Tampereen yliopisto

Lassi Alm

KEINOJA DE NOVO GENOMIN KOKOAMISEN NOPEUTTAMISEEN

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Elokuu 2023

TIIVISTELMÄ

Lassi Alm: Keinoja De novo genomin kokoamisen nopeuttamiseen
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma, tietotekniikka
Elokuu 2023

Eliön koko DNA voi koostua sadoista tuhansista satoihin miljooniin emäspareista ja tätä biologista dataa pystytään lukemaan alati kiihtyvällä nopeudella. Mikroprosessorien laskunopeuden kasvu ei riitä pysymään mukana kehityksessä. Niinpä datan kokoamiseen ja analysoimiseen on kehitettävä nopeampia menetelmiä tulevaisuuden genetiikan sovellusten mahdollistamiseksi. Biologisen datan lukeminen tapahtuu nykyisellä teknologialla osissa, ja alkuperäinen sekvenssi kootaan tietokoneella näistä sadoista tuhansista vain satojen emäsparien mittaisista osista. Graafiteoria on ollut olennaisessa osassa kokoamisen mahdollistamisessa, ja erityisesti niin sanotut de Bruijn -graafit ovat osoittautuneet hyödyllisiksi nopeimmissa kokoamislagoritmeissa.

Tässä tutkielmassa selvitetään genomien sekvensoinnista saatavan datan *de novo* -kokoamiseen liittyviä ongelmia ja esitellään joitakin tietorakenteita ja menetelmiä, joilla kokoamista on saatu nopeutettua. Lisäksi katsotaan, kuinka sekvensointiteknologian kehitys voisi vaikuttaa kokoamiseen lähitulevaisuudessa.

De novo -kokoamisella tarkoitetaan kokoamista ilman mallisekvenssiä. Pienenkin sekvenssin kokoamiseen yksinkertaisimmilla algoritmeilla ja tavallisella tietokoneella kuluisi useita päiviä. Rinnakkainen laskenta ja algoritmien optimointi rinnakkaisille tietokoneille ovat olennaisessa osassa kokoamisen nopeuttamisessa.

Tutkielman aluksi lukijalle selvitetään ongelman luonne ja laajuus sekä sen ratkaisemisen yleiset periaatteet ja käytännöt. Seuraavaksi esitellään kirjallisuudessa ehdotettuja ja toteutettuja menetelmiä kokoamisen nopeuttamiseen. Ennen yhteenvetoa ja johtopäätöksiä tarkastellaan vielä, kuinka uusimman sukupolven sekvensointiteknologia mahdollisesti vaikuttaa datan käsittelytapaan ja -nopeuteen lähitulevaisuudessa.

Tutkielma toteutettiin kirjallisuuskatsauksena, jossa tieteellisen aineiston tietokannoista etsittiin tuoreita julkaisuja kokoamista nopeuttavista algoritmeista ja muista tekniikoista.

Tuloksista nähdään klusterityyppisille tietokoneille optimoitujen de Bruijn -graafeihin perustuvien algoritmien olevan nopeimpia sekvenssin kokoamisessa. Muistin rinnakkaistaminen ja muistioperaatioiden optimointi ovat avainasemassa suorituskyvyn kasvattamisessa. Lisäksi kolmannen sukupolven sekvensointiteknologioiden yleistyessä kokoamistuloksista tulee laadukkaampia ja kokoamiseen on mahdollista löytää uusia nopeampia lähestymistapoja. Graafiteorialla ja de Bruijn -graafeilla on toistaiseksi paikka näissä algoritmeissa, mutta uudet menetelmät saattavat viedä alaa nopeastikin toiseen suuntaan.

Avainsanat: DNA, rinnakkaisuus, algoritmit, bioinformatiikka, de Bruijn -graafi, tietorakenteet

SISÄLLYSLUETTELO

1. JOHDANTO

Eliöiden perimä eli *genomi* koostuu miljoonien nukleotidien rihmoista, joita lukemalla saadaan hyödyllistä tietoa eliön toiminnasta ja sen ominaisuuksien perinnöllisyydestä. Tietoa eliöiden genomien rakennusosista kerätään kiihtyvällä vauhdilla ja sen kokoamiseen, jäsentämiseen ja tutkimiseen on kehitettävä yhä edistyneempiä tietojenkäsittelyn laitteistoja ja ohjelmistoja. Elämän rakennusohjeet sisältäviä nukleiinihappoja, DNA:ta ja RNA:ta, ei toistaiseksi pystytä lukemaan virheettömästi päästä päähän. Genomista luetaan eli *sekvensoidaan* sama sarja nukleotideja useita kertoja lyhyemmissä *jaksoissa* (engl. *read*). Tämän jälkeen raa'asta datasta pyritään kokoamaan laskennallisin menetelmin alkuperäinen nukleotidien järjestys. Uusimmat sekvensointitekniikat tuottavat huimia määriä raakaa dataa lyhyessä ajassa, ja sen kokoamisessa alkuperäiseen järjestykseen kestäisi useita päiviä tavallisella tietokoneella ja yksinkertaisimmilla tietorakenteilla ja algoritmeilla.

Tämän tutkielman tavoite on esitellä sekvensoinnista saatavan datan luonnetta, sen kokoamiseen liittyviä haasteita sekä yleisimpiä kokoamisessa hyödynnettyjä menetelmiä. Lisäksi selvitetään joitakin uusimpia tietojenkäsittelyn tekniikoita, jotka nopeuttavat ja tehostavat sekvenssin kokoamista esimerkiksi rinnakkaislaskennan avulla. Tutkielmassa ei keskitytä kokoamisen täydellisyyteen, vaan pääpaino on menetelmien nopeudessa. Käsitellyt menetelmät saattavat vaihdella kokoamisen laadun kannalta, mutta tulokset ovat käyttökelpoisia.

