Verkkoharavoinnin menetelmiä

Heidi Jauhiainen

Kandidaatintutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 19. joulukuuta 2013

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

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

Tekijä — Författare — Author
Heidi Jauhiainen

Työn nimi — Arbetets titel — Title

Verkkoharavoinnin menetelmiä

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Kandidaatintutkielma	19. joulukuuta 2013	18

Tiivistelmä — Referat — Abstract

Tämä tutkielma on katsaus verkkoharavoinnin menetelmiin. Verkkoharavoinnilla tarkoitetaan sivujen automatisoitua hakemista verkosta siten, että jo ladatulta sivulta poimitaan kaikki linkit muihin tiedostoihin, jotka sitten haetaan yksi kerrallaan. Verkkoharavointia voidaan tehdä joko periodisena, tietyn aikaa kestävänä, tai jatkuvana siten, että sivuja päivitetään niiden muututtua. Lisäksi haravointi voi olla yleistä tai johonkin tiettyyn aiheeseen tai muuhun sellaiseen kohdistettua. Tärkeä kysymys missä tahansa haravoinnissa on se, miten laadukkaat sivut saataisiin ladattua mahdollisimman aikaisessa vaiheessa. Jatkuvassa haravoinnissa tulisi lisäksi erityisesti tietää, missä tahdissa mitkäkin sivut muuttuvat ja mikä olisi paras strategia haun järjestämiseksi. Kohdennetussa haravoinnissa on puolestaan tärkeää löytää nimenomaan halutunlaiset tai haluttua aihetta käsittelevät tähdelliset sivut nopeasti.

Tässä tutkielmassa esitellään erilaisia menetelmiä, joilla erityyppisten verkkoharavointien hakujärjestystä voidaan optimoida. Tutkielman aluksi esitellään verkkoharavoinnin kestoon liittyvät käsitteet periodinen ja jatkuva haravointi sekä haravoinnin rajaamiseen liittyviä kysymyksiä. Sen jälkeen käydään läpi sivujen hakujärjestyksen optimointiin suositeltuja menetelmiä.

ACM Computing Classification System (CCS):
- **Information systems∼Web crawling**
- *Information systems∼Page and site ranking*
- Applied computing∼Digital libraries and archives

Avainsanat — Nyckelord — Keywords
Verkkoharavointi, hakujärjestys

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisältö

1 Johdanto **1**

2 Verkkoharavoinnin kesto **3**
2.1 Periodinen haravointi 3
2.2 Jatkuva haravointi 4

3 Verkkoharavoinnin rajaaminen **5**

4 Sivujen hakujärjestys **6**
4.1 Sivun tärkeys 7
4.2 Sivun tuoreus 9
4.3 Sivun tähdellisyys 11

5 Yhteenveto **13**

Lähteet **15**

1 Johdanto

Verkkoharava (web crawler), jota myös *robotiksi* tai *spideriksi* kutsutaan, on ohjelma, joka automaattisesti käy läpi internetissä olevia verkkosivuja [OlN10]. Verkkoharavointi lähtee liikkeelle annetuista *siemen-URL:eista* (seed). Ohjelma lataa yhden verkkosivun kerrallaan ja kerää siitä kaikki linkit sekä sivun omiin resursseihin, kuten kuviin, että sivun ulkopuolelle. Saadut uudet URL:it laitetaan jonoon, josta ne vuoron perään otetaan ladattaviksi. Siltä osin kuin ei toisin mainita, tässä johdantoluvussa esitetty katsaus verkkoharavoiden toimintaan yleisellä tasolla perustuu Olstonin ja Najorkin katsaukseen vuodelta 2010 [OlN10].

Verkkoharavoita käyttävät muun muassa hakukoneet, jotka indeksoivat löytämänsä sivut muodostaen tietokantoja, joista haun tulokset etsitään [OlN10]. Internet Archive puolestaan käyttää verkkoharavointia kerätäkseen verkkosivuja kokonaisuudessaan talteen tulevia sukupolvia ja tutkijoita varten. Verkkoharavoita voidaan käyttää myös esimerkiksi haluttujen sivujen tallentamiseen käytettäväksi verkkoyhteyden ulkopuolella.

Hyvän verkkoharavan rakentaminen vaatii usean seikan huomioonottamista [CGP98][OlN10]. Monet näistä seikoista riippuvat siitä, mitä haravoinnilla halutaan saada aikaiseksi. Kohteliaisuus on kuitenkin yksi asia, joka jokaisen verkkoharavan tulisi huomioida [ThS06]. Thelvall ja Stuart pohtivat artikkelissaan [ThS06] verkkoharavoinnin eettisiä kysymyksiä. Heidän mukaansa verkkoharavoiden tulisi välttää yksittäisten sivustojen ja palvelinten ylikuormitusta. Toisin sanoen haravan tulisi odottaa ennen kuin se tekee uudelleen pyynnön samalle palvelimelle. Verkkoharavoiden tulisi myös noudattaa niin sanottua *robotin rajausstandardia* (robots exclusion standard). Koster [Kos95] kertoo, että hän ehdotti tätä epävirallista standardia vuonna 1993. Standardin mukaan verkkosivujen ylläpitäjillä on mahdollisuus lisätä palvelimilleen niin

sanottu *robots.txt-tiedosto*, jossa he määrittävät mitkä haravat, jos mitkään, saavat hakea mitäkin sivuja [Kos94]. Tiedostossa voi myös spesifioida kuinka kauan verkkoharavan tulisi odottaa ennen seuraavaa hakua (ns. *crawl-delay directive*) [Robex]. Useimmat hakupalvelinten verkkoharavat noudattavat nykyään kyseistä standardia [Robex].

