RNA-sekundaarirakenteiden tietokoneanalyysi

Tommi Lehtinen

Tampereen Yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro Gradu -tutkielma
Joulukuu 2006

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tommi Lehtinen: RNA-sekundaarirakenteiden tietokoneanalyysi
Pro gradu –tutkielma, 51 sivua
Joulukuu 2006

Ihmisen genomin sekvensointi yhdessä uusien molekyylibiologian tutkimusmenetelmien kanssa on tuottanut tarpeen myös uusille tilastollisille sekä algoritmisille menetelmille aineistojen analysoimiseksi. Ribonukleiinihappo (RNA) toimii solun sisällä geneettisen tiedon välittäjänä sekä joidenkin biologisten prosessien säätelijänä. RNA-molekyylin rakenteelliset ominaisuudet määräävät pitkälti sen toiminnallisuuden. RNA-sekundaarirakenteita voidaan ennustaa ja analysoida algoritmisilla menetelmillä sekä niihin perustuvilla ohjelmistoilla.

Tässä tutkielmassa tarkastellaan RNA-molekyylin sekundaarirakenteen ennustamisessa ja analysoinnissa käytettyjä keskeisiä algoritmisia menetelmiä. Lisäksi käsitellään jonkin verran keskeisiä DNA-mikrosirujen klusterointimenetelmiä, sekä niiden soveltamista RNA-sekundaarirakenteiden analysoinnissa. Kokeellisessa osuudessa testataan erilaisia menetelmiä, joilla voi luokitella samankaltaisia RNA-sekundaarirakenteita omiin klustereihinsa, sekä tutkitaan löytyykö analysoitavasta DNA-mikrosiruaineistosta samankaltaisten RNA-sekundaarirakenteiden rikastamia klustereita.

Avainsanat ja –sanonnat: RNA-sekundaarirakenne, editointietäisyys, klusterointi.
CR-luokat: G.3, I.5.3

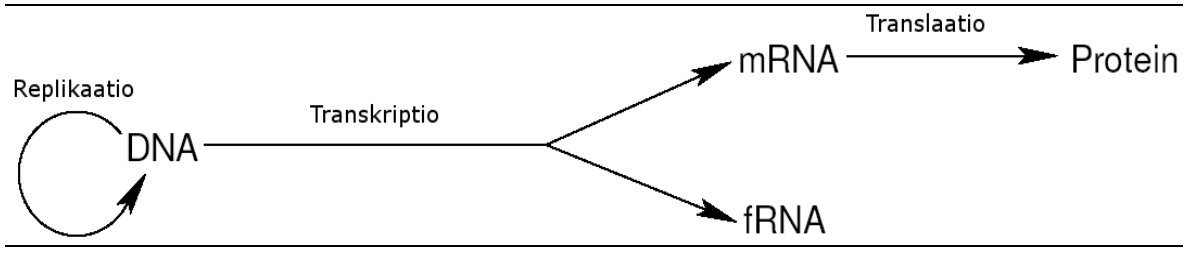
Sisällysluettelo

1. Johdanto

Vuonna 1953 Watson ja Crick julkaisivat artikkelin [Watson and Crick, 1953] deoksiribonukleiinihapon eli DNAn rakenteesta. Molekyylibiologian tieto on siitä lähtien kasvanut yhä kiihtyvällä vauhdilla. Uudet kehittyneet mittaustekniikat, esimerkiksi DNA-sekvensointi, ovat asettaneet uusia haasteita myös biologisen tiedon analysoinnille.

Elävät organismit rakentuvat soluista. Solut jaetaan yleisesti prokaryootteihin, joita ovat lähinnä bakteerit, sekä eukaryootteihin, joita ovat kaikki ihmis-, eläin- ja kasvisolut poislukien punasolut. Soluilla on useita erilaisia muotoja ja funktioita, niiden perusrakenteet ovat kuitenkin samankaltaisia. Yleisimpiä solujen orgaanisia yhdisteitä ovat hiilihydraatit, lipidit ja proteiinit. 60-90 prosenttia solusta on vettä. Kaikista aitotumallisista soluista löytyy tuma, jonka sisällä sijaitsevat kromosomiparit, jotka sisältävät organismin perimän. Kromosomit rakentuvat DNA-kaksoiskierteestä. Tieto DNA-sekvenssissä perustuu neljän nukleotidin: guaniinin, adeniinin, tymiinin ja sytosiinin (G,A,T,C) järjestyksen vaihteluihin. Esimerkiksi ihmisen 23 kromosomiparissa on yhteensä n. 3 miljardia emäsparia, jotka sisältävät tiedon n. 22000 geenistä. Geenien tiedon perusteella muodostuvat proteiinit, jotka määräävät yksilön perityt ominaisuudet.

Yksittäinen geeni on jokin tietty alue DNA-ketjussa, joka jakaantuu vielä pienempiin sekvensseihin. Transkriptiossa tietyt osa-alueet geenin DNA-sekvenssistä kopioituvat RNA-sekvenssiksi, josta poistuvat vielä introni-alueet. Tämän jälkeen pelkistä eksoni-alueista koostuva lähetti-RNA (mRNA, messenger RNA) siirtyy tumasta sytoplasmaan, jossa ribosomi suorittaa translaation muodostaen aminohapoista koostuvan proteiinisekvenssin lähetti-RNA:n välittämän informaation perusteella. Translaation jälkeen proteiinisekvenssi poimuttuu kolmiulotteiseen muotoonsa. Proteiinin tai RNA:n laskostuneen muodon tunteminen on tärkeää, koska muoto määrää näiden molekyylien funktion.



Kuva 1*:* geenistä proteiiniksi

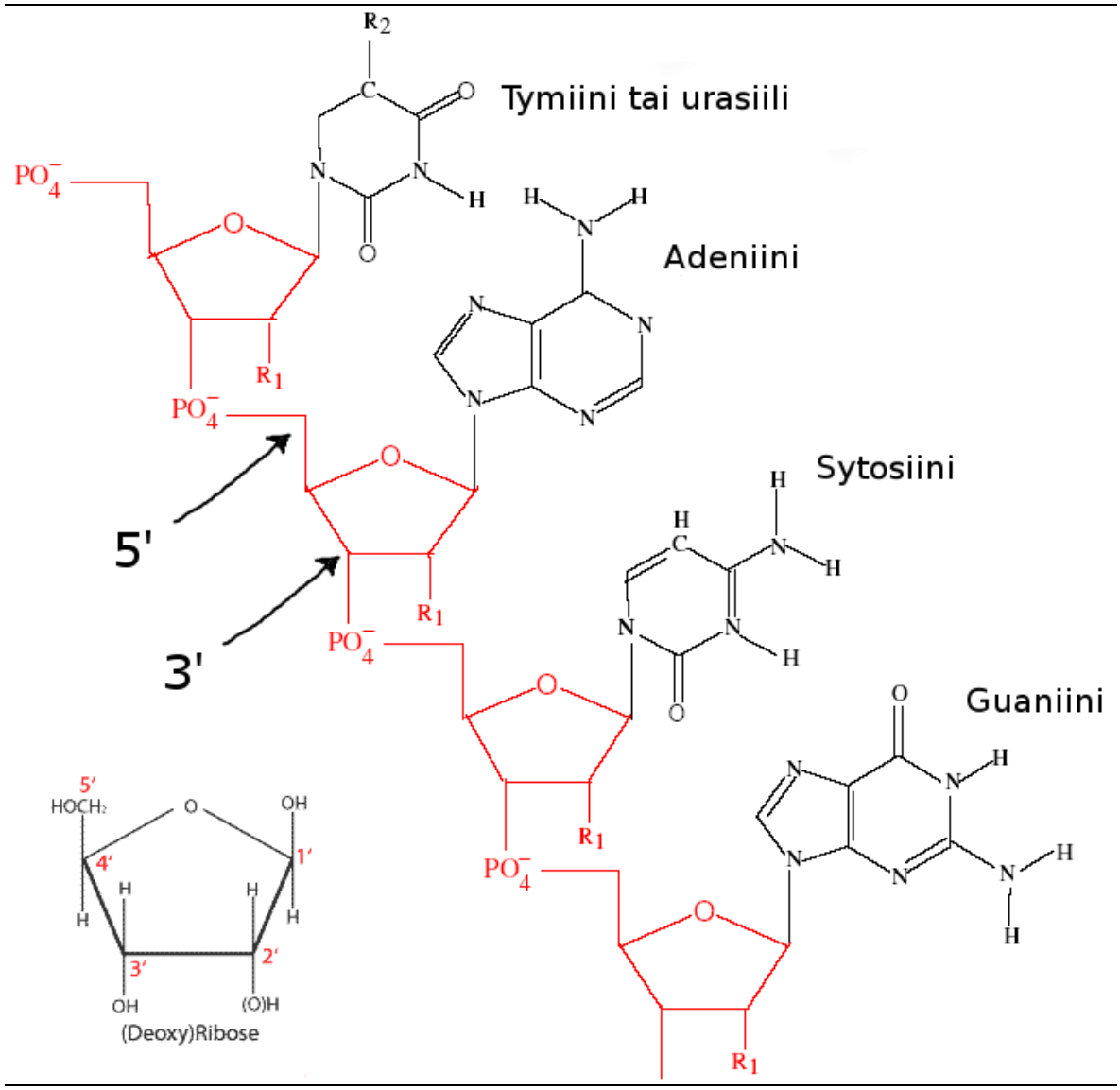
Molekyylibiologian keskeinen opinkappale (Central Dogma) on kuvassa 1 näkyvä prosessi DNA-sekvenssistä toiminnalliseksi proteiiniksi. Tutkielman aiheeseen liittyen kuvassa näkyy myös ehdotettu laajennus [Dennis, 2002], jossa transkriptiossa erotellaan mRNA ja toiminnallinen RNA (fRNA). Aivan viime vuosina on selvinnyt, että RNA:lla on paljon muitakin funktioita elimistössä kuin geenien tiedon välittäminen. Toiminnalliset RNA-molekyylit vaikuttavat esimerkiksi proteiinien laskostumisen säätelyyn tai voivat estää joidenkin geenien toiminnan RNA-häirinnällä (RNA interference). RNA-häirinnästä on tullut nopeasti keskeinen menetelmä uusien hoitomenetelmien etsimiseen eri sairauksille. Vuoden 2006 lääketieteen nobelin saivat RNA-häirinnän löytäjät, Andrew Fire ja Craig Mello.

Proteiinit ovat monimutkaisia ja suuria molekyylejä, jotka toimivat esimerkiksi solujen ja kudosten rakennusaineina, entsyymeinä, vasta-aineina, hormoneina ja kuljettajamolekyyleinä. Proteiinien toiminnan määrää pitkälti niiden kolmiulotteinen rakenne, joka muodostuu, kun aminohapoista koostuva polypeptidiketju poimuttuu esimerkiksi aminohappojen järjestyksen, ympäristön pH-arvon ja lämpötilan perusteella. Kaikkia laskostumiseen vaikuttavia tekijöitä ei vielä tunneta. Proteiinien laskostumisen ja siihen liittyvien tekijöiden selvittäminen on molekyylibiologian ja lääketieteen suurimpia haasteita. Luotettavat ennustusmenetelmät mullistaisivat lääketeollisuuden ja auttaisivat ymmärtämään ihmiskehon toimintaa entistä paremmin.

Tässä tutkielmassa käsitellään menetelmiä RNA-sekundaarirakenteiden ennustamiseen ja analysointiin. Sekä RNA:n että proteiinien sekundaarirakenteiden ennustaminen on mahdollista algoritmisilla menetelmillä. Proteiinien sekundaarirakenne muodostuu karkeasti yleistettynä pääketjun (backbone) atomien välisten vetysidosten perusteella. RNA-sekundaarirakenne sensijaan muodostuu pääasiassa molekyyliketjun muodostavien neljän emäksen välisten vetysidosten perusteella. Ennustusmenetelmät proteiinien ja RNA:n sekundaarirakenteille ovat selkeästi erilaisia. Useimmiten sekundaarirakenteet vaikuttavat ympäristöönsä silmukoiden vapailla nukleotideillä. Suhteellisen yksinkertaisesta perusperiaatteesta johtuen RNA-sekundaarirakenteiden ennustaminen on huomattavasti helpompaa kuin proteiinien sekundaarirakenteiden ennustaminen.

2. RNA

Ribonukleiinihappo (RNA, ribonucleic acid) on biologinen polymeeri joka on neljän nukleotidin: adeniiniin (A), sytosiinin (C), guaniinin (G) ja urasiilin (U) muodostama ketjumainen molekyyli. Nukleotidi on kemiallinen yhdiste, joka koostuu fosfaatti-, sokeri- ja emäsosasta. Nukleotidit pariutuvat kemiallisin sidoksin sokeriosan 3. tai 5. hiiliatomin kautta, jonka perusteella määritellään nukleotidisekvenssin päät 5’ ja 3’ päiksi. RNA-sekvenssi on siis nukleotideista koostuva merkkijonoketju, jolla on suunta alkaen 5’ päästä loppuen 3’ päähän. Kuva 2 esittää nukleotidiketjun molekyylirakennetta.



Kuva 2: sokeri-fosfaatti ketju

Nukleotidien järjestys sekvenssissä yhdessä ympäristön kemiallisten ja fysikaalisten ominaisuuksien kanssa määrää lopulta RNA-molekyylin 2- ja 3-ulotteisen rakenteen.

RNA-molekyyleillä on useita funktioita. Transkriptiossa syntyvä lähetti-RNA (messenger RNA, mRNA) on kopio geenistä, joka kuljettaa geneettisen informaation tumasta sytoplasmaan, jossa ribosomi muodostaa nukleotidien järjestyksen perusteella aminohapot proteiiniketjuksi. Kuvassa 1 näkyvä toiminnallinen RNA (functional RNA, fRNA), josta käytetään myös termiä ncRNA (non coding RNA), on RNA-ketju, joka ei koodaa proteiinia, mutta toimii kuitenkin jollakin tavalla solun sisällä. Esimerkiksi siirtäjä-RNA (tRNA,transfer RNA) on pieni solun sisällä toimiva RNA, jolla on erityinen nelihaarainen sekundaarirakenne ja L-muotoinen tertiäärirakenne. tRNA siirtää aminohapot oikealle paikalleen proteiinisekvenssissä. Yhdessä sekundaarirakenteen silmukassa sijaitseva kolmen nukleotidin pätkä, kodoni, vastaa tiettyä aminohappoa. snRNA:t (small nuclear RNA) toimivat avustajina prosessissa, jossa mRNA siirtyy tumasta sytoplasmaan. siRNA:t (small interfering RNA) taas ovat pieniä, n. 20-25 nukleotidista koostuvia molekyylejä, jotka sitoutuvat mRNA molekyyliin vaikuttaen proteiinien koodaukseen. [Nelson and Cox, 2000]

2.1. RNA-sekundaarirakenne

RNA on tärkeä tekijä monissa biologisissa prosesseissa, joiden ymmärtäminen edellyttää usein myös RNAn eri rakennetasojen tuntemista. 1-ulotteista RNA-sekvenssiä kutsutaan primäärirakenteeksi. Nukleotidien järjestyksen perusteella RNA poimuttuu 2-ulotteiseksi sekundaarirakenteeksi ja edelleen 3-ulotteiseksi tertiäärirakenteeksi, joka lopulta määrää RNAn funktion. RNAn tertiääri- ja sekundaarirakenne voidaan lukea suoraan röntgensäde-kristallografia-analyysillä, mikä on hyvin hidasta ja kallista. Sekundaarirakenteen ennustamiseen onkin kehitetty useita matemaattisia malleja ja tietokoneohjelmia. Sekundaarirakenteen sidokset ovat yleisesti vahvempia kuin tertiäärirakenteen sidokset, ja sekundaarirakenteen ominaisuuksista voidaan päätellä paljon myös koko RNAn toiminnasta. Toisin kuin proteiinit RNA muodostaa sekundaarirakenteen nukleotidien välisten vetysidosten perusteella. Normaaleissa fysiologisissa olosuhteissa, esimerkiksi ihmiskehossa 37 asteen lämpötilassa, RNA-molekyyli poimuttuu sekundaarirakenteeksi tyypillisesti Watson-Crick –emäsparien (A-U,C-G) ja vähän heikompien G-U parien mukaan.