Tässä tutkielmassa keskitytään erityisesti *de novo* -kokoamiseen, eli sekvenssin kokoamiseen ilman malliesimerkkiä pelkän raakadatan perusteella. Tätä lähestymistapaa tarvitaan erityisesti, jos ei tiedetä mistä näyte on tullut, näyte on otettu aiemmin tuntemattomasta eliöstä tai se eroaa selvästi aiemmin tutkituista yksilöistä. *De novo* -kokoamisen laskennalliset tekniikat eroavat selvästi mallin perusteella tehtävästä kokoamisesta.

Ensin on ymmärrettävä ongelman luonne, eli minkälaista dataa käsitellään ja mistä data tulee. Luvussa 2 esitellään DNA:n rakenne sekä selitetään, miten sekvensointi käytännössä tapahtuu korkealla tasolla. Keskeisiä ongelmia biologisen datan käsittelyssä ovat laskennan rinnakkaistettavuus ja erityisesti yleisesti käytettyjen tietorakenteiden muistinkäyttö. Monissa kokoamisalgoritmeissa hyödynnetään graafiteoriaa ja erityisesti niin sanottuja de Bruijn -graafeja, joita esitellään luvussa 3. Graafien luomisessa tehdään huo-

mattavan paljon muistioperaatioita, jolloin muistin kaistanleveys tulee rajoittavaksi tekijäksi, kun ytimet jakavat saman keskusmuistin. Ongelman ratkaisussa keskitytään siksi muistin hajauttamiseen useille laskentayksiköille sekä muistin tarpeen vähentämiseen. Menetelmiä sekvenssien kokoamisessa ilmenevien ongelmien ratkaisemiseksi esitellään tarkemmin luvussa 4.

Toisen sukupolven sekvensointiteknologiat ja niiden tuottama data on ollut valtavirtaa lähes 20 vuotta, mutta markkinoille on tulossa uusia kolmannen sukupolven sekvensointiteknologioita, jotka lupaavat pidempiä yhtenäisiä lukujaksoja jopa yhtä matalalla virheiden määrällä kuin toisen sukupolven teknologiat. Luvussa 5 tarkastellaan teknologian muutoksen vaikutuksia laskennan vaatimuksille. Tutkielman viimeisessä luvussa vedetään yhteen käsitellyt asiat, verrataan esiteltyjä menetelmiä tehokkuuden ja nopeuden kannalta sekä pohditaan alan lähitulevaisuuden kehitystä.

2. SEKVENSOINTI

Sekvensoinnilla tarkoitetaan DNA- tai RNA-rihman informaatiota koodaavien emästen järjestyksen lukemista. Elämän rakennusohjeina toimivat nukleiinihapot (deoksiribonukleiinihappo eli DNA sekä ribonukleiinihappo eli RNA) koostuvat pitkistä *nukleotidien* rihmoista. Nukleotidit puolestaan koostuvat sokeri- ja fosfaattiosista sekä informaation koodaavista emäsosista. Emäksiä ovat adeniini, sytosiini, guaniini, tymiini sekä urasiili. Nukleotidit lyhennetään yleensä emäksensä englanninkielisen alkukirjaimen mukaan A, C, G, T ja U. Urasiili esiintyy RNA:ssa DNA:n tymiinin vastineena. Nukleotideilla on tietyt vastinparit DNA:n kaksoiskierrerakenteessa tai kun DNA kopioituu RNA:han. Näitä niin sanottuja emäspareja ovat adeniini-tymiini (RNA:ssa urasiili) sekä sytosiini-guaniini. *Kromosomi* on yhtenäinen DNA-rihma. Esimerkiksi yhdessä ihmisen kromosomissa on noin 150 miljoonaa emäsparia. Kromosomien määrä vaihtelee eliöstä riippuen alle kymmenestä useisiin kymmeniin. Eliön kaikkea geneettistä informaatiota kutsutaan kollektiivisesti eliön perimäksi eli *genomiksi*. Ihmisen genomi koostuu normaalisti 23:sta kromosomiparista. Pareista toinen on peritty isältä ja toinen äidiltä. Kokonaisuudessaan kromosomeja on siis 46 ja niissä on yhteensä noin 6,3 miljardia emäsparia.

Ensimmäiset menetelmät DNA:n sekvensointiin kehitettiin 1970-luvulla. Tärkein näistä on *Sanger*-sekvensointimenetelmä, joka on yhä käytössä pienemmissä projekteissa, tai kun sekvensointivirheiden vähäisyys on erityisen tärkeää. Vuosituhannen vaihteessa markkinoille tuli kuitenkin massiiviseen rinnakkaisuuteen perustuvia sekvensointimenetelmiä, joilla sekvenssejä voitiin lukea paljon aiempia menetelmiä nopeammin. Nämä niin sa-

notut toisen sukupolven sekvensointiteknologiat ovat joukko hieman erilaisiin kemioihin sekä nukleotidien tunnistusmenetelmiin perustuvia teknologioita, joille on yhteistä lyhyet lukujaksot (n. 50 - 700 emäsparia) ja suuri rinnakkaisen lukemisen mahdollistama lukunopeus. Samat emäsparit sekvensoidaan useampaan kertaan, jotta jaksoilla olisi limittyviä kohtia ja jotta virheelliset lukujaksot olisi mahdollista tunnistaa. (Goodwin, McPherson ja McCombie 2016)

Joillakin toisen sukupolven sekvensointitekniikoilla on mahdollista sekvensoida pilkottuja DNA-rihmoja molemmista päistä yhtä aikaa. Koko rihmaa ei välttämättä saada luettua, mutta sen kokonaispituus saadaan tietoon ja dataan kirjoitetaan mitkä lukujaksot tulevat samasta rihmasta. Nämä paritetut lukujaksot (engl. *paired-end reads*) ovat hyödyllisiä esimerkiksi kokoamisen viimeisessä vaiheessa, jossa kootut jaksot sijoitetaan oikeaan järjestykseen. (Illumina 2021)