Verkkoharavointi voidaan hajauttaa eri prosesseille tai koneille [OlN10]. Jotta samaa sivua ei haettaisi useaan otteeseen ja haravointi noudattaisi edelleen edellä mainittuja kohteliaisuussääntöjä, täytyy jokaisen prosessin olla vastuussa omasta URL:ien osa-alueestaan. Päätettäväksi jää tällöin tapa, jolla prosessit välittävät toisille prosesseille sellaiset löytämänsä URL:it, jotka eivät kuulu niiden omaan alueeseen. Turhilta yhteydenotoilta säästytään, jos jokainen prosessi pitää lisäksi listaa jo välittämistään URL:eista.

Yksi tekijä, joka vaikuttaa verkkoharavoinnin tulokseen, on keräyksen aloittavien siemen-URL:ien valinta [ZDG09]. Eräs yksinkertainen mutta yleisesti käytetty menetelmä on linkkikirjaston pääsivun valitseminen haravoinnin siemeneksi [OlN10]. Zheng ja kumppanit [ZDG09] esittelevät lisäksi useita algoritmeja siemen-URL:ien valitsemiseen internetin verkkoarkkitehtuurin pohjalta. Tässä katsauksessa en kuitenkaan käsittele tätä asiaa tarkemmin.

Verkkoharavointiin liittyy läheisesti kysymys järjestyksestä, jossa sivut haetaan, eli siitä miten saataisiin kerättyä haravoinnin tarkoituksen kannalta tärkeät sivut mahdollisimman aikaisessa vaiheessa [OlN10]. Verkkoharavointia voidaan tehdä siten, että jokainen sivu haetaan ainoastaan kerran ja tietyn rajan jälkeen haravointi loppuu. Toinen tapa tehdä verkkoharavointia on jatkuva haravointi, jossa jo haettu sivu palautetaan haettavien sivujen jonoon ja joka jatkuu periaatteessa ikuisesti. Tällöin tulee erityisen tärkeäksi kysymys siitä, kuinka usein sivut muuttuvat sekä milloin ja missä järjestyksessä ne

pitäisi hakea uudelleen. Kumpaakin haravointitapaa voidaan käyttää yleiseen verkkosivujen keräämiseen tai jollakin lailla rajattuun haravointiin.

Tässä katsauksessa yritän selvittää, mitkä ovat verkkoharavoinnin tekemisen päätavat ja tärkeimmät menetelmät. Alla käsittelen ensin haun keston, mutta myös tarkoituksen, perusteella eroavat verkkoharavointitavat periodinen haravointi ja jatkuva haravointi. Sen jälkeen esittelen verkkoharavoinnin rajaamiseen liittyviä kysymyksiä. Lopuksi käyn läpi sivujen parhaan hakujärjestyksen päättelemiseen esitettyjä ratkaisuja.

2 Verkkoharavoinnin kesto

2.1 Periodinen haravointi

Verkkoharavointia, jossa jokainen sivu haetaan ainoastaan kerran ja joka päättyy tietyn ajan kuluttua, kutsutaan muun muassa *periodiseksi haravoinniksi*, englanniksi periodic crawling [ChG00a]. Tällaista haravointia voidaan käyttää ikään kuin kuvan ottamiseen internetistä tai halutusta osasta sitä [OlN10], ja sitä kutsutaankin joskus myös nimellä snapshot crawling [Sig05a] [Sig05b]. Kolmas englannin kielessä käytetty termi kyseiselle haravointitavalle on batch crawling [OlN10]. Periodisessa haravoinnissa, kun sivu on ladattu ja siitä on kerätty halutut tiedot, se poistetaan haettavien URL:ien jonosta [Sig05a]. Verkkoharavan täytyy kuitenkin pitää kirjaa jo nähdyistä sivuista, jotta sivua ei haettaisi uudelleen, kun sille löytyy linkki joltain myöhemmin tarkastellulta sivulta [OlN10].

Periodinen haku ei siis tuota duplikaatteja, mutta haravointi voidaan toistaa, jolloin edellisen kerran jälkeen sivuissa tapahtuneet muutokset saadaan talteen [Sig05a][OlN10]. Sigurðsson [Sig05a] kirjoittaa verkkoharava Heritrixiä käsittelevässä pro gradu -tutkielmassaan, että tällaisen uudelleenharavoinnin

pitäisi tuottaa suunnilleen sama sisältö kuin edellisenkin. Kuitenkin omien kokemusteni perusteella ainakin tämän hetkisessä Heritrixissä peräkkäiset, samasta siemenestä lähtevät haravoinnit voivat olla hyvinkin erilaisia. Joka tapauksessa Sigurðsson [Sig05b] suosittelee periodista haravointia erityisesti suurikokoisiin keräyksiin, joilla pyritään löytämään sivuja mahdollisimman laajasti, sillä se vaatii vähemmän kullakin hetkellä tallennettavaa tilatietoa kuin alla esitelty jatkuva haravointi. Periodista haravointia käyttävätkin muun muassa monet kansalliskirjastot [Sig05a].

