T-79.194 Tietojenkäsittelyteorian seminaari
Raportti
3.4.2003

Heikki Rantanen
hrantane@cc.hut.fi
48282D

Lauselogiikan toteutuvuusongelman probabilistisesta ratkaisemisesta

Tiivistelmä Lauselogiikan toteutuvuusongelma (k–SAT) on yksi tietojenkäsittelyteorian tutkituimmista ongelmista. Tämä raportti on katsaus ongelman erääseen epätäydelliseen ratkaisualgoritmiin, joka on luonteeltaan probabilistinen ja perustuu paikalliseen hakuun. Algoritmin on analysoinut Schöning (1999), jonka mukaan suoritusajan odotusarvo on luokkaa $O(1.3334^n)$. Algoritmia 3-SAT:in tapauksessa on myöhemmin parantanut Hofmeister et al. (2002), joka esittää tuloksen $O(1.3302^n)$.

1 Johdanto

*Lauselogiikan toteutuvuusongelma*ssa tehtävänä on löytää annetulle lauselogiikan lauseelle *F malli*, eli sellainen F:n atomilauseiden totuusjakelu, jolla F evaluoituu todeksi. Vaihtoehtoisesti tulee osoittaa, ettei mallia ole, eli että F on *toteutumaton*. F tulee esittää konjunktiivisessa normaalimuodossa (CNF), jolloin meillä on joukko klausuuleita, joissa kussakin esiintyy korkeintaa k literaalia ja puhumme k-SAT -ongelmasta (SAT=satisfiability). Ongelma on merkityksellinen, koska lauselogiikalla on useita sovelluskohteita. Itseasiassa 3-SAT tiedetään NP-täydelliseksi.

Ongelmaan on esitetty lukuisia ratkaisualgoritmeja, joita osa on *deterministisiä* ja osa *probabilistisia*. Algoritmi on probabilistinen, mikäli sen tulokseen tai suoritusaikaan samalle syötteelle liittyy satunnainen epävarmuustekijä eri toistojen kesken. Muutoin algoritmi on deterministinen. Tyypillisesti kun on tehtävä valinta joukosta vaihtoehtoja, deterministinen algoritmi päättelee oikean vaihtoehdon tai tutkii järjestelmällisesti ne kaikki, ja probabilistinen algoritmi valitsee satunnaisesti jonkin niistä.

Olettaen n atomilausetta naiivi deterministinen algoritmi k-SAT ongelmaan on kokeilla jokaista 2^n totuusjakelua järjestyksessä, jolloin suoritusaika on luokkaa $O(2^n)$. Naiivi probabilistinen algoritmi puolestaan olisi kokeilla totuusjakeluita satunnaisesti niin, että kukin niistä valitaan aina samalla todennäköisyydellä $\frac{1}{2^n}$. Tarvittavien kokeilujen määrä on tällöin geometrisesti jakautunut satunnaismuuttuja

odotusarvona 2^n, joten myös naiivin probabilistisen algoritmin suoritusaika on luokkaa $O(2^n)$.

Lähteessä [1] on esitetty k-SAT ongelmalle probabilistinen ratkaisualgoritmi, joka perustuun ratkaisun paikalliseen hakemiseen ja on luonteeltaan epätäydellinen. Lähteessä [2] on tätä perusalgoritmia 3-SAT:in tapauksessa hieman parannettu aikavaativuuden suhteen. Perusalgoritmia voidaan käyttää myös CSP-ongelmaan (Constrain Satisfaction Problem), jota voidaan ajatella toteutuvuusongelman yleistyksenä, mutta tässä raportissa keskitytään ainoastaan k-SAT -ongelmaan. Esiintyvät symbolit ja nimet on pyritty pitämään yhtenevinä em. lähteisiin.