Noin kymmenen viime vuoden aikana on kehitetty kolmannen sukupolven sekvensointiteknologioita, joilla pystytään lukemaan paljon pidempiä yhtenäisiä jaksoja DNA:ta. Aluksi niiden tuloksissa oli huomattavasti enemmän virheitä, mutta uusimmilla teknologioilla päästään jo lähes yhtä vähäisiin virheiden määriin kuin toisen sukupolven menetelmillä. (Wenger et al. 2019)

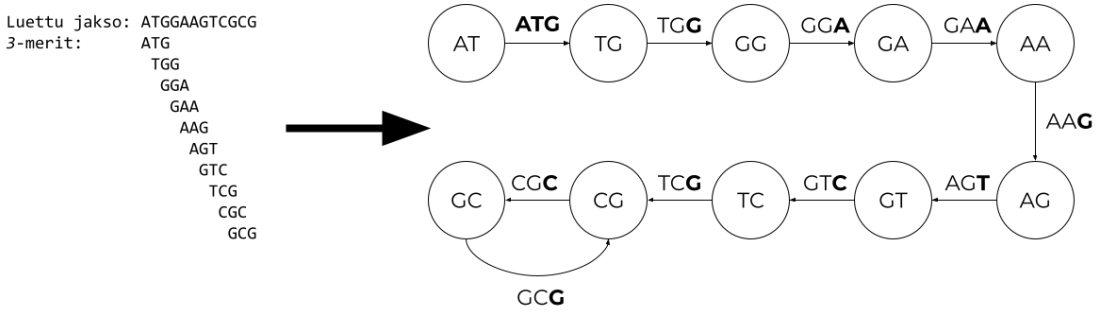
Sekvensoinnin tuloksena on miljoonia tai jopa miljardeja sekaisin olevia jaksoja, joista on laskettava limittyvien osien perusteella alkuperäinen järjestys. Jos jaksoja olisi vain kourallinen, olisi helppoa vain verrata jokaista jaksoa toisiin, löytää limittyvät kohdat ja sitä kautta alkuperäinen järjestys. Jaksojen määrän n kasvaessa vertailuiden määrä kuitenkin kasvaisi nopeudella $O(n*n)$, eli miljoonalla (10^6) jaksolla vertailuita tulisi biljoona (10^{12}), miljardilla (10^9) jaksolla taas kvintiljoona (10^{18}) (Compeau, Pevzner ja Tesler 2011).

Genomin alkuperäisen sekvenssin määrittämiseen kaikista näistä luetuista jaksoista on kehitetty erilaisia toinen toistaan tehokkaampia algoritmeja. Tiedon määrän vuoksi laskentatehoa vaaditaan silti paljon. Luvuissa 3 ja 4 käsitellään yleisimpiä genomin kokoamiseen käytettyjä tietomalleja ja algoritmeja, niiden tuomia laskennallisia haasteita sekä niille ehdotettuja ja toteutettuja ratkaisuja.

3. DE BRUIJN -GRAAFIT

Graafiteoriaan perustuvat menetelmät ovat osoittautuneet tehokkaimmiksi genomin kokoamiseen lyhyistä lukujaksoista. Niin sanottu *de Bruijn -graafi* on yleisin graafin muoto toisen sukupolven sekvensoinnin tuottamien lukujaksojen kokoamiseen suunnitelluissa algoritmeissa. De Bruijn graafi on tietyssä aakkostossa toimiva merkkien päällekkäisyyksiä kuvaava suunnattu graafi. (Khan et al. 2018)

Luetuista jaksoista muodostetaan de Bruijn -graafi pilkkomalla ne edelleen limittyviksi k:n mittaisiksi osajaksoiksi niin, että jokaisen merkin kohdalta alkaa uusi osajakso. Näitä osajaksoja kutsutaan yleensä nimellä $k-mer$. *Mer* tulee kreikan kielen osaa tarkoittavasta sanasta ja k on vakio. Graafi luodaan tallentamalla solmuihin $(k-1)$:n merkin mittaisia jaksoja. Solmuja yhdistää suunnattu kaari, mikäli ne sisältävät alkuperäiseen k-merjoukkoon kuuluvan alkion alku- ja loppuosan ja näin graafin kaarille muodostuu alkuperäinen k-merien joukko. Emästen alkuperäinen järjestys kootaan kulkemalla graafin *Eulerin polku*. Eulerin polku tarkoittaa polkua, jossa käydään graafin jokainen kaari läpi täsmälleen kerran. Jokaisella askeleella sekvenssiin lisätään solmun viimeinen merkki, kunnes graafi on käyty läpi päästä päähän. Erityisesti lyhyillä ja lukuisilla jaksoilla tämä on paljon tehokkaampi tapa koota genomi kuin yksinkertaisin tapa, eli kaikkien luettujen jaksojen vertaaminen toisiinsa limittyvien kohtien löytämiseksi. (Compeau, Pevzner ja Tesler 2011)



Kuva 1. *Esimerkki de Bruijn -graafin rakentamisesta.*

Yksinkertaistettu esimerkki kuvassa 1 havainnollistaa graafin rakentamisen ja läpikäymisen vaiheita. Ensin luettu jakso pilkotaan k-mereiksi, tässä tapauksessa 3-mereiksi. Tyypillisesti vain $(k-1)$:n mittaiset jaksot tallennetaan solmuihin ja k-merit löytyvät graafista vain epäsuorasti solmuja yhdistäviä kaaria tarkastelemalla. Käytännön toteutuksissa

tietorakenne voidaan kuitenkin toteuttaa monilla eri tavoilla. Lopuksi emästen järjestys saadaan käymällä graafi läpi alkaen solmusta, johon osoittavien kaarien lukumäärä on pienempi kuin siitä pois osoittavien. Läpi kuljettaessa sekvenssiin lisättävät merkit on lihavoitu kuvassa. Ideaalissa tilanteessa luetut jaksot limittyvät tarpeeksi, jotta k-merien limitys jatkuu aina seuraavaan jaksoon ja rakennettu graafi kuvaa koko sekvensoitua rihmaa päästä päähän.