2.2 Jatkuva haravointi

Toinen verkkoharavoinnin tapa on *jatkuva haravointi* (continuous crawling [EMT01][Sig05a]), joka ei periaatteessa ole koskaan valmis. Cho ja Garcia-Molina [ChG00a] käyttävät tällaisesta verkkoharavasta nimitystä steady crawler. Toisin kuin periodisessa haravoinnissa, jatkuvassa haravoinnissa ei pyritä mahdollisimman laajaan sivujen tallennukseen vaan niissä tapahtuvien muutosten taltiointiin, ja sitä kutsutaankin myös nimellä incremental crawling [EMT01]. Muutosten tallentamiseksi sivuilla täytyy vierailla useammin kuin kerran, joten sivua ei poisteta haettavien URL:ien jonosta [Sig05a].

Jatkuva haravointi alkaa kuin periodinen haravointi [Sig05a], mutta vähitellen osa ajasta käytetään siihen, että vieraillaan jo löydetyillä sivuilla yhä uudelleen [EMT01][OlN10]. Vaikka tällaisen haravan täytyykin pystyä poistamaan jonosta sellaiset sivut, joita ei enää löydetä [OlN10], sen haettavien URL:ien jono kasvaa kasvamistaan mikäli hakualuetta ei ole mitenkään rajattu [Sig05a]. Jos verkkoharava pystyy arvioimaan eri sivujen muutostahdin, sen ei kuitenkaan tarvitse aina vierailla kaikilla sivuilla [ChG00a].

3 Verkkoharavoinnin rajaaminen

Verkkoharavoinnissa ladattavat sivut voidaan rajata johonkin tiettyyn aiheeseen, maantieteelliseen alueeseen, muotoon tai muuhun sellaiseen [OlN10]. Olston ja Najork [OlN10] käyttävät tällaista taktiikkaa hyödyntävästä verkkoharavasta englanninkielistä nimitystä *scoped crawler*. Heidän mukaansa tiettyyn aiheeseen *kohdennettu haravointi* (topical tai focused crawling) on näistä eniten tutkittu, mutta muilla tavoin rajatut haravoinnit käyttävät usein samanlaisia menetelmiä.

Kohdennetussa haravoinnissa pyritään siis yksinkertaisesti löytämään ne sivut, jotka käsittelevät jotain etukäteen määriteltyä aihetta [CBD99][DCL00]. Vaikeaksi asian tekee löytää nopeasti maksimaalinen määrä relevantteja sivuja käymättä läpi kaikkia sivustoja [DCL00]. Yleinen verkkoharavointi ei tähän sovellu, sillä vaikka sellainen verkkoharava aloittaisi relevanteilta sivuilta, se pian kiertelisi sivustoja, joita ei lainkaan tarvita [CBD99].

Menzerin ja kumppaneiden [MPS04] mukaan liian *ahne* (greedy) kohdennettu haravointi ei myöskään välttämättä ole hyvä, sillä se saattaa lopettaa keräyksen käymättä läpi kaikkia relevantteja sivustoja; tarvitaan siis tasapainoinen hakujärjestys. Kohdennettuun verkkoharavointiin liittyykin läheisesti kysymys siitä, miten sivujen haku järjestetään niin, että kyseiselle haravoinnille tähdelliset sivut löydetään mahdollisimman nopeasti.

4 Sivujen hakujärjestys

Yksinkertainen tapa saada kattava listaus internetin sivuista on *leveyssuuntainen läpikäynti* (breadth-first search), jota Pinkerton [Pin94] käytti vuonna 1994 rakentamassaan hakukoneessa. Leveyssuuntainen läpikäynti tarkoittaa verkkoharavoinnin yhteydessä sitä, että sivut haetaan siinä järjestyksessä, jossa ne löydettiin ensimmäisen kerran [NaW01]. Verkkoharavoinnissa ei kuitenkaan aina ole mahdollista hakea kaikkia sivuja, joten on tarpeellista löytää haravoinnin tarkoituksen kannalta tärkeät sivut mahdollisimman nopeasti [CGP98].

Olston ja Najork [OlN10] listaavat kolme seikkaa, joiden perusteella voidaan päättää, missä järjestyksessä sivut haetaan. Päätös voidaan tehdä sen perusteella, kuinka tärkeä tai laadukas sivu on verrattuna toiseen sivuun. Tällaista päätöksentekoa voidaan käyttää sekä periodisessa että jatkuvassa haravoinnissa. Silloin kun tehdään jatkuvaa haravointia, yksittäisten sivujen muutostiheys vaikuttaa usein siihen, missä järjestyksessä sivut ladataan uudelleen. Kohdennetussa haravoinnissa ratkaisevana tekijänä on puolestaan se, kuinka relevantti sivu on suhteessa haettuun.