2 Perusalgoritmi

2.1 Toimintaperiaate

Seikkaa, että lause F on CNF-muodossa, hyväksi käyttämällä voidaan hyvin yksinkertaisella probabilistisella algoritmilla [1] päästä naiivia tapaa huomattavasti parempiin tuloksiin. Aluksi valitaan satunnaisesti totuusjakelu a. Mikäli a ei toteuta lausetta, on ainakin yksi klausuuli saanut arvon epätosi. Valitaan jokin tällainen klausuuli ja sieltä satunnaisesti yksi atomilause, jonka arvo vaihdetaan. Tätä toistetaan, kunnes lause toteutuu. Mikäli F on toteutumaton, jäädään siis ikuiseen silmukkaan ja näin ollen algoritmi on epätäydellinen. Schöningin[1] mukaan algoritmi kannattaa käynnistää uudelleen $3n$ askeleen jälkeen, mikäli mallia ei ole löytynyt. Algoritmin pseudokoodi on esitetty kuvassa 1.

```
procedure RandomWalk ()
    repeat 3n kertaa
        if F(a) = true
            return true
        C := jokin F : n toteutumaton klausuuli
        t := satunnaisesti valittu C : n atomilause
        a := totuusjakelu joka seuraa vaihdettaessa t : n arvo
    end
    return false
end

program Search (CNF F)
    repeat
        valitse satunnaisesti totuusjakelu a ∈ {0, 1}^n
    until RandomWalk(a) = true
    return a
end
```

Kuva 1. Schöningin algoritmi pseudokoodina.

2.2 Vaativuusanalyysi

Oletetaan, että F on toteutuva. Algoritmin vaatima aika mallin löytämiseen on suoraan verrannollinen otettujen askelten määrään, joka on siis satunnaismuuttuja, sanotaan Y. Aikavaatimuksen kannalta pahin tapaus on se, että malleja on vain yksi, joten olkoon a^* F:n ainoa malli.

Proseduuri RandomWalk löytää mallin jollain todennäköisyydellä p. Proseduurin eri suoritukset ovat toisistaan riippumattomia, joten tarvittavien proseduurikutsujen määrä on geometrisesti jakautunut satunnaismuuttuja Z odotusarvona $E[Z] = 1/p$. Näin ollen pätee

$$\begin{aligned} 3n(Z-1) \le Y &\le 3nZ \\ E[3n(Z-1)] \le E[Y] &\le E[3nZ] \\ 3n(E[Z]-1) \le E[Y] &\le 3nE[Z] \\ 3n/p - 3n \le E[Y] &\le 3n/p. \end{aligned} \tag{1}$$

Laskemme seuraavaksi p:n. Merkitään X_t:llä a:n ja a^*:n välistä Hammingin etäisyyttä, eli arvoltaan poikkeavien bittien (atomilauseiden) lukumäärää, t:nnen askeleen jälkeen. Mikäli a ei toteuta lausetta, eli $X_t > 0$, tiedetään, että toteutumattoman klausuulin k:sta atomilauseesta jonkun arvo on vaihdettava. Valinta osuu oikeaan todennäköisyydellä vähintään $\frac{1}{k}$. Tarkestellaan edelleen pahinta tapausta; on siis voimassa

$$\begin{gathered} P(X_{t+1} = j-1 \mid X_t = j) = \tfrac{1}{k} \\ P(X_{t+1} = j+1 \mid X_t = j) = \tfrac{k-1}{k}, \quad 0 < j < n, \\ \text{sekä} \\ P(X_{t+1} = 0 \mid X_t = 0) = 1, \\ P(X_{t+1} = n-1 \mid X_t = n) = 1. \end{gathered} \tag{2}$$

Proseduuria RandomWalk voidaan nyt tarkastella satunnaiskulkuna $\{X_t,\ t = 0, 1, \ldots, 3n\}$. Sille pätee *Markovin ominaisuus* [3] $P(X_{t+1} = x_{t+1} \mid X_0 = x_0, \ldots, X_t = x_t) = P(X_{t+1} = x_{t+1} \mid X_t = x_t)$, joten se on *Markovin ketju.* Ketjulla on aikahomogeeniset siirtotodennäköisyydet $P(X_{t+1} = i \mid X_t = j) = p_{i,j}$ (2) ja yksi absorboiva tila $X_t = 0$. Ketjun alkujakauma on binomijakauma parametrilla n, $\frac{1}{2}$ eli

$$P(X_0 = j) = \binom{n}{j} \left(\tfrac{1}{2}\right)^n,\ j = 0 \ldots \mathrm{n}. \tag{3}$$

