hyväksymispäivä arvosana

arvostelija

Assosiaatiosääntöjen louhinnan tehostaminen

Tuomas Tanner

Helsinki 27.03.2008

Tiedon louhinnan seminaari, kevät 2008
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedeta/Osasto – Fakultet/Sektion –Faculty/Section	Laitos – Institution – Department
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen laitos

Tekijä – Författare – Author

Tuomas Tanner

Työn nimi – Arbetets titel – Title

Assosiaatiosääntöjen louhinnan tehostaminen

Oppiaine – Läroämne – Subject

Tietojenkäsittelytiede

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Seminaaritutkielma	27.3.2008	18 sivua

Tiivistelmä – Referat – Abstract

Tämä tutkielma esittelee kaksi parannusta perinteiseen assosiaatiosääntöjen louhintaan apriori-algoritmilla. Assosiaatiosääntöjen louhinta suljettujen kattavien alkiojoukkojen avulla parantaa louhinnan tehokkuutta sekä prosessorikäytön että tietokantahakujen suhteen. Louhinnan aikavaatimus on myös vähemmän riippuvainen tietokantatransaktioiden alkioiden lukumäärästä kuin apriori-algoritmilla. Kirjoitelma käsittelee myös Galois-yhteyttä ja sen soveltamista assosiaatiosääntöjen louhinnassa.

δ-toleranssi-periaatteen avulla löydettyjen kattavien joukkojen ja niistä generoitavien assosiaatiosääntöjen määrää voidaan tehokkaasti karsia ilman, että assosiaatiosääntöjen tarkkuus kärsii. Tutkielma selittää δ-toleranssi-periaatteen ja esittelee algoritmeilla saavutettuja käytännön suorituskyvyn ja tulosten parannuksia.

Avainsanat – Nyckelord – Keywords

tiedon louhinta, assosiaatiosäännöt, apriori, Galois-yhteys, δ-toleranssi

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Assosiaatiosääntöjen louhinta on tiedonlouhinnan yksi keskeisistä sovellusalueista. Assosiaatiosääntöjen avulla voidaan havaita yhteyksiä tietokannan alkioiden välillä hyvin suurista tietokannoista. Menetelmän alkuperäinen ja eniten käytetty sovellusalue on ostoskorianalyysi, joka kertoo kuinka kaupasta ostetut tuotteet liittyvät toisiinsa. Esimerkiksi 80% asiakkaista, jotka ostivat sokeria ja muroja ostivat myös maitoa. Samaa menetelmää voidaan soveltaa yleisesti ongelmiin, joissa halutaan löytää tiettyjen tekijöiden välisiä yhteyksiä, kuten väestönlaskennassa (useiden eri faktoreiden yhtäaikainen vaikutus tiettyyn ominaisuuteen) ja lääketieteessä (oireiden ja tietyn sairauden välinen assosiaatio).

Ongelmana perinteisessä assosiaatiosääntöjen louhinnassa on ollut louhinnan hitaus varsinkin aineiston ollessa vahvasti korreloivaa (esimerkiksi väestölaskenta-aineistossa). Toinen vakavampi ongelma on ollut assosiaatiosääntölouhinnan huono skaalautuvuus tietokannan transaktioiden koon suhteen. Kolmas ongelma assosiaatiosääntöjen louhinnassa on ollut generoitujen sääntöjen suuri määrä, jolloin merkityksellisten sääntöjen löytäminen kaikkien sääntöjen joukosta on ollut vaikeaa.

Tässä seminaariraportissa esittelen kaksi menetelmää, jolla näitä ongelmia on ratkaistu. Assosiaatiosääntöjen louhinta suljettujen kattavien alkiojoukkojen avulla (luku 3) tehostaa assosiaatiosääntöjen louhintaa vahvasti korreloivassa aineistossa, sekä parantaa huomattavasti louhinnan tehokkuutta tietokannan transaktioiden koon kasvaessa [PBT99a ja PBT99b]. δ-toleranssin kattavien joukkojen avulla (luku 4) pystytään tehokkaasti karsimaan redundantteja assosiaatiosääntöjä kuitenkin samalla säilyttäen tuen ja luottamuksen tarkkuuden jäljelle jäävissä assosiaatiosäännöissä [CKN06 ja CKN08].

2 Assosiaatiosääntöjen louhinta apriori-algoritmilla

Assosiaatiosääntöjen idean esittelivät Agrawal ja kumppanit [AIS93, AgS94]. He määrittelevät assosiaatiosäännön implikaationa $X \Rightarrow Y$, missä X ja Y ovat jotkin alkiojoukot kaikista alkioiden joukosta I' tietokannassa D' ja $X \cap Y = \emptyset$. Assosiaatiosääntö on voimassa luottamuksella c, jos $c\%$ tietokannan D' transaktioista, jotka sisältävät X:n, sisältävät myös Y:n. Assosiaatiosäännöllä on tuki s, jos $s\%$ tietokannan D' transaktiois-

ta sisältää $X \cup Y$ [AgS94]. Luottamus näin ollen kuvaa säännön vahvuutta ja tuki tilastollista merkitsevyyttä [AIS93].

2.1 Apriori-algoritmi

Agrawal ja kumppanit kehittivät myös apriori-algoritmin assosiaatiosääntöjen tehokkaaseen louhintaan. Se perustuu apriori-periaatteeseen, jonka mukaan jokaisen kattavan joukon osajoukko on myös kattava. Aprirori-algoritmi toimii kahdessa vaiheessa [AgS94]:

1. Etsi kaikki alkiojoukot tietokannasta *D′*, joiden tuki on suurempi tai yhtäsuuri kuin minimituki (*minsupport*). Löydettyjä alkiojoukkoja kutsutaan kattaviksi joukoiksi.

