Laskentaa DNA- ja RNA-molekyyleillä

Olli Niemitalo, 7.12.2011

Sisällys

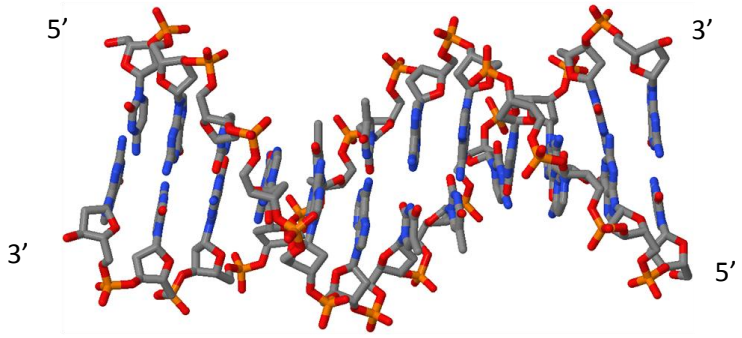
1 Johdanto

Deoksiribonukleiinihappo- (DNA) ja ribonukleiinihappomolekyyleille (RNA) on hahmoteltu käyttötapoja laskuvälineinä. Tämän kirjallisuustutkielman tarkoitus on selvittää, miten DNA- ja RNA-laskennan menetelmät toimivat ja mihin niitä voi käyttää. Tutkimusala lähti liikkeelle toivosta siitä, että molekyylien välisiin vuorovaikutuksiin perustuvat tietokoneet voisivat ratkaista suuria laskennallisia ongelmia sähköisiä tietokoneita tehokkaammin. Nykyisellään tästä ajatuksesta on melkein kokonaan luovuttu, ja sovelluksia haetaan sieltä, missä perinteisiä tietokoneita ei voi käyttää. Tutkimusala onkin pääosin sulautunut synteettiseen biologiaan eli elävien organismien ja niiden osien keinotekoiseen suunnitteluun ja rakentamiseen.

Tässä johdannossa tutustutaan DNA:n ja RNA:n ominaisuuksiin sekä laskennan teoreettisiin perusteisiin, jotka ovat riippumattomia käytännön toteutuksesta ja siten pätevät biomolekyyleihin perustuvaan laskentaan siinä missä sähköisiin tietokoneisiinkin.

1.1 DNA ja RNA

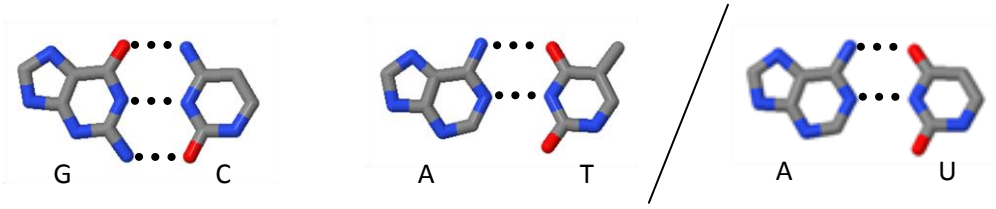
Sekä DNA että RNA ovat haarautumattomia, neljästä vaihtoehtoisesta nukleotidialayksikkötyypistä koostuvia ketjumaisia molekyylejä. Nukleotidialayksiköt eroavat toisistaan emäsryhmän osalta. DNA:n nukleotideissa emäs on joko adeniini (A), guaniini (G), sytosiini (C) tai tymiini (T). RNA:lla tymiinin korvaa urasiili (U), mutta muuten emäsvaihtoehdot ovat samat. Emäsryhmä on liittynyt sokeriryhmään, joka DNA:lla on deoksiriboosi ja RNA:lla riboosi. Peräkkäiset nukleotidit ovat kiinnittyneet toisiinsa sokeriryhmiensä 3’- ja 5’-hiilistä fosfaattiryhmän välityksellä. Sokerin rakenne antaa DNA- ja RNA- molekyyleille suunnan. Normaali tapa ilmaista emäsjärjestys on luetella emäkset 5’→3’-suunnassa. Esimerkiksi emäsjärjestys ATGCAGT kuvaa seitsemän nukleotidin (nt) ketjua, jonka ensimmäisen adeniinin sokeriryhmän 5’-hiili ja viimeisen tymiinin sokeriryhmän 3’-hiili eivät ole kiinnittyneet mihinkään nukleotidiin.



Kuva 1. Kaksoiskierteisen DNA:n rakenne (PDB ID: 1BNA, Drew *et al.* 1981). Kaksoiskierteinen DNA koostuu kahdesta toisiinsa nähden vastakkaissuuntaisesta juosteesta, jotka ovat sitoutuneet toisiinsa emäsparien välisin vetysidoksin.

Yhteen juosteeseen DNA:ta tai RNA:ta voi emäsryhmien välityksellä sitoutua toinen, suunnaltaan päinvastainen juoste (Kuva 1). Kestävin emäspariutuminen tapahtuu komplementaaristen emästen välillä, eli silloin kun toisiinsa sitoutuvien molekyylien emäsjärjestys (5’→3’-suuntaan luettuna) on käänteiskomplementaarinen. DNA:lla komplementaarisia emäspareja ovat kahdella vetysidoksella toisiinsa sitoutuvat A ja T ja kolmella vetysidoksella toisiinsa sitoutuvat C ja G (Kuva 2). Emästen rakenteelliset erot ovat suotuisat erityisesti näiden emäsparien muodostaman kaksoiskierteisen DNA:n vakaudelle. Vetysidosten lisäksi tärkein juosteiden välisen sidosenergian muodostava tekijä on peräkkäisten pariutuneiden aromaattisten emästen pinoutuminen DNA-kaksoiskierteessä (Every & Russu 2007). Merkittävää juosteiden pariutumista tapahtuu, kun käänteiskomplementaarisuus on riittävä ja tarpeeksi yhtenäinen. Myös lämpötilalla on merkitystä. Matalammassa lämpötilassa sitoutumi-

nen on voimakkaampaa, ja riittävän korkeassa lämpötilassa sitoutumista ei tapahtu käytännössä ollenkaan. Samat sitoutumissäännöt koskevat RNA:ta niillä poikkeuksin, että RNA:ssa T:n korvaa U, ja myös G ja U saattavat pariutua (Faulhammer *et al.* 2000). DNA- ja RNA-molekyyli voivat emäspariutumisen säännöin sitoutua myös toisiinsa (Sugimoto *et al.* 2000) – tai itseensä, jolloin syntyvää emäsparien kuvaamaa rakennetta kutsutaan sekundäärirakenteeksi.



Kuva 2. DNA:n (GC ja AT) ja RNA:n (GC ja AU) komplementaariset emäsparit (PDB ID: 1BNA, Drew *et al.* 1981 ja PDB ID: 1RNA, Dock-Bregeon *et al.* 1989). Katkoviiva tarkoittaa vetysidosta.

RNA-virukset pois lukien kaikki tunnetut elämänmuodot käyttävät pitkäaikaiseen perinnöllisen informaation tallentamiseen rakenteeltaan kestävää kaksoiskierteistä DNA:ta. RNA-molekyylien luonnolliset päätehtävät taas ovat proteiinisynteesissä, jossa RNA-polymeraasi kopioi geneettisen DNA:n sisältämän tiedon lähetti-RNA:han (mRNA), josta tieto kopioidaan edelleen RNA:sta (rRNA) koostuvien ribosomien ja siirtäjä-RNA:iden (tRNA, englanniksi transfer RNA) toimesta tuotettavaan proteiiniin aminohappojärjestyksenä. Bakteereilla prosessiin kulkuun vaikuttavat joidenkin geenien osalta säätelevästi pienet RNA:t (sRNA, englanniksi small RNA), jotka sitoutuvat sellaiseen mRNA:han, jossa on emärjärjestykseltään sRNA:han nähden komplementaarinen kohta (Vogel & Wagner 2007). sRNA-molekyylien toimintatapa on esimerkki nukleiinihappojen kyvystä löytää ja tunnistaa toinen nukleiinihappo ja sitoutua siihen emäspariutumisen säännöin. Ribosomien rRNA puolestaan kuuluu entsymaattisesti aktiivisiin RNA-molekyyleihin, eli sellaisiin, joilla on katalyyttisiä ominaisuuksia.

Erilaisille organismeille on kehittynyt runsaasti DNA:ta ja RNA:ta käsitteleviä entsyymejä. Suuri joukko näitä entsyymejä tai niiden johdannaisia on käytössä molekyylibiologian työkaluina mahdollistaen RNA- ja erityisesti DNA-molekyylien käsittelemisen halutulla tavalla. Käytettävissä on moninaisia restriktioentsyymejä, joista kukin tunnistaa ja katkaisee emäsjärjestykseltään tietynlaisen kohdan kaksijuosteisessa DNA:ssa. Kaksijuosteisen DNA:n monistamiseksi voidaan käyttää polymeraasiketjureaktiota (PCR), jossa reaktioseokseen lisätyt DNA-alukkeet kiinnittyvät alkuperäiseen DNA:han tai jo monistuneisiin kopioihin. DNA-polymeraasientsyymi täydentää 5’→3’-suunnassa DNA-juosteen puuttuvan loppuosan niin, että syntyvä juosteen osa on vastinjuosteen kanssa komplementaarinen. Lämpötilaa muuttamalla voidaan reaktion kulkua hallita ja sen vaiheita toistaa. Kun molempia juosteita varten on oma aluke, kaksinkertaistuu alukkeiden rajaaman DNA:n määrä reaktion joka kierroksella. DNA:ta, jolla on haluttu emäsjärjestys, voidaan valmistaa synteettisen kemian menetelmin. Halutuissa kohdissa olevat emäkset voidaan satunnaistaa käyttämällä niiden syntetisoinnissa useampaa kuin yhtä emästä sisältävää reaktioseosta. DNA:ta voidaan kääntää RNA:n muotoon RNA-polymeraasientsyymin avulla. Se rakentaa 5’→3’-järjestyksessä RNA-juosteen, jonka emäsjärjestys on käänteiskomplementaarinen mallina toimivan DNA-juosteen kanssa. Käänteiskopioijaentsyymi taas rakentaa DNA:ta RNA:n pohjalta. RNA:ta katkaisevia entsyymejä kutsutaan ribonukleaaseiksi (RNaasi).

1.2 Boolen algebra

Perinteisen laskuopin keinoin voidaan ilmaista luvuilla tehtyjä laskutoimituksia. Kun lukujen sijaan käytetään Boolen muuttujia eli muuttujia, jotka saavat joko arvon epätosi tai tosi, löytyvät tarvittavat laskutoimitukset Boolen algebrasta (Whitehead 1898). Boolen algebran operaattoreita ovat

muun muassa JA (konjunktio), TAI (disjunktio), ERI, EI-JA ja EI-TAI, jotka ovat konnektiiveja, sekä negaatio EI. Operaattoreiden toiminta voidaan kuvata totuustaulukoin (Taulukko I).

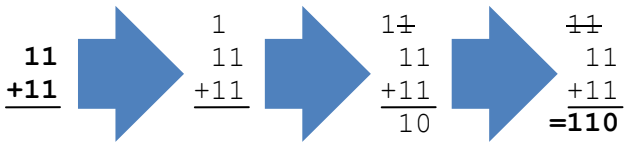
Taulukko I. Yleisesti käytettyjä Boolen algebran operaattoreita kuvaavat totuustaulukot.

Syötteet		**Laskutoimitus ja tulos**					
a	b	a JA b	a TAI b	a ERI b	EI a	a EI-JA b	a EI-TAI b
epätosi	epätosi	epätosi	epätosi	epätosi	tosi	tosi	tosi
epätosi	tosi	epätosi	tosi	tosi	tosi	tosi	epätosi
tosi	epätosi	epätosi	tosi	tosi	epätosi	tosi	epätosi
tosi	tosi	tosi	tosi	epätosi	epätosi	epätosi	epätosi

Yhdistelemällä operaattoreita, muuttujia ja laskujärjestystä ilmaisevia sulkeita voidaan koostaa mikä tahansa Boolen muuttujia käsittelevä laskutoimitus eli lauselogiikan lause. Kaikkia operaattoreita ei tarvita funktionaalisesti täydelliseen Boolen algebraan. Operaattoria EI-JA tai EI-TAI voi käyttää yksistäänkin kaikkien laskutoimitusten ilmaisemiseen (Krutz 1980).

1.3 Binääriluvut

Boolen algebra on sellaisenaan käypä laskutoimituksiin *binäärinumeroilla* (biteillä) eli kaksiarvoisilla luvuilla, kun totuusarvot epätosi ja tosi korvataan lukuarvoilla 0 ja 1. Arvojoukoltaan suurempia lukuja, esimerkiksi kokonaislukuja, voi käsitellä Boolen algebralla kuvaamalla niitä useammalla bitillä. N bitillä voidaan kuvata lukuja, joiden arvojoukon koko on 2^N. Kantaluku, eli peräkkäisten numeroiden merkittävyyksien suhde on binääriluvuilla 2, kun kymmenjärjestelmässä se on 10. Esimerkiksi desimaalijärjestelmän luku $16 = 1 \cdot 10^1 + 6 \cdot 10^0$ on binäärijärjestelmässä $1000_\mathrm{b} = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$, missä alaindeksi b tarkoittaa binäärijärjestelmää. Kuva 3 antaa esimerkin yhteenlaskun suorittamisesta binäärijärjestelmässä.



