Hahmojen etsintä rakenteisesta biomolekyyliaineistosta

Tomi Kauppinen
Tomi.Kauppinen@cs.helsinki.fi

Tiedon louhinta biomolekyyliaineistoista
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Raportti C–2003–52, s. 69–79, marraskuu 2003

Tiivistelmä

Biologista aineistoa sisältävien tietokantojen määrä on kasvanut ja kasvaa edelleen nopeasti. Samalla on kiinnitetty huomiota datan rakenteisuuden lisäämiseen, sillä se antaa uusia mahdollisuuksia tietämyksen muodostamiseen esimerkiksi verkkoteorian menetelmillä. Toistuvien, kiinnostavien tai funktionaalisten hahmojen löytäminen lisää mahdollisuuksia datan tulkintaan ja tiivistämiseen. Tutkittavan verkon hahmojen eli alirakenteiden etsintätekniikkaa voidaan käyttää löytämään likimääräisiä, sumeita hahmoja, jotka ovat jollakin editointimitalla arvioituna kiinnostavia ja ovat silti editointietäisyydelle annetun kynnysarvon sisällä. Tekniikalla voidaan myös tiivistää dataa käyttäen Okkamin partaveitsenä tunnetun säännön Bayesilaista muunnosta nimeltään MDL-periaate. MDL-periaatteen mukaan koko tietojoukon kuvauksen pituuden minivoiva teoria myös kuvaa sen parhaiten. Hahmonsovitustekniikalla voidaankin muodostaa hierarkisia alirakennekuvauksia. Vaikka MDL-periaate on hyödyllinen etsittäessä alirakenteita, jotka tiivistävät datan maksimaalisesti, on usein hyödyllistä käyttää tietyn alueen tietämystä tai oletuksia etsintäprosessissa. Mallitietämystä ja verkonsovitussääntöjä voidaankin yhdessä MDL-periaatteen kanssa käyttää parantamaan kiinnostavien hahmojen löytymistä. Tässä työssä esitellään Subdue-järjestelmää, joka yhdistää eri keinoja ja menetelmiä hyödyllisten alirakenteiden löytämiseksi. Kun alirakenteet on löydetty, niitä käytetään järjestelmässä tiivistämään dataa sekä abstrahoimaan yksityiskohdat ja siten helpottamaan datan tulkintaa.

1 Johdanto

Kyky tunnistaa kiinnostavia ja toistuvia alirakenteita eli hahmoja on välttämätöntä muodostettaessa tietämystä rakenteisuutta sisältävästä datasta. Tässä työssä esitellään Subdue-menetelmää [CoH94, CHD95, CHD94, DCH95, HoC93, HCD94, CHG99], joka perustuu *minimum description length*-periaatteeseen (MDL-periaate).

MDL-periaate on eräänlainen bayesilainen versio Okkamin partaveitsenä tunnetusta säännöstä, jonka mukaan yksinkertaisin hypoteesi, joka kuvaa datan on myös sen paras kuvaus. On olennaista, että MDL-periaatteen hypoteesi koostuu sekä kuvaussäännöistä että datasta, sillä kuvaussääntöjen pituus ja datan määrä ovat usein kääntäen verrannollisia toisiinsa: pitkillä kuvaussäännöillä saadaan data usein hyvin pieneen tilaan alkuperäiseen verrattuna. Subdue-menetelmä perustuu yhteisten alirakenteiden tunnistamiseen. Tavoitteena on löytää sellaiset alirakenteet, jotka pystyvät tiivistämään datan ja tunnistamaan ne käsitteellisesti kiinnostavat alirakenteet, jotka helpottavat datan tulkintaa.

Alirakenteiden etsintä on siis menetelmä niiden käsitteiden tunnistamiseen, jotka kuvaavat kiinnostavia tai toistuvia alirakenteita rakenteisessa datassa. Kun alirakennekäsite on löydetty, voidaan sitä käyttää yksinkertaistamaan dataa korvaamalla alirakenteen ilmentymiä viittauksella (pointer) löydettyyn käsitteeseen. Löydetyt alirakennekäsitteet sallivat alkuperäisen datan yksityiskohtaisen rakenteen abstrahoinnin ja tarjoaa uusia, relevantteja attribuutteja datan tulkitsemiseen. Alirakenteen etsintä- ja korvausmenetelmän iterointi muodostaa hierarkkisen kuvauksen rakenteisesta datasta löydettyjen alirakenteiden muodossa. Tämä hierarkia tarjoaa vaihtelevia tasoja tulkintaan - niitä voidaan tarkastella tarkemmin data-analyysin tavoitteista riippuen.

Tämä työ jakaantuu seuraavasti. Luvussa 2 esitellään Subdue-järjestelmää ja määritellään alirakenteiden löytämisen ongelma sekä erilaisia vaihtoehtoja sen ratkaisemiseksi. Luvussa 3 syvennetään aiheen käsittelyä ja esitellään sovelluskohtaisen tietämyksen käyttöä alirakenteiden etsinnässä. Luvussa 4 käsitellään etsintämenetelmän skaalautuvuusongelmaa.

