi optimaalista pidempää reittiä, päästäisi jossakin tapauksessa parempaan lopputulokseen, kun kriteerinä on tarvittavien värien lukumäärä. Vaikka mukaan otettiin vain lyhimmät reitit pisteparien välillä, saadaan niistäkin hyvin suuri joukko aikaan. Tämä tarkoittaa tietenkin sitä, että kaikkien mahdollisuuksien järjestelmällinen läpikäyminen on käytännössä mahdotonta.

Taulukkoon 6.5 on koottu esimerkkiverkkojen tiedot ja tulokset. Kun tarkastellaan eri linkkien kuormitusta reitityksen jälkeen, niin olisi tietenkin suotavaa, että mahdollisimman monella linkillä olisivat kaikki n väriä käytössä. Tällöin voidaan ajatella, että WDM-verkon kapasiteetti olisi "hyvin" hyödynnetty. Kunkin esimerkkiverkon linkkien kuormitusjakauma on esitetty kyseisen verkon kohdalla.

Reitityksen ja aallonpituusallokaation vaikeutta monesti hyvin kuvaava indeksi on verkon fyysisen yhteydellisyyden (physical connectivity) aste [17]:

$$\alpha = \frac{\mathrm{L}}{\mathrm{L}_{FC}} = \frac{2 \cdot \mathrm{L}}{\mathrm{N}^2 - N}$$

joka on käytettävissä olevien linkkien lukumäärän suhde haluttujen yhteysparien lukumäärään. Mitä suurempi suhde on, sitä helpompi reititys- ja aallonpituusallokointitehtävä yleensä on. Parametrin käänteisarvo on selvästi ehdoton alaraja tarvittavien värien lukumäärälle.

Paljon parempi alaraja saadaan jakamalla verkko sopivasti kahteen osaan, joita yhdistää pieni määrä linkkejä. Tällöin pätee

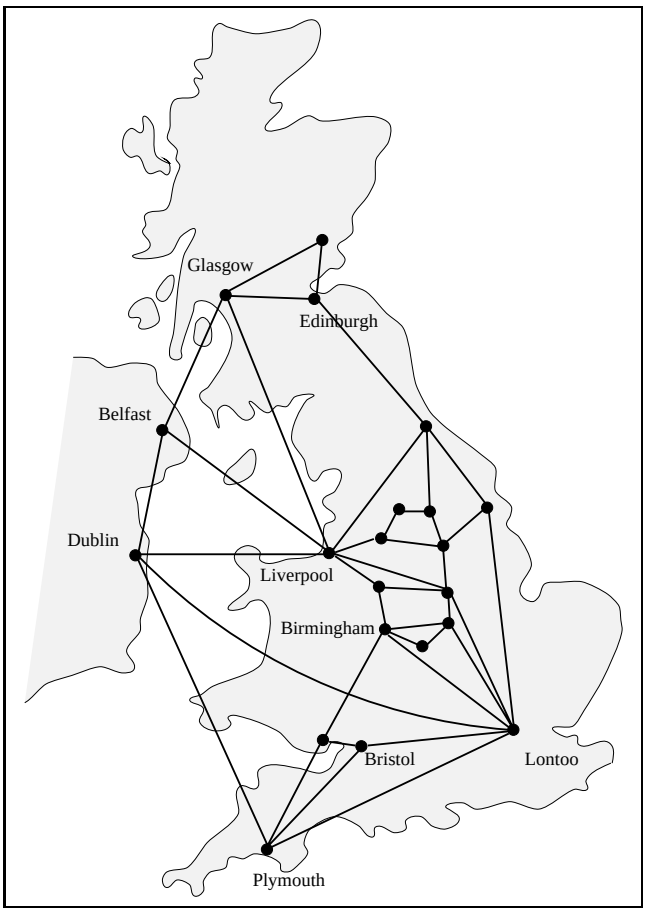
$$W = \left\lceil \frac{K \cdot (N - K)}{C} \right\rceil,$$

missä toisessa osajoukossa on K solmua ja toisessa $N - K$ solmua. Osajoukkoja yhdistää C linkkiä. Tällöin W on selvästi ehdoton alaraja tarvittavien aallonpituuksien lukumäärälle. Hakemalla W:n maksimi yli kaikkien mahdollisten solmujen jakojen, saadaan yleensä suhteellisen hyvä alaraja.

6.3.1 Brittein saarten verkko

Kuvassa 6.4 on Brittein saarten päälle rakennettu verkko, jota Tan ja Pollard käyttivät testatessaan algoritmiaan [15]. Solmuja verkossa on 21 kappaletta, joten yhteyspareja syntyy 210 kpl. Kuvasta nähdään, että Lontooseen ja Liverpooliin on sijoitettu suurehko määrä yhteyksiä.

Käsittely tässä poikkeaa artikkelissa [15] esitetystä siinä, että maantieteellisiä etäisyyksiä ei otettu lainkaan huomioon reittejä haettaessa, vaan etäisyyden kahden solmupisteen välillä ajatellaan olevan vakio, esimerkiksi yksi.

