BLOCKS-lohkotietokanta

Juho Snellman
juho.snellman@cs.helsinki.fi

Tiedon louhinta biomolekyyliaineistoista
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Raportti C–2003–52, s. 11–20, marraskuu 2003

Tiivistelmä

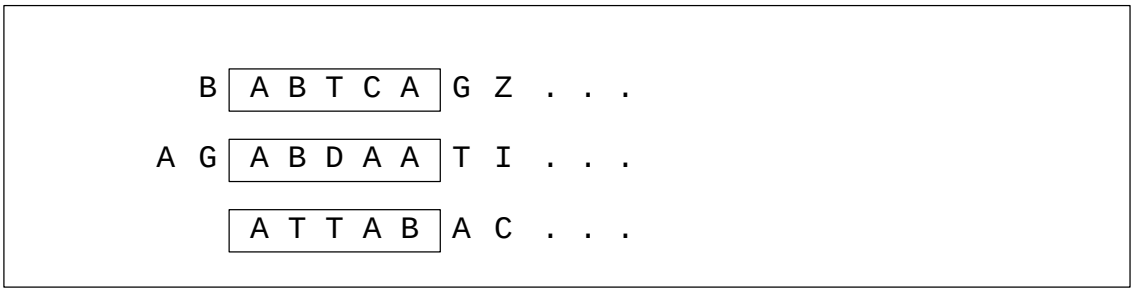
Seminaariraportissa esitellään BLOCKS-tietokannan rakentamiseen käytetty algoritmi, jolla muodostetaan tietokanta proteeiniperheiden sisäisistä lohkoista, eli aukottomista monen sekvenssin paikallisista rinnastuksista (*ungapped multiple local alignment*). Lohkot kuvaavat alueita, jotka ovat samankaltaisia perheen sekvensseissä. Algoritmin päävaiheet ovat lohkojen etsiminen MOTIF-algoritmilla, lohkojen pisteiden optimoiminen, ja parhaan yhteensopivan lohkojoukon kokoaminen. Lisäksi raportissa kuvaillaan toinen algoritmi, jolla luokitellaan tuntemattomia sekvenssejä lohkotietokannasta löytyviin proteiiniryhmiin. Jälkimmäinen algoritmi perustuu tutkittavan sekvenssin pisteyttämiseen kaikkia tietokannan lohkoja vastaan, ja tästä saatujen tulosten merkittävyyden arviointiin hypergeometrisen jakauman avulla.

1 Johdanto

Useimmissa proteiiniperheissä tietyt sekvenssien alueet ovat evoluution kuluessa muuttuneet muita alueita vähemmän. Nämä muuttumattomat alueet (*conserved regions*) ovat pysyneet samanlaisina esimerkiksi, koska ne ovat tärkeitä proteiinin toiminnan tai rakenteen kannalta. Kun tällaisen perheen jäsenten sekvensseistä etsitään samankaltaisuuksia, löydetään todennäköisesti juuri tällaisia säilyneitä alueita. Lisäksi voidaan olettaa, että samat alueet ovat säilyneet suhteellisen muuttumattomina myös useissa muissa kyseisen perheen sekvensseissä [HeH96].

Eräs menetelmä sekvenssien samankaltaisuuksien kuvaamiseen on aukoton monen sekvenssin paikallinen rinnastus (*ungapped multiple local alignment*) eli lohko (kuva 1). Paikallisessa rinnastuksessa sekvensseistä etsitään samankaltaisia alisekvenssejä. Monen sekvenssin rinnastuksella puolestaan tarkoitetaan sitä, että tarkoituksena on löytää useasta sekvenssistä alueet, jotka ovat keskenään samankaltaisia. Aukottomassa rinnastuksessa alueet koostuvat ainoastaan merkeistä, erotuksena aukolliseen rinnastukseen, jossa alueet voisivat sisältää merkkien lisäksi usean merkin kokoisia aukkoja, joilla sekvenssien väliset yhtäläisyydet saadaan osumaan kohdalleen mahdollisimman tarkasti [DEK98].

Tämän seminaariraportin luvussa 2 kuvataan menetelmä, jolla rakennetaan etukäteen luokitellusta sekvenssidatasta lohkoja sisältävä tietokanta. Luvussa 3 esitellään menetelmä, jolla voidaan luokitella tuntemattomia sekvenssejä niihin luokituksiin, joita käytettiin tietokantaa luotaessa.



Kuva 1: Kolmesta sekvenssistä löydetty lohko (aukoton monen sekvenssin paikallinen rinnastus).

2 Tietokannan rakentaminen

Tässä luvussa esitellään BLOCKS-lohkotietokannan rakentamiseen käytetty menetelmä. Lohkotietokanta rakennetaan antamalla algoritmille syötteeksi proteiiniperheisiin luokiteltuja sekvenssejä. Kunkin proteiiniperheen sisältä etsitään mahdollisimman hyvä joukko keskenään yhteensopivia lohkoja, jotka kirjoitetaan kyseisen perheen tietueeseen. Etsintäalgoritmi on kolmivaiheinen. Ensimmäisessä vaiheessa (luku 2.1) etsitään ryhmän sekvensseistä lohkoja. Toisessa vaiheessa (luku 2.2) optimoidaan ensimmäisessä vaiheessa löydettyjen lohkojen pistemääriä. Kolmannessa vaiheessa (luku 2.3) kootaan optimoiduista lohkoista paras joukko, jossa joukon lohkot ovat kaikissa syötesekvensseissä samassa järjestyksessä [HeH91].