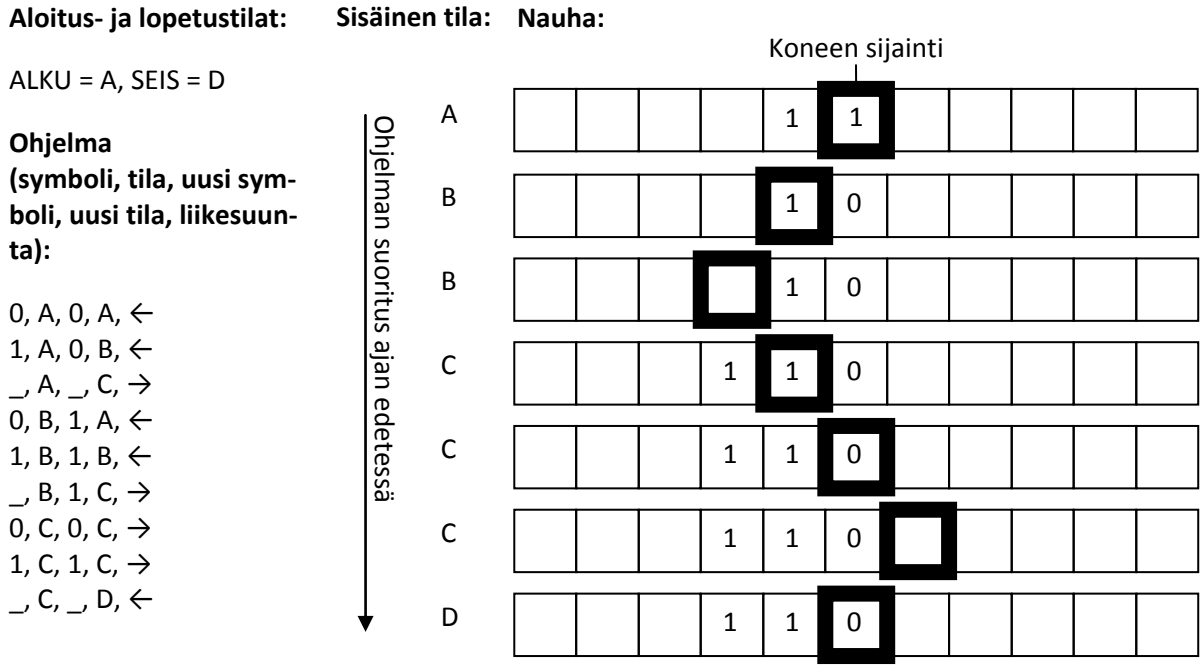
Kuva 3. Binäärilukujen yhteenlasku $11_\mathrm{b} + 11_\mathrm{b} = 110_\mathrm{b}$ allekkain. Binäärinumero pyörähtää ympäri 1:n jälkeen samoin kuin desimaaliluku pyörähtää ympäri 9:n jälkeen, ja yli menneet kahdet (vertaa: kymmenet) kirjoitetaan muistiin laskettavaksi yhteen seuraavan numeron kohdalla.

Boolen algebralla on mahdollista ilmaista mille tahansa binäärilukujen tavalliselle laskutoimitukselle, kuinka vastauksen kukin bitti riippuu syötteen biteistä (Krutz 1980). Koska binäärijärjestelmä ei ole sen heikompi kuin mikään muukaan tavallinen lukujärjestelmä, voidaan lauselogiikkaa pitää universaalina tapana kuvata laskentaa. Nykyaikaisten tietokoneiden toiminta perustuukin Boolen algebran toteuttamiseen mikroelektronisten piirien avulla, joissa totuusarvoja vastaavat komponenttien eri toimintatilat (Krutz 1980).

1.4 Turingin kone

Digitaalisen tietokoneen voi kuvata laitteena, joka ottaa vastaan tietoa syötteenä, suorittaa ohjelmansa määräämät laskutoimitukset ja palauttaa tuloksen. Eräs käsitteellinen malli digitaalisesta tietokoneesta on Alan Turingin kuvitteellinen Turingin kone (Turing 1936). Turingin kone käyttää tiedon lukemiseen ja tallentamiseen nauhaa, jonka pituus on kumpaankin suuntaan ääretön. Nauha koostuu soluista, joista kuhunkin voi kirjoittaa yhden symbolin. Vaihtoehtoisia symboleja on äärellinen määrä.

Alussa kukin solu on alustettu tyhjää ilmaisevalla symbolilla, tai se voi sisältää syötetietoja. Nauhaa lukeva Turingin kone näkee yhden solun kerrallaan. Koneen ohjelma on joukko sääntöjä, joiden mukaan koneen sisäisen tilan ja koneen näkemän symbolin yhdistelmä määrää minkä symbolin kone kirjoittaa nauhalle entisen tilalle, kumpaan suuntaan se liikkuu nauhalla askeleen vai liikkuuko ollenkaan ja mikä tulee koneen seuraavaksi sisäiseksi tilaksi. Mahdollisia sisäisiä tiloja on äärellinen määrä. Ohjelman suoritus alkaa aloitustilasta (ALKU) ja päättyy lopetustilaan (SEIS). Kuva 4 esittää esimerkkiohjelman binääriluvun kertomisesta kahdella sekä ohjelman suorituksen kulun.



Kuva 4. Annetun binääriluvun kahdella kertova Turingin kone. Luku syötetään koneelle nauhalla sillä tavoin, että vähiten merkitsevä numero on koneen näkemällä kohdalla ja vasemmanpuoleinen numero on eniten merkitsevä. Kone kirjoittaa vastauksen syötteen päälle laajentaen lukua tarvittaessa vasemmalle päin ja palaa luvun loppuun. Lopullinen tulos on saavutettu, kun koneen tilaksi tulee D. Ohjelman toimintaperiaate on laskea luku yhteen itsensä kanssa allekkainlaskun tapaan (Kuva 3). Tila B tarkoittaa, että nähtyyn numeroon tulee lisätä sen itsensä lisäksi vielä edellisestä numerosta muistiin jäänyt 1. Ohjelman listauksessa alaviiva tarkoittaa tyhjää ruutua.

Turingin kone ei ota vastaan uutta syötetietoa kesken toimintansa, missä mielessä se ei täysin vastaa toiminnallisuudeltaan nykyaikaista tietokonetta. Kaikessa yksinkertaisuudessaan sen uskotaan kuitenkin olevan kykenevä suorittamaan mitä tahansa laskutoimituksia, joita yleensäkään on mahdollista suorittaa. Tämä väite tunnetaan Chuch-Turingin teesinä, joka on nimetty Alonzo Churchin ja Alan Turingin mukaan. Church julkaisi samoihin aikoihin Turingin kanssa oman, abstraktiin matematiikkaan perustuvan tapansa mallintaa laskentaa, Lambdakalkyylin (Church 1936).

1.5 Tilakone

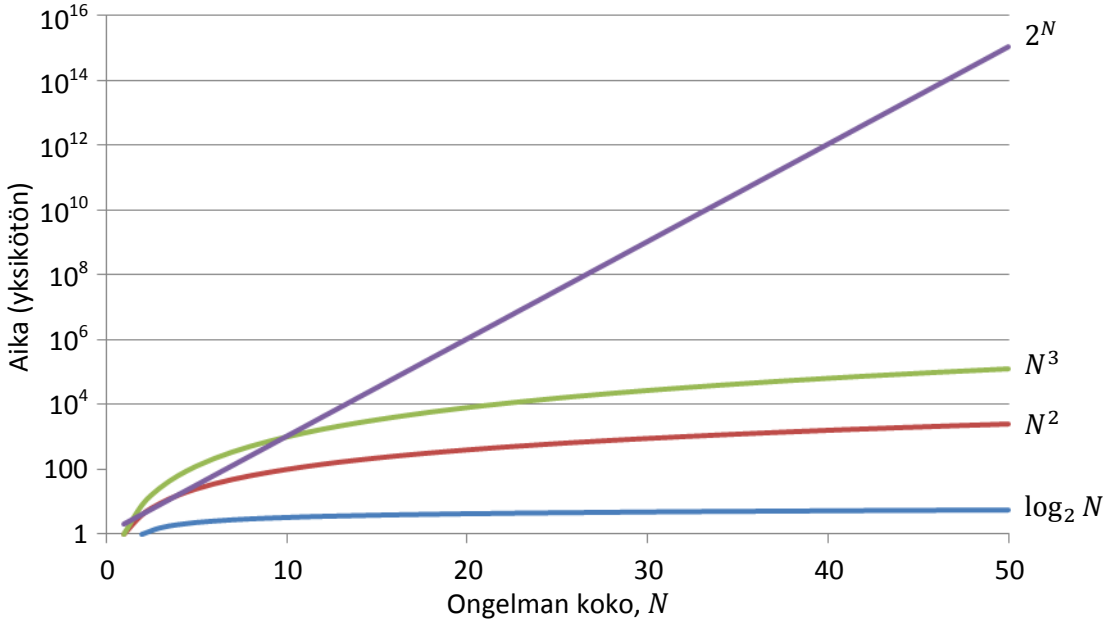
Yleisnimitys tilasta toiseen tiettyjen ehtojen mukaan etenevälle koneelle on tilakone. Eräs tilakoneen tyyppi on äärellinen automaatti (McCulloch & Pitts 1943, Hopcroft *et al.* 2000). Äärellinen automaatti on suppeampi versio Turingin koneesta. Erona Turingin koneeseen se ei kirjoita nauhalle ja liiku siinä vapaasti vaan lukee syötettään symboli kerrallaan ja vaihtaa tilaansa toimintaansa kuvaavien sääntöjen yksiselitteisesti sanelemalla tavalla riippuen nykyisen tilan ja symbolin yhdistelmästä. Jotkut tilat ovat lopetustiloja. Vajaavuuksiensa vuoksi äärellinen automaatti ei ole yleispätevä tietokone (Su & Smith 2004). Se soveltuu kuitenkin mainiosti sellaisten laskujen laskemiseen, jotka etenevät aina samantapaisesti ja joissa ei tarvita paljon muistia, esimerkiksi binärilukujen yhteenlaskuun. Stokasti-

sessa äärellisessä automaatissa (Rabin 1963) tilasiirtymä valitaan satunnaisesti. Tilan ja syötteen symbolin yhdistelmä määrää yksiselitteisesti eri tiloihin johtavien siirtymien todennäköisyydet.

1.6 Laskettavuusteoria

Kumpikin yleispätevä laskennan malli, Turingin kone ja Lambdakalkyyli, antaa saman vastauksen siihen, voiko annettuun kysymykseen laskea vastauksen äärellisessä ajassa. Vaikka laskutoimitukseen kuluva aika olisi äärellinen, se voi olla epäkäytännöllisen pitkä riippuen ongelman koosta sekä laskentamenetelmän eli algoritmin tehokkuudesta. Esimerkkinä toimikoon puhelinnumeron löytäminen nimien perusteella aakkosjärjestyksessä olevasta listasta, jossa on N henkilön tiedot. Alkeellinen ratkaisu on käydä koko lista alusta alkaen läpi. Tehokkaampi tapa on katsoa keskelle listaa, poistaa näkyvistä se listan puolisko, jolla haettu nimi ei aakkosjärjestyksen perusteella ole ja toistaa tätä kunnes haettu nimi sattuu kohdalle. Alkeellinen algoritmi vaatii enintään N nimen tarkistusta. Tehokkaampi algoritmi vaatii enintään $\log_2 N$ nimen tarkistusta. Esimerkiksi 10 000 nimen lista vaatisi alkeellisella algoritmilla enintään 10 000 ja tehokkaammalla algoritmilla enintään 14 nimen tarkistusta.

Algoritmit voidaan jakaa aikakompleksisuusluokkiin sen mukaan, kuinka paljon aikaa ne vievät pahimmassa tapauksessa. Polynominen aika tarkoittaa sitä, että jos ongelman koko on N, ohjelma palauttaa vastauksen ajassa, joka on N:n polynomi, esimerkiksi N^3. Vaikkakaan aikakompleksisuudeltaan polynominen algoritmi ei ole yhtä nopea kuin logaritminen, voidaan sitä sanoa nopeaksi, kun vertailukohde on eksponentiaalinen aikakompleksisuus (Kuva 5).



Kuva 5. Esimerkkejä eri kompleksisuusluokkiin kuuluvien algoritmien suoritusajoista ongelman koon funktiona. Aika-akselin asteikko on eksponentiaalinen. Aika-arvojen voidaan ajatella ilmoittavan, kuinka monta ohjelman ydinosan toistoa ratkaisualgoritmi vaati. Jos $N = 50$ ja ydinosan suoritus vie mikrosekunnin, kuluttaa eksponentiaalisen kompleksisuusluokan algoritmi (2^N) kymmeniä vuosia eli käytännössä liikaa aikaa ollakseen käyttökelpoinen. Kompleksisuudeltaan polynomisella algoritmilla (N^2 ja N^3) ratkaisu saataisiin alle sekunnissa. Logaritmiseen kompleksisuusluokkaan kuuluva algoritmi ($\log_2 n$) veisi vain muutamia mikrosekunteja.