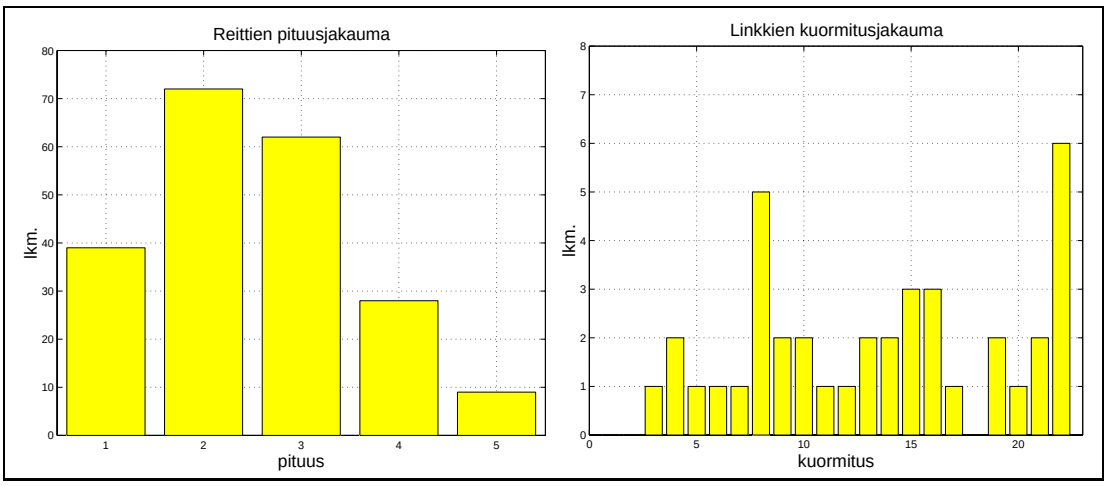


Kuva 6.4: Brittein saarten päälle rakennettu kuvitteellinen verkko.

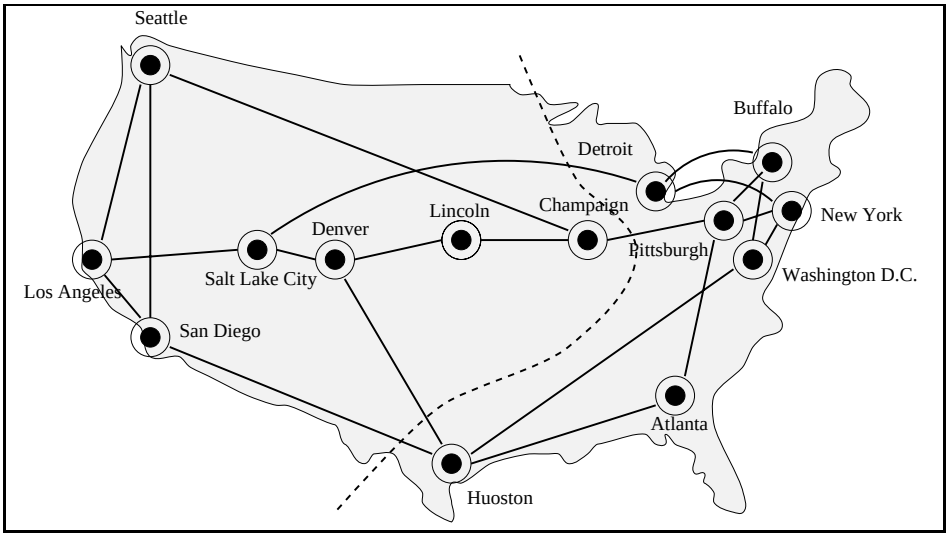
Käytetty algoritmi löysi 22 aallonpituutta käyttävän konfiguraation. Kun tarkastellaan linkkien kuormitusjakaumaa (kuva 6.5), voidaan saavutettua tulosta pitää hyvänä, sillä linkkien keskimääräinen käyttöaste on varsin hyvä. Lisäksi kuormitusjakauman maksimi sijaitsee juuri 22 yhteyden kohdalla.

Reittien pituusjakauma on hyvin tyypillinen tasoverkoille. Verkon pisimmät yhteydet sisältävät viisi hyppyä, kun taas suurin osa yhteyksistä on 2-3 hypyn pituisia.

Kyseessä on jo suhteellisen monimutkainen tehtävä johtuen suurehkosta määrästä solmuja. Lopullisen konfiguraation löytämiseen aikaa menikin suunnilleen yksi minuutti. Kaikkien algoritmien testitulokset on koottu taulukkoon 6.5.



Kuva 6.5: Brittein saarten verkon pituus- ja kuormitusjakaumat.