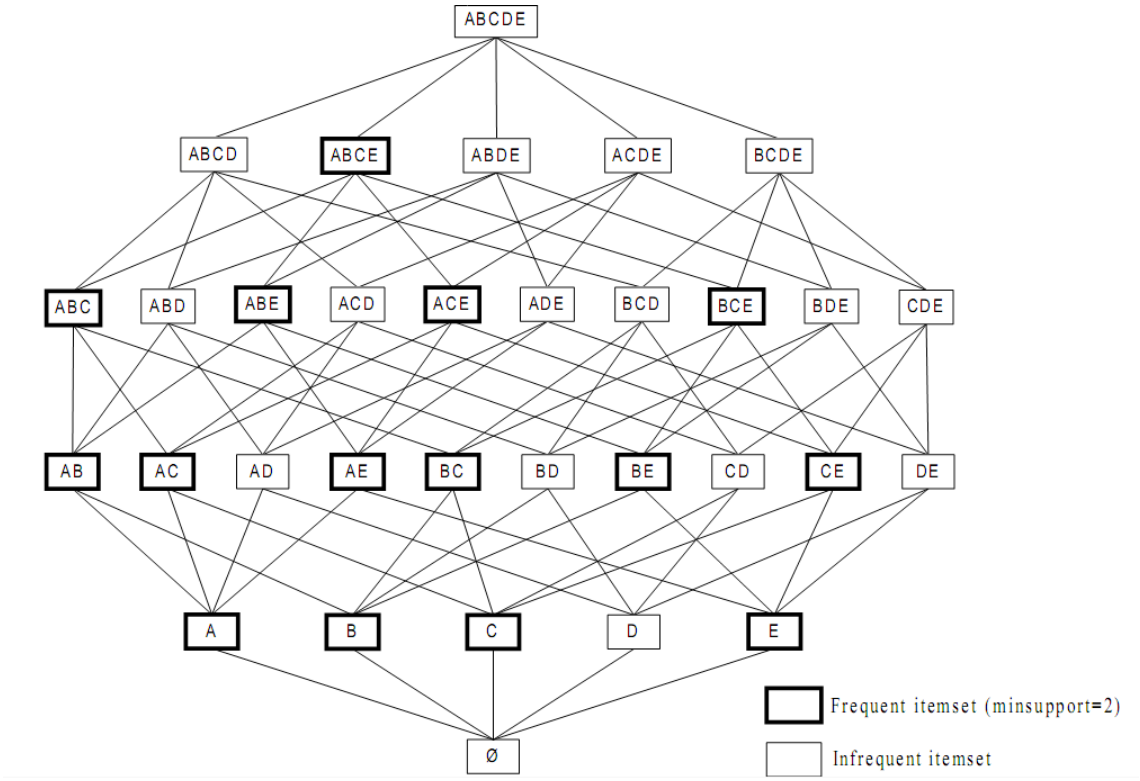
2. Hae jokaisen kattavan joukon *L* kaikki ei-tyhjät osajoukot (myös nämä ovat kattavia apriori-periaatteen mukaisesti). Generoi jokaiselle osajoukolle *A* assosiaatiosääntö $A \Rightarrow (L - A)$, jos säännön luottamus on suurempi tai yhtäsuuri kuin *minconfidence*.

Apriori-algoritmi hakee kattavat joukot ja generoi näistä assosiaatiosäännöt *k* iteraatiossa, missä *k* on suurimman kattavan joukon alkioiden lukumäärä. Ensimmäisellä iteraatiolla haetaan 1-alkioiset kattavat joukot ja jokainen iteraatio kasvattaa kattavien joukkojen kokoa yhdellä. Jokaisella iteraatiolla tietokanta käydään läpi kerran, joten Apriori ja muut sen kaltaiset algoritmit vaativat *k* -tietokannan läpikäyntiä kattavien alkiojoukkojen löytämiseen [PBT99b].

Kuvassa 1 on esimerkkitietokanta *D′* ja kuvassa 2 sen alkiojoukkolattiisi. Lattiisi (eli hila) on osittaisjärjestetty joukko, jolla yksi yhteinen alkusolmu ja yksi yhteinen päätesolmu. Kuvan 2 alkusolmuna on tyhjä joukko ja päätesolmuna *{A,B,C,D,E}* (ks. myös määritelmä 5 luvusta 3.2.1). Kuvassa 2 vahvennetut solmut ovat kattavat joukot minimituella 2 (40%) [PBT99b].

TID	Items			
1	A	C	D	
2	B	C	E	
3	A	B	C	E
4	B	E		
5	A	B	C	E

Kuva 1: Esimerkkitietokanta D' [PBT99b]



Kuva 2: D':n alkiojoukkolattiisi [PBT99b, s.29]

2.2 Algoritmin ongelmat

Apriori-algoritmin suurin ongelma on se, että vaadittava tietokannan läpikäyntien lukumäärä on suoraan verrannollinen suurimman kattavan alkiojoukon kokoon. Lisäksi maksimaalisen kattavan joukon koon kasvaessa kandidaattijoukkojen lukumäärä kasvaa exponentiaalisesti.

Algoritmi perustuu apriori-periaatteeseen, jonka mukaan kaikki kattavien joukkojen osajoukot ovat myös kattavia ja kaikki ei-kattavien joukkojen ylijoukot ovat ei-kattavia. Kaikki tähän periaatteeseen perustuvat algoritmit toimivat tehokkaasti tietokantoihin, joissa tieto ei ole korreloivaa, kuten ostoskoritietokantoihin. Tietokannan datan ollessa vahvasti korreloivaa, algoritmin tehokkuus kärsii selvästi. Tämä johtuu siitä, että vahvasti korreloivasta aineistosta löytyy vain ”itsestäänselvyyksiä” korkeilla tukiarvoilla. Jos minimitukea lasketaan, kattavan alkiojoukkolattiisin koko kasvaa hyvin nopeasti liian suureksi [PBT99b].

Luvussa 3.3 esiteltävä suljettujen kattavien joukkojen louhintaan perustuva Close-algoritmi vähentää kyseisiä ongelmia, sillä sen kandidaattijoukkojen lukumäärä kasvaa huomattavasti aprori-algoritmin tuottamien kandidaattijoukkojen lukumäärää hitaammin. Algoritmi toimii suhteellisen hyvin hajanaisissa tietokannoissa ja erityisen tehokkaasti korreloivaa dataa sisältävissä tietokannoissa [PBT99b] .

