Satunnaisalgoritmit

Topi Paavilainen

Laskennan teorian opintopiiri
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 23. helmikuuta 2014

1 Johdanto

Satunnaisalgoritmit ovat algoritmeja, joiden suorituksen määräämiseen käytetään satunnaislukuja. Yksinkertaisin malli satunnaisalgoritmien suorittamiseen on Turingin kone, jolla on kyky tuottaa satunnaislukuja joukosta $\{0, 1\}$, eli ikäänkuin kyky heittää kolikkoa. Tätä mallia kutsutaan propabilistiseksi Turingin koneeksi. Sen suoritus haarautuu jokaisessa satunnaisuutta vativassa kohdassa kahteen haaraan, joista edetään satunnaisluvun määräämään haaraan. Propabilistisella Turingin koneella on siis yksi tai useampi mahdollinen lopputila samalla syötteellä. Satunnaisalgoritmeista puhuttaessa ollaan usein kiinnostuneita sen virheen todennäköisyyttä. Virheen todennäköisyys tarkoittaa niiden suoritusten osuutta kaikista algoritmin suorituksista, joissa algoritmi antaa satunnaisuudesta johtuen väärän tuloksen.

Tämän kirjoitelman taustamateriaalina on Michael Sipserin kirja *Introduction to the Theory of Computation*[2].

2 Bounded-error Propabilistic Polynomial (BPP)

Bounded-error Propabilistic Polynomial, eli tästä lähtien BPP, on yleisin niin sanottujen tehokkaiden satunnaisalgoritmien luokka. Luokkaan kuuluvilla algoritmeilla väärän vastauksen todennäköisyys tulee olla vakio, joka on pienempi kuin $\frac{1}{2}$. Olennainen ero luokkaan kuulumattomiin satunnaisalgoritmeihin on siinä, että virheen todennäköisyys ei saa riippua syötteestä. Syötteestä riippumaton virheen todennäköisyys on vahva ominaisuus, joka mahdollistaa satunnaisalgoritmin muokkaamisen mielivaltaisen tarkaksi seuraavan lauseen perusteella.

Lause 1 (Amplification lemma)**.** *Olkoon $P \in BPP$ propabilistinen polynomiaikainen Turingin kone, jonka virheen todennäköisyys on $\epsilon < \frac{1}{2}$. Tällöin on olemassa propabilistinen polynomiaikainen Turingin kone P', jonka virheen todennäköisyys on $2^{-\epsilon}$.*

Todistus. BPP:hen kuuluvan satunnaisalgoritmin virhe saadaan mielivaltaisen pieneksi toistamalla algoritmia useita kertoja, ja valitsemalla yleisin tulos. Saatu Turingin kone on myös polynomiaikainen, sillä virhe on riippumaton syötteen koosta oletuksen perusteella. Siten kaikille syötteille riittää, että algoritmia toistetaan jonkun vakion verran, jotta saadaan halutun pieni virheen todennäköisyys. Yksityiskohtaisempi todistus lauseelle löytyy kirjasta [2]. □

Amplification lemma on merkittävä tulos tehokkaille satunnaisalgoritmeille. Virheen pienentäminen mielivaltaisen pieneksi polynomiaikaisuuden silti säilyessä mahdollistaa lähes täysin varmojen satunnaisalgoritmien tekemisen. Voidaan valita vaikka niin pieni virheen todennäköisyys, että algoritmin väärä tulos johtuu todennäköisemmin laiteviasta tietokoneessa

kuin algoritmin satunnaisuudesta. Toki niin pienellä virheellä todennäköisesti algoritmin suoritusaika voisi olla hyvin pitkä. Olennaista kuitenkin on, että se on polynominen suhteessa syötteen kokoon.

3 Esimerkki: alkuluvun testaaminen

Sen testaaminen, onko luku alkuluku vai ei, on ollut perinteisesti laskennallisesti hyvin haastava ongelma. Viime aikoina siihen on löydetty kuitenkin polynomiajassa toimiva algoritmi, mutta tämä vanhempi satunnainen versio on silti mielenkiintoinen esimerkki hyödyllisestä satunnaisalgoritmista. Algoritmi perustuu Fermatin pieneen teoreemaan. Sen mukaan kaikilla alkuluvuilla p ja sitä pienemmillä positiivisilla kokonaisluvuilla a pätee seuraava lause:

$$a^p \equiv 1 \bmod p$$

Algoritmin idea on ajaa Fermatin testiä haluttu määrä erilaisilla satunnaisesti valituilla a:n arvoilla. Jos yksikään testi ei epäonnistu, algoritmi sanoo luvun olevan alkuku. Jos yksikin testi epäonnistuu, algoritmi sanoo luvun olevan yhdistetty luku eli ei-alkuluku.

Algoritmi ei anna aina oikeaa tulosta, sillä löytyy yhdistettyjä lukuja, jotka läpäisevät Fermatin testit. Kuitenkin useimpien lukujen kohdalla suorittamalla enemmän testejä löytyy joku testitapaus, jossa Fermatin testi epäonnistuu. Oikean alkuluvun algoritmi luokittelee aina oikein, sillä alkuluku ei voi olla läpäisemättä Fermatin testiä.