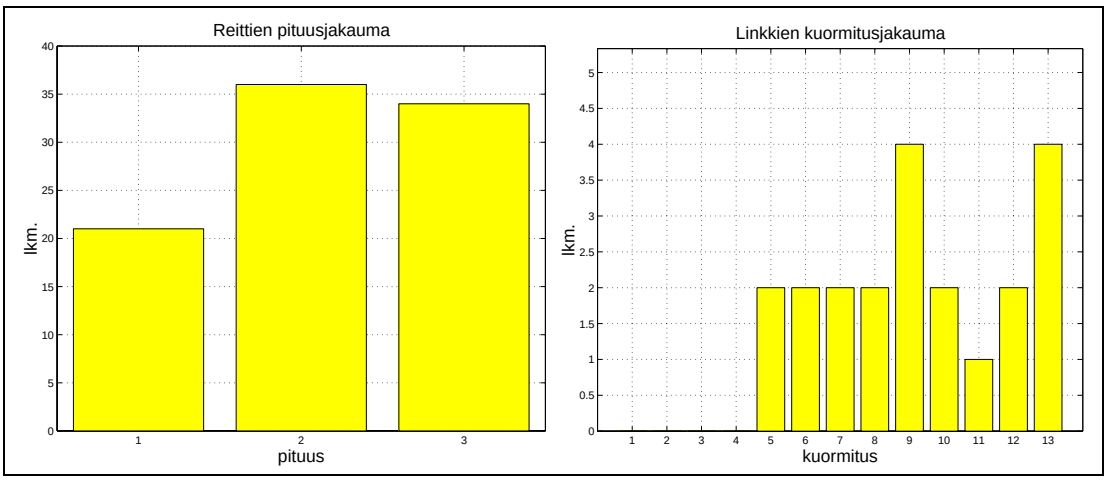


Kuva 6.6: NSFNET vuonna 1993.

6.3.2 NSFNET:in runkoverkko

Toinen esimerkkiverkko on Yhdysvalloissa sijainnut NSFNET:in (National Science Foundation Network, vastannee FUNET:ia Suomessa) runkoverkko vuodelta 1993. NSFNET:issä solmupisteitä oli 14 kappaletta, joten reititystehtävänä sen tulisi olla helpompi kuin Ison-Britannian kuvitteellinen verkko.

NSFNET:issä ei minkään solmuparin välinen yhteys ole kolmea hyppyä pidempi, kuten reittien pituusjakaumasta voidaan nähdä. Tämä viittaa siihen, että verkon suunnittelussa on pyritty minimoimaan etenemisviiveet. Kuvasta 6.6 nähdään, että verkossa onkin muutama pitkä yhteys yli lähimpien naapureiden.



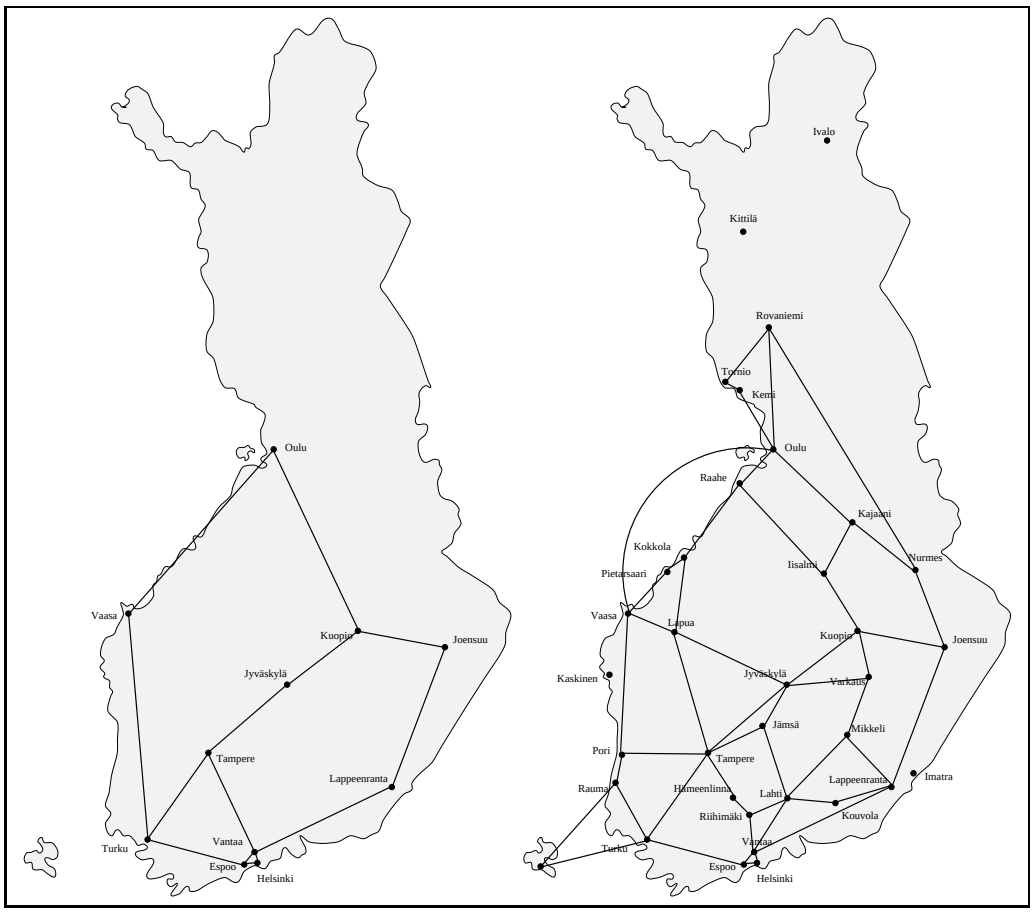
Kuva 6.7: NSFNET -verkon pituus- ja kuormitusjakaumat.

