Harva k-meeri hajautustauluun perustuvien linjausalgoritmien tehostajana

Jouko Niinimäki

Kandidaatin tutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 21. joulukuuta 2015

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Jouko Niinimäki

Työn nimi — Arbetets titel — Title

Harva k-meeri hajautustauluun perustuvien linjausalgoritmien tehostajana

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatin tutkielma	21. joulukuuta 2015	23

Tiivistelmä — Referat — Abstract

DNA-sekvenssien vertailu lokaalisti linjaamalla on tärkeä bioinformatiikan osa-alue. Tässä tutkielmassa käsitellään lokaalien linjausten luomista tarkoin, perinteisin metodein sekä heuristisella menetelmällä, joka perustuu k:n mittaisten osamerkkijonojen, k-meerien, etsimiseen verrattavien sekvenssien väliltä. Erityisesti keskitytään menetelmässä hyödynnettävän harvan k-meerin käyttöön ja sen tuomiin suorituskykyparannuksiin, jotka osoittavat selkeästi harvan k-meerin paremmuuden tavalliseen k-meeriin nähden. Tutkielman lopussa käsitellään lyhyesti optimaalisten mallien, joihin harva k-meeri perustuu, ja malliyhdistelmien laskemiseen liittyviä ongelmia ja heuristisia ratkaisuja.

ACM Computing Classification System (CCS):
Theory of computation → Design and analysis of algorithms → Data structures design and analysis → Pattern matching
Applied computing → Life and medical sciences → Computational biology → Molecular sequence analysis

Avainsanat — Nyckelord — Keywords
harva k-meeri, bioinformatiikka, lokaali linjaus, merkkijonojen vertailu

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto

DNA eli deoksiribonukleiinihappo on ketjumuotoinen molekyyli, joka sisältää tiedon siitä, miten eliö rakentuu [RUC+11, sivut 132-135]. DNA-molekyylin koodaavana tekijänä toimivat neljä emästä: tymiini, adeniini, sytosiini ja guaniini. Emäkset ovat asettuneena peräkkäisinä pareina, *emäspareina*, muodostaen kaksoiskierteisen ketjun. Koska kaksoisrakenteessa tymiini toimii aina adeniinin ja sytosiini guaniinin vastinparina, voidaan ketjua kuvata pelkästään ketjun toisen puolen emäksillä. Emäkset voidaan lyhentää kirjaimin T, A, C ja G. DNA-molekyylin tietosisältöä kuvataankin usein näiden neljän merkin muodostamana merkkijonona, *DNA-sekvenssinä*.

Eliön DNA-sekvenssistä voidaan eritellä eri mittaisia alueita, *geenejä*, jotka sisältävät jonkin selkeän kokonaisuuden, kuten proteiinin rakenteen [RUC+11, sivut 275, 478]. Eliön koko DNA:ta kutsutaan sen *genomiksi*. Genomin koko voi vaihdella eräiden bakteerien miljoonista emäspareista aina joidenkin kasvien yli sataan miljardiin emäspariin.

Mutaatiot, eli muutokset DNA-sekvenssissä, aiheuttavat eroja eri lajien ja yksilöiden välille [Xio06, sivut 31-34]. Evolutiivisesti kaukana toisistaan olevilla lajeilla on usein suuremman mutaatiomäärän takia suurempia eroja DNA:ssa kuin läheistä sukua olevilla lajeilla. Eroja lajien tai yksilöiden välillä voidaan tutkia vertailemalla geenien DNA-sekvenssejä keskenään ja näin päätellä mutaatioiden ja yhteneväisyyksien avulla lajien tai yksilöiden sukulaisuussuhteita. Toisaalta sekvenssejä vertaamalla voidaan myös tutkia ennestään tuntematonta geeniä: jos tutkittavan geenin ja toiminnaltaan tunnetun geenin väliltä löytyy yhteneväisyyksiä, voidaan tästä mahdollisesti päätellä jotain geenin toiminnasta.

Sekvenssien vertailua tarvitaan myös eliön genomia koottaessa *sekvensoinnista* saadusta datasta [PPDS04; Xio06, sivut 243-250]. DNA:n sekvensoinnilla tarkoitetaan DNA-molekyylin ”lukemista” sekvenssimuotoon. Nykyisillä sekvensointimenetelmillä koko DNA:ta ei saada luettuna yhtenä kokonaisuutena, vaan tuloksena on mahdollisesti useita miljardeja lyhyitä, enintään 150 emäksen mittaisia[1], osittain päällekkäin meneviä sekvenssin osia eli *frag-*

[1]Luvut tuotettujen fragmenttien määristä ja koista haettu 21.12.2015 osoitteesta https://www.illumina.com/

mentteja. Genomi voidaan koota joko vertailemalla näitä sekvenssipätkiä jo tunnettuun kokonaiseen saman lajin yksilön genomisekvenssiin tai, uuden genomin tapauksessa, päättelemällä rakenne esimerkiksi fragmenttien päällekkäisyyksien avulla.

Sekvenssien vertailu tapahtuu usein *linjaamalla* niitä joko lokaalisti tai globaalisti [Xio06, sivut 34-35]. *Lokaalilla linjauksella* tarkoitetaan sekvensseistä löytyvien läheisesti toisiaan muistuttavien paikallisten alueiden, eli *homologisten alueiden*, asettamista rinnakkain. *Globaalilla linjauksella* puolestaan tarkoitetaan sekvenssien asettamista rinnakkain kokonaisuudessaan. Globaali linjaus soveltuu lähinnä melko samanlaisten ja saman mittaisten sekvenssien vertailuun, kun lokaaleja linjauksia käyttämällä voidaan etsiä yhteneväisyyksiä evolutiivisesti hyvin kaukanakin toisistaan olevista sekvensseistä.

Linjausten etsimiseen on kehitetty useita erilaisia menetelmiä [AGM$^+$90; LH10; Xio06, sivut 35-41]. Perinteiset, jokaisen mahdollisen emäksen vertailuun pohjautuvat algoritmit tuottavat tarkkoja tuloksia, mutta ovat epäkäytännöllisen hitaita, kun linjauksia etsitään miljardien emäksien mittaisista genomeista tai kun sekvenssille haetaan parasta linjausta suurten ja alati kasvavien geenitietokantojen sekvensseistä [AGM$^+$90; JP04, sivut 35-41; Xio06, sivut 35-41]. Ratkaisuksi tähän on kehitetty erilaisia lokaaleja linjauksia etsiviä heuristisia algoritmeja, joista monet perustuvat lyhyiden, ennalta asetetun k:n mittaisten osamerkkijonojen, eli *k-meerien*, etsimiseen sekvenssien väliltä hajautustauluja hyödyntäen[AGM$^+$90, LH10]. Menetelmän heikkoutena on, että siltä jää osa linjauksista löytämättä, koska k-meerien tulee täsmätä sekvenssien välillä täydellisesti.

Ongelmaa korjaamaan on kehitetty *harva k-meeri*, joka sallii virheet etsittävien k-meerien välillä ennalta määrätyissä kohdissa. Sopivaa harvaa k-meeriä käyttämällä algoritmin herkkyyttä löytää linjauksia saadaan nostettua ilman että sen nopeus kärsii [MTL02]. Useampaa erilaista harvaa k-meeriä käyttämällä herkkyyttä voidaan nostaa entisestään [LMKT04]. Harvoja k-meerejä on kuitenkin lukuisia erilaisia, eivätkä kaikki k-meerit tai niiden yhdistelmät toimi yhtä tehokkaasti [CZ04, BBV04].

Tässä tutkielmassa käydään ensin (luku 2) läpi lokaalin linjauksen muodostusta eri metodein. Luvussa 2.2 valaistaan hajautustaulua hyödyntävää

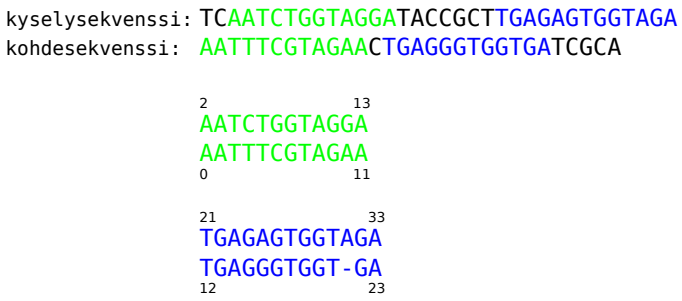
heuristista menetelmää. Harvoja k-meerejä, niiden tuomia hyötyjä sekä optimoinnin vaikeutta käsitellään luvussa 3.