2.1.1. Sekundaarirakenteen määritelmä

RNA:n sekundaarirakenne S määritellään muodollisesti joukkona emäspareja (i,j), i < j, siten, että kullakin kahdella emäspareilla (i,j) ja (k,l), joilla i ≤ k, on voimassa ehto:

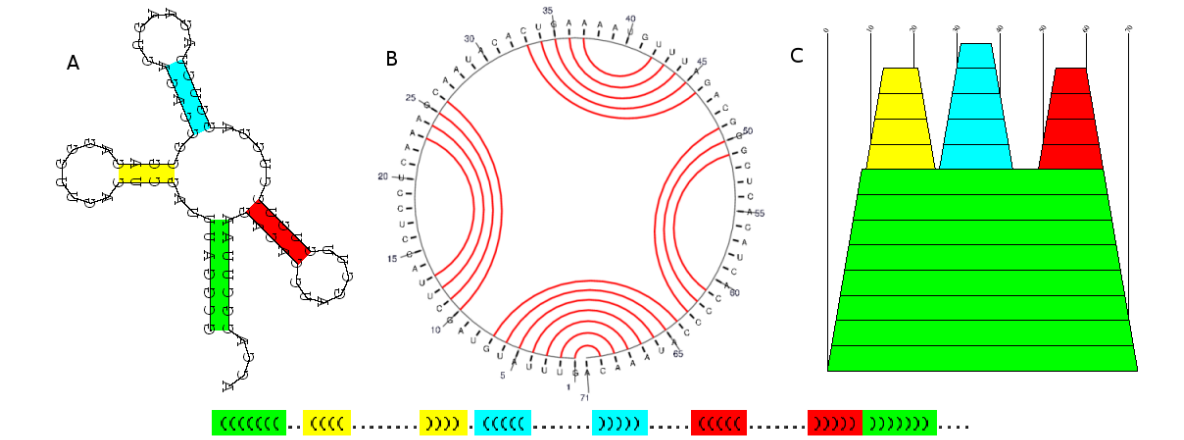
1. i = k jos ja vain jos j = l
2. Jokaiselle emäsparille (i,j) ja (k,l)
   täytyy päteä ehto i < k < l < j tai k < i < j < l

Ensimmäinen ehto määrittelee yksinkertaisesti sen, että jokaisella nukleotidilla voi olla korkeintaan yksi pari. Tertiäärirakenteessa voi kuitenkin ilmetä poikkeuksia tähän sääntöön. Toinen ehto takaa, että sekundaarirakenne voidaan esittää tasograafina, jolloin niin sanottuja pseudosolmuja ei voi esiintyä. Pseudosolmut ovat tärkeä osa joissakin RNA-molekyyleissä, mutta ne voidaan laskea tertiäärirakenteen osaksi [Waterman and Smith, 1972]. Solmujen ja pseudosolmujen kieltäminen on edellytys useimmissa ennustusmenetelmissä käytetyille dynaamisille algoritmeille [Hofacker et al., 1994]. Myös pseudosolmut hyväksyviä dynaamisia algoritmeja on kehitetty [Akutsu, 2000].

2-ulotteinen sekundaarirakenne voidaan esittää merkkijonona, joka koostuu (,), ja . merkeistä esittäen pariutuneita nukleotideja kaarilla ja parittomia nukleotideja pisteellä. Merkintätapaa kutsutaan sulkunotaatioksi (bracket notation). Rakenteen merkkijonoesitys mahdollistaa rakenteiden vertailun merkkijonotäsmäyksen menetelmillä. Seuraava sekvenssi on internetistä löytynyt satunnainen ihmisen tRNA – sekvenssi, sulkunotaatio sen alla on ennustettu RNAfold –ohjelmalla. Sekvenssi on sama kuin kuvan 5 sekundaarirakenteissa.

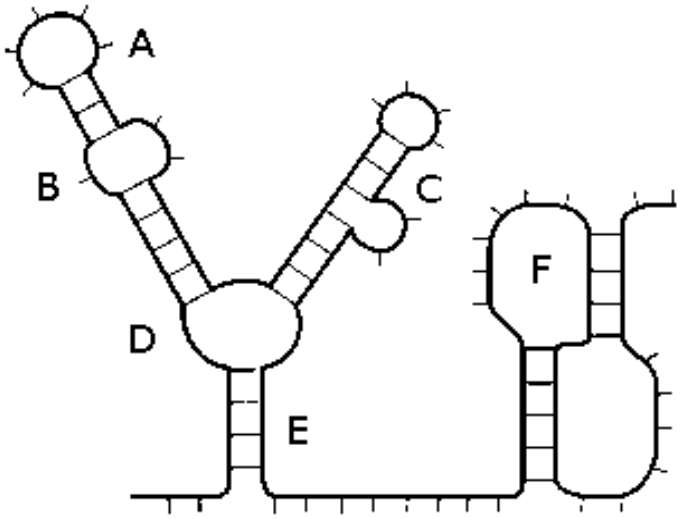
```
GUUUAUGUAGCUUACCUCCUCAAAGCAAUACACUGAAAAUGUUUAGACGGGCUCACAUCACCCCAUAAAC
(((((((..((((.........)))).....(((((....)))))..(((.........)))))))))
1111111  2222         2222      33333    33333  444         4441111111
```

Numerot sulkunotaation alla kertovat keskenään yhdistyvät alueet sekvenssistä. Muita keskeisiä sekundaarirakenteen esitystapoja ovat kuvassa kolme A: perinteinen, B: ympyrä (circle), ja C: vuoristo- (mountain) esitystavat. Värjätty sulkunotaatio kuvan alareunassa vastaa kuvien A ja C värejä.



Kuva 3: sekundaarirakenteen osat

Sekundaarirakenteen osia ovat kuvassa 4: A silmukka (hairpin loop), B sisäinen silmukka (internal loop), C pullistuma (bulge loop), D keskussilmukka (multibranched loop), E varsi (stem) ja F pseudosolmu (pseudo knot).



Kuva 4: sekundaarirakenteen esitystavat

3. Sekundaarirakenteen ennustaminen

Ensimmäinen ja yksinkertaisin menetelmä sekundaarirakenteen ennustamiseksi oli laskea sekvenssille sekundaarirakenne, jossa on mahdollisimman paljon emäspareja. Menetelmä on kuitenkin monissa tapauksissa käyttökelvoton, koska se voi antaa liian paljon tuloksia eikä ole välttämättä luotettava. [Waterman, 1995] Nykyään RNA sekundaarirakenteiden ennustamiseen käytetään pääasiallisesti kahta erilaista metodia. Suosituimmat menetelmät perustuvat rakenteen vapaan energian minimointiin dynaamisen ohjelmoinnin algoritmeilla. Toinen keskeinen menetelmä on hyödyntää fylogenetiikan ja sekvenssirinnastuksen tuottamaa tietoa stokastisilla kontekstittomilla kieliopeilla (Stochastic Context Free Grammar, SCFG).

Fylogenetiikka pyrkii sekvenssejä vertailemalla ennustamaan eliöiden evolutiivisia suhteita. Kahta DNA-sekvenssiä vertailtaessa täytyy ottaa huomioon evoluution ja mutaatioiden tuotokset, sama pätee myös biologisten rakenteiden vertailuun. Kaksi proteiinia tai RNA-rakennetta voivat olla toiminnaltaan samanlaisia olematta identtisiä, primäärirakennetasolla ne voivat olla hyvinkin erilaisia. Koska tietyt sekvenssit hyvin usein poimuttuvat tietyllä tavalla, käytetään sekvenssirinnastuksella saatua tietoa lisäämään ennustuksen tarkkuutta. Tilastollista ja fylogeneettistä tietoa hyödynnetään erityisesti SCFG menetelmillä, esimerkiksi Pfold –algoritmissa [Dowell and Eddy, 2004], sekä RNAalifold –ohjelmassa, joka ennustaa konsensus-sekundaarirakenteen ClustalW –ohjelmalla rinnastetuille sekvensseille.

3.1 Dynaaminen ohjelmointi ja termodynaamiset pisteytysfunktiot

Todennäköisimmät keskenään pariutuvat alueet RNA-sekvenssistä voidaan määrittää laskennallisesti suoraan Watson-Crick parien (C-G, A-U) perusteella. Tällä tavalla muodostuvista useista mahdollisista rakenteista voidaan valita se tai ne, joilla on mahdollisimman vakaa minimienergiatila (MFE, Minimum Free Energy). Tämä mahdollistaa rekursiivisten dynaamisten algoritmien käytön minimienergiatilan laskemiseksi. RNA-sekundaarirakenteen termodynaaminen energiatila voidaan laskea sekvenssin molekyylitiedon, ympäristön lämpötilan ja Boltzmanin vakion avulla [Hofacker et al., 1994]. RNA sekundaarirakenteen vapaan energian katsotaan olevan rakenteen silmukoiden energiatilojen summa, ja todennäköisimmin luonnossa muodostuvan sekundaarirakenteen oletetaan olevan minimienergiatilassa [Zuker et al, 1999].

Tällä hetkellä suosituin menetelmä, jota käyttävät esimerkiksi MFold ja RNAFold, perustuu dynaamiseen ohjelmointiin ja termodynaamisen energiamallin pisteytysfunktioihin. Dynaamisen ohjelmoinnin perusidea on se, että pienemmät osaongelmat ratkaistaan ensin, ne taulukoidaan ja osista kootaan suurempien osaongelmien ratkaisuja.

Kun sekundaarirakenne muodostuu, nukleotidit pariutuvat vetysidoksin. Kahden pariutuneen nukleotidin vapaa energia on nolla. Mitä enemmän nukleotideja pariutuu, sitä vähemmän rakenteessa on vapaata energiaa ja sitä vakaampi on myös RNAn rakenne, ainakin teoriassa. Energiaparametrit yksittäisille loopeille ja tietyille pariutumisille on määritetty kokeellisesti. Ne riippuvat luupin muodosta, koosta ja nukleotidien sijainnista silmukassa [Waterman and Smith, 1972]. Kokeellisia parametreja tutkitaan ja tarkennetaan edelleen [Mathews et al, 1999].

3.1.1. Perusmenetelmä ja dynaaminen ohjelmointi

Yksinkertaisin menetelmä sekundaarirakenteen ennustamiseksi olisi generoida sekvenssistä kaikki mahdolliset rakenteet, laskea niiden energiatilat, ja valita optimaalisin ratkaisu. Yksinkertaisimmillaan siis hakea rakenne, jossa on mahdollisimman paljon emäspareja. Mahdollisten rakenteiden määrä kasvaa kuitenkin eksponentiaalisesti sekvenssin pituuden suhteen. Teoreettinen sekundaarirakenteiden määrä N pituiselle sekvenssille on keskimäärin $(1.8)^N$. Tällä menetelmällä esimerkiksi 100 nukleotidin mittaisella sekvenssillä olisi n. 10^{25} mahdollista rakennetta. Eli modernilta tietokoneelta, joka laskee vapaan energian 10000 rakenteelle sekunnissa, kestäisi 1013 vuotta optimaalisen sekundaarirakenteen ennustamiseen. Ensimmäinen ja edelleen suosituin menetelmä laskennan nopeuttamiseksi, on käyttää dynaamista ohjelmointia [Nussinov and Jacobson, 1980] sekä termodynaamisia parametrejä. [Zucker and Stiegler, 1981]

Dynaamiseen ohjelmointiin perustuvat algoritmit käyvät epäsuorasti läpi kaikki sekundaarirakenteet, ilman että niitä täytyy oikeasti tuottaa. Tämä tapahtuu kahdessa vaiheessa. Ensimmäiseksi käydään läpi kaikki mahdolliset sekvenssin palaset, alkaen pienimmistä palasista, laskien näille pienin vapaa energia. Kun siirrytään aina suurempiin sekvenssin osiin, näiden energiatilojen laskemisessa hyödynnetään aiemmin laskettujen pienempien osien parhaita tuloksia. Tämä toistuu rekursiivisesti, kunnes koko sekvenssin pienin vapaa energia tiedetään. Tämän jälkeen toisessa vaiheessa, määritetään laskettujen vapaiden energiatilojen perusteella se rakenne, jolla on matalin kokonaisenergiatila. Nopeimpien algoritmien kompleksisuus on $O(N^3)$ eli sekvenssin pituuden

kaksinkertaistaminen kahdeksankertaistaa laskemiseen tarvittavan ajan. Nämä algoritmit eivät kuitenkaan voi ennustaa pseudosolmuja. Myös pseudosolmut ennustavia dynaamisen ohjelmoinnin algoritmeja on kehitetty. Rivas ja Eddyn algoritmi [28] kykenee ennustamaan lähes kaikki tunnetut pseudosolmut mutta sen kompleksisuus on $O(N^6)$. Akutsun [Akutsu, 2000] algoritmin kompleksisuus on $O(N^4)$ mutta se ei kykene ennustamaan aivan kaikkia teoreettisia pseudo-rakenteita.

3.1.2. Mfold

Michael Zuckerin kehittämä, vuonna 2003 julkaistu, Mfold on ensimmäinen ja suosittu MFE-menetelmään perustuva vapaasti käytettävä ohjelmisto. [Zucker, 2003] Zukerin ja Stieglerin aiemmin kehittämä algoritmi [Zucker and Stiegler, 1981] olettaa, että vahvimmin energiatilaltaan toisiinsa sitoutuneet, yleensä myös pisimmät alueet sekvenssissä, esiintyvät myös luonnollisissa sekundaarirakenteissa. Emäkset pariutuvat vetysidoksin, jotkut emäsparit (G-C) pariutuvat kolmella vetysidoksella, jotkut vain kahdella (A-T). Tästä syystä Zucker ja Stiegler käyttivät eri emäspareille laskettuja energiatiloja parametreina osarakenteiden energiatilojen laskemiseen. Näitä parametrejä on tarkennettu vuosien mittaan. [Mathews et al, 1999]

Zuckerin ennen Mfoldia julkaisema FOLD-ohjelma ennusti sekvenssille vain yhden rakenteen, jolla on pienin vapaa energia. Tämä ei kuitenkaan aina riitä, koska yhdenkin nukleotidin muutos sekvenssissä voi antaa täysin erilaisen ennusteen sekundaarirakenteelle. Ja toisaalta luonnossa sekundaarirakenteet eivät aina muodostu niin kuin algoritmit ennustavat. Mfold toi ennustamiseen mukaan alioptimaaliset (suboptimal) rakenteet, joilla on erilainen rakenne, mutta lähes yhtä pieni minimienergiatila kuin energiatilan suhteen optimaalisella rakenteella. Mfold on vapaasti käytettävissä internet-palvelimella tai ladattavissa omalle koneelle.

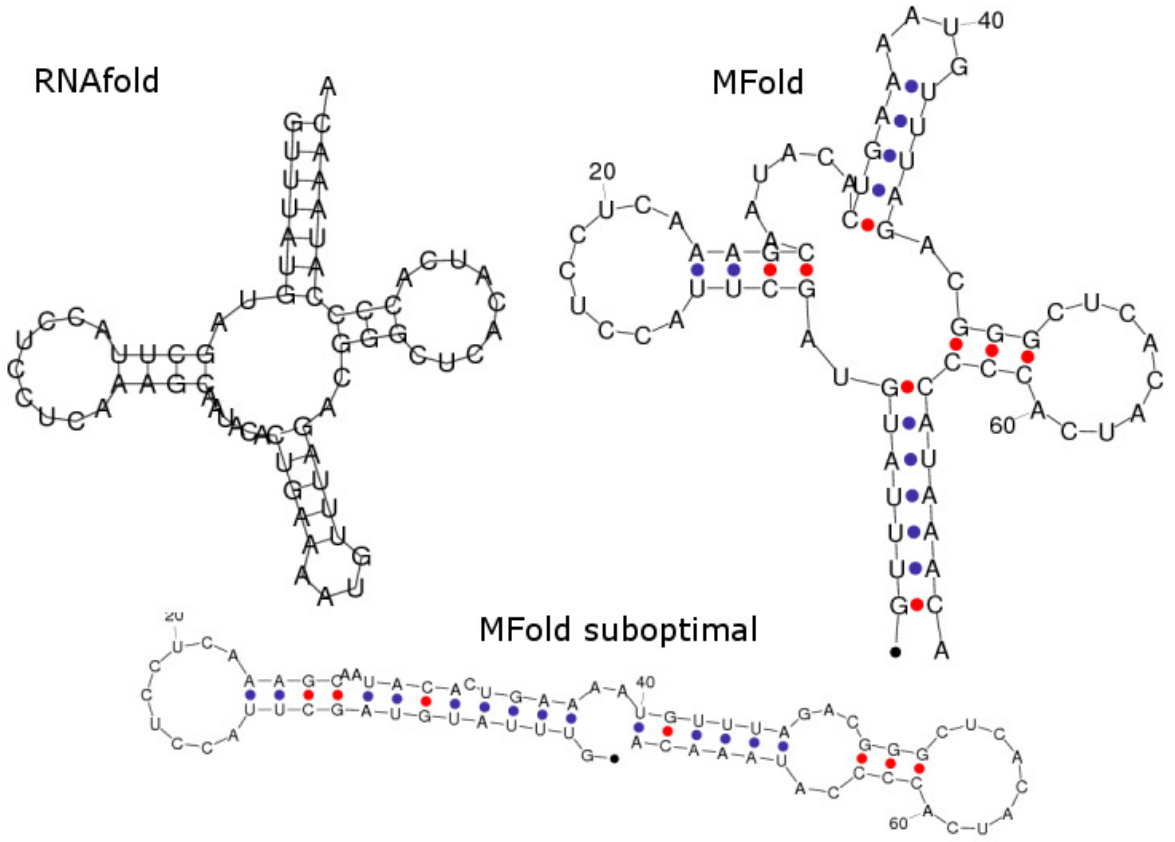
MFoldin algoritmi perustuu nk. lähimmän naapurin energiasääntöihin. Sensijaan että kokonaisenergiatilan pienimmät osaenergiat laskettaisiin yksittäisille emäspareille, ne lasketaan vierekkäisten nukleotidien muodostamille emäspariyhdistelmille. Niistä lasketaan edelleen dynaamisesti silmukoiden energiatilat. Lopullinen MFE-rakenne koostetaan silmukoiden energioiden pohjalta. Energiasäännöt eri nukleotidipareille ovat tallennettuina matriiseihin.

