Juha Riikonen

Needleman–Wunsch-algoritmi biosekvenssien rinnastuksessa

Tietotekniikan kandidaatintutkielma

4. kesäkuuta 2019

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Juha Riikonen

Yhteystiedot: juha.s.riikonen@student.jyu.fi

Työn nimi: Needleman–Wunsch-algoritmi biosekvenssien rinnastuksessa

Title in English: The Needleman–Wunsch-algorithm in biosequence alignment

Työ: Kandidaatintutkielma

Sivumäärä: 26+0

Tiivistelmä: Jatkuvasti kasvava biologisen datan määrä asettaa uusia vaatimuksia biosekvenssien rinnastusalgoritmeille bioinformatiikan alalla. Tässä kandidaatintutkielmassa on aihetta käsittelevän lähdekirjallisuuden avulla kartoitettu bioinformatiikan ja molekyylibiologisten tietokantojen nykytilannetta, ja tarkasteltu biosekvenssien rinnastuksessa käytetyn Needleman–Wunsch-algoritmin toimintaa ja tehokkuutta. Aikoinaan uraauurtava Needleman–Wunsch-algoritmi vaatii kuitenkin kohtuuttoman paljon laskenta-aikaa suurissa tietokantahauissa, joita mielekäs biologisen datan käsittely nykypäivänä vaatii. Tässä kohtaa kuvioihin astuvat tulosten optimaalisuudessa joustavat heuristiset rinnastusalgoritmit, joiden avulla usean pitkänkin sekvenssin samanaikainen rinnastus on mahdollista kohtuullisessa ajassa.

Avainsanat: Bioinformatiikka, Needleman–Wunsch-algoritmi, Biosekvenssien rinnastaminen

Abstract: The constantly increasing amount of biological data is setting new standards for the efficiency of biosequence alignment algorithms in the field of bioinformatics. In this literature review we survey the current state of bioinformatics and biological databases, and focus on the Needleman–Wunsch algorithm used in biosequence alignment. However, the once groundbreaking algorithm proves to be too time consuming in performing large database queries, which are required to efficiently utilize the data in existing biological databases. This is where heuristic alignment algorithms step in, which are able to align multple long sequences simultaneously in a reasonble time at the cost of specificity.

Keywords: Bioinformatics, Needleman–Wunsch algorithm, Biosequence alignment

Kuviot

Sisältö

1 Johdanto

DNA- ja aminohapposekvenssit ovat esimerkkejä biosekvensseistä, joiden rinnastus on keskeisessä asemassa laskennalliseen biologiaan sisältyvällä bioinformatiikan alalla. Sekvensoinnilla tarkoitetaan peräkkäisten biomolekyylien kuvaamista merkkijonojen muodossa, joita kutsutaan sekvensseiksi. Esimerkiksi DNA-sekvenssi esitetään useimmiten neljästä aakkosesta (A, C, T ja G) muodostettuna merkkijonona, jossa kukin merkki kuvaa yhtä DNA-rihman muodostavaa emästä (adeiini, sytosiini, guaniini ja tymidiini) sen alkukirjaimen mukaisesti (Tuimala 2007, s. 34). Vastaavasti peräkkäisistä aminohapoista muodostuva proteiinisekvenssi kuvataan siten, että yksi aminohappo vastaa yhtä merkkiä 20 merkin aakkostosta. Biologista dataa varastoivat molekyylibiologiset tietokannat voivat sisältää sekvenssejä, jotka merkkijonona esitettyinä voivat olla jopa miljardien merkkien pituisia. Sekvenssien rinnastuksella taas tarkoitetaan näiden merkkijonojen samankaltaisuuden tarkastelua tietokoneavusteisesti. Setubalin ja Meidanisin (1997, s. 47) mukaan sekvenssien rinnastus on tärkein ja perustavanlaatuisin operaatio laskennallisessa biologiassa, jolla voidaan ratkaista monia molekyylibiologisia ongelmia.

Tutkielmani on toteutettu kirjallisuuskatsauksena, jossa alan tieteellisen lähdekirjallisuuden avulla esitellään tutkimusongelmat ja pyritään vastaamaan tutkimuskysymyksiin. Tavoitteena on pääpiirteittäin esitellä laskennallisen biologian ja sekvenssien rinnastuksen käsitteet, ja keskittyä tarkastelemaan yhtä sekvenssien rinnastuksessa käytettyä algoritmia, Needleman–Wunsch-algoritmia. Tutkielmassa haluan selvittää kuinka algoritmi toimii, kuinka aikavaativa se on ja minkälaisiin sekvenssien rinnastusta vaativiin molekyylibiologian ongelmien ratkaisemiseen se soveltuu.

Johdannon lisäksi tutkielma koostuu kolmesta sisältöluvusta sekä yhteenvetoluvusta. Luvussa 2 avataan bioinformatiikan käsitettä sen avulla, kuinka se on aihetta käsittelevässä lähdekirjallisuudessa määritelty. Luvun tarkoitus on antaa lukijalle yleiskuva bioinformatiikasta tieteenalana, biologista dataa sisältävistä tietokannoista, biosekvenssien rinnastamisesta sekä tieteenalan kehityksestä sen alkuajoista nykypäivään. Tämän myötä tulevissa sisältöluvuissa on mielekkäämpää keskustella alalla käytetyistä algoritmeista. Luvussa 3 esitellään biosekvenssien rinnastamiseen käytetyn Needleman–Wunsch-algoritmi, sekä tarkastellaan

sen toimintaa. Luvussa 4 vertaillaan Needleman–Wunsch-algoritmin toimintaperiaatteita ja tehokkuutta muiden sekvenssien rinnastukseen käytettyjen algoritmien kanssa. Samassa luvussa tutkitaan myös sitä, minkälaisia rajoitteita Needleman–Wunsch-algoritmilla on, ja milloin on aiheellista harkita jotain muuta sekvenssien rinnastusalgoritmia. Viimeinen, 5. luku, on yhteenveto aikaisempien lukujen sisällöstä. Yhteenvetoluvussa vedetään johtopäätelmiä aikaisemmin todetusta, ja pohditaan mahdollisia aiheita jatkotutkimuksille.

2 Bioinformatiikka

Ennen kuin on merkityksellistä syventyä biosekvenssien rinnastuksessa käytettyihin algoritmeihin sisältöluvuissa 3 ja 4, on tarpeellista luoda yleinen katsaus bioinformatiikasta, ja hieman avata bioinformatiikan tieteenalalla käytettyjä käsitteitä sekä termistöä. Tässä luvussa pyritään määrittelemään bioinformatiikan käsite, ja kartoitetaan bioinformatiikan kehitystä sen syntysijoista nykytilanteeseen. Lisäksi tarkastellaan yksityiskohtaisemmin molekyylibiologisia tietokantoja ja biosekvenssien rinnastusta, jotka molemmat ovat oleellisia kokonaisuuksia bioinformatiikassa sekä tässä tutkielmassa.

