Lauselogiikan toteutuvuusongelman ratkaiseminen laskennallisessa gridissä

Antti E. J. Hyvärinen
Teknillinen korkeakoulu
Tietojenkäsittelyteorian laboratorio
Antti.Hyvarinen@tkk.fi

1 Johdanto

Laskennalliset gridit ovat vakiinnuttaneet asemansa tiedon arkistoinnin ja luetteloinnin sovelluksissa. Näistä hyvänä esimerkkinä toimii Google^{TM}-hakukone [4]. Grid-ympäristö on kuitenkin edullisuutensa vuoksi houkutteleva vaihtoehto myös laskennallisesti vaativalle tietojenkäsittelylle. Tunnetuimpia menestystarinoita grid-laskennan soveltamisesta on Seti@Home [17]. Gridiin liittyvät pitkät maantieteelliset etäisyydet ja laaja laitekanta, ja näistä johtuvat viestintäviiveet sekä suuri vikautumistodennäköisyys asettavat kuitenkin erityishaasteita algoritmeille, joiden suoritus riippuu voimakkaasti syötteestä.

Lauselogiikan toteutuvuusongelmassa [15] (SAT) kysytään, onko annetulle boolen funktiolle olemassa toteuttavaa totuusjakelua. Ongelma kuuluu NP-täydellisten ongelmien luokkaan [7]. Näille ongelmille ei tunneta yleistä tehokasta determinististä ratkaisualgoritmia, ja kaikkien tunnettujen algoritmien ajoaika on pahimmassa tapauksessa eksponentiaalinen syötteen pituuden suhteen.

SAT-ongelmatapauksilla on keskeinen asema monissa käytännön ongelmissa. Lauselogiikka on luonnollinen esitysmuoto esimerkiksi suunnittelu- ja verifiointiongelmille, ja näille aloille löytyykin runsaasti SAT-pohjaisia sovelluksia, esimerkiksi [2, 3, 12, 18]. SAT-toteutuvuustarkistimia on käytetty myös on-line ohjeistuksen konsistenssin tarkistuksessa [16], piirien testitapausten luonnissa [13] ja monissa muissa hakutehtäviksi kuvattavissa ongelmissa. SAT-tarkistinten tehokkuuden kasvu on ilmeistä, jos seurataan vuosittain järjestettävää SAT-kilpailua [6], ja tätä taustaa vasten onkin ymmärrettävää, että rajoiteohjelmat [14] ja monet ennen binäärisillä päätösdiagrammeilla [5] kuvatut ongelmat voidaan nykyään ratkaista tehokkaammin SAT-tarkistimilla [1, 10].

Tutkimuksen tavoitteena on kehittää menetelmiä ratkaista laskennallisesti haastavia lauselogiikan toteutuvuusongelmatapauksia hyödyntäen laskennallisia gridejä. Tätä tarkoitusta varten työssä [11] esitellään *sirottaminen*, joka on hajautusmenetelmä ympäristöön, jossa kommunikointi on rajoitettua. Sirottaminen soveltuu käytettäväksi minkä tahansa SAT-tarkistimen kanssa, mukaanlukien kaupalliset suljetun lähdekoodin tarkistimet. Lisäksi menetelmä mahdollistaa oman erillisen sirotusheuristiikan käytön osaongelmien luonnissa. Tässä tiivistelmässä hajautusmenetelmän käytännön toimivuutta testataan tuotantokäytössä olevalla gridillä erityyppisillä lauselogiikan ongelmilla, kuten tekijöihinjako, kryptoanalyysi ja satunnaisesti luodut ongelmat. Lisäksi hajautusmenetelmän avulla ratkaistaan joitakin tiettävästi ennen ratkaisemattomia SAT-toteutuvuusongelmatapauksia.

Koska sirottamisen soveltuvuus gridympäristöön riippuu menetelmän kyvystä toimia tehokkaasti suuriviiveisessä ympäristössä ja kyvystä sietää vikautuvia laskentoja, alkuperäisessä työssä [11] esitellään lisäksi erilaisia lähestymistapoja viiveiden minimointiin ja mitataan lähestymistapojen vaikutusta kokonaisajoaikaan.