De Bruijn graafeihin perustuvassa kokoamisessa vaadittu muistin määrä on paljon suurempi kuin luetuilla jaksoilla tai valmiissa sekvenssissä. Jatkuvan sekvenssin tallentaminen vaatii kaksi bittiä emästä kohden, mutta k-mereihin jaettuna vaadittujen bittien määrä kasvaa. Esimerkiksi 100 emäksen jakso vie 30-mereiksi jaettuna 4200 bittiä k-merien päällekkäisyyden takia. Muistin hakuoperaatiot ovat myös suorittimen kannalta jokseenkin ennalta arvaamattomia, mikä vähentää välimuistien hyödyllisyyttä suurimman osan hakuoperaatioista kohdistuessa keskusmuistiin. (Zhou et al. 2021)

Todellisuudessa sekvensoidussa datassa on yleensä virheitä ja geneettisessä datassa itsessään esiintyy toisteisuutta, eikä graafista saa siksi muodostettua yhtä suoraviivaista graafia. Graafissa on useita epäjatkuvuuskohtia sekä haarautuvia kohtia. Käyttämällä lyhyempiä k-mereja voidaan kasvattaa todennäköisyyttä virheettömien limittyvien kohtien löytämiseen. Toisaalta käyttämällä pidempiä k-mereja löytyy vähemmän samoja k-mereja, jolloin graafia läpikäytäessä siinä on vähemmän silmukoita ja se on yksiselitteisempi. Sopivimman k:n arvon valitseminen on olennainen osa de Bruijn -graafeilla tehtävää kokoamisprosessia. (Georganas et al. 2014)

De Bruijn graafien ja niihin perustuvien graafitietorakenteiden luominen tietokoneella jakautuu yleensä seuraaviin vaiheisiin: k-merien laskeminen, graafin rakentaminen, virheenkorjaus, jatkuvuuskohtien generointi ja *kehystäminen* (engl. *scaffolding*). Luvuissa 3.1-3.4 selitetään tarkemmin nämä vaiheet ja niiden merkitys kokoamisen nopeudelle.

3.1 K-merien generointi

Kokoaminen alkaa jakamalla raaka sekvenssidata lyhyempiin k-mereihin. K-mereista voidaan saada jo tässä vaiheessa hyödyllistä informaatiota esimerkiksi laskemalla esiintymistaajuuksia. K-merien luominen ja laskeminen on yksinkertaista rinnakkaistaa useille ytimille tai prosessoreille jakamalla data tasan laskentayksiköiden kesken esimerkiksi hajautusfunktion avulla.

Esimerkiksi Chapmanin ja muiden (2011) kehittämä *Meraculous* -kokoamisalgoritmi tunnistaa sekvensointivirheitä jo generoimisvaiheessa valitsemalla seuraavaan vaiheeseen vain k-merit, joita esiintyy enemmän kuin ennalta määritetty minimimäärä. Koska sekvensoinnissa genomin jokainen kohta tulee luetuksi useamman kerran, yksittäin esiintyvä k-mer on todennäköisesti seurausta sekvensointivirheestä. Virheiden poistaminen varhai-

sessa vaiheessa vähentää laskentataakkaa seuraavissa vaiheissa, kun graafiin muodostuu vähemmän turhia pisteitä ja haaroja, jotka olisi myöhemmissä vaiheissa poistettava.

3.2 Graafin rakentaminen

Graafin rakentaminen tarkoittaa k-merien kokoamista sopivaan tietorakenteeseen de Bruijn -graafin tai vastaavan mukaisesti niin, että solmujen arvot limittyvät $(k-1)$:n merkin verran. Yleinen tietorakenne on hajautustaulu, jossa k-merit toimivat avaimina, ja arvoksi tallennetaan graafiin liittyvä informaatio, kuten edeltävä ja seuraava nukleotidi. Hajautustaulu mahdollistaa alkioiden haun asymptoottisella suoritusajalla $O(n)$.

Hajautustaulu on kuitenkin muistia vaativa tietorakenne, joten esimerkiksi Liun ja muiden (2011) kehittämä *PASHA*-kokoamisohjelma tallentaa k-merit järjestettyyn vektoriin. Tämä mahdollistaa alkioiden binäärisen haun asymptoottisella suoritusajalla $O(log(n))$. Graafiin liittyvä informaatio tallennetaan k-merien yhteyteen kahdeksalla bitillä, joilla merkitään graafin jatkumista kullakin neljästä nukleotidista molempiin suuntiin. Esimerkiksi, jos graafi jatkuu taaksepäin T-nukleotidilla ja eteenpäin A-nukleotidilla, asetetaan näitä vastaavat bitit ykkösiksi.

3.3 Jatkuvuuskohtien generointi

Ideaalisessa tilanteessa kokoamisohjelman lopputuloksena olisi täydellinen genomi yhtenäisinä kromosomeina. Käytännössä prosessissa esiintyy aina virheitä ja graafissa esiintyy katkoja, joten ohjelma tuottaa tulokseksi jatkuvuuskohdat, joita kutsutaan englanninkielisessä kirjallisuudessa yleensä sanalla *contig* (lyhennys sanasta contiguous). Ohjelma käy läpi edellisessä vaiheessa rakennetun graafin Eulerin polut. Jatkuvuuskohtien lukumäärää käytetäänkin kokoamisen laadun mittarina. Mikäli jatkuvuuskohdat kattavat saman osan koko genomin pituudesta, pienempi jatkuvuuskohtien lukumäärä ja jatkuvuuskohtien pidempi mediaanipituus merkitsevät parempaa kokoamisen laatua.

Ennen lopullista jatkuvuuskohtien generointia kannattaa graafia pelkistää poistamalla esimerkiksi sekvensointivirheistä johtuvia *kärkiä* ja *kuplia*. Kärjet ovat lyhyitä umpikujia ja kuplat taas lyhyitä ylimääräisiä vierekkäisiä polkuja graafissa.

Jatkuvuuskohtien generointia on mahdollista rinnakkaistaa käymällä useita polkuja läpi usealla laskentayksiköllä, eri kohdista ja eri suuntiin. Muistin nopeus ja kaistanleveys tulevat jälleen rajoittaviksi tekijöiksi erityisesti tavallisiin keskusmuistiin ja keskusyksikköön perustuvissa tietokoneissa.