3 Assosiaatiosääntöjen louhinta suljettujen kattavien alkiojoukkojen avulla

Tässä luvussa esittelen Pasquier'n ja kumppanien kehittämän tehokkaan kattavien joukkojen louhintamenetelmän, joka perustuu suljetun kattavan alkiojoukkolattiisin karsintaan [PBT99a ja PBT99b]. Suljettu alkiojoukkolattiisi on apriori-algoritmissa käytetyn alkiojoukkolattiisin alijärjestys ja siksi usein tätä paljon pienempi, mikä tekee kehitetystä close-algoritmista apriori-algoritmia tehokkaamman. Seuraavaksi selvitän Galois-yhteyden periaatteen ja sen sovelluksen kattavien joukkojen louhinnassa.

3.1 Galois-yhteys

Galois-yhteys kuvaa kahden osittaisjärjestetyn joukon välistä yhteyttä. Nämä Galois-yhteydet yleistävät osaryhmien ja osakenttien välisen vastaavuuden, jota Galois-teoriassa tutkitaan [EKM91]. Alla määrittelen Galois-yhteyden osittaisjärjestettyjen joukkojen järjestyksen säilyttävän, määritelmän mukaan.

Määritelmä 1 (Galois-yhteys) Olkoon $(A, \leq)$ ja $(B, \leq)$ kaksi osittaisjärjestettyä joukkoa sekä $F: A \rightarrow B$ ja $G: B \rightarrow A$ kaksi monotonista (järjestyksen säilyttävää)

funktiota. Nämä funktiot muodostavat Galois-yhteyden, jos kaikille $a \in A$ ja $b \in B$ pätee

$$F(a) \leq b \; joss \; a \leq G(b)$$

[DaP02, EKM91].

Tässä määritelmässä funktiota F kutsutaan G :n alemmaksi adjungaatiksi ja G :tä F :n ylemmäksi adjungaatiksi. Nämä toistensa vastapareina olevat adjungaatit voidaan merkitä vastaavasti f_* ja f^* merkinnöillä.

Määritelmän funktioita F ja G voidaan katsoa saman Galois-yhteyden eri puolina. Kun jokin transaktio muokataan ensimmäisen funktion avulla uuteen avaruuteen ja sen jälkeen funktion vastaparilla takaisin omaan avaruuteensa, saavutetaan stabiili tila, jonka jälkeen kaikki seuraavat muunnokset tuottavat saman tuloksen [EKM91].

Lisätietoa Galois-yhteyksistä ja niiden sovelluksista löytyy lähteistä [EKM91] ja [DaP02]. Näistä lähteistä löytyy myös Galois-yhteyden alkuperäinen määritelmä, joka määrittelee yhteyden kahden antitonisen (järjestyksen kääntävän) funktion muodostamana parina.

3.2 Suljetut kattavat joukot

Pasquier ja kumppanit sovelsivat Galois-yhteyttä kattavien joukkojen tehokkaassa etsinnässä. He ovat kuvanneet menetelmänsä kahdessa julkaisemassaan raportissa [PBT99a] ja [PBT99b]. Seuraavaksi esittelen Pasquire'n ja kumppaneiden määritelmät tiedonlouhintakontekstille, Galois-yhteydelle tiedonlouhinnan kontekstissa, suljetulle joukolle ja suljetun joukon lattiisille. Näillä määritelmillä he liittävät Galois-yhteyden kattavien alkiojoukkojen louhintaan.

3.2.1 Määritelmät

Määritelmä 2 (Tiedonlouhintakonteksti) Tiedonlouhintakonteksti on tripletti $D' = (O', I', R')$, missä O' on transaktioiden äärellinen joukko, I' on alkioiden äärellinen joukko ja R' on näitä yhdistävä relaatio. Jokaisen parin *(o,i)*, joka kuuluu binäärirelaatioon R', kuva kertoo, onko *o* relaatiossa *i*:hin vai ei. Tiedonlouhintakonteks-

ti voi olla relaatio, luokka tai SQL kyselyn tulos. Laajennamme määritelmän sisältämään tiedonlouhintakontekstiksi myös tietokannan [PBT99b].

Määritelmä 3 (Galois-yhteys tiedonlouhintakontekstissa) Olkoon $D'=(O', I', R')$ määritelmän 2 mukainen tripletti ja $O\subseteq O'$ sekä $I\subseteq I'$, joille määrittelemme:

$f(O): 2^{O'}\rightarrow 2^{I'}$ ja $f(O)=\{i\in I': \forall o\in O, (o,i)\in R'\}$

sekä

$g(I): 2^{I'}\rightarrow 2^{O'}$ ja $g(I)=\{o\in O': \forall i\in I, (o,i)\in R'\}$

$f(O)$ siis assosioi O :hon kaikki ne alkiot, jotka ovat yhteisiä kaikille monikoille $o\in O$ ja $g(I)$ assosioi I :hin kaikki ne monikot, joihin sisältyy kaikki alkiot $i\in I$. Nämä kaksi funktiota muodostavat Galois-yhteyden O' potenssijoukon (ts. $2^{O'}$) ja I' potenssijoukon (ts.$2^{I'}$) välille.

Määrittelemme myös Galois'n sulkeumaoperaatiot $h()=f(g())$ joukossa $2^{I'}$ ja $h'()=g(f())$ joukossa $2^{O'}$. Galois-yhteyden (f,g) ollessa voimassa seuraavat ominaisuudet pätevät kaikille $I, I_1, I_2\subseteq I'$ ja $O, O_1, O_2\subseteq O'$ [PBT99b]:

(1) $I_1\subseteq I_2\Rightarrow g(I_1)\supseteq g(I_2)$	(1') $O_1\subseteq O_2\Rightarrow f(O_1)\supseteq f(O_2)$
(2) $I\subseteq h(I)$	(2') $O\subseteq h'(O)$
(3) $h(h(I))=h(I)$	(3') $h'(h'(O))=h'(O)$
(4) $I_1\subseteq I_2\Rightarrow h(I_1)\subseteq h(I_2)$	(4') $O_1\subseteq O_2\Rightarrow h'(O_1)\subseteq h'(O_2)$
(5) $h'(g(I))=g(I)$	(5') $h(f(O))=f(O)$

(6) $O\subseteq g(I)\Leftrightarrow I\subseteq f(O)$

Määritelmä 4 (Suljetut alkiojoukot) Olkoon $C\subseteq I'$ alkiojoukko tripletistä *D'*. *C* on suljettu alkiojoukko $joss\, h(C)=C$. Pienin (minimaalinen) alkiojoukon *I* sisältävä suljettu alkiojoukko saadaan funktiolla $h(I)$ [PBT99b].