2 Perusteet

Lauselogiikan kaavat voidaan esittää *konjunktiivisessa normaalimuodossa*: *literaalien*, positiivisten tai negatiivisten boolen muuttujien, disjunktioina, eli *klausuuleina*, jotka on yhdistetty konjunktioksi. Normaalimuoto on yleisesti käytössä oleva tapausten esitysmuoto lauselogiikan toteutuvuustarkistimille. Lauselogiikan toteutuvuustarkistimet, eli SAT-tarkistimet, vastaavat kysymykseen, onko annetussa kaavassa esiintyville muuttujille olemassa totuusarvot siten, että koko kaava tulee todeksi. Käytännössä tämä tapahtuu joko etsimällä toteuttava totuusjakelu jos sellainen on olemassa tai osoittamalla, että toteuttavaa totuusjakelua ei ole olemassa. Esimerkiksi lauselogiikan kaavalle

$$\begin{aligned} F = \; & (x_2 \vee \bar{x}_5) \wedge (x_1 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \\ & \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (x_5) \end{aligned} \tag{1}$$

löytyy kaksi toteuttavaa totuusjakelua, joista toinen on $x_1 = T, x_2 = T, x_3 = E, x_4 = E, x_5 = T$ ja toinen $x_1 = T, x_2 = T, x_3 = E, x_4 = T, x_5 = T$.

Useimmat SAT-ratkaisumenetelmät pohjautuvat Davis-Putnam-Logemann-Loveland -algoritmiin [8] (DPLL).

3 Ratkaiseminen gridissä

Tällä hetkellä tarjolla olevat gridympäristöt, kuten NorduGrid (`http://www.nordugrid.org/`) [9], on suunniteltu tyypillisesti fysiikan sovelluksiin. Nämä sovellukset vaativat vähän kommunikaatiota töiden välillä, runsaasti levytilaa ja muistia, sekä suurta laskentatehoa. Grid-teknologia on hyvin tuoretta, mistä seurauksena laskentojen vikaantumistodennäköisyys on huomattava. Ympäristö poikkeaa siis monin osin perinteisestä rinnakkaislaskentaympäristöstä, mikä on huomioitava hajautettua SAT-algoritmia kehitettäessä. Työssä kehitetään hajautusalgoritmi nimeltä *sirotus* (engl. *scattering*), joka mahdollistaa ongelmaan mukautuvan algoritmin mutta ei vaadi suoraa kommunikaatiota töiden välillä.

3.1 Sirotusalgoritmi

Sirottaminen pyrkii jakamaan SAT-ongelman useampaan SAT-ongelmaan siten, että niiden tuloksista pystyy päättelemään alkuperäisen ongelman ratkaisun, ne ovat helpompia ratkaista kuin alkuperäinen ongelma ja joiden ratkaisuavaruudet ovat erillisiä. Jotta menetelmä olisi skaalautuva, täytyy sirottamisen aikana ratkeavien ongelmien vaikuttaa sirotuksen etenemiseen. Eräs menetelmä tuottaa tällaisia ongelmia on muodostaa lauselogiikan kaavoja $S_1, \ldots, S_n$ siten, että

(*i*) $S_1 \vee \cdots \vee S_n$ on tosi kaikissa totuusjakeluissa, ja

(*ii*) $\forall i \neq j$: kaikilla totuusjakeluilla P pätee, että jos S_i on tosi P:ssä, S_j ei ole tosi P:ssä, ja muodostaa uusia aliongelmia F_i siten että $F_i = F \wedge S_i$. Tämän *sirotusaskeleen* jälkeen syntyneitä aliongelmia voidaan ratkoa ja niiden tulosten perusteella tehdä uusia sirotusaskelia aliongelmille. Erityisesti, mikäli jokin ongelma osoittautuu toteutumattomaksi, kaikki siitä sirottamalla saatavat aliongelmat tiedetään toteutumattomiksi, ja jos jollekin ongelmalle F_i löytyy ratkaisu, se on ratkaisu myös ongelmalle F.