3.4 Kehystäminen

Sekvenssin kokoamisen viimeissä vaiheessa, kehystämisessä (engl. *scaffolding*), määritetään jatkuvuuskohtien oikea järjestys alkuperäisiä luettuja jaksoja hyödyntämällä. Esimerkiksi paritettujen lukujaksojen avulla etsitään kohtia, joissa valmiiden jatkuvuuskohtien päät voidaan yhdistää. Paritetusta lukujaksosta nähdään, mihin järjestykseen jatkuvuuskohdat tulevat ja kuinka pitkä tuntematon jakso niiden väliin jää. Näin jatkuvuuskohdat sijoittuvat oikeille paikoilleen kromosomin alusta loppuun. Vaikka tulokseen jää vielä epätäydellisiä kohtia, koko genomi on kuitenkin oikean muotoinen ja geenit ovat kohdistettuna oikeille paikoilleen, mikä mahdollistaa tuloksen käytön genetiikan tutkimuksissa. (Liu, Schmidt ja Maskell 2011)

4. MENETELMÄT

Toisen sukupolven sekvensoinnin yleistyessä sekvenssin kokoamisen nopeuttamiseen on ehdotettu ja toteutettu useita menetelmiä. Luvussa 3 selitettiin ilmeisimmät tavat rinnakkaistaa graafin ja lopullisen sekvenssin luomista. Tässä luvussa esitellään joitain uusia menetelmiä kokoamisen nopeuttamiseen edelleen.

4.1 Hajautetun muistin järjestelmät

Hajautetun muistin tietokoneessa eli klusterissa voi olla tuhansia laskentayksikköjä omine paikallisine muisteineen. Järjestelmä mahdollistaa yksiköiden eli noodien välisen kommunikoinnin, mutta on tehokkaampaa välttää kommunikointia ja jakaa tehtävät noodeille mahdollisimman itsenäisesti suoritettaviksi. Lopuksi tulokset on toki kerättävä yhtenäiseen tietorakenteeseen.

Useimmat hajautetun muistin järjestelmille suunnitellut kokoamisalgoritmit hyödyntävät *Message Passing Interface (MPI)*-standardia, joka on hajautetun muistin järjestelmien säieturvallisen kommunikaation mahdollistava ohjelmointirajapinta. Sen avulla on mahdollista kirjoittaa erilaisille klusteritietokoneille siirrettävää C- tai Fortran-koodia, joka hyödyntää klusteritietokoneiden rinnakkaisuutta. MPI:n avulla luodaan klusterin noodeille erillisiä MPI-prosesseja, jotka sitten voivat kommunikoida lähettämällä viestejä MPI-metodien avulla. MPI-koodia voidaan suorittaa myös tavallisella jaetun muistin tietokoneella. Lisäksi MPI on suunniteltu toimimaan myös hybriditietokoneissa, jotka voivat koostua erilaisista yhdistelmistä tietokoneita, kunhan niillä on tapa kommunikoida. (Message Passing Inter-

face Forum 2021)

Eräs MPI-protokollaa käyttävä kokoamisohjelmisto on Ghoshin ja muiden (2021) kehittämä PaKman. Se sisältää myös k-mer-pohjaisen geneettisen datan kokoamista varten uuden tietorakenteen: PaK-graafin. Se muistuttaa prosessin alussa jonkin verran de Bruijn -graafia. Tavallisessa de Bruijn -graafissa samat nukleotidit esitetään useita kertoja k-merien limittäisyyden takia. PaK-graafin solmuja pakataan iteratiivisesti niin, että niin sanotuissa makrosolmuissa on aina mukana mielivaltaisen pituiset prefiksi- ja suffiksimerkkijonot. Iteraatioilla solmuja poistuu ja prefiksit ja suffiksit kasvavat, mikä poistaa redundanssia. Tehokkaan rinnakkaisesti suoritetun pakkaamisen jälkeen graafin läpikäyminen valmiiden jaksojen tuottamiseksi on paljon tavallista de Bruijn -graafia tehokkaampaa. Tuloksissa PaKman-algoritmi osoitti hyvää skaalautuvuutta käytettyjen ytimien mukaan ja tutkijat onnistuivat luomaan sen avulla valmiit jatkuvuusjaksot ihmisen genomista 39,1 sekunnissa 16 000 ytimen hajautetun muistin järjestelmällä. PaKman saavutti keskimäärin jopa kaksinkertaisen nopeutuksen muihin suosituimpiin kokoamisalgoritmeihin verrattuna.

4.2 Bloom-suodatin

Bloom-suodatin (Bloom 1970) on todennäköisyyspohjainen tietorakenne, jonka avulla on mahdollista vähentää esimerkiksi k-merien viemää tilaa tavallisiin hajautustauluihin verrattuna. Bloom-suodattimessa joukkoon lisääminen ja joukkoon kuulumisen tarkistus tapahtuvat vakioajassa. Tarkistettaessa joukkoon kuuluvia k-merejä, Bloom-suodattimissa on pieni väärien positiivisten mahdollisuus, mutta joukkoon kuulumattomaksi lajiteltu alkio ei varmasti kuulu joukkoon.

Bloom-suodatin koostuu bittitaulukosta, jonka pituus on m. Tyhjässä joukossa kaikki bitit on asetettu nollaan. Kun joukkoon halutaan lisätä arvo, siitä lasketaan mahdollisesti usealla tiivistefunktiolla tiiviste, joka osoittaa tiettyihin taulukon bitteihin. Funktion osoittamat bitit asetetaan ykkösiksi. Kun halutaan tarkistaa arvon kuuluminen joukkoon, suoritetaan sama tiivisteen laskeminen tarkasteltavalla arvolla. Jos funktioiden osoittamat bitit ovat ykkösiä tarkistus hyväksytään, jos yksikin on nolla, tarkistus hylätään. Tästä seuraa mahdollisuus sille, että funktiot osoittavat taulukossa ykkösiin, vaikka arvoa ei olisikaan alun perin lisätty joukkoon. Väärien positiivisten määrää voidaan vähentää kasvattamalla bittitaulukon kokoa, jotta pienempi osa biteistä täytetyssä taulukossa olisi ykkösiä. Säätämällä bittiavaruuden tilaa, pyritään löytämään sopiva tasapaino taulukon viemän tilan ja väärien positiivisten määrän välillä.

