Dominointianalyysi

Teppo Niinimäki

Helsinki 10.5.2010

Approksimointialgoritmit

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä — Författare — Author
Teppo Niinimäki

Työn nimi — Arbetets titel — Title

Dominointianalyysi

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Artikkelitiivistelmä	10.5.2010	6 sivua

Tiivistelmä — Referat — Abstract

Kombinatoristen optimointiongelmien likimääräiseen ratkaisuun tarkoitettuja algoritmeissa analysoidaan tyypillisesti approksimointisuhdetta. Tässä tiivistelmässä tutustutaan vaihtoehtoiseen algoritmin hyvyyden analysointimenetelmään, *dominointianalyysiin*, sekä luodaan katsaus muutamiin tunnettuihin tuloksiin eri ongelmilla saavutettavista dominointisuhteista.

ACM Computing Classification System (CCS):
F.2.0 [Analysis of Algorithms and Problem Complexity]: General
G.1.6 [Numerical Analysis]: Optimization
G.2.1 [Discrete Mathematics]: Combinatorics

Avainsanat — Nyckelord — Keywords
approksimointialgoritmit, dominointianalyysi

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Sisältö

1 Johdanto

Lukuisia yleisiä kombinatorisia optimointiongelmia ei tunnetusti pystytä ratkaisemaan tarkasti polynomisessa ajassa, ellei P ole NP. Näihin tapauksiin on kehitetty nopeammin toimivia algoritmeja, jotka pyrkivät kaikkien käypien ratkaisujen joukosta löytämään mahdollisimman hyvän ratkaisun. Tällaisten algoritmien hyvyyttä mitataan usein johtamalla teoreetisia ylä- ja alarajoja algoritmien approksimointisuhteelle, eli algoritmin tuottaman ratkaisun ja optimiratkaisun arvojen suhteelle.

Dominointianalyysi (DA) on vaihtoehto approksimointisuhteen analysoimiselle algoritmin hyvyyttä tutkittaessa. Siinä tarkastellaan kuinka hyvin algoritmin löytämän ratkasun voidaan taata sijoittuvan kaikkien käypien ratkaisujen joukossa. Sanotaan, että löydetty ratkaisu *dominoi* niitä käypiä ratkaisuja, jotka ovat korkeintaan yhtä hyviä kuin se itse. *Dominointisuhde* eli dominoitujen ratkaisujen osuus kaikista ratkaisuista pahimmassa tapauksessa haluttaisiin luonnollisesti saada mahdollisimman suureksi.

Monissa tapauksissa algoritmin tai ongelman dominointianalyysi on hyödyllistä ja valaisevaa. Dominointianalyysi ja approksimointianalyysi katsovat tarkasteltavaa algoritmia eri näkökulmista; kumpaakin hyödyntämällä saa monipuolisemman kuvan algoritmin suorituskyvystä. Joissakin tapauksissa approksimointianalyysia ei myöskään välttämättä voi käyttää luonnollisella tavalla, mutta ratkaisujen dominointisuhteet voidaan silti helposti määritellä. Tällaisia ovat esimerkiksi ongelmat, joissa optimoidaan useaa kohdefunktiota samanaikaisesti.

Dominointianalyysillä on myös eräitä toivottavia teoreettisia ominaisuuksia. Eräs tällainen ominaisuus liittyy kauppamatkustajan ongelmaan eli lyhimmän kaikissa solmuissa kerran vierailevan kierroksen löytämiseen painotetussa verkossa: Kombinatorinen optimointiongelma kuuluu luokkaan APX, jos sitä voidaan polynomisessa ajassa approksimoida vakiosuhteella. Tunnetusti kauppamatkustajan ongelma (TSP) ei kuulu luokkaan APX, vaikka vastaava reitin pituuttu maksivoiva max-TSP kuuluu siihen. Tätä epäsymmetriaa on pidetty approksimointisuhteen kannalta huonona puolena, sillä hyvä max-TSP-algoritmi voidaan helposti muuntaa hyväksi TSP-algoritmiksi ja näin ollen ongelmien voisi intuitiivisesti olettaa kuuluvan samaan luokkaan. Dominointianalyysin kannalta TSP ja max-TSP sen sijaan ovat ekvivalentteja ongelmia.

Tämän artikkelitiivistelmän loppuosa rakentuu seuraavasti: Luvussa 2 määrittelemme tarkemmin tärkeimmät dominointianalyysissä käytettävät käsitteet. Tämän jäl-

keen luvussa 3 tutustumme erilaisiin dominointianalyysillä saavutettuihin tuloksiin. Tiivistelmä perustuu Gutinin ja Yeon tekniseen raportiin [GY04] aiheesta.