2 Alirakenteiden löytäminen

2.1 Alirakenteen etsintäjärjestelmä

Subdue- järjestelmä[CoH94] löytää ne alirakenteet, jotka pakkaavat alkuperäisen datan ja jotka esittävät datan rakenteiset käsitteet. Korvaamalla aikaisemmin löydetyt alirakenteet, Subdue-järjestelmän useampi ajokerta tuottaa hierarkkisen kuvaksen datassa esiintyvistä rakenteisista säännönmukaisuuksista. Subdue-järjestelmä käyttää likimääräistä verkon sovitusta (inexact graph match)[HoC93], joka identifioi samanlaiset, muttei identtiset, alirakenteen ilmentymät. Se myös löytää likimääräisen kahden alirakenteen läheisyysmitan silloin kun se on laskennallisissa rajoissa. MDL-periaatteen lisäksi Subdue-järjestelmä voi käyttää myös muuta taustatietoa sopivia alirakenteita haettaessa.

Subdue-järjestelmä [CHG99] esittää rakenteisen datan nimettynä verkkona (labeled graph). Tämä on perusteltua, sillä laskenta nimetyillä verkoilla on huomattavasti helpompaa kuin nimeämättömillä verkoilla [Wil02]. Alirakenteella tarkoitetaan [Wil02] verkon g sellaista aliverkkoa s (subgraph), jonka muodostavat ne verkon g solmut V ja kaaret E, jotka ovat itse tai joiden loppupisteet ovat aliverkossa s. Datan oliot vastaavat verkon solmuja tai pieniä aliverkkoja. Olioiden väliset suhteet vastaavat verkon suunnattuja tai suuntaamattomia kaaria. Alirakenne on yhdistetty aliverkko esitettynä verkkomuodossa. Tämä toimii syötteenä alirakenteen etsintäjärjestelmälle (ks. Kuva 1) Alirakenteen ilmentymä syötteenä annetussa verkossa on joukko sellaisia kaaria ja solmuja syöteverkosta, jotka täsmäytyvät isomorfisesti alirakenteen graafiseen esitykseen. Ne ovat siis isomorfisia alirakenteen kanssa. Alirakennetta voidaan käyttää tiivistämään verkon esitys (ks. Kuva 2).

Alirakenteiden etsintää motivoivat useat psykologiset tekijät, joita ovat esimerkiksi[HoC93]:

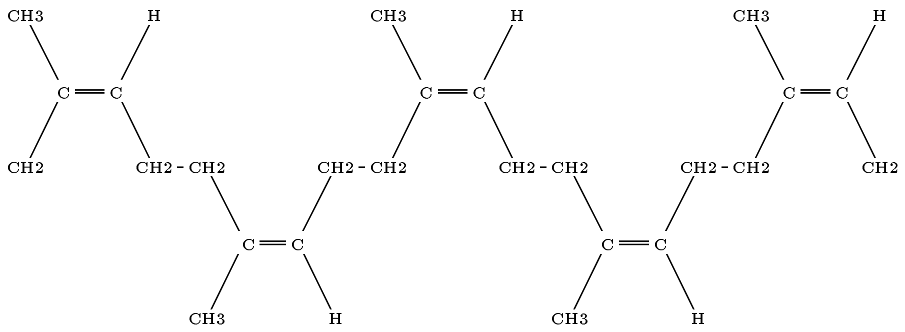
Kognitiiviset säästöt kun verkkorakenne on abstrahoidussa muodossa.

Verkkorakenteen tiiviys määritellään solmujen ja kaarien lukumäärien avulla.

Liittyneisyys mittaa ulkoisten liittymien määrää alirakenteen ilmentymissä.

Kattavuus mittaa alirakenteen kykyä kuvata koko rakennetta.

Subdue-järjestelmän [CoH94] käyttämä alirakenteen etsintäalgoritmi on sädehaku (beam search), jonka toimintaperiaate on esitetty algoritmissa 1. Algoritmi lähtee liikkeelle alirakenteesta, joka täsmäytyy yksittäiseen verkon solmuun v. Algoritmin jokainen läpikäynti



Kuva 1: Luonnollisen kumin atomirakenne [CHG99]

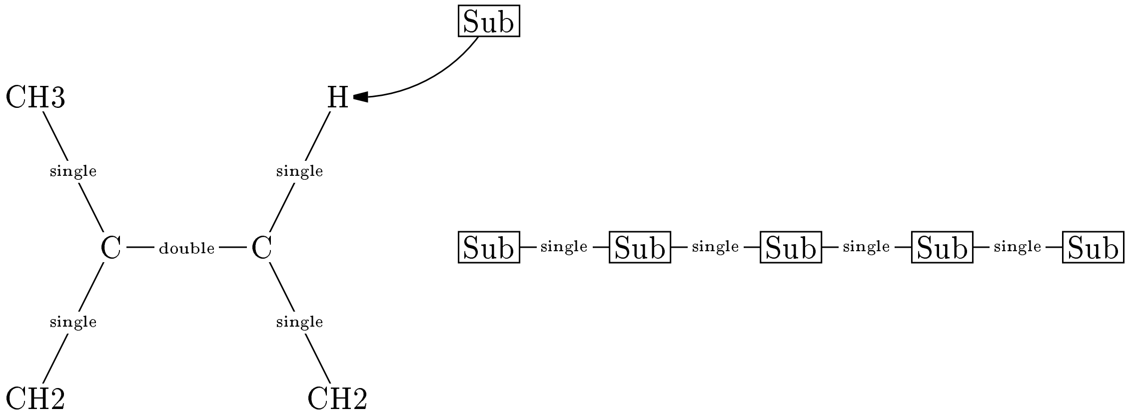
valitsee parhaan alirakenteen ja laajentaa alirakenteen ilmentymiä yhdellä naapurikaarella kaikilla mahdollisilla eri tavoilla. Algoritmi etsii parasta alirakennetta kunnes kaikki mahdolliset alirakenteet on harkittu tai laskennan vaativuudelle annettu kynnysarvo ylittyy.

