# Hakurobotit

Leena Salmela
leena.salmela(at)hut.fi

Seminaariesitelmä 26.4.2006
T-106.850 Tiedonhaku
Tietotekniikan osasto
Teknillinen korkeakoulu

**Tiivistelmä**

Hakurobotit hakevat verkkosivuja ja tallettavat ne esimerkiksi hakukoneen indeksoijaa varten. Hakurobotit myös huolehtivat, että haetut sivut ovat mahdollisimman tuoreita. Hakurobotin toteutukseen liittyy monia varsin vaikeitakin ongelmia: Kuinka saadaan suuri osa verkossa olevasta materiaalista haettua ja kuinka se pidetään ajan tasalla? Lisäksi hakurobotit toteutetaan usein hajautettuina ohjelmistoina, jolloin erilliset hakurobotteja täytyy jotenkin koordinoida. Tässä esityksessä käsitellään joitain mahdollisia ratkaisuja yllä mainittuihin ongelmiin. Esitys perustuu lähinnä Junghoo Chon väitöskirjaan [1].

## 1 Johdanto

Hakurobotti on ohjelma, joka hakee verkkosivuja ja tallettaa ne paikalliselle levylle, josta muut ohjelmat, esim. hakukoneen indeksoija, käsittelevät niitä. Haetuista verkkosivuista erotetaan linkit muihin sivuihin ja uudet linkit lisätään hakurobotin työjonoon. Hakurobotti hakee miljardeja sivuja ja se huolehtii myös kopion synkronoinnista, kun alkuperäinen sivu muuttuu.

Hakurobotit käyttävät jatkuvasti muiden organisaatioiden resursseja. Siksi on tärkeää, että hakurobotti pyrkii minimoimaan vaikutukset näihin resursseihin. Tätä varten on kehitetty Robots Exclusion protokolla [5], jonka avulla WWW-palvelimen ylläpitäjä voi rajoittaa robottien hakemia sivuja. Yksittäisten sivujen osalta robottien toimintaa voi ohjata robots meta tagilla. Sillä voi kieltää tai sallia sivun indeksoimisen ja kieltää tai sallia linkkien seuraamisen kyseiseltä sivulta. Nykyisin tosin vain harvat robotit tukevat robots meta tag määrittelyjä.

Hakurobotit voidaan jakaa kahteen luokkaan. Osa hakuroboteista pyrkii kattamaan mahdollisimman suuren osan verkkosivuista, kun taas toiset pyrkivät hakemaan vain johonkin aiheeseen liittyvät sivut. Tässä esityksessä paneudutaan lähinnä ensimmäisen tyyppisiin hakurobotteihin.

Hakurobotin toteutuksessa on monenlaisia haasteita. Robotin täytyy päättää, missä järjestyksessä se hakee sivuja. Kun sivut on haettu paikalliselle levylle, hakurobotin pitää huolehtia sivujen tuoreudesta. Tähän liittyen sen tulee arvioida, kuinka usein haetut sivut muuttuvat, ja tästä edelleen päätellä, kuinka usein sen pitää synkronoida paikallinen kopio. Haettavia verkkosivua on paljon, joten useimmiten kannattaa käynnistää useita rinnakkaisia hakurobotteja, joiden koordinoiminen on haastava ongelma.

## 2 Hakujärjestys

WWW-sivujen voidaan ajatella muodostavan suunnatun verkon, jossa hakurobotti tekee hakua. Yksinkertaisimmillaan haku voidaan tehdä leveysjärjestyksessä, jolloin hakurobotti on ennen pitkää käynyt kaikilla sivuilla, joille haun alkupisteestä pääsee.

Monissa tapauksissa hakurobotin ei kuitenkaan ole mahdollista hakea kaikkia mahdollisia verkkosivuja. WWW-sivuja on paljon, joten hakurobotti ei välttämättä halua tai voi varastoida niitä kaikkia. Toisaalta sivujen hakeminen vie aikaa ja hakurobotin pitää ehkä synkronoida jo hakemiaan sivuja ennen kuin se on saanut kaikki mahdolliset sivut haettua.