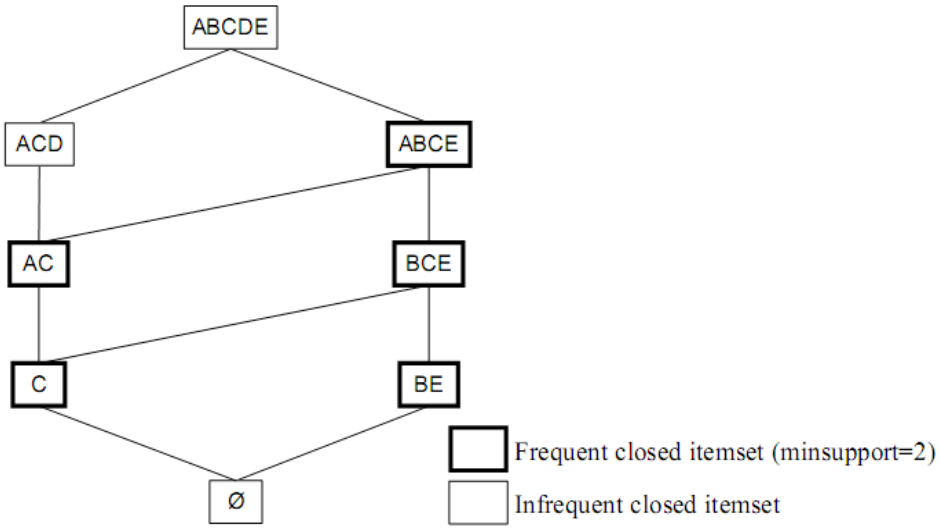
Määritelmä 5 (Suljettu alkiojoukkolattiisi) Olkoon *C* joukko suljettuja alkiojoukkoja, jotka on johdettu *D'*:stä Galois-yhteyden avulla. Pari $L_c=(C,\leq)$ on täydellinen lattiisi, jota kutsutaan alkiojoukkolattiisiksi. Lattiisirakenne sisältää kaksi ominaisuutta

i) Lattiisin elementeillä on osittaisjärjestys siten, että jokaiselle alkiolle $C_1, C_2\in L_c, C_1\leq C_2\, joss\, C_1\subseteq C_2$

ii) Kaikilla L_c:n osajoukoilla S on yksi suurin alaraja *Join*-elementti ja yksi pienin yläraja *Meet*-elementti. Nämä ovat määritelty seuraavasti: Kaikille $S \subseteq L_c$ pätee

$$Join\ (S) = h(\bigcup_{C \in S} C), \qquad Meet\ (S) = \bigcap_{C \in S} C$$

Kuvassa 3 on esimerkki D':n suljetusta alkiojoukkolattiisista. Tässä lattiisissa *Meet* on tyhjä joukko ja *Join on {A,B,C,D}* [PBT99b].



Kuva 3: D':n suljettu alkiojoukkolattiisi [PBT99b]

Toisin sanoen suljettu alkiojoukko on maksimaalinen joukko alkioita, jotka ovat jollekin transaktiojoukolle yhteisiä. Esimerkiksi kuvan 1 tietokannassa *D'* alkiojoukko *{B,C,E}* on suljettu alkiojoukko, sillä se on maksimaalinen joukko alkioita, joka on yhteinen transaktioille *{2,3,5}*. Tämä alkiojoukko on myös kattava suljettu alkiojoukko minimitukiarvolla 2, sillä $tuki(\{B, C, E\}) = \|\{2,3,5\}\| = 3 \geq 2$. Jos *D'* on ostoskoritietokanta, tarkoittaa tämä sitä, että 60% asiakkaista (3/5) ostaa enintään tavarat *{B,C,E}*. Alkiojoukko *{B,C}* ei ole suljettu alkiojoukko, sillä se ei ole maksimaalinen joukko jollekin transaktiojoukolle yhteisiä alkioita – kaikki jotka ostavat tavarat *B* ja *C* ostavat myös tavaran *E* [PBT99b].

3.2.2 Suljettujen kattavien joukkojen ominaisuudet

Suljetut kattavat alkiojoukot sopivat hyvin assosiaatiosääntöjen louhintaan, sillä suljettuja kattavia alkiojoukkoja on yleensä paljon vähemmän kuin kattavia alkiojoukkoja. Näin hakuavaruuden läpikäytävää osaa rajaamalla vähennetään sekä tietokannan läpikäyntejä että kattavien joukkojen generoinnista syntyvää prosessorikuormaa. Tämä prosessorikuorma on merkittävä, sillä kattavien joukkojen lattiisin koko $\|L_s\|$ on exponentiaalinen tietokannan alkioiden lukumäärän $2^{\|I'\|}$ suhteen. Pahimmassa tapauksessa myös suljettujen kattavien joukkojen lattiisin koko on exponentiaalinen. Kuitenkin käytännön kokeissa ja tasaisen jakauman oletuksella suljetun alkiojoukkolattiisin $\|L_c\|$ koko on keskimäärin pienempi tai yhtäsuuri kuin tietokannan koko kerrottuna transaktioiden keskiarvokoolla: $\|L_c\| \leq \mu \|D'\|$, missä μ on tietokannan transaktioiden koon keskiarvo. Tämä kokoero on helposti huomattavissa vertailtaessa D':n alkiojoukkolattiisia (kuva 2) ja suljettua alkiojoukkolattiisia (kuva 3) [PBT99b].

Edellisessä luvussa esitellyistä määritelmistä saamme johdettua seuraavat alkiojoukkojen ominaisuudet.

i) Kattavan alkiojoukon kaikki osajoukot ovat kattavia.

ii) Kaikki ei-kattavan joukon laajennokset ovat ei-kattavia.

iii) Kaikki kattavan suljetun alkiojoukon suljetut osajoukot ovat kattavia.

iv) Kaikki ei-kattavan joukon suljetut joukkolaajennokset ovat ei-kattavia.

v) Maksimaalisten kattavien joukkojen joukko on identtinen maksimaalisten suljettujen kattavien joukkojen joukon kanssa.

vi) Kattavan ei-suljetun joukon *I* tuki on sama kuin tuki pienimmälle kattavalle suljetulle alkiojoukolle, joka sisältää *I* :n (täten kattavan joukon sulkeuma on myös kattava).