Kuvattava algoritmi on pääpiirteiltään pysynyt samana 90-luvun alusta, joskin sen C-kieliseen toteutukseen on lisätty vaihtoehtoisia menetelmiä joidenkin työvaiheiden suorittamiseen. Kyseinen toteutus on alunperin tarkoitettu ajettavaksi pienitehoisilla työasemilla tai PC-koneilla, joten algoritmin suorituskyky on selvästi ollut yksi tärkeimmistä suunnittelukriteereistä alusta alkaen. Käsiteltävät tietomäärät kasvavat jatkuvasti; esimerkiksi Swiss-Prot tietokanta, jota käytettiin BLOCKS-tietokannan lähdemateriaalina [HeH91] on kasvanut yli lineearista vauhtia ensimmäisestä vuonna 1986 julkaistusta versiosta alkaen [SP03]. Siksi on mielestäni todennäköistä, että suorituksen nopeuttamiseksi tehdyt yksinkertaistukset algoritmiin ovat edelleen mielekkäitä.

Abstraktilla tasolla lohkon tietosisältö on suhteellisen yksinkertainen. Lohko koostuu pituudesta, sekä joukosta indeksi/sekvenssi-pareja, joissa indeksi kertoo mistä kohtaa kyseistä sekvenssiä lohko alkaa. Varsinaisessa toteutuksessa tietorakenne sisältää paljon muutakin tietoa, esimerkiksi lohkon pistearvon, sekvenssien painoarvot, lohkokohtaiset pisteytysmatriisit, ja sekvenssikohtaisen etäisyyden muihin lohkoihin.

Algoritmin kuvauksessa mainitaan useita maagisia vakioita, kuten esimerkiksi MOTIF-algoritmin aukkojen maksimipituus tai 60 merkin rajoitus lohkojen koolle. Osaa näistä arvoista ei ole mitenkään perusteltu löytämissäni lähteissä, tai perustelut eivät ole erityisen selviä. Todennäköisesti vakioiden arvot on asetettu iteratiivisesti antamalla niille lähtöarvot, tutkimalla saatuja tuloksia, ja muuttamalla arvoja kunnes suorituskyvyn ja tulosten laadun välinen tasapaino oli sopiva. Näiden maagisten vakioiden käyttö on mielestäni algoritmin epämiellyttävin osuus, eikä niiden tarkkoihin arvoihin kannata kiinnittää erityisesti huomiota raporttia luettaessa.

2.1 Lohkojen etsiminen

Tietokannan rakentaminen alkaa lohkojen etsimisellä annetusta sekvenssidatasta. Tähän työvaiheeseen on olemassa useita eri algoritmeja, joista oletusarvoisesti käytetään muunneltua versiota Smithin, Annaun ja Chandrasegaranin [SAC90] MOTIF-algoritmista. MOTIF palauttaa suuren määrän suhteellisen epävarmoja motiiveja, kun taas useimmat muut vastaavat algoritmit palauttavat vähemmän, mutta laadukkaampia tuloksia [HeH91]. Myöhemmin kuvattava parhaan lohkojoukon kokoamiseen käytetty algoritmi toimii parhaiten ensin mainitussa tilanteessa. Muita algoritmeja lohkojen etsintään voidaan kuitenkin käyttää esimerkiksi tulosten varmentamiseen [HHA95].

MOTIF-algoritmi etsii sekvenssidatasta kolmesta merkistä ja niitä erottavasta kahdesta aukosta muodostuvia motiiveja (esim. `A-C--T`). Aukkojen maksimipituus on rajattu etukäteen tehokkuussyistä, MOTIF-algoritmin oletusasetus tälle parametrille on 10. BLOCKS-tietokantaa rakennettaessa käytettiin suhteellisen suurta maksimipituutta 17, jolloin potentiaalisia lohkoja löytyy enemmän kuin oletusasetuksilla [HeH91].

Algoritmi käy läpi sekvenssit, ja listaa jokaista sekvenssiä kohti kaikki kyseisen sekvenssin sisältämät motiivit. Jokaista tällä tavalla löydettyä motiivia kohti talletetaan taulukkoon tieto siitä, mistä kyseinen motiivi löytyi. Taulukon indeksit johdetaan motiivin merkeistä ($A = 1\ldots$) ja aukkojen pituuksista niin, että jokaiselle motiiville tulee yksiselitteinen indeksi [SAC90]. Sekvenssiä ei huomioida indeksiä laskettaessa, joten samaan taulukon soluun voidaan tallettaa tietoa monesta tietyn motiivin ilmentymästä. Esimerkiksi 20 eri merkillä ja korkeintaan 17 merkin pituisilla aukoilla suurin indeksi olisi $20 \times 20 \times 20 \times 18 \times 18 = 2592000$ (18, eikä 17, koska myös nollan merkin pituiset aukot on huomioitava). Tämän työvaiheen aikavaatimus on varsin vaatimaton – $O(n \times m)$, jossa n on sekvenssien yhteenlaskettu pituus ja m aukkojen maksimipituus.