Kun käytetään De Bruijn -graafeja, oikeilla joukkoon kuuluvilla k-mereilla on päällekkäisyyksiä. Niinpä yksi tapa vähentää Bloom-suodattimen antamien väärien positiivisten määrää, on tarkistaa myös tarkasteltavaa k-meria mahdollisesti edeltävät tai seuraavat k-merit. Tarkasteltavasta k-merista otetaan siis joko ensimmäinen tai viimeinen merkki pois

ja lisätään vastaavasti toiseen päähän A, C, T tai G, ja testataan tätä Bloom-suodatinta vastaan. Mikäli de Bruijn -graafin seuraavia tai edeltäviä k-merejä ei löydy, on todennäköistä että alkuperäinen on väärä positiivinen. (Pellow, Filippova ja Kingsford 2017)

Bloom-suodattimista on hyötyä erityisesti k-merien laskentavaiheessa, jossa sopivalla k-merien pituudella uniikit k-merit tarkoittavat yleensä sekvensointivirhettä. Yhden nukleotidin virhe aiheuttaa k virheellistä k-meria, joten k-mereista jopa puolet tai suurempi osa saattaa olla virheellisiä. Kaikki k-merit käydään läpi sekventiaalisesti. Jos k-mer ei löydy Bloom-suodattimesta, se lisätään siihen. Jos k-mer taas löytyy jo Bloom-suodattimesta, se lisätään erilliseen hajautustauluun. Lopuksi hajautustaulussa ovat k-merit, jotka esiintyvät datassa useammin, sekä pieni määrä vääriä positiivisia. (Melsted ja Pritchard 2011)

4.3 Datanläheinen prosessointi

Genomien kokoamiseen käytettyjä algoritmeja rajoittaa nykyaikaisilla tietokoneilla lähinnä muistin määrä ja muistioperaatioiden hitaus. Algoritmeja voisi suorittaa vähintään yhtä nopeasti pienemmälläkin laskentateholla, kunhan muistin määrä ja yhteydet olisivat paremmat.

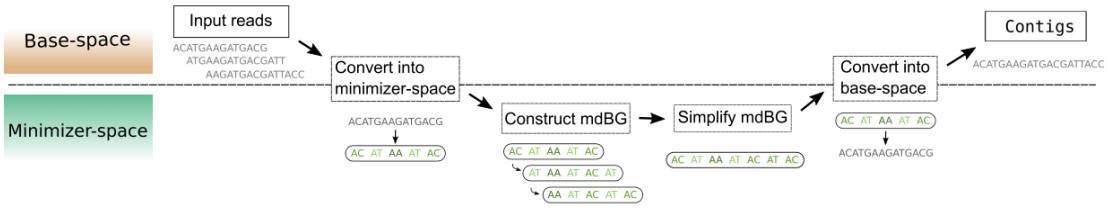
Zhou ja muut (2021) ehdottavat ratkaisuksi *datanläheistä prosessointia* (engl. *near-data processing*, NDP), jossa laitteisto ja ohjelmisto suunnitellaan yhteistoiminnassa hyödyntämään 3D-pinottuja muistipiirejä laskentayksiköinä perinteisen keskusyksikön ja keskusmuistin sijaan. Tutkimuksen esittämä toteutus perustuu Micronin kehittämään *Hybrid Memory Cube* (HMC) -arkkitehtuuriin. HMC-moduuli koostuu 4-8:sta päällekkäisestä muistipiiristä sekä integroidusta logiikkapiiristä, mitkä on yhdistetty toisiinsa *piiläpivienneillä* (engl. *Through-Silicon Via*, TSV). Tämä mahdollistaa kupariliitoksia pienemmän viiveen pinottujen osien välillä. Jokainen moduuli on jaettu vertikaalisiin pinoihin, joissa pinon alin osa on logiikkaydin ja sen päällä ovat muistipiirit. Pinoja voidaan hyödyntää datanläheisinä laskentayksiköinä. Yhdistämällä useita HMC-moduuleja verkkoon nopeilla sarjayhteyksillä olisi mahdollista rakentaa erittäin energiatehokas ja nopea genomin kokoamiseen optimoitu massiivisesti rinnakkainen tietokone. Tutkijat simuloivat tällaista datanläheistä prosessointijärjestelmää sekä muokattua algoritmia ja arvioivat toteutuksen olevan kokonaisuudessaan 24 kertaa nopeampi ja 22 kertaa energiatehokkaampi verrattuna CPU-pohjaiseen järjestelmään.

Micron lakkautti tuen HMC-alustalle vuonna 2018, mutta tutkimuksen tulokset ovat sovellettavissa myös Samsungin suositummaksi osoittautuneeseen *High Bandwidth Memory* (HBM) -arkkitehtuuriin, jota on käytetty muun muassa nVidian ja AMD:n näytönohjainten muistimoduuleissa.