Algoritmi: SUBDUE(G,kynnysarvo,säde)
while *(laskenta < kynnysarvo) and* $(S \neq \emptyset)$ **do**
    järjestä S parhaimmasta huonoimpaan käyttäen minimikuvauspituutta (MDL) ja taustatietämyssääntöjä;
    S = ensimmäinen S:n alirakenteiden *säde(beam)*;
    b = first(S);
    $D = D \cap \{$b$\}$;
    $E = \{b$ laajennettuna yhdellä kaarella kaikilla mahdollisilla tavoilla$\}$;
    $S = S \cap E$;
end
return D;

Algoritmi 1: Sädehaulle annetaan syötteeksi annetaan verkon solmut G, kynnysarvo sekä säde(beam)[CoH94].

2.2 MDL-periaate

Minimum description length- periaate (MDL-periaate) [Mit97] sanoo, että teoria, joka minimoi koko tietojoukon kuvauksen pituuden myös kuvaa sen parhaiten. MDL-periaate on siis eräänlainen Bayesiläinen versio Okkamin partaveitsenä tunnetusta säännöstä. MDL-periaate suosittelee lyhintä menetelmää opetusaineiston uudelleen koodaamiseen. Lyhyisyyden arvioimisessa mitataan sekä käytetyn hypoteesin koko, että kaikki ylimääräinen hypoteesin aiheuttama datan koodauksen kustannus. MDL-periaate tarjoaa tavan päästä eroon hypoteesin



Kuva 2: Luonnollisen kumin atomirakenteen aliverkon tiivistetty esitys[CHG99]. Verrattuna kuvan 1 esittämään atomirakenteeseen on aliverkkoa pystytty käsitteistämään viiden subkäsitteen muodostamaksi yksinkertaiseksi verkkorakenteeksi.

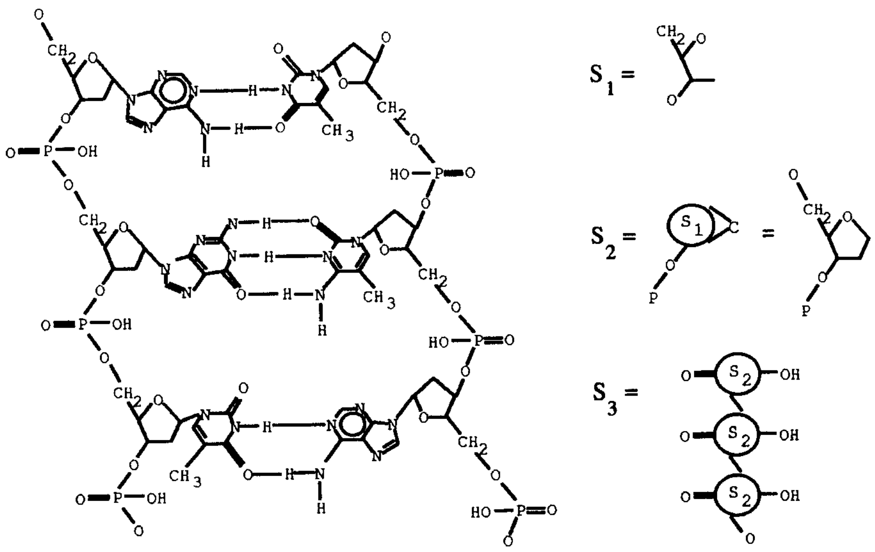
kompleksisuudesta, tosin hypoteesin mahdollisesti aiheuttamien virheiden kustannuksella. Se saattaa valita lyhemmän hypoteesin, joka tekee muutamia virheitä verrattuna pidempään hypoteesiin, joka täydellisesti luokittelee opetusaineiston. Tässä valossa se tarjoaa yhden menetelmän käsitellä datan ylioppimista (overfitting). Ylioppimisessa menetelmä oppii opetusaineiston piirteet liian hyvin, eikä menetelmä tämän jälkeen ole helposti yleistettävissä muille datajoukoille.

MDL-periaatetta voidaan soveltaa monin eri tavoin tiivistämään dataa, esimerkiksi järjestämisalgoritmien kuvauspituuksien laskemisen[DCH95] avulla. MDL-periaatetta voidaan käyttää löytämään alirakenteita myös kompleksisesta biomolekyyliaineistosta[CHG99]. Alirakennetta evaluoidaan tällöin erityisesti sen perusteella, kuinka hyvin se pystyy tiivistämään koko tietojoukon. Verkon lyhin kuvauspituus(MDL) on siis tässä pienin määrä bittejä, jotka tarvitaan kuvaamaan koko verkko. Saavutettu tiivistys määritellään[CoH94] olevan tiivistetyn verkon kuvauspituus(DL) jaettuna alkuperäisen verkon kuvauspituudella.