Kun kaikki ryhmän sekvenssit on käsitelty, käydään taulukon sisältö läpi, ja karsitaan pois kaikki motiivit, jotka eivät esiintyneet tarpeeksi monessa sekvenssissä (tarkka raja määritetään algoritmin suorituksen yhteydessä). Lisäksi jokaisen samassa sekvenssissä moneen kertaan esiintyvän motiivin ylimääräiset esiintymät poistetaan automaattisesti. Jäljelle jääneistä motiiveista tehdään kyseisten motiivien pituisia lohkoja, jotka sisältävät kaikki sekvenssit joissa motiivi esiintyi [SAC90].

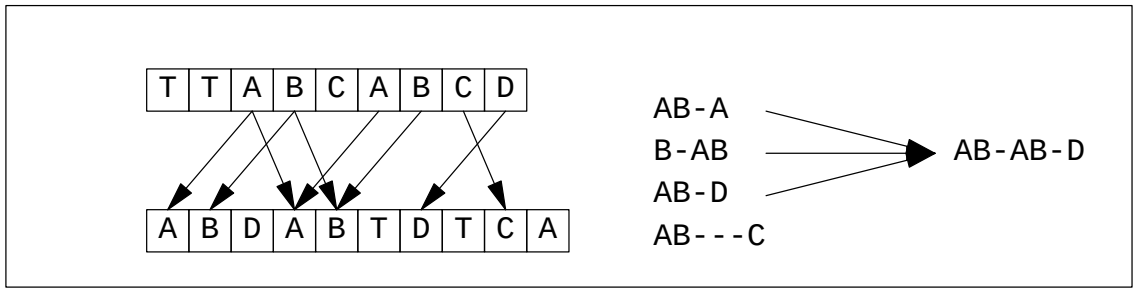
Myöhemmät työvaiheet luottavat siihen, että kaikki joukon lohkot sisältävät samat sekvenssit. Lohkoihin, jotka eivät sisällä kaikkia sekvenssejä, lisätään puuttuvat sekvenssit. Tämä tapahtuu pisteyttämällä[1] lohkoon kuulumattomat sekvenssit kaikissa niiden indekseissä, ja lisäämällä sekvenssit lohkoon parhaan pistemäärän saaneeilla indekseillä. Lopuksi lohkot pisteytetään vielä kerran ja 50 parhaat pisteet saanutta lohkoa välitetään seuraavaan vaiheeseen [SAC90].

2.2 Lohkojen optimointi

Edellisessä vaiheessa löydetyt lohkot ovat ainoastaan suuntaa antavia, joten hyvien tulosten saavuttamiseksi niitä täytyy optimoida. Tämä tapahtuu yhdistämällä päällekkäisiä lohkoja, ja laajentamalla lohkoja molempiin suuntiin.

Lohkot, jotka ovat yhtenäisesti päällekkäisiä kaikissa sekvensseissä, yhdistetään yhdeksi lohkoksi (kuva 2). Lohkojen yhdistämisen tarkoituksena on löytää keskimääräistä

[1] Pisteytykseen käytetään parien summa-menetelmää, normalisoituna lohkon pituuden neliöjuurella, joka muistuttaa myöhemmissä vaiheissa käytettyä pisteytysmenetelmää. Kokonaisalgoritmin toiminnan kannalta ei kuitenkaan ole oleellista, että lohkojen etsimisessä käytettäisiin samaa pisteytystä kuin myöhemmissä vaiheissa.



Kuva 2: Päällekkäiset motiiveista `AB-A`, `B-AB`, ja `AB-D` luodut lohkot yhdistetään. Motiivista `AB---C` luotua lohkoa ei yhdistetä muihin lohkoihin, vaikka se onkin niiden kanssa päällekkäinen, koska päällekkäisyydet eivät ole yhdenmukaisia molemmissa sekvenssissä.

korkealaatuisempia lohkon siemeniä, joita myöhemmin kasvatetaan. Jotta tämä voitaisiin ottaa huomioon myöhemmässä pisteytyksessä, pidetään lohkon tietorakenteessa kirjaa siitä, kuinka se muodostui [HeH91].

Kuten aiemmin mainittiin, on syötesekvenssit etukäteen luokiteltu kuuluviksi samaan proteeiniryhmään. Ryhmän sisällä sekvenssit voivat kuitenkin olla vaihtelevasti sukua toisilleen. Jotta liian samankaltaiset sekvenssit eivät vääristäisi myöhempää pistelaskua, jaetaan rakenteellisesti samankaltaiset sekvenssit ryppäisiin. Jokaisen n:stä sekvenssistä koostuvaan ryppääseen kuuluvan sekvenssin painoarvoksi asetetaan $\frac{1}{n}$ [HeH96].