3.1.3. ViennaRNA

Vienna RNA -paketin kehittivät tutkijat Wienin Yliopiston Teoreettisen Biokemian Instituutista. [Hofacker et al., 1994] Vienna RNA sisältää kirjastoja ja joitakin erillisiä ohjelmia RNA-sekundaarirakenteiden ennustamiseen ja vertailuun. Paketti on vapaasti käytettävissä ja ladattavissa internetistä. Seuraavat paketin keskeisimmät ohjelmat lukevat syötteitä standardisyötevirrasta.

RNAfold – ennustaa MFE sekundaarirakenteen ja emäspariutumisen todennäköisyydet.
RNAeval – arvioi sekundaarirakenteen energian.
RNAinverse – määrittää sekvenssit, jotka poimuttuvat syötettyyn muotoon
RNAdistance – laskee sekundaarirakenteiden välisiä etäisyyksiä
RNApdist – vertailee emäsparien todennäköisyyksiä
RNAsubopt – alioptimaalinen ennustus
RNAplot – piirtää sekundaarirakenteen PostScriot, SVG tai GML formaateissa
RNAalifold – ennustaa yhteisen sekundaarirakenteen useille rinnastetuille sekvensseille
RNALfold – ennustaa pitkästä sekvenssistä paikallisia optimaalisia rakenteita.

Kuvan 5 sekundaarirakenteet on ennustettu samasta tRNA-sekvenssistä RNAfoldilla ja Mfoldilla. Sekä Mfold että RNAfold perustuvat samaan Zuckerin ja Stieglerin algoritmiin. [Gardner and Giegerich, 2004] Valitsin tRNA-sekvenssin ohjelmien testaamiseen, koska tRNA:n kolmen silmukan apilalehtimuoto on valmiiksi tunnettu ja odotettavissa oleva ennustustulos. Molemmat ohjelmat ennustivat perusasetuksilla täsmälleen saman ja silmämääräisesti oikean rakenteen, jonka vapaa energia on -11.40 kcal/mol. Mfold antoi lisäksi alioptimaalisen rakenteen jonka vapaa energia on pienempi, -10.40 kcal/mol, mutta ainakin tässä tapauksessa ennuste olisi väärä vaikka sillä onkin pienin vapaa energia.



Kuva 5: eri ohjelmilla laskostettu tRNA

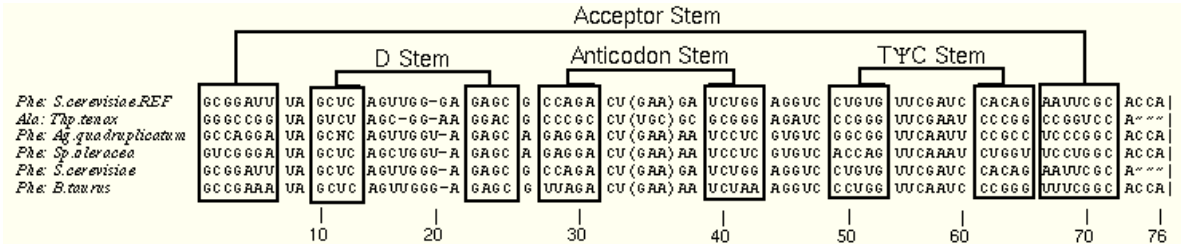
Gardner ja Giegerich vertailivat artikkelissaan [Gardner and Giegerich, 2004] eri ennustusmenetelmien tarkkuutta ja saivat RNAfoldin ja Mfoldin keskimääräiseksi sensitiivisyydeksi 56% ja positiiviseksi ennustearvoksi 46%. MFE-menetelmien tarkkuutta rajoittavat useat tekijät. Lähimmän naapurin vapaan energian malli ei ole täydellinen, lisäksi MFE-algoritmit jättävät monia muitakin laskostumiseen vaikuttavia tekijöitä pois ennustuksesta. Lisäksi joillakin RNA-sekvensseillä voi olla useita erilaisia toiminnallisia rakenteita. Tästä syystä alioptimaalisten rakenteiden ennustaminen on tärkeää. RNAfold ja Mfold ennustavat kuitenkin optimaalisen MFE-rakenteen lisäksi vain muutaman vaihtoehtoisen alioptimaalisen rakenteen käymättä läpi kaikkia potentiaalisia vaihtoehtoja. [Mathews, 2006] Muutamiin aiempiin artikkeleihin perustuen Lawrence & Ding julkaisivat uuden algoritmin, joka käy tilastollisilla menetelmillä läpi teoriassa kaikki alioptimaaliset rakenteet. [Ding and Lawrence, 2003] Algoritmiin perustuva ohjelmisto on nimeltään Sfold.

3.1.4. Sfold

Sfold on joukko kirjastoja ja funktioita RNA-sekvenssien analysointiin. Ohjelmisto on ladattavissa vapaasti omalle koneelle tai käytettävissä WWW-palvelimella. Sfold-ohjelma ennustaa RNA-sekundaarirakenteen muodostamalla tilastollisen otannan RNA-sekvenssin Boltzmannin jakaumasta ja lajittelemalla sekundaarirakenteiden jakauman käyttämällä uusimpia vapaan energian sääntöjä [Ding and Lawrence, 2003]. Sfold tarjoaa uudenlaisen, todennäköisesti aiempia MFE-menetelmiä tarkemman, menetelmän RNA-sekundaarirakenteiden ennustamiseen.

3.2. RNA-sekvenssien kovarianssianalyysi

Toinen yleisesti käytetty metodi sekundaarirakenteiden etsimiseen tai ennustamiseen on RNA sekvenssien emästen kovariaatiot eli yhteisvaihtelut. Menetelmän perusidea on etsiä sekvensseistä malli-RNAn kanssa Watson-Crick parien suhteen samankaltaisia positioita, jotka todennäköisesti myös poimuttuvat samalla tavalla. Kuvassa 6 on selvitetty tRNA-sekvenssien kovariaatiot merkkijonotäsmäyksellä.



Kuva 6: tRNA-kovariaatiot

Eddyn ja Durbinin kehittämä kovarianssimalli [Eddy and Durbin, 1994] käyttää stokastisia kontekstittomia kielioppeja (SCFG, Stochastic Context Free Grammar). Kontekstiton kielioppi on joukko sääntöjä, joissa säännön vasemmalla puolella on yksi merkki ja oikealla puolella nolla tai enemmän merkkejä. Eddyn ja Durbinin mallissa SCFG voidaan muodostaa täsmätyistä RNA-sekvensseistä sekä niihin liittyvistä sekundaarirakenteista. Käyttämällä Eddyn ja Durbinin algoritmia [Eddy, 2002] voidaan RNA-sekvenssistä

muodostaa SCFG ja kovarianssimalli, jota hyödyntäen voidaan koko genomista tai sekvenssitietokannasta etsiä samankaltaisia rakenteita.

3.2.1. Kontekstittomat kieliopit

Noam Chomskyn alun perin luonnollisille kielille kehittämä formaalien kielten teoria [Chomsky, 1956], on keskeinen tietojenkäsittelyn teoria, sekä merkkijonotäsmäyksen työkalu. Chomskyn hierarkia luokittelee formaalit kieliopit neljään luokkaan. Kontekstittomat kieliopit kuuluvat luokkaan 2 ja ne voidaan tunnistaa pinoautomaatilla.

Chomskyn kielioppi on järjestelmä G = (Vt,Vn,P,S), missä

Vt on äärellinen apumerkkien aakkosto
Vn on äärellinen perusmerkkien aakkosto
P on äärellinen sääntöjen joukko
S on joukon Vn erillinen alkumerkki

Kontekstittomilla kielillä jokainen sääntö P on kontekstiton, sääntöjen vasemmat puolet muodostuvat täsmälleen yhdestä apumerkistä $Vn \rightarrow (Vt \ U \ Vn)^*$.

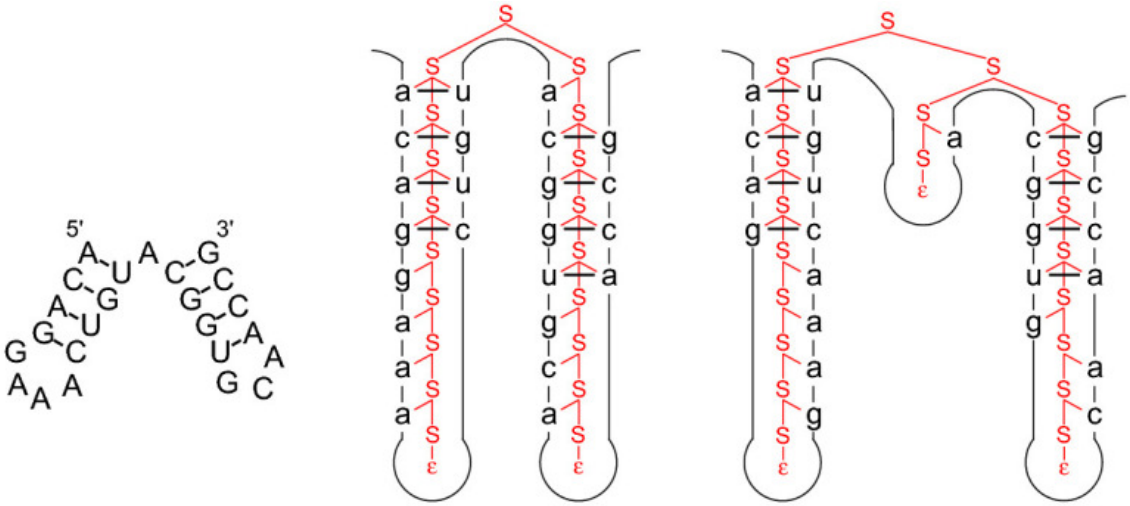
Esimerkiksi seuraava kielioppi muodostaa RNA rakenteen, ε on lopetusmerkki eli tyhjä merkkijono. Vn = {S}, Vt = {a,c,g,u} ja P =

$S \rightarrow aSu \mid uSa \mid cSg \mid gSc$
$S \rightarrow aS \mid cS \mid gS \mid uS$
$S \rightarrow Sa \mid Sc \mid Sg \mid Su$
$S \rightarrow SS$
$S \rightarrow \varepsilon$

Eli apumerkkien joukko Vt koostuu RNA nukleotideista ja säännöt yksinkertaistaen hyväksyvät joko Watson-Crick emäsparit tai yksittäiset nukleotidit. Säännöt voidaan kirjoittaa myös lyhyemmin käyttämällä merkkiä a kuvaamaan kaikkia nukleotideja, Vt = {a} ja sääntöä S → aSâ merkitsemään sitä että a pariutuu â kanssa.

P = $S \rightarrow aSâ \mid aS \mid Sa \mid SS \mid \varepsilon$

Eri emäsparien pariutumistodennäköisyydet voidaan myös tallettaa matriisiin. Kieliopin hyväksymä rakenne voidaan esittää jäsennyspuulla (parse tree), joka on jäsentäjän tuottama kieliopillinen rakenne muodollisen kieliopin pohjalta. Kuvassa 7 on RNA-sekundaarirakenteen muodostava jäsennyspuu.



Kuva 7: RNA sekvenssin jäsennyspuu [Dowell and Eddy, 2004]

3.2.1. RNAalifold

Dynaamista ohjelmointia ja termodynaamisia parametreja yhdistävä Alifold-algoritmi [Hofacker et al, 2002] on lisätty osaksi ViennaRNA pakettia. RNAalifold-ohjelma ennustaa yhteisen samankaltaisen rakenteen joukolle rinnastettuja sekvenssejä. Algoritmin kompleksisuus L pituisilla sekvensseillä on $O(L^3)$, mikä on selkeästi pienempi SCFG–algoritmeihin nähden. RNAalifold hyväksyy joukon rinnastettuja sekvenssejä ClustalW –formaatissa. ClustalW on tunnettu sekvenssirinnastuksen työkalu [Thompson et al, 1994].

3.2.2. Pfold

Pfold [Knudsen and Hein, 2003] yhdistää fylogeneettistä informaatiota ja kontekstittomia kielioppeja määrittelemään parhaan yhteisen sekundaarirakenteen joukolle RNA-sekvenssejä, joilla oletetaan olevan samankaltainen rakenne. Pfoldin substituutiomatriisi perustuu suureen joukkoon julkaistuja tRNA sekvenssejä. Pfoldin kontekstiton kielioppi

osoittautui tehokkaimmaksi Eddy et al. artikkelissa [Dowell and Eddy, 2004], jossa testattiin yhdeksän erilaisen kieliopin ennustustarkkuutta. Pfoldin käyttämä kielioppi oli testijoukon yksinkertaisin, mutta silti suhteellisen tehokas.

Pfoldin ja alifoldin suurin ongelma, on valmiiden täsmättyjen sekvenssien vaatimus syötteenä. Pfold olettaa että kaikilla syötetyillä sekvensseillä on identtinen sekundaarirakenne. Tämä muodostaa ongelmallisen kehän kun usean sekvenssin yleinen täsmäys vaatii kaikille samankaltaisen sekundaarirakenteen, mutta sellaisen sekundaarirakenteen määrittäminen vaatisi ensin hyvän yleisen täsmäyksen.

3.2.3. RNAprofile

RNAprofile –algoritmi etsii konservoituneimmat alueet joukolle RNA-sekvenssejä, käyttäen samankaltaisuusmittana sekä sekvenssin että sekundaarirakenteen ominaisuuksia. RNAprofile mahdollistaa määrättyjen rakenteiden ja alueiden etsimisen joukosta RNA-sekvenssejä. [Pavesi et al, 2004] Algoritmi tarvitsee syötteenä haetun sekvenssin sekä haettavien sekvenssien lisäksi vain halutun silmukoiden määrän yhteisessä rakenteessa. Algoritmi valitsee ensin syötesekvensseistä joukon potentiaalisia osasekvenssejä, joista muodostuu haluttu määrä silmukoita. Valinta perustuu dynaamiseen ohjelmointiin ja MFE-parametreihin. Kun potentiaaliset osasekvenssit on valittu, niille määritellään etäisyydet suhteessa toisiinsa Needleman-Wunsch-algoritmin [Needleman and Wunsch, 1970] variaatiolla, joka huomioi sekä sekvenssin että sekundaarirakenteen ominaisuudet. Tulokset muunnetaan profiileiksi, joita lopulta verrataan haetusta sekvenssistä muodostettuun profiiliin. Kun koko aineisto on käsitelty, saadaan tuloksena aineistosta parhaiten esimerkkirakenteeseen täsmäävät alueet.

4. Sekundaarirakenteiden rinnastus

Sulkunotaatio kuvaa 2-ulotteisen sekundaarirakenteen 1-ulotteisena sekvenssinä mahdollistaen periaatteessa rakenteiden vertailun sekvenssirinnastuksella. Käsittelen seuraavassa kappaleessa sekvenssirinnastuksen perusmenetelmää ja muutamaa keskeistä bioinformatiikan algoritmia.

4.1. Merkkijonotäsmäyksestä

Kahden merkkijonon välinen editointietäisyys, Levenshteinin etäisyys, on pienin määrä operaatioita, joiden avulla toinen merkkijono voidaan muuttaa toiseksi. Sallittuja operaatioita ovat normaalisti yhden merkin lisääminen, poistaminen tai korvaaminen muulla merkillä. Levenshteinin etäisyyden laskeva dynaamiseen ohjelmointiin perustuva algoritmi käyttää (n + 1) × (m + 1) -kokoista matriisia, missä n ja m ovat merkkijonojen pituudet. Oheinen perusdynaamisen ratkaisun funktio saa syötteeksi kaksi merkkijonoa, A, jonka pituus on lenA, ja B, jonka pituus on lenB, laskien niiden välisen editointietäisyyden.

```
1:   int LevenshteinDistance(char A[1..lenA], char B[1..lenB])
2:
3:   declare int d[0..lenA, 0..lenB]
4:   declare int i, j, cost
5:
6:   for i from 0 to lenA:
7:       d[i, 0] = i
8:   for j from 0 to lenB:
9:       d[0, j] = j
10:
11:  for i from 1 to lenA:
12:      for j from 1 to lenB:
13:          if A[i] = B[j] then cost = 0
14:                         else cost = 1
15:          d[i, j] = minimum (
16:                              d[i-1, j  ] + 1,     // poisto
17:                              d[i  , j-1] + 1,     // lisäys
18:                              d[i-1, j-1] + cost   // korvaus
19:                            )
20:  return d[lenA, lenB]
```

Algoritmi 1: Levenshtein distance

Riveillä 3 ja 4 varataan muistia (lenA + 1) × (lenB + 1) kokoiselle kustannusmatriisille, laskurimuuttujille sekä kustannukselle. Rivien 6 ja 8 silmukat alustavat matriisin ensimmäisen rivin ja sarakkeen rinnastettavien sekvenssien merkkien indekseillä. Rivien 11 ja 12 silmukat käyvät läpi matriisin alkiot läpi riveittäin ja sarakkeittain. Jokaisen alkion kohdalla lasketaan kustannukset eri operaatioille kyseisessä kohdassa, hyödyntäen aiemmin laskettuja arvoja, jonka jälkeen pienin kustannus tallennetaan uudeksi alkioksi. Algoritmi laskee alusta alkaen yhä pidempien ja pidempien sekvenssien välisen etäisyyden, hyödyntäen dynaamisen ohjelmoinnin periaatteiden mukaan aiemmin laskettua tietoa, kunnes lopulta saadaan kokonaisten sekvenssien välinen editointietäisyys.