Näistä syistä monet hakurobotit vierailevat verkkosivuilla jonkinlaisessa tärkeysjärjestyksessä. Vaikka hakurobotti aikoisikin hakea kaikki löytämänsä verkkosivut, tärkeysjärjestyksessä vierailussa on sekin hyvä puoli, että uudet tärkeät sivut löytyvät todennäköisesti nopeammin kuin jos hakurobotti käyttäisi leveysjärjestystä.

Verkkosivujen tärkeyttä voidaan mitata hyvin monella tapaa. Alla on lueteltu joitakin mahdollisia mittoja tärkeydelle:

1. Samankaltaisuus annetun kyselyn kanssa. Soveltuu lähinnä tiettyyn aiheeseen liittyvien sivujen hakuun.
2. Viittausten määrä (backlink count).
3. PageRank.
4. Linkkien määrää (forward link count) voidaan käyttää painottamaan hakemistosivuja.
5. Jokin URL:iin pohjautuva mitta. Esimerkiksi URL:eja, joissa on vähän '/'-merkkejä voidaan pitää tärkeämpinä.

Näitä mittoja voidaan myös yhdistellä.

Lukuunottamatta URL:iin pohjautuvia mittoja hakurobotti ei voi käyttää yllä esitettyjä tärkeysmittoja suoraan, koska sivua, jonka tärkeyttä halutaan mitata, ei ole vielä haettu ja toisaalta viittausten määrän tai PageRank:in laskemiseksi tarvittaisiin kaikki muutkin verkkosivut, jotka eivät myöskään ole välttämättä vielä saatavilla.

Viittausten määrää ja PageRank:ia on melko suoraviivaista arvioida. Niille voidaan laskea arviot niiden sivujen perusteella, jotka on jo haettu. Linkkien määrään perustuvaa tärkeysmittaa ei ole mahdollista arvioida ilman sivua itseään. Sivun samankaltaisuutta kyselyn kanssa sen sijaan voidaan arvioida laskemalla ankkuritekstin samankaltaisuus kyselyn kanssa.

Chon [1] mukaan PageRank:in antamaa järjestystä noudattavat hakurobotti löytää korkean PageRank:in ja korkean viittausmäärän sivut nopeasti. Jos järjestämiseen käytetään jotain muuta mittaa, tässä mitassa tärkeät sivut eivät löydy yhtä nopeasti. Jos taas haetaan sivuja, jotka ovat samankaltaisia annetun haun kanssa, PageRank:in yhdistäminen ankkuritekstin samankaltaisuuteen antaa parhaat tulokset.

## 3 Paikallisen kopion synkronointi

Verkkosivut muuttuvat jatkuvasti. Niiden linkkirakenne muuttuu kuten myös sivujen sisältö. Hakurobotin täytyy siis synkronoida hakemiaan sivuja, jotta ne pysyisivät tuoreina.

Eri verkkosivut muuttuvat eri nopeuksilla. Chon [1] arvion mukaan noin neljännes sivuista muuttuu päivittäin ja 30 % muuttuu harvemmin kuin neljästi vuodessa. Sivujen muuttumisnopeus vaihtelee

myös domainittain. Esim. com-domainin sivut vaihtuvat usein, kun taas edu- ja gov-domainin sivut harvemmin. Myös sivujen elinaika vaihtelee. Jotkin ovat olemassa alle viikon kun taas toiset ovat olemassa vuosia.

Jotta voisimme verrata, kuinka tuoreina erilaiset hakurobotit pitävät sivukokoelmaansa, tarvitsemme mallin verkkosivujen muuttumiselle. Toisaalta tämä myös auttaa kehittämään parempia metodeja tuoreuden säilyttämiseksi. Hyvä malli verkkosivujen muuttumiselle on Poisson-prosessi. Tällöin oletetaan, että muutokset tapahtuvat satunnaiseen aikaan siten, että keskimääräinen muutosfrekvenssi on vakio.