Lohkoja laajennetaan molempiin suuntiin, korkeintaan 60 merkin kokoisiksi, ja kullekkin lohkolle valitaan korkeimman pistemäärän saanut laajennus. Kun kaikkien lohkojen pistemäärät on optimoitu tällä tavalla, poistetaan liian huonot lohkot. Yksittäisen lohkon pisteiden huonous riippuu muiden lohkojen pisteiden jakaumasta. Huonoiksi määritellään tässä lohkot, joiden pistemäärä on vähintään keskihajonnan verran keskiarvoa alhaisempi [HeH91].

Lopuksi yhdenmukaisesti päällekkäiset lohkot yhdistetään vielä kerran. Yhdistämistä ei kuitenkaan tehdä, jos tuloksena syntyisi yli 60 merkin pituinen lohko, koska liian pitkät lohkot ovat yleensä liian erikoistuneita, eivätkä siten hyödyllisiä hakusovelluksissa. Koska päällekkäiset lohkot eivät voi algoritmin seuraavassa vaiheessa päästä samaan lohkojoukkoon, supistetaan näiden pitkien ja yhdenmukaisesti päällekkäisten lohkojen rajoja, kunnes ne eivät enää ole päällekkäisiä [HeH91].

2.2.1 Pisteytys

Algoritmin käyttämä pisteytysmenetelmä on tavanomainen parien summa (*sum of pairs, SP*), joillakin muutoksilla[2]. Lohkon jokainen sarake pisteytetään erikseen, laskemalla yhteen pistytysmatriisin osoittama arvo sarakkeen kaikkille merkkipareille kerrottuna merkkiparin sekvenssien painoilla (kuva 3):

$$S(c_i) = \sum_{k<l} s(c_i^k, c_i^l) \times w_k \times w_l$$

jossa $S(c_i)$ on sarakkeen i pistearvo, $s(a, b)$ on pistematriisin arvo merkeille a ja b, c_i^k on sarakkeen i sekvenssin k merkki, ja w_s on sekvenssin s painoarvo.

[2]Henikoff [Hen99] väittää, että kyseessä olisi *mean of pairs*. PROTOMAT-järjestelmän lähdekoodista ei kuitenkaan nähdäkseni löydy mitään keskiarvoon viittaavaa, vaan sarakkeiden pisteet lasketaan yllä kuvatulla kaavalla. Koska lohkojen pisteitä verrataan ainoastaan toisiin saman sekvenssiryhmän lohkoihin, ei keskiarvolla uskoakseni edes olisi mitään merkitystä.

```
    A  B  C  D        ... A ... [0.5]
A   5 -1  1 -3        ... B ... [1]
B  -1  3 -2 -2        ... C ... [0.5]
C   1 -2  4  0        ... A ... [1]
D  -3 -2  0  5

                  S = s(A,B)*0.5 + s(A,C)*0.5*0.5 +
                      s(A,A)*0.5 + s(B,C)*0.5 +
                      s(B,A) + s(C,A)*0.5
                    = -1/2 + 1/4 + 5/2 + -2/2 + -1 + 1/2
                    = 3/4
```

Kuva 3: Sarake ABCA pisteytytetään, laskemalla yhteen vasemmalla olevasta pisteytysmatriisista löytyvät arvot kaikille sarakkeen merkkipareille. Ensimmäinen ja kolmas sekvenssi ovat samassa ryppäässä, joten niiden painoarvoja on vähennetty.

Yksinkertainen menetelmä lohkon pisteyttämiseen olisi kaikkien sarakepisteiden laskeminen yhteen. Toisaalta hyvänkin lohkon keskeltä voi löytyä yksittäisiä sarakkeita, jotka saavat erityisen huonoja pisteitä. Tämä ongelma voidaan välttää laskemalla yhteen vain positiiviset sarakepisteet. Jotta isot lohkot eivät olisi yksiselitteisesti pieniä parempia, normalisoidaan lohkon pistemäärä jakamalla se lohkon pituuden neliöjuurella. Lisäksi pistemäärä kerrotaan lohkon painoarvolla, joka määräytyy lohkoa luotaessa yhdistettyjen alkulohkojen määrän perusteella [Hen99].

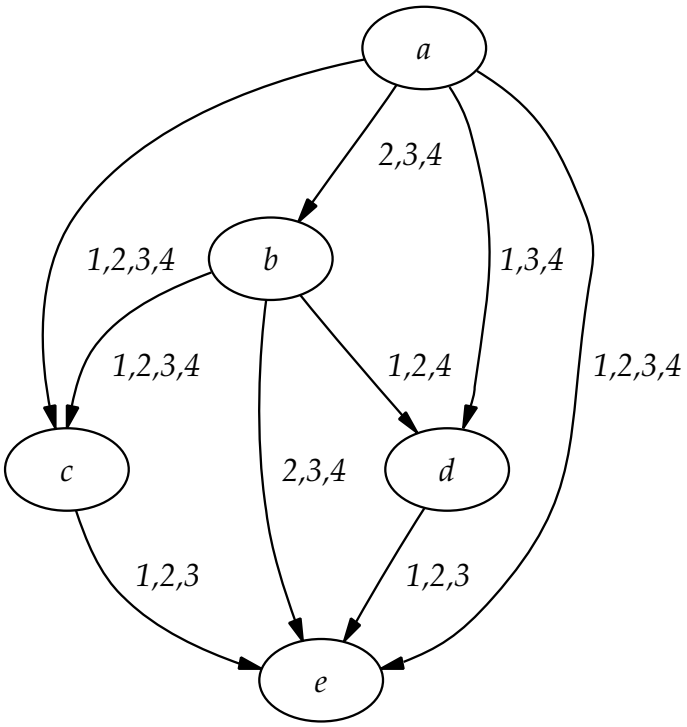
Pisteytys on koko algoritmin toiminnan kannalta oleellisin osuus, ja pisteytyksen laatuun taas vaikuttaa merkittävästi käytetty pisteytysmatriisi. Suuntaa antavia tuloksia voidaan saada yksinkertaisella identiteettimatriisilla, mutta parhaat tulokset saavutetaan aineistoon soveltuvalla pistematriisilla. Esimerkiksi varsinaista BLOCKS-tietokantaa luotaessa käytetään BLOSUM62-matriisia, joka on laskettu BLOCKS-tietokannan sisältämien lohkojen datasta[HeH92].