Joissakin tapauksissa voi olla tarkoituksenmukaista määritellä editointikustannukset riippuvaisiksi käsiteltävistä merkeistä. Tähän perustuu mm. Needleman-Wunsch –algoritmi.

4.1.2. Needleman-Wunsch

Jo vuonna 1970 julkaistu Needleman-Wunsch algoritmi kehitettiin kahden proteiinisekvenssin globaaliin rinnastukseen. Needleman-Wunsch on ensimmäinen dynaamiseen ohjelmointiin perustuva biologisten sekvenssien rinnastusalgoritmi [Needleman and Wunsch, 1970]. Geenien koodaavat alueet sijaitsevat DNA-kierteessä lyhyinä pätkinä. Vastaavasti kun kahta eri mittaista biologista sekvenssiä verrataan toisiinsa, täytyy sekvenssejä usein pilkkoa parhaan täsmäyksen saavuttamiseksi. Termi avauskustannus (gap penalty) tarkoittaa jotakin lukemaa, joka lisätään kahden sekvenssin väliseen etäisyyteen, mikäli sekvenssiin on avattu aukkoja täsmäystä varten. Rinnastuksen avauskustannus voidaan laskea kaavalla c + (p-1)d, missä c on aukon avauskustannus (gap opening penalty), p on aukon pituus ja d on aukon laajentamisen kustannus (gap extension penalty). Eri merkkien pariutumistodennäköisyydet tallennetaan matriisiin. Nukleotidi- ja proteiinisekvensseillä tyypilliset parametrit ovat tulleet vuosien mittaan tutkituista molekyylibiologian ominaisuuksista.

Kun lasketaan merkkijonojen n ja m välinen täsmäys, muodostetaan aluksi n × m kokoinen matriisi F johon lasketaan kaikkien merkkien täsmäykset kaavalla

$$F[i,j] = \max\{ F[i-1, j-1] + S[n_i, m_j], F[i, j-1] + d, F[i-1, j] + d \},$$

missä $S[n_i, m_j]$ tarkoittaa kahden merkin välistä pariutumistodennäköisyyttä ja d poiston kustannusta. Seuraava funktio muodostaa F matriisin kahdelle sekvenssille.

```
1:   int[] computeFmatrix(char A[1..lenA], char B[1..lenB])
2:
3:   declare int F[0..lenA, 0..lenB]
4:   declare int i, j, d
5:
6:   for i from 0 to lenA-1:
7:     F[i,0] = 0
8:   for j from 0 to lenB-1:
9:     F[0,j] = 0
10:
11:  for i from 1 to lenA:
12:    for j from 1 to lenB:
13:      F[i,j] = maximum (
14:                          F[i-1,j-1] + S(A[i], B[j])      // valinta 1
15:                          F[i-1,  j] + d                  // valinta 2
16:                          F[i,  j-1] + d                  // valinta 3
17:                        )
18:  return F
```

Algoritmi 2: Needleman-Wunsch F-matriisi

Rivillä 14 funktio S(A[i], B[j]) palauttaa kahden merkin pariutumistodennäköisyyden matriisista. Kun F matriisi on muodostettu, sen oikeassa alakulmassa on suurimman pariutumisen kustannus. Optimaalinen rinnastus ja pienin editointietäisyys jäljitetään nyt matriisista alkaen oikeasta alakulmasta takaisin vasempaan yläkulmaan.

```
1:   declare string alignA, alignB
2:
3:   declare int i = lenA - 1
4:   declare int j = lenB - 1
5:
6:   while (i > 0 and j > 0):
7:    score = F[i,j]
8:    diag = F[i-1, j-1]
9:    up = F[i, j-1]
10:   left = F[i-1, j]
11:
12:   if (score == diag + S(A[i], B[j])) then
13:         alignA = A[i] + alignA
14:         alignB = B[j] + alignB
15:         i = i - 1
16:         j = j - 1
17:
18:   else if (score == left + d) then
19:         alignA = A[i] + alignA
20:         alignB = "-" + alignB
21:         i = i - 1
22:
23:   else (score == up + d)
```

```
24:          alignA = "-" + alignA
25:          alignB = B[j] + alignB
26:          j = j - 1
27:
28:   while (i >= 0):
29:    alignA = A[i] + alignA
30:    alignB = "-" + alignB
31:    i = i - 1
32:
33:   while (j >= 0):
34:    alignA = "-" + alignA
35:    alignB = B[j] + alignB
36:    j = j - 1
```

Algoritmi 3: Needleman-Wunsch

Rivillä 1 määritellään tyhjät merkkijonot rinnastusten tallentamiseen. Riveillä 3 ja 4 Indeksit i ja j asetetaan matriisin oikeaan alanurkkaan. Tämän jälkeen haetaan rivin 6 silmukalla matriisista optimaalinen editointipolku siten, että indeksin lukemaa verrataan rivillä 12 matriisissa vasemmalla yläviistossa olevaan lukemaan. Mikäli indeksin lukema on saatu siitä, tallennetaan molempiin merkkijonoihin täsmäys ja siirretään indeksiä kyseiseen kohtaan. Riveillä 18 ja 23 vastaavasti siirrytään matriisissa vasemmalle tai ylös ja avataan toiseen sekvenssiin aukko. Matriisin läpikäynti loppuu vasempaan ylänurkkaan F[0,0]. Rivien 28 ja 33 silmukat suorittavat täsmäyksen loppuun mikäli i tai j ovat vielä nollaa suurempia. Tämän jälkeen rivillä 1 määritellyissä merkkijonoissa on optimaalinen globaali täsmäys.

Seuraava esimerkki suorittaa Needleman-Wunsch täsmäyksen kahdelle lyhyelle merkkijonolle A: babcca ja B: bccca. Kuvassa matriisi S (Similiarity matrix) tallettaa merkkien pariutumistodennäköisyydet; tässä tapauksessa ne ovat keksittyjä, avauskustannus d = -1.

S	a	b	c
a	2	0	1
b	0	2	-1
c	1	-1	2

F		b	a	b	c	c	a
	0	0	0	0	0	0	0
b	0	**2**	0	2	1	0	0
c	0	1	**3**	**2**	4	3	2
c	0	0	2	2	**4**	6	5
c	0	-1	1	1	4	**6**	7
a	0	0	0	0	2	6	**8**

Taulukko 1: S ja F matriisit

Matriisiin F on ensin laskettu suurimmat lukemat eri merkeille algoritmilla 2: computeFmatrix. Rivi 0 ja sarake 0 alustetaan ensin tyhjäksi, jotta voidaan käyttää algoritmin laskusääntöjä. Matriisin täyttö alkaa indeksistä F[1,1], johon tulee suurin arvo joka on matriisin S perusteella F[0,0] + S[1,1] = 0+2 = 2. Seuraavaan indeksiin F[1,2] tulee nyt F[1,1] + d = 2-1 = 1 kunnes lopulta matriisin oikeaan alanurkkaan saadaan täsmäyksen suurin tulos 8. Optimaalinen täsmäys voidaan nyt tulostaa palaamalla matriisissa takaisin algoritmin 3 mukaan; tarkistamalla vasemmalta yläviistosta, ylhäältä ja vasemmalta se lukema, josta nykyiseen indeksiin on päädytty, jonka jälkeen tulostetaan A tai B -sekvenssiin aukon merkki ”_”, riippuen editointipolun suunnasta. Tämän esimerkin tapauksessa indeksistä F[3,2] on siirrytty vasemmalle, jolloin merkkijonoon B on avattu aukko. Täsmäys on

A: babcca
B: bc_cca

ja suurin tulos saadaan laskemalla S(b,b)+S(a,c)+d+S(c,c)+S(c,c)+S(a,a) = 2+1-1+2+2+2 = 8 eli siis matriisin F oikeaan alanurkkaan syntynyt tulos.

4.1.3. Smith-Waterman

Needleman-Wunsch algoritmiin perustuvalla Smith-Waterman algoritmilla [Smith and Waterman, 1981] saadaan kahden sekvenssin optimaalinen paikallinen rinnastus, joka on molekyylibiologiassa huomattavasti käytetympi sovellus kuin globaali rinnastus. Suurin ero Smith-Watermanin ja Needleman-Wunschin välillä on se että Smith-Waterman-algoritmin F matriisiin ei tule negatiivisia lukuja, mikä tuo esiin paikalliset optimit rinnastukset. Editointipolun jäljitystä ei aloiteta matriisin oikeasta alanurkasta vaan suurimmasta luvusta, joka on myös suurimman täsmäyksen tulos. Ja editointipolkua seurataan takaisin vasemmalle, viistoon tai ylös suurimman luvun perusteella. Algoritmista on olemassa useita variaatioita. Kuten Needleman-Wunschilla, täsmäyksen laatu ja tulos riippuvat suuresti S-matriisin luvuista sekä editointietäisyydestä. Seuraavat pseudokoodiset algoritmit sisältävät kuitenkin pääperiaatteen, käyn ne läpi kahdella sekvenssillä A: abacabac ja B: ccaba. S matriisi on sama kuin Taulukossa 1 avauskustannus d = 1.

```
1:   declare int F[0..lenA, 0..lenB]
2:   declare int i, j, maxI, maxJ, diag, left, up
3:   declare int cost = 1
4:
5:   for i from 0 to lenA-1:
6:    F[i,0] = 0
7:   for j from 0 to lenB-1:
8:    F[0,j] = 0
9:
10:  for i from 1 to lenA:
11:   for j from 1 to lenB:
12:         diag = F[i-1,j-1] + S(A[i], B[j])
13:         left = max(0,F[i-1,j] - cost)
14:         up = max(0,F[i,j-1] - cost)
15:         F[i,j] = max(diag,left,up)
16:         if(F[i,j] > F[maxI,maxJ]) then
17:               maxI = i
18:               maxJ = j
```

Algoritmi 4: Smith-Waterman F-matriisi

F-matriisi on alussa kuten Needleman-Wunschilla, ensimmäinen rivi ja ensimmäinen sarake alustetaan nollaksi. Tämän jälkeen käydään loput indeksit läpi rivien 10 ja 11 silmukoilla siten että muuttujaan diag tallennetaan viereinen indeksi vasemmalta ylhäältä lisättynä indeksin merkkien täsmäystodennäköisyydellä matriisista S. Muuttujaan left tulee joko nolla tai vasemman indeksin arvo vähennettynä avauskustannuksella, mikäli se on nollaa suurempi. Muuttujaan up arvo lasketaan vastaavasti yläpuolella olevasta indeksistä. Tämän jälkeen rivillä 15 tallennetaan indeksiin suurin kolmesta aiemmin lasketusta luvusta. Lisäksi riveillä 16-18 säilytetään ja tarkistetaan matriisin F suurimman alkion koordinaatit, tämän voi toki suorittaa myöhemminkin, algoritmin kompleksisuuteen sillä ei ole vaikutusta. F-matriisin laskemisen jälkeen editointipolun jäljitys aloitetaan matriisin suurimmasta alkiosta ja jäljitys päättyy nollaan.

```
1:   declare i = maxI
2:   declare j = maxJ
3:
4:   while (F[i,j] > 0)
5:    diag = F[i-1,j-1]
6:    top = F[i,j-1]
7:    left = F[i-1,j]
8:
9:    if (diag >= top and diag >= left) then
10:         alignA = A[i] + alignA
11:         alignB = B[j] + alignB
12:         i = i - 1
13:         j = j - 1
14:
15:   else if (top >= diag and top >= left) then
16:         alignA = A(i) + alignA
17:         alignB = "-" + alignB
18:         j = j - 1
19:
20:   else
21:         alignA = "-" + alignA
22:         alignB = B(j) + alignB
23:         i = i - 1
24:
25:  while (i >= 0):
26:     alignA = A(i) + alignA
27:     alignB = "-" + alignB
28:     i = i - 1
29:
30:  while (j >= 0):
31:     alignA = "-" + alignA
32:     alignB = B(j) + alignB
33:     j = j - 1
```

Algoritmi 5: Smith-Waterman

Esimerkin tapauksessa suurin indeksin arvo on nyt 9, josta takaisinjäljitys aloitetaan. Riviltä 4 alkavaa silmukkaa suoritetaan kunnes indeksissä on nolla. Indeksin vasen, ylä ja viistosti vasemmalla ylhäällä olevat indeksien arvot tallennetaan muuttujiin, jonka jälkeen näistä suurin valitaan seuraavaksi indeksiksi. Mikäli joka suunnassa on sama indeksi, valitaan ensin viisto suunta. Taulukossa 2 siirrytään siis suurimmasta alkiosta 9 viistosti 7-5-3 kunnes suurin löytyy ylhäältä.

F		a	b	a	c	a	b	a	c
	0	0	**0**	0	0	0	0	0	0
c	0	1	0	**1**	**2**	1	0	1	2
c	0	1	0	1	**3**	3	2	1	3
a	0	2	1	2	2	**5**	4	4	3
b	0	1	4	3	2	4	**7**	6	5
a	0	2	3	6	5	4	6	**9**	8

Taulukko 2: Smith-Waterman F-matriisi

Eli kun esimerkin tapauksessa sekvenssien A ja B suorat täsmäykset häviävät, avautuu sekvenssiin A aukko. Mikäli avauskustannus d olisi -2, aukkoa ei syntyisikään vaan algoritmi täsmäisi a ja c merkit substituutiomatriisin S lukujen mukaisesti.

A: aba_cabac
B: ___ccaba_

Algoritmista 5 puuttuu sekvenssien lopusta mahdollisesti puuttuvien merkkien täydennys, mikä tosin onnistuu kahdella vastaavalla silmukalla kuin riveillä 25 ja 30. Nuo ovat kuitenkin enemmän sovelluksesta riippuvaisia ominaisuuksia.

4.1.4. Substituutiomatriisit

Substituutiomatriisia käytetään tietyissä rinnastusalgoritmeissa kertomaan kuinka hyvin tai millä todennäköisyydellä tietyt sekvenssin alkiot täsmäävät muihin alkioihin. Mikäli jokainen sekvenssin alkio täsmäisi vain samanlaiseen alkioon, substituutiomatriisi olisi identiteettimatriisi, jossa diagonaaliakseli koostuu ykkösistä ja muut alkiot nollasta. Koska

sekvenssirinnastusta käytetään bioinformatiikassa pääasiassa geeni- tai proteiinisekvenssien rinnastamiseen, ja biologisilla sekvensseillä on tapana muuttua vähitellen evoluutiossa syntyvien mutaatioiden johdosta, käytetään biologisten sekvenssien rinnastuksessa usein tiettyjä matriiseja, joiden painotukset on selvitetty kokeellisesti ajan myötä aminohappojen tai nukleotidien kemiallisiin, rakenteellisiin tai funktionaalisiin ominaisuuksiin perustuen, sekä intuitiivisesti kokeiden kautta. Yleisimmin käytetyt matriisit ovat PAM ja BLOSUM – matriisien eri versioita. [Henikoff and Henikoff, 1992]

4.1.5. BLAST

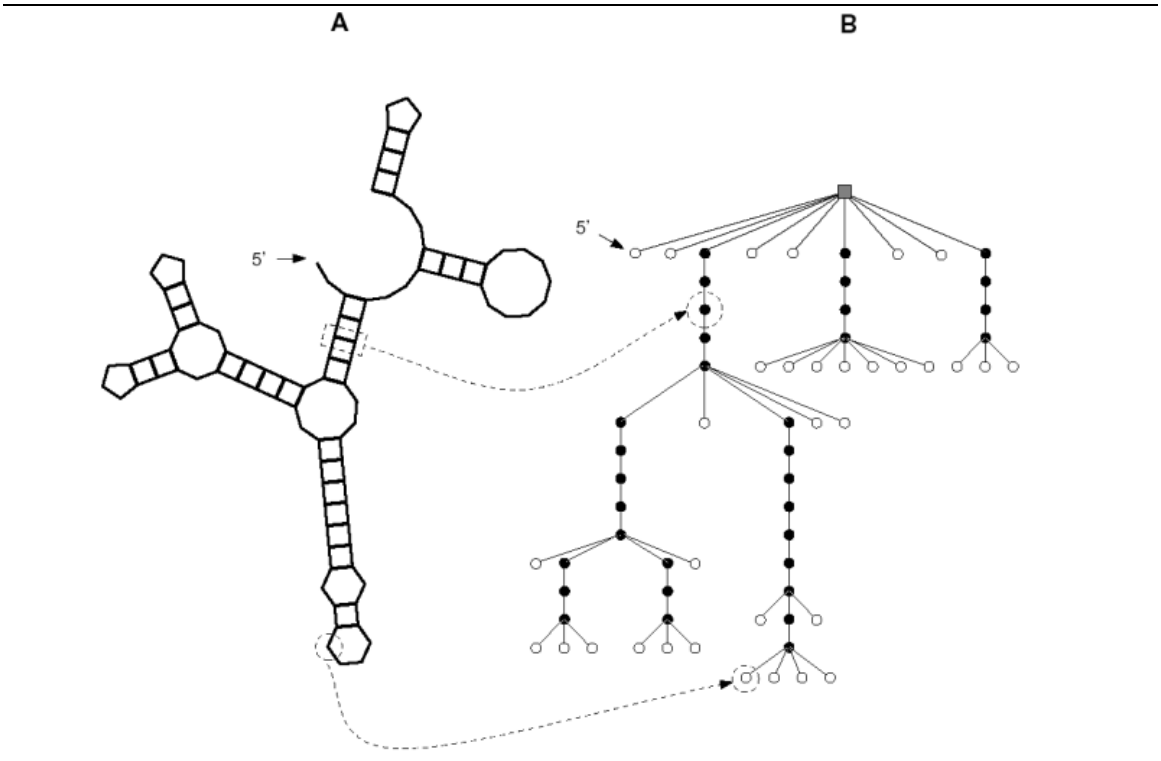
Basic Local Align Search Tool eli BLAST on yleisesti käytetty bioinformatiikan työkalu nukleotidi- tai aminohapposekvenssien lokaaliin rinnastamiseen. Vuonna 1990 julkaistu artikkeli [22] oli 90-luvun kaikkien tieteenalojen eniten viitatuin tieteellinen artikkeli [ScienceWatch, 2003]. BLASTin suosio perustuu sen nopeuteen, joka mahdollistaa sekvenssien etsimisen nopeasti genomin mittaisista tietokannoista. BLAST ei ole käytännössä yhtä tarkka kuin Smith-Waterman mutta suurilla aineistoilla se on huomattavasti nopeampi ja kuitenkin useissa tapauksissa riittävän tarkka menetelmä sekvenssien täsmäytykseen.