Kaikille ongelmille ei tunneta tehokkaita ratkaisualgoritmeja. Erästä tärkeää ryhmää ongelmia kutsutaan NP-täydellisiksi (englanniksi nondeterministic polynomial time complete) (Cook 1971). Ongelma on NP-täydellinen, jos se kuuluu luokkaan NP ja kaikki ongelmat luokassa NP voidaan polynomisessa ajassa muuntaa sen muotoon. Ongelma kuuluu luokkaan NP (nondeterministic polynomial time), jos

vastaus siihen on Boolen muuttuja ja sille on olemassa kuvitteellisella epädeterministisellä Turingin koneella toimiva ratkaisualgoritmi, joka antaa vastauksen tosi polynomisessa ajassa, jos vastaus on tosi. Epädeterministinen Turingin koneen ohjelma sisältää ristiriitaisia ohjeita, joista kone selittämättömällä tavalla valitsee ne, joka nopeimmin johtavat lopetustilaan ja vastaukseen tosi, jos vastaus on tosi. Epädeterministisen ratkaisualgoritmin voi jakaa epädeterministiseen ehdotusosaan ja deterministiseen tarkastusosaan, joista jälkimmäinen on ohjelmaltaan ristiriidaton. Jos ratkaisu on olemassa, kone kirjoittaa epädeterministisesti nauhalle toimivan ratkaisuehdotuksen sekä deterministisesti tarkastaa sen ja palauttaa vastauksen tosi. Koska tarkastusosan suorittaminen ei voi olla hitaampaa kuin koko ohjelman suorittaminen, on luokan NP ongelmien ratkaisuehdotusten tarkastaminen mahdollista polynomisessa ajassa deterministisellä, toteutettavissa olevalla Turingin koneella.

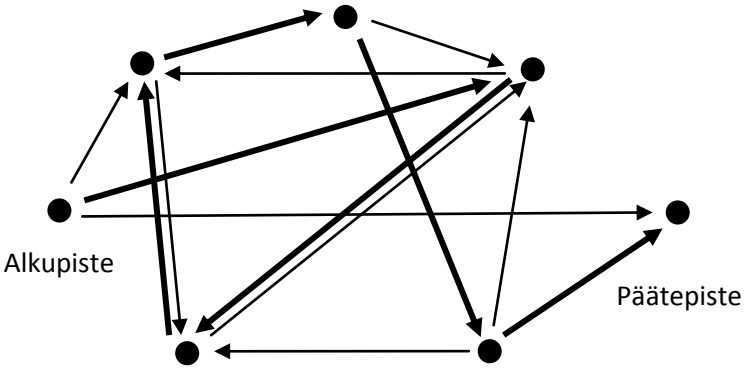
Deterministisiä polynomisessa ajassa toimivia NP-täydellisten ongelmien ratkaisualgoritmeja ei ole löydetty, eikä niiden olemassaoloon yleisesti uskota asiantuntijapiireissä (Gasarch 2002). Koska deterministisellä Turingin koneella voidaan eksponentiaalisessa ajassa, esimerkiksi ajassa 2^N, kun ratkaisu on ilmaistavissa N bitillä, käydä läpi kaikki mahdolliset ratkaisuvaihtoehdot, on luokan NP ongelmille ja niihin sisältyville NP-täydellisille ongelmille aina olemassa eksponentiaalisessa ajassa toimiva deterministinen ratkaisualgoritmi. Jos yhdellekään NP-täydelliselle ongelmalle löydettäisiin polynomisessa ajassa toimiva ratkaisualgoritmi, tarkoittaisi tämä sitä, että kaikki luokan NP ongelmat ratkeaisivat polynomisessa ajassa eli kuuluisivat luokkaan P (englanniksi polynomial time). Avoimen kysymyksen ”onko $\mathrm{P} = \mathrm{NP}$?” ratkaisijalle on luvassa miljoonan Yhdysvaltain dollarin palkinto yhdysvaltalaiselta Clay-instituutilta. Kysymys on tärkeä paitsi laskettavuusteorian kannalta myös siksi, että kompleksisuusluokan NP ongelmat, joiden ei ole voitu osoittaa kuuluvan luokkaan P, ovat hyvin yleisiä. Salausmenetelmät, joihin tietoliikenneverkkojen turvallisuus perustuu, käyttävät hyväkseen NP-täydellisten ongelmien ratkaisualgoritmien hitautta (Goldreich 2005). Logistiikassa oleellinen kauppamatkustajan ongelma eli lyhimmän reitin, joka kulkee kaikkien kaupunkijoukon kaupunkien kautta, löytäminen. Kauppamatkustajan ongelma kuuluu NP-täydellisiin ongelmiin, kun se esitetään muodossa: ”Onko olemassa annettua pituutta lyhyempi reitti, joka kulkee kaikkien kaupunkien kautta?” (Christos H. 1977). Kauppamatkustajan ongelma on niin sanottu kombinatorinen ongelma, mikä tarkoittaa sitä, että ongelmassa yritetään löytää määritellyllä tavalla paras vaihtoehtoisten osien yhdistelmä, tässä tapauksessa kaupunkien välisten reittien yhdistelmä.

2 Raa’an voiman algoritmit

2.1 Ongelma: Hamiltonin polku

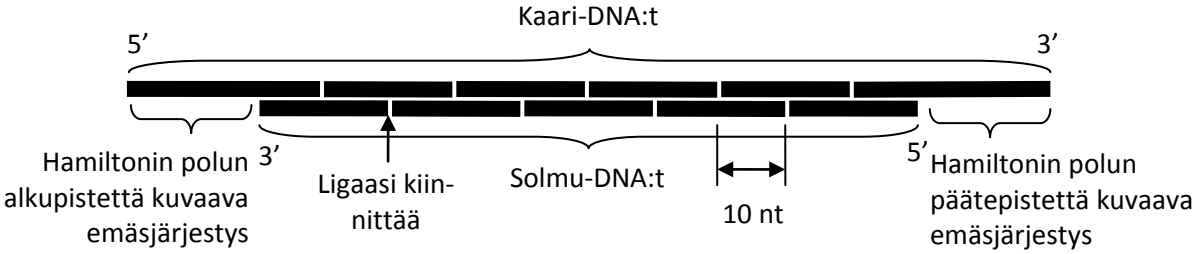
Mahdollisuus käyttää DNA-molekyylejä laskuvälineenä tuli yleiseen tietoisuuteen Leonard M. Adlemanin artikkelin ”Molecular Computation of Solutions to Combinatorial Problems” (Adleman 1994) myötä. Adleman oli onnistunut ratkaisemaan pienikokoisen Hamiltonin polkuun liittyvän ongelman DNA-molekyylien avulla. Graafiteoriassa graafi määritellään joukkona solmuja sekä joukkona kaaria, joista kukin alkaa tietystä solmusta ja päättyy tiettyyn solmuun. Hamiltonin polku kulkee graafin kaaria pitkin sen jokaisen solmun kautta käymättä samassa solmussa kahdesti. Jos kaarien määrittämien solmujen välisten yhteyksien lisäksi kaarien suunnat ovat merkitykselliset, sanotaan graafin olevan suunnattu. Adlemanin algoritmilla pystyi löytämään suunnatusta graafista Hamiltonin polun, joka alkoi annetusta solmusta ja päättyi toiseen annettuun solmuun (Kuva 6).

Adleman sekoitti koeputkessa ylimäärin kaaria ja solmuja paitsi Hamiltonin polun päätepistesolmuja vastaavia yksijuosteisia DNA-molekyylejä ja antoi niiden pariutua vapaasti. Molekyylien emäsjärjestykset oli suunniteltu siten, että kaari-DNA:t tarttuivat alku- ja päätepistesolmujansa kuvaaviin DNA-molekyyleihin 10 nt:n mittaisten käänteiskomplementaaristen alueiden välityksellä. Tuotteena oli kaksijuosteista DNA:ta, jonka toinen juoste koostui välittömästi peräkkäin olevista kaari-DNA:ista ja toinen juoste samalla tavoin solmu-DNA:ista (Kuva 7). Juosteet tehtiin yhtenäisiksi DNA-ligaasientsyymin avulla.



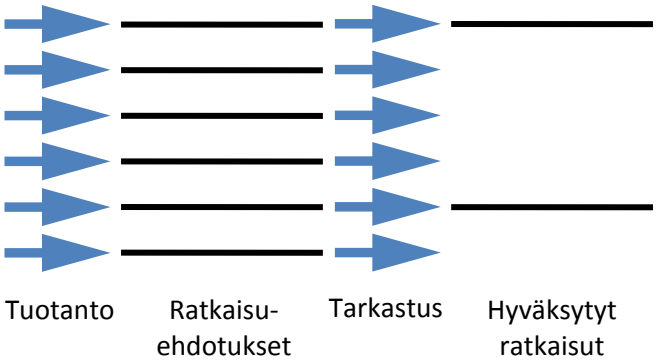
Kuva 6. Suunnattu graafi, jonka alku- ja päätepisteiltään määritellyn Hamiltonin polun Adleman löysi DNA-molekyylien avulla. Ratkaisupolkuun kuuluvat kaaret on lihavoitu.

Polku-DNA, joka alkoi ja päättyi vaadittuihin solmuihin, monistettiin polymeraasiketjureaktiolla (PCR). Alukkeina käytettiin Hamiltonin polun päätepistesolmu-DNA:ta vastaavaa sekä alkupistesolmu-DNA:n kanssa komplementaarista DNA:ta. Monistetuista polku-DNA:ista eroteltiin ne, jotka olivat seitsemän solmun mittaisia, käyttäen agaroosigeelielektroforeesia, jossa eri pituiset DNA-molekyylit kulkevat sähkökentän avulla agaroosigeelissä eri nopeudella ja näin erottuvat omiksi vyöhykkeikseen. Seitsemän solmua sisältävistä polku-DNA:ista valikoitiin tämän jälkeen ne, jotka sisälsivät kaikki muutkin solmut jo varmistettujen päätepistesolmujen lisäksi. Tätä varten polku-DNA:sta tehtiin yksijuosteista sisältäen enää vain kaarista koostuvan juosteen. Viidellä peräkkäisellä reaktiolla eroteltiin kullakin yhden vielä varmistamattoman solmun sisältävä DNA käyttäen aina edellisen reaktion puhdistettua lopputuotetta. Erottelu tehtiin magneettisiin helmiin sidotulla solmu-DNA:lla, johon tarttuivat vain sen kanssa komplementaarisen osan sisältävät DNA-molekyylit. Lopputuotteesta voitiin lukea löydetty Hamiltonin polku PCR:n avulla käyttäen kaikkia solmu-DNA:ita 5’-alukkeina erillisissä reaktioissa ja alkupistesolmu-DNA:n kanssa komplementaarista emäsjärjestystä 3’-alukkeena ja määrittämällä kunkin PCR-tuotteen koko agaroosigeelielektroforeesilla. Kokojärjestys pienimmästä suurimpaan ilmaisi 3’-alukkeissa käytettyjen solmujen järjestyksen Hamiltonin polulla eli annetun ongelman ratkaisun.



Kuva 7. Hamiltonin polkua kuvaava kaksijuosteinen DNA, joka syntyi Adlemanin kokeessa, koostui toisiinsa ligatoiduista kaaria ja solmuja kuvaavista DNA-pätkistä.

Hamiltonin polun löytäminen on NP-täydellinen ongelma (Karp 1972). Graafi, jota Adleman käytti ongelmassaan oli kuitenkin hyvin pieni, vain seitsemän solmua. Merkittävää olikin ongelman ratkaiseminen nimenomaan DNA-molekyylejä käyttäen, mikä herätti ajatuksen siitä, että kombinatoristen ongelmien ratkaisemiseen saattaisi olla uudenlainen, tehokas, molekyylien itseorganisaatioon perustuva lähestymistapa (Kuva 8).

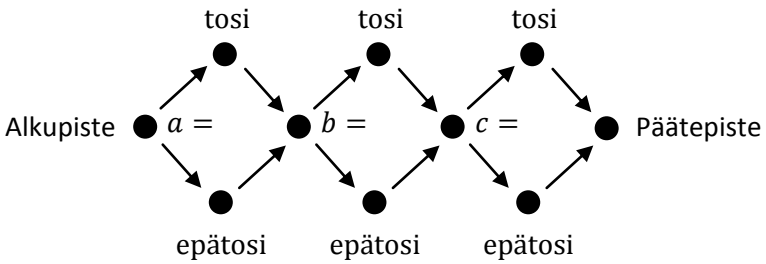


Kuva 8. Useimpien DNA- ja RNA-algoritmien laskennallinen rakenne kombinatorisia ongelmia ratkaistessa. Ratkaisuehdotukset tuotetaan rinnakkain eli yhtäaikaisesti ja myös tarkastetaan rinnakkain.

2.2 Ongelma: SAT

Eräs usein esiintyvä laskennallinen ongelma on NP-täydellinen niin sanottu toteutuvuusongelma (SAT-ongelma, englanniksi satisfiability problem) (Cook 1971), jossa tehtävänä on selvittää, löytyykö annetulle loogiselle lauseelle muuttujien arvojen yhdistelmä, joka tekee lauseen todeksi. Lause, joka koostuu konnektiivilla JA yhdistetyistä klausuuleista, joista kukin taas koostuu konnektiivilla TAI yhdistetyistä muuttujista tai niiden negaatioista, sanotaan olevan konjunktiivisessa normaalimuodossa. Esimerkiksi lause $(a \text{ TAI } b \text{ TAI EI } c) \text{ JA } \big((\text{EI } a) \text{ TAI } b \text{ TAI } c\big) \text{ JA } \big((\text{EI } a) \text{ TAI } (\text{EI } b) \text{ TAI } c\big)$ on tässä muodossa. Ongelma nimetään klausuuleissa olevien muuttujien ja niiden negaatioiden yhteismäärän perusteella, tässä tapauksessa 3-SAT-ongelmaksi. Mielenkiintoiseksi muodon tekee se, että jokaista Boolen algebran lausetta vastaa konjunktiivisen normaalimuodon lause, joka on sen kanssa totuusarvoltaan sama (Järvisalo 2004). Toteutuvuusongelman ratkaisualgoritmi eli toteutuvuustarkastin, jolle ongelma syötetään tässä muodossa, on siis yleispätevä lauselogiikan yhtälöiden ratkaisualgoritmi.