2.3 Parhaan lohkojoukon kokoaminen

Edellisessä vaiheessa optimoiduista yksittäisistä lohkoista etsitään paras yhteensopivien lohkojen joukko. Yhteensopivuus tarkoittaa, etteivät lohkot ole päällekkäisiä missään sekvenssissä, ja että ne ovat samassa järjestyksessä kaikissa sekvenseissä. Toisaalta ei ole mielekästä valita huonoista lohkoista koostuvaa joukkoa vain siksi, että muuten paremman lohkojoukon lohkot olivat väärässä järjestyksessä yhdessä sekvenssissä. Siksi tietyissä tilanteissa voidaan sekvenssejä tiputtaa pois ryhmästä.

Lohkoista luodaan kehätön suunnattu verkko, jossa jokaista lohkoa kohti on yksi solmu (kuva 4). Lohkon A solmusta luodaan kaari lohkon B solmuun, jos löytyy tarpeeksi monta sekvenssiä, joissa lohkot eivät ole päällekkäisiä ja joissa lohko A esiintyy ennen lohkoa B. "Tarpeeksi monta" on niin sanottu kriittinen määrä, jonka arvo määritellään automaattisesti. Kriittinen määrä on aina yli puolet sekvenssien määrästä, millä varmistetaan, että verkosta tulee kehätön. Jokaiseen kaareen merkitään tieto siitä, missä sekvenseissä kaaren luomiseen tarvittavat ehdot täyttyivät [HeH91].

Verkon kaikki polut käydään läpi, esimerkiksi tavallisella syvyyssuuntaisella haulla. Kaikki polut, joissa kaarien sekvenssijoukkojen leikkaus sisältää kriittistä määrää vähem-



Kuva 4: Esimerkki lohkoista ja niiden keskinäisistä suhteista muodostetusta verkosta. Jos kriittinen sekvenssimäärä on 3, olisivat esimerkiksi polut *abc* ja *de* hyväksyttäviä. Polku *abce* ei kelpaisi, koska ainoastaan sekvenssit 2 ja 3 esiintyvät kaikissa polun kaarissa.

män sekvenssejä, karsitaan pois (kuva 4). Loput polut pisteytetään laskemalla yhteen kaikkien polun sisältämien lohkojen pisteet, ja kertomalla tämä summa polun kaikkiin kaariin kuuluvien sekvenssien määrällä [Hen99]. Lopulta parhaan pistemäärän saaneen polun sisältämät lohkot kirjoitetaan tietokantaan, joskin lohkoista poistetaan sekvenssit, jotka eivät esiintyneet kaikissa polun kaarissa.

3 Tietokannan hyödyntäminen

Lohkotietokantaan kerättyä tietoa voidaan hyödyntää useilla tavoilla, joita Henikoffit ovat kuvanneet [HeH96]. Pääasiallinen käyttötarkoitus on sekvenssien luokittelu johonkin tietokannan sisältämään ryhmään [HeH94], joten tässä kirjoituksessa keskitytään ainoastaan tällaisiin hakuihin.

Sekvenssin luokittelu johonkin tietokannan sisältämään ryhmään kuuluvaksi koostuu kolmesta työvaiheesta. Ensin kaikki tietokannan lohkot pisteytetään hakusekvenssiä vastaan, ja pistemäärät kalibroidaan yhteensopiviksi. Tämän jälkeen muodostetaan tulosjoukkoja hakutuloksista, joiden suhteelliset sijainnit hakusekvenssissä ja tietokannassa vastaavat suunnilleen toisiaan. Tulosjoukot eivät ole suoraan vertailukelpoisia keskenään, mutta viimeisessä vaiheessa lasketaan arvio sille, kuinka todennäköisesti tulosjoukko olisi voinut esiintyä satunnaisesti.