BLAST vaatii ensin syötteenä yhden tai useamman sekvenssin, joista muodostetaan indeksoitu tietokanta tietyn mittaisille osasekvensseille, joiden pituus on w. Tyypillisesti proteiinisekvensseillä w = 3 ja nukleotidisekvensseillä w = 11. Myös valmista tietokantaa, yleensä eri organismien genomeja tai proteiinilistoja, voi käyttää. Tämän jälkeen ensimmäisessä vaiheessa BLAST lukee syötesekvenssit, joita verrataan tietokantaan. Syötesekvenssit pilkotaan myös w mittaisiksi niin, että yhdestä sekvenssistä, jonka pituus on n, muodostuu n – w + 1 osasekvenssiä. Toisessa vaiheessa osasekvensseille haetaan tietyn rajan ylittävät vastineet tietokannasta ja nämä lisätään tarkemmin analysoitavien sekvenssien listaan. Täsmäystulosten laskemiseen käytetään sekvensseistä riippuen PAM tai BLOSUM-matriiseja. Myös muita matriiseja voi käyttää. Kun sekvenssien osille on saatu lista osatäsmäyksiä, niiden täsmäytystä jatketaan molempiin suuntiin. Mikäli löytyy täsmäys riittävän suurella tuloksella se pääsee kolmanteen vaiheeseen. Kolmannessa vaiheessa lasketaan lopullinen täsmäys Smith-Waterman algoritmin variaatiolla, jonka jälkeen tilastollisesti merkitsevät täsmäykset näytetään käyttäjälle. BLASTista on olemassa useita sovelluksia ja ohjelmistoversioita.

4.2. Puun editointietäisyys

Puu on tietojenkäsittelytieteessä hyvin keskeinen tietorakenne, joka mahdollistaa nopean tiedon etsinnän ja lajittelun. Tietoalkiot tallennetaan puissa solmuihin. Juuri on puun aloittava solmu ja polku on kahden solmun välinen yhteys. Vanhempi viittaa solmun yllä olevaan solmuun, lapsi on solmun alapuolella oleva solmu. Lehti on solmu, jolla ei ole lapsia. Taso viittaa syvyyteen, jolla solmu on. Juuren taso on nolla. Samalla tasolla olevat solmut ovat sisaruksia. Metsäksi kutsutaan puuta, joka sisältää alkioissaan muita puita; myös puun osaa voidaan kutsua osametsäksi. Järjestetyn puun solmujen lapsilla on jokin järjestys; useat puu-algoritmit vaativat että solmujen lapsia voi käsitellä järjestyksessä.

RNA-sekundaarirakenteet voidaan esittää puumuodossa. Esimerkiksi kuvassa 8 puu muodostetaan alkaen juuresta siten että laskostunutta sekvenssiä luetaan alkaen 5' päästä, muodostaen jokaiselle nukleotidille oma solmu juuren lapseksi. Mikäli nukleotidi on pariton eli sulkunotaatiossa on piste, luodaan seuraava solmu samalle tasolle. Parillisen '(' nukleotidin tapauksessa seuraava solmu luodaan tämän lapseksi, vastakkaisen parin ')' kohdalla taas palataan puussa ylöspäin edelliselle tasolle järjestys säilyttäen.



Kuva 8: RNA-sekundaarirakenteen puumuoto [Hofacker et al., 1994]

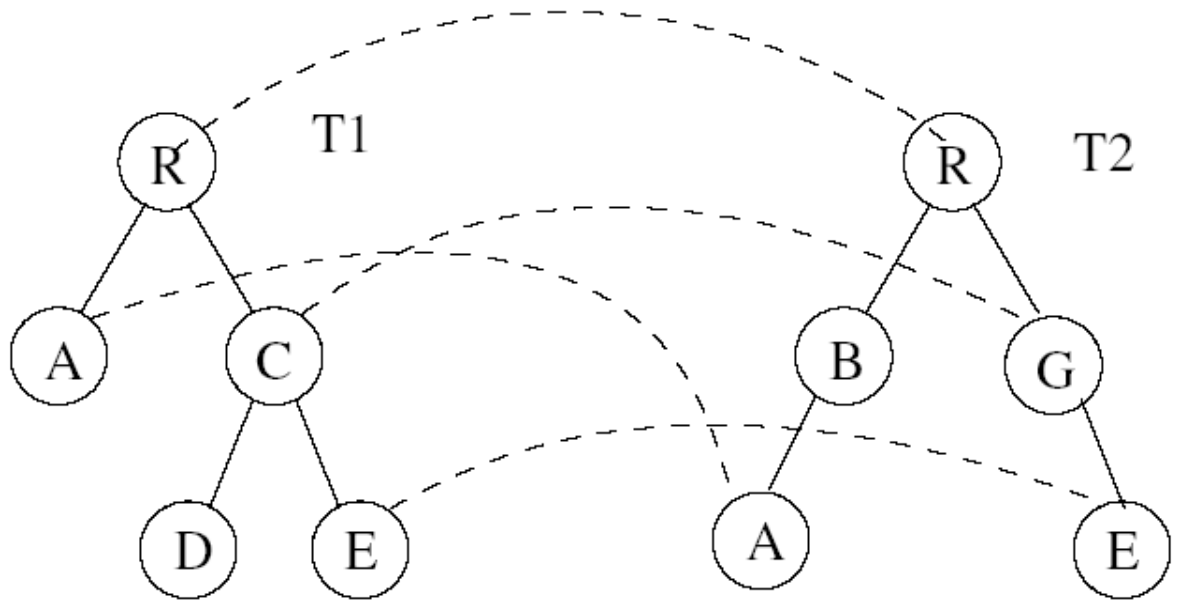
Puiden vertailua sovelletaan laskennallisen biologian lisäksi mm. XML-dokumenttien vertailussa, kuvantunnistuksessa ja kääntäjien optimoinnissa. Yleinen tapa vertailla kahta järjestettyä puuta on määrittää niiden editointietäisyys, jolla puu voidaan muuttaa toiseksi. Kahden puun välisen editointietäisyyden määrittäminen on vanha ja melko paljon tutkittu tietojenkäsittelytieteen ongelma. Ensimmäisen algoritmin esitti Tai jo vuonna 1979 [Tai, 1979]. Nopeammat ja nykyään keskeiset puiden vertailuun käytetyt algoritmit ovat Zhang-Shasha [Shapiro and Zhang, 1988] ja Klein [Klein, 1998]. Molemmat algoritmit perustuvat dynaamiseen ohjelmointiin ja ne ovat myös periaatteessa Levenshteinin etäisyyden laajennoksia. Järjestetty puu, joka koostuu pelkistä lehdistä, voidaan toisaalta yleistää tavalliseksi sekvenssiksi. Zhang-Shashan pahimman tapauksen kompleksisuus on $O(n^4)$, Kleinin $O(n^3 \log(n))$. Molempien algoritmien käytännön tehokkuus on kuitenkin paljolti riippuvainen verrattavien puiden muodosta.

4.2.1. Perusmääritelmät

Oletetaan että puiden T1 ja T2 jokainen solmu on nimetty jollakin merkillä aakkostosta Σ, ja λ tarkoittaa tyhjää merkkiä, joka ei kuulu aakkostoon Σ. Editointioperaatiot voidaan esittää muodossa $a \rightarrow b$, missä a on joko λ tai jokin puun T1 solmun symboli ja b on joko λ tai jokin puun T2 solmun symboli.

Muodollisesti editointietäisyys D kahden puun T1 ja T2 välillä voidaan määrittää $D(T1, G) = \min \{ \gamma(S) \}$, missä S on editointioperaatioiden sekvenssi, joilla puu T1 muokkautuu T2 puuksi ja γ on editointioperaation kustannus. Editointioperaatioita on kolme, vaihto, poisto ja lisäys. Operaatio $a \rightarrow b$ on vaihto mikäli $a \neq \lambda$ ja $b \neq \lambda$, poisto mikäli $b = \lambda$ ja lisäys mikäli $a = \lambda$. Kustannus γ voidaan määritellä tapauskohtaisesti, kuitenkin niin että $\gamma(a \rightarrow b) \geq 0$, $\gamma(a \rightarrow a) = 0$, $\gamma(a \rightarrow b) = \gamma(b \rightarrow a)$ ja $\gamma(a \rightarrow c) \leq \gamma(a \rightarrow b) + \gamma(b \rightarrow c)$. Puiden T1 ja T2 välinen etäisyys $\gamma(S)$ on siis pienin yksittäisten editointioperaatioiden summa.

Editointietäisyyden kartoitus (mapping) tarkoittaa editointioperaatioiden esitystä kahden puun T1 ja T2 välillä. Kartoitus voidaan tallentaa matriisiin, jonka koko on $L(T1) \times L(T2)$, missä L(T) tarkoittaa puun T solmujen määrää. Olkoon t[i] jokin puun T solmu ja T[i] osapuu, jonka juuri sijaitsee kohdassa t[i].



Kuva 9: kartoitus

Kuvassa 9 on esimerkki kartoituksesta puiden T1 ja T2 välillä. Muodollisesti kartoitus on kolmikko (M,T1,T2), missä M on mikä tahansa pari indeksejä (i,j) joille pätee:

1. $1 \leq i \leq |T1|$, $1 \leq j \leq |T2|$
2. Jokaiselle parille (i_1,i_2) ja (j_1,j_2)
   a. $i_1 = i_2$ jos ja vain jos $j_1 = j_2$
   b. $t_1[i_1]$ on $t_1[i_2]$ vasemmalla puolella jos ja vain jos $t_2[j_1]$ on $t_2[j_2]$ vasemmalla puolella
   c. $t_1[i_1]$ on $t_1[i_2]$ jälkeläinen jos ja vain jos $t_2[j_1]$ on $t_2[j_2]$ jälkeläinen

Tain algoritmi käyttää esijärjestystä solmujen nimeämiseen. Esijärjestyksessä juuri saa ensimmäisen numeron, tässä tapauksessa 1, jonka jälkeen kaikki osapuut numeroitaan vasemmalta oikealle. Olkoon D(T1[1..i], T2[1..j]) editointietäisyys T1[Hofacker et al., 1994] ja T1[i] sekä T2[Hofacker et al., 1994] ja T2[j] välillä. Tai käyttää vaihtoa, lisäystä ja hävitystä määrittääkseen editointietäisyyden puiden T1 ja T2 välillä.

4.2.2. Zhang-Shasha

Zhang-Shasha-algoritmi kahden puun välisen editointietäisyyden määrittämiseksi käyttää samoja perusmäärittelyjä kuin Tain algoritmi. Esijärjestyksen sijaan Zhang-Shasha kuitenkin käyttää jälkijärjestystä, jossa numerointi aloitetaan vasemmanpuoleisesta latvasta, käyden läpi saman alipuun jälkeläiset, niiden vanhemman ja sitten vastaavasti oikeanpuoleiset alipuut, kunnes viimein juuri saa suurimman järjestysnumeron, joka on samalla puun solmujen lukumäärä. Algoritmi laskee dynaamisesti kartoituksen kaikille solmuille ja niiden alipuille myös tässä järjestyksessä.

Olkoon T[*i*..*j*] puun osarakenne, joka sisältää kaikki solmut *i* ja *j* välillä. Tätä osarakennetta kutsutaan metsäksi (forest). Yleisesti T[*i*..*j*] on järjestetty metsä. Zhang-Shasha perustuu kolmeen lemmaan, joissa *forestdist* tarkoittaa taulukkoa, joka sisältää metsien etäisyydet ja *treedist* taulukkoa, joka sisältää puiden etäisyydet. Olkoon $l(i)$ alipuun, jonka juuri on solmun *i* vasemmanpuoleisin jälkeläinen, *desc*(*i*) tarkoittaa solmun *i* jälkeläisten joukkoa. Θ tarkoittaa tyhjää puuta.

Lemma 1

(i) $forestdist(\Theta, \Theta) = 0$

(ii) $forestdist(l(i_1)..i, \Theta) = forestdist(l(i_1)..i\text{-}1, \Theta) + \gamma(t_1[i] \rightarrow \lambda)$

(iii) $forestdist(\Theta, l(j_1)..j) = forestdist(\Theta, l(j_1)..j\text{-}1) + \gamma(\lambda \rightarrow t_1[j])$

Lemma 2

Olkoon $i \in desc(i_1)$ ja $j \in desc(j_1)$

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i\text{-}1, l(j_1)..j) + \gamma(t_1[i] \rightarrow \lambda) \\ forestdist(l(i_1)..i, l(j_1)..j\text{-}1) + \gamma(\lambda \rightarrow t_2[j]) \\ forestdist(l(i_1)..l(i)\text{-}1, l(j_1)..l(j)\text{-}1) \\ \quad + forestdist(l(i_1)..i\text{-}1, l(j_1)..j\text{-}1) + \gamma(t_1[i] \rightarrow t_2[j]) \end{cases}$$

Lemma 3

Olkoon $i \in desc(i_1)$ ja $j \in desc(j_1)$

(1) jos $l(i) = l(i_1)$ ja $l(j) = l(j_1)$

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i\text{-}1, l(j_1)..j) + \gamma(t_1[i] \rightarrow \lambda) \\ forestdist(l(i_1)..i, l(j_1)..j\text{-}1) + \gamma(\lambda \rightarrow t_2[j]) \\ forestdist(l(i_1)..i\text{-}1, l(j_1)..j\text{-}1) + \gamma(t_1[i] \rightarrow t_2[j]) \end{cases}$$

(2) jos $l(i) \neq l(i_1)$ tai $l(j) \neq l(j_1)$

$$forestdist(l(i_1)..i, l(j_1)..j) = \min \begin{cases} forestdist(l(i_1)..i\text{-}1, l(j_1)..j) + \gamma(t_1[i] \rightarrow \lambda) \\ forestdist(l(i_1)..i, l(j_1)..j\text{-}1) + \gamma(\lambda \rightarrow t_2[j]) \\ forestdist(l(i_1)..i\text{-}1, l(j_1)..j\text{-}1) + treedist(i,j) \end{cases}$$

Todistukset lemmoille 1-3 löytyvät Apostolico et al. kirjasta. [Apostolico and Gali, 1997] Lemmat ehdottavat dynaamista ohjelmointia koska puiden editointietäisyydet lasketaan rekursiivisesti osapuiden editointietäisyyksistä.