Tämän katsauksen puitteissa ei ole mahdollista esitellä kaikkia verkkoharavoinnin hakujärjestyksen optimointiin esitettyjä menetelmiä. Keskityn siksi vanhempiin ja yksinkertaisempiin menetelmiin, joihin monet uudemmat algoritmit perustuvat. Tukeudun menetelmien valinnassa erityisesti Olstonin ja Najorkin katsaukseen verkkoharavoinnista [OlN10]. Alla esittelen ensin ratkaisuja, joilla löydetään niin sanotut tärkeät sivut mahdollisimman aikaisessa vaiheessa. Sitten käyn läpi sivujen tuoreutta edistäviä menetelmiä ja lopuksi käsittelen metodeja, joilla löydetään tehokkaasti kohdennetulle haravoinnille tähdelliset sivut.

4.1 Sivun tärkeys

Cho ja kumppanit [CGP98] antavat 'tärkeälle' sivulle useita määritelmiä. Sivu voidaan esimerkiksi tulkita sitä tärkeämmäksi, mitä enemmän sille on linkkejä muilta sivuilta (ns. *backlink count*). Sivulle tulevalle linkille voidaan lisäksi antaa enemmän arvoa, jos se tulee tärkeältä sivulta. Tätä kutsutaan nimellä *PageRank* (ks. myös [PBM98][ZYG09]). Sivu voidaan määritellä tärkeäksi myös siitä lähtevien linkkien määrän (ns. *Forward Link Count*) tai sen URL:in ulkoisen rakenteen (ns. *Location Metric*) perusteella.

Cho ja kumppanit [CGP98] testasivat leveyssuuntaista läpikäyntiä, backlink-järjestämistä sekä PageRank-järjestämistä ja toteavat, että PageRank-järjestäminen toimii vähän paremmin kuin leveyssuuntainen läpikäynti. Castillon ja kumppaneiden [CMR04] mielestä Chon ja kumppaneiden PageRank-järjestäminen voi kuitenkin käytännössä olla hidasta. Najork ja Wiener [NaW01] arvostelevat PageRankin aikavaativuuden lisäksi sen tilavaativuutta. He tutkivat itse leveyssuuntaisen läpikäynnin tehokkuutta isommalla haravoinnilla kuin Cho ja kumppanit [CGP98] tekivät. He toteavat, että leveyssuuntainen läpikäynti löytää tärkeiksi luokitellut sivut hyvin haravoinnin alussa, mutta huonommin sivujen määrän kasvaessa (ks. myös [BCM05]). Najork ja Wiener suosittelevat kuitenkin leveyssuuntaista läpikäyntiä, koska tärkeiden sivujen löytäminen on erityisen tähdellistä haravoinnin alussa [NaW01].

Abiteboul ja kumppanit [APC03] kehittivät *OPIC*-algoritmin (On-line Page Importance Computation), joka laskee uudelleen sivun tärkeyden aina, kun siihen löytyy linkkejä muilta sivuilta. Siinä jokaisella sivulla on niin sanottu *cash*-arvo, joka jaetaan tasaisesti kaikille sivuille, joihin siitä on linkki. Hakujärjestyksessä seuraavana on sivu, jolla on suurin cash-arvo. Baeza-Yatesin ja kumppaneiden [BCM05] mukaan tämä menetelmä on nopeampi

kuin PageRank. Zareh Bidoki ja kumppanit [ZYG09] huomauttavat kuitenkin, että metodi on melko hidas, sillä jokainen sivu ladataan useaan kertaan.

Castillo ja kumppanit [CMR04] vertailivat erilaisia järjestämisalgoritmeja, muun muassa leveyssuuntaista läpikäyntiä ja muutamaa eri versiota PageRank-järjestämisestä. He tulivat siihen tulokseen, että yksinkertainen mutta paras menetelmä on sellainen, jossa seuraavaksi haettava sivu otetaan jonosta, joka on sillä hetkellä pisin. Samat tutkijat, Baeza-Yates ja kumppanit [BCM05], kutsuvat myöhemmin tätä menetelmää nimellä *Larger-sites-first*. He laajensivat vertailunsa sisältämään backlink- ja OPIC-järjestämiset. OPIC ja Larger-sites-first olivat tässä vertailussa parhaita. Baeza-Yatesin ja kumppaneiden [BCM05] mukaan Larger-sites-first-menetelmän etuna on, että se vaatii vähemmän laskenta-aikaa ja tilaa kuin OPIC. He toteavat kuitenkin, että Larger-sites-first-menetelmän käyttäminen isoissa verkkoharavoinneissa on käytännössä haastavaa, sillä se pitää suuren määrän jonoja muistissa.

Zareh Bidoki ja kumppanit [ZYG09] puolestaan esittelivät algoritmin, jota he kutsuvat nimellä *FICA* (Fast Intelligent Crawling Algorithm). Siinä prioriteetti on sivuilla, joiden logaritminen etäisyys siemensivusta on pienin. Heidän mukaansa algoritmi simuloi käyttäjää, joka aloittaa nettisurffauksen joltain mielivaltaiselta sivulta mutta joka oppii sivujen valitsemista edetessään. Zareh Bidokille ja kumppaneille tärkeä sivu tarkoittaakin sivua, joka on käyttäjien joukossa suosittu [ZYG09]. Heidän mukaansa FICA on nopeampi löytämään tärkeät sivut kuin edellä mainitut menetelmät, joihin sitä verrattiin.