Käytetty algoritmi löysi noin neljässä sekunnnissa konfiguraation, joka tarvitsee 13 väriä. Linkkien kuormitusjakaumasta voidaan päätellä, että saavutettu tulos on täysin tyydyttävä. Teoreettinen alaraja värien lukumäärälle kyseisessä verkossa on myös 13, kuten helposti nähdään leikkaamalla verkko noin keskeltä kahteen osaan (kuva 6.6). Tällöin kummallekin puolelle jää 7 solmua ja puolikkaita yhdistää kolme linkkiä, josta saadaan keskimäärin 12.25 yhteyttä linkkiä kohti. NSFNET-verkon konfigurointia voidaankin pitää helppona tehtävänä.

6.3.3 Suomen päälle sijoitetut verkot

Kolmas ja neljäs esimerkkitapaus on rakennettu Suomen kartan päälle. Ensimmäisessä verkossa solmupisteet on sijoitettu suurimpiin kaupunkeihin (kuva 6.8): Helsinki, Vantaa, Espoo, Tampere, Turku, Vaasa, Oulu, Kuopio, Joensuu, Jyväskylä ja Lappeenranta, eli verkossa on solmupisteitä yksitoista kappaletta. Teoreettinen alaraja tarvittavien värien lukumäärälle saadaan leikkaamalla verkko kahdeksi yhtäsuureksi osaksi Vaasa-Oulu, Tampere-Jyväskylä ja Vantaa-Lappeenranta yhteyksien kohdalta. Koska rajapinnan yli kulkee $5 * 5 = 25$ yhteyttä kolmella linkillä, tiedetään, että laillisen värityksen löytämiseksi tarvitaan vähintään 9 väriä.

Reitityksen ja aallonpituusallokaation ratkaiseva algoritmi löysi tyypillisesti konfiguraation, joka tarvitsi 10 väriä. Ratkaisua voidaan pitää hyvänä. Algoritmin vaatima suoritusaika oli vain muutamia sekunteja.



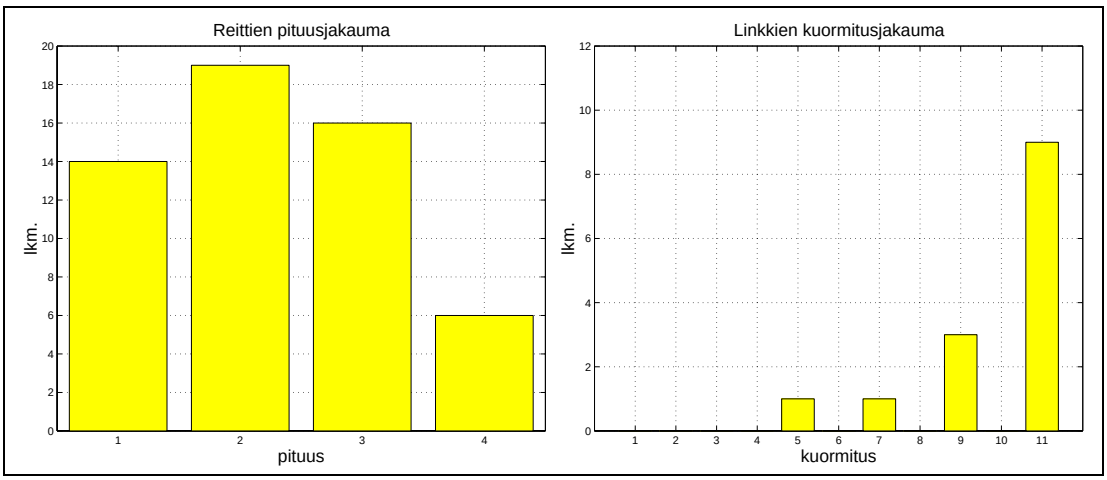
Kuva 6.8: Suomeen rakennettu kuvitteellinen WDM-verkko.

Kuvassa 6.9 on esitetty lyhimpien reittien pituusjakauma ja reitityksen ja aallonpituusallokoinnin jälkeen linkkien kuormitusjakauma. Kuormitusjakauma on toivotunlainen, suurimmalla osalla linkeistä on koko kapasiteetti hyötykäytössä. Tämä ei sinänsä ole yllätys, koska saavutettu tulos on hyvin lähellä teoreettista ala-rajaa. Mielenkiintoista kuormituskuvassa on se, että värien lukumäärät kaikilla yhteysväleillä ovat parittomia lukuja. Tämä ei ole mikään yleinen tapaus vaan lähinnä sattuma.