Subdue-menetelmässä[CHG99] verkon yhteydet kuvataan vierusmatriisilla (adjacency matrix). Tämä on perusteltua [CLR90], mikäli käsiteltävä verkko on tiheä ja mikäli tarvitaan nopeasti tieto siitä, mitä kahta solmua yhdistää kaari. Biomolekyyliaineistoissa verkko voi kuitenkin olla myös harvempi (kuten kuvan 3 DNA-sekvenssin tapauksessa), jolloin on perustellumpaa käyttää vieruslistaa (adjacency list) kuvaamaan verkon yhteyksiä mikäli kaarien lukumäärä on selvästi pienempi kuin solmujen neliö. Esimerkiksi kuvan 3 DNA-sekvenssissä on kaarien lukumäärä $|E| = 27$ ja solmujen lukumäärä $|V|^2 = 729$, jolloin vieruslista olisi ollut tilaa säästävä valinta.

Yhteyslistan käyttö on perusteltua myös siinä tapauksessa, että halutaan säästää muistia, sillä vierusmatriisi vaatii aina $\theta(V^2)$ muistia, riippumatta verkon kaarien lukumäärästä. Subdue-menetelmästä ei myöskään selviä [CHG99] käytetäänkö siinä ns. pakattua rivitallennusta vai koordinaattitallennusta [HHL02].

Menetelmässä oletetaan, että dekooderilla on käytössään alkuperäisen verkon g ainutlaa-



Kuva 3: DNA-rakenteen käsitteistäminen käyttäen Subdue-menetelmää. [CHG99]

tuisista solmujen nimistä (label) koostuva taulu. Tällöin verkon koodaus koostuu seuraavista vaiheista[CHG99]:

1. Selvitä niiden bittien *vbits* lukumäärä, jotka tarvitaan koodaamaan verkon g solmujen nimet.

2. Selvitä niiden bittien *rbits* lukumäärä, jotka tarvitaan koodaamaan vierusmatriisin A rivit. Yhteysmatriisin A rivit ja sarakkeet vastaavat verkon g solmuja.

3. Selvitä niiden bittien *ebits* lukumäärä, jotka tarvitaan vierusmatriisin A alkioiden $A[i, j] = 1$ esittämien kaarien koodaukseen.

Koko verkon koodaus vie $(vbits + rbits + ebits)$ bittiä. [CHG99]

2.3 Likimääräinen verkonsovitus

Tarkka verkonsovitus määritellään seuraavasti [Ber79]:

Verkonsovitus Oletetaan yksinkertainen graafi g = (X, E). Tällöin täsmäys (match) määritellään olevan joukko E_0 s.e. joukon E_0 mitkään kaksi kaarta eivät ole vierekkäin. Jos E_0 on täsmäävä, ja jos $E_1 \subset E_0$, niin myös E_1 on täsmäävä.

Tarkalla verkonsovituksella ei kuitenkaan saavuteta[CHG99] kovinkaan hyviä tuloksia, sillä se antaa vain pienen osan kiinnostavista hahmoista syötteenä annetussa verkossa. Tarkka verkonsovitus siis vaatii, että verkot g_1 ja g_2 ovat isomorfisia, vaikka esiintyisivätkin eri muodoissa (Kuva 4). Monet kiinnostavimmat alirakenteet näyttäytyvät datassa erilaisessa muodossa kuin osataan etsiä. Erot mallissa ja varsinaisessa aineistossa selittyvät esimerkiksi mittausmenetelmien vaillinaisuudesta, joka aiheuttaa kohinaa. Likimääräisessä verkonsovituksessa (inexact graph match) etsitään sellaisia aliverkkoja (siis hahmoja), jotka valittua metriikkaa käyttäen eniten muistuttavat haettavia hahmoja. Metriikka antaa siis mallirakenteen ja löydetyn alirakenteen välisestä eroavasuudesta kertovan etäisyysmitan. Subdue-järjestelmän[CHD95] kehittäjät ovat perustaneet menetelmänsä Bunken ja Allermanin [BuA83] likimääräiseen verkonsovitusmenetelmään. Sovituksessa jokaiseen verkon muuntumaan (distortion) liitetään valittu kustannus.Muuntuma kuvataan seuraavien editointioperaatioiden avulla:

E1. Poisto : Verkon g solmun v tai kaaren e poisto.

E2. Lisäys : Solmun v tai kaaren e lisäys verkkoon g.

E3. Muutos : Verkon g solmun v nimen l (label) tai kaaren e päivitys.

Käyttäjä voi määritellä eri muuntumien kustannukset ja painottaa siten sovitusten etsintää tiettyihin muuntumatyyppeihin ja samalla siis vähentää etsintää toisten muuntumatyyppien kohdalla. Biomolekyyliaineistossa tästä on selvää etua, sillä pienetkin molekyylirakenteessa tapahtuvat muunnokset saattavat muuttaa aineen koostumusta olennaisesti eivätkä ne siten enää ole kiinnostavia hahmonsovituksen kannalta.

Verkkojen g_1 ja g_2 välinen likimääräinen verkonsovitus olettaa aluksi, että verkko g_2 on verkon g_1 muuntunut versio. Formaalisti likimääräinen verkonsovitus on kuvaus

$$f : N_1 \rightarrow N_2 \bigcup \lambda,$$

missä N_1 ja N_2 ovat verkon g_1 ja verkon g_2 vastinsolmuista koostuvia joukkoja. Solmu $v \in N_1$ kuvattuna λ:lle poistetaan, toisin sanoen $f(v) = \lambda$. Tällöin sillä ei siis ole vastaavaa solmua verkossa g_2.