Sivujen synkronoinnin suhteen tässä tarkastellaan kahta seikkaa. Ensinnäkin pyritään estimoimaan eri sivujen muutosfrekvenssiä, jotta voitaisiin käytttää synkronointipolitiikkaa, joka käyttää näitä hyväkseen. Sitten tarkastellaan erilaisten synkronointipolitiikkojen vaikutusta sivukokoelman tuoreuteen.

## 3.1 Muutosfrekvenssien estimointi

Jos tiedämme, että sivu on muuttunut $X$ kertaa ajassa $T$, voimme estimoida muutosfrekvenssiä lausekkeella $X/T$. Täydellisen tiedon mallissa tämä estimaattori on harhaton ja konsistentti.

Hakurobotin tapauksessa emme kuitenkaan tiedä tarkalleen, kuinka monta kertaa sivu on muuttunut, koska se on saattanut muuttua useamman kerran vierailujemme välillä. Tällöin estimaattori $X/T$ on harhainen antaen keskimäärin pienempiä arvoja kuin todellinen muutosfrekvenssi. Estimaattori ei myöskään ole konsistentti. Jos tarkkailemme sivua pidempään frekvenssillä kerran viikossa, estimaattorin arvo ei lähene sivun todellista muutosfrekvenssiä.

Oletetaan, että sivun muutokset noudattavat Poisson-prosessia parametrilla $\lambda$ (muutosfrekvenssi) ja että tarkkailufrekvenssimme on $f$. Tällöin todennäköisyys, että sivu ei muutu aikayksikössä $I = 1/f$ on

$$q = Pr\{X(t+I) - X(t) = 0\} = \frac{\lambda^0 e^{-\lambda I}}{0!} = e^{-\lambda I} = e^{-\lambda/f}$$

Tästä saamme muutosfrekvenssille lausekkeen $\lambda = -f \log q$. $q$:ta taas voimme estimoida helposti lausekkeella $q = \overline{X}/n$, missä $\overline{X}$ on niiden tarkkailukertojen määrä, kun sivu ei muuttunut ja $n$ on tarkkailukertojen kokonaismäärä.

Estimaattori $-f \log(\overline{X}/n)$ on muuten hyvä, mutta jos $\overline{X} = 0$ eli sivu muuttui joka kerralla, estimaattori tuottaa arvon $\infty$. Näin ollen tämä estimaattorin on myös harhainen. Tästä ongelmasta päästään eroon lisäämällä pieni vakio sekä estimaattorin osoittajaan että nimittäjään. Uusi estimaattori on siis $-f \log(\frac{\overline{X}+0.5}{n+0.5})$. Tämäkään estimaattori ei ole harhaton, mutta se on vähemmän harhainen kuin $X/T$. Lisäksi uusi estimaattori on konsistentti.

Vastaavia estimaattoreita voidaan myös kehitellä tilanteille, joissa sivua tarkkaillaan epäsäännöllisesti.

## 3.2 Synkronointipolitiikka

Jotta erilaisia synkronointipolitiikkoja voitaisiin vertailla, täytyy ensin määritellä sivukokoelmalle tuoreus ja ikä, joihin vertailut voidaan perustaa. Yhden sivun $e_i$ tuoreus $F(e_i, t)$ ajanhetkenä $t$ hakurobotin kokoelmassa määritellään seuraavasti:

$$F(e_i, t) = \begin{cases} 1 & \text{jos } e_i \text{ on tuore ajanhetkenä } t \\ 0 & \text{muuten} \end{cases}$$