Richard J. Lipton tarkasteli teoreettisesti Adlemanin Hamiltonin polun löytämiseen käyttämää algoritmia ja esitteli samankaltaisen DNA-algoritmin SAT-ongelman ratkaisemiseen (Lipton 1995). Liptonin algoritmissa tietynlaisen suunnattun graafin tietystä solmusta alkavat ja tiettyyn solmuun päättyvät polut kuvaavat SAT-ongelman kaikkien muuttujien arvojen yhdistelmiä (Kuva 9).



Kuva 9. Liptonin tapa kuvata Boolen muuttujien arvoja perustui polkuun suunnatussa graafissa. Jos polku kulki ylärivin solmun kautta, sai tietty muuttuja arvon tosi. Jos se sen sijaan kulki alarivin solmun kautta, muuttujan arvoksi tuli epätosi.

Kukin polku, joka sisältää sekä alku- että päätepistesolmun, vastaa siis SAT-ongelman ratkaisuehdotusta. Kaikki mahdolliset polut voidaan tuottaa sekoittamalla kaikkia solmuja ja kaaria vastaavia lyhyitä, yksijuosteisia DNA-molekyylejä, jotka on suunniteltu ja jotka pariutuvat samaan tapaan kuin Hamiltonin algoritmissa (Kuva 7). Syntyvät kaksijuosteiset polku-DNA-molekyylit eheytetään ligaasientsyymin avulla. Alku- ja päätepistesolmujen läsnäolo poluissa voidaan varmistaa PCR:n avulla käyttäen

niitä vastaavia alukkeita. Ohjelman tarkastusosa suoritetaan klausuuli kerrallaan jättäen vain klausuulin toteuttavat polut jäljelle seuraavan klausuulin toteutumisen tarkastusta varten. Lopputuloksena on seos polkuja, jotka toteuttavat kaikki klausuulit. Yksittäisen klausuulin toteuttavat polut voidaan erotella muista siten, että polku-DNA:ista poimitaan muuttuja kerrallaan DNA-koettimen avulla erilleen ne, jotka määrittävät muuttujan arvoksi sen, joka on vaadittu klausuulin toteutumiseen. Kaikkien muuttujien kohdalla erilleen poimitut polku-DNA:t yhdistetään samaan koeputkeen, joka sisältää siten kaikki siihenastiset klausuulitarkastukset läpäisseet polku-DNA-molekyylit. Jos kaikkien tarkastusten jälkeen jäljellä on yhtään polku-DNA:ta, on SAT-ongelman vastaus ”kyllä” (tosi). Muutoin vastaus on ”ei” (epätosi).

Adlemanin ja Liptonin algoritmit toimivat liuoksissa, mutta vaihtoehto tälle oli sitoa kukin DNA- tai RNA-molekyyli kiinteään pintaan, mikä helpottaisi reaktioiden hallintaa. Puolitoista vuotta Adlemanin artikkelin julkaisun jälkeen Qinghua Liu työtovereineen esitteli suunnitelmansa pintakemiaan perustuvasta SAT-tarkastimesta (Liu *et al.* 1996). Liun ja työtovereitten SAT-tarkastimessa ratkaisuehdotuksia kuvaavat lasipintaan sidotut yksijuosteiset DNA-molekyylit, jotka sisältävät satunnaissynteesillä tuotetun, muuttujien arvoja yksittäisillä emäksillä kuvaavan osan. Konjunktiivisessa normaalimuodossa olevan loogisen lauseen klausuulit tarkastetaan yksitellen, poistaen ne vaihtoehdot, jotka eivät toteuta klausuulia. Esimerkiksi kolmen muuttujan klausuuli a TAI b TAI EI c voidaan kirjoittaa muotoon $(a = \text{tosi})$ TAI $(b = \text{tosi})$ TAI $(c = \text{epätosi})$. Klausuulin jättävät toteuttamatta ainoastaan ne ratkaisuvaihtoehdot, joissa kaikkien muuttujien arvot ovat klausuulissa lueteltujen negaatioita. Sellaiset merkitään poistettaviksi kiinnittämällä niihin koetin, joka tunnistaa muuttujien arvojen yhdistelmän. Poistaminen tapahtuu määrittelemättömällä, kaksijuosteisen DNA:n tunnistavalla nukleaasientsyymillä. Toinen tapa tarkastaa klausuuli on merkitä koettimella klausuulin toteuttavat ratkaisuehdotukset muuttuja kerrallaan, ja lopuksi poistaa merkitsemättä jääneet ratkaisuehdotukset Eksonukleaasi I -entsyymillä, joka tunnistaa ja pilkkoo yksijuosteisen DNA:n. Kutakin klausuulissa mainittua muuttujaa vastaavat, merkitsemiseen käytetyt koettimet syntetisoidaan erikseen käyttäen muiden muuttujien kohdalla satunnaisia nukleotideja. Klausuulien välissä koettimet poistetaan huuhtelemalla pintaa tislatulla vedellä, joka ei sisällä juosteiden välisiä sidoksia ylläpitävää suolaa. Jos annettuun SAT-ongelmaan on vain yksi ratkaisu, se voidaan lukea määrittämällä jäljelle jääneen DNA:n emäsjärjestys. Suunnitelma vaatii, että yksittäiset, virheellisten emäsparit havaitaan muilta osin komplementaarisista ratkaisuvaihtoehto-koetin -pareista. Koska useamman muuttujan tapauksessa tämä olisi hankalaa, Liu ja työtoverit ehdottivatkin, että muuttujien arvoja voisivat kuvata yksittäisten nukleotidien sijaan noin 15–20 nt:n mittaiset sanat, ja kehittivät menetelmän käyttökelpoisen sanajoukon koostamiseksi (Frutos *et al.* 1997). Neljän vuoden kuluttua ehdotuksesta julkaistiin heidän toteutuksensa 4 muuttujan 3-SAT-tarkastimesta (Liu *et al.* 2000). Ratkaisuehdotusten merkintä ja poistaminen ei toiminut täydellisesti. Osa ratkaisuehdotus-DNA-juosteista ei pariutunut koettimen kanssa, jonka avulla niitä merkittiin säästettäväksi. Toisaalta joskus koetin sitoutui osittaisesti väärään ratkaisuehdotus-DNA:han. Myöskään Eksonukleaasi I ei toiminut täydellisesti, vaan jätti noin 6 % yksijuosteisesta DNA:sta pilkkomatta. Algoritmin laajennettavuuden kannalta ongelmallista oli myös se, että liittyen Eksonukleaasi I:n käyttöön, muuttujien arvot sisällytettiin samaan sanaan, joten niiden arvoja ei voitu tarkastaa erikseen.

Samoihin aikoihin Dirk Faulhammer ja työtoverit ratkaisivat šakkipeliin liittyvän, 9 muuttujan SAT-ongelman DNA- ja RNA-molekyylien avulla (Faulhammer *et al.* 2000). Kyseessä oli ensimmäinen kerta, kun RNA:ta käytettiin laskutoimitusten suorittamiseen. Sanojen käyttö oli tarkoituksenmukaisempaa, sillä yksi sana kuvasi yhden Boolen muuttujan yhtä arvoa ja ratkaisuehdotuksien merkintä yksittäisiä sanoja tunnistavien koettimien avulla oli mahdollista. Ratkaisuehdotus-RNA koostui 5’→3’ järjestyksessä 24 emäksen mittaisesta 5’-osasta, 15 emäksen mittaisista yksilöllisistä sanoista, jotka kuvasivat ongelman muuttujien arvoja ja joiden väleissä oli 5 emäksen mittaiset niiden väliin tilaa tuovat väliosat, ja 32 nt mittaisesta 3’-osasta. Ratkaisuehdotuksia merkittiin poistettavaksi lisäämällä koeputkeen DNA:ta, jonka emäsjärjestys oli käänteiskomplementaarinen siihen sanaan nähden, joka kuvasi ehdon rikkovaa muuttujan arvoa ja siten sitoutui sanan sisältävään ratkaisuehdotus-RNA:han.

Poistaminen tapahtui lisäämällä koeputkeen RNaasi H-entsyymiä, joka tunnistaa ja katkaisee RNA:n, johon on sitoutunut DNA:ta. Ratkaisuvaihtoehto-RNA käännettiin DNA:n muotoon käänteiskopioija-entsyymin avulla. Täysimittainen DNA, joka vastasi ehdot toteuttaneita ratkaisuehdotuksia, monistettiin PCR:n avulla käyttäen ratkaisuehdotuksen 5'- ja 3'-osat tunnistavia alukkeita. Ratkaisuehdotukset käännettiin takaisin RNA:n muotoon T7 RNA-polymeraasilla seuraavan klausuulin tarkastusta varten. Annetulla SAT-ongelmalla oli useita ratkaisuja. Näiden erottamiseksi toisistaan viimeisen PCR:n tuotteet liitettiin antibioottivastustuskyvyn antavaan plasmidi-DNA:han, joka siirrettiin bakteerisoluihin. Yhteen bakteerisoluun siirtyi vain yksi plasmidi, ja kasvattamalla batkeereja antibioottia sisältävällä kasvatusalustalla voitiin yksittäisiä ratkaisuja lukea poimimalla yksittäisistä antibiootille vastustuskykyisistä bakteereista kasvaneita pesäkkeitä ja määrittämällä niissä olevien ratkaisu-DNA:itten kuvaamat muuttujien arvot. Arvojen määritystä varten käytettiin PCR-alukkeina yhdessä koeputkessa kunkin muuttujan arvoa epätosi, ja toisessa arvoa tosi kuvaavia sanoja. Erottelemalla PCR-reaktioiden tuotteet erilleen koon mukaan agaroosigeelielektroforeesilla voitiin päätellä, mitä arvoja mitkäkin muuttujat saivat. Kaiken kaikkiaan luettiin 43 eri ratkaisua, joista 31 olivat yksilöllisiä. Ratkaisuista 1 oli virheellinen. Virheen syyksi paljastui satunnainen mutaatio; yhdestä muuttujan arvoa kuvaavasta sanasta puuttui nukleotidi, ja lähellä sijaitseva emäs oli muuttunut toiseksi.

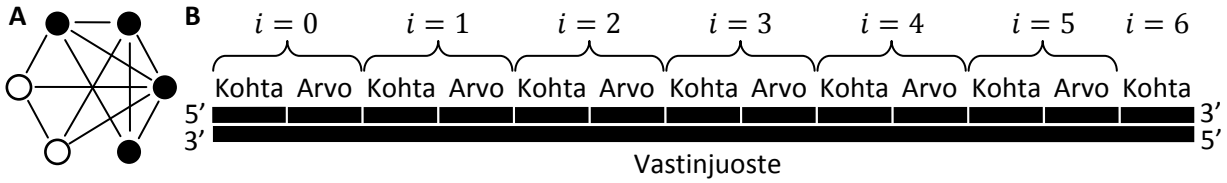
Samana vuonna julkaistiin Kensaku Sakamoton ja työtovereitten DNA:n sekundäärirakenteisiin perustuva 6 muuttujan ja 10 klausuulin 3-SAT-ratkaisualgoritmi (Sakamoto *et al.* 2000): Konjunktiivisessa normaalimuodossa olevan loogisen lauseen toteuttamiseen riittää, että ratkaisu toteuttaa kustakin klausuulista yhden jollekin muuttujalle asetetun ehdon. Ratkaisuehdotus voidaan kuvata klausuulien määrän pituisena ketjuna sanoja, jossa kukin sana vastaa yhtä siinä klausuulissa olevaa muuttujaa tai sen negaatiota, jonka järjestysnumero lausekkeessa on sama kuin sanan järjestysnumero ketjussa. Ratkaisuehdotuksessa on sisäinen ristiriita, jos siinä esiintyy sekä muuttujaa että sen negaatiota tarkoittava sana. Näiden DNA-sanaparien emäsjärjestykset valittiin niin, että ne sitoutuisivat toisiinsa kaksijuosteiseksi DNA:ksi. Näin ollen poistettavat ratkaisuehdotukset eli DNA-sanojen ketjut voitiin tunnistaa siitä, että niihin muodostui sekundäärirakenteita. Tunnistamiseen käytettiin kahta menetelmää: 1. Sanoihin oli liitetty tietyn restriktioentsyymin kaksijuosteisesta DNA:sta tunnistama kohta. 2. Ratkaisuehdotuksia monistettiin PCR:llä käyttäen pientä ratkaisuehdotus-DNA-pitoisuutta, jolloin sekundäärirakenteet pääsivät muodostumaan ja estivät polymeraasin toiminnan. Algoritmi tuotti oikean ratkaisun lisäksi runsaasti ristiriitaisia ja muutamia mutaatioiden sotkemia ratkaisuehdotuksia.