2 Määritelmä

Olkoon P optimointiongelma ja $\mathcal{A}$ algoritmi sille. *Dominointiluku* $\mathrm{domn}(\mathcal{A}, n)$ on suurin sellainen kokonaisluku $m(n)$, että kaikille P:n kokoa n oleville tapauksille I algoritmin $\mathcal{A}$:n palauttama ratkaisu on vähintään yhtä hyvä kuin ainakin $m(n)$ käypää ratkaisua tapaukselle I. Siis riippumatta ongelman tapauksesta aina voidaan taata, että algoritmin ratkaisu dominoi ainakin $\mathrm{domn}(\mathcal{A}, n)$ käypää ratkaisua. Vastaavasti algoritmin *dominointisuhde* $\mathrm{domr}(\mathcal{A}, n)$ on dominointiluvun osuus kaikkien käypien ratkaisujen lukumäärästä. Selvästi dominointiluku on aina vähintään 1, sillä mikä tahansa ratkaisu dominoi ainakin itseään. Dominointisuhde on siis välillä $(0, 1]$, ja optimiratkaisun aina löytävän tarkan algoritmin dominointisuhde on 1.

On ehdotettu, että optimointiongelman likimääräisen ratkaisun laadun mitan tulisi noudattaa seuraavia luonnollisia ominaisuuksia: mitta pienenee, kun ratkaisu paranee; se saa arvon 0, jos ratkaisu on optimaalinen ja se on sama ekvivalenttien ongelmatapausten toisiaan vastaaville ratkaisuille. Approksimointianalyysin yhteydessä käytetty approksimointisuhde ei esimerksi välttämättä noudata viimeistä ominaisuutta. Sen sijaan kaikki vaatimukset pätevät suureelle $1 - r$, missä r on dominointisuhde.

Dominoituvuuteen liittyen on luonnollista määritellä seuraavat luokitukset: Algoritmi $\mathcal{A}$ on $\mathcal{DOM}$-hyvä, mikäli se toimii polynomisessa ajassa ja on olemassa sellainen polynomi $p(n)$, että $\mathcal{A}$:n dominointisuhde on vähintään $1/p(n)$ millä tahansa ongelmakoolla n. Ongelma P on $\mathcal{DOM}$-helppo, jos sille on olemassa $\mathcal{DOM}$-hyvä algoritmi, ja vastaavasti P on $\mathcal{DOM}$-vaikea jos sellaista algoritmia ei ole.

3 Tuloksia

Seuraavissa aliluvuissa esitellään muutamia dominointianalyysillä saavutettuja tuloksia. Skedulointiesimerkkiä lukuunottamatta esitetään pääasiassa vain saavutettu tulos eikä todistuksia käydä läpi.

3.1 Skedulointi monella suorittimella

Monen suorittimen skedulointiongelmassa (MMSP) on annettu töiden joukko $J = \{1, 2, \ldots, n\}$, suorittimien lukumäärä m sekä jokaiselle työlle $j \in J$ suoritusaika $\sigma(j)$. Päämääränä on jakaa työt suorittimille siten, että kokonaissuoritusaika on mahdollisimman pieni, eli löytää töiden joukolle J sellainen m-osiointi $(B_1, B_2, \ldots, B_m)$, joka minimoi kokonaissuoritusajan $\max_{1 \leq i \leq m} \sigma(B_i)$, missä $\sigma(B) = \sum_{j \in B} \sigma(j)$. Oletetaan, että $m < n$, sillä muussa tapauksessa ongelma voidaan ratkaista triviaalisti antamalla jokaiselle työlle oma suoritin.

Tarkastellaan erikoistapausta, jossa suorittimien lukumäärä on $m = 2$. Huomataan, että tässä tapauksessa töiden kokonaiskeston minimointi tarkoittaa samaa kuin suorittimien kuormien eron $|\sigma(A_1) - \sigma(A_2)|$ minimointi. Muodostetaan seuraavanlainen ahne algoritmi: Järjestetään työt keston mukaan laskevaan suuruusjärjestykseen $\sigma(\pi(1)) \geq \sigma(\pi(2)) \geq \cdots \geq \sigma(\pi(n))$. Tämän jälkeen työt annetaan tässä järjestyksessä aina sille koneelle, jolle siihen mennessä annettu kuorma on kyseisellä hetkellä pienempi. Jos kuormat ovat yhtä suuret, annetaan työ jommalle kummalle. Selvästi raskaimmalle työlle $\pi(1)$ pätee $|\sigma(A_1) - \sigma(A_2)| \leq \sigma(\pi(1))$. Olkoot (C_1, C_2) mielivaltainen työnjako vastaavalle ongelmalle $J - \{\pi(1)\}$, josta on poistettu raskain työ. Nyt ratkaisuun (C_1, C_2) lisätään työ $\pi(1)$ suorittimelle, jonka kuorma on raskaampi, saadaan alkuperäiseen ongelmaan ratkaisu (D_1, D_2), jolle $|\sigma(D_1) - \sigma(D_2)| \geq \sigma(\pi(1))$. Koska tällaisia ratkaisuja (D_1, D_2) on ainakin puolet kaikkien ratkaisujen määrästä ja algoritmin tuottama ratkaisu on niitä kaikkia parempi, on saavutettava dominointisuhde siis vähintään $1/2$.