2 Lokaalit linjaukset sekä menetelmiä niiden löytämiseen

Kun geenien väliltä halutaan etsiä mahdollisia yhteneväisyyksiä tai sekvensointidatan fragmenteista halutaan koota kokonainen genomi vertaamalla niitä jo tunnettuun genomiin, voidaan apuna käyttää lokaaleja linjauksia etsiviä algoritmeja [Xio06, sivut 34-35]. Lokaalilla linjauksella tarkoitetaan sekvensseistä löytyvien homologisten alueiden asettamista allekkain siten, että niiden merkit vastaavat mahdollisimman hyvin toisiaan. Linjausta voidaan kuvata seuraavasti: Olkoot linjauksessa esiintyvät alueet (sekvenssien osamerkkijonot) S_1 ja S_2 ja niiden pituudet l_1 ja l_2. Linjaus on $2 \times l$-matriisi, jossa ensimmäisen rivi koostuu merkkijonon S_1 merkeistä ja toinen rivi merkkijonon S_2 merkeistä siten, että merkit ovat samassa järjestyksessä kuin merkkijonoissa itsessään [JP04, sivut 167-169]. Merkkien välissä voi olla tyhjiä soluja, joita merkitään usein merkillä '-', mutta samassa sarakkeessa ainoastaan jommallakummalla rivillä. Lisäksi tulee päteä $l < (l_1 + l_2)$

Matriisin sarakkeita, joissa merkit vastaavat toisiaan, kutsutaan *täsmäyksiksi* (engl. match) ja sarakkeita, joissa merkit eroavat, *epätäsmäyksiksi* (engl. mismatch). Tyhjiä (viivalla merkittyjä) sarakkeita kutsutaan *aukoiksi*. Jos oletetaan, että linjauksen alueet todella olevat evolutiivisesti sukua toisilleen, voivat epätäsmäykset olla seurausta *pistemutaatioista*, jotka tarkoittavat emäksen muuttumista toiseksi [JP04, sivut 167-169; RUC+11, sivut 344-345]. Aukot puolestaan johtuvat *deleetioista* tai *insertioista*, joilla tarkoitetaan emäksen katoamista tai ilmestymistä sekvenssiin. Linjauksessa verrattavia sekvenssejä kutsutaan *kyselysekvenssiksi* (mitä verrataan; esimerkiksi fragmentti tai tuntematon geeni) ja *kohdesekvenssiksi* (mihin verrataan; esimerkiksi genomi tai jo tunnettu geeni). Lokaaleja linjauksia on havainnollistettu kuvassa 2.1.

Linjauksia voidaan etsiä useilla erilaisilla tarkoilla tai heuristisia menetelmiä hyödyntävillä algoritmeilla [AGM+90; LH10; Xio06, sivut 35-62]. Tässä



Kuva 2.1: Esimerkki kahdesta DNA-sekvenssistä sekä niistä löytyneistä lokaaleista linjauksista. Linjausten aloitus- ja päättymisindeksit näkyvät linjausten ylä- ja alapuolilla.

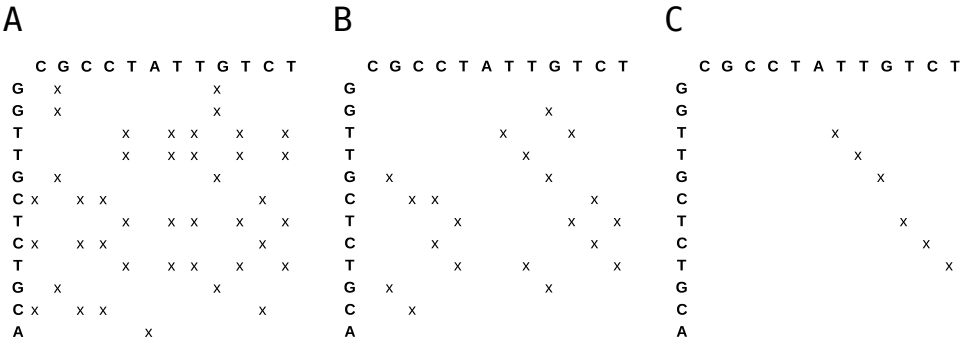
luvussa käsitellään lyhyesti lokaalien linjauksien muodostamista kahdella perinteisellä menetelmällä, pistematriisilla ja dynaamisella ohjelmoinnilla, sekä nopeammalla heuristisella, hajautustaulun käyttöön perustuvalla menetelmällä.

2.1 Perinteiset linjausmenetelmät

Perinteiset linjausmenetelmät perustuvat jokaisen emäksen vertailuun kysely- ja kohdesekvenssin välillä [SW81; Xio06, sivut 35-41]. Menetelmät tuottavat tarkkoja tuloksia, mutta niiden suuren aikavaativuuden ($O(l_1 \cdot l_2)$, missä l_1 ja l_2 ovat sekvenssien pituudet) johdosta ne eivät ole käytännöllisiä esimerkiksi suurissa geenitietokantahauissa tai käsiteltävien sekvenssien ollessa mahdollisesti miljardien emäksien mittaisia genomeja [AGM$^+$90; JP04, sivut 324-325; Xio06, sivu 52].

Pistematriisimenetelmässä (jota ei tule sekoittaa aminohapposekvenssien linjauksessa käytettävään *pisteytysmatriisiin*) luodaan matriisi, jonka rivit vastaavat kyselysekvenssin ja sarakkeet kohdesekvenssin emäksiä [Xio06, sivut 35-37, 41-46]. Kyseessä on siis $l_1 \times l_2$-matriisi. Matriisi käydään läpi solu kerrallaan, merkiten täytetyiksi ne solut, joissa sekvenssien emäkset täsmäävät. Formaalimmin: Olkoon S_1 ja S_2 kysely- ja kohdesekvenssit ja $S[i]$ sekvenssin i:s emäs ($i \in |S|$). Matriisista merkitään ne solut (i, j), joilla $S_1[i] = S_2[j]$. Muut solut jätetään tyhjiksi.

Linjaukset voidaan havaita matriisista diagonaalin suuntaisina ”viivoina” [Xio06, sivut 35-41]. Epätäsmäykset tuottavat viivoihin katkoksia ja insertiot ja deleetiot puolestaan ”hyppyjä” (jotka siis vastaavat aukkoja tuotetussa linjauksessa) diagonaalin vastaisessa suunnassa. Koska DNA-sekvenssi koostuu vain neljästä eri kirjaimesta, on jokaisella solulla keskimäärin yhden neljäsosan todennäköisyys tulla merkityksi. Tällöin matriisi täyttyy kohinasta ja siitä tulee hyvin vaikeasti luettava. Luettavuutta voidaan parantaa asettamalla solu merkityksi vain jos se on osa diagonaalin suuntaista viivaa, jonka pituus on vähintään k. Liian suurilla k:n arvoilla menetelmän tarkkuus kuitenkin kärsii ja osa oleellisesta tiedosta saattaa hävitä. Kuvassa 2.2 on havainnollistettu pistematriisimenetelmää kolmella eri k:n arvolla. Pistematriisimenetelmän heikkoutena (hitauden lisäksi) voidaan pitää sitä, ettei se suoraan tuota linjauksia, vaan ne on etsittävä matriisista yhdistelemällä sopivia diagonaalin suuntaisia viivoja. Menetelmä ei myöskään kerro suoraan mitään linjauksien tilastollisesta merkitsevyydestä.