5. KOLMANNEN SUKUPOLVEN SEKVENSOINTI

Kolmannen sukupolven sekvensointiteknologioita yhdistää toiseen sukupolveen verrattuna paljon pidemmät yhtenäiset lukujaksot. Kun toisen sukupolven teknologialle tyypillinen jaksonpituus on alle 300 emäsparia, kolmannen sukupolven teknologialla jaksot voivat olla jopa kymmenientuhansien emäsparien pituisia tai pidempiäkin. Kolmannen sukupolven teknologiat ovat kuitenkin olleet suhteessa kalliimpia ja niitä on vaivannut sekvensointivirheiden suurempi määrä toisen sukupolven teknologioihin verrattuna. Viimeisen kolmen vuoden aikana markkinoille on tullut kuitenkin PacBio:n sekvensointiteknologia CCS (*Circular Consensus Sequencing*) (Wenger et al. 2019), joka lupaa toisen sukupolven teknologioita vastaavaa tai jopa parempaa tarkkuutta 13 500 emäsparin keskimääräisellä lukujaksojen pituudella. Uuden teknologian yleistyessä ja kustannusten madaltuessa on syytä tarkastella olemassaolevien kokoamisalgoritmien soveltuvuutta niiden tuottamalle datalle. Ovatko de Bruijn -graafiin perustuvat algoritmit yhä tehokkampia, vai mahdollistavatko pidemmät lukujaksot jonkin tehokkaamman lähestymistavan?

Ekim ja muut (2021) ovat kirjoittaneet Rust-ohjelmointikielellä kolmannen sukupolven datalle optimoidun algoritmin, joka hyödyntää de Bruijn -graafeja. Lisäksi heidän algoritminsa optimoi datan määrää ja käsittelyä muuttamalla raa'an datan ensin *minimointiavaruuteen* (engl. *minimizer-space*), jossa käytössä on laajempi aakkosto, joka rakentuu emäksien yhdistelmistä. De Bruijn -graafin kokoaminen tapahtuu tehokkaammin minimointiavaruudessa, minkä jälkeen sekvenssi palautetaan *emäsavaruuteen* (engl. *base-space*).



Kuva 2. *Jatkuvuuskohtien generointi minimointiavaruuden avulla. (Ekim, Berger ja Chikhi 2021)*

Kuvassa 2 esitetään, kuinka de Bruijn -graafin luominen tehdään minimointiavaruudessa ja näin jatkuvuuskohtien luomisprosessista saadaan tehokkaampi. Ensin sekvensoinnista saadut lukujaksot muutetaan minimointiavaruuteen valitsemalla tietty joukko sekvenssin pätkiä, jotka koodataan uuden aakkoston merkeiksi. Näitä merkkejä voi olla paljon enem-

män, kuin mitä millään tavallisella aakkostolla voidaan merkitä. Luetut sekvenssit käydään läpi tietyn minimointiaakkosten määrän mittaisella ikkunalla muodostaen minimointiavaruuden k-merien vastineet eli niin sanotut k'-min-merit. Minimointiavaruuden de Bruijn -graafissa on huomattavan paljon vähemmän solmuja kuin vastaavassa emäsavaruuden graafissa, joten graafin luominen helpottuu merkittävästi erityisesti muistin käytön ja muistioperaatioiden kannalta. Graafin luomisen ja sieventämisen jälkeen k'-min-merit muutetaan takaisin emäsavaruuteen alkuperäisistä lukujaksoista tallennettujen tietojen avulla, ja tuloksena on lopulliset jatkuvuuskohdat. Tutkijat onnistuivat menetelmän avulla kokoamaan ihmisen genomin alle kymmenessä minuutissa tavallisella pöytätietokoneella, jossa oli 8-ydinsuoritin ja 10 gigatavua keskusmuistia. Tämä on jopa monta kertaluokkaa nopeampaa, kuin yleisimmät toisen sukupolven sekvensointiteknologialle suunnitellut menetelmät.

Menetelmästä on hyötyä lähinnä kolmannen sukupolven teknologian mahdollistamilla pidemmillä lukujaksoilla, koska lyhyemmillä lukujaksoilla k'-min-merit jäisivät liian lyhyiksi. Silloin graafissa olisi liikaa kohtia, joissa sama k'-min-mer vastaa emäsavaruudessa eri kohtia. Näitä ei ole menetelmällä mahdollista erotella, joten jatkuvuuskohdat jäisivät lyhyemmiksi ja katkonaisemmiksi.

Kuten edellisestä esimerkistä nähdään, pidempien lukujaksojen kokoamiseen voi löytyä yllättäenkin hyvin tehokkaita uusia menetelmiä, jotka nopeuttavat kokoamista merkittävästi. Tällaisia kehitysaskeleita nähdään todennäköisesti lähivuosina useita, kun uusi sekvensointiteknologia yleistyy ja tutkijat kehittävät datan kokoamiseen uusia menetelmiä kiihtyvällä vauhdilla.

6. YHTEENVETO JA JOHTOPÄÄTÖKSET

Varsinkin de Bruijn -graafiin perustuvat menetelmät ovat osoittautuneet tehokkaiksi DNA-sekvenssien *de novo* kokoamisessa. Suurin osa tehokkaimmista kokoamisohjelmistoista hyödyntää niitä jossain muodossa. De Bruijn -graafit kuitenkin vievät paljon muistia ja, kun raakaa dataa on kokoamisessa käsiteltävänä valtavia määriä, keskusmuistin määrä sekä muistioperaatioiden määrä ja viiveet nousevat suurimmiksi rajoittaviksi tekijöiksi prosessin nopeuttamisessa. Muistin käyttöä vähentävät ja muistin rinnakkaisuutta lisäävät teknologiat ja algoritmit voidaankin todeta tehokkaimmiksi kokoamisen nopeuttamisessa edelleen.

Toisen sukupolven sekvensointidatalle tällaisia tekniikoita ovat esimerkiksi luvussa 4 mai-

nitut hajautetun muistin järjestelmät, eli supertietokoneet, tai datanläheinen prosessointi. Molemmissa menetelmissä genomin kokoaminen nopeutuu merkittävästi, mutta datanläheisellä prosessoinnilla on potentiaalia olla energiatehokkaampi ratkaisu, sillä muistin ja muistin kaistanleveys on paremmassa suhteessa laskentayksikön tehoon ongelman luonnetta ajatellen.