Olkoon nyt $n = 4$. Markovin ketjun tilansiirtomatriisi $P = (p_{i,j})$ on tällöin

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{k-1}{k} & 0 & \frac{1}{k} & 0 & 0 \\ 0 & \frac{k-1}{k} & 0 & \frac{1}{k} & 0 \\ 0 & 0 & \frac{k-1}{k} & 0 & \frac{1}{k} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4}$$

Olkoon v, q vaakavektoreita niin että $v_j = P(X_0 = j)$ kuten kaavassa (3) ja $q_j = P(X_{3n} = j)$. Ne siis esittävät ketjun alku- ja loppujakaumia. Tällöin lähteen [3] perusteella on voimassa

$$q = v\,P^{3n}. \tag{5}$$

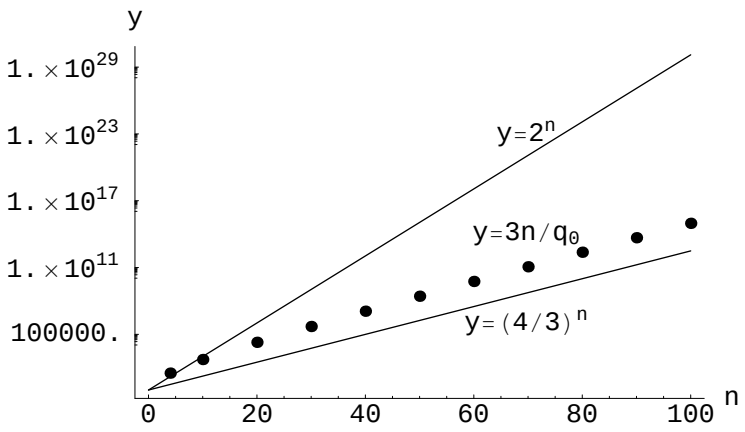
Olkoon $k = 3$ jolloin

$$v = (\tfrac{1}{16}\ \tfrac{1}{4}\ \tfrac{3}{8}\ \tfrac{1}{4}\ \tfrac{1}{16}),\ P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{2}{3} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{6}$$

$$q = v\,P^{12} \approx (0.3636\ 0.0425\ 0.1234\ 0.2810\ 0.1895).$$

Todennäköisyys että proseduuri RandomWalk löytää mallin on siis $q_0 \approx 0.3636$ kun $n = 4$, $k = 3$. Kaavalla (1) $E[Y] \le 3n/q_0 \approx 33$. Näitä arvoja eri n:illä kun $k = 3$ on esitetty kuvassa 2.

n	E[Y]
4	33.0052
10	549.892
20	19289.5
30	508184.
40	1.19759×10^7
50	2.65294×10^8
60	5.64819×10^9
70	1.16969×10^{11}
80	2.37342×10^{12}
90	4.7411×10^{13}
100	9.35423×10^{14}



Kuva 2. Perusalgoritmin askelmäärän odotusarvot kun atomilauseiden määrä 3-SAT -ongelmassa on n.