Kuva 2.2: Esimerkki pistematriisimenetelmän käytöstä. Kohdassa **A** näytetään kaikkien emäksien yhtenevyydet, kohdassa **B** vähintään kahden ja kohdassa **C** kolmen emäksen mittaiset. Kuva pohjautuu kirjassa [JP04, sivu 326] esitettyyn esimerkkiin.

Dynaamista ohjelmointia hyödyntämällä voidaan linjauksille laskea pistemäärä ja täten löytää paras lokaali linjaus sekvenssien väliltä [SW81; Xio06, sivut 35-41]. Dynaamisella ohjelmoinnilla tarkoitetaan ongelman hajottamista pienempiin osaongelmiin ja ratkaisun kasaamista pala kerrallaan. Myös tämä menetelmä perustuu matriisin täyttämiseen.

Dynaamista ohjelmointia hyödyntävässä menetelmässä, jota kutsutaan myös Smith-Waterman-algoritmiksi, luodaan $(l_1+1)\times(l_2+1)$-matriisi, jonka ensimmäinen rivi ja sarake täytetään nollilla [SW81; Xio06, sivut 35-41]. Ennen matriisin täyttämistä asetetaan täsmäyksille (t), epätäsmäyksille (e) ja insertion tai deleetion tuottamille hypyille (d) pistearvot (esimerkiksi $t = 1$, $e = -1$, $d = -1$). Olkoon täytettävä matriisi M ja matriisin solun arvo M_{ij} riviltä i ja sarakkeesta j. Matriisin tyhjiä soluja lähdetään täyttämään siten, että

$$M_{ij} = \max \begin{Bmatrix} M_{(i-1)(j-1)} + \begin{cases} t, & \text{jos } S_1[i] = S_2[j] \\ e, & \text{jos } S_1[i] \neq S_2[j] \end{cases}, \\ M_{(i-1)j} + d, \\ M_{i(j-1)} + d, \\ 0 \end{Bmatrix}.$$

Siis solun arvo on maksimi neljästä vaihtoehdosta: (1) edellisen rivin ja edellisen sarakkeen solun arvo lisättynä täsmäyspisteillä, jos solun emäksen täsmäävät, tai epätäsmäyspisteillä, jos solun emäkset eivät täsmää; (2) saman sarakkeen edellisen rivin solun arvo lisättynä hyppysakolla; (3) saman rivin edellisen sarakkeen solun arvo lisättynä hyppysakolla; (4) nolla. Sekvenssien välinen paras lokaali linjaus löytyy etsimällä matriisista suurimman arvon saanut solu ja jäljittämällä reitti, jota kautta arvo soluun on saatu. Linjaus päättyy ensimmäiseen vastaan tulevaan nollaan. Muita mahdollisia linjauksia löytyy luonnollisesti muiden suuria arvoja saaneiden solujen reittejä jäljittämällä. Menetelmää on havainnollistettu kuvassa 2.3.

Kuten aikaisemmin mainittiin, sekä pistematriisimenetelmän että Smith-Waterman-algoritmin suuren aikavaativuuden $(O(l_1 \cdot l_2))$ takia ne eivät ole käytännöllisiä, kun linjattavat sekvenssit ovat kooltaan isoja tai linjuksia etsitään kokonaisista geenitietokannoista. Tähän soveltuvat paremmin nopeat, hajautustaulun käyttöön perustuvat linjausalgoritmit, joiden toimintaperiaatetta käsitellään seuraavassa aliluvussa [AGM+90].

		C	G	C	C	T	A	T	T	G	T	C	T
	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	2	1	0	0	0	0	0	2	1	0	0
G	0	0	2	1	0	0	0	0	0	2	1	0	0
T	0	0	1	1	0	2	1	2	2	1	4	3	2
T	0	0	0	0	0	2	1	3	4	3	3	3	5
G	0	0	2	1	0	1	1	2	3	6	5	4	4
C	0	2	1	4	3	2	1	1	2	5	5	7	6
T	0	1	1	3	3	5	4	3	3	4	7	6	9
C	0	2	1	3	5	4	4	3	2	3	6	9	8
T	0	1	1	2	4	7	6	6	5	4	5	8	11
G	0	0	3	2	3	6	6	5	5	7	6	7	10
C	0	2	2	5	4	5	5	5	4	6	6	8	9
A	0	1	1	4	4	4	7	6	5	5	5	7	8

Kuva 2.3: Dynaamista ohjelmointia käyttävällä Smith-Waterman-algoritmilla laskettu lokaali linjaus. Kyseessä on sama linjaus, joka näkyy kuvassa 2.2 **C**. Kuva pohjautuu kirjassa [Xio06, sivu 39] annettuun esimerkkiin.

2.2 Hajautustauluun perustuvat heuristiset linjausalgoritmit

Hajautustauluihin perustuvien linjausalgoritmien idea on hyvin yksinkertainen: kysely- ja kohdesekvenssin väliltä etsitään lyhyitä yhteisiä, ennalta määritellyn k:n pituisia osamerkkijonoja, eli *k-meerejä* [AGM+90]. Löydetyt yhteiset k-meerit, *osumat*, laajennetaan lokaaleiksi linjauksiksi. Algoritmin toiminta voidaan jakaa kolmeen vaiheeseen: (1) k-meerien luontiin, (2) osumien etsimiseen ja (3) osumien laajennukseen.

Ensimmäisessä vaiheessa (1) jokainen kyselysekvenssissä esiintyvä k-meeri tallennetaan hajautustauluun avaimena, jonka arvona on sen sijainti (indeksi) [AGM+90]. Tämän jälkeen (2) kohdesekvenssi käydään läpi alusta loppuun k-meeri k-meeriltä. Kutakin k-meeriä ja sitä vastaavaa arvoa etsitään hajautustaulusta. Jos k-meeri löytyy, tiedetään, että sama k-meeri löytyy kyselysekvenssistä arvon osoittamasta kohdasta. Tämä tieto tallennetaan osumana. Osumia, siis alustavia k:n mittaisia linjauksia, lähdetään (3) laajentamaan erikseen kumpaankin suuntaan merkkipari kerrallaan. Linjaukselle lasketaan samalla pistemäärää, hieman kuten Smith-Waterman-algoritmissa, antamalla täsmäyksille positiivinen ja epätäsmäyksille negatiivinen arvo. Lisäksi pidetään yllä tietoa pistemäärästä, jonka kyseinen linjaus on laajennuksen

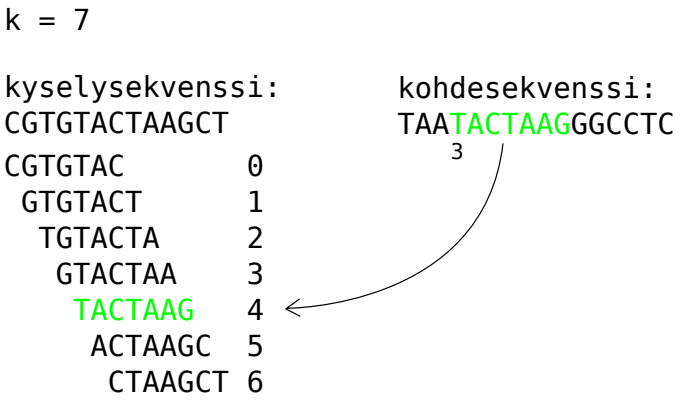
aikana maksimissaan saanut. Laajennus lopetetaan, kun linjauksen pistemäärä laskee ennalta asetetun kynnyksen verran tämän maksimipistemäärän alapuolelle. Lopullinen linjaus hyväksytään vain, jos se on tilastollisesti merkitsevä. Hajautustaulun käyttöä algoritmissa on havainnollistettu kuvassa 2.4.

Algoritmi ei tällaisenaan hyväksy tuotetuissa linjauksissa aukkoja, mutta useita algoritmin laajennuksia aukollisten linjausten löytämiseksi on kehitetty [MTL02, AMS+97, CWC04]. Nämä menetelmät kuitenkin sivuutetaan, koska ne eivät oleellisesti liity tutkielman aiheeseen.