4.2 Sivun tuoreus

Periodisella haravoinnilla saatua sivukokoelmaa voidaan päivittää suorittamalla haravointi uudelleen. Tämä ei kuitenkaan ota huomioon sitä seikkaa, että verkossa olevat sivut muuttuvat eri tahtiin [ChG00a]. Sitä kuinka usein verkkosivuja muutetaan, ovat tutkineet muun muassa Coffman Jr. ja kumppanit [CLW98], Edwards ja kumppanit [EMT01] sekä Cho ja Garcia-Molina [ChG03b].

Cho ja Garcia-Molina vertasivat eräässä toisessa tutkimuksessaan [ChG00b] *tasaista* (uniform) ja sivujen muutosvauhtiin *suhteutettua* (proportional) sivujen päivittämistä (ks. myös [ChG03a]). Koska yllättäen tasainen sivujen päivittämistaktiikka oli heidän testeissään parempi, he kehittivät uuden menetelmän, jota he kutsuvat *optimaaliseksi sivujen päivittämiseksi* (optimal synchronization). Siinä usein muuttuvia sivuja päivitetään harvemmin kuin harvoin muuttuvia. Usein muuttuvat sivut ovat nimittäin joka tapauksessa epätuoreita suurimman osan ajasta ollessaan varastoituina, joten kannattaa ladata uudelleen sivut, jotka pystytään pitämään tuoreina [OlN10].

Cho ja Ntoulas [ChN02] puolestaan kehittivät otantaa hyväksikäyttävän algoritmin. Siinä jokaisesta sivustosta otetaan näytteeksi tietty määrä sivuja, joiden tuoreuden perusteella päätetään, tarvitseeko sivusto ladata uudelleen. Lisäksi he vertasivat *suhteellista* (proportional) ja *ahnetta* (greedy) lataamista. Heidän suhteellisessa menetelmässään jokaiselta sivustolta tallennetaan sille lasketun muutosprosentin verran sivuja. Heidän ahne menetelmänsä tarkoittaa puolestaan, että kulloinkin suurimman muutosprosentin omaavalta sivustolta ladataan kaikki sivut ennen kuin siirrytään seuraavaan sivustoon. Cho ja Ntoulas kehittivät ahneesta menetelmästään lisäksi adaptiivisen version [ChN02]. Siinä sivuston muutosprosentti arvioidaan jokaisen tallennuskerran aikana uudelleen kaikkien siihen mennessä tehtyjen otantojen perusteella.

Näistä kolmesta menetelmästä ahne osoittautui heidän testeissään parhaaksi. Tan ja Mitra [TaM10] huomauttavat kuitenkin, ettei tämä menetelmä ota huomioon sitä, että yhden sivuston kaikki sivut eivät välttämättä muutu samaan tahtiin.

Edellä kuvaillut menetelmät [ChG00b][ChN02] on tarkoitettu tietokantaan tallennettujen sivujen päivittämiseen. Wolf ja kumppanit [WSY02] pohtivat puolestaan hakukoneiden indeksin tuoreutta. Jotta vältyttäisiin noloilta linkinpainalluksilta, joissa haun tuloksena saatu linkki ei toimikaan tai ei sisällä haluttua tietoa, he ehdottavat tehtävän jakamista kahteen osaan. Ensinnäkin pitää päätellä, mikä olisi kunkin sivun paras päivitysväli. Toiseksi täytyy laskea optimaalinen aikataulu, jolla kaikki sivut saataisiin päivitettyä mahdollisimman hyvin suhteessa päivitysväliinsä.

Pandeyn ja Olstonin [PaO05] mielestä pelkkä sivujen tuoreus ei välttämättä takaa hakukoneen käyttäjän kannalta parasta tulosta. He ovat sitä mieltä, että tähän vaikuttaa myös indeksin laatu. Heidän mielestään laadukkaassa indeksissä sivut ovat paitsi ajan tasalla myös sellaisia, joita käyttäjä klikkaa hakutuloksista ja jotka siis ovat tuloksissa alkupäässä. He kehittivät menetelmän, jossa otetaan huomioon käyttäjien todennäköinen käytös ja lasketaan sivun mahdollisen uudelleenlataamisen vaikutus indeksin laatuun. Tämä menetelmä soveltuu kuitenkin ainoastaan hakukoneille [PaO05].

Barbosa ja kumppanit [BSC05] esittelivät menetelmän, jossa sivun muuttumisen frekvenssi päätellään sivun sisäisten staattisten ominaisuuksien, kuten linkkien määrän ja Last-Modified -tekstin olemassaolon, perusteella. Joka kerta, kun sivu ladataan uudelleen, sen muuttumisennustetta muokataan kerättyjen todellisten muuttumistietojen perusteella. Sivut luokitellaan ennusteiden mukaan neljään eri ryhmään, joilla jokaisella on oma päivittämisaikataulunsa. Tan ja Mitra [TaM10] kehittivät tätä menetelmää edelleen käyttämällä sataa

ryhmää, joilla kullakin on oma aikataulunsa. Verkkoharava lataa otannan kunkin ryhmän sivuista, minkä perusteella päätellään, ovatko ryhmän sivut muuttuneet edellisen päivittämisen jälkeen.