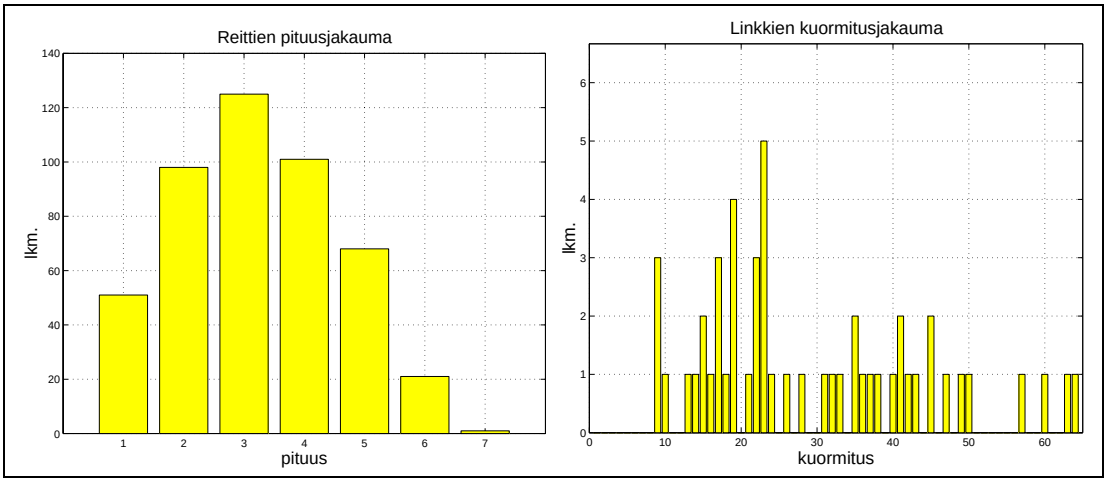
Vaikka kyseessä on näinkin yksinkertainen verkko, ei oikeiden reittien valitseminen ole aivan helppoa. Mahdollisia vaihtoehtoja valita reitit kaikkien pisteparien välille on noin 10^5 kappaletta.

Toisessa esimerkkiverkossa solmupisteiden lukumäärää on lisätty 31:een, joten täysin kytketyssä verkossa on 465 yhteysparia. Tämän kokoluokan verkon reitityksen ja aallonpituusallokoinnin suorittaminen alkaa olla jo vaikeaa. Myös ajoaika kasvaa kohtalaisen suureksi eli noin kymmeneen minuuttiin.

Tämän monimutkaisen verkon reititys ja aallonpituusallokaatio on ollut algoritmille sel-



Kuva 6.9: Suomi I -verkon pituus- ja kuormitusjakaumat.



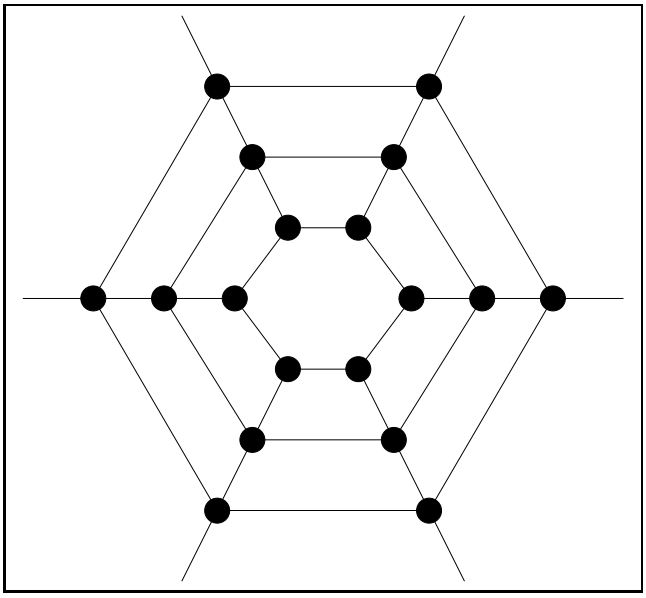
Kuva 6.10: Suomi II -verkon pituus- ja kuormitusjakaumat.

västi vaikeampi tehtävä. Linkkien kuormitusjakaumasta huomataan, että verkon kapasiteettia ei käytetä hyväksi kovin tehokkaasti. Vain yhdellä linkillä yhteyksiä kulkee 64 kappaletta ja muutenkin kuormitusjakauman yläpäässä on kovin tyhjää. Esimerkiksi lisäämällä ylimääräinen optinen kuitu viidelle kuormitetuimmalle yhteysvälille, putoaisi tarvittavien värien lukumäärä varmasti alle 50 värin.

