DNA-ketjun laskennalliset ominaisuudet

Antti Takalahti

Helsinki 28.10.2004

Vaihtoehtoiset laskentaparadigmat

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Matemaattis-luonnontieteellinen	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author

Antti Takalahti

Työn nimi — Arbetets titel — Title

DNA-ketjun laskennalliset ominaisuudet

Oppiaine — Läroämne — Subject

Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Seminaarityö	28.10.2004	9 sivua

Tiivistelmä — Referat — Abstract

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Molekyylibiologiassa on tehty viime aikoina valtavaa edistystä. DNA-ketjua pystytään käsittelemään monin tavoin, ja onkin mielenkiintoista pohtia, millaisia laskennallisia ominaisuuksia reaktoilla voisi olla. Esitelmä pohjaa Takashi Yokomorin artikkeliin [Yok02], jossa Yokomori esittelee useita edistysaskeleita. Esitelmä keskittyy tarkastelemaan molekyylikoneita laskennan teorian kautta.

Toinen luku osoittaa, miten DNA-molekyyleistä voidaan muodostaa epädeterministinen äädellinen automaatti. Kolmas luku esittelee erään, perinteisesti vaikean, tietojenkäsittelytieteen ongelman ja tarjoaa siihen varsin yksinkertaisen ratkaisun. Kolmannessa luvussa esitellään ns. *hiuspinnirakenne*, jota analysoidaan tarkemmin neljännessä luvussa.

2 Äärellinen automaatti

Äärellinen automaatti on eräs tietojenkäsittelytieteen perinteisimpiä malleja. Se on myös laskettavuuden kannalta mieluinen malli, sillä äärellinen automaatti on eräs laskettavuuden mitta. Kun tiedetään jonkin asian olevan yhtä vahva laskennallinen malli kuin äärellinen automaatti, tiedetään sen laskentavoima suhteessa muihin malleihin.

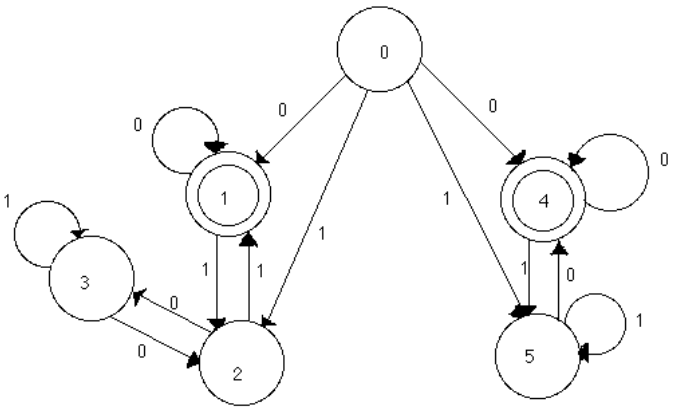
Epädeterministinen äärellinen automaatti on viisikko $M = (Q, \Sigma, \delta, p_0, F)$, jossa Q on tilojen joukko, Σ syöteaakkosto, $p_o \in Q$ automaatin alkutila, $F \subseteq Q$ lopputilojen joukko ja δ kuvaa tilojen välistä siirtymäfunktiota.

Gao et al. esittelivät artikkelissaan [GGM99] oman mallinsa epädeterministisyyden toteuttamiseksi DNA:n avulla. Menetelmä koostuu kolmesta vaiheesta:

1. *koodataan* automaatin siirtymät syötteillä DNA-molekyyleihin.

2. *pariutuminen* ja *yhteenliittyminen* (engl. ligation), jotka simuloivat tilasiirtymiä. Pariutumisella (engl. annealing tai hybridization) tarkoitetaan komplementaaristen yksinauhaisten DNA-ketjujen yhdistymistä. Komplementaarisuudella tarkoitetaan, että `C`:tä vastaa aina `G`, vastaavasti nauhalla `A`:ta vastaa aina `T`.)

3. eristetään tulos ja päätetään hyväksytäänkö syöte, vai ei.