Algoritmin aikavaativuus on $O(W+S+E)$, missä W on kyselysekvenssin k-meerien hajautustauluun tallentamiseen kuluva aika, S kohdesekvenssin läpikäymiseen ja osumien etsimiseen kuluva aika ja E osumien pidennykseen kuluva aika [AGM+90]. Koska hajautustaulu toimii asymptoottisessa vakioajassa, on W lineaarinen suhteessa kyselysekvenssin pituuteen ja S lineaarinen suhteessa kohdesekvenssin pituuteen. Osumien pidennyksen vaatima aika E riippuu tuotettujen osumien määrästä. Osumien odotettu määrä puolestaan riippuu sekvenssien pituuksien lisäksi niiden *samankaltaisuudesta* $p \in [0,1]$ (engl. similarity) sekä k:n arvosta [AGM+90, MTL02]. Samankaltaisuudella tarkoitetaan sekvenssien täsmäävien merkkien suhteellista osuutta. Pienentämällä arvoa k yhdellä, kasvaa odotettujen osumien määrä noin nelinkertaiseksi, koska tällöin täsmättävänä on yksi merkki vähemmän ja satunnaisten merkkien täsmäystodennäköisyys on $0,25$.[2]

DNA-sekvenssien linjauksessa hyväksi havaitulla oletusarvolla $k = 11$ hajautustauluun perustuvalla algoritmilla saattaa jäädä osa lokaaleista linjauksista löytämättä, koska kaikki pistemäärältä vahvatkaan linjaukset eivät välttämättä sisällä ainuttakaan yhtenäistä 11-meeriä, joka löytyisi molemmista sekvensseistä, eikä linjauksen alueelta löydy siksi yhtään osumaa [AGM+90]. k:n arvoa pienentämällä saadaan osumatodennäköisyyttä parannettua, mutta samalla myös sellaisten osumien, joiden laajennus ei tuota hyväksyttävää linjausta, määrä kasvaa. Näiden turhien osumien takia algoritmin kolmas vaihe, osumien laajennus, toistuu useampaan kertaan hidastaen algoritmin toimintaa.

[2]Koska DNA-sekvenssi sisältää neljää eri kirjainta, on kahden sekvenssin satunnaisissa indekseissä i_1 ja i_2 olevien merkkien täsmäystodennäköisyys $0,25$.



Kuva 2.4: Esimerkki osumien etsimisestä hajautustauluun perustuvissa algoritmeissa. Kyselysekvenssin jokainen 7-meeri tallennetaan hajautustauluun avaimen ja sen indeksi arvona. Jokaista kohdesekvenssin 7-meeriä haetaan tämän jälkeen hajautustaulusta. Löydetyt osumat (kuvassa kyselysekvenssin indeksissä 4 ja kohdesekvenssin indeksissä 3) tallennetaan muistiin myöhempää pidennystä varten.

3 Harva *k*-meeri herkkyyden parantajana

k-meerin pituuden pienentämisestä seuraavaa suoritusaikaongelmaa silmällä pitäen on kehitetty niin kutsuttu *harva k-meeri* (engl. spaced seed tai gapped seed). Harvan k-meerin idea on sallia epätäsmäyksiä k-meerissä itsessään ennalta valitun *mallin* osoittamissa kohdissa [MTL02]. Harvaa k-meeriä käyttämällä algoritmin *herkkyyttä* (engl. sensitivity) on mahdollista nostaa rajoitetusti hidastamatta kuitenkaan algoritmin toimintaa. Herkkyydellä tarkoitetaan (harvan) k-meerin todennäköisyyttä saada vähintään yksi osuma homologiselle alueelle pituudeltaan L, kun sekvenssien samankaltaisuus alueella on p. Käytännössä herkkyys kuvaa algoritmin kykyä löytää sekvenssiparissa esiintyviä linjauksia: mitä suurempi arvo, sitä paremmin linjauksia löytyy.

Tämän luvun ensimmäisessä aliluvussa käsitellään harvojen k-meerien toiminnallista ideaa sekä käyttöä. Toinen aliluku keskittyy harvojen k-meerien tuomiin suorituskykyparannuksiin ja mallin herkkyyteen vaikuttaviin tekijöihin. Kolmannessa aliluvussa käsitellään useamman mallin yhtäaikaista

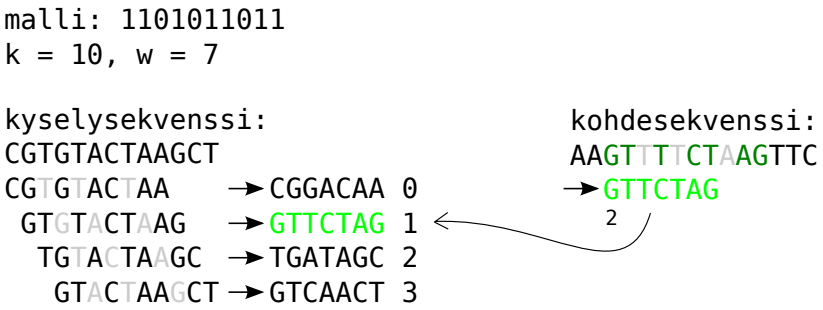
käyttöä ja optimointia sekä tällaisen yhdistelmän entisestään herkkyyttä lisäävää vaikutusta. Viimeisessä aliluvussa kerrotaan lyhyesti optimaalisten mallien ja malliyhdistelmien laskemiseen liittyvistä ongelmista ja kuvataan tehokkaiksi todettuja keinoja kiertää näitä.

3.1 Harvan *k*-meerin toiminta ja käyttö

Harva k-meerin idea perustuu haettavan k-meerin mittaisiin *malleihin*, jotka sisältävät kohtia, joissa merkkien tulee täsmätä, ja kohtia, joissa niiden ei tarvitse täsmätä [MTL02]. Mallia voidaan kuvata merkeistä `1` ja `0` koostuvalla k:n mittaisella binäärimerkkijonolla siten, että täsmättäviä kohtia kuvataan merkillä `1` ja kohtia, joiden ei tarvitse täsmätä, merkillä `0`. Täsmättävien kohtien määrää w, kutsutaan mallin *painoksi*. Esimerkiksi verrattaessa 5-meeriä `GATAT` mallilla `11011` hyväksyttäisiin osumiksi 5-meerin itsensä lisäksi myös 5-meerit `GAGAT`, `GACAT` ja `GAAAT`.

Harvaa k-meeriä hyödyntävän algoritmin toimintaperiaate on pääpiirteiltään sama kuin edellisessä luvussa esitellyn yhtenäisiä, eli tavallisia, k-meerejä käyttävän algoritmin. Ensiksi käydään kyselysekvenssi läpi indeksi indeksiltä tallentaen jokainen sekvenssistä löytyvä k-meeri avaimeksi ja sen indeksi arvoksi hajautustauluun. k-meeriä ei kuitenkaan tallenneta kokonaisuudessaan, vaan ainoastaan mallin osoittamat täsmättävät kohdat sisältävä w-meeri [MTL02]. Siis jos käytössä on malli `11011` ja kyselysekvenssin indeksissä i esiintyvä 5-meeri on `ACTAG`, tallennetaan hajautustauluun 4-meeri `ACAG`. Tämän jälkeen käydään kohdesekvenssi vastaavasti läpi luoden mallin avulla jokaisesta k-meeristä mallin mukainen lyhyempi w-meeri ja etsitään mahdollista osumaa (kuva 3.1). Osumien pidentäminen ja hyväksyminen toimivat kuten aikaisemminkin. Aikavaativuuteen harvan k-meerin käytöllä ei luonnollisesti ole merkitsevää vaikutusta.