Toistaiseksi suurin DNA- ja RNA-algoritmein ratkaistu SAT-ongelma on 20 muuttujaa käsittävä 3-SAT-ongelma (Braich *et al.* 2002). SAT-tarkastinalgoritmi, jota Ravinderjit Braich työtovereineen käytti, toimi samaan tapaan kuin useimmat muutkin, karsimalla DNA:n muodossa olevia ratkaisuehdotuksia klausuulin tarkastus kerrallaan (Kuva 8). Ratkaisuehdotuksia kuvaavia DNA-molekyylejä liikuteltiin elektroforeesin avulla geelitäytteisten osastojen välillä. Klausuulintarkastusosastossa geeliin pysyvästi sidotut DNA-koettimet tunnistivat ja pysäyttivät klausuulin toteuttavat ratkaisuehdotus-DNA-molekyylit niissä olevien muuttujien arvoja kuvaavien sanojen perusteella ja antoivat muiden liikkua lävitseen. Klausuulintarkastusosastoon jääneet ratkaisuehdotus-DNA:t irrotettiin koettimista lämpötilaa nostamalla, jotta ne voitiin siirtää kohti seuraavaa klausuulintarkastusosastoa.

2.3 Ongelma: Suurin klikki

Klikillä tarkoitetaan sellaista suuntaamattoman graafin solmujen joukkoa, jonka kaikki solmut ovat kytkeytyneet kaarilla kaikkiin joukon muihin solmuihin. Suurimman klikin ongelma on löytää klikki, jota suurempaa graafi ei sisällä. Ongelma on sopivassa muodossa ilmaistuna NP-täydellinen (Garey & Johnson 1978). Kolme vuotta Adlemanin artikkelin julkaisun jälkeen Qi Ouyang ja työtoverinsa esittelivät algoritmin, jolla voitiin ratkaista suurimman klikin ongelma DNA-molekyyleillä (Ouyang *et al.* 1997). Algoritmin toimivuus osoitettiin löytämällä erään kuuden solmun kokoisen graafin suurin klikki (Kuva 10 A). Kuuden solmun klikki voidaan ilmaista kuuden Boolen muuttujan vektorilla. Kukin muuttuja ilmaisee, kuuluuko solmu klikkiin. Joukkoon A sijoitettiin aluksi kaikki kuuden muuttujan mittaiset vektorit. Täten A sisälsi myös ongelman ratkaisuklikkiä kuvaavan vektorin. Koska ongelmassa

annettu graafi sisälsi vain osan mahdollisista kaarista, poistettiin A:sta niitä klikkejä kuvaavat vektorit, jotka sisälsivät solmuja, joiden välisiä kaaria annetussa graafissa ei ollut. Tämä onnistui käymällä läpi kaikki annettuun graafiin kuulumattomat kaaret (i, j) ja poistamalla A:sta vektorit, joissa kohdissa i ja j olevien muuttujien arvo oli tosi. Jäljelle jäivät annetun graafin klikkejä kuvaavat vektorit, joista eniten tosi-arvoja sisältävä kuvasi suurinta klikkiä.



Kuva 10. A) Graafi, jonka suurimman klikin (mustat täytetyt ympyrät) Ouyang työtovereineen löysi DNA-laskennan avulla (Ouyang *et al.* 1997). B) Menetelmässä käytetyn, klikkiä kuvaavan DNA:n rakenne.

DNA:lle käännettynä algoritmi toimi seuraavalla tavalla: Kutakin vektoria vastasi kaksijuosteinen DNA, joka koostui vuorottelevista muuttujan kohtaa vektorissa ja muuttujan arvoa kuvaavista sanoista (Kuva 10 B). Arvoa kuvaavan sanan pituus riippui kyseisessä kohdassa olevan muuttujan arvosta. Joukon A vektoreita vastaavat DNA:t tuotettiin antamalla tietynlaisten, i:n arvoja $\{0, 1, 2, 3, 4, 5\}$ vastaavien yksijuosteisten DNA-molekyylien yhdistyä. Kukin molekyyli sisälsi kolme sanaa, jotka olivat 5’→3’-järjestyksessä kohta i, kohdassa i olevan muuttujan arvo ja kohta $i + 1$. Parittomilla i:n arvoilla käytettiin käänteiskomplementaarista emäsjärjestystä, jotta peräkkäisiä i:n arvoja vastaavat yksijuosteiset DNA-molekyylit voisivat sitoutua toisiinsa. Vastinjuosteitten puuttuvat sanat rakennettiin DNA-polymeraasientsyymillä. PCR:n avulla monistettiin täysimittaista klikki-DNA:ta. Klikki-DNA jaettiin kahteen koeputkeen lisäämällä ensimmäiseen restriktioentsyymiä, joka tunnisti sanan: kohdassa i olevan muuttujan arvo $=$ tosi, ja toiseen restriktioentsyymiä, joka tunnisti sanan: kohdassa j olevan muuttujan arvo $=$ tosi, ja sekoittamalla restriktiotuotteet. Restriktiokohdat oli lisätty arvoja vastaaviin DNA-emäsjärjestyksiin niitä suunniteltaessa. Restriktioreaktiot toistettiin peräjälkeen kaikille annettuun graafiin kuulumattomille kaarille (i, j). Jäljelle jääneet täysimittaiset klikki-DNA:t monistettiin PCR:n avulla. Suurinta klikkiä vastasi lyhin DNA, joten se voitiin erottaa muista agaroosigeelielektroforeesin avulla.

2.4 Sanojen suunnittelu

Joitain graafeihin liittyviä ongelmia on kokeiltu ratkaista rakentamalla DNA:sta fyysinen versio graafin osasista, ja antamalla niiden muodostaa kolmiulotteinen rakenne, joka muodoltaan vastaa graafia (Wu *et al.* 2009). Yleensä, kuten edellä kuvatuissa algoritmeissa, DNA- ja RNA-laskenta on kuitenkin symbolista. DNA- ja RNA-sanojen eli emäsjärjestysten joukko voidaan luoda systemaattisesti asettaen sille erityisiä vaatimuksia: Kunkin sanan tulisi olla yksilöllinen, ja se tulisi olla tunnistettavissa emäsjärjestykseltään käänteiskomplementaarisen juosteen eli vastinsanan avulla. Emäspariutumisen näiden välillä tulisi tapahtua täydellisesti eikä siten, että pariutuminen tapahtuu hieman väärään kohtaan. Sanojen ja vastinsanojen välisten sidosten tulisi olla suunnilleen yhtä voimakkaita, eikä vastinsana saisi tunnistaa vääriä sanoja (Frutos *et al.* 1997). Sanat eivät myöskään saisi pariutua sanojen kanssa tai muodostaa sisäisiä sekundäärirakenteita (Shortreed *et al.* 2005). Jos edeltävä vaatimus täyttyy, vältytään DNA:n tapauksessa kyseisiltä ongelmilta luultavasti myös vastinsanojen kohdalla. Yksinkertaistettu DNA-DNA-sidosenergiamalli (Owczarzy *et al.* 1997), jossa sidosenergiat eivät muutu, kun emäkset vaihdetaan vastinemäksiinsä, perustelee väittämän. RNA:n kohdalla voi olla syytä välttää G:n käyttöä, jotta ei-toivottuja GU-emäspareja ei syntyisi (Faulhammer *et al.* 2000). Sanojen suunnittelu voidaan ajatella optimointiongelmana, jossa pyritään luomaan haluttu määrä pituudeltaan rajoitettuja sanoja, joiden ominaisuudet vastaavat mahdollisimman hyvin edellä kuvattuihin vaatimuksiin (Frutos *et al.* 1997), tai jossa halutaan tuottaa mahdollisimman monen sanan joukko, joka toimii riittävän hyvin (Tulpan *et al.* 2005). Mahdollinen biologinen ympäristö tulisi myös ottaa huomioon sano-

ja suunniteltaessa, mutta koska vuorovaikutuksia sen kanssa on hankala ennustaa (Stein 2001), täytyisi kunkin sanan yhteensopivuus kohdeympäristön kanssa luultavasti kokeilla erikseen.

Eräs lähestymistapa on huolehtia sanojen ja vastinsanojen sitoutumisen kohdistamisesta siten, että kunkin sanan alkuosat ovat samat ja loppuosat ovat samat, ja laskennallisesti optimoida tunnistamiseen käytetyt sanojen yksilölliset keskiosat eli tunnisteosat erikseen. Ratkaisuna on esitetty 108:n DNA-sanan joukkoa, jossa tunnisteosan pituus on 8 emästä ja kummankin kohdistusosan pituus on 4 emästä. Jotta virheellisen sitoutumisen todennäköisyys jäisi pieneksi, sanojen ei tule olla kovin pitkiä, sillä pidemmät DNA-molekyylit sitoutuvat toisiinsa yleensä voimakkaammin kuin lyhyemmät (Frutos *et al.* 1997). Vaihtoehtoisesti voidaan sanajoukko optimoida vapaasti ilman keinotekoista jakoa tunniste- ja kohdistusosiin (Shortreed *et al.* 2005, Tulpan *et al.* 2005). Sanoista voidaan muodostaa usean sanan ketjuja (Frutos *et al.* 1997). Tällöin mahdollisten kohdistusosien voidaan myös ajatella olevan sanojen ulkopuolisia elementtejä, jolloin sanalla tarkoitetaan ainoastaan tunnisteosaa (Faulhammer *et al.* 2000).

2.5 Ratkaisujen tasapainotus

Jos ongelmalla on useita ratkaisuja ja tarkastusten jälkeen jäljelle jääneistä ratkaisuehdotuksista halutaan lukea useita, tulee pitää huoli, että ratkaisuehdotusten jakauma ei vääristy algoritmin työvaiheissa. Sanojen suunnittelussa (josta lisää jäljempänä) tämä heijastuu siinä, että sanan ja sen tunnistavan koettimen sitoutumisen tehokkuus tulisi olla mahdollisimman vakio. Tämän lisäksi kaikille yksittäisiä tarkastuksia läpäiseville ratkaisuehdotuksille tulisi olla sama määrä läpäisyreittejä, jotta minkään ratkaisun määrä ei kasvaisi toista suuremmaksi. Epätasapainoinen jakauma vaatii suuremman tilastollisen näytteen ottamista ratkaisuista, jos kaikki ratkaisut halutaan selvittää. Esimerkiksi SAT-ongelmassa klausuuli $a \text{ TAI } b$ voidaan tarkastaa jakamalla ratkaisuehdotukset kahteen koeputkeen, suorittamalla ensimmäisessä putkessa oleville ratkaisuehdotuksille tarkastus a ja toisessa putkessa oleville tarkastus b ja yhdistämällä putkien sisällöt. Lauseen $a \text{ JA } b$ toteuttavat ratkaisuehdotukset läpäisisivät tarkastukset kummassakin putkessa, ja niiden määräksi tulisi kaksinkertainen verrattuna vain jommankumman tarkastuksen läpäisseisiin ratkaisuehdotuksiin. Ratkaisuehdotukset voidaan tasapainottaa poistamalla ennen yhdistämistä toisesta putkesta ne ratkaisuvaihtoehdot, jotka toteuttavat osalauseen a (Faulhammer *et al.* 2000). Yhtälön $a \text{ TAI } b = a \text{ TAI } (b \text{ JA EI } a)$ perusteella tällä tavalla suoritettu tasapainotus on loogisesti neutraali toimenpide.

2.6 Käyttökelpoisuus

Edellä kuvattujen algoritmien laskennallinen rakenne sisältää heikkouden, tarpeen tuottaa kaikkia mahdollisia ratkaisuehdotuksia vastaavat DNA-molekyylit (Kuva 8). Koska ratkaisuehdotukset tuotetaan satunnaissynteesillä, on myös olemassa riski, että tarkastuksen läpäisevä ratkaisuehdotus jää tuottumatta. Tuottamalla ratkaisuehdotus-DNA:ta ylimäärin voidaan tätä riskiä kuitenkin hallita. Esimerkiksi N muuttujan SAT-ongelmalle on 2^N erilaista ratkaisuehdotusta. Jos oikeita ratkaisuja on vain yksi, ja ratkaisuehdotuksia vastaavia molekyylejä tuotetaan satunnaisesti 2^M kappaletta, on todennäköisyys olla tuottamatta ratkaisua vastaavaa molekyyliä $(1 - 1/2^N)^{2^M}$ tai suurille N:n ja M:n arvoille noin $1/e^{2^{M-N}}$, jossa $e \approx 2.718$ on luonnollisen logaritmin kantaluku. Näin ollen, jos vastauksen lukemiseen riittää yksi DNA-molekyyli, mikä on mahdollista (Schmidt *et al.* 2004), ja jos algoritmin missään vaiheessa ei hukata DNA:ta, yhden prosentin epäonnistumistodennäköisyys alitetaan tuottamalla noin viisinkertainen määrä ratkaisuehdotuksia ratkaisuvaihtoehtojen määrään nähden. SAT-ongelman koolle $N = 100$ tämä tarkoittaisi Liptonin menetelemällä noin kymmentä miljoonaa tonnia ratkaisuehdotuksia kuvaavaa kaksijuosteista DNA:ta, mikä on täysin epäkäytännöllinen määrä. Raa'an voiman DNA-algoritmit, jotka nojaavat massiiviseen rinnakkaisuuteen, epäonnistuvat suurten ongelmien ratkaisussa, koska DNA:n määrä kasvaa liian suureksi. Pintakemiallisilla menetelmillä tämä raja tulee nopeammin vastaan, sillä kaksiulotteiselle pinnalle voi olla kiinnittyneenä vähemmän DNA:ta, kuin mitä kolmiulotteiseen liuostilavuuteen mahtuu (Liu *et al.* 1996).