Kuva 1: Eräs epädeterministinen äärellinen automaatti

Yokomori [Yok02] käyttää esimerkkinä kuuden tilan automaattia, joka on esitelty kuvassa 1. Ensimmäistä vaihetta varten automaatin tilaa kuvataan neljän merkin mittaisella koodilla seuraavasti:

tila 0	`atat`	tila 1	`ttat`
tila 2	`gctg`	tila 3	`ctca`
tila 4	`tatt`	tila 5	`gtcg`

Alkutilaa kuvaava koodi `atat` on liitetty kaksoisketjuun. DNA-ketjuun muodostuu näin avoin pää. Avoin pää (engl. sticky end) tarkoittaa molekyyliä, joka on valmis pariutumaan toisen molekyylin. kanssaJokainen siirtymä $p \rightarrow^a q$, jossa p, q ovat tiloja ja a on nolla tai yksi, koodataan molekyyliin erillisellä *pullistumalla*. Pullistuma

(engl. bulge loop) tarkoittaa tilannetta, jossa toinen ketju taittuu niin, että siihen tulee silmukka. Esimerkiksi ketjujen `AAAA` ja `TTCGCGTT` yhdistyessä siten, että `AAAA` ja `TTTT` pariutuvat, muodostaa väliin jäävä `CGCG` pullistuman.

Automaatin M suorittamaa laskentaa simuloidaan kolmessa vaiheessa: ensimmäisessä vaiheessä aloitetaan molekyylillä, jossa `atatCTTAA` on *avoin pää*. Oletetaan ensimmäisen käsiteltävän merkin olevan 0; kyseessä on siis siirtymä tilasta 0 tilaan 1 tai 4. Kaksi mahdollista `tata`-molekyyliä voivat pariutua `atat`:n kanssa. Pariutumisen jälkeen ne molekyylit, jotka eivät reagoineet kerätään talteen. Toisessa vaiheessa lisätään SmaI-entsyymiä, joka leikkaa molekyylin kahtia. Kolmantena lisätään *EcoRI-entsyymi*, joka tunnistaa oikean molekyylin ja leikkaa sen, samalla oikaisten pullistuman ja paljastaen tilasiirtymän. EcoRI ja SmaI ovat rajoiteentsyymejä (engl. restriction enzyme), jotka voivat leikata kaksisäikeisen molekyylin x: niin, että molekyyli katkaistaan ensimmäisen säikeen 4 ensimmäisen emäksen ja toisen säikeen 8 ensimmäisen *emäksen* kohdalta. Emäksellä puolestaan tarkoitetaan DNA-molekyylin rakenneosaa `A, T, G` tai `C`.

Juuri pullistuma mahdollistaa laskennan, sillä se estää väärän molekyylin yhdistymisen pariutumisvaiheessa. EcoRI-entsyymi paljastaa kätketyn pullistuman piilottaman tilasiirtymän ja suoristaa sen, jolloin molekyyli on taas muodossa `xxxxGAATT` (tässä tapauksessa `tataGAATT`). Tätä sykliä toistamalla voidaan automaatin M laskentaa simuloida kokonaisuudessaan. Menetelmällä on useita etuja: nykytilaa kuvaavavan molekyylin koko pysyy aina samana. Toinen etu on se, että reaktiot etenevät sykleissä ja siten mahdollistavat prosessin automatisoinnin. Kolmantena etuna on järjestelmän yleistyvyys; kun tilat ja niiden välisten siirtymien koodauksen on määritelty, voidaan luoda kirjasto kaikista tilasiirtymistä riippumatta koneen toteutuksesta.

3 Toteutuvuusongelma

Aina ei kuitenkaan tarvitse palauttaa laskentaa automaatiksi. DNA-molekyyleillä on useita hyviä ominaisuuksia, joita voidaan hyödyntää erilaisten ongelmien ratkaisussa. Eräs merkittävimpiä etuja on ns. polymeraasiketjureaktio (engl. polymerase chain reaction, PCR). Reaktion avulla voidaan kopioida DNA-ketjua lämpötilaa säätelemällä. Kopioiduista kappaleista voidaan seuraavalla kierroksella valmistaa yhä uusia kopioita ja näin saadaan rinnakkaisesti luotua eksponentiaalinen määrä identtisiä kopioita.