Kun yhtenäisen k-meerin kohdalla voidaan herkkyyttä lisätä arvoa k pienentämällä, harvoja k-meerejä käytettäessä herkkyyttä voidaan nostaa pienentämällä harvan k-meerin painoa w [MTL02]. Haittapuolena tässä on suoritusajan kasvu, sillä aivan kuten k:n arvoa pienentämällä yhtenäisillä k-meereillä, myös painoa w vähentämällä kasvaa turhien osumien määrä. Odotettu osumamäärä onkin riippuvainen ainoastaan mallin painosta w



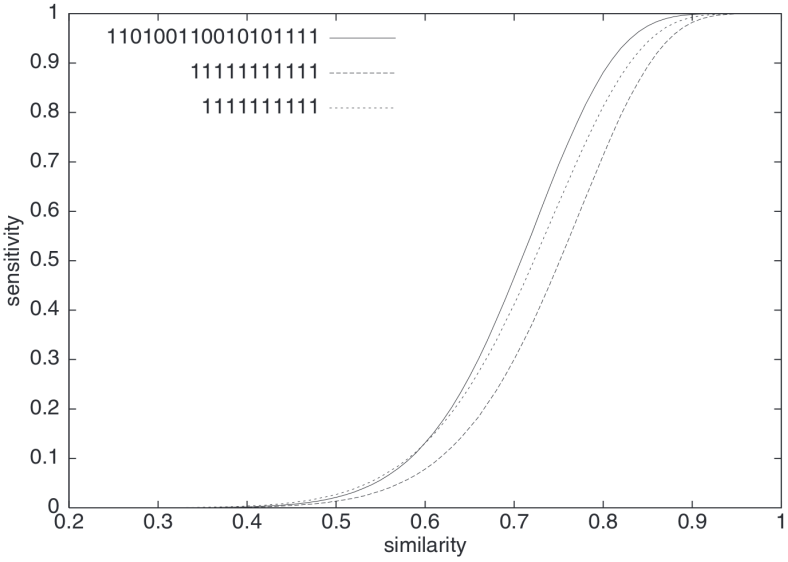
Kuva 3.1: Esimerkki harvan k-meerin käytöstä. Kuvassa on käytössä 10 merkin mittainen malli, jonka paino on seitsemän. Algoritmi toimii hyvin samankaltaisesti kuin kuvassa 2.4, mutta hajautustauluun tallennettaessa 10-meerit muutetaan 7-meereiksi ja samoin tehdään jokaisen kohdesekvenssissä olevan 10-meerin kohdalla.

(yhtenäisellä k-meerillä $w = k$), sen pituudesta k sekä sekvenssien samankaltaisuudesta p. Täsmättävien kohtien sijainnilla mallissa ei ole osumamäärään vaikutusta. Tästä johtuen mallin valinnalla voidaan herkkyyteen vaikuttaa ilman negatiivisia seurauksia algoritmin suoritusaikaan [MTL02, CZ04].

3.2 Optimaalinen malli

Sopivan mallin käytöllä voidaan algoritmin herkkyyttä nostaa useita kymmeniä prosentteja [MTL02]. Esimerkiksi kuvan 3.2 mallia `110100110010101111` käyttävän 18-meerin ($w = 11$) on laskettu tuottavan herkkyyteen noin 50 prosentin parannuksen yhtenäiseen 11-meeriin verrattuna, kun etsittävät linjaukset ovat 64 emäksen mittaisia ja linjattavien alueiden samankaltaisuus on 0.7. Toisaalta kuvassa 3.3 näkyy, kuinka mallia `111001010011011` käyttävän 15-meerin ($w = 9$) herkkyys on yhtenäistä 8-meeriä suurempi isoilla samankaltaisuuksilla, mutta laskee 8-meerin herkkyyden alapuolelle samankaltaisuuden ollessa alle 0.64. Toki on huomioitava, että edellä mainituista syistä yhtenäinen 8-meeri tuottaa noin neljä kertaa suuremman määrän osumia kuin painon 9 harva k-meeri.

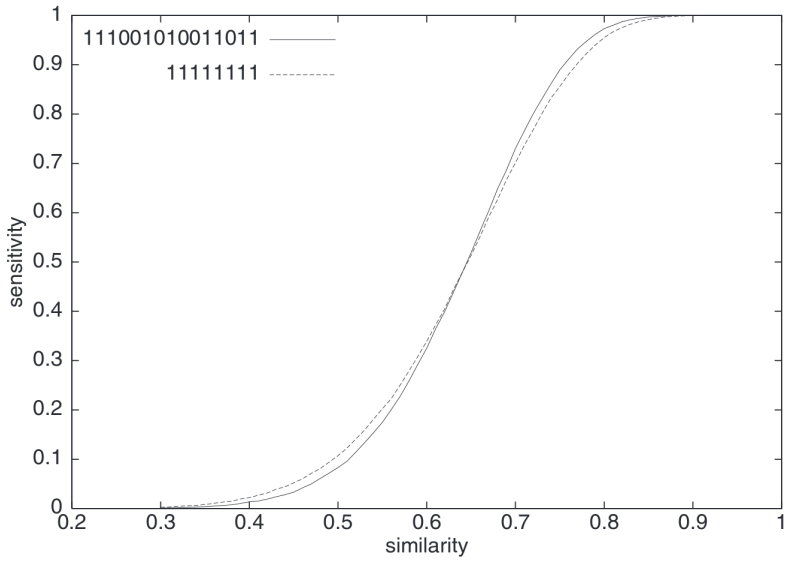
Painolle w ei olekaan olemassa yhtä ainoaa optimaalista mallia, vaan kullekin pituuden k, painon w sekä homologisten alueiden samankaltaisuuden p ja pituuden L yhdistelmälle on teoriassa laskettavissa optimaalinen malli



Kuva 3.2: Herkkyyden vertailua lähes optimaalisen painon 11 harvan 18-meerin (yhtenäinen viiva), yhtenäisen 11-meerin (katkoviiva) ja yhtenäisen 10-meerin (pisteviiva) välillä. Kuva kopioitu artikkelista [MTL02].

[CZ04, LMZ06, KLMT04]. Optimaalinen malli voidaan myös laskea asymptoottiselle homologisen alueen pituudelle, jolloin pituus voidaan oleellisesti jättää huomiotta sopivaa mallia valittaessa [LMZ06].

Taulukossa 3.1 nähdään herkkyysvertailua seitsemän erilaisen painon 10 mallin välillä. (Mallin herkkyydellä tarkoitetaan sitä käyttävän k-meerin herkkyyttä.) Vaikka yhtenäinen malli, joka on käytössä tavallisessa k-meerissä, pärjää keskimäärin huonosti muihin malleihin nähden, on huomioitavaa, että se ei ole vertailun heikoin malli. Alhaisimman herkkyyden saanut malli `101010101010101011` muistuttaakin läheisesti *tasaisesti jakautunutta mallia*, jolla on osoitettu olevan jopa yhtenäistä mallia huonompi herkkyys [CZ04, BBV04]. Tasaisesti jakautuneella mallilla tarkoitetaan mallia, joka on muotoa $(\texttt{10}^a)^n\texttt{1}$, missä $a, n \geq 1$, eli mallia josta löytyy merkkejä `1` ainoastaan tasavälein (esimerkiksi `1001001001`). Pelkistä merkeistä `1` koostuva yhtenäinen malli onkin tasaisesti jakautuneen mallin erikoistapaus, jossa $a = 0$ [II07a].



Kuva 3.3: Painon 9 harvan 15-meerin ja yhtenäisen 8-meerin herkkyyden vertailua. Kuva kopioitu artikkelista [MTL02].

Se miksi jotkut mallit ovat herkempiä kuin toiset, selittyy sillä, että eri mallit tuottavat eri määriä osittain päällekkäisiä osumia [LMZ06, II07a, II07b]. Useat päällekkäiset osumat ovat turhia, sillä niistä saadaan luotua lopulta maksimissaan yksi linjaus. Koska saman painoisilla malleilla on sama odotettu kokonaisosumamäärä, ovat päällekkäiset osumat merkki osumien kasautumisesta. Mallin herkkyys onkin vahvasti riippuvainen ei-päällekkäisten osumien lukumäärästä ja siis kääntäen verrannollinen päällekkäisten osumien määrään nähden.

Optimaalisinkaan harva k-meeri ei yllä herkkyydessä samalle tasolle kuin dynaamiseen ohjelmointiin perustuvat algoritmit, koska osa linjauksista saattaa jäädä edelleen huomaamatta, jos mutaatio sattuu olemaan mallin kannalta väärässä paikassa. Esimerkiksi samankaltaisuudella 0.7 optimaaliseksi osoitettu painon 11 malli `111010010100110111` ei selkeästi voi tuottaa osumaa, jos verrattavissa k-meereissä on kolme perättäistä epätäsmäystä [MTL02]. Tähän tuovat apua seuraavaksi käsiteltävät useamman mallin yhdistelmät.