DNA- tai RNA-juosteen tunnistaminen komplementaarisen koettimen avulla vaatii, että molekyylit kohtaavat toisensa satunnaisen diffuusion avulla, ja siten sisältää aina epäonnistumisen riskin, joka korostuu suurilla DNA:n ja RNA:n määrillä ja suurissa liuostilavuuksissa. Toinen tulosten virheellisyyttä lisäävä ongelma DNA-laskennassa ovat satunnaiset mutaatiot (Faulhammer *et al.* 2000, Sakamoto *et al.* 2000). Niiden vuoksi suuri DNA:n määrä aiheuttaa ongelmia, jo ennen kuin DNA:n määrä kasvaa välittömäksi esteeksi, sillä suuri DNA:n määrä antaa satunnaisille mutaatioille enemmän tilaisuuksia ilmetä.

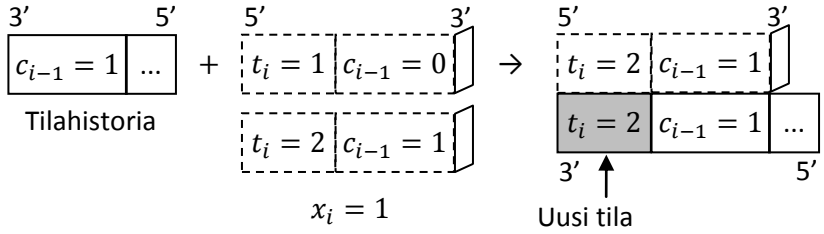
Myös perinteisellä tietokoneella kombinatorisia ongelmia voi yrittää ratkaista raa'alla voimalla, eli tuottamalla ja tarkastamalla peräkkäin kaikki ratkaisuehdotukset, mutta vaikka tietokoneet ovat nopeita, kuluisi tähän suurilla ongelmilla liikaa aikaa, esimerkiksi koon $N = 100$ SAT-ongelmalla triljoonia vuosia käyttäen vain yhtä suoritinydintä. Perinteisillä tietokoneilla toimivat toteutuvuustarkastimet, jotka on suunniteltu sellaisiksi, että ne toimivat käytännön ongelmissa raa'an voiman algoritmeja tehokkaammin, kehittyvät jatkuvasti (Järvisalo 2004). Perinteiset tietokoneet ovat myös joustavia ja luotettavia. Niiden ohjelmointia eivät hidasta molekyylibiologiset työvaiheet, eivätkä nykyiset suorittimet käytännöllisesti katsoen tee virheitä laskuissansa. Näistä syistä johtuen raa'an voiman DNA-algoritmit eivät ole kilpailukykyinen vaihtoehto valittaessa lähestymistapaa kombinatoristen ongelmien ratkaisemiseen.

3 Tilakoneet

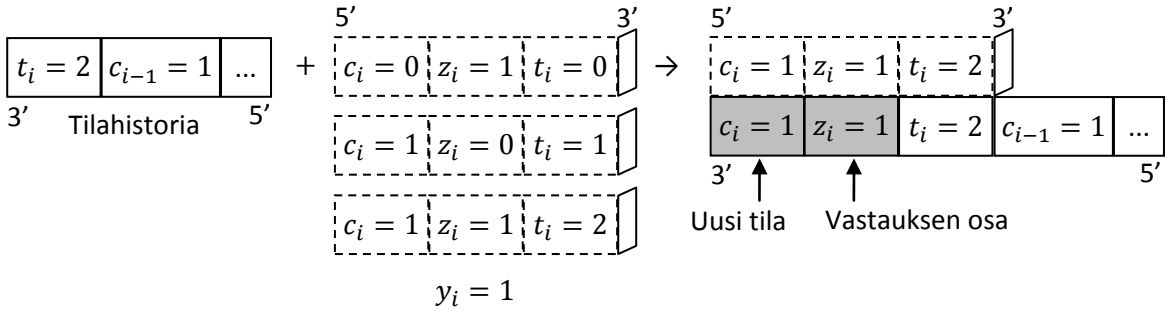
Edellä esitettyjen raa'an voiman DNA- ja RNA-algoritmien ongelmien myötä painopiste alan tutkimuksessa siirtyi yksinkertaisiin logiikkaverkkoihin sekä tilakoneisiin. Frank Guarnieri ja työtoverit esittelivät kahdeksan kuukautta Adlemanin artikkelin julkaisemisen jälkeen DNA:ta käyttävän, epänegatiivisten kokonaislukujen yhteenlaskualgoritmin (Guarnieri *et al.* 1996). Toimintaperiaatteeltaan algoritmi on eräänlainen äärellinen automaatti, ja sitä voidaan siten pitää askeleena kohti yleispätevää DNA-tietokonetta. Algoritmin heikkoutena syötteen muoto, jota voidaan pitää muutenkin hankalakäyttöisenä, ei ole yhteensopiva tuloksen muodon kanssa (Guarnieri *et al.* 1996). Algoritmin aikakompleksisuus on suhteessa lukujen kokoon eli polynominen.

Selitetään algoritmin toiminta esimerkillä $z = x + y$ (Kuva 11)., jossa edetään allekkainlaskun (Kuva 3) tapaan, ja ollaan tällä hetkellä tuottamassa vastauksen numeroa, jonka merkitsevyys on 2^i. Numeroa merkitään kunkin muuttujan kohdalla alaindeksillä i. Numeropari, joka sattuu olemaan $x_i = 1$ ja $y_i = 1$, lasketaan yhteen ottaen huomioon, että edellisten numeroparien yhteenlaskusta on sattunut jäämään muistiin 1, mikä ilmaistaan $c_{i-1} = 1$. Lasketaan väliaikaistulos, $t_i = c_{i-1} + x_i$, jonka arvojoukko on $\{0, 1, 2\}$. Lasketaan lopullinen tulos, $t_i + y_i$, jonka arvojoukko on $\{0, 1, 2, 3\}$, mikä voidaan ilmaista kaksibittisenä lukuna. Luvun merkitsevämpi bitti sijoitetaan muistiin muuttujaan c_i ja vähemmän merkitsevä bitti vastausmuuttujaan z_i. Tämän jälkeen voidaan siirtyä seuraavan numeroparin, x_{i+1} ja y_{i+1} yhteenlaskuun, joka suoritetaan vastaavalla tavalla. Jokaisella i:n arvolla kunkin x_i, y_i, z_i, c_i ja t_i jokaista mahdollista arvoa kuvaa lyhyehkö yksilöllinen DNA-emäsjärjestys. Vastaus-DNA muodostuu vuorottelevista muuttujien t_i, z_i ja c_i arvoista 5'→3'-järjestyksessä. Syötteet x_i ja y_i voidaan lisätä seokseen DNA:n muodossa kaikille i:n arvoille kerrallaan siten, että mukana on x_i:n tapauksessa seos kahdenlaista DNA:ta, joista kukin tunnistaa keskeneräisestä vastaus-DNA:sta tietyn c_{i-1}:n arvon ja y_i:n tapauksessa kolmenlaista DNA:ta, jotka tunnistavat eri t_i:n arvot. Tunnistus perustuu DNA-molekyylien komplementaarisuuteen. Syöte-DNA-molekyylit sisältävät yhteenlaskujen mahdollisia seurauksia ilmaisevat osat, joista toteutuneet lisätään tunnistettuun vastaus-DNA-molekyyliin DNA-polymeraasientsyymin avulla. Lyhyt 3'-lisäosa estää polymeraasientsyymiä jatkamasta syöte-DNA-molekyylejä.

Edellisen numeroparin yhteenlaskusta on muistissa $c_{i-1} = 1$. Lisätään numero $x_i = 1$:



Muistissa on väliaikaistulos $t_i = 2$. Lisätään numero $y_i = 1$:

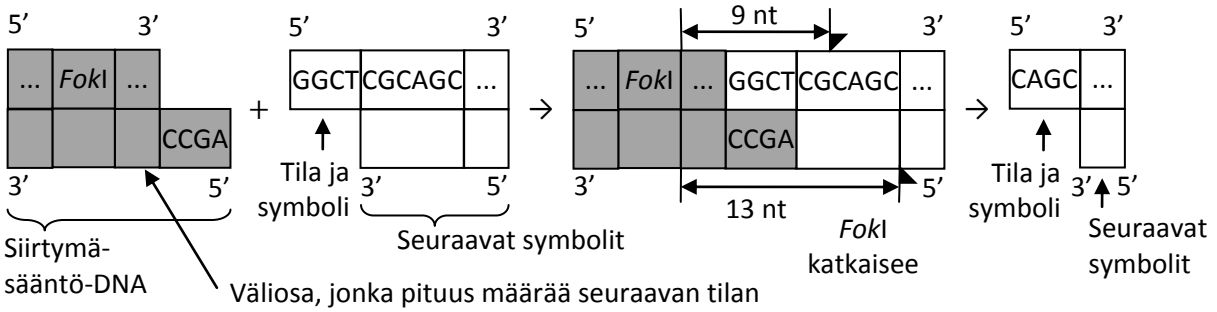


Saatiin vastaus $z_i = 1$, ja muistissa on $c_i = 1$ (käytetään seuraavan numeroparin yhteenlaskussa)

Kuva 11. Guarnierin ja työtovereitten yhteenlasku-DNA-algoritmin (Guarnieri *et al.* 1996) toiminta yhteenlaskulle $z = x + y$, jossa ollaan menossa bitissä i. Syötteiden x ja y bittejä ei ole suoranaisesti kirjoitettu DNA:han, vaan niiden arvoista riippuen seokseen on lisätty tilakoneen tilan mahdollisia muutoksia kuvaavia molekyylejä, jotka tunnistavat nykyisen tilan ja asettavat sen ja syötteen yhdistelmän perusteella koneelle uuden tilan. Väillä myös vastauksen bitti tallennetaan alati kasvavaan tilahistoria-DNA:han. Katkoviiva tarkoittaa, että sana on käänteiskomplementaarisessa emäsjärjestyksessä. Polymeraasilla tuotettavat sanat on tummennettu. Lisäosaa, joka estää polymeraasientsyymiä jatkamasta syöte-DNA-molekyyliä, symboloi suunnikkas.

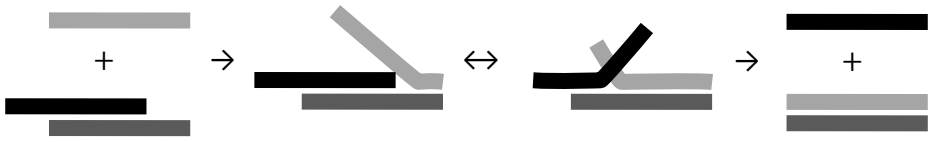
Nelisen vuotta Guarnierin yhteenlaskualgoritmin julkaisun jälkeen Yaakov Benenson ja työtoverit esittelivät varsin luotettavan, DNA:ta käyttävän äärellisen automaatin (Benenson *et al.* 2001). Kuva 12 selvittää algoritmin toimintaa. Syöte ja alkutila annettiin samassa kaksijuosteisessa DNA:ssa, jossa oli uloke, jonka kaksijuosteinen siirtymäsääntö-DNA tunnisti oman komplementaarisen ulokkeensa avulla. DNA:t yhdistettiin ligaasientsyymillä, mikä antoi käytetylle *Fok*I-restriktioentsyymille tilaisuuden toimia: Siirtymäsääntö-DNA:ssa oli *Fok*I:n tunnistuskohta, josta tarkkaan määrätyn matkan päästä *Fok*I katkaisi syöte-tila-DNA:n paljastaen uutta tilaa ja symbolia kuvaavan emäsjärjestyksen. Se, kuinka paljon syöte-tila-DNA:ta poistui, riippui siirtymäsääntö-DNA:n *Fok*I-tunnistuskohdan ja syöte-tila-DNA:n tunnistavan ulokkeen välisen osan pituudesta. Kukin syöte-tila-DNA:n sana oli sellainen, että kun se leikattiin sopivasta kohtaa *Fok*I:lla, automaatin uudeksi tilaksi asettui haluttu tila kahdesta vaihtoehtoisesta tilasta. Syötteen symboli, joita oli myös kaksi erilaista, oli kummassakin vaihtoehdossa sama. Koska ulokkeen pituus oli vain kaksi, ei kovin montaa symbolien ja tilojen yhdistelmää voitu tällä tavoin ilmaista. Viimeinen syöte-tila-DNA:sta paljastuva sana asetti automaatin ylimääräiseen kolmanteen tilaan, lopetustilaan, mikä havaittiin siirtymäsääntö-DNA:n sijaan lopetustilantunnistus-DNA:lla. Tilojen ja syötteen symbolien määrän kasvattamiseksi tarvittaisiin *Fok*I:n kaltaisia, tunnistuskohdan ulkopuolelta DNA:n katkaisevia restriktioentsyymejä, jotka tuottaisivat vielä pidemmän ulokkeen DNA:n päähän (Benenson *et al.* 2001). Ligaasientsyymin käytöstä luovuttiin automaatin myöhemmässä versiossa, mistä oli myös se etu, että siirtymäsääntömolekyylejä voitiin käyttää uudelleen (Benenson *et al.* 2003). Toiminnan luotettavuus säilyi suurena: 99,9 % tilasiirtymistä tapahtui oikein. Guarnierin yhtelaskualgoritmiin (Guarnieri *et al.* 1996) verrattuna Benensonin äärellisessä automaatissa oli edistyksellistä muun muassa se, että mielivaltaisen pitkä syöte pystyttiin an-

tamaan yhtenäisenä DNA:na, jossa symboleja kuvaavat sanat olivat samat riippumatta niiden sijainnista syötteessä.