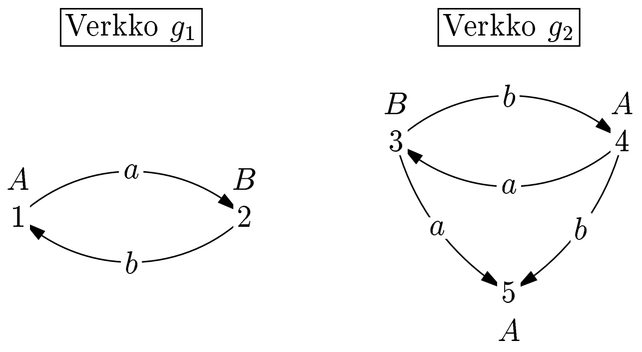
Oletetaan nyt, että tietyt muuntumakustannukset on määritelty käsiteltävälle verkolle. Tällöin likimääräisen verkonsovituksen kokonaiskustannus $cost(f)$ määritellään kaikkien kuvauksesta f saatujen yksittäisten muunnosten (eli editointioperaatioiden) summana. Lisäksi määritellään $matchcost(g_1, g_2)$, jonka arvo on vähiten kustannuksia aiheuttava f, joka kuitenkin kuvaa verkon g_1 verkolle g_2. Jos g_1 on isomorfinen verkon g_2 kanssa, niin kustannus on 0. Sovituksen kustannus kasvaa siis samalla kun verkkojen g_1 ja g_2 välisten eroavaisuuksien määrä kasvaa. Kokonaiskustannus $matchcost(g_1, g_2)$ voidaan selvittää käyttäen puuhakua. Hakupuun yksi tila (state) vastaa osittaista täsmäystä, joka kuvaa g_1:n solmujen alijoukon vastaavasti verkon g_1 solmujen alijoukkoon. Hakumenetelmä koostuu seuraavista vaiheista: (ks. Kuva 5)

1. Lähdetään liikkeelle hakupuun juuresta tyhjästä kuvauksesta

2. Tilan laajentaminen vastaa solmuparin lisäämistä tähän mennessä rakennettuun kuvaukseen. Solmuparilla tarkoitetaan siis yhtä solmua molemmista verkoista g_1 ja g_2.

3. Viimeinen tila hakupuussa on täsmäys, joka kuvaa kaikki g_1:n solmut g_2:een tai λ:an s.e mille tahansa kaarelle g_1:n solmujen välissä on vastaavien solmujen välissä kaari myös g_2:ssa.

Solmujen viereen merkityt numerot kuvassa 5 esittävät kunkin tilan kustannuksia. Tässä esimerkissä kaikille editointikustannuksille on annettu arvo 1. Koska olemme lopulta kiinnostuneita pienimmän kustannuksen omaavasta kuvauksesta, saa jokainen tila hakupuussa arvokseen esittämänsä osittaisen kuvauksen kustannuksen. Täten hakupuumenetelmän tavoitteena on se lopputila, jolla on pienin kustannus. Kuvasta 5 voidaan päätellä, että halvin likimääräisen verkonsovituksen täsmäys g_1:n ja g_2:n välillä saadaan aikaan kuvauksella $f(1) = 4$, $f(2) = 3$. Tämän kuvauksen kustannus on 3.



Kuva 4: Keskenään isomorfiset verkot g_1 ja g_2 [CHG99]

Olkoon g_1 verkko, jossa on n solmua ja g_2 verkko, jossa on m solmua ja $m > n$. Tällöin täydellisen likimääräisen verkontäsmäyksen vaativuus on luokkaa $O(n^{m+1})$. Koska tätä menetelmää käytetään paljon etsintä- ja evaluointiprosesseissa, voi algoritmin vaativuus merkittävästi huonontaa koko järjestelmän suorituskykyä.

Verrattuna alkuperäiseen Bunken ja Allermanin[BuA83] menetelmään on Subduen käytössä olevan likimääräisen verkonsovitusalgoritmin suorituskykyä parannettu soveltamalla *branch-and-bound* -hakua puuhun. Kustannus puun juuresta annettuun solmuun lasketaan kuten yllä on kuvattu. Järjestys, missä solmuja arvioidaan pariutusta varten, riippuu siitä, miten paljon solmusta lähtee kaaria. Toisin sanoen ne solmut, joiden haarautumisaste on suurin, arvioidaan ensin. Tämä rajoittaa jatkossa tarvittavien täsmäystarkistusten lukumäärää.