Näiden ominaisuuksien todistukset löytyvät lähteestä [PBT99b].

3.3 Assosiaatiosääntöjen louhinta Close-algoritmilla

Close algoritmi louhii assosiaatiosäännöt aiemmin tässä luvussa esitettyjen määritelmien ja ominaisuuksien perusteella kolmessa vaiheessa:

1. Etsi kaikki tietokanta D':n kattavat suljetut alkiojoukot (eli suljetut alkiojoukot, joiden tuki on suurempi tai yhtäsuuri kuin *minsupport*).

2. Muodosta kohdassa 1 saaduista suljetuista alkiojoukoista kaikki kattavat alkiojoukot. Tässä vaiheessa generoidaan maksimaalisten suljettujen alkiojoukkojen kaikki osajoukot ja lasketaan niiden tuki suljetuista alkiojoukoista.

3. Generoi kaikki ne assosiaatiosäännöt vaiheessa 2 saaduista kattavista alkiojoukoista, joiden luottamus on suurempi tai yhtäsuuri kuin *minconfidence*

Vaihe 1 on algoritmin laskennallisesti vaativin osuus. Tietokantahakuja tarvitsee tehdä vain kohdassa yksi. Vaiheet 2 ja 3 saadaan suoraviivaisesti ratkaistua keskusmuistissa. Seuraavaksi kuvaan algoritmin ensimmäisen vaiheen toiminnan. Vaiheiden 2 ja 3 tarkat kuvaukset löytyvät lähteestä [PBT99b].

3.3.1 Close-algoritmi

Kuvassa 4 on pseudokoodinen algoritmi suljettujen kattavien alkiojoukkojen louhintaan. Algoritmi käyttää hyväkseen funktioita Gen-Closure (kuva 5) ja Gen-Generator (kuva 6). Alla oleva taulukko selvittää algoritmin muuttujat ja niiden sisällöt.

Joukko	Kenttä	Sisältö
FCC_i	*generator*	generoija, jonka koko on i
	closure	generoijan sulkeumasta muodostettu mahdollinen suljettu alkiojoukko
	support	suljetun alkiojoukon tuki *support = count(closure) = count(generator)*
FC_i	*generator*	suljetun kattavan alkiojoukon generoija
	closure	suljettu kattava alkiojoukko
	support	suljetun kattavan alkiojoukon tuki

Algoritmissa käytetty suljetun alkiojoukon *c* generoija *p* on yksi pienimmistä (voi olla usea) alkiojoukoista, joista saadaan Galois'n sulkeumaoperaatiolla generoitua *c*: *h(p)* = *c* [PBT99b].

```
1)  generators in FCC_1 ← {1-itemsets};
2)  for (i ← 1; FCC_i.generator≠ ∅; i++) do begin
3)       closures in FCC_i ← ∅;
4)       supports in FCC_i ← 0;
5)       FCC_i ← Gen-Closure(FCC_i)†;          // Produces generator closures
6)       forall candidate closed itemsets c ∈ FCC_i do begin
7)            if (c.support ≥ minsupport) then          // If c is frequent
8)                 FC_i ← FC_i ∪ {c};                   // Append c to FC_i
9)       end
10)      FCC_{i+1} ← Gen-Generator(FC_i);   // Creates generators of iteration i+1
11) end
12) Answer FC ← ⋃_{j=1}^{j=i-1} {FC_j.closure, FC_j.support};
```

Kuva 4: Close-algoritmi [PBT99b]

Algoritmi alkaa alustamalla FCC_i :n *generators* -muuttujan tietokannan alkioilla (eli joukolla I') (rivi 1). Jokainen algoritmin iteraatio koostuu kolmesta vaiheesta. Ensin jokaiselle FCC_i :n generoijalle suoritetaan sulkeumafunktio Gen-Closure (ks. luku 3.3.2), josta saadaan kandidaattisulkeumajoukot ja niiden tuki-arvot (rivi 5). Tämän jälkeen saadusta kandidaattisulkeumajoukosta liitetään suljettuun kattavaan alkiojoukkoon ne joukot, joiden tuki on suurempi tai yhtäsuuri kuin *minsupport*. Lopuksi alustetaan uudet generaattorit FCC_{i+1} :lle Gen-Generator -funktiolla (ks. luku 3.3.3) seuraavaa iteraatiota varten [PBT99b].

3.3.2 Gen-Closure -funktio

Kuvan 5 Gen-Closure funktio pohjaa seuraavaan propositioon [PBT99b, sivu 34].

Propositio 1 Suljettu alkiojoukko $h(I)$ on leikkaus kaikista tietokannan transaktioista jotka sisältävät *I* :n.

Generoijien sulkeumien ja niiden tukien laskeminen propositio 1:n avulla vaatii vain yhden tietokannan läpikäynnin.

```
1)  forall objects o ∈ O do begin
2)       G_o ← Subset(FCC_i.generator, f({o})); //Gen. that are subsets of f({o})
3)       forall generators p ∈ G_o do begin
4)            if (p.closure = ∅) then
5)                 p.closure ← f({o});
6)            else p.closure ← p.closure ∩ f({o});
7)            p.support++;
8)       end
9)  end
10) Answer ← ⋃{c ∈ FCC_i | c.closure ≠ ∅};
```

Kuva 5: Funktio Gen-Closure [PBT99b]

Funktio toimii seuraavasti: Jokaiselle transaktiolle *o* tietokannassa *D'* luodaan joukko G_o (rivi 2). G_o sisältää kaikki FCC_i :n generoijat jotka ovat kyseisen transaktio *o*:n alkiojoukon *f({o})* osajoukkoja. Tämän jälkeen jokaiseen G_o :n generoijaan *p* liittyvä suljettu alkiojoukko ja sen tuki päivitetään (rivit 3–7). Jos *p* :n sulkeuma on tyhjä, *p*:n sulkeuma saa arvokseen koko o:n alkiojoukon *f({o})*. Muuten p:n sulkeuma leikataan o:n alkiojoukon *f({o})* kanssa.