Bioinformatiikkaa voidaan pitää monitieteellisenä alana, jossa yhdistyvät biologia, tietojenkäsittelytieteet sekä laskennalliset tieteet. Jeremy Ramsdenin (2015, s. 2) mukaan sopiva määritelmä bioinformatiikalle sen nykyisessä muodossaan on tiede, joka tutkii kuinka informaatiota luodaan, lähetetään, vastaanotetaan, varastoidaan, prosessoidaan ja tulkitaan biologisissa järjestelmissä, tai hieman tiiviimmin ilmaistuna ”informaatiotieteiden soveltaminen biologiassa”.

On syytä huomauttaa, että termillä ”laskennallinen biologia” on perinteisesti tarkoitettu bioinformatiikkaa, koska laskennallisia menetelmiä hyödynnettiin alunperin pääasiassa molekyylibiologiaa käsittelevällä biologian osa-alueella. Nykyisellään termeillä on kuitenkin eroa, ja laskennallisen biologian voidaan pitävän sisällään myös esimerkiksi neurotieteen ja ekologian tutkimusta. Lähteestä riippuen termejä käytetään ristikkäin, ja niiden määritelmät vaihtelevat, mutta tässä tutkielmassa bioinformatiikalla sekä laskennallisella biologialla tarkoitetaan laskennallisten tieteiden hyödyntämistä nimenomaan DNA- ja proteiinisekvenssejä käsittelevän molekyylibiologian osa-alueella.

Luscombe, Greenbaum ja Gerstein (2001) lähestyvät bioinformatiikan määritelmää sen pyrkimysten näkökulmasta. Yksinkertaisimmillaan bioinformatiikka organisoi biologista dataa, kuten DNA- ja proteiinisekvenssejä niitä sisältävissä tietokannoissa siten, että tutkijat pääsevät dataan käsiksi ja pystyvät lisäämään omia tuloksia tähän aineistoon. Toinen tieteenalan pyrkimys on kehittää työkaluja ja resursseja, jotka helpottavat datan analysoimista. Tästä esimerkkinä voidaan pitää tietokantaan lisätyn proteiinisekvenssin samankaltaisuuden tarkaste-

lua tietokannassa jo olemassa olevien sekvenssien kanssa. Tämänkaltaista hakusekvenssin perusteella tapahtuvaa tietokantahakua kutsutaan sekvenssihauksi (engl. *sequence search*), ja ne ovatkin yksi käytetyimmistä sovelluksista bioinformatiikan alalla (Tuimala 2007, s. 81). Kolmas Luscomben, Greenbaumin ja Gersteinin (2001) mainitsema pyrkimys on edellä mainittujen työkalujen hyödyntäminen biologisen datan analysoimisessa ja tulkitsemiseen siten, että tulosten tulkinta on biologisesti mielekästä.

Itse sekvenssien vertailu sekä tietokantahaut tapahtuvat sekvenssien rinnastusalgoritmeilla, jotka vertailevat kahta tai useampaa merkkijonona esitettyä sekvenssiä keskenään. Tietojenkäsittelytieteiden sekä sekvenssejä käsittelevien algoritmien näkökulmasta yksittäinen nukleotidi voidaan ajatella merkkinä ja sekvenssi merkkijonona.

Poikkitieteellinen ala vaatii Ramsdenin (2015, s. 3) sanoin ymmärrystä vähintään matematiikan, biologian, fysiikan ja kemian tieteenaloilta, ja sen soveltaminen lisäksi esimerkiksi tietotekniikkaa, kemiantekniikkaa, bioteknologiaa tai lääketiedettä. Tuimala (2007, s. 20) kuvailee bioinformatiikan käsitteen sisältävän esimerkiksi myös molekyylisystematiikkaa, proteiinien kiderakenteiden analysointia ja geenikartoitusta. Hänen mukaansa bioinformatiikan tutkimuskenttä on niin laaja, ettei käsitteen yksiselitteinen määritteleminen ole helppoa. Kaikkia tutkimuskohteita yhdistää kuitenkin se, että ne kaikki liittyvät biologiaan, ja niihin liittyvät ongelmat vaativat tietokoneavusteisia ratkaisuja.

Bioinformatiikan synnyn tieteenalana voi lähdekirjallisuuden avulla ajoittaa noin 1950-luvun ja 1960-luvun vaihteeseen. Hagen (2000) luonnehtii sen syntyneen 1960-luvun alkupuolella, kymmenen vuotta aikaisemmin, kuin ihmisen DNA:n sekvensointi oli edes toteutettavissa tietokoneavusteisesti. Termiä ”bioinformatiikka” käytettiin kuitenkin ensi kerran vasta vuonna 1978, jolloin se määriteltiin ”tieteenä, joka tutkii informaatioprosesseja biologisissa järjestelmissä” (Selzer, Marhöfer ja Rohwer 2008, s. 9). Tästä huolimatta ymmärrettiin, että yhdistämällä tietokoneavusteinen laskenta sekä molekyylibiologia voitaisiin ratkaista perustavanlaatuisia biotieteiden ongelmia (Hagen 2000, s. 231). Kuitenkin Gauthierin ym. (2018, s. 2) mukaan jo 1950-luvun lopulla lyhyitä aminohapposekvenssejä sekvensoitiin automatisoidusti. Selzerin ym. (Selzer, Marhöfer ja Rohwer 2008, s. IX) määritelmä bioinformatiikan syntysijoista eroaa melko radikaalisti Hagenin ja Gauthierin näkemyksestä. Hänen mukaansa bioinformatiikka syntyi Needlemanin ja Wunschin (1970) ensimmäisen sekvenssienrin-

nastusalgoritmin julkaisun yhteydessä vuonna 1970. Eriävät määritelmät bioinformatiikan synnystä ja sen sisällöstä voidaan ajatella luonnehtivan nuoren tieteenalan laaja-alaisuutta ja nopeaa kehitystä.

Gauthierin (2018) mukaan molekyylibiologian keskusdogmina (engl. *central dogma*) tunnetun teorian oletusten mukaisesti huomattiin, että kaiken elämän määrittelevien proteiinien aminohapposekvenssit (ja täten niiden kolmiulotteinen rakenne) pystyttäisiin johtamaan saman eliön DNA-sekvenssistä. Tämä avasi 1970-luvulla uuden kiinnostuksen kehittää yksinkertaisia ja kustannustehokkaita tapoja sekvensoida DNA:ta. DNA oli biologeille paljon proteiinisekvenssejä informatiivisempaa esimerkiksi lajien geneettistä etäisyyttä kuvaavien fylogeneettisten puiden suunnittelussa (Gauthier ym. 2018, s. 5–6). Tuimalan (2007, s. 34) mukaan myös nykypäivänä sekvensoinnissa suositaan usein DNA:ta aminohapposekvenssien sijaan, sillä se on aminohappojen sekvensointia yksinkertaisempaa, nopeampaa sekä halvempaa.