Toteutuvuusongelma (engl. satisfiability problem, SAT) on ongelma, jossa tehtävänä on ratkaista, onko olemassa sellainen määrittely muuttujille, että annettu lausekalkyylin kaava toteutuu. Ongelma on *NP-täydellinen*, mikä tarkoittaa, ettei ongelmaan tunneta yhtään tehokasta ratkaisualgoritmia. Sakamoto et al. esittelivät oman DNA-toteutuksensa, *SAT-Enginen*, vuonna 2000 [SGK00].

Esimerkki SAT-lauseesta on viidestä ehtolauseesta koostuva lause F, jossa on kolme literaalia.

$$\begin{aligned} F &= C_1 \cdot C_2 \cdot C_3 \cdot C_4 \cdot C_5 \\ &= (a \vee b \vee c) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c) \end{aligned}$$

Lause on *toteutuva*, jos löytyy sellainen liteeraalien yhdistelmä, joka toteuttaa jokaisen ehdon. *Toteutumattomaksi* sanotaan sellaista lausetta, jolle ei ole yhtään sen toteuttavaa yhdistelmää. Määritellään F:n *litereaalimerkkijono* $u_F = t_1t_2t_3t_4t_5$, jossa jokainen t_i on joku C_i:n literaali. Valituksi tulee siis yksi arvo jokaisesta ehdosta. Esimerkkejä u_F:stä ovat $aabb\neg c$ ja $\neg abb\neg bc$. Valitut arvot t_i voidaan tulkita siten, että $aabb\neg c$:sta tulee $(a, b, c) = (1, 1, 0)$. Kyseinen u_F toteuttaa lauseen F, ja on siten kelvollinen todiste lauseelle. Merkkijono $\neg abb\neg bc$ puolestaan on ristiriitainen, sisältäen sekä b:n että $\neg b$:n.

Algoritmi toteutuvuusongelman ratkaisuksi koostuu kolmesta osasta:

1. Kaavasta F, jossa on n ehtoa, tehdään kaikki mahdolliset n:n mittaiset u_F:t.
2. Poistetaan kaikki literaalimerkkijonot, joilla F ei toteudu.
3. Jos jäljellä on yhtään literaalimerkkijonoa, on F toteutuva. Jos taas kaikki literaalimerkkijonot poistettiin vaiheessa 2, tiedetään, ettei F ole toteutuva.

Ensimmäisessä vaiheessa k:n literaalin kaavasta tulee maksimissaan k^n literaalimerkkijonoa.

Algoritmin toteutus hyödyntää rinnakkaisuutta ja toimii seuraavasti:

1. Pariutetaan literaaleja kuvaavat DNA-molekyylit rinnakkain, jotta saadaan kaikki literaalimerkkijonot.
2. Koodataan literaalit a ja $\neg a$ komplementaarisiksi. Näin ristiriitaiset literaalimerkkijonot muodostavat ns. *hiuspinnirakenteen*. Hiuspinni (engl. hairpin) tarkoittaa yksisäikeistä DNA-ketjua, joka on yhdistynyt itsensä kanssa muodostaen hiuspinniä muistuttavan rakenteen.
3. Poistetaan kaikki literaalimerkkijonot, joissa on hiuspinnirakenne.

Jos jäljelle jäi literaalimerkkijonoja, on F toteutuva. Samalla saadaan kaikki sen ratkaisevat yhdistelmät.

Menetelmä on yksinkertaisuudessaan hämmästyttävä. Rinnakkain, polymeraasiketjureaktiota käyttäen, suoritettava kopiointi mahdollistaa huomattavan suurienkin lauseiden ratkaisemisen. Sakamoto et al. [SGK00] käyttivät algoritmiaan ratkaisemaan ongelman

$$\begin{aligned} F =&(a \vee b \vee \neg c) \wedge (a \vee c \vee d) \wedge (a \vee \neg c \vee \neg d) \wedge (\neg a \vee \neg c \vee d) \wedge (a \vee \neg c \vee e) \\ &\wedge (a \vee d \vee \neg f) \wedge (\neg a \vee c \vee d) \wedge (a \vee c \vee \neg d) \wedge (\neg a \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d) \end{aligned}$$

Literaalimerkkijonoja on yhteensä 3^{10}, eli 59049 kappaletta. Niiden joukossa on 24 F:n toteuttavaa literaalimerkkijonoa. Nämä 24 literaalimerkkijonoa vastaavat kuitenkin yhtä totuusjakaumaa, joka onkin ainoa käypä vastaus, eli $(a, b, c, d, e, f) = (0, 1, 1, 0, 1, 0)$.