Algoritmi voidaan yleistää useammalle suorittimelle seuraavalla tavalla: Olkoon $s = \Theta(n + \sum_{i=1}^{n} \log \sigma(i))$ annetun ongelman koko. Jos $s \geq m^n$, voidaan ongelma ratkaista tarkasti polynomisessa ajassa s:n suhteen. Muussa tapauksessa valitaan ensin $r = \lceil \log n / \log m \rceil$ pitkäkestoisinta työtä, ja jaetaan ne suorittimille optimaalisesti käyttämällä tarkkaa algoritmia. Tämän jälkeen annetaan loput työt järjestyksessä pisimmästä lyhimpään aina pienimmän kuorman omaavalle suorittimelle samaan tapaan kuin kahden suorittimien tapauksessa. Saadulle algoritmille $\mathcal{A}$ voidaan osoittaa seuraava yleinen dominointitulos:

Teoreema 3.1. *Algoritmi $\mathcal{A}$ toimii ajassa $O(s^2 \log s)$, missä s on ongelmatapauksen koko. Lisäksi pätee*

$$\lim_{s \to \infty} \operatorname{domr}(\mathcal{A}, s) = 1.$$

Teoreeman mukaan ongelmakokoa kasvattamalla saadaan algoritmin dominointisuh-

de mielivaltaisen lähelle ykköstä. Tämä yleinen tulos on selkeästi vahvempi kuin yllä kahden suorittimen erikoistapaukselle todistettu.

3.2 Ahne algoritmi

Riippumaton järjestelmä on äärellisestä joukosta E ja osajoukkoperheestä $\mathcal{F} \in \mathcal{P}(E)$ muodostuva pari, joka täyttää seuraavat kaksi ehtoa:

1. Tyhjä joukko kuuluu perheeseen $\mathcal{F}$.
2. Jos $X \in \mathcal{F}$ ja $Y \subset X$, niin $Y \in \mathcal{F}$.

Perheen $\mathcal{F}$ joukkoja kutsutaan *riippumattomiksi joukoiksi*, ja maksimaalisesta riippumattomasta joukosta käytetään nimitystä *kanta*. Selvästi riippumaton järjestelmä voidaan esittää yksikäsitteisesti sen kantojen avulla. Riippumaton järjestelmä on *matroidi* jos sille pätee myös kolmas ehto

3. Jos U ja V kuuluvat perheeseen $\mathcal{F}$ ja $|U| > |V|$, niin on olemassa $x \in U - V$, jolle $V \cup \{x\} \in \mathcal{F}$.

Useat kombinatoriset optimointiongelmat voidaan esittää esittää riippumattomana järjestelmänä $(E, \mathcal{F})$ ja painofunktiona $w : E \to \mathbb{R}$. Joukon $S \in \mathcal{F}$ painoksi $w(S)$ määritellään sen alkioiden painojen summa. Optimointiongelman tavoitteena on löytää painoltaan kevyin kanta $B \in \mathcal{F}$. Sanotaan näin määriteltyä ongelmaa $(E, \mathcal{F})$-optimointiongelmaksi.

Joukolle $S \in \mathcal{F}$ merkitään $I(S) = \{e \notin S \mid S \cup \{x\} \in \mathcal{F}\}$. Joukko $I(S)$ sisältää siis ne alkiot, jotka voidaan lisätä S:n siten, että saatu joukko on yhä riippumaton (eli kuuluu $\mathcal{F}$:ään). *Ahne algoritmi* yllä kuvatulle optimointiongelmalle aloittaa tyhjästä joukosta $X = \emptyset$, ja jokaisella askeleella lisää sen hetkiseen joukkoon X keveimmän alkion $e \in I(X)$. Algoritmi jatkaa askeleita kunnes päätyy johonkin kantaan.

Ahneelle algoritmille on osoitettu seuraava yläraja:

Teoreema 3.2. *Olkoon $(E, \mathcal{F})$ riippumaton järjestelmä ja $B' = \{x_1, x_2, \ldots x_k\}$ kanta, missä $k \geq 2$. Oletetaan, että seuraava pätee jokaiselle kannalle $B \in \mathcal{F}$, $B \neq B'$:*

$$\sum_{j=0}^{k-1} |I(x_1, x_2, \ldots, x_j) \cap B| < k(k+1)/2.$$

Tällöin ahneen algoritmin dominointiluku $(E, \mathcal{F})$-optimointiongelmalle on 1.