Malli	Ihminen vs. Banaanikärpänen	Hiiri
11011011000011011	86 %	94 %
11011000011011011	85 %	94 %
11001011001011011	82 %	93 %
11011011011011	81 %	92 %
111001001001010111	59 %	87 %
1111111111	45 %	82 %
101010101010101011	38 %	80 %

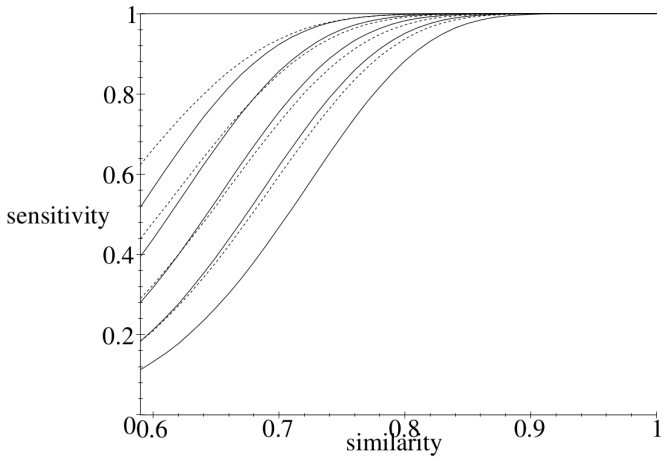
Taulukko 3.1: Vertailua erilaisten biologisten sekvenssimallien pohjalta luotujen optimaalisten mallien välillä, kun useita ihmisen geenejä on linjattu hiiren ja banaanikärpäsen vastaavan geenien kanssa. Prosenttiluvut kuvaavat malleilla löytyneiden lokaalien linjausten osuutta kaikista löydettävissä olevista linjauksista. Taulukon tiedot poimittu artikkelissa [BBV04] esiintyvästä laajemmasta taulukkosta.

3.3 Malliyhdistelmät ja niiden optimointi

Yhden mallin sijaan voidaan käyttöön ottaa myös useampia malleja yhtaikaisesti. Yhtaikaisella käytöllä tarkoitetaan sitä, että verrattavat k-meerit hyväksytään osumaksi, jos yhdenkin käytössä olevan mallin mukaan luodut w-meerit täsmäävät. Tämä on osoittautunut tehokkaaksi keinoksi nostaa herkkyyttä entisestään [LMKT04].

Kuvassa 3.4 nähdään, miten optimaalisen malliyhdistelmän mallien määrän n kaksinkertaistaminen lisää herkkyyttä lähes samassa suhteessa kuin yksittäisen mallin painon vähentäminen yhdellä. Kuten aikaisemmin todettiin, k-meerin painon vähentäminen yhdellä lisää osumien määrän odotusarvoa noin nelinkertaiseksi. Sen sijaan mallien määrän tuplaaminen ainoastaan kaksinkertaistaa osumien määrän [LMKT04]. Tämä johtuu siitä, että erilaiset mallit tuottavat erilaisia w-meerejä ja täten mallien määrän kaksinkertaistuessa kaksinkertaistuu myös hajautustaulun w-meerien määrä. Osumien määrä puolestaan kasvaa samassa suhteessa hajautustaulun koon kanssa.

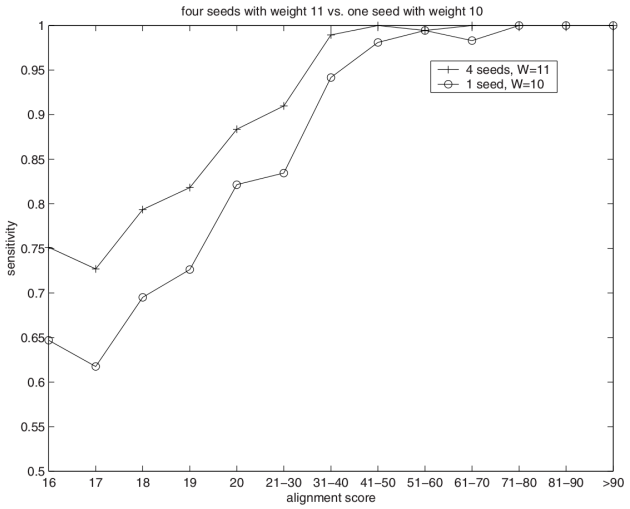
Lisääntynyttä osumien määrää voidaan kompensoida kasvattamalla malliyhdistelmän mallien painoa [YWC+04]. Koska mallin painon nostaminen yhdellä vähentää odotetun osumamäärän neljäsosaan, saadaan osumien määrä pidettynä samalla tasolla käyttämällä nelinkertaista määrää painoltaan



Kuva 3.4: Mallin painon vähentämisen ja useamman mallin käytön vaikutuksia herkkyyteen. Yhtenäiset viivat kuvaavat painon 11 malliyhdistelmien (järjestyksessä alhaalta ylös: $n = 1, 2, 4, 8, 16$) herkkyyksiä erilaisilla samankaltaisuuksilla. Katkoviivat kuvaavat eri painoisia yksittäisiä malleja (järjestyksessä alhaalta ylös: $w = 10, 9, 8, 7$). Kuva kopioitu artikkelista [LMKT04].

pykälää suurempaa mallia. Optimaalisella yhdistelmällä herkkyys nousee kokonaisuudessaan tällöin samassa suhteessa kuin mallin painon vähentäminen yhdellä. Sopivaa malliyhdistelmää käyttämällä voidaan siis saada nostettua herkkyyttä samassa suhteessa kuin painoa vähentämällä, vaikuttamatta kuitenkaan osumien odotettuun määrään. Tätä on havainnollistettu kuvassa 3.5, jossa on vertailtu neljän mallin yhdistelmän ja painoltaan yhtä pienemmän yksittäisen optimaalisen mallin herkkyyksiä suhteessa tuotettujen linjausten pistemäärään. Edellä mainituista syistä odotettu osumamäärä on molemmilla sama, mutta yhdistelmällä voidaan herkkyydessä päästä jopa 15 prosentin suhteelliseen parannukseen yksittäiseen malliin nähden.

Malliyhdistelmiä käytettäessä kannattaa mallit valita siten, että ne eroavat toisistaan useammasta kohtaa, jotta tunnistuskattavuus saadaan mahdollisimman suureksi [LMKT04]. Jos mallit eroavat toisistaan vain vähän ja ensimmäinen malli ei linjausta löydä, jää linjaus todennäköisesti huomaamatta muiltakin malleilta. Kuvassa 3.6 on havainnollistettu, miten kahdelta toisiaan muistuttavalla mallilta jää huomaamatta linjauksia, joita kaksi



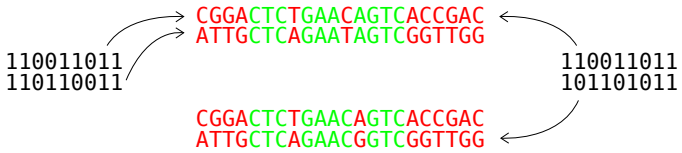
Kuva 3.5: Heuristisella menetelmällä tuotetun neljän mallin yhdistelmän (ylempi viiva) ja yksittäisen mallin (alempi viiva) herkkyysarvoja suhteessa tuotettujen linjauksien pistemäärään. Eri painoista johtuen malliyhdistelmä ($w = 11$) ja yksittäinen malli ($w = 10$) tuottavat keskimäärin saman määrän osumia. Kuva kopioitu artikkelista [XBLM04].

enemmän toisistaan eroavaa mallia havaitsee.