Työssä käytetty sirotusmenetelmä muodostaa ensimmäisessä vaiheessa aliongelmat $F_1, \ldots, F_n$ SAT-ongelmatapauksesta F siten, että

$$F_i = \begin{cases} F \wedge T_1 & \text{jos } i = 1 \\ F \wedge \neg T_1 \wedge \cdots \wedge \neg T_{i-1} \wedge T_i & \text{jos } 1 < i < n \\ F \wedge \neg T_1 \wedge \cdots \wedge \neg T_{n-1} & \text{jos } i = n, \end{cases} \tag{2}$$

missä T_i:t ovat konjunktioita $(l_1^i) \wedge \cdots \wedge (l_{d_i}^i)$ ja negatoitu kaava $\neg T_i$ on $\neg T_i = (\bar{l}_1^i \vee \cdots \vee \bar{l}_{d_i}^i)$. Nyt siis $S_1 = T_1$, $S_i = \neg T_1 \wedge \cdots \wedge \neg T_{i-1} \wedge T_i$, kun $2 \le i \le n-1$ ja $S_n = \neg T_1 \wedge \cdots \wedge \neg T_{n-1}$. Literaalit l_i^j valitaan samantapaisesti kuin DPLL-algoritmin haarautumisliteraalit.

Koska SAT-ongelman ratkaisemiseen kuluvaa aikaa on vaikea arvioida etukäteen, staattinen aliongelmiin pilkkominen ei ole yleisessä tapauksessa riittävää. Niinpä yksittäisessä sirotusaskeleessa syntyneitä ongelmia sirotetaan uudestaan siten, että muodostuu *sirotuspuu*. Sirotuspuun juuressa on alkuperäinen ratkaistava ongelma F, ja yksittäisen solmun lapsia ovat solmuun liittyvän SAT-ongelman sirottamisesta syntyvät aliongelmat. Näin muodostuvaan puuhun voi tehdä mielivaltaisen poikittaisen *leikkauksen* $\mathcal{F}$, ts. solmujen joukon jonka jäsenet eivät ole vanhemmuussuhteessa toisiinsa ja jokainen polku puun juuresta lehteen kulkee täsmälleen yhden leikkaukseen kuuluvan solmun kautta. Esimerkki sirotuspuusta, johon on merkitty harmaalla eräs leikkaus, on esitetty kuvassa 1. Hausta saadaan dynaaminen etsimällä sopiva leikkaus kokeilemalla ongelmien ratkaisua grid-ympäristössä, ja kun leikkaus on löytynyt saadaan tulos luettua leikkaukseen kuuluvien ongelmien tuloksista.

3.2 Toteutus

Työssä käytetty sirotusalgoritmi pohjautuu DPLL-algoritmiin. Jotta syntyneet aliongelmat olisivat saman kokoisia, täytyy kaavassa (2) konjunktiossa T_i kiinnitettävien muuttujien määrää d_i säätää sen mukaan kuinka monta aliongelmaa on jo luotu alkuperäisestä ongelmasta. Olkoon $t(F)$ funktio kaavalta F sen ratkaisuun kuluvalle ajalle. Kun ongelma jaetaan n:ään yhtäsuureen aliongelmaan, tulee yhden ongelman ratkaisuajan olla $t(F)/n$. Jos oletamme että ongelmat eivät ole päällekäisiä, on i:nnen osaongelman luonnissa käytettävissä ratkaisuaika $t(F) - (i-1)t(F)/n$. Tästä saadaan johdetuksi yhtälö

$$\frac{t(F)}{n} = \left(t(F) - (i-1)\frac{t(F)}{n} \right) r_i.$$

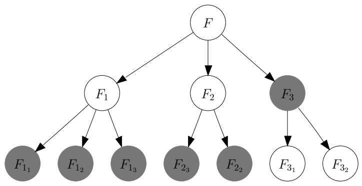
Ratkaisemalla tästä skaalaustermi r_i saadaan

$$r_i = \frac{1}{n-i-1}.$$

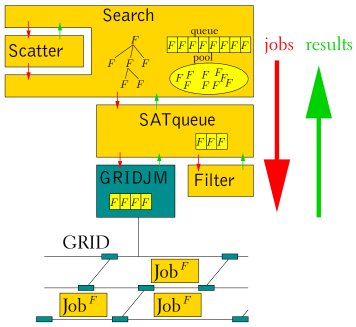
Jotta r_i:tä pystytään arvioimaan kiinnittämällä literaaleja kaavassa (2), tehdään oletus, että yhden literaalin kiinnittäminen pienentää DPLL-algoritmin läpikäymän hakuavaruuden puoleen. Jotta literaalien määrä d_i aproksimoisi r_i:tä mahdollisimman tarkkaan, valitaan d_i siten, että erotus $|r_i - \frac{1}{2^{d_i}}|$ on mahdollisimman pieni.