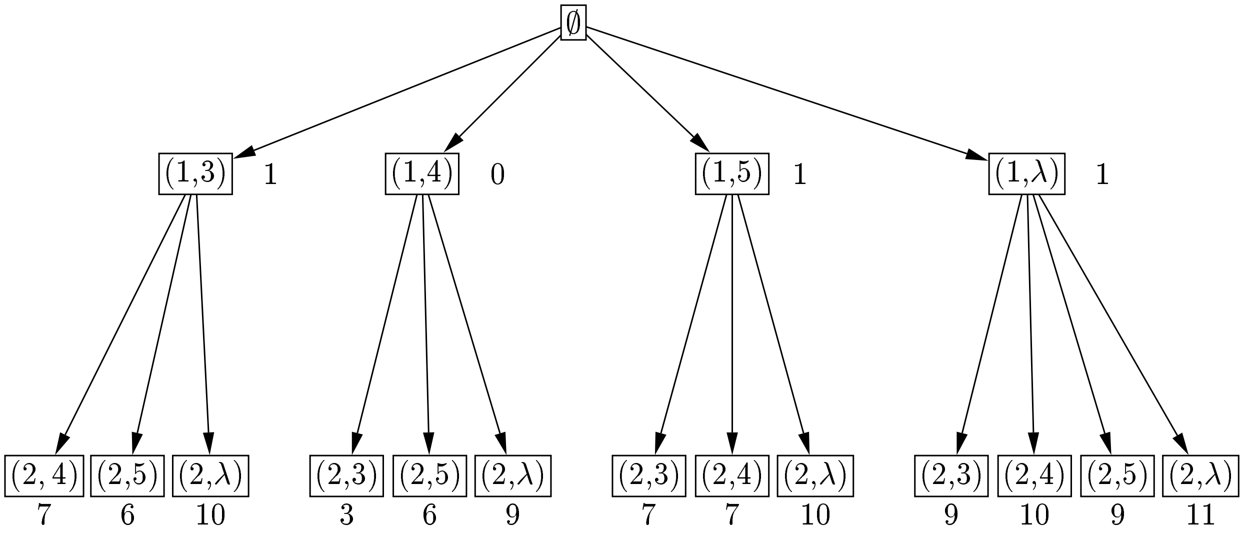
Lisäksi käyttäjä voi antaa ylärajan tutkittavien solmujen määrälle *branch-and-bound* -menetelmässä. Yläraja määritellään syötteenä saatavien verkkojen funktiona. Kun laajennettavien solmujen lukumäärä verkossa saavuttaa annetun rajan, haku turvautuu vuorikiipeilymenetelmään (hill climbing) ja käyttää siihen mennessä tehdyn kuvauksen kustannuksia mittana valitessaan parasta solmua leveyssuunnassa. Tälläisen rajan määrittely mahdollistaa merkittävän nopeutuksen, mutta se vähentää $matchcost(g_1, g_2)$:n laskennan tarkkuutta.

3 Sovellusaluekohtaisen tietämyksen käyttö alirakenteiden etsinnässä

Alirakenteiden etsintäjärjestelmä Subdue kehitettiin alunperin[CHG99] käyttäen vain aihealueriippumatonta (domain independent) heuristiikkaa potentiaalisten alirakenteiden evaluoimiseksi. Tästä seuraa se, että jotkut löydetyt alirakenteet eivät ehkä ole käyttökelpoisia ja relevantteja tietyillä sovelluskohtaisilla kiinnostuksen kohteena olevilla alueilla.

3.1 Mallitietämys

Vaikka MDL-periaate on hyödyllinen etsittäessä alirakenteita, jotka tiivistävät datan maksimaalisesti, on usein hyödyllistä käyttää tietyn alueen tietämystä tai oletuksia etsintäprosessissa. Mallitietämys tarjoaa [CHG99][DCH95] etsintäjärjestelmälle erikoisrakennetyyppejä,



Kuva 5: Hakupuu $matchcost(g_1, g_2)$:n laskemiseksi [CHD95]

jotka suurella todennäköisyydellä esiintyvät tietokannassa ja jotka ovat järjestelmää käyttävän tutkijan erityisen mielenkiinnon kohteena. Mallitietämys on organisoitu hyvin tunnistettujen alkeisrakenteiden hierarkiaan ja niiden yhdistelmiin. Esimerkiksi ohjelmointikielissä tälläisiä malleja ovat esimerkiksi iterointi, järjestäminen, ehtolausekkeet jne. Mallintajalla tulee olla [HHL02] selkeä käsitys mallissa esiintyvistä suureista ja mallinnuksen tavoitteista, jotta hän pystyy tuottamaan tiettyyn hakutehtävään parhaiten sopivia malleja. Mallitietämystä käytettäessä vaarana on helposti se, että mallit ohjaavat liikaa hakuprosessia eli *mitä etsii, sen havaitsee.*

Vaikka MDL-periaate ohjaa myös mallitietämyksen käytössä etsintäprosessia[DCH95], käytetään mallitietämystä prosessin painottamisessa tiettyihin alirakenteisiin. Subdueta voidaan muuttaa keskittämään etsintää mallihierarkiassa määriteltyihin rakennetyyppeihin. Etsintäprosessi alkaa yksittäisen syötteenä annetun verkon solmun sovituksella mallitietämyshierarkian alkeissolmuihin (primitive nodes). Jos alkeissolmut eivät täsmää syötesolmuihin, jatketaan hierarkian korkeamman tason solmuihin. Syöteverkon solmujen kanssa täsmäytyneet mallisolmut tunnistetaan ja niiden edustamat hierarkian mallit valitaan kandidaattimalleiksi, joita yritetään seuraavaksi täsmäyttää syöteverkon alirakenteisiin. Jokaisella prosessin iteraatiokierroksella Subdue valitsee syöteverkosta alirakenteen, joka täsmää parhaiten yhteen valituista malleista ja jota voidaan käyttää syöteverkon tiivistämiseen.

Täsmäys voi olla joko alirakenteen täsmäys tai koko verkon täsmäys. Jos se on alirakenteen täsmäys, niin Subdue laajentaa parhaan alirakenteen ilmentymiä yhdellä naapurikaarella kaikilla mahdollisilla tavoilla. Tällä tavoin tuotetusta alirakenteesta tulee seuraavan iteraatiokierroksen kandidaattimalli. Jos koko verkko täsmää, prosessi on löytänyt halutun alistruktuurin, jota voidaan sitten käyttää tiivistämään koko syöteverkko. Prosessi etenee kunnes alirakenne on löydetty tai kun kaikki mahdolliset alirakenteet on käyty läpi.