Ongelmana useamman erilaisen mallin yhtäaikaisessa käytössä on muun muassa suuremman hajautustaulun vaatima tila [LMKT04]. Yhden mallin tuottamien w-meerin tilavaativuus on $O(w(S - w + 1))$, missä S on kyselysekvenssin pituus. Kun jokainen malli tuottaa erilaiset w-meerit hajautustauluun, kasvaa hajautustaulun koko lineaarisesti käytössä olevien mallien määrän suhteen. Ratkaisuksi tähän voidaan pitkät kyselysekvenssit jakaa osiin ja algoritmi suorittaa jokaiselle osalle erikseen. Mahdolliset katkeavat linjaukset voidaan yhdistää jälkikäteen. Toinen, hyvin hankalaksi osoittautunut ongelma on optimaalisten malliyhdistelmien laskeminen, jota käsitellään seuraavassa aliluvussa.

3.4 Optimaalisten mallien ja malliyhdistelmien laskeminen

Jotta voisi löytää optimaalisen mallin halutulle painolle w, k-meerin pituudelle k, homologisten alueiden samankaltaisuudelle p ja pituudelle L,



Kuva 3.6: Esimerkki malliyhdistelmän valinnan seurauksista. Kuvassa vasemmalla oleva kahden mallin yhdistelmä hyväksyy linjauksessa esiintyvät kaksi mutaatiota jos ne ovat etäisyyksillä 1, 3 tai 4, kun oikealla olevalla yhdistelmällä etäisyydet voivat olla 1, 2, 3, 4 ja 5. Tästä syystä vasemmanpuoleisella yhdistelmällä löytyy vain toinen kuvassa olevista linjauksista, kun oikealla olevalla yhdistelmällä löytyy molemmat.

on ensin kyettävä laskemaan mallin herkkyys. Mallin herkkyyden eksaktiin laskemiseen on esitetty useita dynaamista ohjelmointia hyödyntäviä, aikavaativuudeltaan eksponentiaalisia algoritmeja, ja onkin osoitettu, että herkkyyden laskeminen on NP-kova ongelma [LMKT04, ML07, LMZ06]. Koska eri mallivaihtoehtoja on eksponentiaalinen määrä suhteessa pituuden ja painon erotukseen $(k - w)$, on myös parhaan mallin valitseminen NP-kova ongelma [LMZ06]. Malliyhdistelmien laskeminen on niin ikään NP-kova ongelma, sillä myös yhdistelmävaihtoehtoja on eksponentiaalinen määrä [LMKT04, LMZ06]. Vaikka jokainen näistä operaatioista täytyy suorittaa periaatteessa vain kertaalleen, kasvaa suoritusaika painoltaan suurilla malleilla ($w > 13$) liian suureksi. Malliyhdistelmiä ei voida näillä menetelmillä käytännössä laskea [II07a]. Useita heuristisia algoritmeja herkkyyden, optimaalisten mallien ja optimaalisten malliyhdistelmien laskemiseen onkin kehitetty [LMZ06, ML07, II07a, CZ04, YWC+04, LMKT04].

Eräät herkkyyden laskemiseen tarkoitetuista algoritmeista käyttävät arviointiin jotain herkkyyden kanssa korreloivaa, mutta helpommin laskettavaa ominaisuutta [YWC+04, II07a, CZ04]. Jokaiselle mallille voidaan esimerkiksi laskea *päällekkäisyyskompleksisuus* (engl. overlap complexity) liu'uttamalla mallia itsensä ohitse indeksi indeksiltä ja laskemalla, montako kertaa mallin merkit 1 osuvat päällekkäin (kuva 3.7) [MTL02, YWC+04, II07a]. Päällekkäisyyskompleksisuudella on vahva korrelaatio mallin tuottamien päällekkäisten osumien suhteelliseen määrään, jonka aikaisemmin todettiin olevan kääntäen

verrannollinen mallin herkkyyteen (ja jonka laskeminen on myös NP-kova ongelma). Lisäksi on osoitettu, että herkkyydeltään heikoimmilla tasaisesti jakautuneilla malleilla on kaikista malleista suurimmat päällekkäisyyskompleksisuudet. Aikavaativuus päällekkäisyyskompleksisuuden laskemiseen on neliöllinen suhteessa mallin pituuteen.

					1	0	1	1	0	1						
1	0	1	1	0	1											1
	1	0	1	1	0	1										0
		1	0	1	1	0	1									2
			1	0	1	1	0	1								2
				1	0	1	1	0	1							1
					1	0	1	1	0	1						4
						1	0	1	1	0	1					1
							1	0	1	1	0	1				2
								1	0	1	1	0	1			2
									1	0	1	1	0	1		0
										1	0	1	1	0	1	1
																16

Kuva 3.7: Päällekkäisyyskompleksisuuden lasku mallille `101101`. Rivit kuvaavat eri tilanteita, kun malli liu'utetaan itsensä ohitse. Jokaiselle riville on laskettu päällekkäin osuvien merkkien `1` määrä. Lopullinen päällekkäisyyskompleksisuus (16) saadaan summaamalla rivit yhteen.

Pienimpien päällekkäisyyskompleksisuuksien omaavien mallien on todettu olevan lähes yhtä herkkiä kuin optimaaliseksi osoitettujen mallien, vaikka päällekkäisyyskompleksisuus ei ota homologisten alueiden samankaltaisuutta lainkaan huomioon [II07a, II07b]. Esimerkiksi samankaltaisuuksilla $0.7 \leq p \leq 0.9$, painon $9-18$ optimaalisten[3] mallien herkkyyksien vaihdellessa välillä $0,12...0,99$, on päällekkäisyyskompleksisuudeltaan pienimpien mallien keskimääräinen ero herkkyydessä optimaalisiin malleihin nähden alle $0,0005$. Huomioitavaa onkin, että vaikka homologisten alueiden samankaltaisuudella on vaikutusta mallin optimaalisuuteen, koostuu mallien parhaimmisto eri samankaltaisuuksilla pääasiassa samoista malleista, mutta hieman eri järjestyksessä.

[3]Koska varmasti optimaalisia malleja, joilla $w > 13$, ei voida nykymenetelmin tuottaa, ei kaikkien viitatussa mittauksessa käytettyjen mallien optimaalisuudesta ole täyttä varmuutta. Mallit on tuotettu edellä mainittua menetelmää hitaammalla menetelmällä, joka pienemmillä painoilla ($w \leq 13$) tuottaa optimaaliseksi todettuja malleja [CZZ04].

Parasta mallivaihtoehtoa etsittäessä voidaan kaikkien eri mallivaihtoehtojen läpikäymisen sijaan esimerkiksi lähteä yksinkertaisesta mallista liikkeelle ja ahneella algoritmilla vaihtaa yksitellen merkkien `0` ja `1` paikkoja valitsemalla vaihto, joka tuottaa herkkyyteen suurimman parannuksen [II07a, II07b]. Tätä jatketaan kunnes herkkyyttä parantavaa vaihtoehtoa ei enää löydy. Käyttämällä herkkyyden laskemiseen päällekkäisyyskompleksisuutta, saadaan algoritmi toimimaan polynomisessa ajassa, minkä ansiosta voidaan tuottaa malleja hyvinkin suurilla painolla ($w > 64$). Menetelmällä tuotettujen mallien keskimääräinen herkkyysero edellisessä kappaleessa mainituilla painoilla ja samankaltaisuuksilla optimaalisin malleihin nähden on alle $0,0015$.

Lähes optimaalisia malliyhdistelmiä voidaan tuottaa vastaavasti edellä mainittuja menetelmiä hyödyntäen [II07a]. Yhden mallin sijaan algoritmi muokkaa n:ää eri mallia toteuttaen vaihdon aina siinä mallissa, jossa se tuottaa suurimman parannuksen herkkyyteen. Menetelmän aikavaativuus on polynominen, kuten yhden mallin tapauksessa, ja esimerkiksi 16 mallin yhdistelmä ($w = 11$) voidaan tuottaa sillä alle kymmenessä minuutissa.