Tässä tutkielmassa on käsitelty vain joitakin yleisimpiä kokoamiseen käytettyjä menetelmiä ja niiden olennaisimpia innovaatioita. Viitatuissa tutkimuksissa on käytetty myös useita oheistekniikoita, joilla saattaa olla osaltaan vaikutusta koko ohjelman nopeuteen tai virheiden määrään. Edistyneimmät kokoamisohjelmistot hyödyntävät eri tavoilla yhdisteltyinä useita eri menetelmiä, kuten Bloom-suodattimia sekä hajautetun muistin supertietokonetta ja niille optimoituja algoritmeja. Ohjelmistoissa käytetään myös erilaisia virheenkorjaustyökaluja. Virheellisen datan korjaaminen ennen kokoamista nopeuttaa itsessään kokoamista vähentämällä käsiteltävän tiedon määrää muisti-intensiivisimmässä vaiheessa.

Tämä tutkielma keskittyi enimmäkseen menetelmien nopeuteen, ja joissain tapauksissa menetelmät saattavat tuottaa tuloksiin enemmän virheitä hitaampaan menetelmään verrattuna. Nopeuden ja virheiden määrän suhdetta on useimmissa viitatuissa tutkimuksissa punnittu tarkemmin. Enemmän virheitä tai vähemmän yhtenäisiä tuloksia tuottavat menetelmät voivat kuitenkin olla hyödyllisiä, jos ne ovat merkittävästi nopeampia. Silloin niillä voidaan esimerkiksi nopeasti tarkistaa lähdedatan laatua tai luoda yleiskuva suuresta datamäärästä, jonka täydellisempään kokoamsieen kuluisi muussa tapauksessa liian pitkä aika. Esimerkiksi Ekim ja muut (2021) kokosivat tutkimuksessaan 661 405:n bakteerin genomista koostuvan *pangenomin*, eli laajan lajiryhmän esityksen yhdessä haettavassa graafissa. Kokonaiskuvan luominen lajin tai lajiryhmän geneettisestä variaatiosta nopeuttaa geenien hakua ja näin geenitutkimustyötä yleisesti.

Mikäli tulevaisuuden sekvensointiteknologiat mahdollistavat nopean ja lähes täydellisen DNA:n lukemisen rihman päästä päähän, laskenta saattaa siirtyä pääasiassa sekvenssien relevanttien kohtien hakuun ja vertailemiseen, mitä ei tässä tutkielmassa ole käsitelty. Toistaiseksi vaikuttaa siltä, että graafiteoreettiset menetelmät kuten de Bruijn -graafi tulevat säilyttämään jonkinlaisen aseman kokoamisohjelmistoissa ainakin lähitulevaisuudessa.

LÄHTEET

Bloom, Burton H. (heinäkuu 1970). ”Space/time trade-offs in hash coding with allowable errors”. *Communications of the ACM* 13.7, s. 422–426. DOI: 10.1145/362686.362692.

Chapman, Jarrod A. et al. (elokuu 2011). ”Meraculous: De Novo Genome Assembly with Short Paired-End Reads”. *PLoS ONE* 6.8. Toim. Steven L. Salzberg, e23501. DOI: 10.1371/journal.pone.0023501.

Compeau, Phillip E C, Pavel A Pevzner ja Glenn Tesler (marraskuu 2011). ”How to apply de Bruijn graphs to genome assembly”. *Nature Biotechnology* 29.11, s. 987–991. DOI: 10.1038/nbt.2023.

Ekim, Barış, Bonnie Berger ja Rayan Chikhi (lokakuu 2021). ”Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer”. *Cell Systems* 12.10, 958–968.e6. DOI: 10.1016/j.cels.2021.08.009.

Georganas, Evangelos et al. (marraskuu 2014). ”Parallel De Bruijn Graph Construction and Traversal for De Novo Genome Assembly”. Teoksessa: *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, s. 437–448. DOI: 10.1109/sc.2014.41.

Ghosh, Priyanka, Sriram Krishnamoorthy ja Ananth Kalyanaraman (toukokuu 2021). ”PaKman: A Scalable Algorithm for Generating Genomic Contigs on Distributed Memory Machines”. *IEEE Transactions on Parallel and Distributed Systems* 32.5, s. 1191–1209. DOI: 10.1109/tpds.2020.3043241.

Goodwin, Sara, John D. McPherson ja W. Richard McCombie (toukokuu 2016). ”Coming of age: ten years of next-generation sequencing technologies”. *Nature Reviews Genetics* 17.6, s. 333–351. DOI: 10.1038/nrg.2016.49.

Illumina, Inc. (2021). ”Paired-End vs. Single-Read Sequencing”. Viitattu 31.05.2023. URL: https://www.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html.

Khan, Abdul Rafay et al. (tammikuu 2018). ”A Comprehensive Study of De Novo Genome Assemblers: Current Challenges and Future Prospective”. *Evolutionary Bioinformatics* 14.117693431875865. DOI: 10.1177/1176934318758650.

Liu, Yongchao, Bertil Schmidt ja Douglas L Maskell (elokuu 2011). ”Parallelized short read assembly of large genomes using de Bruijn graphs”. *BMC Bioinformatics* 12.1. DOI: 10.1186/1471-2105-12-354.

Melsted, Páll ja Jonathan K Pritchard (elokuu 2011). ”Efficient counting of k-mers in DNA sequences using a Bloom filter”. *BMC Bioinformatics* 12.1. DOI: 10.1186/1471-2105-12-333.

Message Passing Interface Forum (kesäkuu 2021). *MPI: A Message-Passing Interface Standard Version 4.0*. URL: https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf.

Pellow, David, Darya Filippova ja Carl Kingsford (kesäkuu 2017). ”Improving Bloom Filter Performance on Sequence Data Using ik/i-mer Bloom Filters”. *Journal of Computational Biology* 24.6, s. 547–557. DOI: 10.1089/cmb.2016.0155.

Wenger, Aaron M. et al. (elokuu 2019). ”Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome”. *Nature Biotechnology* 37.10, s. 1155–1162. DOI: 10.1038/s41587-019-0217-9.

Zhou, Minxuan et al. (syyskuu 2021). ”Ultra Efficient Acceleration for De Novo Genome Assembly via Near-Memory Computing”. Teoksessa: *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE, s. 199–212. DOI: 10.1109/pact52795.2021.00022.