4 Satunnaisalgoritmien vaativuusluokat

Edellisen esimerkin algoritmi kuuluu luokkaan *co-RP* (co- Randomized Polynomial time). Luokkaan kuuluvat algoritmit antavat varmasti oikean vastauksen, jos oikea vastaus on “hyväksy”. Jos taas oikea vastaus on “hylkää”, *co-RP*:hen kuuluvat ongelmat voivat antaa väärän vastauksen tietyllä virhemarginaalilla. Luokalla RP nämä ovat johdonmukaisesti toisin päin. *Co-RP*:llä ja RP:llä virhe on siis toispuoleinen. Toisessa puolikkaassa tapauksista algoritmi antaa oikean vastauksen ilman virheen mahdollisuutta.

Co-RP:n ja RP:n leikkausta kutsutaan ZPP:ksi (Zero Propability Polynomial time). Tämän luokan ongelmissa virheen mahdollisuutta ei ole kummallakaan oikean vastauksen vaihtoehdolla. ZPP:hen kuuluvat algoritmit voivat silti käyttää satunnaisuutta hyödyksi, mutta satunnaisuus ei vaikuta algoritmin lopputulokseen. Satunnaisuudella voidaan päättää esimerkiksi resurssien käytöstä. Näitä algoritmeja kutsutaan Las Vegas -algoritmeiksi. Esimerkki Las Vegas-algoritmista on quick sort (pahimman tapauksen aikavaativuus n^2, yleensä $O(n \log n)$), jossa jakopisteen valinta suoritetaan satunnaisesti deterministisen valinnan sijaan. Tämä valintatapa tekee todennäköisem-

mäksi sen, että quick sort suoriutuu järjestämisestä aina $O(n \log n)$ ajassa, vaikka syöte olisi mahdollisimman huono algoritmille.

Kaikkien satunnaisalgoritmien luokka on PP (Propabilistic polynomial). Siihen kuuluu niin ensimmäisenä mainittu luokka BPP, kuin RP, *co-RP* ja ZPP. PP:n määritelmän mukaan siihen kuluu algoritmit, jotka antavat oikean vastauksen todennäköisyydellä, joka on suurempi kuin $\frac{1}{2}$. PP kuulostaa hyvin samalta kuin osajoukkonsa BPP, mutta niillä on merkittävä ero. PP:ssä virhe voi riippua syötteen koosta, joten luokan PP algoritmeista ei aina ole lauseen 1 mukaisia polynomiaikaisia versioita.

5 P = BPP?

Satunnaisalgoritmeihin liittyy mielenkiintoinen avoin kysymys. Kysymys on, ovatko luokat P (polynomiaikaiset ei-propabilistiset Turingin koneet) ja BPP oikeastaan sama luokka? Jos ne olisivat, se tarkoittaisi, että kaikille BPP:n algoritmeille olisi olemassa polynomiaikainen deterministinen vaihtoehto. Kysymys on avoin, sillä on ongelmia, joihin tiedetään BPP:hen kuuluva ratkaisu, mutta ei polynomiaikaista determinististä ratkaisua. Ongelman ydin onkin se, että antaako satunnaisuus ja virheen salliminen tehokkaamman mallin laskennalle kuin deterministinen Turingin kone.

Useille satunnaisalgoritmeille on kuitenkin löydetty deterministinen vastine. Tästä on esimerkki edellä esitelty alkulukujen testaaminen. Ongelman kuuluminen luokkaan P oli pitkään avoin kysymys. Vuonna 2002 intialaiset tietojenkäsittelytieteilijät esittivät paperissaan polynomiaikaisen deterministisen algoritmin alkuluvun testaamiselle [1]. Täten oli osoitettu ongelman kuuluminen luokkaan P.

6 Yhteenveto

Satunnaisuuden salliminen algoritmeille antaa paljon uusia mahdollisia menetelmiä algoritmien suunnitteluun. Tässä kirjoitelmassa esiteltiin vain muutamia satunnaisalgoritmien sovelluksia, mutta moniin tärkeisiin ongelmiin löytyy satunnaisuutta hyödyntävä ratkaisu. Usein jopa oikean tuloksen todennäköisyys saadaan niin pieneksi, ettei satunnaisuus enää juuri vaikuta algoritmin lopputulokseen. Tästä päästäänkin edellisessä kappaleessa esiteltyyn ongelmaan P:n ja BPP:n suhteesta. Jos virhe saadaan kerran BPP:hen kuuluvilla algoritmeilla mielivaltaisen pieneksi polynomiaikaisuuden säilyessä, voisiko kaikkia BPP:n algoritmeja vastata polynomiaikainen deterministinen algoritmi?

Lähteet

[1] Agrawal, Manindra, Kayal, Neeraj ja Saxena, Nitin: *PRIMES is in P.* Ann. of Math, 2:781–793, 2002.

[2] Sipser, Michael: *Introduction to the Theory of Computation.* Thomson, 2. painos, 2006.