Toteutus, joka on esitetty kuvassa 2, koostuu viidestä osasta. Osat ovat

- Scatter, joka tekee yhden sirotusaskeleen,



Kuva 1: Sirotuspuu ja leikkaus



Kuva 2: Ohjelman arkkitehtuuri

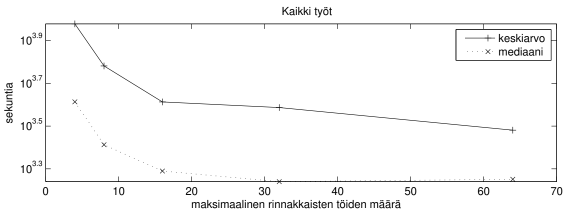
- Search, joka muodostaa sirotuspuun ja etsii siitä leikkauksen, josta alkuperäisen lauselogiikan ongelman toteutuvuus päätellään,
- SATQueue, joka välittää ongelmia laskettavaksi gridiin,
- Filter, joka ratkoo helpot osaongelmat paikallisesti, ja
- GRIDJM, joka hoitaa grid-kohtaisen kommunikaation.

Jotta gridiin ja laskennallisesti vaativaan sirotukseen liittyviä aikaviiveitä voidaan tasoittaa, Search'ssä ja SATQueuessa on jonoja, jotka sisältävät valmiita osaongelmia. Mikäli jollekin puun solmulle on gridistä saatu tulos, joka kertoo solmuun liittyvän ongelmatapauksen olevan toteutumaton, ei solmua siroteta enempää.

4 Tulokset

Jotta työssä esitettyjen ajatusten soveltuvuutta käytäntöön voidaan arvioida, sirotusalgoritmista on tehty toteutus, jonka nimi on SATU (SAT Ubiquitous). Testeissä sitä on ajettu 500 MHz Pentium III -tietokoneessa, jossa on 500 MB muistia ja Linux-käyttöjärjestelmä. Laskennalliseksi gridiksi valittiin NorduGrid [9], pohjoismainen tuotantokäytössä oleva gridympäristö. Tulosten yhteenveto on esitetty kuvassa 3, joka näyttää kaikkien ajettujen ajojen aikojen keskiarvon ja mediaanin suhteessa maksimimäärään käytössä olevia grid-solmuja.

SATU pystyy ratkaisemaan joitakin tiettävästi ennen ratkaisemattomia toteutuvuusongelmatapauksia SAT2005-konferenssin yhteydessä järjestetystä SAT-toteutuvuustarkistinkilpailusta. Osa



Kuva 3: Mediaani ja keskiarvo ajoajoista kaikille kaavoille

Taulukko 1: Valikoituja ratkaisemattomia ongelmia SAT2005 toteutuvuustarkistinkilpailusta

Teollisuusongelmat		
Nimi	Aika (s)	Tulos
vmpc_32	108	sat
vmpc_36	43200	aika loppui
Generoidut ongelmat		
Nimi	Aika (s)	Tulos
eulcbip-7-UNSAT	43200	aika loppui
eulcbip-8-UNSAT	43200	aika loppui
eulcbip-9-UNSAT	43200	aika loppui
gensys-ukn007	19192	unsat
gensys-ukn008	13217	unsat
linvrinv6	43200	aika loppui
linvrinv7	43200	aika loppui
linvrinv8	43200	aika loppui
linvrinv9	43200	aika loppui
mod2c-rand3bip-sat-230-1	3208	sat
mod2c-rand3bip-sat-230-3	1302	sat
mod2c-rand3bip-sat-240-2	17900	sat
mod2c-rand3bip-sat-240-3	43200	aika loppui
mod2c-rand3bip-sat-250-1	1692	sat

tapauksista on esitetty taulukossa 1, ja SATUlla ratkaistuihin on merkitty tulos (sat/unsat).