Edellä mainitut Olston ja Pandey kehittivät myöhemmin [OlP08] päivittämisalgoritmin, joka ottaa huomioon sivun sisällön osien *pitkäaikaisuuden* (longevity). Jotkut sivuilla olevat asiat ovat nimittäin jatkuvasti muuttuvaa tietoa, kuten mainoksia, kun taas osa sisällöstä on staattisempaa ja tärkeämpää. Olstonin ja Pandeyn algoritmi tutkii ladatut sivut *paloina* (shingles, ks. [BCM97]), pitää kirjaa kullakin sivulla tapahtuvista muutoksista (ns. *change profile*) ja tekee latauspäätöksen näiden tietojen perusteella.

4.3 Sivun tähdellisyys

Verkkoharavat voidaan ennen niiden käynnistämistä opettaa lataamaan tietynlaisia sivuja. Esimerkiksi Chakrabarti ja kumppanit [CBD99] käyttivät kohdennetun verkkoharavansa opettamiseen portaalisivustoja, kun taas Diligenti ja kumppanit [DCL00] kouluttivat omaansa verkosta tehdyillä hauilla. Kuitenkin tiedetään, että tiettyä aihetta käsittelevät sivut sisältävät usein linkkejä samaa aihetta käsitteleviin sivuihin [CBD99][AGY01]. Monet kohdennettua haravointia varten kehitetyt algoritmit käyttävätkin tätä havaintoa hyödykseen etsiessään sivuja, jotka ovat haun kannalta tähdellisiä [OlN10].

Yksi varhainen kohdennetun verkkoharavoinnin menetelmä on De Bran ja Postin [BrP94] kehittämä niin sanottu *kala-etsintä* (fish search). Siinä haulle relevantista dokumentista löydetyt linkit asetetaan jonon kärkeen ja asiaankuulumattomasta saadut jonon päähän. Linkkejä lisättäessä dokumentin sivuston ulkopuoliset linkit saavat lisäksi korkeamman prioriteetin kuin sivuston sisäiset. Linkki voidaan siirtää jonon kärkeen, jos se löydetään uudelleen relevantimmalta sivulta kuin aikaisemmin. Mikään linkki ei kuitenkaan

ole listalla kahteen kertaan. Hersovicin ja kumppaneiden [HJM98] mukaan kala-etsinnän ongelmana on, että siinä monet sivut saavat saman arvon, jolloin jotkut hyvinkin tähdelliset sivut joutuvat odottamaan vuoroaan. He kehittivät kala-etsinnästä version, jonka pitäisi erotella eritasoiset tähdelliset sivut paremmin. Menetelmässä, jota he kutsuvat *hai-etsinnäksi* (shark-search), sivulle lasketaan binäärisen arvon sijaan nollasta yhteen oleva sumea arvo. Sivun arvoon vaikuttaa sen sivun arvo, josta siihen päästään, ja sivulle tuovan linkin valintatekstin tähdellisyys kyseiselle haulle.

Menczer [Men97] kehitti algoritmin, jossa linkin relevanttius kyseiselle haulle päätellään sen ympärillä olevista sanoista. Aggarwal ja kumppanit [AGY01] puolestaan kehittivät haravointisysteemin, jossa linkin tähdellisyys arvioidaan muidenkin ominaisuuksien perusteella. Arviointiin vaikuttaa itse URL:in teksti ja muun muassa niiden sivujen sisältö, joista kyseinen linkki löydetään. Lisäksi tutkitaan linkin jo ladattuja *sisarlinkkejä*, eli samalta sivulta lähteviä linkkejä, ja sitä kuinka moni niistä on tähdellinen. Algoritmi myös tutkii, onko kyseisen haun kriteerit täyttävillä sivuilla tapana linkittyä muille tähdellisille sivuille (ns. *short range topic locality*). Näin menetelmä oppii kohdentamaan hakua haravoinnin aikana halutun aiheen mukaisesti.

Cho ja kumppanit [CGP98] testasivat leveyssuuntaisen läpikäynnin, backlink-järjestämisen ja PageRank-järjestämisen tehokkuutta tähdellisten sivujen löytämisessä. Heidän testeissään leveyssuuntainen läpikäynti oli paras keino löytää etukäteen määriteltyä aihetta käsitteleviä sivuja. Tässä tehtävässä auttaa heidän mukaansa niiden URL:ien priorisointi, jotka sisältävät tai joiden valintatekstissä on haettu sana. Myös muutaman hypyn päässä tähdellisestä sivusta olevat sivut kannattaa heidän mielestään panna etusijalle.