Siis jos $(E, \mathcal{F})$-optimointiongelmalle täyttää teoreemassa vaaditun ehdon, niin mille tahansa algoritmille löytyy aina tapaus, jossa algoritmi tuottaa huonoimman mahdollisen käyvän ratkaisun. Tulos on hieman yllättävä, sillä tunnetun tuloksen mukaan matroidien tapauksessa ahne algoritmi löytää aina optimiratkaisu ja saavuttaa siten dominointiluvun $|\mathcal{F}|$. Erityisesti saadusta teoreemasta seuraa, että sijoitteluongelmalle ja sekä symmetriselle että epäsymmetriselle kauppamatkustajan ongelmalle minkä tahansa ahneen algoritmin dominointiluku on 1.

3.3 Kauppamatkustajan ongelma

Asymmetrisessä kauppamatkustajan ongelmassa (ATSP) tavoitteena on löytää minimipainoinen Hamiltonin kehä (kaikissa solmuissa täsmälleen kerran vieraileva kierros) painotetussa täydellisessä suunnatussa verkossa K_n^*. Edellä todettiin, että ahneen algoritmin dominointiluku ATSP:lle on 1. Tässä tarkastallaan ongelmaa yleisemmin rajoittumatta mihinkään tiettyyn yksittäiseen algoritmiin. Mikäli oletetaan algoritmi käyttää ainakin yhden aikayksikön jokaista harkitsemaansa kaarta kohden, saadaan seuraava dominointilukuun liittyvä tulos:

Korollaari 1. *Olkoon $\mathcal{A}$ aikavaativuuden $t(n)$ algoritmi ATSP-ongelmaan. Jos pätee $t(n) \geq en$, niin $\mathcal{A}$:n dominointiluku on korkeintaan $(t(n)/n)^n$.*

Tiedetään, että on olemassa ajassa $O(n)$ toimivia ATSP-algoritmeja, joiden dominointiluku on $2^{\Theta(n)}$. Suurilla n tämä on selvästi pienempi kuin kaikkien ratkaisujen määrä $(n-1)!$. Korollaarista seuraa, että kyseistä tulosta ei voida parantaa.

Edellinen tulos on triviaali, jos tarkasteltavan algoritmin aikavaativuus on neliöllinen tai suurempi. Yleisesti polynomisessa ajassa toimiville ATSP-algoritmeille voidaan johtaa seuraava dominoituvuusraja:

Teoreema 3.3. *Jos oletetaan $P \neq NP$, ei ole olemassa polynomisessa ajassa toimivaa ATSP-algoritmia, jonka dominointiluku olisi vähintään $(n-1)! - \lfloor n - n^\alpha \rfloor!$ jollain vakiolla $\alpha < 1$.*

3.4 Muita tuloksia ja ongelmia

Tässä esiteltyjen lisäksi eri algoritmeille tunnetaan paljon myös muita dominointituloksia. ATSP-ongelmaan esimerkiksi on olemassa useita dominointisuhteen $\Omega(1/n)$ saavuttavia polynomisia algoritmeja. Sen sijaan yhtään vakiodominointisuhteen $\Omega(1)$

saavuttavaa polynomista algoritmia ei kyseiselle ongelmalle tunneta. Toisaalta painotettuun maksimileikkausongelmaan ja painotettuun k-SAT-ongelmaan on kehitetty vähintään vakiodominointisuhteeseen pääsevät algoritmit. Maksimileikkauksen tapauksessa vakiosuhde saavutetaan jopa lineaariaikaisella algoritmilla. Myös yleisen painottamattoman SAT-ongelman maksimointiversion tiedetään olevan $\mathcal{DOM}$-helppo. Sen sijaan painotetusta versiosta ei tiedetä, kuuluuko se $\mathcal{DOM}$-helppoihin ongelmiin vai ei.

Maksimiklikkiongelman ja pienimmän solmupeitteen ongelman on todistettu olevan $\mathcal{DOM}$-kovia ellei $P = NP$. Niille ei siis luultavasti ole mahdollista kehittää polynomisessa ajassa toimivaa käänteisesti polynomiseen dominointisuhteeseen pääsevää algoritmia.

4 Yhteenveto

Dominointianalyysi on tavanomaisesta approksimointianalyysista poikkeava tapa tutkia optimointiongelmaan tarkoitettujen likimääräisten algoritmien hyvyyttä. Joissakin tapauksissa se mahdollistaa mielenkiintoisten ja ehkä yllättävien havaintojen tekemisen algoritmin toiminnasta. Algoritmin dominointiluvun tai dominointisuhteen selvittäminen tavanomaisen approksimointisuhteen lisäksi antaa laajemman kokonaiskuvan algoritmin suorituskyvystä.

Lähteet

GY04 Gutin, G. ja Yeo, A., Introduction to domination analysis. Tekninen raportti, Royal Holloway, University of London, 2004.