1990-luvun ja 2000-luvun välissä sekvensoinnin hyödyntämissä teknologioissa tapahtui merkittävää kehitystä, ja sekvensointiin liittyvät kustannukset laskivat (Gauthier ym. 2018, s. 1). Tämä johti datan määrän eksponentiaaliseen kasvuun, ja termiä ”Big Data” alettiin käyttää kuvaamaan valtavaa biologista datamäärää. 2000-luvulta eteenpäin lukuisia uusia sekvensointitekniikoita ja verkossa toimivia palveluita ja tietokantoja on kehitetty jatkuvasti (Selzer, Marhöfer ja Rohwer 2008, s. XI-XII).

2.1 Molekyylibiologiset tietokannat

Tietokoneavusteinen sekvensointi on entistä halvempaa ja helpompaa (Cook ym. 2015), mikä mahdollistaa tutkimuksia ja kokeita, jotka hyödyntävät ja tuottavat valtavia määriä dataa. Erilaiset biologista dataa sisältävät tietokannat mahdollistavat sekvenssidatan tallentamisen, hakemisen ja vertailun muiden tietokantojen sekvenssien kanssa. Kansainvälisesti ensisijaisia sekvenssitietokantoja ovat GenBank, EMBL (European Molecular Biology Laboratory) ja DDBJ (DNA Data Bank of Japan) (Tuimala 2007, s. 44).

Biologiset tietokannat voidaan jakaa eri luokkiin niiden sisällyttämän datan luonteen perusteella. Whitfieldin ym. (2006, s. 629) mukaan osa tietokannoista sisältää niin sanottua ensi-

sijaisesti laboratoriotutkimuksista saatua primääridataa, eli nukleotidisekvenssejä. Tunnistetietokannat (kutsutaan myös sekundääritietokannoiksi) sisältävät primääridatasta johdettua dataa, kuten sekvenssien yhteisiä piirteitä. Tuimala (2007, s. 51) kutsuu tunnistetietokantoja arvokkaiksi tietolähteiksi, sillä niiden avulla on mahdollista selvittää tuntemattomien proteiinien funktioita, tai niiden sukulaissekvenssejä. Toinen Whitfieldin ym. mainitsema yleinen tietokantaformaatti sisältää aminohapoista muodostettuja proteiinisekvenssejä. Proteiinisekvenssejä sisältävät tietokannat voivat olla myös nukleotidisekvensseistä johdettua dataa, sillä nukleotidisekvenssistä pystytään päättelemään siitä muodostuvat aminohappomolekyylit ja niiden järjestys.

Näiden lisäksi on useita tiettyyn aihealueeseen erikoistuneita tietokantoja, sekä tietokantoja, joissa on integroitu dataa useammasta tietokannasta. InterPro on esimerkki integroidusta tietokannasta, joka integroi kymmenen proteiinidataa sisältävän tietokannan datan sekä niiden hyödyntämät työkalut yhden tietokannan alle (Whitfield, Pruess ja Apweiler 2006, 636). Integroiduissa tietokannoissa on samanaikaisesti saatavilla enemmän dataa ilman usean tietokannan tarkastelun luomaa mahdollista datan toisteisuutta. Tiedon haku yhdestä integroidusta tietokannasta usean sijaan tekee käyttäjän tiedonhausta nopeampaa ja tehokkaampaa.

2000-luvun alusta eteenpäin biosekvenssien lukumäärä tietokannoissa on kasvanut valtavaa vauhtia. Esimerkiksi vuonna 2002 EMBL-tietokanta sisälsi noin 31 miljardia nukleotidia 20 miljoonassa eri sekvenssissä, ja sekvenssien lukumäärä lähes kaksinkertaistuu vuosittain. (Tuimala 2007, s. 43). Vuonna 2005 samat lukemat olivat Whitfieldin ym. (2006, s. 631) mukaan lähes 60 miljoonaa sekvenssiä ja yli 100 miljardia nukleotidia. Vaikka sekvenssien määrän kasvu kolmessa vuodessa ei aivan vastaakaan Tuimalan arviota, on sekvenssien määrän kolminkertaistuminen valtava muutos yhden tietokannan sisällä. Samaan aikaan sekvenssidataa sisältävien tietokantojen määrä on myös kasvanut. Nucleic Acids Research -aikakausilehden tietokantoja käsittelevä lehtiliite listasi vuonna 2001 tietokantojen lukumääräksi 96, ja vuonna 2007 tietokantoja oli yli 800 kappaletta (Galperin 2007). Nykyään lukumäärän voidaan olettaa olevan paljon suurempi.

Suuri datan määrä tietokannoissa asettaa haasteita tiedon mielekkäässä käsittelyssä ja tehokkaiden tietokantahakujen tekemisessä. Tietokantojen välinen variaatio datassa ja datan esitysformaateissa hankaloittaa tietokantariippumattomien tietokantakyselyiden ja yksikä-

sitteisen datan esitysmuodon vakiintumista (Goble ja Stevens 2008, 687). Tietokantojen integroiminen, sekä järjestelmäriippumattomien datan tallennustyyppien ja tietokantahakualgoritmien kehittäminen tehostaisi tietokantojen sisältämän datan käsittelyä. Tietokantahauissa hyödynnetään sekvenssien rinnastusalgoritmeja, joita käsitellään tarkemmin alaluvussa 2.2, sekä luvuissa 3 ja 4.

2.2 Biosekvenssien rinnastus

Sekvensoitujen proteiini- ja DNA-sekvenssien samankaltaisuutta voidaan vertailla erilaisten rinnastusalgoritmien avulla. Biologisia sekvenssejä rinnastetaan usein siksi, että ymmärrettäisiin niiden rakennetta ja toiminnallisuutta. Yleisin sovellutus sekvenssien rinnastukselle on sekvenssien noutaminen tietokannasta samankaltaisuuden perusteella (Xiong 2006, s. 51). Sung (2009, s. 3-4) toteaa, että jos kahdella proteiini-, DNA-, tai RNA-sekvenssillä on samankaltaiset sekvenssit, on niillä myös samankaltaiset rakenteet tai samankaltaista toiminnallisuutta. Sekvenssien rinnastuksella voidaan ratkaista useita molekyylibiologian ongelmia, joista hän antaa useita esimerkkejä. Esimerkiksi tuntemattoman geenin biologista toiminnallisuutta voidaan ennustaa toteamalla sen sekvenssi samankaltaiseksi, kuin toisen geenin sekvenssi, jonka toiminnallisuus tunnetaan. Toinen mahdollisuus on estimoida kahden lajin geneettistä etäisyyttä niiden genomien samankaltaisuuden perusteella.