Kun kaikki tietokannan transaktiot käydään näin läpi, on jokaiselle generaattorille saatu sen sulkeuma tukiarvoineen vain yhdellä transaktioiden läpikäynnillä. Mahdollisia kattavia alkiojoukkoja pidetään tasapainotetussa alkuosapuurakenteessa. Näin funktio Subset saa tehokkaasti haettua kaikki transaktion *o* alijoukkoina olevat generoijat ja niiden sulkeumat [PBT99b].

Esimerkkinä funktion toiminnasta voidaan katsoa kuinka generoijan *{A}* sulkeuma *{A,C}* muodostuu esimerkkitietokannassamme *D'*. Aluksi funktio käsittelee transaktiota 1. Generoija *{A}* on transaktion 1:n osajoukko, joten *p* sisältää *{A}*:n. G_o:n läpikäynnissä (rivit 3–7) *{A}*:n sulkeumaksi merkitään *{A,C,D}* sillä *{A}*:n sulkeuma on vielä tyhjä. Transaktio 2 ei sisällä *{A}*:ta, joten generoija *p*:tä ei käsitellä lainkaan. Transaktio 3 taas sisältää *{A}*:n joten *{A}*:n sulkeuma *{A,C,D}* leikataan joukon *{A,B,C,E}* kanssa ja päivitetty sulkeuma *{A,C}* tallennetaan tuella 2. Tietokannan neljäs transaktio ei sisällä *{A}*:ta joten sitä ei käsitellä. Viidennellä iteraatiolla transaktion alkiojoukko on sama kuin kolmannella, joten sulkeuma *{A,C}* ei muutu, mutta sulkeuman tuki kasvatetaan kolmeen.

3.3.3 Gen-Generators -funktio

Gen-Generators funktiolla luodaan uudet i+1 kokoiset generoijat Close-algoritmin seuraavaa iteraatiota varten. Tämän jälkeen algoritmi karsii potentiaaliset generoijat, jotka johtaisivat turhaan laskentaan (ei-kattavat suljetut alkiojoukot) tai redundanssiin (kattavat suljetut alkiojoukot, jotka on jo aiemmin generoitu).

Lemmat ja korollaarit, joihin algoritmi perustuu on esitelty ja todistettu lähteessä [PBT99b, sivut 35–36].

```
1)  insert into FC_{i+1}.generator
2)  select p.item_1, p.item_2, ..., p.item_i, q.item_i
3)  from FC_i.generator p, FC_i.generator q
4)  where p.item_1 = q.item_1, ..., p.item_{i-1} = q.item_{i-1}, p.item_i < q.item_i;
5)  forall generators p ∈ FC_{i+1}.generator do begin
6)       forall i-subsets s of p do begin
7)            if (s ∉ FC_i.generator) then
8)                 delete p from FC_{i+1}.generator;
9)       end
10) end
11) forall generators p ∈ FC_{i+1}.generator do begin
12)      S_p ← Subset(FC_i.generator,p); // Generators that are subsets of p
13)      forall s ∈ S_p do begin
14)           if (p ⊆ s.closure) then
15)                delete p from FC_{i+1}.generator;
16)      end
17) end
18) Answer ← ⋃{c ∈ FC_{i+1}};
```

Kuva 6: Funktio Gen-Generator [PBT99b]

Algoritmi käyttää alussa samaa kombinatorista tekniikkaa kuin apriori-algoritmin generoija uusien mahdollisten generoijien luontiin (rivit 1–4). Tämän jälkeen (apriori-generoijan tavoin) algoritmi tarkistaa kandidaattigeneroijan osajoukkojen kuulumisen FC_i:n generoijiin. Tämä karsii apriori-periaatteen mukaan kaikki generoijat, joiden osajoukko on ei-kattava (rivit 5–10). Toinen osa karsintaa (rivit 11–17) karsii kaikki ne generoijat, jotka kuuluvat jo jonkin osajoukkonsa sulkeumaan. Tämä karsii ne redundantit generoijat, jotka johtaisivat jo löydettyyn suljettuun alkiojoukkoon.

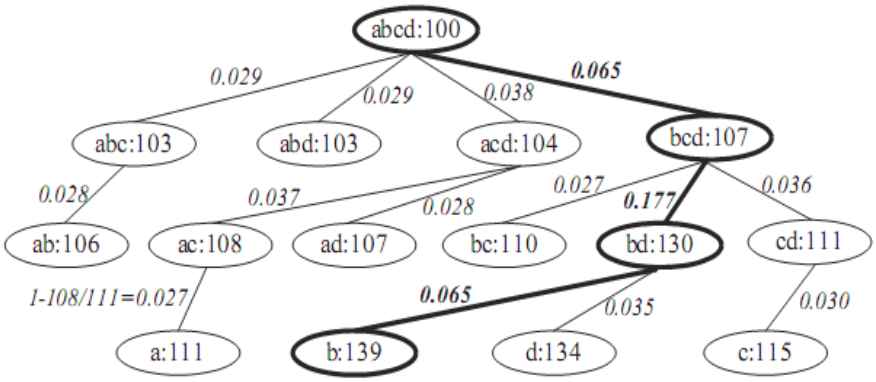
Kun Close-algoritmi on näin luonut kattavat suljetut joukot, saadaan nämä muutettua helposti kattaviksi joukoiksi, joista assosiaatiot voidaan laskea apriori-algoritmin mukaisesti. Nämä vaiheet on tarkemmin selitetty lähteessä [PBT99b].

4 Louhinnan tehostaminen δ-toleranssin avulla

Assosiaatiosääntöjen louhinta suljettujen alkiojoukkojen avulla tehostaa louhimista huomattavasti. Close -algoritmin, kuten apriorin, ongelma on kuitenkin suuri louhinnan tuloksena saatu kattavien alkiojoukkojen ja assosiaatiosääntöjen määrä.