Syöteverkon esittäminen käyttämällä mallihierarkiasta löydettyä alirakennetta vaatii ylimääräistä työtä, kun alirakenteen ilmentymät korvataan osoittimella mallihierarkiaan. Joissakin tapauksissa mallin määrittely sisältää myös parametrejä, jotka pitää kuvata. Kun alirakenne on löydetty, sen jokainen ilmentymä korvataan osoittimella mallihierarkian sitä esittävään käsitteeseen.

3.2 Verkon sovitussäännöt

Subduen keskeinen komponentti on likimääräinen verkon sovitusalgoritmi[CHG99], joka etsii isomorfisia alirakenteen ilmentymiä. Koska monet alirakenneilmentymät voivat esiintyä hieman erilaisessa muodossa datassa, ja koska jokainen näistä eroavaisuuksista kuvataan verkonsovitusmenetelmän tekeminä perusmuunnoksina, voidaan verkon sovitussääntöjä käyttää liittämään jokaiseen muunnokseen käyttöalueeseen perustuva kustannus. Tämän tyyppistä sovellusalaan liittyvää informaatiota esitetään käyttäen if-then sääntöjä. Säännöt voidaan esittää esimerkiksi algoritmin 2 esittämällä tavalla.

```
Syöte: Molekyylirakenne, esimerksi DNA
if sovellusalue on biokemia then
    if ((Solmu1 = HO and Solmu2 = OH) or (Solmu1 = OH and Solmu2 = HO))
    then
        Solmun kustannus = z;
    end
end
```

Algoritmi 2: Verkontäsmäyssääntöjen käyttäminen kustannusten laskemisessa[CHG99]

Verkonsovitussäännöt mahdollistavat sen, että voidaan määritellä hyväksyttävä määrä yleistyksiä alirakennemäärittelyn ja sen ilmentymien välillä syöteverkossa. Jos järjestelmälle annetaan verkot g_1 ja g_2 ja joukko muunnoskustannuksia, samanlaisuuden laskeminen voidaan tehdä yllä olevaa hakumenetelmää käyttäen. Niin kauan kuin samanlaisuus on käyttäjän määrittelemän kynnysarvon sisäpuolella, ovat verkot g_1 ja g_2 isomorfiset.

Subduen kehittäjät ovat testanneet[CHG99] Subduen suorituskykyä käyttäen mallitietämystä, verkonsovitussäännöillä sekä ilman taustatietämystä. Suorituskyvyn metriikkana käytetään tiivistyksen määrää, joka saavutetaan parhaimmaksi arvioidun alirakenteen avulla ja sen funktionaalisuuden määrällä. Funktionaalisuusmetriikan laativat joukko sovellusalaa tuntevia asiantuntijoita ja se esitään asteikolla 1 - 5. Paras funktionaalisuus ja tiivistys saavutettiin käyttämällä sekä malli-, että verkkosovitussääntötietämystä. Pelkästään verkonsovitussääntöjä käyttäen saatu tiivistys jäi kuitenkin pienemmäksi kuin ilman mitään taustatietämystä tehty tiivistys.

4 Skaalautuvuus

Tietämyksen muodostaminen tietokannoista (Knowledge Discovery in Databases, KDD) on ala, jonka tavoitteena on löytää tietämystä suurista tietokannoista, joita ei voida tehokkaasti käydä läpi ihmisvoimin[CHG99]. Tästä syystä KDD- järjestelmiltä vaaditaan kykyä hallita erittäin suuria tietokantoja. Valitettavasti useimmat KDD-järjestelmät ovat laskennallisesti vaativia ja useimmissa tapauksissa KDD-järjestelmän resurssivaatimukset kasvavat tietokannan kasvaessa. Näistä syistä johtuen tutkijoiden huomio on vaihtumassa uusien KDD-järjestelmien kehittelyn sijasta olemassaolevien KDD-järjestelmien skaalautuvuuden parantamiseen.

KDD-järjestelmän, kuten Subduen[CHG99] skaalautuvuutta voidaan parantaa muuttamalla järjestelmää hyväksymään suurempia syötteitä. Tämän sijasta tai tämän lisäksi voidaan lyhentää ajoaikaa. Yhden koneen resurssien lisääminen ei kuitenkaan selvästikään riitä; työ on voitava jakaa useammalle koneelle ja tässä tarvitaan rinnakkaisalgoritmeja. Monimutkaisen järjestelmän siirtäminen rinnakkaisympäristöön sisältää sekä ongelmaan itseensä liittyviä kysymyksiä että kannanottoa arkkitehtuurikysymyksiin.

Useat eri menetelmät voivat jakaa laskennan usean eri yksittäisen prosessorin kesken. Näissä menetelmissä on kuitenkin usein se huono puoli, että ne tallentavat koko tietokannan jokaiselle mukana olevalle koneelle. Subduessa on haluttu tarjota skaalautuvuus sekä tallennuksen että laskennan suhteen. Tämän johdosta kirjoittajat esittelevät datan osittelu- lähestymistavan (data partitioning) nimeltään Static-Partitioning Subdue (SP-Subdue)[CHG99].

SP-Subdue-menetelmää[CHG99] käyttäen syöteverkko jaetaan n:ään eri osaan n:lle prosessorille. Jokainen prosessori ajaa sarjallisen Subdue-algoritmin paikallisesa verkko-osassaan ja lähettää parhaat löytämänsä alirakenteet muille prosessoreille. Tämän jälkeen jokainen prosessori evaluoi välitetyt alirakenteet paikallisessa osiossaan. Kun kaikki evaluoinnit on suoritettu loppuun saakka, keskusprosessori kerää tulokset ja selvittää koko aineiston parhaat löydöt.