Koska absoluuttisesti optimaalisten malliyhdistelmien laskeminen ei nykytekniikalla ole käytännössä mahdollista, on menetelmän tehokkuutta vaikea arvioida. Tästä huolimatta, kuten kuvasta 3.4 nähdään, voidaan heuristisilla menetelmillä tuotettuja malliyhdistelmiä käyttämällä selkeästi nostaa algoritmin herkkyyttä.[4]

4 Yhteenveto

Biologisen aineiston valtavan koon ja sen alati kasvavan määrän takia sen käsittelyyn tarvitaan tehokkaita algoritmeja. Muun muassa sekvensointidatan koonnissa ja homologisten alueiden etsimisessä käytettävään DNA-sekvenssien lokaalien linjauksien etsimiseen onkin kehitetty vuosien saatossa useita, toinen toistaan nopeampia ja tarkempia algoritmeja. Osa näistä algoritmeista perustuu yhteisten k:n mittaisten osamerkkijonojen eli k-meerien

[4]Kuvan malliyhdistelmät on tuotettu esitettyä heuristiikkaa hitaammalla heuristiikalla, mutta vastaavat herkkyydeltään kappaleessa kuvaillulla menetelmällä tuotettuja yhdistelmiä [II07a, LMKT04].

etsintään haku- ja kohdesekvensseistä hajautustaulua hyödyntäen.

Toisin kuin lopulliset linjaukset, tavalliset k-meerit eivät itsessään kuitenkaan voi sisältää mutaatioita, vaan niiden tulee olla identtisiä. Tämä aiheuttaa tilanteita, joissa osa hyvistäkin linjauksista jää löytymättä. k:n arvoa pienentämällä voidaan nostaa osuman todennäköisyyttä ja saada hukattujen linjausten määrää pienennettyä. Samalla kasvaa myös turhien, linjausta tuottamattomien osumien määrä, mikä hidastaa algoritmin toimintaa.

Ratkaisuksi tähän on kehitetty harva k-meeri, joka sallii epätäsmäyksien esiintyä tietyissä kohdissa k-meerissä itsessään ilman, että tällä on vaikutusta algoritmin nopeuteen. Harvan k-meerin toiminta perustuu malleihin, jotka kertovat, missä kohtaa k-meeriä epätäsmääviä merkkejä saa esiintyä. Optimaalisia malleja käyttävien harvojen k-meerien on laskettu ja kokeellisesti osoitettu löytävän herkemmin lokaaleja linjauksia suhteessa tuotettujen osumien määrään kuin tavallisten, yhtenäisten k-meerien. Useamman mallin yhtäaikaisella käytöllä herkkyyttä voidaan nostaa entisestään.

Ei kuitenkaan ole olemassa yhtä ainoaa optimaalista mallia, vaan mallin optimaalisuus riippuu homologisten alueiden samankaltaisuudesta ja pituudesta sekä harvan k-meerin pituudesta ja sen painosta. Optimaalisen mallin laskeminen on osoittautunut NP-kovaksi ongelmaksi, kuten myös optimaalisen yhdistelmän. Heuristisilla menetelmillä saadaan kuitenkin tuotettua malleja, joiden herkkyys jää vain murto-osan optimaalisten mallien herkkyydestä homologisten alueiden samankaltaisuudesta riippumatta. Kun myös malliyhdistelmiä voidaan tuottaa heuristisia menetelmiä hyödyntäen, on harvaa k-meeriä hyödyntämällä mahdollista päästä lähes maksimaaliseen herkkyyteen nopeuksilla, jotka eivät yhtenäisillä k-meereillä olisi mahdollisia. Tämä mahdollistaa muun muassa sekvensointidatan nopeamman kokoamisen sekä alati kasvavien geenitietokantojen entistä tehokkaamman hyödyntämisen.

Lähteet

[AGM+90] Altschul, Stephen F, Gish, Warren, Miller, Webb, Myers, Eugene W ja Lipman, David J: *Basic local alignment search tool.* Journal of Molecular Biology, 215(3):403–410, 1990.

[AMS+97] Altschul, Stephen F, Madden, Thomas L, Schäffer, Alejandro A, Zhang, Jinghui, Zhang, Zheng, Miller, Webb ja Lipman, David J: *Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.* Nucleic Acids Research, 25(17):3389–3402, 1997.

[BBV04] BREJOVÁ, BROŇA, Brown, Daniel G ja VINAŘ, TOMÁŠ: *Optimal spaced seeds for homologous coding regions.* Journal of Bioinformatics and Computational Biology, 1(04):595–610, 2004.

[CWC04] Cameron, Michael, Williams, Hugh E ja Cannane, Adam: *Improved gapped alignment in BLAST.* IEEE/ACM Transactions on Computational Biology and Bioinformatics, 1(3):116–129, 2004.

[CZ04] Choi, Kwok Pui ja Zhang, Louxin: *Sensitivity analysis and efficient method for identifying optimal spaced seeds.* Journal of Computer and System Sciences, 68(1):22–40, 2004.

[CZZ04] Choi, Kwok Pui, Zeng, Fanfan ja Zhang, Louxin: *Good spaced seeds for homology search.* Teoksessa *Fourth IEEE Symposium on Bioinformatics and Bioengineering*, sivut 379–386. IEEE, 2004.

[II07a] Ilie, Lucian ja Ilie, Silvana: *Fast computation of good multiple spaced seeds.* Teoksessa *Algorithms in Bioinformatics*, sivut 346–358. Springer, 2007.

[II07b] Ilie, Lucian ja Ilie, Silvana: *Long spaced seeds for finding similarities between biological sequences.* BIOCOMP, 7:25–28, 2007.

[JP04] Jones, Neil C ja Pevzner, Pavel A: *An Introduction to Bioinformatics Algorithms.* MIT Press, 2004.

[KLMT04] Keich, Uri, Li, Ming, Ma, Bin ja Tromp, John: *On spaced seeds for similarity search.* Discrete Applied Mathematics, 138(3):253–263, 2004.

[LH10] Li, Heng ja Homer, Nils: *A survey of sequence alignment algorithms for next-generation sequencing.* Briefings in Bioinformatics, 11(5):473–483, 2010.

[LMKT04] Li, Ming, Ma, Bin, Kisman, Derek ja Tromp, John: *PatternHunter II: Highly sensitive and fast homology search.* Journal of Bioinformatics and Computational Biology, 2(03):417–439, 2004.

[LMZ06] Li, Ming, Ma, Bin ja Zhang, Louxin: *Superiority and complexity of the spaced seeds.* Teoksessa *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, sivut 444–453. Society for Industrial and Applied Mathematics, 2006.

[ML07] Ma, Bin ja Li, Ming: *On the complexity of the spaced seeds.* Journal of Computer and System Sciences, 73(7):1024–1034, 2007.

[MTL02] Ma, Bin, Tromp, John ja Li, Ming: *PatternHunter: faster and more sensitive homology search.* Bioinformatics, 18(3):440–445, 2002.

[PPDS04] Pop, Mihai, Phillippy, Adam, Delcher, Arthur L ja Salzberg, Steven L: *Comparative genome assembly.* Briefings in Bioinformatics, 5(3):237–248, 2004.

[RUC+11] Reece, Jane B, Urry, Lisa A, Cain, Michael L, Wasserman, Steven A, Minorsky, Peter V ja Jackson, Robert B: *Campbell Biology.* Pearson Education, 9. painos, 2011.

[SW81] Smith, Temple F ja Waterman, Michael S: *Identification of common molecular subsequences.* Journal of Molecular Biology, 147(1):195–197, 1981.

[XBLM04] Xu, Jinbo, Brown, Daniel G, Li, Ming ja Ma, Bin: *Optimizing multiple spaced seeds for homology search.* Teoksessa *Combinatorial Pattern Matching*, sivut 47–58. Springer, 2004.

[Xio06] Xiong, Jin: *Essential Bioinformatics.* Cambridge University Press, 2006.

[YWC+04] Yang, I Hsuan, Wang, Sheng Ho, Chen, Yang Ho, Huang, Pao Hsian, Ye, Liang, Huang, Xiaoqiu, Chao, Kun Mao *et al.*: *Efficient methods for generating optimal single and multiple spaced seeds.* Teoksessa *Fourth IEEE Symposium on Bioinformatics and Bioengineering*, sivut 411–416. IEEE, 2004.