Algoritmi 6 laskee puiden T1 ja T2 välisen editointietäisyyden laskemalla dynaamisesti pienimmät kustannuskartoitukset, jotka lasketaan jälkijärjestyksen perusteella jokaiselle solmulle ennen solmun läpikäyntiä. Tämä tapahtuu pitämällä kirjaa avainjuurista (keyroots). Avainjuuria ovat puun juuri ja kaikki solmut, joilla on vasen sisarus. Algoritmissa K[T] on taulukko kaikista puun T avainjuurista. Avainjuuria kuvan 9 puussa T1 ovat E,C ja R.

```
1: function ZhangShashaDistance(T1[0..n],T2[0..m])
2:
3: FD[0,0,0,0] = 0
4: for each x in K[T1]
5: for each y in K[T2]
6:  for i = L(x)..x do
7:    FD[L(x),i,0,0] = Access(L(x),i-1,0,0)+cost(delete(t1[i]))
8:  for j = L(y)..y do
9:    FD[0,0,L(y),j] = Access(0,0,L(y),j-1)+cost(insert(t2[i]))
10: for i = L(x)..x do
11:   for j = L(y)..y do
12:     m = min(Access(L(x),i-1,L(y),j)+cost(delete(t1[i])),
13:             Access(L(x),i,L(y),j-1)+cost(insert(t2[j])))
```

```
14:     if L(i) == L(x) and L(j) == L(y) then
15:       FD[L(x),i,L(y),j]=min(m,
16:            Access(L(x),i-1,L(y),j-1)+ cost(t1[i]→t2[i]))
17:     else
18:       FD[L(x),i,L(y),j]=min(m,
19:            Access(L(x),L(L(i))-1,L(y),L(L(j))-1)+D[i,j]))
20: return D[n,m]
```

Algoritmi 6: Zhang-Shasha Distance

Algoritmissa 6 esiintyvän Access –funktion toteutus on kuvattu algoritmissa 7.

```
1:  function Access(l,m,n,o)
2:
3:  lp = l, mp = m, np = n, op = o
4:  if l < m then
5:    lp = 0
6:    mp = 0
7:  if n < 0 then
8:    np = 0
9:    op = 0
10: return FD[lp,mp,np,op]
```

Algoritmi 7: Zhang-Shasha Access

Zhang-Shasha –algoritmin pahimman tapauksen kompleksisuus kahdelle puulle, joiden solmujen määrä on m ja n, on $O(m^2n^2) = O(n^4)$. [Shapiro and Zhang, 1988]

4.2.3. Klein

Kleinin algoritmi perustuu samoihin lemmoihin kuin Zhang-Shasha. Ero tulee pääasiassa siitä että Zhang-Shasha –algoritmissa rekursio suoritetaan aina osametsien oikeanpuoleisille juurille. Kleinin algoritmi suorittaa rekursion vasemmanpuoleiselle juurelle, jos suuremman verrattavan osapuun vasemmanpuoleisessa juuressa on vähemmän solmuja kuin oikeanpuoleisessa juuressa. Kleinin algoritmin pahimman tapauksen kompleksisuus on $O(m^2n \log n) = O(n^3 \log n)$.

Vuonna 2006 Demaine et al. esittivät abstraktissaan $O(n^3)$ ajassa toimivan algoritmin puiden editointietäisyyksien laskemiseksi. Demaine et al. algoritmi perustuu Kleinin algoritmiin mutta on selkeästi nopeampi. [Demaine et al, 2006]

4.2.4. RNAdistance

RNAdistance on ViennaRNA–pakettiin kuuluva ohjelma, jolla voi vertailla kahta sekundaarirakennetta joko puu- tai sekvenssirinnastuksella. Ohjelma lukee syötevirrasta sekundaarirakenteita joko sulkunotaatiolla (pseudo bracket) tai karkeasti kuvaavalla (coarse grained) notaatiolla, jossa rakenteet esitetään omilla merkeillään: H silmukka (hairpin), I sisäinen silmukka (interior loop), B pullistuma (bulge), M keskussilmukka (multiloop), S pino (stack), E ulkoinen emäs (external base). Merkit voidaan myös painottaa numeroilla. Lisäksi käytössä on nk. HIT-notaatio (Homeomorphically Irreducible Trees), jossa U tarkoittaa paritonta ja P parillista, numerot merkkien jälkeen ovat painotuksia. Seuraava esimerkki on ajettu kahdelle yksinkertaiselle silmukan ja varren muodostavalle rakenteelle, joilla suoran sekvenssitäsmäyksen etäisyys on 7.

```
[tl67597@bioinf ~]$ RNAdistance -DfhwcHWC -B

Input structure;  @ to quit
...(((...)))...
..(((((....)))))..
f: 7                                        full (tree edit)
...__(((..._)))__...
..._(((((....)))))..._
h: 7                                        HIT (tree edit)
(U3)((U3)P3)(U3)
(U2)((U4)P5)(U2)
w: 6                                        weighted coarse (tree edit)
(((H3)S3)E6)
(((H4)S5)E4)
c: 0                                        coarse (tree edit)
((H)S)
((H)S)
H: 7                                        HIT (string align)
(U3U3)(P3(U3U3)P3)(U3U3)
(U2U2)(P5(U4U4)P5)(U2U2)
W: 6                                        weighted coarse (string align)
(E6(S3(H3H3)S3)E6)
(E4(S5(H4H4)S5)E4)
C: 0                                        coarse (string align)
(S(HH)S)
(S(HH)S)
```

Notaation valinnalla on suuri merkitys etäisyyden määritykseen. Esimerkiksi karkeasti kuvaavalla notaatiolla kahden merkkijonon, joiden suora editointietäisyys on 7, etäisyydeksi muodostuu 0, koska molempien perusrakenne on sama ((H)S), silmukka ja varsi.

Vertailtaessa keskenään useita eri mittaisia rakenteita, ongelmaksi muodostuu myös lyhyen sekvenssin rakenteiden saamat keskinäiset lyhyet etäisyydet, vaikka rakenteet eivät välttämättä ole toisiaan lähellä. Ja vastaavasti hyvin täsmäävät pitkät rakenteet voivat saada suhteellisen pitkän etäisyyden toisiinsa nähden. Toisaalta karkeasti kuvaavalla notaatiolla lyhyet rakenteet täsmäävät usein täydellisesti suurten rakenteiden osiin, muodostaen näin suuren joukon 0-etäisyyden täsmäyksiä. Valitsin koeaineiston testaamiseen painotetun karkeasti kuvaavan notaation, joka toisaalta hävittää suurten ja pienten rakenteiden välistä eroa, mutta ei kuitenkaan ole liian karkea.

5. Klusterointi

Klusterointi (clustering), tunnetaan myös ryvästämisenä, voidaan määritellä lyhyesti tiedon automaattiseksi ryhmittelyksi samankaltaisten havaintojen joukkoihin. Klusterointi on yksi tiedonlouhinnan (data mining) perusmenetelmistä ja hyvin keskeinen tilastollinen menetelmä DNA-mikrosirujen (DNA-microarray) analysoinnissa.

5.1. Metriikat

Ennen klusterointia tulee päättää metriikka, jolla määritellään käsiteltävän aineiston alkioiden etäisyydet toisiinsa, metriikan valinta vaikuttaa hyvin paljon klusteroinnin lopputulokseen. Yleisimmin käytetty metriikka on euklidinen etäisyys kahdelle pisteelle x ja y joiden suora etäisyys D n-ulotteisessa avaruudessa saadaan kaavalla

$$D = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \ .$$

Toinen yleinen metriikka on Manhattan- tai City Block-etäisyys, jossa pisteiden x ja y etäisyys saadaan kaavalla

$$D = \sum_{i=1}^{N} | x_i - y_i | .$$

DNA-mikrosiru aineistojen klusteroinnissa hyvin yleisesti käytetty metriikka on Spearmanin korrelaatio, joka antaa korrelaatiokertoimen r kahdelle numerosarjalle
$x = \{ x_1, x_2, ..., x_n \}$ ja $y = \{ y_1, y_2, ... , y_n \}$, kaavalla

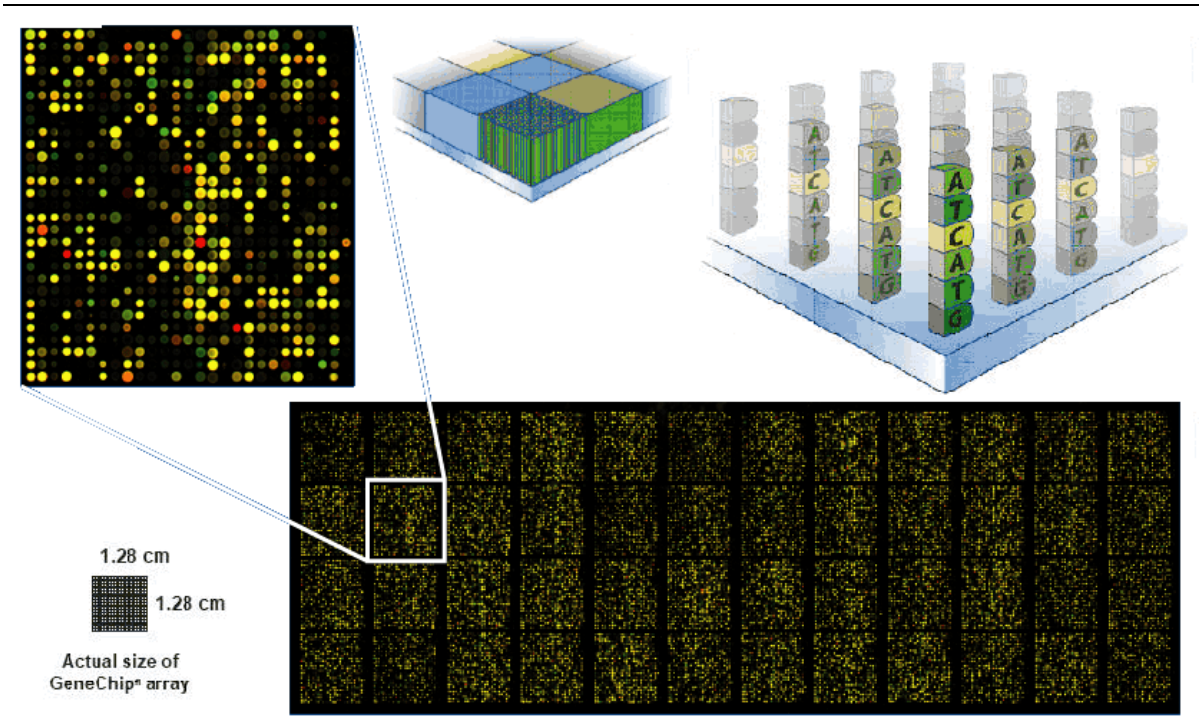
$$r = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\sigma_x} \right) \left(\frac{y_i - \bar{y}}{\sigma_y} \right),$$

jossa $\bar{x}$ tarkoittaa sarjan x keskiarvoa ja σ_x näiden keskihajontaa.

5.2. Funktionaalinen genomiikka ja DNA-mikrosirut

Funktionaalinen genomiikka on molekyylibiologian osa-alue, joka käsittelee lähes kaikkea DNA-sekvensoinnin jälkeistä genomiikan tutkimusta. Esimerkiksi geenien ilmentymistä eli kuinka geeni koodataan proteiiniksi sekä proteomiikkaa; proteiinien ilmentymisen, rakenteen ja funktion tutkimista.

DNA-mikrosirutekniikasta on useita variaatioita, mutta perusperiaate on se että mikroskooppilasille asetetaan organismin kaikista geeneistä lyhyet koettimet, joihin sitoutuu tutkittavasta näytteestä emäspariutumisella vastaavat geenit eli mRNA-sekvenssit. Näin leimautuneet geenit voidaan lukea koneellisesti geenieksressiodataksi. DNA-mikrosirutekniikassa syntyvät datamäärät ovat tyypillisesti suuria ja tietoa käsitellää usein klusteroituna. [Pasanen et al, 2003]



Kuva 10: DNA-mikrosiru

5.3. Hierarkkinen klusterointi

Hierarkkinen klusterointi on yksinkertainen menetelmä sisällöltään homogeenisten klustereiden etsintään, se voidaan jakaa kokoaviin (agglomerative) ja jakaviin (divisive) menetelmiin. Kokoavassa menetelmässä n kappaletta klusteroitavana olevia alkioita ryhmitellään etäisyysmatriisiin laskettujen etäisyyksien perusteella siten että alkiot yhdistetään lyhimmän välin perusteella. Jakavassa menetelmässä edetään käänteisesti, alussa kaikki alkiot kuuluvat samaan klusteriin ja ne erotellaan omiin klustereihinsa. Kokoavat menetelmät ovat yksinkertaisempia toteuttaa sekä myös yleisimmin käytettyjä.

5.3.1. Etäisyysmatriisi

Etäisyysmatriisin muodostaminen on keskeinen osa klusterointia. Se miten etäisyys kahden alkion välillä määritellään on tapauskohtaista. Yleisimmin käytetty etäisyysmitta lienee euklidinen etäisyys, joka antaa kahden alkion välimatkan n-ulotteisessa koordinaatistossa. Kun hierarkkinen klusterointi etenee iteratiivisesti, voidaan etäisyysmatriisi laskea monella tavalla. Lähimmän naapurin (Single-linkage) menetelmässä kahden klusterin välinen etäisyys ajatellaan lyhimmäksi etäisyydeksi klusterien alkioiden välillä. Kauimmaisen naapurin (Complete-linkage) menetelmässä klusterien etäisyys on niiden kauimmaisten alkioiden välinen etäisyys. Naapurikeskiarvossa (Average-linkage) etäisyys määräytyy klusterin alkioparien etäisyyksien keskiarvosta. Muita menetelmiä ovat vielä mm. Wardin menetelmä sekä klusterin keskipisteiden etäisyyksiin perustuva menetelmä.

Seuraava esimerkki havainnollistaa hierarkkista klusterointia lähin naapuri –menetelmällä. Taulukossa 3 on määritelty etäisyysmatriisi D viidelle klusteroitavalle pisteelle {A,B,C,D,E}. Esimerkin alkiot ja etäisyydet ovat keksittyjä.

	A	B	C	D	E
A	0	3	2	10	9
B	3	0	5	7	6
C	2	5	0	8	7
D	10	7	8	0	4
E	9	6	7	4	0

Taulukko 3: Etäisyysmatriisi D = {A,B,C,D,E}

Taulukon 1 matriisissa pienin etäisyys on alkioiden A ja C välillä, nämä siis muodostavat ensimmäisen uuden klusterin. Tämän jälkeen muodostetaan uusi etäisyysmatriisi niin että syntynyt klusteri toimii yhtenä alkiona. Uudet etäisyydet lasketaan lähin naapuri – menetelmällä niin että klusterin ja alkion väliseksi etäisyydeksi tulee lyhin etäisyys alkion ja jonkin klusterissa olevan alkion välillä.

	A,C	B	D	E
A,C	0	3	8	7
B	3	0	7	6
D	8	7	0	4
E	7	6	4	0

Taulukko 4: Etäisyysmatriisi D = {(A,C),B,D,E}

Syntyneessä uudessa matriisissa lyhin etäisyys on nyt alkioiden (A,C) ja B välillä. Nämä yhdistetään uudeksi klusteriksi ja lasketaan uusi etäisyysmatriisi.

	A,B,C	D	E
A,B,C	0	7	6
D	7	0	4
E	6	4	0

Taulukko 5: Etäisyysmatriisi D = {(A,B,C),D,E}

Nyt lyhin etäisyys on alkioiden D ja E välillä, jotka muodostavat uuden klusterin. Tämän jälkeen kaikki alkiot yhdistyvät samaan klusteriin ja klusterointi on valmis. Klustereita voidaan tämän jälkeen tarkastella muodostuneesta dendrogrammista, jollainen on kuvassa 11.

5.4. K-Means

K:n keskiarvon menetelmällä (K-means) voidaan klusteroida aineisto k määrään klustereita haluttujen parametrien perusteella. Algoritmi jakaa ensin aineiston k klusteriin joko satunnaisesti tai mahdollisesti jollakin muulla aineiston ominaisuuksiin liittyvällä periaatteella. Tämän jälkeen algoritmi laskee klusterien keskipisteet ja muodostaa uudet klusterit liittäen jokaisen aineiston alkion lähimpään keskipisteeseen. Uusille klustereille lasketaan taas uudet keskipisteet ja kierroksia toistetaan kunnes alkiot eivät enää vaihda klustereita tai vaihtoehtoisesti klusterien keskipisteet eivät enää muutu.

K-means algoritmi on suhteellisen nopea mutta se ei välttämättä tuota optimaalista klusterointia, lopullinen tulos riippuu suuresti annettujen klusterien määrästä. K-means klusterointi voidaan kuitenkin toistaa nopeasti useita kertoja ja valita paras klusterointi. [Jain and Dubes, 1988]

5.5. Itseorganisoituva kartta

Itseorganisoituva kartta (Self Organizing Map, SOM) on Teuvo Kohosen kehittämä luokka erilaisia menetelmiä, jotka perustuvat neuroverkkoihin ja oppimiseen. Järjestelmän yksiköt kilpailevat yhteisestä informaatiosta ja erikoistuvat kuvaamaan erityyppistä tietoa. Samalla aineistosta ja siihen liittyvistä relaatioista muodostuu kartta. Itseorganisoituvalla kartalla voidaan helposti visualisoida moniulotteista dataa yleensä kaksiulotteideksi kartaksi. Geeniekspressiodatan visualisoinnissa itseorganisoituvia karttoja käytetään usein.

SOM-algoritmin perusperiaatteet ovat seuraavat:

1. Valitaan kartan topologia, tyypillisesti nelikulmainen tai heksagonaalinen.

2. Valitaan kartan solujen lukumäärä ja liitetään soluihin satunnainen painovektori, johon syötevektoreita vertaillaan.

3. Jokaisella suorituskerralla vertaillaan yhtä syötevektoria kaikkien solujen painovektoreihin.

4. Soluista valitaan se, jonka painovektori muistuttaa eniten syötevektoria. Samankaltaisuus voidaan määrittää esimerkiksi euklidisena etäisyytenä tai vektorien välisenä pistetulona.