4 Hiuspinnikielten laskentavoima

Aiemmissa luvuissa esiteltiin toteutuksia, joissa yksisäikeisen DNA-molekyylin taittumista hiuspinnimuoton käytettiin apuna laskennassa. Tässä luvussa pohditaan, kuinka ilmaisuvoimainen hiuspinnirakenne on.

Useat hiuspinnirakennetta käyttävät algoritmit ovat hyvin samantyyppisiä. Kuten toteutuvuusongelman ratkaisseessa algoritmissakin, aluksi tehdään paljon molekyylejä, joiden koodaus on toteutettu niin, että paritutumisvaiheen jälkeen voidaan tunnistaa halutut molekyylit juuri niiden muodostaman hiuspinnirakenteen avulla. Näin voidaan karsia halutut molekyylit pois, ja jäljelle jää ainoastaan tulos.

Morfismi h_w on *Watson-Crick -morfismi* (engl. Watson-Crick morphism), jos jokaiselle sanalle z on komplementtisana $\overline{z}$. `ATCG`:n komplementtisana olisi siis `TAGC`. Näin ollen $h(h(a)) = a$, ja jos $h(a) = b$, niin a ja b ovat komplementaarisia. Näin ollen myös $h(b) = a$. Merkintä $mi(v)$ puolestaan tarkoittaa v:n peilikuvaa. Käytännössä peilikuva tarvitaan, jotta x ja v voivat yhdistyä, sillä ne ovat samassa ketjussa. Taittuessa ketjun alku yhdistyy loppupään kanssa.

Määritellään, että uH_k tarkoittaa rajoittamatonta k:n asteen hiuspinnikieltä. Kieli uH_k koostuu viidestä mahdollisesta osasta, jotka ovat z, v, w, x ja y. Osat v ja x ovat komplementaarisia ja siten muodostavat parin. Osa z on ketjussa v:n jälkeen, w on mahdollinen v:n ja x:n väliin jäävä osa ja y on ketjun alussa. Rakenne voi siis muodostua kahdeksalla eri tavalla. Nämä kahdeksan eri mahdollisuutta on kuvattu

alla.

1. (u) $uH_k = \{zvwxy | z, v, w, x, y \in V^*, x = mi(h(v)), and |x| \geq k\}$,
2. (b) $bH_k = \{vwxy | v, w, x, y \in V^*, x = mi(h(v)), and |x| \geq k\}$,
3. (c) $cH_k = \{zvxy | z, v, x, y \in V^*, x = mi(h(v)), and |x| \geq k\}$,
4. (f) $fH_k = \{zvwx | z, v, w, x \in V^*, x = mi(h(v)), and |x| \geq k\}$,
5. (bc) $bcH_k = \{vxy | v, x, y \in V^*, x = mi(h(v)), and |x| \geq k\}$,
6. (bf) $bfH_k = \{vwx | v, w, x \in V^*, x = mi(h(v)), and |x| \geq k\}$,
7. (cf) $cfH_k = \{zvx | z, v, x \in V^*, x = mi(h(v)), and |x| \geq k\}$,
8. (bcf) $bcfH_k = \{vx | v, x \in V^*, x = mi(h(v)), and |x| \geq k\}$

Yokomori todistaa kielten (u), (b), (f), (c) ja (bf) olevan säännöllisiä, kun $k \geq 1$ [Yok02]. Kielet (bc), (cf) ja (bcf) ovat puolestaan lineaarisia, mutteivät säännöllisiä, kun $k \geq 1$. Kielistä, joissa ei ole osaa w, voidaan muodostaa muotoa $a^n b^n$ oleva kieli, joka ns. pumppauslemman nojalla ei ole säännöllinen.