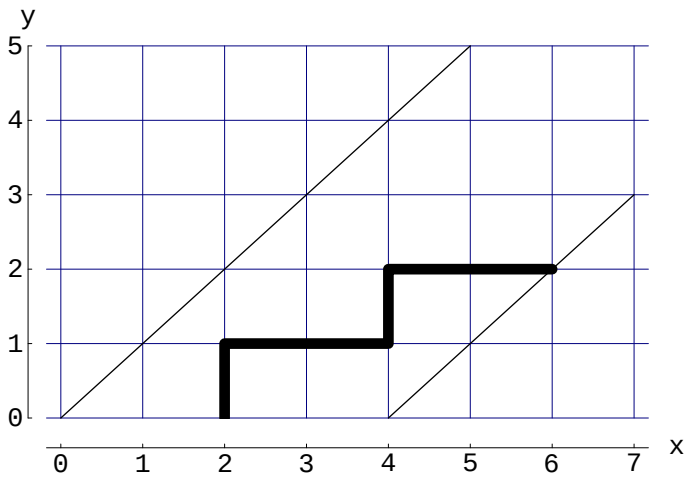
2.3 Schöningin tulos $O(1.3334^n)$

Edellä ollut analyysi antaa jotain kuvaa algoritmin toimivuudesta. Yleisen tapauksen analyysin vaikeus on siinä, että pitäisi tietää p:n lauseke n:n ja k:n funktiona. Schöningin[1] tapa lähestyä ongelmaa ei perustu Markovin ketjuun, vaikka hän toteaakin algoritmin olevan sellainen.

Satunnaiskulun $\{X_t, t = 0, 1, \ldots, 3n\}$ realisaatio $\{x_0, \ldots, x_{3n}\}$ voidaan esittää koordinaatistossa nk. *lattice path* -polkuna. Polun alkupiste on x−akselin piste $(n - x_0, 0)$ ja siirtymä tilasta x_t tilaan $x_t - 1$ vastaa liikkumista yhden ruudun oikealle ja siirtymä tilaan $x_t + 1$ liikkumista yhden ruudun ylös. Kuvassa 3 on tällä tavoin esitetty realisaatio $\{2\,3\,2\,1\,2\,1\,0\}$. Esittääkseen jotain realisaatiota on ruudukkopolun pysyttävä suorien $y = x$ ja $y = x - n$ välissä. Onnistunutta realisaatiota esittävät polut päättyvät kosketukseen jälkimmäisen kanssa.

Onnistumistodennäköisyys kun $X_0 = v_j$ saadaan laskemalla yhteen onnistuneiden polkujen lukumäärät pisteestä $(v_j, 0)$ pisteisiin $(n, 0)$, $(n + 1, 1)$, …, $(2n + \lfloor \frac{v_j}{2} \rfloor, n + \lfloor \frac{v_j}{2} \rfloor)$, ja jakamalla niiden summa kaikkien mahdollisten polkujen lukumäärällä. Ongelmana tässä on puolestaan se, että kirjallisuus ei ilmeisesti tunne ratkaisua tällaiseen lattice path -ongelmaan.

Schöningin analyysissä rajaa $y = x$ ei ole otettu huomioon. Lisäksi polut lasketaan vain pisteeseen $(2n - v_j, n - v_j)$ asti. Kumpaakaan yksinkertaistusta ei perustella mitenkään ja välivaiheita on muutenkin niukasti. Lisäksi tehdään eräs kolmas aika karkealta tuntuva approksimointi. Kaikki nämä nostavat vaativuuden ylärajaa ylöspäin, ja siitä huolimatta saadaan tulos $\text{poly}(n) \cdot O(1.3334^n)$, joka on puhtaalta kasvunopeudeltaan hämmästyttävän hyvä. Kuvan 2 perusteella voisi arvata yksinkertaistusten vaikuttavan vain polynomiseen osaan, mutta tätä ei ole paperissa mainittu saati todistettu.



Kuva 3. Satunnaiskulun $\{X_t\}$ realisaatio $\{2\,3\,2\,1\,2\,1\,0\}$ esitettynä koordinaatistossa.

3 Parannettu algoritmi

Kaksi klausuulia C ja C' ovat keskenään *riippumattomia*, joss. niissä ei esiinny samoja atomilauseita. Merkitään C:llä sellaista F:n klausuulijoukon osajoukkoa, jossa kaikki klausuulit ovat keskenään riippumattomia ja jonka laajentaminen yhdelläkin klausuulilla rikkoisi tämän vaatimuksen. Näin ollen C ei ole yksikäsitteinen, ja sellainen voidaan muodostaa ottamalla jokin klausuuli, jota joukkoa laajennetaan kunnes vaatimus rikkoutuisi. Hofmeister et al.[2] on esittänyt Schöningin algoritmiin parannuksen hyödyntäen tätä joukkoa C.