Tuimalan (2007, s. 8) mukaan rinnastuksella voidaan samankaltaisuuden lisäksi selvittää kohdat sekvensseissä, joissa on tapahtunut evolutiivisia muutoksia, kuten mutaatioita, insertioita tai deleetioita. Insertiolla tarkoitetaan sekvenssissä tapahtunutta mutaatiota, jossa sekvenssiin on tapahtunut yhden tai useamman molekyylin, eli merkin, lisäys. Deleetiossa puolestaan yksi tai useampi molekyyli on poistunut sekvenssistä. Muita Tuimalan mainitsemia mahdollisia mutaatiotyyppejä ovat inversio ja kahdentuma. Inversio on mutaatio, jossa osa sekvenssin merkeistä on kääntynyt päinvastaiseen järjestykseen. Kahdentumassa taas osa sekvenssiä on kopioitunut, ja täten esiintyy useamman kerran pitkin sekvenssiä.

Sekvenssien rinnastusalgoritmit ottavat syötteenä rinnastettavat sekvenssit, pisteyttävät ne ja palauttavat optimaalisimman rinnastuksen. Rinnastuksista optimaalisin on se, joka tietyillä kriteereillä tuottaa suurimman pistemäärän (Tuimala 2007, s. 70). Ramsdenin (2015, s. 210)

sanoin sekvensseissä tapahtuvat mutaatiot, insertiot sekä deleetiot hankaloittavat merkkijonojen samankaltaisuuden vertailua. Tästä syystä rinnastusalgoritmien täytyy sallia merkkijonojen merkkien väliin asetettavat aukot, jotka korvaavat yhden molekyyliä kuvaavan aakkoston merkin tyhjällä merkillä. Jokainen merkkijonoon lisätty aukko vähentää rinnastuksen pistemäärää, mutta joskus niiden käyttö on välttämätöntä maksimaalisen pistemäärän tavoittamiseksi.

Tuimala (2007, s. 28–29) erottelee sekvenssien rinnastusmetodit sen perusteella, onko vertailun kohteena olevia sekvenssejä kaksi vai useampi. Parittaisessa sekvenssirinnastuksessa (engl. *pairwise sequence alignment*) kaksi sekvenssiä rinnastetaan keskenään. Tämä antaa tutkijalle luvun alussa kuvailtua tietoa sekvenssien samankaltaisuudesta ja mahdollisista mutaatioista. Peräkkäisiä parittaisia rinnastuksia käytetään usein esimerkiksi silloin, kun haetaan samankaltaisia sekvenssejä niitä sisältävästä tietokannasta (Selzer, Marhöfer ja Rohwer 2008, s. 42). Usean sekvenssin rinnastuksessa (engl. *multiple sequence alignment*) tarkastelun kohteena on useampi sekvenssi.

Vaikka samaan tulokseen usean sekvenssin samankaltaisuusasteen selvittämisessä päästäisiin myös peräkkäisillä parittaisilla rinnastuksilla, voidaan usean sekvenssin rinnastuksella saada tietoa tietyistä sekvenssien ominaisuuksista parittaisia rinnastuksia yksinkertaisemmin. Tarkastelemalla useampaa sekvenssiä samanaikaisesti voidaan helpommin hahmottaa esimerkiksi proteiinisekvenssien konservoituneita alueita, jotka luonnehtivat kokonaisten proteiiniperheiden proteiinien välistä sukulaisuutta. Konservoituneilla alueilla tarkoitetaan usein proteiinisekvenssien toiminnallisia alueita, joita vertaamalla proteiinin kolmiulotteiseen rakenteeseen voidaan saada lisää informaatiota proteiinin toiminnasta. Myös eri yksilöiden saman geenin sisäiselle vaihtelulle alttiiden kohtien tarkastelu on mielekkäämpää usean sekvenssin samanaikaisella rinnastamisella. Tällöin mielenkiinnon kohteena on usein yksittäisen nukleotidin vaihtelu geenisekvenssien välillä, tai tauteja aiheuttavien mutaatioiden löytäminen. Kolmas Tuimalan antama sovellutus usean sekvenssin rinnastukselle on sen keskeinen asema geenien evoluutiohistoriaa tutkivassa molekyylisystematiikassa.

Sung (2009, 30, 39) jakaa rinnastusalgoritmit kolmeen luokkaan sen perusteella, miten vertailtavien sekvenssien samankaltaisuus määritellään. Kokonaisrinnastuksessa (engl. *global alignment*) sekvenssien samankaltaisuutta tarkastellaan niiden koko pituudelta. Paikallis-

rinnastus (engl. *local alignment*) etsii sekvenssin sisäisesti samankaltaisia osamerkkijonoja (engl. *substring*). Osittaisrinnastus (engl. *semi-global alignment*) tarkastelee merkkijonoja koko pituudeltaan kuten kokonaisrinnastus, mutta ei ota huomioon sellaisten aukkojen pistesakkoa, jotka ovat asetettu ennen sekvenssin ensimmäistä merkkiä, tai viimeisen merkin jälkeen (Sung 2009, s. 41; Setubal, Meidanis ja Setubal-Meidanis 1997, s. 56).

3 Needleman–Wunsch-algoritmi

Vuonna 1970 Saul B. Needleman sekä Christian D. Wunsch kehittivät ensimmäisen dynaamista ohjelmointia hyödyntävän kokonaisrinnastusalgoritmin parittaisen sekvenssien rinnastukseen (Needleman ja Wunsch 1970). Gauthierin ym. (2018, s. 4) mukaan aikaisemmat rinnastusmetodit soveltuivat lähinnä lähes samanlaisten ja samanpituisten proteiinien rinnastukseen. Siinä missä vanhat rinnastusmetodit osoittautuivat epäkäytännöllisiksi, tai tuottivat virheellisiä tuloksia, dynaamista optimointia hyödyntävä Needleman–Wunsch-algoritmi pärjäsi paremmin. Dynaamista optimointia tarkastellaan tarkemmin luvussa 4.

Tutkielman luvussa 2 käsiteltiin biologista dataa sisältäviä tietokantoja, sekä datan keskinäiseen vertailuun ja tietokantakyselyihin käytettyjä rinnastualgoritmeja. Kun rinnastusalgoritmien merkitys sekä käyttökohteet tiedostetaan, voidaan syventyä tarkastelemaan yksittäisen rinnastusalgoritmin toimintaa. Tässä luvussa kuvataan vaiheittain, kuinka Needleman–Wunsch-algoritmi toimii, ja arvioidaan sen aikavaativuutta. Luvussa 4 algoritmia tullaan käyttämään vertailukohteena muihin bioinformatiikassa käytettyihin rinnastusalgoritmeihin.