Kuva 12. Benensonin DNA-tilakoneen (Benenson *et al.* 2001) toimintaperiaate. Komplementaaristen juosteiden sisältöä ei ole erikseen ilmaistu. Syötteen symboleja kuvasi kaksijuosteinen DNA, jonka päässä oli neljän nukleotidin uloke. Ulokkeen emäsjärjestys ilmaisi koneen senhetkisen tilan ja syötteen yhdistelmää. Yhdistelmää vastaava siirtymäsääntö-DNA tunnisti emäsjärjestyksen, ja kaksijuosteiset DNA:t yhdistettiin ulokkeistaan ligaasientsyymillä.

Muitakin lähestymistapoja DNA- ja RNA-logiikkaan ja -tilakoneisiin ehdotettiin seuraavina vuosina. Xingping Sun ja Lloyd M. Smithin pintakemiallinen tilakone (Su & Smith 2004) perustui pitkälti siihen, että yksijuosteiset DNA-vastinsanat irtosivat lämpötilaa nostettaessa pintaan kiinnitetyssä DNA-juosteessa peräkkäin olevista sanoista, ellei useampi vastinsana ollut tarttunut peräkkäisiin sanoihin ja ligatoitunut yhteen, jolloin kaksoiskierteen sulamislämpötila oli sen pituudesta johtuen korkeampi. Menetelmä vaati kuitenkin jatkuvia muutoksia koeolosuhteisiin. Myös DNA-molekyylien itsejärjestymisestä epäsäännöllisiksi kiteiksi hahmoteltiin äärellisten automaattien toteuttamistapaa (Rothemund *et al.* 2004), mutta toimiva sovellus on jäänyt toistaiseksi puuttumaan. Deoksiribotsyymeihin perustuvat logiikkaverkot (Stojanovic *et al.* 2002, Stojanovic & Stefanovic 2003, Lederman *et al.* 2006) vaikuttavat hitailta ja vaikeilta laajentaa. Pelkästään DNA-molekyyleistä koostuvat, DNA:n toisen juosteen syrjäyttämiseen (Kuva 13) perustuvat logiikkaverkot ovat kehittyneet viime vuosina (Seelig *et al.* 2006, Zhang *et al.* 2007, Phillips & Cardelli 2009) ja vaikuttavat lupaavilta. On kehitetty myös transkriptioverkkoja, joiden avulla saadaan loogisia operaattoreita toteuttavat komponentit lukkiutumaan eri tiloihin (Simpson *et al.* 2009). Niissä kukin RNA:n muodossa oleva lopputulos estää vaihtoehtoisen lopputuloksen transkriptiota.



Kuva 13. Juosteen syrjäyttäminen, jota käytetään joissain DNA- ja RNA-laskennan menetelmissä (Phillips & Cardelli 2009, Seelig *et al.* 2006, Zhang *et al.* 2007). Vaaleanharmaa juoste löytää kanssaan komplementaarisen tummanharmaan juosteen, sillä osa siitä ei ole pariutunut mustan juosteen kanssa. Vaaleanharmaa juoste syrjäyttää mustan juosteen tummanharmaan juosteen parina, koska vaaleanharmaa juoste muodostaa useampia emäspareja tummanharmaan juosteen kanssa kuin musta juoste.

4 Biologinen ympäristö

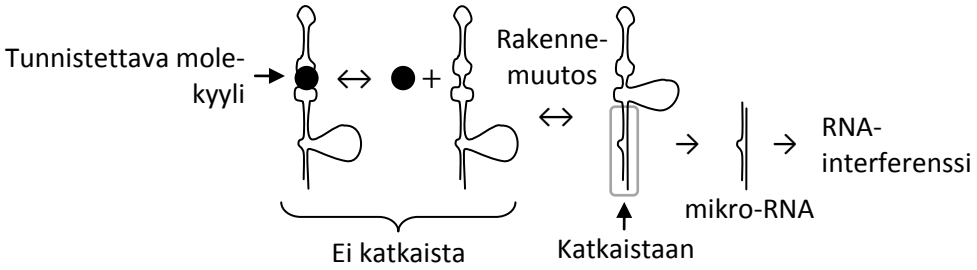
Benensonin äärellisen automaatin mullistavin ominaisuus oli se, että sen toimintaan ei tarvinnut puuttua ulkoisesti esimerkiksi lämpötilaa muuttelemalla tai lähtöaineita ja katalyyttejä lisäilemällä, vaan kone toimi itsenäisesti käynnistämisestään aina laskutoimituksen loppuun saakka. Tämä edistysaskel oli vaatimuksena, jos automaatin haluttiin toimivan jonkin organismin sisällä esimerkiksi sen sisäisen toiminnan geneettiseksi ohjaamiseksi. Benenson kehitti automaattiaan edelleen kohti tätä käyttötarkoitusta (Benenson *et al.* 2004). *In vitro* -kokein osoitettiin automaatin sellaisen version toimivuus, jossa tiettyjen lähetti-RNA:iden läsnäolo toimi syötteenä ja eri lopetustiloista seurasi tilaa vastaavan, tietyn yksijuosteisen DNA-molekyylin vapauttaminen automaatista. Koska lähetti-RNA-molekyyli eivät aina kohdanneet automaattia, ei niiden tunnistaminen ollut täydellistä. Tästä johtuen automaatti oli nähtävä stokastisena äärellisenä automaattina. Läsnäolevien syöte-mRNA:iden pitoisuudet muuttivat tilasiirtymien todennäköisyyksiä. Koska automaatista oli useita kopioita ja koska kahden eri lopetustilan tuottamat DNA:t olivat sellaisia, että ne pystyivät pariutumaan keskenään kaksijuosteiseksi DNA:ksi, jäljelle jäi lähinnä vain sitä lopetustilaa vastaavaa yksijuosteista DNA:ta, johon useimmat automaatit olivat päätyneet. Tämä vähensi yksittäisten automaattien tekemien virheiden merkitystä lopputuloksen kannalta.

Benensonin automaatin syöte- ja lopputulosmolekyylien emäsjärjestys oli vapaasti valittavissa käyttötarkoituksen mukaan. Otollinen, nykyään vielä kokeellinen käyttökohde DNA- ja RNA-tietokoneille on automaattinen lääketieteellinen diagnosointi mikrosirulla potilaalta otetun näytteen sisältämien molekyylien perusteella (Konry & Walt 2009, Zhang *et al.* 2009) ja lääkeaineiden annostelu (Benenson *et al.* 2004, Stojanovic & Stefanovic 2003) jopa solujen sisällä (Stojanovic *et al.* 2002, Beisel *et al.* 2008). Nämä käyttökohteet asettavat vaatimuksia koneen hyväksymien syötteiden ja sen tuottaman lopputuloksen muodolle. Päätös lääkinnän tarpeesta voidaan suorittaa esimerkiksi sellaisen loogisen lauseen perusteella, jossa muuttujat vastaavat tautiin liittyvien mRNA-molekyylien läsnäoloa (Benenson *et al.* 2004), tiettyjen proteiinien läsnäoloa (Konry & Walt 2009) tai esimerkiksi glukoosipitoisuutta sokeritaudissa (Taylor & Stojanovic 2007). Lääkeaine taas voi olla esimerkiksi mRNA:han sitoutuva yksijuosteinen DNA, joka estää tautia ylläpitävän proteiinin tuotannon (Benenson *et al.* 2004). Benensonin automaatin syötteiden avulla pystyttiin tunnistamaan mRNA-molekyylejä, ja lopputulos-DNA:ta pystyi käyttämään halutun proteiinin translaation estämiseen (Benenson *et al.* 2004). Jos tilakoneen ja sen biologisen ympäristön signaalit ovat eri muodoissa voidaan käyttää erillisiä niin sanottuja kääntäjiä. Esimerkkejä signaalin muodon muutoksista, joita varten on kehitetty kääntäjiä, ovat DNA→proteiini: aptameeri-DNA:n kahlitsema proteiini vapautetaan aptameeri-DNA:han sitoutuvan lopputuote-DNA:n avulla (Beyer & Simmel 2006), DNA→DNA: DNA-molekyyli-signaalin vaihtaminen emäsjärjestykseltään erilaiseen (Tabor *et al.* 2006), DNA → entsyymin aktiivisuuden muutos (Gianneschi & Ghadiri 2007).

Tilakoneen toiminta ei saisi häiriintyä biologisen ympäristön vuoksi, eivätkä sen osat saisi häiritä biologisen ympäristön toimintaa. Benensonin automaatin *Fok*I-entsyymi olisi lääketieteellisessä käytössä yhteensopimaton solunsisäisen ympäristön kanssa (Condon 2004). Biologinen ympäristö tekee myös koneessa käytettyjen, erityisesti yksijuosteisten DNA-molekyylien emäsjärjestyksien valitsemisen hankalaksi, sillä niillä voi olla vaikeasti ennakoitavia vuorovaikutuksia ympäristön kanssa (Stein 2001). Näiden ongelmien ratkaisemiseksi Israel M. Martínez-Pérez ja työtoverit suunnittelivat niin sanotun laskennallisen geenin (englanniksi computational gene) (Martinez-Perez *et al.* 2007), joka oli aluksi kahtena palasena, mutta yhdistyi toimivaksi geeniksi, jos läsnä oli tietyn mutatoituneen geenin mRNA:ta (Kuva 14). Laskennallisen geenin voitiin olettaa sopivan hyvin tarkoitettuun biologiseen ympäristöönsä, ihmissolun sisään, sillä se koostui pelkästään DNA:sta ja oli emäsjärjestyksiltään ihmisen genomin kaltaista. Laskennallisen geenin proteiinin tuoton ehtolause on mahdollista laajentaa muotoon $\mathrm{mutaatio}_1\ \mathrm{JA}\ \mathrm{mutaatio}_2\ \mathrm{JA}\ \mathrm{mutaatio}_3$, jossa mutaatio-muuttujat vastaavat tiettyjen mutatoituneiden geenien läsnäoloa mRNA:n muodossa. Tämä onnistuu pilkkomalla keinotekoisen geenin introni useampaan kuin kahteen osaan (Martinez-Perez *et al.* 2007). Laskennallinen geeni on rajoittunut JA-operaattoriin, eli se ei ole yleispätevän tilakoneen arkkitehtuuri.

Kuva 14. Laskennallinen geeni (Martinez-Perez *et al.* 2007), jonka koodaaman proteiinin tuotanto riippuu tietyn mRNA:n läsnäolosta. Epätäydellisesti komplementaarisista juosteista koostuvan osin kaksijuosteisen, keinotekoisen diagnoosi-DNA:n ensimmäinen juoste sitoutuu tunnistettavaan mRNA:han, johon nähden se on täydellisesti komplementaarinen. Syrjäytynyt toinen juoste kiinnittää keinotekoisen geenin osaset toisiinsa tehden sen toimivaksi. Solun oma RNaasi H tuhoaa mRNA:n, johon on sitoutunut DNA:ta.

Eräs mekanismi, jonka avulla ihmissolu voi vähentää tietyn proteiininsa tuotantoa, on mikro-RNA-interferenssi. Mikro-RNA on pieni, noin 23 nt pitkä RNA-molekyyli, jonka solun oma koneisto irrottaa sen hiuspinnimäisestä esiasteesta ja auttaa sitoutumaan sen tunnistamaan lähetti-RNA:han, jossa se estää proteiinin tuottoa (David P. 2009). Mekanismia voidaan käyttää myös keinotekoiseen proteiinin tuotannon säätelyyn, jossa syötteenä toimii tunnistettavan molekyylin läsnäolo (Beisel *et al.* 2008). Tunnistettavan molekyylin sitoutuminen Chase L. Beiselin ja työtovereitten suunnittelemaan RNA-kytkimeen estää kytkintä asettumasta sekundäärirakenteeseen, joka johtaisi mikro-RNA:n vapauttamiseen siitä (Kuva 15). Menetelmän on osoitettu toimivan viljellyissä ihmissoluissa, jotka saatiin geenisiirron avulla tuottamaan teofylliiniä sitovaa kytkin-RNA:ta, joka ilman teofylliiniä toimi mikro-RNA:n esiasteena. Menetelmässä ei sinällään suoriteta loogisia laskutoimituksia, vaan se voidaan ymmärtää signaalin kääntäjäksi: tunnistettava molekyyli → EI mikro-RNA → proteiini. Myös käännöksen mRNA → mikro-RNA tuottava, juosteen syrjäyttämiseen perustuva menetelmä on kehitetty, mutta sitä on testattu vain solu-uutteessa (Xie *et al.* 2010).



Kuva 15. Molekyylin tunnistava ja RNA-interferenssiä säätelevä RNA-kytkin (Beisel et al. 2008). RNA-kytkimellä on kaksi vaihtoehtoista sekundäärirakennetta. Tunnistettavan molekyylin sitoutuminen kytkimeen vakauttaa näistä ensimmäisen. Solun oma RNA-interferenssikoneisto tunnistaa toisen rakenteen mikro-RNA:n esiasteeksi ja irrottaa siitä osan, joka toimii translaatiota estävänä mikro-RNA:na.