Olkoon $C = \{C_1, \ldots, C_{\hat{m}}\}$. Lauseen F:n atomilauseet voidaan nimetä uudelleen niin, että $C_1 = x_1 \vee x_2 \vee x_3$, …, $C_{\hat{m}} = x_{3\hat{m}-2} \vee x_{3\hat{m}-1} \vee x_{3\hat{m}}$. Nyt tiedämme jo ennalta, että esim. totuusjakelu jossa $x_1 = x_2 = x_3 =$ false ei voi olla malli. Kuitenkin mikäli nyt valittaisiin totuusjakelu niin, että $P(x_i = \text{true}) = \frac{1}{2}$, olisi tämän tapahtuman todennäköisyys $\frac{1}{8}$. Valitaan siis atomilauseiden $x_1, \ldots, x_{3\hat{m}}$ arvot niin että C toteutuu — loput $x_{3\hat{m}+1}, \ldots, x_n$ voidaan valita kuten aikaisemminkin. Parannettu algoritmi on esitetty kuvassa 4.

```
procedure Ind-Clauses-Assign(p1, p2, p3) # tulee olla 3 p1 + 3 p2 + p3 = 1
    a ∈ {0}^n
    for i = 0 … m̂ − 1
        Aseta a : ssa (x_{3i+1}, x_{3i+2}, x_{3i+3}) = (k, l, m) niin, että
        P( (k, l, m) ∈ {(0, 0, 1), (0, 1, 0), (1, 0, 0)} ) = p1,
        P( (k, l, m) ∈ {(0, 1, 1), (1, 0, 1), (1, 1, 0)} ) = p2,
        P( (k, l, m) = (1, 1, 1) ) = p3.
    end
    aseta satunnaisesti a : ssa (x_{m̂+1}, …, x_n) ∈ {0, 1}^{n−m̂}
    return a
end

procedure Search (CNF F)
    build C
    relabel atoms of F according to C
    repeat
        a = Ind-Clauses-Assign (1/7, 1/7, 1/7)
    until RandomWalk() = true
    return a
end
```

Kuva 4. Parannettu Schöningin algoritmi pseudokoodina (3−SAT).

Esitetty parannus vaikuttaa Markovin ketjun alkujakaumaan siirtäen sen painopistettä lähemmäs tilaa nolla. Jakauman laskeminen on kuitenkin mutkikasta. Schöningin p:tä koskeva tulos, jota siis en pystynyt varmentamaan, antaa kuitenkin mahdollisuuden laskea parannuksen vaikutus puhtaaseen kasvunopeuteen.

Hofmeister et al. esittää myös kokonaan toisenlaisen 3-SAT -algoritmin. Voidaan nimittäin ajatella kiinnitettävän aluksi vain atomilauseiden $x_1, \ldots, x_{3\hat{m}}$ arvot loppujen ollessa edelleen muuttujan asemassa. Jäljelle jäävät siis ne klausuulit, jotka eivät kuulu C:hen, ja kussakin niistä esiintyy vähintään yksi vakio C:n määritelmän mukaisesti. Kun näin saadulle kaavalle suoritetaan vakioiden poisto, jää jäljelle 2-SAT ongelma, johon tunnetaan polynominen ratkaisualgoritmi [4]. 3-SAT ongelmalle löytyy nyt malli, vain jos 2-SAT ongelmalle löytyy malli jollain C:n toteuttavalla $(x_1, \ldots, x_{3\hat{m}})$ arvolla, joita on $7^{\hat{m}}$ kappaletta. Koska $\hat{m} \leq \frac{n}{3}$ on näin saadun algoritmin vaativuus $7^{\hat{m}}\,\mathrm{poly}(n) \leq 7^{n/3}\,\mathrm{poly}(n)$. Riittävän pienellä $\hat{m}$, joka raja voidaan helposti laskea, se on aikaisemmin kuvattuja algoritmeja tehokkaampi. Käyttäen $\hat{m}$:n arvosta riippuen sitä tai em. parannetua probabilistista algoritmia päästään Hofmeister et al.:n mukaan vaativuuteen $O(1.330258^n)$.