3.1 Pistetaulukon alustaminen

Oletetaan kaksi sekvenssiä esittävää merkkijonoa *S[1..n]* ja *T[1..m]*, joissa n ja m ovat kussakin merkkijonossa olevien merkkien määriä. Olkoon *V(i,j)* optimaalinen pistetulos merkkijonojen S ja T merkkien *S[1..i]* ja *T[1..j]* rinnastukselle, jossa i ja j esittävät merkkijonojen vastaavien merkkien indeksejä. Parittaisen rinnastuksen tapauksessa rinnastettavat merkkijonot ja yksittäisten merkkien väliset optimaaliset pistetulokset on yksinkertaisinta kuvata kaksiulotteisen taulukon muodossa. Needleman–Wunsch-algoritmissa kaksi sekvenssiä asetetaan taulukon ensimmäiselle pysty- ja vaakariveille vastakkain siten, että yksi sekvenssin kirjain vastaa yhtä taulukon solua (Tuimala 2007, s. 73). Taulukon yhdessä solussa on kyseisestä rivistä ja sarakkeesta löytyvien merkkien rinnastuksen tuottama pistemäärä. Rekursiivinen kaava käsittää kaksi tapausta: joko (1) $i = 0$ tai $j = 0$, ja (2) molemmat $i > 0$ ja $j > 0$. Ensimmäisessä tapauksessa, kun $i = 0$ tai $j = 0$, kaksi tyhjää merkkijonoa rinnastetaan toistensa kanssa. Täten kolme kaavaa määrittää taulukon ensimmäisen rivin, sarakkeen sekä

solun *V(0,0)* seuraavasti:

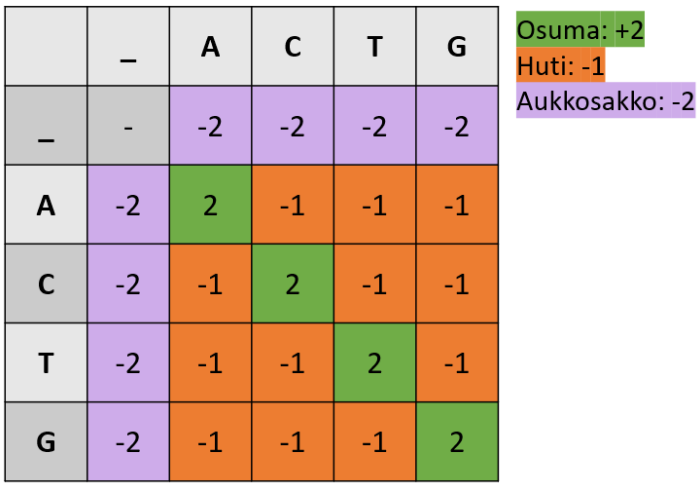
$$V(0,0) = 0, \tag{3.1}$$

$$V(0,j) = V(0,j-1) + \delta(\_, T[j]), \text{ toistetaan arvoilla } j\text{=}1,\ldots,m, \tag{3.2}$$

$$V(i,0) = V(i-1,0) + \delta(S[i], \_), \text{ toistetaan arvoilla } i\text{=}1,\ldots,n. \tag{3.3}$$

Kaavat (3.1)–(3.3) täyttävät pisteytystaulukon ensimmäisen rivin sekä sarakkeen kumulatiivisesti kasvavalla sakolla. Sakko syntyy aukon asettamisesta toiseen sekvensseistä. Lisäksi solu *V(i,j)* asetetaan nollaksi. Näiden toimenpiteden johdosta algoritmin palauttama pistemäärä alustuu alussa nollaksi, ja mahdolliset aukkosakot pystytään ottamaan huomioon jo sekvenssien aluista alkaen.

Parittaisen rinnastuksen tapauksessa ensimmäiseen sekvenssiin asetettavaa aukkoa kutsutaan lisäykseksi (engl. *insert*) ja toiseen sekvenssiin asetettavaa aukkoa poistumaksi, tai deleetioksi (engl. *deletion*) (Sung 2009, s. 31). Jos aukkoa ei aseteta, kahden merkin rinnastuspari voi olla osuma (engl. *match*), eli merkit ovat samat, tai muussa tapauksessa pari on huti (engl. *mismatch*). Kahden rinnastettavan merkin pistemäärä saadaan pisteytysmatriisista (engl. *scoring matrix*) δ. Esimerkiksi symbolien x ja y rinnastuksen pistemäärä esitetään $\delta(x,y)$. Kuten itse rinnastettavat sekvenssit, myös yksittäisten merkkien rinnastusta kuvaava pistematriisi esitetään usein kaksiulotteisen taulukon muodossa. Pisteytysmatriisissa jokainen solu kuvaa sen riviä ja saraketta vastaavan merkin rinnastuksen tuottamaa pistelisäystä tai -sakkoa. Yksinkertainen esimerkki pisteytysmatriisista esitetään kuviossa 1. Kuviosta huomataan, että samat merkit rinnastettuna vastakkain (vihreä) tuottavat positiivisen pisteluvun, hudin (oranssi) ja aukon (violetti) tapauksessa negatiivisen. Selzerin ym. (Selzer, Marhöfer ja Rohwer 2008, s. 39) mukaan aminohapposekvenssejä rinnastaessa jotkut muutokset sekvenssissä evoluution myötä ovat harvinaisempia kuin toiset. Tähän tarkoitukseen voidaan käyttää pisteytysmatriiseja, jotka ottavat huomioon todennäköisyyden kunkin aminohapon vaihtumiseen toiseksi evoluution myötä. Useimmiten käytettävät aminohappojen rinnastuksessa käytettävät pisteytysmatriisit ovat PAM (engl. *position accepted mutation*)- ja BLOSUM (engl. *blocks substitution matrix*)-pisteytysmatriisit.



Kuvio 1. Esimerkki pisteytysmatriisista.

3.2 Pistetaulukon täyttäminen

Rekursiivisen kaavan toisessa tapauksessa, kun $i > 0$ ja $j > 0$, täytetään loput pistetaulukon V solut. Tiedetään, että kahden merkin *S[1..i]* ja *T[1..j]* rinnastus voi olla joko osuma/huti, tai toiseen sekvenssiin asetettava aukko. Yksittäisen solun parhaaksi pistetulokseksi *V(i,j)* valitaan näiden kolmen vaihtoehdon maksimi alla esitetyn kaavan mukaisesti:

$$V(i,j) = max \begin{cases} V(i-1,j-1) + \delta(S[i],T[j]), \text{ jos osuma/huti,} \\ V(i-1,j) + \delta(S[i],\_), \text{ jos poistuma,} \\ V(i,j-1) + \delta(\_,T[j]), \text{ jos lisäys.} \end{cases} \tag{3.4}$$

Soluun *V(i,j)* valitaan siis suurin mahdollinen pistemäärä seuraavista kolmesta vaihtoehdosta:

1. Solun *V(i-1, j-1)* pistemäärän ja solun *V(i,j)* riviä ja saraketta vastaavan merkin rinnastuksen tuottaman pistemäärän summa.