Yhteyksien pituusjakauma muistuttaa hiukan Gaussin jakaumaa. Verkon pisin yhteysväli kulkee seitsemän linkin kautta. Verkon linkkien suhde päästä päähän -yhteyksiin (α) on noin 0.1, joten suurehko värien määrä ei ole yllätys. Toisin sanoen, jotta päästäisiin parempaan kuormitusjakaumaan, fyysiseen verkkoon tulisi lisätä riittävä määrä yhteyksiä. Muutama pitkä yhteys poikki maan varmaan takaisikin huomattavasti paremman konfiguraation. Muilla esimerkkiverkoilla kyseinen suhdeluku on ollut hiukan yli 0.2, mikä

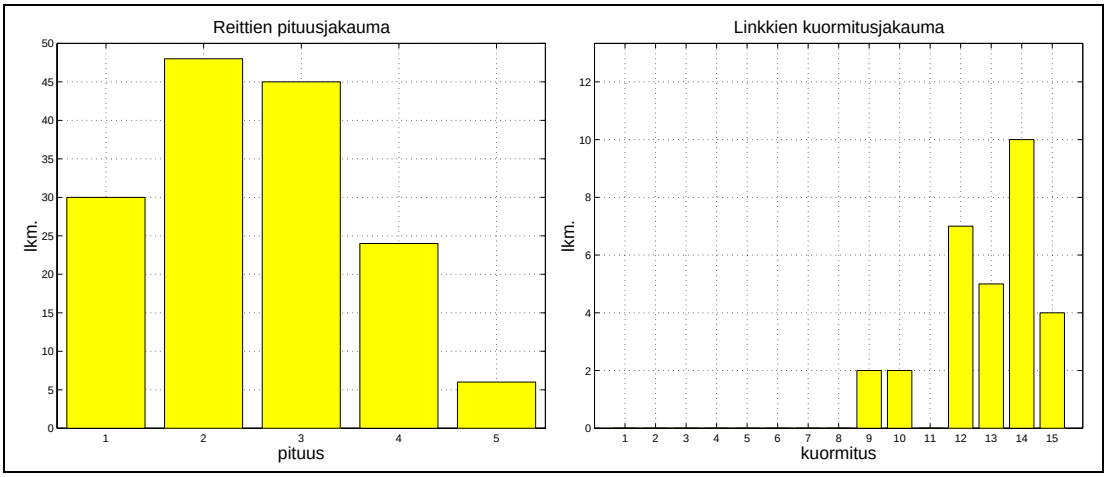
ilmeisesti on riittävä.

6.3.4 Symmetrinen verkko



Kuva 6.11: Symmetrinen verkko.

Reititysalgoritmia kokeiltiin myös säännöllisellä “hämähäkinverkolla”. Kuvasta nähdään, että pullonkaulana toimii keskimmäinen “rengas”, sillä ulko- ja sisäreuna ovat reitityksen kannalta samanarvoisia. Verkko voidaan tavallaan pyöräyttää ympäri keskimmäisen renkaan suhteen, jolloin sisäreunasta tulee ulkoreuna ja päinvastoin.



Kuva 6.12: Symmetrisen verkon pituus- ja kuormitusjakaumat.

verkko	N	L_{FC}	L	α	R	T	C_0	C_1
Brittein saaret	21	210	39	0.169	41	55	26	22
NSFNET	14	91	21	0.200	7	4	16	13
Suomi I	11	55	1.5	0.212	5	2	13	10
Suomi II	31	465	51	0.103	79	622	74	64
symmetrinen verkko	18	153	30	0.175	55	19	22	15

Taulukko 6.5: Esimerkkiverkot: (N=solmujen lkm., $L_{FC} = (N^2 - N)/2$= yhteysparien lkm., L=linkkien lkm., $\alpha = L/L_{FC}$, R=reittivaihtoehtojen lkm. 10^x, T=ajoaika/s, C_0=värien lkm. alussa, C_1=värien lkm. lopussa).

Tälle verkolle algoritmi löysi 15 väriä tarvitsevan konfiguraation. Linkkien kuormitusjakaumasta nähdään, että keskimääräinen käyttöaste linkeillä on suhteellisen hyvä, ja täten voidaan ratkaisua pitää myös hyvänä. Reittien pituusjakauma on tyypillinen. Aikaa algoritmi tarvitsi tässä tapauksessa noin 15 sekuntia.

Näinkin säännöllisen verkon linkkien lukumäärän suhde päästä päähän -yhteyksien lukumäärään on vain 0.175. Erilaisia reittivaihtoehtoja verkolla on noin 10^{55} kappaletta, mikä onkin hyvin suuri luku. Toisaalta koska kyseessä on säännöllinen verkko, suurin osa vaihtoehdoista ovat keskenään ekvivalentteja (kahden yhteyden reititykset voidaan “vaihtaa” keskenään ja päädytään samanlaiseen väritysgraafiin).