Chen, Ke ja Ng ovat kehittäneet kattavan suljetun alkiojoukon määritelmän lievennyksen, jonka avulla lähes samanlaiset kattavat joukot voidaan karsia samalla vähentäen saatujen samankaltaisten assosiaatiosääntöjen määrää. Tämän menetelmän nimi on δ-TCFI (δ-Tolerance Closed Frequent Itemsets) eli δ-toleranssin suljetut kattavat alkiojoukot. He käsittelevät artikkelissaan [CKN06] näiden kattavien alkiojoukkojen louhintaa sekä artikkelissa [CKN08] näistä kattavista joukoista luotuja δ-TAR -assosiaatiosääntöjä (δ-Tolerance Association Rules).

δ-toleranssi-periaate perustuu siihen, että Close-algoritmilla louhitusta suljetusta kattavasta alkiojoukkolattiisista voidaan karsia hyvin samanlaiset redundantit kattavat joukot. Suljetun alkiojoukkolattiisin vaatimus on, että kattavan alkiojoukon sulkeuman kaikki osajoukot ovat myös kattavia. δ-toleranssi on tämän vaatimuksen lievennys.



Kuva 7: δ-toleranssi suljettu kattava alkiojoukkolattiisi

Kuvassa 7 on esimerkki suljetusta kattavasta alkiojoukkolattiisista. Tässä kuvassa lattiisin jokaiseen solmuun on merkitty solmun alkiot ja niiden tuki. Jokaisen kaaren suhdeluku on laskettu seuraavalla kaavalla $\delta = \left(1 - \frac{Y:ntuki}{X:ntuki}\right)$, missä Y on X:n pienin ylijoukko, jolla on suurin tuki. Kuvassa on näytetty esimerkkinä δ-arvon lasku joukolle {a}. Jos kyseessä olisi suljettu kattava alkiojoukko, olisi δ-ehtona 0. Jos lievennämme

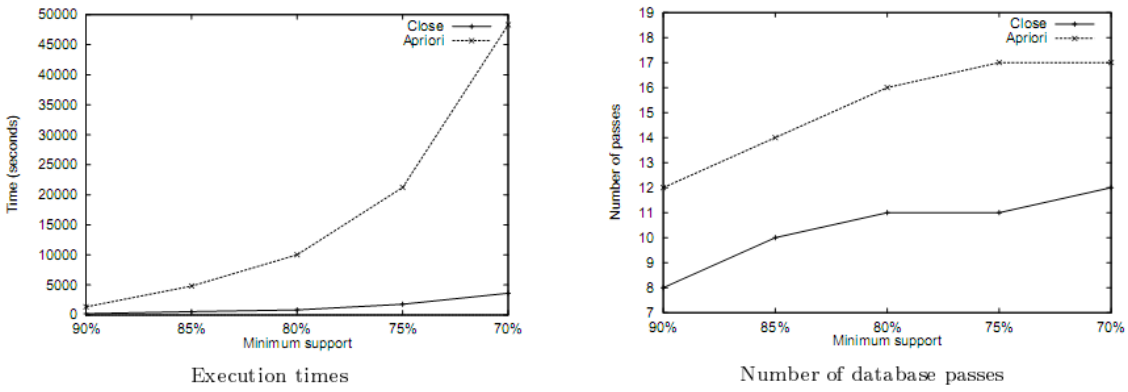
tätä ehtoa hieman asettamalla esimerkiksi $\delta \leq 0{,}04$, saamme karsittua 11 kattavaa alkiojoukkoa tästä 15 alkiojoukon suljetusta lattiisista [CKN06].

Cheng ja kumppanit osoittivat, että redundanttien kattavien joukkojen karsiminen δ-toleranssilla pienentää saadut kattavat joukot murto-osaan suljettujen kattavien joukkojen lukumäärästä. He osoittivat myös, että virheiden osuus saaduissa kattavissa joukoissa on huomattavasti δ-ehtoa pienempi ja että se virheiden osuus kasvaa paljon δ-toleranssin kasvua hitaammin [CKN06].

Cheng ja kumppanit kehittivät kaksi eri algoritmia δ-TCFI:n laskemiseen. Ensimmäinen nimeltään CFI2TCFI nimensä mukaisesti generoi ensin suljetun kattavan alkiojoukkolattiisin, josta se karsii ylimääräiset alkiojoukot yllä olevan periaatteen mukaisesti. Toinen algoritmi MineTCFI suorittaa δ-toleranssi-karsinnan kandidaattijoukkojen haun yhteydessä. Tämä algoritmi on huomattavasti CFI2TCFI:tä tehokkaampi, sillä siihen ei vaikuta sulkeuman kasvava koko, kuten CFI2TCFI:ssä.

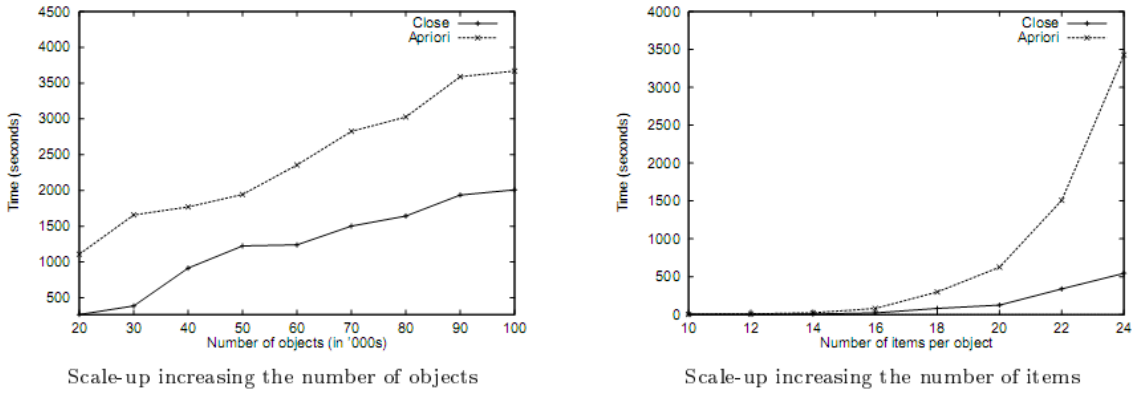
5 Louhinta-algoritmien tulosten vertailua

Tässä luvussa esittelen yllä kuvattujen algoritmien käytännön suorituskykyä [PBT99b, CKN06]. Kuvat 8 ja 9 kuvaavat Close-algoritmin suorituskykyä verrattuna Apriori-algoritmiin [PBT99b].