5. Uudeksi solun painovektoriksi tulee nyt vanhan painovektorin ja syötevektorin keskiarvo.

6. Muokataan solun lähimpien naapureiden painovektoreita.

7. Iteroidaan eli suoritetaan algoritmia niin monta kertaa kuin tarvitaan riittävän tarkan lopputuloksen saamiseksi.

5.6. Klusterointimenetelmien vertailua

Itseorganisoituvat kartat ja k:n keskiarvon menetelmä sopivat hyvin suurten tietomäärien käsittelyyn, erityisesti jos haluttujen klusterien määrä voidaan päättää. Suurilla aineistoilla hierarkkisten menetelmien dendrogrammit voivat muodostua muodostuvat helposti liian suuriksi, mikä joissakin tapauksissa häiritsee klusterien visualisointia. Hierarkkiset menetelmät ovat myös herkkiä datan muutoksille. K:n keskiarvon menetelmä ja itseorganisoituvat kartat vaativat haluttujen klustereiden määrän etukäteen, mikä voi rajoittaa niiden soveltuvuutta aineiston analysointiin. Kaikilla menetelmillä on omat hyvät ja huonot puolensa erilaisissa tapauksissa.

Sekundaarirakenteiden luokittelemiseksi valitsin hierarkkisen klusteroinnin, koska aineisto oli suhteellisen suppea, etäisyysmatriisi oli helppo muodostaa ja klusterien määrällä ei ollut tässä tapauksessa väliä. DNA-mikrosiruaineistoille valitsin k:n keskiarvon menetelmän.

6. Kokeet

Human Genome Project tuotti raakaversion ihmisen genomista vuonna 2000 ja julkaisi lähes valmiin version vuonna 2003. Ihmisen genomia tutkitaan ja tarkennetaan edelleen. Pääkoordinaattori projektissa oli Yhdysvaltojen National Institutes of Health (NIH), jonka alainen National Center for Biotechnology Information (NCBI) ylläpitää sekä kehittää useita julkisia molekyylibiologian tietokantoja ja työkaluja. Euroopassa vastaava instituutio on European Molecular Biology Laboratory (EMBL), jonka alainen on myös European Bioinformatics Institute (EBI). Monet muutkin valtiolliset ja yksityiset tutkimuslaitokset ylläpitävät omia tietokantojaan. Eri tietokantojen ja tietojen yhdistäminen on eräs keskeinen ongelma bioinformatiikassa.

Toteutin aiemmin suuren MySQL-tietokannan, joka yhdistää muunmuassa NCBI, EMBL ja Uniprot tietokannat sekä muita pienempiä tietokantoja. Tietokannan voi luoda PERL-skripteillä, jotka ensin lataavat eri tietokannat julkisilta ftp-palvelimilta omissa formaateissaan, jonka jälkeen skriptit purkavat tiedot sopiviksi taulurakenteiksi ja luovat datasta tietokannan MySQL-palvelimelle. Tätä tietokantaa on hyödynnetty aikaisemmissa projekteissa [Sharabiani et al, 2005] ja myös tässä tutkielmassa.

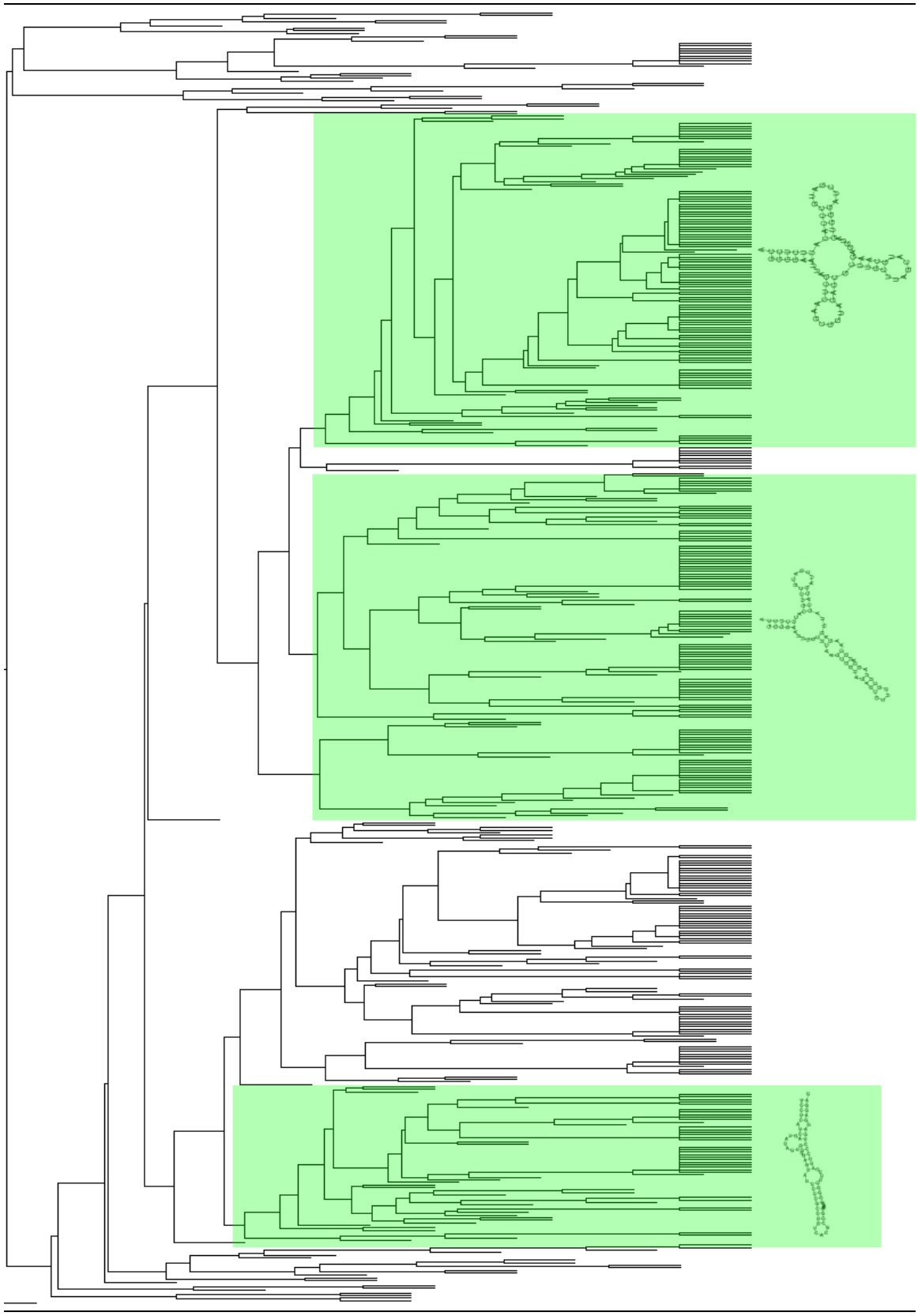
6.1. Menetelmätesti

Halusin ensiksi testata menetelmiä ja ohjelmistoja, sekä RNAfold-ennustuksen että RNAdistance-etäisyyksiin perustuvan klusteroinnin toimivuutta. Perusongelma oli varmistaa onko klusterointimenetelmillä mahdollista luokitella samankaltaisia rakenteita omiin klustereihinsa. Lähtöaineistoksi oli luonnollista valita joukko tRNA-sekvenssejä, koska ne ovat suunnilleen saman pituisia, mikä ei häiritse RNAdistance-ohjelmaa. Lisäksi siirtäjä-RNA:t tuottavat luonnossa suunnilleen samanlaisen rakenteen, joten tuloksista voi arvioida jotakin myös ennustusmenetelmän tarkkuudesta. Sopivat sekvenssit löytyivät Genomic tRNA tietokannasta (http://lowelab.ucsc.edu/GtRNAdb/), josta valitsin ihmisen genomista tRNAscan-SE -ohjelmalla [Lowe and Eddy, 1997] rinnastetut 514 kappaletta tRNA-sekvenssejä. Ennustin sekvensseille sekundaarirakenteet RNAfoldilla perusasetuksilla. Lisäksi laskin RNAdistancella etäisyysmatriisiin kaikkien etäisyydet toisiinsa. Valmiille etäisyysmatriisille oli luonnollisinta suorittaa hierarkkinen klusterointi. Käytin klusterointiin R-ohjelmaa, R on tilastolliseen laskentaan ja visualisointiin tarkoitettu vapaa ohjelmisto. Klusterien etäisyysmitaksi valitsin naapurikeskiarvon.

Klusteroinnin tuloksena syntyi kuvan 11 dendrogrammi, josta erottuu selvästi erilaisia klustereita. Todellisuutta vastaavia siirtäjä-RNA rakenteita tuli yllättävän pieni määrä, noin. 25%. Ennustuksen tarkkuus tässä tapauksessa parantuisi selvästi käyttämällä esimerkiksi RNAalifold –ohjelmaa konsensus-sekundaarirakenteiden ennustamiseksi, koska aineiston luonne ja odotettavissa oleva tulos, samankaltainen siirtäjä-RNA rakenne, on tiedossa.

Valitsin kuvan 11 dendrogrammista kolme selkeästi erottuvaa suurempaa klusteria joiden rakenteita tarkastelin tarkemmin. Ylimmässä värjätyssä klusterissa rakenteet ovat siirtäjä-RNA:lle tyypillisessä kolmen varren ja silmukan muodostamassa apilanlehtimuodossa. Keskimmäisessä klusterissa silmukallisia varsia esiintyy vain kaksi, ennustus ei ole osannut valita muodoltaan oikeampaa rakennetta. Alimmaisen klusterin muodostavat lähinnä yhdestä varresta koostuvat rakenteet.

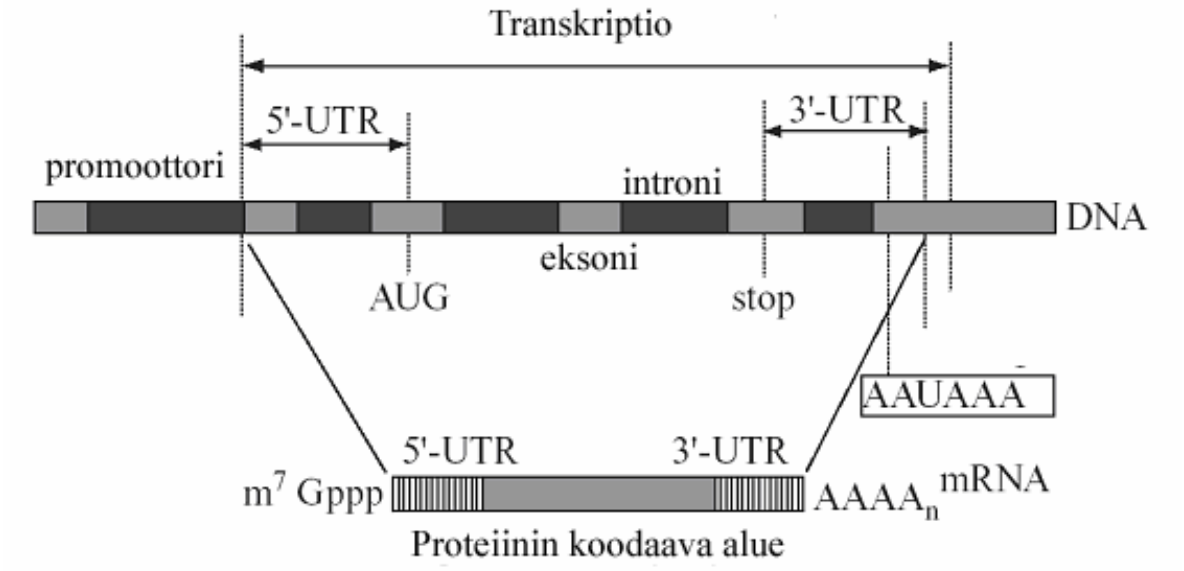
Klusterien sisäisistä sekvensseistä RNAalifold–ohjelmalla ennustetut konsensus-rakenteet olivat yhtäläisiä kuvan 11 esimerkkien kanssa. Satunnaisilta keskimmäisen klusterin sekvensseiltä löytyi Mfold-ohjelman ennustamista alioptimaalisista rakenteista myös oikeita siirtäjä-RNA rakenteita. Testi kuvasi hyvin rakenteiden ennustamisen ongelmaa. Kun mahdollisia ratkaisuja on erittäin paljon lähellä toisiaan, mitkä ovat lopulta ne säännöt, joiden perusteella rakenteet määräytyvät sellaisiksi kun ne oikeasti esiintyvät.



Kuva 11: dendrogrammi tRNA-sekundaarirakenteista

6.2. Koeaineisto

Tutkimusaineistoksi valittiin ihmisen geenien 5'UTR (5 prime Untranslated Region) alueet, koska ne liittyvät geenien ilmentymiseen ja niillä on selkeä yhteys DNA-mikrosiruaineistoihin. Lisäksi geenien 5'UTR-alueet ovat yleensä suhteellisen lyhyitä ja niitä on tutkittu muissakin artikkeleissa. Lähteenä toimi NCBIn tuottama ihmisen genomi. UTR-alue on tietty osa mRNA-sekvenssin alussa alkaen transkriptiokohdasta ja päättyen translaatiokohtaan. Geenin eri alueet on esitelty kuvassa 12.



Kuva 12: UTR-alueet

Geenien UTR-alueet säätelevät geenien ilmentymistä, usein niin että UTR-alueet toimivat sitoutumiskohtina proteiineille, jotka vaikuttavat mRNA-sekvenssin translaatiossa [Vlden and Thomas, 1999]. Yleinen esimerkki on Iron Response Element (IRE), n. 30 nukleotidin mittainen UTR-elementti, joka muodostaa silmukan muotoisen sekundaarirakenteen, johon Iron Regulation Protein (IRP) sitoutuu [Casey et al, 1988]. UTR-alueet ovat myös melko lyhyitä ja siksi nopeasti ennustettavia suurissakin määrissä.

Olin aiemmin toteuttanut Perl-skriptin, joka hakee haluttuja geenisekvenssejä NCBIn raakagenomista eli .gbk–tiedostoista. Gbk-tiedostojen formaatista on tarkempi kuvaus Heikki Hyyrön pro gradu –tutkielmassa [Hyyrö, 2000]. Tarkistaessani uudelleen skriptin antamien UTR-sekvenssien oikeellisuutta, löysin internetistä palvelun, jolla voi helposti noutaa ihmisen geenien 5' UTR-sekvenssit fasta-formaatissa. University of California, Santa Cruz (UCSC) ylläpitämä Genome Browser on laaja tietokanta ja työkalu

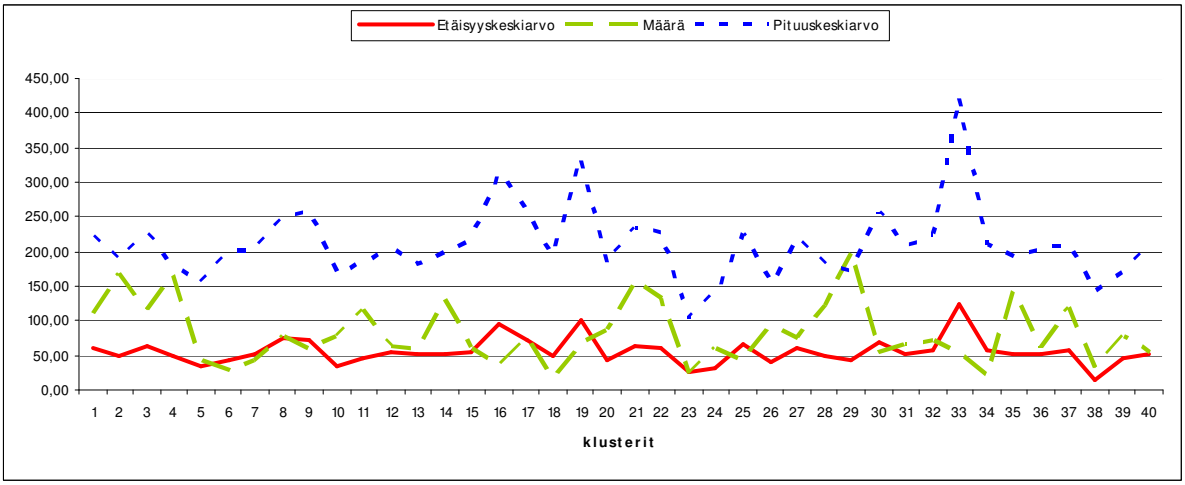
sekvenssidatan käsittelyyn. Genome Browserin data on NCBIn tuottamaa [Karolchik et al, 2003]. Käsin suorittamillani satunnaisilla tarkastuksilla Genome Browserin antamat UTR-sekvenssit täsmäsivät NCBIn antamiin sekvensseihin, joten päädyin käyttämään tätä dataa oman skriptini sijaan. Genome Browser sijaitsee kirjoittamishetkellä osoitteessa http://hgwdev.cse.ucsc.edu/ .