Yokomori esittelee kaksi teoreemaa kielen (bcf) komplementeille.

1. Jos L on säännöllinen kieli, niin $L - bcfH_k$, $k \geq 1$ on lineaarinen kieli, muttei välttämättä säännöllinen.
2. Jos L on kontekstivapaa kieli, niin $L - bcfH_k$, $k \geq 1$ on kontekstista riippuva kieli, muttei välttämättä kontekstivapaa.

Kielten (bc) ja (cf) kohdalla tilanne on hieman toinen. Yokomori [Yok02] sanoo kielten $L - bcH_k$ ja $L - cfH_k$ sijoittumisen kieliperheisiin olevan vielä ratkaisematta.

Yokomori esittelee vielä ratkaisun suunnatun Hamiltonin kehän ongelmaan (engl. directed Hamiltonian path problem, DHPP). Ratkaisu pohjautuu jälleen kerran koodaukseen, jossa jokainen solmu on itsensä komplementti ja siten yhdistyy itsensä kanssa. Riittääkin muodostaa kaikki tietyn mittaiset ketjut, joiden alussa on haluttu alkusolmu ja lopussa polun päättävä solmu. Koodauksen ansiosta ketjut, joissa on kaksi samaa solmua, muodostavat hiuspinnirakenteen, joka tunnistetaan ja poistetaan. Jäljelle jää ainoastaan hyväksyttäviä polkuja.

Hiuspinnirakenne ja molekyylikoneet tarjoavat yksinkertaisuudestaan huolimatta, tai juuri siksi, lupaavia ratkaisumalleja moniin perinteisesti vaikeisiin laskennallisiin ongelmiin. Yokomori kuvaa vasta teoreettisia sovelluksia, mutta toteutus on mahdollista jo nykyisillä menetelmillä. Tarvitaan vielä tutkimusta, mutta jo nyt molekyylikoneiden tulevaisuus näyttää varsin lupaavalta.

5 Yhteenveto

Molekyylikoneiden tutkimus on vasta alussa, mutta tässä esiteltiin muutama lupaava tulos. Alan biologinen pohjakaan ei ole vielä vanhaa, ja on oletettavaa, ettei biologinen tutkimus aina tue tietojenkäsittelytiedettä täysin, mutta jo nyt molekyylibiologian tulosten avulla voidaan luoda DNA-ketjuista äärellinen automaatti. Myös erilaisilla hiuspinnirakenteella näyttää olevan paljon potentiaalia.

Tulokset ovat lupaavia, kunhan polymeraasiketjureaktiota saadaan parannettua. Parannuksessa automatisoinnilla on suuri osa. Puutteista ei kuitenkaan kannata lannistua, polymeraasiketjureaktio antaa huomattavia lupauksia rinnakkaisuudellaan, ja onkin ymmärrettävää, ettei biokemian tutkimus aina huomio laskennan teoriaa ja sen vaatimuksia. Uskon kuitenkin myös biokemian hyötyvän prosessin nopeuttamisesta, enkä koe DNA-ketjujen tarkistuksesta johtuvaa viivettä ylivoimaiseksi esteeksi molekyylikoneiden tiellä.

Molekyylikoneiden suurin lupaus onkin niiden mahdollistama rinnakkaisuus. Oikella koodauksella voidaan toteuttaa monenlaisia etsimisalgoritmeja ja suorittaa niitä rinnakkain, kuten toteutuvuusongelman ratkaissut algoritmi osoitti.

Lähteet

BPA01 Yaakov Benenson, Tamar Paz-Elizur, Rivka Adar et al. *Programmable and autonomous computing machine made of biomolecules* Nature 414 (2001) 430–434

GGM99 Gao Y, Garzon M, Murphy RC et al. *DNA implementation of nondeterminism* DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 48 (1999) 137–148

SGK00 K Sakamoto, H Gouzu, K Komiya et al. *Molecular computation by DNA hairpin formation* Science 288 (2000) 1223–1226

Yok02 Takashi Yokomori *Molecular computing paradigm - toward freedom from Turing's charm* Natural Computing 1 (2002) 333–390