2. Solun *V(i-1, j)* pistemäärän ja solun *V(i,j)* saraketta vastaavan merkin ja tyhjän merkin rinnastuksen tuottaman pistemäärän summa.

3. Solun *V(i, j-1)* pistemäärän ja solun *V(i,j)* riviä vastaavan merkin ja tyhjän merkin rinnastuksen tuottaman pistemäärän summa.

Pistetaulukon *V(1..n, 1..m)* solut täytetään rivi riviltä käyttämällä edellä mainittua rekursiivista yhtälöä. Merkkijonojen *S[1..n]* ja *T[1..m]* rinnastuksen paras tulos löytyy taulukon viimeiseksi täytettävästä solusta *V(n,m)*. Tässä vaiheessa selvittämättä on kuitenkin vielä koko sekvenssien rinnastus, joka johtaa parhaaseen pistetulokseen. Taulukon soluja täytettäessä pidetään kirjaa niistä soluista, joiden läpi parhaan pistetuloksen perässä edettiin (Tuimala 2007, s. 74). Viimeiseksi tapahtuu niin kutsuttu takaisinjäljitys (engl. *traceback*), jonka avulla selvitetään kahden sekvenssin optimaalinen rinnastus.

3.3 Takaisinjäljitys

Takaisinjäljitysvaiheessa taulukossa liikutaan oikeasta alanurkasta takaisin vasempaan yläkulmaan (Tuimala 2007, s. 74). Sekvenssien optimaaliset rinnastukset muodostuvat liikkuessa taulukon läpi. Oikean alanurkan solusta (Sungin esimerkissä solu *V(n,m)*) siirrytään vaaka-, pysty- tai diagonaalisuunnassa kohti vasenta ylänurkkaa siten, että valitaan solut, joista solun *V(i,j)* pistemäärä muodostui. Tätä toistetaan, kunnes taulukon vasen yläkulma on saavutettu. Jos taulukossa liikutaan diagonaalisesti, ovat solun riviä ja saraketta vastaavat merkit osuma tai huti (Tuimala 2007, s. 74) riippuen kyseisen solun riviä ja saraketta vastaavista merkeistä. Vaaka- tai pystysuunnassa liikkuessa toiseen sekvenssiin asetetaan aukko.

Esimerkki täytetystä pistetaulukosta, jossa on rinnastettu merkkijonot $S = AACTC$ ja $T = ACAC$, löytyy kuviosta 2. Kuviossa kuhunkin soluun osoittavilla harmailla nuolilla on kuvattu, mistä solusta tähän soluun on siirrytty. Jos soluun vie useampi kuin yksi harmaa nuoli, on suurin mahdollinen pistemäärä saatu useammasta kuin yhdestä ympäröivästä solusta. Punaisilla nuolilla on kuvattu takaisinjäljityksen reitti taulukon oikean alakulman solusta *V(n,m)*

takaisin soluun *V(0,0)*. Takaisinjäljityksen luomat kaksi optimaalisinta rinnastusta ovat

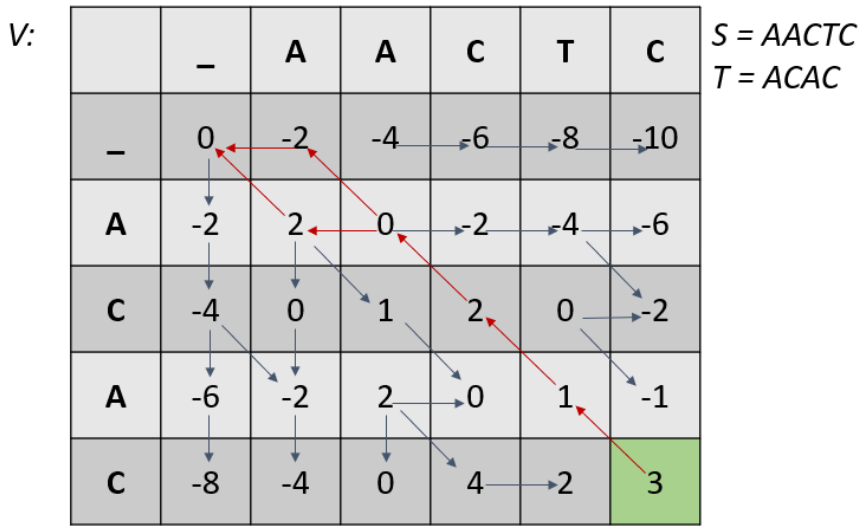
```
AACTC
  **|*
_ACAC
```

ja

```
AACTC
*  *|*
A_CAC
```

joissa merkkijonon *S* ensimmäisen tai toisen merkin kohdalle on asetettu aukko. Edellä olevassa rinnastuksessa merkkien välistä osumaa merkitään *-merkillä, hutia |-merkillä ja aukkoa tyhjällä merkillä siten, että toiseen sekvenssiin on asetettu _-merkki aukkoa vasten. Pisteytysmatriisina on käytetty kuvion 1 matriisia.

Sung (2009, s. 34) huomauttaa, että vaikka lopullinen rinnastus on optimaalinen, ei se välttämättä ole ainoa mahdollinen parhaan pistemäärän tuottava rinnastus. Tuimala (2007, s. 74) havannoi tätä tilanteella, jossa algoritmi voisi edetä takaisinjäljityksen yhteydessä useam-



Kuvio 2. Needleman–Wunschin rinnastamien sekvenssien muodostama pistetaulukko *V* ja optimaaliset rinnastukset.

paan yhtä suuren pistemäärän tuottavaan suuntaa, valitsee se seuraavan solun mielivaltaisesti. Tämä tarkoittaa, että optimaalisia rinnastuksia voi olla useita kappaleita, vaikka algoritmi palauttaisikin vain yhden rinnastuksen. Tarvittaessa algoritmia voidaan suhteellisen yksinkertaisesti modifioida muotoon, jonka avulla voidaan kuvata kaikki mahdolliset optimaaliset rinnastukset. Takaisinjäljitysvaihetta voidaan toistaa useita kertoja, valiten aina eri reitti tilanteessa, jossa useammasta kuin yhdestä ympäröivästä solusta muodostuu solun pistemäärä. Kuviosta 2 aikaisemmin huomattiin, että optimaalisen rinnastuksen tuottaa useampi kuin yksi rinnastus, riippuen siirrytäänkö solusta *V(2,1)* kohti vasenta ylänurkkaa vaaka- vai diagonaalisuunnassa.