Jokaiselle lohkolle on etukäteen luotu sijaintiriippuvainen pisteytysmatriisi (*position specific scoring matrix, PSSM*), joka sisältää pistemäärän jokaiselle lohkon indeksille, jokaista mahdollista merkkiä kohti. Henikoff ja Henikoff [HeH96] selittävät pisteytysmatriisien luomisen tarkemmin.

```
    1  2  3  4        A  C  B  A  C  D  B
A  -1  6  3 -1       -1 -1  0 -1              = -3
B   2  1  0 -2          0  1  3  0           =  4
C   0 -1  2  0             2  6  2  3        = 13
D  -2 -5  1  3                -1 -1  1 -2    = -3
```

Kuva 5: Sijaintiriippuvainen pisteytysmatriisi (vasemmalla), ja sekvenssin ACBACDB alisekvenssien pisteytys sitä käyttäen (oikealla).

Sekvenssi pisteytetään jokaista tietokannan sisältämää lohkoa vastaan. Pisteytys suoritetaan hakemalla sekvenssistä kaikki kyseisen lohkon pituiset alisekvenssit (n merkin pituiselle sekvenssille, ja l pituiselle lohkolle löytyy $n - l + 1$ alisekvenssiä), ja laskemalla yhteen alisekvenssin jokaista merkkiä kohti sijaintiriippuvaisen pisteytysmatriisin arvo kyseiselle merkki/sijainti-parille (kuva 5) [HeH94].

Lohkojen raakapisteet eivät ole suoraan vertailukelpoisia keskenään. Esimerkiksi lohkon muodostamiseen käytettyjen sekvenssien määrä ja lohkon pituus voivat vaikuttaa merkittävästi niin pisteytysmatriisiin, kuin varsinaisten pisteiden jakaumaan. Tarkoitus on löytää lohkokohtainen raja, jota suuremmat pistemäärät ovat todennäköisesti kiinnostavia. Raakapisteet jaetaan ensin tällä kalibrointiarvolla, ja kerrotaan lopuksi tuhannella[3] [HeH94].

Kalibrointiarvo lasketaan pisteyttämällä jokainen tietokannan lohko kaikkia tietokantaa muodostettaessa käytettyjä sekvenssejä vastaan, paitsi niitä, jotka jo kuuluvat kyseiseen lohkoon[4]. Jos olisi varmaa, että aineisto on täydellisesti luokiteltua, olisi oikeutettua asettaa kalibrointiarvoksi korkein saatu pistemäärä. Koska näin ei kuitenkaan ole, huomioidaan nämä mahdolliset virheelliset negatiiviset luokittelut (*false negatives*) asettamalla kalibrointiarvoksi 99.5:nen prosenttiluvun sekvenssin saama pistemäärä [HeH96].

Seuraavaan vaiheeseen hyväksytään 400 parhaat pisteet saanutta osumaa (lohko/sijaintiparia). Sama lohko voidaan siis hyväksyä moneen kertaan eri osumissa. Pyrkimyksenä on yhteensopivien osumien niputtaminen ryhmiin. Osumat sopivat yhteen, jos niiden lohkot kuuluvat samaan lohkojoukkoon, ja osumien suhteellinen sijainti vastaa suunnilleen kyseisten lohkojen sijaintia lohkojoukossa. Suunnilleen samalla tarkoitetaan tässä tapauksessa, että osumat ovat oikeassa järjestyksessä, ja että osumien etäisyys on korkeintaan alkuperäisten lohkojen pienimmän ja suurimman etäisyyden summa.

Osumajoukot muodostetaan valitsemalla ankkuriosuma, jonka ympärille muita saman lohkojoukon huonompia osumia yritetään sijoittaa tukemaan ankkuriosumaa. Pelkästään heikkolaatuisista osumista koostuva hakutulos ei todennäköisesti ole kiinnostava, joten ankkurilohkon kalibroidun pistemäärän on oltava vähintään 1000. Tukiosumia käsitellään ahneesti, laskevassa pistejärjestyksessä; jos tietylle ankkuriosumalle on olemassa yhteensopivat tukiosumat $A, B,$ ja C pisteillä $900, 899, 898$ valitaan tukiosumaksi aina A, riippumatta A:n, B:n ja C:n keskinäisestä yhteensopivuudesta [HeH94].

[3] Kertolasku tuhannella tehdään oletettavasti, jotta voidaan käyttää kokonaislukuja laskutoimituksiin liukulukujen sijasta

[4] Lohko pisteytetään erikseen myös siihen kuuluvia sekvenssejä vastaan. Vertailemalla näiden sekvenssiryhmien pisteiden jakaumia voidaan arvioida, kuinka tehokas kyseinen lohko on aineiston luokittelussa. Tämän tehokkuusarvon laskeminen ei kuitenkaan ole tarpeen tässä kirjoituksessa kuvattua luokittelua varten. [HeH96]