Sivukokoelman $S$ tuoreus taas on yksittäisten sivujen tuoreuden keskiarvo:

$$F(S,t) = \frac{1}{N}\sum_{i=1}^{N} F(e_i,t)$$

Yhden sivun ikä taas määritellään seuraavasti:

$$A(e_i,t) = \begin{cases} 0 & \text{sivu } e_i \text{ on tuore ajanhetkellä } t \\ t - \text{sivun } e_i \text{ muutosaika} & \text{muuten} \end{cases}$$

ja sivukokoelman ikä näiden keskiarvona:

$$A(S,t) = \frac{1}{N}\sum_{i=1}^{N} A(e_i,t)$$

Näiden lisäksi on hyödyllistä määritellä tuoreuden ja iän aikakeskiarvot koko sivukokoelmalle:

$$\overline{F}(S) = \lim_{t\to\infty} \frac{1}{t}\int_0^t F(S,t)dt \qquad \overline{A}(S) = \lim_{t\to\infty} \frac{1}{t}\int_0^t A(S,t)dt$$

Synkronointipolitiikka koostuu kolmesta osasta, synkronointitiheydestä, synkronointijärjestyksestä ja resurssien allokoinnista. On selvää, että mitä useammin synkronoimme sivukokoelmaa sitä tuoreempi se on. Mielenkiintoisempia komponentteja ovat synkronointijärjestys ja resurssien allokointi.

Cho [1] on tutkinut kolmea erilaista synkronointijärjestystä. Sivut voidaan synkronoida aina samassa järjestyksessä tai sitten ne voidaan synkronoida jollain tavalla satunnaisessa järjestyksessä. Satunnaisuus voidaan toteuttaa joko siten, että aina synkronoidaan kaikki sivut, mutta synkronointijärjestys on satunnainen, tai sivut voidaan synkronoida täysin satunnaisesti, jolloin synkronoitava sivu arvotaan satunnaisesti kaikkien sivujen joukosta aina, kun synkronointia tehdään. Osoittautuu, että sivukokoelma pysyy parhaiten tuoreena ja mahdollisimman nuorena, kun sivut synkronoidaan aina samassa järjestyksessä.

Jos sivut muuttuvat eri nopeudella, voisi ajatella olevan edullista synkronoida joitain sivuja nopeammin kuin toisia. Tästä on kyse resurssien allokoinnissa. Cho [1] on tutkinut tasaisen resurssien jaon ja sivujen muutosnopeuteen suhteutettujen resurssien jaon vaikutusta sivukokoelman tuoreuteen.

Osoittautuu, että tasainen resurssien jako on aina parempi kuin suhteutettu jako. Vaikka tulos vaikuttaakin aluksi epäintuitiiviselta, se voidaan selittää seuraavan esimerkin kautta. Oletetaan, että sivukokoelmassa on kaksi sivua, joista toinen muuttuu kerran päivässä ja toinen yhdeksän kertaa päivässä. Oletamme tässä lisäksi, että sivu 1 muuttuu täsmälleen kerran päivässä ja sivu 2 muuttuu täsmälleen kerran 1/9 päivässä. Mietitään nyt tilannetta, jossa saamme synkronoida vain yhden sivun päivässä ja synkronointi tapahtuu täsmälleen keskellä päivää, mikä sattuu olemaan myös keskellä sivun 2 viidettä päivitysperiodia. Kumpi sivu kannattaa synkronoida? Jos synkronoimme sivun 1, todennäköisyydellä 0.5 se on jo muuttunut ja sivu pysyy tuoreena koko loppu päivän. Saavutamme siis keskimäärin 0.25 lisäyksen kokoelman tuoreuteen. Toisaalta, jos synkronoimme sivun 2, todennäköisyydellä 0.5 se on jo päivittynyt, jolloin se on tuore korkeintaan kyseisen päivitysvälin lopun ja seuraavan päivitysvälin. Joka tapauksessa kokoelman tuoreuteen tuleva lisäys on huomattavasti pienempi kuin 0.25.

Tasainen resurssien jako ei kuitenkaan ole optimaalinen politiikka. Optimaalinen politiikka saadaan ratkaisemalla seuraava ongelma. Annettuna on keskimääräiset päivitystiheydet kullekin sivulle. Tehtävänä on maksimoida sivukokoelman aikakeskiarvotettu tuoreus, kun sivuja voidaan synkronoida yhteensä $f$ kertaa aikayksikössä. Tämä ongelma voidaan ratkaista matemaattisesti ja näin lasketut synkronointitiheydet kullekin sivulle tuottavat tuoreimman sivukokoelman myös oikealla aineistolla.