6.3. Toteutus

Hyödynsin koeaineistona valmista B-solu mikrosiruaineistoa [Ollila and Vihinen, 2002]. B-solut ovat lymfosyyttejä, valkoisia verisoluja, joilla on merkittävä rooli ihmisen immuunijärjestelmässä. Aineistossa oli aluksi 4359 geeniä, joista karsittiin pois saman geenin useat eri ilmentymät, sekä sellaiset, joille ei löytynyt sekvenssiä NCBI:n sekvenssitietokannasta. Karsinnasta saaduista 2053 geenistä poistettiin vielä sellaiset, joiden 5'UTR-alueet olivat lyhyempiä kuin 20 nukleotidia tai pidempiä kuin 500 nukleotidia. Erittäin lyhyiden tai pitkien sekvenssien rakenteiden ennustaminen ei ole tutkimuksen kannalta oleellista, lisäksi ylipitkien sekvenssien poistaminen lyhensi merkittävästi rakenteiden ennustamiseen kuluvaa aikaa. Tutkittavaksi aineistoksi jäi 1633 geeniä.

Klusteroin aineiston neljäänkymmeneen klusteriin k:n keskiarvon menetelmällä, klusterien määrä valittiin tarkoituksella samaksi kuin saman aineiston aikaisemmassa tutkimuksessa [Ollila and Vihinen, 2002]. Klusterointiin käytin Open Source Clustering Softwarea [Hoon et al, 2004]. Etäisyysmitaksi valitsin euklidisen etäisyyden, iteraatioiden määrä oli 100. Klusteroin aineiston myös 200 ja 500 iteraatiolla, suurempi iteraatiokertojen määrä ei enää vaikuttanut tulokseen. Aineisto jakaantui suhteellisen tasaisesti selkeisiin klustereihin.

Ennustin aineistolle sekundaarirakenteet RNAfold –ohjelmalla perusasetuksilla. Laskin aluksi kaikkien ennustettujen rakenteiden etäisyydet toisiinsa RNAdistance-ohjelmalla käyttäen karkeasti kuvaavaa (Coarse Grained) notaatiota ja puun editointietäisyyttä, tarkoituksena saada rakenteiden välille mahdollisimman karkea ja suuripiirteinen täsmäys. Kuvasta 13 näkee selvän korrelaation klusterin sisäisten etäisyyksien keskiarvon sekä sekvenssien pituuskeskiarvon välillä. Tulos oli odotusten mukainen, sillä pidemmillä sekvensseillä etäisyyserotkin muodostuvat luonnollisesti suuremmiksi, jopa karkeimmalla notaatiolla.



Kuva 13: Klusterien keskiarvot

Klustereissa erityisen rikastuneiden samankaltaisten rakenteiden löytämiseksi laskin aineistolle hypergeometrisen jakauman, eli määrätyn osajoukon esiintymisten jakauman. Hypergeometrisen jakauman pistetodennäköisyysfunktio on

$$\mathrm{P}\{\mathrm{X} = i\} = \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}}$$

missä N on perusjoukon alkioiden lukumäärä, M määrätyn osajoukon alkioiden lukumäärä, sellaisten joilla on jokin haluttu ominaisuus A, ja n on otosten lukumäärä. Kun X on niiden alkioiden lukumäärä otoksessa, joilla on ominaisuus A, niin X~Hyperg(N,M,n).

Laskin odotusarvot jokaiselle rakenteelle siten, että N oli kaikkien rakenteiden lukumäärä, M toisiaan lähellä olevien rakenteiden lukumäärä, n kaikkien rakenteiden määrä klusterissa ja i läheisten rakenteiden määrä klusterissa. Koska rakenteiden etäisyyksien keskiarvo oli karkeasti kuvaavalla notaatiolla 56, valitsin aluksi toisiaan lähellä olevien rakenteiden etäisyydeksi korkeintaan 5. Koska tuolla rajalla tuli yllättävän paljon merkittäviä ($P < 0.005$) tuloksia, laskin lopulta läheisen rakenteen rajaksi vain yhden merkin etäisyyden. Karkealla notaatiolla yksi merkki voi tarkoittaa jopa yhden silmukan puuttumista. Tulosten tarkastelu osoitti klustereissa rikastuneet rakenteet lähes poikkeuksetta kaikkein lyhyimpien sekvenssien sekundaarirakenteiksi. Syy tähän on se, että

käytetyillä menetelmillä lyhyiden sekvenssien väliset sekundaarirakenteiden etäisyydet muodostuvat aina pienemmiksi kuin pidemmillä sekvensseillä.

Laskin koeaineistolle uuden etäisyysmatriisin käyttämällä etäisyyden määritelmänä RNAdistance-ohjelman tarkinta mittaa, suoraa merkkijonotäsmäystä. Lisäksi laskin toisen etäisyysmatriisin käyttämällä painotettua karkeasti kuvaavaa notaatiota. Käyttämällä tarkinta notaatiota merkittäviä samankaltaisten rakenteiden rikastumia ei löytynyt muuten kuin nostamalla samankaltaisten rakenteiden etäisyyksien raja-arvoa. Tällä tavalla löytyneet samankaltaiset rakenteet olivat kuitenkin edelleen aineiston lyhyimpien sekvenssien rakenteita, joiden etäisyys ei edes teoriassa voi muodostua koko aineistoon nähden erityisen suureksi.

Painotetulla karkeasti kuvavalla notaatiolla määritetty etäisyysmatriisi tuntui parhaalta valinnalta käytettävälle aineistolle. Tälläkään etäisyysmitalla ei löytynyt merkittäviä samankaltaisten rakenteiden rikastumia ilman, että samankaltaisten rakenteiden etäisyysväli kasvoi liian suureksi. Mahdollisesti pienempi klusterien määrä voisi paljastaa tuloksia paremmin. Sekundaarirakenne sinällään sisältää kuitenkin suhteellisen vähän informaatiota. Esimerkiksi käyttämällä karkeasti kuvaavaa notaatiota hyvin lyhyet sekvenssit voivat täsmätä täydellisesti monta kertaa lähes kaikkiin pidempien sekvenssien sekundaarirakenteisiin. Sopivan etäisyysmitan ja analysointimenetelmän löytäminen edellyttäisi myös tarkempaa tietoa biologisen aineiston ominaisuuksista sekä mahdollisesti haluttavista tuloksista.

7. Yhteenveto

Ribonukleiinihapon sekundaarirakenne voidaan teoriassa ennustaa suoraan RNAn nukleotidisekvenssin emäsjärjestysten perusteella. Nykyisissä menetelmissä emästen pariutumistodennäköisyydet on määritetty kokeellisesti, lisäksi kaikkia poimuttumiseen vaikuttavia tekijöitä ei välttämättä tunneta tai kyetä mallintamaan. Sekundaarirakenteiden ennustaminen on kuitenkin tiettyyn rajaan asti mahdollista tietyllä tarkkuudella.

Tutkielmassa käytiin ensiksi läpi keskeisiä menetelmiä RNA-sekundaarirakenteiden ennustamiseen. Erityisesti käsittelin rakenteen vapaan energian mininointiin perustuvia menetelmiä, joilla voidaan dynaamista ohjelmointia hyödyntäen laskea optimaaliset sekundaarirakenteet, joilla on pienin vapaa energia. Tähän menetelmään perustuvaa RNAfold –ohjelmaa hyödynnettiin myös kokeellisessa osuudessa.

Tutkielma tarkasteli myös erilaisia menetelmiä RNA-sekundaarirakenteiden vertailuun ja luokitteluun. RNA-sekundaarirakenne voidaan esittää merkkijonona sulkunotaatiolla tai puumuodossa, mikä mahdollistaa rakenteiden vertailun määrittelemällä niiden välinen editointietäisyys. Käsittelin keskeisiä dynaamiseen ohjelmointiin perustuvia algoritmeja sekä merkkijonojen että puiden välisten editointietäisyyksien laskemiseksi. Lisäksi tutkielmassa käsiteltiin kahta bioinformatiikassa keskeistä merkkijonotäsmäyksen algoritmia sekä joitakin sekundaarirakenteiden vertailuun kehitettyjä ohjelmistoja.

Viidennessä kappaleessa käsiteltiin jonkin verran kolmea keskeistä klusterointimenetelmää, hierarkkista klusterointia, k:n keskiarvon menetelmää sekä itseorganisoituvia karttoja. Ensimmäisessä menetelmätestissä hierarkkinen klusterointi osoittautui toimivaksi menetelmäksi luokitella sekundaarirakenteet samankaltaisten rakenteiden joukkoihin. Suurin ongelma sekundaarirakenteiden luokittelussa on ennustusmenetelmien luotettavuus sekä kahden erikokoisen sekundaarirakenteen etäisyyden määritteleminen. Kokeellisessa osuudessa tutkin lisäksi tässä tutkielmassa käsiteltyjä menetelmiä biologiselle aineistolle. Merkittäviä tuloksia ei löytynyt pääasiassa aineiston suppean koon ja menetelmien epätarkkuuden takia. Menetelmät kuitenkin osoittautuivat toimiviksi.

Lähdeluettelo

[Akutsu, 2000] T. Akutsu, Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics* **104**, 2000, 45-62.

[Altschul *et al*, 1990,] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local Align search tool. Journal of Molecular Biology, **215**, 1990, 403-410.

[Apostolico and Gali, 1997] A. Apostolico, Z. Gali. Pattern Matching Algorithms – Approximate Tree Pattern Matching, Oxford University Press, 1997.

[Casey *et al*, 1988] J. L. Casey et al. Iron-responsive elements: regulatory RNA sequences that control mRNA levels and translation. Science, **240**, 1988, 924-928.

[Chomsky, 1956] N. Chomsky, Three models for the description of language, *IRE Transactions on Information Theory*, 1956.

[Demaine *et al*, 2006] E. D. Demaine, S. Mozes, B. Rossman, O. Weimann, An $O(n^3)$-Time Algorithm for Tree Edit Distance. Abstract, arXiv:cs.DS/0604037, 2006.

[Dennis, 2002] C. Dennis, The Brave new world of RNA. Nature, **418**, 2002, 122-124.

[Ding and Lawrence, 2003] Y. Ding, C. E. Lawrence. A Statistical sampling algorithm for RNA secondary structure prediction, *Nucleic Acids Research*, **31**, 2003, 7280-7301.

[Dowell and Eddy, 2004] R. D. Dowell, S. R. Eddy, Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction, *BMC Bioinformatics*, **5**, 2004, 17-17.

[Dulucq and Tichit, 2003] S. Dulucq, L. Tichit, Analysis of tree edit distance algorithms, Proceedings of 14th Annual Symposium of Combinatorial Pattern Matching, 2003, 25-27.

[Eddy and Durbin, 1994] S. R. Eddy, R. Durbin, RNA sequence analysis using covariance models. *Nucleic Acids Research*, **22**, 1994, 2079-2088.

[Eddy, 2002] S. R. Eddy, A memory-efficient dynamic programming algorithm for optimal Align of a sequence to an RNA secondary structure, *BMC Bioinformatics*, 2002, 18-18.

[Gardner and Giegerich, 2004] P. Gardner, R. Giegerich, A comprehensive comparison of comparative RNA structure prediction approaches, BMC Bioinformatics, **5**, 2004.

[Henikoff and Henikoff, 1992] S. Henikoff, J. G. Henikoff, Amino Acid Substitution Matrices from Protein Blocks. Proceedings of the National Academy of Sciences, **89**, 1992, 10915-10919.

[Hofacker *et al*., 1994] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, P. Schuster, Fast Folding and Comparison of RNA Secondary Structures. *Chemical Monthly* **125**, 1994, 167-188.

[Hofacker *et al*, 2002] I. L. Hofacker, M. Fekete, P. F. Stadler, Secondary structure prediction for aligned RNA sequences. *Journal of Molecular Biology*, **319**, 2002, 1059-1066.

[Hoon *et al*, 2004] M. J. L. de Hoon, S. Imoto, J. Nolan, S. Miyano,Open Source Clustering Software. Bioinformatics, **20**, 2004, 1453-1454.

[Hyyrö, 2000] H. Hyyrö, Merkkijonotäsmäyksestä. Pro Gradu –tutkielma, Tampereen Yliopisto, Tietojenkäsittelytieteiden laitos, 2000.

[Jain and Dubes, 1988] A. K. Jain, R. C. Dubes, Algorithms for clustering data, Prentice Hall, 1988.

[Karolchik *et al*, 2003] D. Karolchik et al. The UCSC Genome Browser Database, Nucleic Acids Research, **1**, 2003, 51-54.

[Klein, 1998] P. N. Klein, Computing the edit-distance between unrooted ordered trees. Proceeding of 6th European Symposium on Algorithms, 1998, 91-102.

[Knudsen and Hein, 2003] B. Knudsen, J. Hein, Pfold: RNA secondary structure prediction using stochastic context-free grammars, *Nucleic Acids Research*, **31**, 2003, 3423-3428.

[Lowe and Eddy, 1997] T. M. Lowe, S. R. Eddy, tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Research, **5**, 1997, 955-64

[Macke, 2001] T. J. Macke, RNAMotif, an RNA secondary structure definition and search algorithm. Nucleic Acids Research **22**, 2001, 4724-4735.

[Mathews, 2006] D. H. Mathews. Revolutions in RNA Secondary Structure Prediction. Journal of Molecular Biology, **359**, 2006, 526-532.

[Mathews *et al*, 1999] D. H. Mathews, J. Sabina, M. Zucker, D. H. Turner, Expanded sequence dependence of thermodynamic parameters provides improved prediction of RNA secondary structure. Journal of Molecular Biology, **288**, 1999, 911-940.

[Mauri and Pavesi, 2005] G. Mauri, G. Pavesi, Algorithms for pattern matching and discovery in RNA secondary structure. Theoretical Computer Science, **335**, 2005, 29-51.

[Mount, 2004] D. W. Mount, *Bioinformatics – Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2004.

[Needleman and Wunsch, 1970] S. B. Needleman, C. D. Wunsch, A general method applicable to the search for similiarities in the amino acid sequences of two proteins. Journal of Molecular Biology, **3**, 1970, 443-453.

[Nelson and Cox, 2000] D. L. Nelson, M. M. Cox, *Lehninger Principles of Biochemistry*. Worth Publishers, 2000.

[Nussinov and Jacobson, 1980] R. Nussinov, A. B. Jacobson, Fast algorithm for predicting the secondary structure of single-stranded RNA. Proceedings of the National Academy of Sciences, **77**, 1980, 6309-6313.

[Ollila and Vihinen, 2002] J. Ollila, M. Vihinen, Microarray analysis of B-cell stimulation. Vitamins and hormones, **64**, 2002, 77-99.

[Pasanen *et al*, 2003] T. Pasanen, J. Saarela, I. Saarikko, T. Toivanen, M. Tolvanen, M. Vihinen, G. Wong, DNA Microarray Data Analysis, CSC, 2003.

[Pavesi *et al*, 2004] G. Pavesi, G. Mauri, M. Stefani, G. Pesole, RNAProfile: an algorithm for finding conserved secondary structure motifs in unaligned RNA sequences. *Nucleic Acids Research,* **32**, 2004, 3258-3269.

[Rivas and Eddy, 1999] E. Rivas, S. R. Eddy, A dynamic programming algorithm for RNA structure prediction including pseudoknots. Journal of Molecular Biology, **285**, 1999, 2053-2068.

[ScienceWatch*, 2003*] Twenty Years of Citation Superstars, *ScienceWatch,* **5**, 2003.

[Shapiro and Zhang, 1988] B. Shapiro, K. Zhang, Comparing multiple RNA secondary structures using tree comparisons. Computational Applied Biosciences, **4**, 1988, 387-393.

[Sharabiani *et al*, 2005] M. T. A. Sharabiani, M. Siermala, T. O. Lehtinen, M. Vihinen, Dynamic covariation between gene expression and proteome characteristics. BMC Bioinformatics, **6**, 2005.

[Smith and Waterman, 1981] T. F. Smith, M. S. Waterman, Identification of Common Molecular Subsequences. Journal of Molecular Biology, **147**, 1981, 195-197.

[Tai, 1979] Kuo-Chung Tai, The Tree-to-Tree Correction Problem. Journal of the Association for Computing Machinery, **26**, 1979, 422-433.

[Thompson *et al*, 1994] J. D. Thompson, D. G. Higgins, T. J. Gibson, CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research,* **22**, 1994.

[Waterman, 1995] M. Waterman, Introduction to Computational Biology: Maps, sequences and genomes, Chapman and Hall, 1995.

[Watson and Crick, 1953] J. Watson, F. Crick, Molecular structure of nucleic acids: A Structure for Deoxyribose Nucleic Acid, *Nature*, 1953.

[Waterman and Smith, 1972] M. S. Waterman, T. F. Smith, RNA secondary structure: A complete mathematical analysis, *Math. Biosci*., **42**, 1972, 257-266.

[Vlden and Thomas, 1999] A. W. Velden, A. A. Thomas, The role of the 5' untranslated region of an mRNA in translation regulation during development. The international journal of biochemistry & cell biology, **1**, 1999, 87-106.

[Zhang and Shasha, 1989] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing, **18**, 1989, 1245 – 1262.

[Zuker *et al*, 1999] M. Zuker, D. H. Mathews, D. H. Turner, Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide. *RNA Biochemistry and Biotechnology*. Kluwer Academic Publishers, 1999.

[Zucker, 2003] M. Zucker, Mfold web server for nucleic acid folding and hybridization prediction, Nucleic Acids Research, **31**, 2003, 3406-3415.

[Zucker and Stiegler, 1981] M. Zucker, P. Stiegler, Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Research, **9**, 1981, 133-148.