Pelkästä ankkuriosumasta koostuvien osumajoukkojen hyvyyttä voidaan verrata yksinkertaisesti vertailemalla ankkuriosumien pisteitä. Monen osuman joukoissa suorat pistevertailut muuttuvat vaikeammiksi joukkojen erilaisten ominaisuuksien takia. Henikoffit esittävät artikkelissaan [HeH94] menetelmän merkittävyyden E arvioimiseksi. E ilmaisee, kuinka todennäköisesti osumajoukon tukiosumat löytyivät sattumalta (pienemmät E:n arvo ilmaisevat siis merkittävämpiä osumajoukkoja). Merkittävyyttä laskettaessa käytetään pisteiden sijasta osumien pistesijoituksia (*rank*), millä saavutetaan yksinkertaisempi matemaattinen malli. E:n laskukaavassa oletetaan, että sen arvoon vaikuttavat todennäköisyydet ovat toisistaan riippumattomia:

$$E = \prod_{i=1}^{s} P(\text{rank}_i) \times P(\text{min}_i \leq \text{dist}_i \leq \text{max}_i)$$

jossa s on tukiosumien määrä, $P(\text{rank}_i)$ ilmaisee todennäköisyyden sille, että joku käsiteltävän lohkoryhmän lohkoista osuisi sijoitukseltaan sopivaan haarukkaan, ja $P(\text{min}_i \leq \text{dist}_i \leq \text{max}_i)$ ilmaisee todennäköisyyden sille, että osuman sijainti sekvenssissä oli ankkuriosumaan ja muihin tukiosumiin verrattuna sopiva.

Pistesijoitusten osumisen todennäköisyys ($P(\text{rank})$) arvioidaan hypergeometrisellä jakaumalla. Hypergeometrista jakaumaa käytetään valittaessa äärellisestä joukosta, joka sisältää kahdenlaisia alkioita, useita alkioita ilman takaisinpanoa. Tässä tapauksessa edellä mainittu äärellinen joukko koostuu kaikista mahdollisista osumista (jos hakusekvenssin pituus on N merkkiä, ja lohkojen määrä on B, on osumia yhteensä $N \times B$). Osumien jaossa kahteen ryhmään kriteerinä on se, kuuluuko kyseisen osuman lohko oikeaan lohkoryhmään.

Merkitään D:llä haarukkaa, johon osuman on pitänyt pisteytyksessä sijoittua. Käytännössä tämä merkitsee erotusta kyseisen osuman sijoituksen, ja arvioitavassa osumajoukossa sitä välittömästi paremmin sijoittuneen osuman sijoituksen välillä:

$$D = \text{rank}_{i-1} - \text{rank}_i$$

jossa rank_x on käsiteltävän osumajoukon x:nneksi parhaan osuman sijoitus.

Merkitään T:llä kaikkia osumia, jotka sijoittuivat nyt käsiteltävää osumaa huonommin. R:llä puolestaan merkitään osumia, joiden lohko kuuluu oikeaan lohkoryhmään, ja joiden lohkoa ei vielä ole käsitelty:

$$T = N \times B - \text{rank}_i$$
$$R = N \times (G - i)$$

jossa N on sekvenssin pituus, B on kaikkien lohkojen määrä, G on käsiteltävän lohkoryhmän lohkojen määrä, ja i on käsiteltävän osuman sijoitus osumaryhmän sisällä (parhaiten sijoittuneelle tukijoukolle $i = 1$, toiseksi parhaalle $i = 2$, ja niin edelleen).

Syöttämällä yllä lasketut arvot hypergeometrisen jakauman kaavaan saamme selville merkittävyyden arvioimista varten tarvitun pistesijoitus-todennäköisyyden:

$$P(\text{rank}) = \frac{\binom{R}{1} \times \binom{T-R}{D-1}}{\binom{T}{D}}$$

Arviointia varten tarvitaan myös sijainnin osumistodennäköisyys, jonka arviointi on huomattavasti suoraviivaisempaa:

$$P(\text{min} \leq \text{dist} \leq \text{max}) = \frac{max - min}{N}$$

jossa N on hakusekvenssin pituus, *min* on alhaisin hakusekvenssin indeksi, johon kyseinen indeksi olisi ankkurilohkon ja muut tukilohkot huomioiden voitu sijoittaa, ja *max* on vastaavasti korkein indeksi. Indeksit *min* ja *max* voidaan laskea yksinkertaisella yhteenlaskulla lohkotietokannasta löytyvistä lohkojen pituuksista ja keskinäisistä etäisyyksistä.

Esimerkki. 100 merkin pituinen sekvenssi S pisteytetään 500 lohkoa sisältävää lohkotietokantaa vasten, jolloin löydetään yhteensä 50000 osumaa.

Eräs yhteensopiva osumaryhmä koostuu osumista A, B ja C. Kyseisten osumien lohkoryhmä sisältää lohkot a, b, c ja d, joista viimeksimainitulle ei löytynyt tähän osumaryhmään yhteensopivaa osumaa.

Osuma	Lohko	Indeksi	Pistemäärä	Sijoitus
A	a	45	1300	10
B	b	30	1000	100
C	c	80	950	250

Tietokantaan talletettujen lohkojen pituuksien ja välimatkojen pohjalta on määritelty osumien indeksien erotuksille seuraavat minimit ja maksimit:

Lohkopari	Minimietäisyys	Maksimietäisyys
$b - a$	10	20
$a - c$	10	50

Osumajoukolle voidaan nyt arvioida merkittäävyys E yllä esitettyjen tietojen perusteella:

$$
\begin{aligned}
E &= P(\text{rank}_B) \times P(\text{min}_B \leq \text{dist}_B \leq \text{max}_B) \times P(\text{rank}_C) \times P(\text{min}_C \leq \text{dist}_C \leq \text{max}_C) \\
&= \frac{\binom{300}{1} \times \binom{(50000-100)-300}{100-10-1}}{\binom{50000-100}{100-10}} \times \frac{(45-10)-(45-20)}{100} \times \\
&\quad \frac{\binom{200}{1} \times \binom{(50000-250)-200}{250-100-1}}{\binom{50000-250}{250-100}} \times \frac{(45+50)-(45+10)}{100} \\
&\approx 0.3167764 \times 0.1 \times 0.3315714 \times 0.4 \\
&\approx 0.004
\end{aligned}
$$

E:n suhteellisen pieni arvo ja ankkuriosumana käytetyn A:n korkea kalibroitu pistemäärä viittaisivat siihen, että testattu sekvenssi voisi olla sukua kyseisen osumaryhmän lohkot muodostaneille sekvensseille.

Käytetty arviointimenetelmä tekee useita oletuksia ja yksinkertaistuksia. Henikoff ja Henikoff [HeH94] suorittivat kokeellisia hakuja testatakseen arvioiden laatua. Tällä menetelmällä lasketun merkittävyyden todettiin vastaavan riittävän tarkasti kokeissa havaittuja todennäköisyyksiä.

4 Yhteenveto

Kirjoituksessa kuvattiin kolmivaiheinen algoritmi lohkotietokannan rakentamiseen. Ensimmäisessä vaiheessa etsitään luokitelluista syötesekvensseistä motiiveja, joista luodaan

lohkoja. Toisessa vaiheessa yhdistetään ja laajennetaan lohkoja niiden pistemäärän optimoimiseksi. Oleellisin osa toista vaihetta on siinä käytetty pisteytysmenetelmä; sekvenssipainotettu ja pituusnormalisoitu parien summa-algoritmi, jossa käytetään syöteaineistoon sopivaa pistematriisia. Viimeisessä vaiheessa lohkoista luodaan suunnattu kehätön verkko, jonka avulla etsitään paras yhteensopivien lohkojen ja sekvenssien joukko.

Lisäksi esiteltiin lohkotietokannan pääasiallinen sovellus, sekvenssien luokittelu tunnettuihin ryhmiin. Luokitteluun käytetyssä hakualgoritmissa pisteytetään luokiteltava sekvenssi tietokannan lohkoista luotuja sijaintiriippuvaisia pisteytysmatriiseja vastaan, ja kalibroidaan saadut pistemäärät yhteensopiviksi. Keskenään yhteensopivista tuloksista muodostetaan tulosjoukkoja. Yksittäisiä tuloksia voidaan vertailla niiden kalibroitujen pistemäärien mukaan. Useammista osumista koostuvia tulosjoukkoja puolestaan voidaan vertailla arvioimalla todennäköisyys sille, että havaittu samankaltaisuus syntyi satunnaisesti.

Viitteet

[DEK98] R. Durbin, S. Eddy, A. Krogh & G. Mitchison, *Biological sequence analysis*, Cambridge University Press, Cambridge, UK (1998).

[Hen99] J. Henikoff, "Assembling Blocks", *Pattern Discovery in Biomolecular Data: Tools, Techniques, and Applications*, pp 24-29. Oxford University Press, November 1999.

[HeH91] S. Henikoff & J.G. Henikoff, "Automated assembly of protein blocks for database searching", *Nucl. Acids Res.* 19:6565-6572 (1991).
`http://blocks.fhcrc.org/papers/BLOCKS.pdf`

[HeH92] S. Henikoff & J.G. Henikoff, "Amino acid substitution matrices from protein blocks", *Proc Natl. Acad. Sci. USA*, 89:10915-10919 (1992).
`http://blocks.fhcrc.org/papers/BLOSUM.pdf`

[HeH94] S. Henikoff & J.G. Henikoff, "Protein family classification based on searching a database of blocks.", *Genomics*, 19:97-107 (1994).
`http://blocks.fhcrc.org/papers/BLOCKSEARCH.pdf`

[HHA95] S. Henikoff, J.G. Henikoff, W.J. Alford & S. Pietrokovski, "Automated construction and graphical presentation of protein blocks from unaligned sequences", *Gene-COMBIS*, Gene 163 (1995) GC17-26.
`http://blocks.fhcrc.org/papers/BLOCKMAKER.pdf`

[HeH96] J.G. Henikoff & S. Henikoff, "Blocks database and its applications", *Meth. Enzymol.*, 266, 88-105 (1996).
`http://blocks.fhcrc.org/papers/APPLICATIONS.pdf`

[SAC90] H. Smith, T. Annau, S. Chandrasegaran, "Finding sequence motifs in groups of functionally related proteins", *Proc. Natl. Acad. Sci*, Vol 87, pp. 826-830, January 1990.
`http://www.pnas.org/cgi/reprint/87/2/826.pdf`

[SP03] Silmämääräinen arvio www-sivulla
`http://us.expasy.org/sprot/relnotes/relstat.html`, *SWISS-PROT PROTEIN KNOWLEDGEBASE RELEASE 42.0 STATISTICS* esitetystä tietosisällön määrästä ajan funktiona, tarkistettu 17 lokakuuta, 2003.