3.4 Aikavaativuus

Merkkijonojen *S[1..n]* ja *T[1..m]* rinnastus raakaa voimaa käyttäviä algoritmeja hyödyntäen kävisi läpi kaikki mahdolliset vaihtoehdot erilaisille rinnastuksille, ja palauttaisi parhaan pistetuloksen antavan rinnastuksen (Sung 2009, 32). Tehtävän koko kasvaa tätä menetelmää käytettäessä eksponentiaalisesti, eikä rinnastuksen laskeminen edes kahden merkkijonon välille ole mielekästä, kun tarkasteltavien merkkijonojen pituudet kasvavat. Needleman–Wunsch-algoritmissa pistetaulukon täyttäminen tapahtuu aikavaativuudella *O(nm)*, jossa *n* ja *m* ovat rinnastettavien merkkijonojen pituudet. Takaisinjäljitysvaihe suoriutuu optimaalisen rinnastuksen muodostamisesta pahimmillaan aikavaativuudella *O(n+m)*. Täten Needleman–Wunsch-algoritmin dynaaminen lähestymistapa rinnastusongelmaan mahdollistaa merkkijonojen *S* ja *T* rinnastuksen aikavaativuudella *O(nm)*. Sung (2009, s. 35) esittää myös ratkaisuja algoritmin tilavaatimuksien pienentämiseen *O(nm)* aina *O(m)* asti, mutta nykytietokoneiden tallennustilan osalta algoritmin vaatima tila tuskin osoittautuisi pullonkaulaksi sekvenssien rinnastuksessa.

Vaikka aikanaan Needleman–Wunsch algoritmi on ollut uraauurtavassa asemassa rinnastusongelmien ratkaisemisessa, täytyy ottaa huomioon, että nykyään rinnastuksessa käsiteltävät sekvenssit saattavat olla miljardien merkkien pituisia. Selzer (2008, s. 45) huomauttaa, että alkuperäisen Needleman–Wunsch-algoritmin dynaaminen lähestymistapa rinnastukseen on liian aikaa vievä suurissa tietokantahauissa. Sung (2009, s. 34) havainnollistaa tätä esimerkillä, jossa ihmisen ja hiiren koko genomien rinnastaminen Needleman–Wunsch-algoritmilla

veisi kellotaajuudeltaan yhden gigahertsin prosessorilta satoja vuosia. Hänen mukaansa kuitenkin vuonna 1980 esiteltiin *O(nm / log n)* aikavaativuudella suoriutuva vastaava algoritmi, joka pohjautuu ns. Four Russians -paradigmaan.

4 Rinnastusalgoritmin valinta

Erityyppiset rinnastusalgoritmit suoriutuvat toisia paremmin eri tilanteissa. Luvussa 2 kerrottiin, kuinka algoritmi voidaan valita esimerkiksi sen perusteella, onko kyseessä kahden vai useamman sekvenssin samanaikainen rinnastus. Valintaan vaikuttaa myös se, onko mielenkiinnon kohteena tarkastella sekvenssejä koko pituudelta, vai etsiä ainoastaan samankaltaisia osamerkkijonoja. Näiden lisäksi oikeanlaisen algoritmin valintaan voidaan ajatella vaikuttavan käytettävissä oleva aika, laskentateho ja vaatimukset rinnastuksen optimaalisuuden tarkkuudelle. Rinnastusalgoritmilla on sille tunnusomaiset laskenta-aikaan vaikuttavat ominaisuudet, ja molekyylibiologiset ongelmat, joiden ratkaisemiseen algoritmi soveltuu. Luvussa 3 tarkastelun alla olleen Needleman–Wunsch-algoritmin tunteminen antaa hyvät edellytykset pohtia rinnastusalgoritmin valintaa, ja algoritmi toimiikin osittain vertailukohteena muille tässä luvussa mainituille algoritmeille. Tässä luvussa esitellään kaksi toisistaan poikkeavaa lähestymistapaa rinnastusongelman ratkaisemiseen, jotka ovat dynaamiset algoritmit ja heuristiset algoritmit.

4.1 Dynaamiset algoritmit

Dynaamiset algoritmit nimensä mukaisesti hyödyntävät niin kutsuttua dynaamista optimointia. Dynaamisen optimoinnin perusperiaatteena on, että suuri ongelma jaetaan useiksi pieniksi ongelmiksi, jotka ratkaistaessa alkuperäinen suurempikin ongelma saadaan ratkaistua (Tuimala 2007, s. 25). Ongelmissa, jotka voidaan tällä tavalla jakaa erikseen ratkaistaviin osaongelmiin, voidaan Tuimalan (2007, s. 26) mukaan helposti hyödyntää rinnakkaislaskentaa.

Luvussa 3 esitelty Needleman–Wunsch-algoritmi hyödyntää dynaamista optimointia. Setubalin ym. (Setubal, Meidanis ja Setubal-Meidanis 1997, s. 40) mukaan Needleman–Wunsch soveltuu hyvin ainoastaan kahdelle lähes samanpituiselle ja samankaltaiselle sekvenssille. Tämän lisäksi Needleman–Wunschia ei voida käyttää paikallisten (engl. *local*) samankaltaisuuksien etsimiseen (Chan 2007, s. 1). Tähän tarkoitukseen soveltuu toinen dynaamista optimointia hyödyntävä Smith–Waterman-paikallisrinnastusalgoritmi. Rekursiivisesti täytettä-

vän pistetaulukon solujen pisteytys tapahtuu seuraavasti:

$$V(i,j) = max \begin{cases} 0 \\ V(i-1, j-1) + \delta(S[i], T[j]), \text{ jos osuma/huti,} \\ V(i-1, j) + \delta(S[i], \_), \text{ jos poistuma,} \\ V(i, j-1) + \delta(\_, T[j]), \text{ jos lisäys.} \end{cases} \tag{4.1}$$

Kaava eroaa Needleman–Wunsch-algoritmin vastaavasta, pistetaulukon rekursiivisesti täyttävästä kaavasta (3.4) vain siltä osin, ettei soluihin voi tulla negatiivisia arvoja (Xiong 2006, s. 40). Takaisinjäljitysvaiheessa taulukon suurimman pistemäärän sisältävä alkio voi olla muuallakin kuin taulukon oikeassa alanurkassa, ja algoritmi palauttaa rinnastuksen heti kohdatessaan ensimmäisen solun, jonka pistemäärä on 0. Tämä tarkoittaa sitä, että toisin kuin Needleman–Wunsch-algoritmi, Smith–Waterman-algoritmi voi palauttaa myös parhaan rinnastustuloksen antavan osamerkkijonon (engl. *substring*) koko sekvenssin pituudelta.