Kombinatoristen ongelmien DNA- ja RNA-laskentaa on pienessä mittakaavassa onnistuneesti kokeiltu tehdä solujen sisällä. Karmella A Haynesin ja työtovereitten menetelmän (Baumgardner *et al.* 2009, Haynes *et al.* 2008) perusajatuksena se, että kukin nopeasti lisääntyvä *Escherichia coli* -bakteerisolu tuottaa itse vähintään yhden uuden ratkaisuehdotuksen 20–30 minuutin lisääntymiskiertonsa aikana. Ratkaisuehdotusten tuottaminen perustuu Hin-rekombinaasiin, joka löytää ja kääntää *HixC*-tunnistuskohtien välissä olevan DNA-jakson suunnan. Kun tunnistuskohtia on tarjolla useampia, tapahtuu

niiden valinta satunnaisesti joskaan ei välttämättä tasapuolisesti. Suurella ongelman koolla menetelmä ei ole käyttökelpoinen edellä mainittujen käytännön ongelmien vuoksi.

5 Tulevaisuudennäkymiä

DNA- ja RNA-laskenta kehittyy synteettisen biologian osana, jossa sitä tulevaisuudessa käytettäneen aluksi yksinkertaisissa keinotekoisissa säätöjärjestelmissä. Myös luonnollisten biokemiallisten säätöverkkojen voidaan ajatella suorittavan yksinkertaista laskentaa (Benenson 2009). Esimerkiksi eräillä bakteereilla laktoosin hajottamisen geenejä säätelee karkeasti ajatellen looginen lauseke ”laktoosi JA EI glukoosi” (Setty *et al.* 2003). On mahdollista, että luultavasti varsin kaukaisessa tulevaisuudessa luodaan laskukyvyltään Turingin konetta vastaava, nopea, yleispätevä molekyylilaskentaan perustuva tietokone, joka on kykenevä toimimaan biologisessa ympäristössä, kommunikoi sen kanssa hallitulla tavalla ja jota voidaan ohjelmoida helposti yksinkertaisella kielellä ilman, että tietokoneen rakennetta täytyy muuttaa jokaista sovellusta varten. Jotta tietokone olisi nopea, tietojen käsittely ei luultavasti tapahdu kokonaan satunnaisen diffuusion aiheuttamien, tietoa sisältävien molekyylien välisten kohtaamisien kautta, vaan on ohjattua samaan tapaan kuin luonnollinen proteiinisynteesi geneettisen DNA:n sisältämän tiedon perusteella. Kyseisen tiedonsiirtoketjun jokaisessa välivaiheessa on mukana komponentteja, jotka aktiivisesti ohjaavat prosessin kulkua. Koska deterministinen tietokone ei saa tehdä juurikaan virheitä, joita molekyylien tasolla tapahtuvassa tiedon käsittelyssä helposti syntyy, täytyy tietokoneen sisältää luonnollisten organismien tapaan mekanismeja virheiden korjaukseen.

Toistaiseksi DNA- ja RNA-laskenta on rajoittunut tieteelliseksi tutkimusalueeksi, eikä siitä ole ollut suoranaista hyötyä. Tilanteen voidaan kuitenkin odottaa muuttuvan, viimeistään kun DNA- ja RNA-logiikkaan perustuvia, sähköä tarvitsemattomia automaattisia diagnostiikkamenetelmiä otetaan laajemmin käyttöön lääketieteellisissä tai vaikka elintarviketurvallisuuteen liittyvissä diagnostiikkasiruissa tai kun niitä yhdistetään lääkeaineisiin uudenlaisiksi solujen tautitiloja tunnistaviksi täsmälääkkeiksi. Nämä käyttökohteet eivät vaadi monimutkaisia laskutoimituksia, joihin vielä tavoittamattomissa oleva yleispätevä DNA- ja RNA-laskentaan perustuva tietokone kykenisi. DNA- ja RNA-laskentaan perustuvien täsmälääkkeiden käyttöönottoa hidastaa se, että niiden rinnalle tulee kehittää menetelmät niiden tuomiseksi solujen sisään. Tapauskohtaista kartoittamista vaativat myös mahdolliset sivuvaikutukset.

6 Kirjallisuusluettelo

Adleman LM (1994) Molecular computation of solutions to combinatorial problems. Science 266(5187): 1021-1024.

Baumgardner J, Acker K, Adefuye O, Crowley ST, Deloache W, Dickson JO, Heard L, Martens AT, Morton N, Ritter M, Shoecraft A, Treece J, Unzicker M, Valencia A, Waters M, Campbell AM, Heyer LJ, Poet JL & Eckdahl TT (2009) Solving a Hamiltonian Path Problem with a bacterial computer. J Biol Eng 3: 11.

Beisel CL, Bayer TS, Hoff KG & Smolke CD (2008) Model-guided design of ligand-regulated RNAi for programmable control of gene expression. Mol Syst Biol 4: 224.

Benenson Y (2009) Biocomputers: from test tubes to live cells. Mol Biosyst 5(7): 675-685.

Benenson Y, Adar R, Paz-Elizur T, Livneh Z & Shapiro E (2003) DNA molecule provides a computing machine with both data and fuel. Proc Natl Acad Sci U S A 100(5): 2191-2196.

Benenson Y, Gil B, Ben-Dor U, Adar R & Shapiro E (2004) An autonomous molecular computer for logical control of gene expression. Nature 429(6990): 423-429.

Benenson Y, Paz-Elizur T, Adar R, Keinan E, Livneh Z & Shapiro E (2001) Programmable and autonomous computing machine made of biomolecules. Nature 414(6862): 430-434.

Beyer S & Simmel FC (2006) A modular DNA signal translator for the controlled release of a protein by an aptamer. Nucleic Acids Res 34(5): 1581-1587.

Braich RS, Chelyapov N, Johnson C, Rothemund PWK & Adleman L (2002) Solution of a 20-Variable 3-SAT Problem on a DNA Computer. Science 296(5567): 499-502.

Christos H. P (1977) The Euclidean travelling salesman problem is NP-complete. Theor Comput Sci 4(3): 237-244.

Church A (1936) A Note on the Entscheidungsproblem. Journal of Symbolic Logic 1: 40-41.

Condon A (2004) Automata make antisense. Nature 429(6990): 351-352.

Cook SA (1971) The complexity of theorem-proving procedures. Proceedings of the third annual ACM symposium on Theory of computing. New York, NY, USA, ACM: 151-158.

David P. B (2009) MicroRNAs: Target Recognition and Regulatory Functions. Cell 136(2): 215-233.

Every AE & Russu IM (2007) Probing the role of hydrogen bonds in the stability of base pairs in double-helical DNA. Biopolymers 87(2-3): 165-173.

Faulhammer D, Cukras AR, Lipton RJ & Landweber LF (2000) Molecular computation: RNA solutions to chess problems. Proceedings of the National Academy of Sciences 97(4): 1385-1389.

Frutos AG, Liu Q, Thiel AJ, Sanner AMW, Condon AE, Smith LM & Corn RM (1997) Demonstration of a word design strategy for DNA computing on surfaces. Nucleic Acids Research 25(23): 4748-4757.

Garey MR & Johnson DS (1978) " Strong " NP-Completeness Results: Motivation, Examples, and Implications. J.ACM 25(3): 499-508.

Gasarch W (2002) Guest column: The P=?NP poll. SIGACT NEWS 33(2 (June)): 34-47.

Gianneschi NC & Ghadiri MR (2007) Design of molecular logic devices based on a programmable DNA-regulated semisynthetic enzyme. Angew Chem Int Ed Engl 46(21): 3955-3958.

Goldreich O (2005) Foundations of cryptography: a primer. Found.Trends Theor.Comput.Sci. 1(1): 1-116.

Guarnieri F, Fliss M & Bancroft C (1996) Making DNA Add. Science 273(5272): 220-223.

Haynes KA, Broderick ML, Brown AD, Butner TL, Dickson JO, Harden WL, Heard LH, Jessen EL, Malloy KJ, Ogden BJ, Rosemond S, Simpson S, Zwack E, Campbell AM, Eckdahl TT, Heyer LJ & Poet JL (2008) Engineering bacteria to solve the Burnt Pancake Problem. J Biol Eng 2: 8.

Hopcroft JE, Motwani R, Rotwani & Ullman JD (2000) Introduction to Automata Theory, Languages and Computability. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc.

Järvisalo M (2004) Lauselogiikan toteutuvuustarkastus: käytännönläheistä teoriaa. Tietojenkäsittelytiede 22: 47-63.

Karp RM (1972) Reducibility Among Combinatorial Problems. In Miller RE & Thatcher JW (eds) Complexity of Computer Computations, Plenum Press: 85-103.

Konry T & Walt DR (2009) Intelligent medical diagnostics via molecular logic. J Am Chem Soc 131(37): 13232-13233.

Krutz RL (1980) Microprocessors and Logic Design. United States of America, John Wiley & Sons, Inc.: 79-110.

Lederman H, Macdonald J, Stefanovic D & Stojanovic MN (2006) Deoxyribozyme-based three-input logic gates and construction of a molecular full adder. Biochemistry 45(4): 1194-1199.

Lipton RJ (1995) DNA solution of hard computational problems. Science 268(5210): 542-545.

Liu Q, Guo Z, Fei Z, Condon AE, Corn RM, Lagally MG & Smith LM (1996) A Surface-Based Approach to DNA Computation. 2nd DIMACS workshop on DNA based computers, American Mathematical Society: 206-216.

Liu Q, Wang L, Frutos AG, Condon AE, Corn RM & Smith LM (2000) DNA computing on surfaces. Nature 403(6766): 175-179.

Martinez-Perez IM, Zhang G, Ignatova Z & Zimmermann KH (2007) Computational genes: a tool for molecular diagnosis and therapy of aberrant mutational phenotype. BMC Bioinformatics 8: 365.

McCulloch W & Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biol 5(4): 115-133.

Ouyang Q, Kaplan PD, Liu S & Libchaber A (1997) DNA Solution of the Maximal Clique Problem. Science 278(5337): 446-449.

Owczarzy R, Vallone PM, Gallo FJ, Paner TM, Lane MJ & Benight AS (1997) Predicting sequence-dependent melting stability of short duplex DNA oligomers. Biopolymers 44(3): 217-239.

Phillips A & Cardelli L (2009) A programming language for composable DNA circuits. J R Soc Interface 6 Suppl 4: S419-36.

Rabin MO (1963) Probabilistic Automata. Information and Control : 230-245.
Rothemund PW, Papadakis N & Winfree E (2004) Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biol 2(12): e424.
Sakamoto K, Gouzu H, Komiya K, Kiga D, Yokoyama S, Yokomori T & Hagiya M (2000) Molecular Computation by DNA Hairpin Formation. Science 288(5469): 1223-1226.
Schmidt KA, Henkel CV, Rozenberg G & Spaink HP (2004) DNA computing using single-molecule hybridization detection. Nucleic Acids Res 32(17): 4962-4968.
Seelig G, Soloveichik D, Zhang DY & Winfree E (2006) Enzyme-free nucleic acid logic circuits. Science 314(5805): 1585-1588.
Setty Y, Mayo AE, Surette MG & Alon U (2003) Detailed map of a cis-regulatory input function. Proc Natl Acad Sci U S A 100(13): 7702-7707.
Shortreed MR, Chang SB, Hong D, Phillips M, Campion B, Tulpan DC, Andronescu M, Condon A, Hoos HH & Smith LM (2005) A thermodynamic approach to designing structure-free combinatorial DNA word sets. Nucleic Acids Res 33(15): 4965-4977.
Simpson ZB, Tsai TL, Nguyen N, Chen X & Ellington AD (2009) Modelling amorphous computations with transcription networks. J R Soc Interface 6 Suppl 4: S523-33.
Stein CA (2001) The experimental use of antisense oligonucleotides: a guide for the perplexed. J Clin Invest 108(5): 641-644.
Stojanovic MN & Stefanovic D (2003) Deoxyribozyme-based half-adder. J Am Chem Soc 125(22): 6673-6676.
Stojanovic MN, Mitchell TE & Stefanovic D (2002) Deoxyribozyme-Based Logic Gates. J Am Chem Soc 124(14): 3555-3561.
Su X & Smith LM (2004) Demonstration of a universal surface DNA computer. Nucleic Acids Res 32: 2004.
Sugimoto N, Nakano M & Nakano S (2000) Thermodynamics-structure relationship of single mismatches in RNA/DNA duplexes. Biochemistry 39(37): 11270-11281.
Tabor JJ, Levy M & Ellington AD (2006) Deoxyribozymes that recode sequence information. Nucleic Acids Res 34(8): 2166-2172.
Taylor S & Stojanovic MN (2007) Is there a future for DNA-based molecular devices in diabetes management? J Diabetes Sci Technol 1(3): 440-444.
Tulpan D, Andronescu M, Chang SB, Shortreed MR, Condon A, Hoos HH & Smith LM (2005) Thermodynamically based DNA strand design. Nucleic Acids Res 33(15): 4951-4964.
Turing AM (1936) On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society. Second Series 42: 230-265.
Vogel J & Wagner EG (2007) Target identification of small noncoding RNAs in bacteria. Curr Opin Microbiol 10(3): 262-270.
Whitehead AN (1898) A treatise on universal algebra, with applications. , Cambridge: The University Press.
Wu G, Jonoska N & Seeman NC (2009) Construction of a DNA nano-object directly demonstrates computation. BioSystems 98(2): 80-84.
Xie Z, Liu SJ, Bleris L & Benenson Y (2010) Logic integration of mRNA signals by an RNAi-based molecular computer. Nucleic Acids Res 38(8): 2692-2701.
Zhang DY, Turberfield AJ, Yurke B & Winfree E (2007) Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA. Science 318(5853): 1121-1125.
Zhang Y, Yu H, Qin J & Lin B (2009) A microfluidic DNA computing processor for gene expression analysis and gene drug synthesis. Biomicrofluidics 3(4): 44105.