Viimeisellä Hofmeister et al.:n parannuksella saadaan kasvunopeutta vielä hitusen alaspäin lukemaan $O(1.330193^n)$. Merkitään m_i:llä, $i = 1, 2, 3$, niiden klausuulien lukumäärää C:ssä, joissa tasan i atomilauseen arvo on tosi. Syy proseduurissa Ind-Assign-Clauses esiintyvien todennäköisyyksien parametrointiin on se, että jos tuntisimme m_i:den arvot ennalta, kannattaisi todennäköisyyksinä käyttää arvoja $m_i / \hat{m}$. Mahdollisuuksia m_i:den arvoiksi on kuitenkin vain polynominen määrä, ja proseduuria voidaan siksi kutsua niillä kaikilla vaikutuksen puhtaaseen kasvunopeuteen ollessa sama, kuin kutsuttaisiin vain kerran oikeilla arvoilla. Tällaisella spekuloinnilla ei tietenkään ole mitään relevanssia implementaation kannalta. Parannus kasvunopeudessakin on vain hienoinen.

4 Yhteenveto ja pohdinnat

Hyvin yksinkertaisella ja intuitiivisella probabilistisella k-SAT algoitmilla on päästy erinomaiselta vaikuttavaan aikavaativuuteen, kun puhutaan puhtaasta kasvunopeudesta. Vaativuuden laskeminen täsmällisesti on kuitenkin erittäin mutkikasta ja tuloksen $O(1.3334^n)$ johdossa [1] on aukkoja. Täsmällinen aikavaatimuksen odotusarvo voidaan laskea myös standardilla Markovin ketjulla kiinteällä n. Näin saadut tulokset (kuva 2) näyttävät pitävän yhtä em. Schöningin kasvunopeustuloksen kanssa, mutta itse askelmäärät ovat hyvin suuria.

Voidaanko sitten deteminististä ja probabilistista (odotusarvoa) vaativuutta verrata? Vastaus yhdellä sanalla on kyllä, mutta käytännössä asia riippuu probabilistisen suoritusajan hajonnasta. Mikäli se on suuri voi probabilistisen algoritmin käyttö muodostua vaivalloisemmaksi kuin odotusarvo ilmaisee.

Esitetyt algoritmit olettavat lauseen CNF-muodon, joten implementaatiossa tarvitaan myös muunto tähän muotoon. Käyttämällä lisäksi tehokasta CNF-sievennintä on mahdollista toteuttaa käyttökelpoinen mallintarkistin.

Samaa algoritmia voidaan käyttää myös probabilistisena UNSAT-algoritmina asettamalla suoritusajalle jokin katto. Epävarmuus olisi tällöin suoritusajan sijasta vastauksen oikeellisuudessa. Korottamalla tätä kattoa saadaan epävarmuus mielivaltaisen pieneksi.

Lähdeviitteet

[1] Schöning, U.: A Probabilistic Algorithm for *k*-SAT and Constraint Satisfaction Problems, in Proc. of the 40th Ann. IEEE Symposium on Foundations of Computer Science (FOCS99), 410-414, 1999

[2] Hofmeister, T.; Schöning, U.; Schuler, R.; Watanabe, O.: A probabilistic 3-SAT algorithm further improved, in Proc. of the 19th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 2285, 192-202, 2002

[3] Laininen, P.: Mat-2.111 Stokastiset Prosessit Luennot ’93, Edita Opetusmonisteet

[4] Aspvall, B; Plass, M.F.; Tarjan, R.E.: A linear-time algorithm for testing the truth value of certain quantified Boolean formulas, Information Processing Letters, 8(3), 121−123,1979