Dynaamista ohjelmointia hyödyntävät algoritmit, kuten Needleman–Wunsch ja Smith–Waterman löytävät aina varmasti optimaalisimman rinnastuksen pisteytyskriteerien perusteella (Tuimala 2007, s. 73). Dynaamisen algoritmin valitseminen on sopivaa, kun halutaan tarkkoja tuloksia, tai kun nopeus ei ole ensisijainen tavoite.

4.2 Heuristiset algoritmit

Selzerin (Selzer, Marhöfer ja Rohwer 2008, s. 42) mukaan vanhemmat, dynaamista ohjelmointia hyödyntävät rinnastusalgoritmit kuten Smith–Waterman ja luvussa 3 esitelty Needleman–Wunsch ovat jopa nykytietokoneilla aivan liian aikavaativia suoriutumaan suurista tietokantahauista. Tähän soveltuvatkin hänen mukaansa paremmin heuristiset algoritmit, kuten Basic Local Alignment Search Tool (BLAST). 1990-luvulla kehitetty BLAST on nykyisin käytetyin sekvenssihakualgoritmi (Tuimala 2007, s. 82).

Heuristiset algoritmit suoriutuvat sekvenssien rinnastusta vaativista tietokantahauista nopeammin, sillä ne käsittelevät vain murto-osan rinnastuksista, joita vastaavat dynaamiset algoritmit käsittelevät. Vaikka heuristiset algoritmit kuten FASTA ja aiemmin mainittu BLAST

suoriutuvat tietokantahauista jopa 50-100 kertaa dynaamisia algoritmeja nopeammin, ne eivät välttämättä löydä sekvenssien optimaalista rinnastusta (Xiong 2006, s. 52). Xiongin (Xiong 2006, s. 65) mukaan BLAST voi arviolta jättää huomioimatta jopa 30% merkitsevästi optimaalisista rinnastuksista. Toisin sanoen se, minkä heuristinen algoritmi voittaa ajassa, häviää sen tulosten tarkkuudessa.

5 Yhteenveto

Tarkasteltaessa biologisten tietokantojen, biologisen datan ja bioinformatiikkaa käsittelevien kirjallisten lähteiden määrän kasvua vuosien saatossa, voidaan todeta bioinformatiikan poikineen paljon kiinnostusta usean tieteenalan keskuudessa. Tietokoneiden laskennallisen tehon ja käytettävissä olevan muistin määrän kasvu on mahdollistanut nykyaikaisten valtavien tietokantojen sisältävän datan käsittelyn entistä mielekkäämmin ja kustannustehokkaammin. Myös biologisten sekvenssien rinnastusalgoritmit ovat olleet keskeisessä asemassa alalla siitä asti, kun sekvenssien vertailua on ollut mielekästä tehdä tietokoneavusteisesti.

Nykyään laskennallisen biologian asettamat vaatimukset rinnastusalgoritmeille liittyvät lähinnä samaan aikaan rinnastettavien sekvenssien lukumäärään ja aikavaativuuden minimointiin. Kuten luvussa 4 todettiin, Needleman–Wunsch-algoritmin kaltaiset dynaamiset rinnastusalgoritmit ovat liian aikavaativia, kun rinnastusalgoritmeja halutaan hyödyntää suurissa tietokantahauissa. Heuristiset rinnastusalgoritmit, kuten BLAST ja FASTA, tai niiden variaatiot, ovat vakiinnuttaneet asemansa tietokantojen hakuoperaatioiden suorittajina. Dynaamisen algoritmin valinta voi kuitenkin olla perusteltua, jos ei haluta joustaa rinnastuksen optimaalisuudessa, eivätkä rinnastettavien merkkijonojen pituus tai niiden lukumäärä pääse kasvamaan liian suuriksi.

Optimaalisen rinnastusalgoritmin valinnan lisäksi tietokantahakuja ja tietojenkäsittelyn mielekkyyttä voitaisiin tehostaa myös muilla keinoilla. Olemassaolevien tietokantojen integroimisella, sekä tietokantariippumattomien tietotyyppien ja tietokantahakualgoritmien kehittämisellä voitaisiin keventää työmäärää, jota käyttäjät näkevät nykyään käsiteltäessä tietoa usealla eri alustalla.

Lähteet

Whitfield, Eleanor J, Manuela Pruess ja Rolf Apweiler. 2006. “Bioinformatics database infrastructure for biotechnology research”. *Journal of biotechnology* 124 (4): 629–639.

Chan, Alexander. 2007. *An analysis of pairwise sequence alignment algorithm complexities: Needleman-wunsch, smith-waterman, fasta, blast and gapped blast.*

Cook, Charles E, Mary Todd Bergman, Robert D Finn, Guy Cochrane, Ewan Birney ja Rolf Apweiler. 2015. “The European Bioinformatics Institute in 2016: data growth and integration”. *Nucleic acids research* 44 (D1): D20–D26.

Galperin, M. Y. 2007. “The Molecular Biology Database Collection: 2007 update”. *Nucleic Acids Research* 35 (Database). doi:10.1093/nar/gkl1008.

Gauthier, Jeff, Antony T Vincent, Steve J Charette ja Nicolas Derome. 2018. “A brief history of bioinformatics”. *Brief Bioinform:* 1–16.

Goble, Carole, ja Robert Stevens. 2008. “State of the nation in data integration for bioinformatics”. Semantic Mashup of Biomedical Data, *Journal of Biomedical Informatics* 41 (5): 687–693. ISSN: 1532-0464. doi:https://doi.org/10.1016/j.jbi.2008.01.008. http://www.sciencedirect.com/science/article/pii/S1532046408000178.

Hagen, Joel B. 2000. “The origins of bioinformatics”. *Nature Reviews Genetics* 1 (3): 231.

Luscombe, Nicholas M, Dov Greenbaum ja Mark Gerstein. 2001. “What is bioinformatics? A proposed definition and overview of the field”. *Methods of information in medicine* 40 (04): 346–358.

Needleman, Saul B, ja Christian D Wunsch. 1970. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. *Journal of molecular biology* 48 (3): 443–453.

Ramsden, Jeremy. 2015. *Bioinformatics: an introduction.* Nide 21. Springer.

Selzer, Paul M, Richard J Marhöfer ja Andreas Rohwer. 2008. “Applied bioinformatics”. *An introduction–Springer, Verlag, Berlin, Heidelberg, Germany* 260.

Setubal, Joao Carlos, Joao Meidanis ja . Setubal-Meidanis. 1997. *Introduction to computational molecular biology.* 04; QH506, S4. PWS Pub. Boston.

Sung, Wing-Kin. 2009. *Algorithms in bioinformatics: A practical introduction.* CRC Press.

Tuimala, Jarno. 2007. *Bioinformatiikan Perusteet. 2. p.* Tieteen tietotekniikan keskus CSC.

Xiong, Jin. 2006. *Essential bioinformatics.* Cambridge University Press.