Algoritmin tärkein vaihe on verkon ositus - siitä riippuu algoritmin nopeutuminen sekä löydettyjen alirakenteiden laatu [CHG99]. Verkon osittamisvaiheessa on haluttu tasapainottaa työmäärää tasaisesti prosessorien kesken, ja samalla säilyttää niin paljon informaatiota kuin mahdollista. Osituksessa katoaa usein paljon oleellista tietoa osien välisten kaarien mukana. SP-Subdue käyttää verkon ositusohjelmaa nimeltään Metis. Metis hyväksyy painotetun verkon, jossa kaarille ja solmuille on annettu jokin painoarvo. Metis pyrkii osittamaan verkon minimoimalla niiden kaarien painojen yhteenlasketun summan, joiden kohdalta verkko ositetaan. Näitä kaaria voidaan kutsua esimerkiksi leikkauskaariksi (cut edge). Painoja liitetään vain kaariin, jotta jokaisessa verkon osassa on osituksen jälkeen suunnilleen yhtä monta solmua. Kaarten painot määrätään siten, että mitä useammin kaaren nimi (label) esiintyy, sen korkeampi on sen saama painoarvo. Tämän painotuksen ideana on se, että useammin esiintyvien kaarien nimet esittävät todennäköisemmin käyttökelpoista tietämystä.

5 Yhteenveto

Paranevien biologisen datan mittaus- ja tallennusmenetelmien ansiosta biologista aineistoa sisältävien tietokantojen rakenteisuus on lisääntynyt. Tämä antaa tietämyksen muodostamisen algoritmeille mahdollisuuden käyttää rakenteisuutta hyväkseen, ja jopa vaatii sitä parempien mielenkiintoisten hahmojen löytämiseksi. Tässä työssä esiteltiin erästä verkkoteoriaan ja MDL-periaatteeseen perustuvaa järjestelmää nimeltään Subdue, joka pyrkii kuvaamaan biomolekyylidatan siinä esiintyvien alirakenteiden avulla eli mahdollisimman yksinkertaisesti. Menetelmän perustuu likimääräiseen verkonsovitusmenetelmään. Subdueta voidaan ohjata myös mallitietämyksen sekä verkonsovitussääntöjen avulla etsimään kiinnostavimpia hahmoja. Riskinä on kuitenkin se, että mallien käyttö ohjaa liikaa etsintäprosessia. Subduen kehittäjät ovat MDL-periaatteesta ponnistaessaan kiinnostuneimpia tiivistämään aineistoa, ja siksi eri metriikalla kiinnostavat hahmot, jotka eivät pysty kuitenkaan tiivistämään dataa jäävät löytymättä. Subdue-järjestelmän kehittäjät eivät myöskään perustele vierusmatriisin käyttöä (vieruslistan sijasta) verkon solmujen kuvaamisessa. DNA molekyylirakenteen tallennuksessa vieruslistan käyttö on tallennuksessa perustellumpaa.

Viitteet

[Ber79] Berge, C. Graphs and HyperGraphs. *North-Holland Publishing Company, Netherlands. Second edition*, 1979.

[BuA83] Bunke H., G. Allerman. Inexact Graph Matching for Structural Pattern Recognition. *Pattern Recognition Letters 1(4):245-253.*, 1983.

[CHG99] Cook D.J., L. B. Holder, G. Galal: Discovering Concepts in Structural Data. Kappale 9 teoksessa J. T. L. Want, B. A. Shapiro, D. Shasha: Pattern Discovery in Biomolecular Data, Tools, Techniques and Applications. *Oxford University Press*, 1999.

[CoH94] Cook D. J., L. B. Holder. Substructure Discovery Using Minimum Description Length and Background Knowledge. *In Journal of Artificial Intelligence Research, Volume 1, pages 231-255*, 1994.

[CHD95] Cook D. J., L. B. Holder, and S. Djoko. Knowledge Discovery from Structural Data. *Journal of Intelligence and Information Sciences, Volume 5, Number 3, pages 229-245*, 1995.

[CHD94] Cook D. J., L. B. Holder, and S. Djoko. Scalable Discovery of Informative Structural Concepts Using Domain Knowledge. *In IEEE Expert, Volume 11, Number 5, pages 59-68*, 1996.

[CLR90] Cormen, C. H., C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms. *McGraw-Hill*, 1990.

[DCH95] Djoko S., D. J. Cook and L. B. Holder. Analyzing the Benefits of Domain Knowledge in Substructure Discovery. In the *Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pages 75-80*, 1995.

[HHL02] Haataja J., Heikonen J., Leino Y., Rahola J., Ruokolainen J., Savolainen V. Numeeriset menetelmät käytönnössä. *CSC- Tieteellinen laskenta*, 2002.

[HoC93] Holder L. B., D. J. Cook. Discovery of Inexact Concepts from Structural Data. In *IEEE Transactions on Knowledge and Data Engineering, Volume 5, Number 6, pages 992-994*, 1993.

[HCD94] Holder L. B., D. J. Cook and S. Djoko. Substructure Discovery in the SUBDUE System. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases, pages 169-180*, 1994.

[Mit97] Mitchell, T. Machine Learning. *McGraw-Hill*, 1997.

[Wil02] Wilf S.H. Algorithms and complexity. *A K Peters*, 2002.