5 Yhteenveto

Työn pääkontribuutio on *sirotus*, tiettävästi uusi lauselogiikan ratkaisemisen hajautusmenetelmä. Menetelmä mahdollistaa erillisen hajautusheuristiikan ja minkä tahansa toteutuvuustarkistimen käyttämisen, mukaan lukien suljetun lähdekoodin tarkistimet. Sirotuksessa annettu SAT-ongelma jaetaan rajoittuneempiin osaongelmiin jotka muodostavat *sirotuspuun*. Puun solmut lähetetään ratkaistavaksi rinnakkain esimerkiksi laskennallisen gridiin. Osaongelmien rajoitteet esitetään klausuulein, jotka muodostetaan käyttäen erillistä heuristiikkaa.

Työssä näytetään että menetelmä on tehokas ja skaalautuu resurssien mukaan. Koeajot ajetaan NorduGrid-nimisessä laskennallisessa gridissä, ja laskenta-ajoissa saavutetaan lineaarinen nopeutuminen käytettävien resurssien suhteen, olettaen että työt ovat riittävän vaikeita suhteessa kommunikaatioviiveisiin.

Kiitokset Tutkimus on tehty Suomen Akatemian rahoittaman projektin numero 211025 puitteissa.

Viitteet

[1] N. Amla, X. Du, A. Kuehlmann, R. Kurshan ja K. McMillan. An analysis of

SAT-based model checking techniques in an industrial environment. *CHARME*:ssa, *LNCS*:n volyymi 3725, ss. 254 – 268. Springer, 2005.

[2] A. Biere, E. Clarke , R. Raimi ja Y. Zhu. Verifying safety properties of a PowerPC microprocessor using symbolic model checking without BDDs. *CAV 1999*:ssä, *LNCS*:n volyymi 1633, ss. 60 – 71. Springer, 1999.

[3] A. Biere, A. Cimatti, E. Clarke ja Y. Zhu. Symbolic model checking without BDDs. *TACAS 1999*:ssä, *LNCS*:n volyymi 1579, ss. 193 – 207. Springer, 1999.

[4] S. Brin ja L. Page. The anatomy of a large–scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[6] Satisfiability Competition. `http://www.satcompetition.org/`.

[7] S. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing*:ssa, ss. 151–158. ACM Press, 1971.

[8] M. Davis, G. Logemann ja D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962

[9] P. Eerola, B. Konya, O. Smirnova, T. Ekelöf, M. Ellert, J. R. Hansen, J. L. Nielsen, A. Wäänänen, A. Konstantinov ja F. Ould-Saada. Building a production grid in Scandinavia. *IEEE Internet Computing*, 7(4):27–35, 2003.

[10] E. Giunchiglia, Y. Lierler ja M. Maratea. SAT-based answer set programming. *AAAI 2004*, ss. 61–66. AAAI Press, 2004.

[11] A. E. J. Hyvärinen. SATU: A system for distributed propositional satisfiability checking in computational grids. Tutkimusraportti A100, Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio, Espoo, Suomi, helmikuu 2006.

[12] H. Kautz ja B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. *AAAI 1996*:ssa, ss. 1194 – 1201. AAAI Press, 1996.

[13] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(1):4–15, 1992.

[14] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3–4):241–273, 1999.

[15] C. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Inc. Boston, MA, 1994.

[16] C. Sinz and W. Küchlin. Verifying the on-line help system of SIEMENS magnetic resonance tomographs. *ICFEM 2004*:ssä, *LNCS*:n volyymi 3308, ss. 391 – 402. Springer, 2004.

[17] W. Sullivan III, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye ja D. Anderson. A new major SETI project based on Project Serendip data and 100,000 personal computers. *Proceedings of the Fifth International Conference on Bioastronomy*:ssä, IAU Colloquiumin numero 161, Bologna, Italia, 1997. Editrice Compositori.

[18] M. Velev ja R. Bryant. Effective use of Boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors. *Journal of Symbolic Computation*, 52(2):73 – 106, helmikuu 2003.