5 Yhteenveto

Verkkoharavointia voidaan tehdä joko periodisesti tai jatkuvana. Periodisen haravoinnin etuna on, että siinä jo ladattujen sivujen URL:it poistetaan hakujonosta, joten sen tilavaativuus on pienempi kuin jatkuvan haravoinnin, jossa URL:it palautetaan jonoihin seuraavaa hakukertaa varten. Yksinkertainen tapa tehdä periodista haravointia on leveyssuuntainen läpikäynti. Kun halutaan löytää niin sanotut tärkeät sivut ensin, voidaan käyttää myös muita menetelmiä, kuten PageRank, OPIC tai FICO. Monet tutkijat suosittelevat kuitenkin leveyssuuntaista läpikäyntiä, sillä se vie vähemmän resursseja kuin muut menetelmät [FCV08] ja löytää kuitenkin tärkeät sivut nopeasti [NaW01].

Jatkuvaa haravointia käytetään, kun halutaan päivittää jo haetuilta sivuilta saatuja tietoja sivujen muututtua. Sivut eivät kuitenkaan muutu kaikki samaan tahtiin, joten haasteena on päätellä, onko sivu muuttunut edellisen lataamisen jälkeen. On olemassa monenlaisia menetelmiä, joissa verkkoharavalle opetetaan, miten mitkäkin sivut muuttuvat. Voidaan esimerkiksi pitää kirjaa siitä, onko sivu muuttunut, ja vähitellen muokata hakuaikataulua sen mukaisesti. Voidaan myös ottaa jostakin ryhmästä otanta, jonka perusteella päätetään ladataanko kaikki ryhmän sivut. Aina ei kuitenkaan kannata hakea uudelleen niitä sivuja, jotka muuttuvat jatkuvasti, sillä niistä saadut tiedot ovat pian taas vanhentuneita.

Jos jatkuvaa haravointia ei rajata millään tavoin, sen tilavaativuus on suuri. Sekä jatkuva että periodinen haravointi voidaan rajata etsimään esimerkiksi tiettyä aihetta käsitteleviä sivuja. Tällöin on tärkeää löytää nuo sivut mahdollisimman nopeasti, jottei tarvitsisi käydä kaikkia internetissä

olevia verkkosivuja läpi. Yksinkertainen tapa on priorisoida linkkejä, jotka löydetään tähdellisiksi todetuilta sivuilta. Lisäapua saa tutkimalla itse linkkiä, sen ympärillä olevaa tekstiä tai linkin jo ladattuja sisarsivuja.

Lähteet

[AGY01] C.C. Aggarwal, F. Al-Garawi, P.S. Yu, Intelligent crawling on the World Wide Web with arbitrary predicates, *Proc. of the 10th Internat. Conf. on World Wide Web*, IW3C2, Hong Kong, 2001, sivut 96–105.

[APC03] S. Abiteboul, M. Preda, G. Cobena, Adaptive on-line page importance computation, *Proc. of the 12th Internat. Conf. on World Wide Web*, IW3C2, Budapest, 2003, sivut 280–290.

[BCM97] A.Z. Broder, S.C. Glassman, M.S. Manasse, G. Zweig, Syntactic clustering of the Web, *Computer Networks and ISDN Systems* 29,8-13 (1997), sivut 1157–1166.

[BCM05] R. Baeza-Yates, C. Castillo, M. Marin, A. Rodriguez, Crawling a country: better strategies than breadth-first for web page ordering, *Proc. of the 14th Internat. Conf. on World Wide Web*, IW3C2, Chiba, 2005, sivut 864–872.

[BSC05] L. Barbosa, A.C. Salgado, F. de Carvalho, J. Robin, J. Freire, Looking at both the present and the past to efficiently update replicas of web content, *Proc. of the 7th Annual ACM Internat. Workshop on Web Information and Data Management*, Bremen, 2005, sivut 75–80.

[BrP94] P.M.E. De Bra, R.D.J. Post, Information retrieval in the World-Wide Web: making client-based searching feasible, *Proc. of the 1st Internat. Conf. on World Wide Web*, IW3C2, Geneva, 1994, sivut 183–192.

[CBD99] S. Chakrabarti, M. Van den Berg, B. Dom, Focused crawling: a new approach to topic-specific Web resource discovery, *Computer Networks* 31,11 (1999), sivut 1623–1640.

[CGP98] J. Cho, H. Garcia-Molina, L. Page, Efficient crawling through URL ordering, *Proc. of the 7th Internat. Conf. on World Wide Web*, IW3C2, Brisbane, 1998, sivut 161–172.

[ChG00a] J. Cho, H. Garcia-Molina, The evolution of the Web and implications for an incremental crawler, *Proc. of the 26th Internat. Conf. on Very Large Data Bases*, VLDB Endowment, Cairo, 2000, sivut 200–209.

[ChG00b] J. Cho, H. Garcia-Molina, Synchronizing a database to improve freshness, *Proc. of the 2000 ACM SIGMOD Internat. Conf. on Management of data*, Dallas, 2000, sivut 117–128.

[ChG03a] J. Cho, H. Garcia-Molina, Effective page refresh policies for Web crawlers, *ACM Transactions on Database Systems* 28,4 (2003), sivut 390–426.

[ChG03b] J. Cho, H. Garcia-Molina, Estimating frequency of change, *ACM Transactions of Internet Technology* 3,3 (2003), sivut 256–290.

[ChN02] J. Cho, A. Ntoulas, Effective change detection using sampling, *Proc. of the 28th Internat. Conf. on Very Large Data Bases*, VLDB Endowment, Hong Kong, 2002, sivut 514–525.