6.3.5 Johtopäätöksiä esimerkkiverkkojen perusteella

Reitityksen toteutus oli varsin yksinkertainen. Paremmalla reititysalgoritmilla saattaisi olla suurikin vaikutus menetelmän toimintaan. Erilaisia reittivaihtoehtoja on hyvin suuri määrä jo yksinkertaisissa verkoissa, kuten taulukosta 6.5 voidaan nähdä. Joten reititysvaihtoehtojen järjestelmällinen läpikäyminen on käytännössä mahdotonta.

Reittejä valittaessa voisikin hyvin soveltaa samoja heuristisia menetelmiä kuin graafin solmuja väritettäessa. Esimerkiksi tabu-haku saattaisi toimia hyvin ja vauhdittaa algoritmin suppenemista. Reitityksessä suurin ongelma on hyvän reitityksen tunnistaminen. Pelkkä yhteyksien lukumäärän minimoiminen yli kaikkien linkkien ei aallonpituusjakoisessa tiedonsiirtojärjestelmässä takaa parasta mahdollista konfiguraatiota. Toisaalta parempien heuristisen graafin solmujen väritysalgoritmien käyttö iteratiivisesti reitityksen hakevassa menetelmässä tuntuu mahdottomalta ajatukselta, joten käytännössä aina joudutaan tekemään jonkinlainen kompromissi menetelmän laadukkuuden ja käytettävissä

olevien resurssien (lähinnä ajan) suhteen.

Huomionarvoista on, että kaikki verkot, joissa verkon linkkien lukumäärän suhde haluttujen yhtyksien lukumäärään oli luokkaa 0.2 tai suurempi, onnistuttiin reitittämään suhteellisen hyvin. Tätä voitaneen pitää nyrkkisääntönä suunniteltaessa verkkoa.

Olisi hyödyllistä tutkia myös tapauksia, joissa kaikkien solmujen väliset liikennemäärät eivät olisi vakioita. Tällöin ei välttämättä olisi tarpeen muodostaa kaikkien solmujen välille suoraa yhteyttä, vaan ne yhteysparit joiden liikennemäärät ovat kovin pienet, voisivat esimerkiksi liikennöidä kolmannen solmun kautta. Tämä tilanne vastaisi paljon paremmin reaalimaailman tarpeita.

Lisäksi pitäisi ehkä sallia ylimääräiset optisen kuidut joillekin linkeille tietyllä lisäkustannuksella. Näin määritelty optimointitehtävä on luonnollisesti huomattavasti monimutkaisempi, mutta samalla se kuvaa paljon paremmin todellista maailmaa, jossa teleoperaattorit toimivat.

Luku 7

Yhteenveto

Aallonpituusjakoinen multipleksointi on epäilemättä tulevaisuuden tekniikka optisessa tiedonsiirrossa. Sen avulla nykyisten optisten runkoverkkojen kapasiteetti pystytään edullisesti moninkertaistamaan ilman uusien optisten kuitujen vetämistä. Optisia kuituyhteyksiä on olemassa huomattava määrä, joten niiden tehokkaampi hyväksikäyttö on taloudellisesti kannattavaa.

Aallonpituusjakoisen multipleksoinnin avulla jokaisessa optisessa kuidussa siirretään useampi signaali yhden sijasta. Tämän hetken kaupalliset järjestelmät kykenevät tyypillisesti käyttämään 16 tai 32 eri aallonpituutta. Laboratoriokokeissa on päästy huomattavasti suurempiin kanavamääriin ja jopa terabitti sekunnissa on onnistuttu siirtämään usean sadan kilometrin päähän. Tällaiset hyvin suuret tiedonsiirtonopeudet tarvitsevat paljon uutta tekniikkaa. Kuituvahvistimet esimerkiksi ovat olennainen osa WDM-järjestelmää. Kuituvahvistin vahvistaa optisen signaalin konvertoimatta sitä missään vaiheessa sähköiseen muotoon. Täten käytetyllä modulaatiolla tai tiedonsiirtonopeudella ei ole vaikutusta vahvistimen toimintaan. Toinen tärkeä ominaisuus kuituvahvistimilla on se, että niiden avulla pystytään vahvistamaan useammalla eri aallonpituudella olevat signaalit samalla kertaa.

Täysoptisessa verkossa reititys solmupisteissä tapahtuu aallonpituuden perusteella. Solmupisteitä on periaatteessa kahdenlaisia. Toisissa voidaan suorittaa aallonpituuskonversio, jolloin verkon reititystehtävä ei poikkea perinteisestä verkosta. Jokaisella linkillä on tietty rajallinen määrä “johtoja” käytettävissä ja reitityksessä tämä tulee ottaa huomioon. Yksinkertaisimmillaan tämä tarkoittaa yhteyksien lukumäärän minimointia yli kaikkien yhteysvälien.

Aallonpituuskonversion suorittaminen solmupisteissä vaatii luonnollisesti ylimääräisiä kom-

ponentteja, mikä tekee siitä kalliimman vaihtoehdon. Joten mikäli aallonpituuskonversiosta voidaan luopua, saadaan järjestelmästä edullisempi. Tämä aiheuttaa kuitenkin ongelmia reitityksessä, sillä jokaisen yhteyden käyttämä aallonpituus pysyy samana jokaisella linkillä reitin varrella ja täten varaa kyseisen kanavan. Joten reitityksen lisäksi verkon konfigurointi vaatii aallonpituusallokoinnin.