# 4 Hajautettu hakurobotti

Hajauttamalla hakuprosessi voidaan saavuttaa merkittäviä etuja. Tällainen hakurobotti skaalautuu paremmin. Lisäksi, jos eri alirobotit sijaitsevat maantieteellisesti eri paikoissa, verkon kuorma jakautuu tasaisemmin. Parhaassa tapauksessa verkon kuorma voi jopa vähentyä, jos esim. Euroopassa sijaitseva alirobotti lataa eurooppalaiset sivut ja Amerikassa sijaitseva amerikkalaiset sivut. Voi toki olla, että sivuja täytyy myöhemmin siirtää muidenkin alirobottien saataville, mutta tällöin ne voidaan esimerkiksi pakata tai voidaan lähettää vain erot edellisestä lähetyksestä.

Hajautetun hakurobotin toteuttamisessa on myös omat ongelmansa. Ensinnäkin on toivottavaa, että useammat alirobotit eivät lataisi samoja sivuja. Toiseksi, jos alirobotit eivät vaihda informaatiota keskenään, niillä on kovin erilainen käsitys verkon rakenteesta ja näin ollen myös sivujen tärkeydestä. Tämä saattaa johtaa siihen, että tärkeimpiä sivuja ei haetakaan ensimmäisenä. Jotta edellä mainittuja ongelmia voitaisiin välttää, alirobottien täytyy vaihtaa keskenään informaatiota. Hajautettua hakurobottia toteuttaessa pitää varmistaa, että robottien väliseen kommunikaatioon ei kulu liikaa resursseja.

Alirobottien väliseen koordinaatioon Cho [1] esittää kolme mallia. Ensimmäisessä mallissa alirobotit toimivat itsenäisesti. Kullekin alirobotille annetaan joitakin URL:ja, joista ne lähtevät liikkeelle, ja tämän jälkeen alirobotit toimivat itsenäisesti. Tässä mallissa ongelmaksi muodostuu alirobottien lataamat päällekkäiset sivut, mutta toisaalta tällaiset alirobotit on helppo toteuttaa ja ne eivät kuluta resursseja kommunikointiin.

Toisessa mallissa haettavat sivut jaetaan aliroboteille dynaamisesti (dynamic assignment). Tällöin tarvitaan erillinen koordinaattoriprosessi, joka hoitaa sivujen jakamisen. Sivut voidaan jakaa aliroboteille esim. domainin perusteella. Tällöin koordinaattori päättää haun aikana, mitä haetaan seuraavaksi ja lähettää sopivan lähtö-URL:in alirobotille. Jos alirobotti törmää osoitteeseen, joka ei kuulu sen alueelle, se lähettää URL:in edelleen koordinaattorille, joka sitten päättää, minkä alirobotin alueelle se kuuluu. Tässä mallissa alirobottien ja koordinaattorin välillä voi olla paljonkin liikennettä ja alirobottien ja koordinaattorin toteutus on monimutkaisempi. Toisaalta eri alirobotit eivät hae samoja sivuja.

Kolmannessa mallissa sivut jaetaan jo ennen hakua eri roboteille (static assignment). Tässä tapauksessa kaikki alirobotit tietävät, mikä niistä on vastuussa mistäkin sivusta. Verkko voidaan jakaa aliroboteille eri tavoin. Hierarkisessa jaossa kukin alirobotti on vastuussa yhdestä tai useammasta domainista. Jako voi myös perustua URL:eista laskettuihin hajautusarvoihin tai URL:ien konenimistä laskettuihin hajautusarvoihin.