[CLW98] E.G. Coffman Jr., Z. Liu, R.R. Weber, Optimal robot scheduling for web search engines, *Journal of Scheduling* 1,1 (1998), sivut 15–29. (= INRIA Research Report, RR 3317, 1997).

[CMR04] C. Castillo, M. Marin, A. Rodriguez, R. Baeza-Yates, Scheduling algorithms for web crawling, *Proc. of the WebMedia & LA-Web 2004 Joint Conf.*, IEEE, Ribeirão Preto, 2004, sivut 10–17.

[DCL00] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori, Focused crawling using context graphs, *Proc. of the 26th Internat. Conf. on Very Large Data Bases*, VLDB Endowment, Cairo, 2000, sivut 527–534.

[EMT01] J. Edwards, K. McCurley, J. Tomlin, An adaptive model for optimizing performance of an incremental Web crawler, *Proc. of the 10th Internat. Conf. on World Wide Web*, IW3C2, Hong Kong, 2001, sivut 106–113.

[FCV08] D. Fetterly, N. Craswell, V. Vinay, Search effectiveness with a breadth-first crawl, *Proc. of the 31st Annual Internat. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Singapore, 2008, sivut 755–756.

[HJM98] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, S. Ur, The shark-search algorithm. An application: tailored Web site mapping, *Computer Networks and ISDN Systems* 30,1 (1998), sivut 317–326.

[Kos94] M. Koster, A standard for robot exclusion, 1994. http://www.robotstxt.org/orig.html [29.10.2013].

[Kos95] M. Koster, Robots in the Web: threat or treat?, 1995. http://www.robotstxt.org/threat-or-treat.html [29.11.2013].

[Men97] F. Menczer, ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods, *Proc. of the 14th Internat. Conf. on Machine Learning*, IMLS, Nashville, 1997, sivut 227–235.

[MPS04] F. Menczer, G. Pant, P. Srinivasan, Topical web crawlers: Evaluating adaptive algorithms, *ACM Transactions on Internet Technology* 4,4 (2004), sivut 378–419.

[NaW01] M. Najork, J.L. Wiener, Breadth-first crawling yields high-quality pages, *Proc. of the 10th Internat. Conf. on World Wide Web*, IW3C2, Hong Kong, 2001, sivut 114–118.

[OlN10] C. Olston, M. Najork, Web crawling, *Foundations and Trends in Information Retrieval*, 4,3 (2010), sivut 175–246.

[OlP08] C. Olston, S. Pandey, Recrawl scheduling based on information longevity, *Proc. of the 17th Internat. Conf. on World Wide Web*, IW3C2, Beijing, 2008, sivut 437–446.

[PaO05] S. Pandey, C. Olston, User-centric Web crawling, *Proc. of the 14th Internat. Conf. on World Wide Web*, IW3C2, Chiba, 2005, sivut 401–411.

[PBM98] L. Page, S. Brin, R. Motwani, T. Winograd, The PageRank citation ranking: bringing order to the Web, Technical Report, Stanford InfoLab, Stanford, 1999.

[Pin94] B. Pinkerton, Finding What People Want: Experiences with the WebCrawler, *Proc. of the 2nd Internat. Conf. on World Wide Web*, IW3C2, Chicago, 1994. [Myös http://www.thinkpink.com/bp/webcrawler/www94.html].

[Robex] Robot exclusion standard. http://en.wikipedia.org/wiki/Robots_exclusion_standard [29.10.2013].

[Sig05a] K. Sigurðsson, Adaptive Revisiting with Heritrix, MS thesis, Department of Computer Science, University of Iceland, 2005. [Myös http://skemman.is/en/stream/get/1946/2071/6500/1/Adaptive_Revisiting_with_Heritrix_-_Thesis.pdf].

[Sig05b] K. Sigurðsson, Incremental crawling with Heritrix, Paper presented at the 5th Internat. Web Archiving Workshop, Vienna, Austria, 2005, http://iwaw.europarchive.org/05/papers/iwaw05-sigurdsson.pdf.

[TaM10] Q. Tan, P. Mitra, Clustering-based incremental web crawling, *ACM Transactions on Information Systems* 28,4 (2010), sivut 1–27.

[ThS06] M. Thelwall, D. Stuart, Web crawling ethics revisited: Cost, privacy, and denial of service, *Journal of the American Society for Information Science and Technology* 57,13 (2006), sivut 1771–1779.

[WSY02] J.L. Wolf, M.S. Squillante, P.S. Yu, J. Sethuraman, L. Ozsen, Optimal crawling strategies for web search engines, *Proc. of the 11th Internal. Conf. on World Wide Web*, ACM, Honolulu, 2002, sivut 136–147.

[ZYG09] A.M. Zareh Bidoki, N. Yazdani, P. Ghodsnia, FICA: A novel intelligent crawling algorithm based on reinforcement learning, *Web Intelligence and Agent Systems* 7,4 (2009), sivut 363–373.

[ZDG09] S. Zheng, P. Dmitriev, C.L. Giles, Graph-based seed selection for web-scale crawlers, *Proc. of the 18th ACM Conf. on Information and Knowledge Management*, Hong Kong, 2009, sivut 1967–1970.