Aallonpituusallokointi tarkoittaa käytännössä reitityksen perusteella syntyvän graafin solmujen väritystä. Graafin solmujen väritystehtävä on NP-hankala ongelma, joten joudutaan turvautumaan heuristisiin algoritmeihin. Graafin solmujen väritystä on tutkittu laajasti ja siitä löytyy myös kirjallisuutta. Tässä työssä on esitetty joukko heuristisia algoritmeja, jotka pyrkivät ratkaisemaan väritysongelman. Kirjallisuusviitteistä löytyvien menetelmien lisäksi pyrittiin löytämään myös uusia tapoja ratkaista ongelma. Täydellisen haun menetelmästä (ei siis heuristinen) johdettu karsittu haku ei kuitenkaan osoittautunut varteenotettavaksi vaihtoehdoksi tässä sovelluksessa. Satunnaisten graafien värityksessä tutkituista algoritmeista ylivoimaisesti parhaiten tuntui toimivan tabu-haku. Ahneet algoritmit toimivat myös varsin hyvin, kun ottaa huomioon niiden hyvin nopean suoritusajan verrattuna muihin algoritmeihin.

Tässä työssä tutkittiin myös sekä reitityksen että aallonpituusallokaation ratkaisevaa algoritmia. Algoritmi pyrkii iteratiivisesti löytämään sellaisen reitityksen, jonka aallonpituusallokaatio onnistuu hyvin. Menetelmää testattiin joukolla kuvitteellisia verkkoja, ja saadut tulokset olivat varsin hyviä. Monimutkaisempien graafin väritysalgoritmien käyttö reitityksen jälkeen ei tuntunut antavan parempia tuloksia kuin mitä ahneella algoritmilla oli jo saatu.

Kirjallisuutta

[1] Colin R. Reeves, Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, 1995.

[2] V.J Rayward-Smith, I.H. Osman, C.R. Reeves and G.D. Smith, Modern Heuristic Search Methods. John Wiley & Sons, 1996.

[3] Edwin F. Beckenbach (ed.), Applied Combinatorial Mathematics. John Wiley & Sons, 1964.

[4] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications. Macmillan Press, 1976.

[5] Bernard Harris (ed.), Graph Theory and Its Applications. Academic Press, 1970.

[6] A. Hertz and D. de Werra, Using Tabu Search Techniques for Graph Coloring. Computing 39, 345-351, 1987.

[7] John Mitchem, On various algorithms for estimating the chromatic number of graph. The Computer Journal, vol 19, 182-183.

[8] D. Brélaz. New methods to colour the vertices of a graph. In Communications of the ACM, volume 22, number 4, 1979.

[9] T. Saaty and P. Kainen, The Four-Color Problem. McGraw-Hill, 1977.

[10] K. Appel and W. Haken, The solution of the Four-Color-Map Problem. Scientific American, volume 237, number 4, 1977.

[11] A.E. Willner, Mining the optical bandwidth for a terabit per second. IEEE Spectrum, April 1997.

[12] E.J. Lerner, Multiple Wavelengths exploit fiber capacity. Laser Focus World, July 1997.

[13] M. Berger, M. Chbat, A. Jourdan, M. Sotom, P. Demeester, B. Van Caenegem, P. Godsvang, B. Hein, M. Huber, R. März, A. Leclert, T. Olsen, G. Tobolka, T. Van den Broeck, Pan-European Optical Networking using Wavelength Division Multiplexing. IEEE Communications Magazine, April 1997.

[14] Helkama-Kaapeli, Valokaapelit tiedonsiirrossa, 1995.

[15] T.K. Tan and J.K. Pollard, Determination of minimum number of wavelength required for all-optical WDM networks using graph colouring. Electronic letters, vol.31, No.22, 1995.

[16] B. Mukherjee, D. Banerjee, S. Ramamurthy, A. Mukherjee, Some Principles for Designing a Wide-Area WDM Optical Network. IEEE/ACM Transactions on Networking, vol. 4, number 5, 1996.

[17] S. Baroni, P. Bayvel, Wavelength Requirements in Arbitrarily Connected Wavelength-Routed Optical Networks. Journal of Lightwave Technology, vol 15, number 2, 1998.

[18] S. Subramaniam, R. A. Barry, Wavelength Assignment in Fixed Routing WDM Networks. IEEE International Conference on Communications 1997, pp. 406-410.

[19] M. Garnot, M. Sotom, F. Masetti, Routing Strategies for Optical Paths in WDM Networks. IEEE International Conference on Communications 1997, pp. 422-426.

[20] Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language : ANSI C Version. 2nd Edition. Prentice Hall, 1988.

[21] Dimitri Bertsekas, Robert Gallager, Data Networks, Prentice-Hall, 1992.