Tässä mallissa alirobotit voivat toimia kolmella eri tavalla. Palomuurimoodissa kukin alirobotti hakee vain sivuja, jotka kuuluvat sen omalle alueelle. Kaikki sellaiset linkit, jotka johtavat yhden alirobotin alueelta toisen alueelle jätetään huomiotta. Tällöin on toki mahdollista, että jokin osa saavutettavasta verkosta jää tutkimatta, mutta kokeelliset tulokset osoittavat, että tämä moodi toimii hyvin, jos alirobotteja on pieni määrä (alle viisi).

Cross-over-moodissa alirobotit preferoivat omalla alueellaan olevia sivuja, mutta jos kaikki tällaiset sivut on haettu, ne hakevat myös muiden robottien alueella olevia sivuja ja saattavat näin löytää lisää oman alueensa sivuja. Tässä moodissa useampi alirobotti saattaa ladata samoja sivuja.

Vaihtomoodissa alirobotit eivät mene toistensa alueelle, mutta tietyin väliajoin robotit lähettävät toisilleen muiden alueelta löytämänsä osoitteet. Tällöin URL:it kannattaa lähettää suuremissa erissä ja useimmin viitatut URL:it (noin 10,000 – 100,000 kappaletta) kannattaa replikoida kaikille roboteille ja jättää lähettämättä. Vaihtomoodia kannattaa käyttää, jos alirobotteja on vähintään viisi tai jos halutaan ladata joka ikinen saavutettavissa oleva sivu.

# 5 Yhteenveto

Yllä on esitetty tärkeimpiä hakurobottien toteutukseen liittyviä kysymyksiä. Näiden lisäksi toteutukseen liityy myös monia muita yksityiskohtia. Esimerkiksi DNS kyselyt voivat muodostua hakurobotin pullonkaulaksi, joten niitä kannattaa pitää välimuistissa [3]. Toisaalta IP-osoitteetkin voivat muuttua usein, joten välimuisti tulee pitää tuoreena.

Toinen esimerkki tässä esityksessä vähälle huomiolle jääneestä yksityiskohdasta on URL:ien kanonisointi [3]. Useat erilaiset URL:it voivat osoittaa samaan sivuun. URL:ien kanonisointi tarkoittaa URL:n muuntamista kanonisoituun muotoon siten, että yhteen sivuun voidaan viitata vain yhdellä kanonisella muodolla. Kanonisoinnissa esimerkiksi muunnetaan kaikki kirjaimet pieniksi kirjaimiksi, poistetaan URL:in lopusta ankkuriosa, lisätään mahdollisesti oletusportti ja lisätään mahdollisesti loppuun '/'-merkki. Lisäksi voidaan käyttää hyödyksi tietoa tunnetuista peilipalvelimista.

Chon väitöskirjan julkaisemisen jälkeen hakuroboteista on tehty tutkimusta erityisesti liittyen annettuun aiheeseen keskittyviin robotteihin [3] ja piilotetussa verkossa toimiviin hakurobotteihin [2, 4]. Piilotetulla verkolla tarkoitetaan sitä osaa WWW:stä, joka on saatavilla ainoastaan erilaisten lomakkeiden tai muiden vastaavien kautta. Esimerkiksi Liddle et. al. [2] ovat kehittäneet järjestelmän, joka tekee riittävän monia kyselyitä lomakkeella saadakseen haettua suurimman osan lomakkeen taakse piilotetusta informaatiosta.

# Viitteet


[1] J. Cho. *Crawling the web: discovery and maintenance of large-scale web data*. PhD thesis, Stanford University, 2001.

[2] S. W. Liddle, D.W. Embley, D. T. Scott, and S. Ho Yau. Extracting data behind web forms. In *Advanced Conceptial Modeling Techniques: ER 2002 Workshops*, pages 402–413, 2002.

[3] G. Pant, P. Srinivasan, and F. Menczer. Crawling the web. In *Web Dynamics*. Springer, 2004.

[4] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th international conference on very large data bases*, pages 129–138, 2001.

[5] The web robots pages. http://www.robotstxt.org/wc/robots.html.