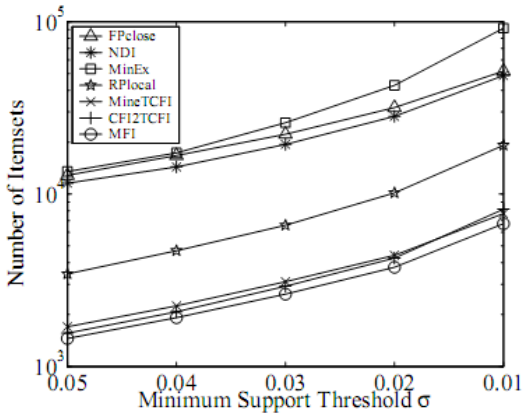
Kuva 8: Minimituen vaikutus Apriori ja Close algoritmien suorituskykyyn [PBT99b]

Kuvasta 8 näkee selvästi kuinka paljon kattavien alkiojoukkojen lukumäärän kasvattaminen (minimituen laskeminen) vaikuttaa Apriori-algoritmin tehokkuuteen. Sulkeumia hyödyntävä Close-algoritmi on paljon tehokkaampi, varsinkin suoritusajassa mitattuna.

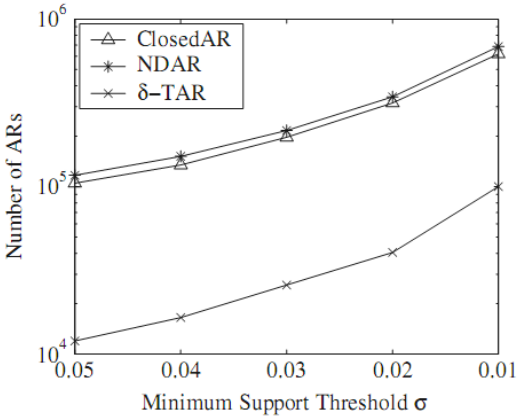


Kuva 9: Tietokannan koon ja transaktioiden koon muutoksen vaikutus [PBT99b]

Kuva 9 esittelee tietokannan koon kasvamisen ja etenkin tietokannan transaktioiden koon kasvamisen vaikutuksen algoritmien tehokkuuteen. Vaikka Close-algoritmikin hidastuu eksponentiaalisesti transaktiokoon kasvaessa, on hidastuminen paljon aprioria hitaampaa.



Kuva 10: Minimituen vaikutus alkiojoukkojen lukumäärään [CKN06]



Kuva 11: Minimituen vaikutus louhittujen assosiaatiosääntöjen lukumäärään [CKN06]

Kuvissa 10 ja 11 vertaillaan δ-toleranssi-menetelmän toimivuutta muihin louhinta-algoritmeihin nähden. Kuvissa FPClose ja ClosedAR ovat aiemmin kuvatun Close-algoritimin tehokkaampi toteutus. Kuten kuvasta näkyy, CFI2TCFI ja MineTCFI pystyvät selvästi vähentämään louhittujen kattavien joukkojen ja näistä generoitujen redundanttien assosiaatiosääntöjen määrää etenkin suhteellisen pienillä tuki-arvoilla [CKN06].

6 Yhteenveto

Tässä tutkielmassa kuvasin kaksi parannusmenetelmää perinteiselle assosiaatiosääntöjen louhinnalle apriori-algoritmin (tai sen johdannaisten) avulla. Assosiaatiosääntöjen louhinta suljettujen kattavien joukkojen avulla [PBT99a, PBT99b] ja siitä kehitetty δ-toleranssi-periaate [CKN06, CKN08] molemmat tehostavat louhintaa omilla osa-alueillaan.

Suljetut kattavat joukot pystyvät tehostamaan itse louhintaprosessia vähentämällä louhinnan vaatimaa prosessoriaikaa sekä tietokantahakujen määrää. Louhinnan tulos on identtinen apriori-algoritmin kanssa, sillä saadut suljetut kattavat joukot ovat kattavien joukkojen häviötön kuvaus [PBT99b].

δ-toleranssi-periaatteen avulla louhituista kattavista joukoista saadaan karsittua redundantit kattavat joukot sekä näistä generoidut hyvin samanlaiset assosiaatiosäännöt. Menetelmä on myös siitä tehokas, että karsinnan aiheuttamien virheiden osuus on pieni ja kasvaa hitaasti. Menetelmä helpottaa merkityksellisten assosiaatiosääntöjen löytämistä aineistosta, sillä enää ei louhinnan tuloksena ole vain lukematon määrä lähes samankaltaisia ”itsestäänselvyyksiä” [CKN06].

Lähteet

AIS93 Agrawal, R., Imielinski, T. ja Swami,A., Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD Int'l Conference on Management of Data*, May 1993, sivut 207-216. [Myös http://rakesh.agrawal-family.com/papers/sigmod93assoc.pdf, 10.3.2008]

AgS94 Agrawal, R., Srikant, R., Fast algorithms for mining association rules, *Proceedings of the 20th int'l conference on very large data bases*, 1994, sivut 478-499. [Myös http://rakesh.agrawal-family.com/papers/vldb94apriori.pdf, 17.3.2008]

CKN06 Cheng, J. , Ke, Y. ja Ng, W. δ-Tolerance Closed Frequent Itemsets, *Proceedings of the Sixth International Conference on Data Mining*, December 18–22, 2006, sivut 139–148. [Myös http://www.cs.ust.hk/~csjames/icdm06.pdf, 10.3.2008]

CKN08 Cheng, J. , Ke, Y. ja Ng, W., Effective elimination of redundant association rules. *Data Mining and Knowledge Discovery*, Volume 16, Number 2 / April, 2008, sivut 221–248. [Myös http://www.springerlink.com/content/t507j24q64h7m273/, 10.3.2008]

DaP02 Davey, B. A. ja Priestley, H. A., *Introduction to lattices and order*, Cambridge University Press, New York, 2002

EKM91 Erne, M., Koslowski, J., Melton, A. ja Strecker, G., A primer on Galois connections. Teoksessa *Papers on general topology and applications, ed. A. R. Todd,* Madison, WI, USA,1991.

PBT99a Pasquier, N., Bastide, Y., Taouil, R. ja Lakhal, L., Discovering frequent closed itemsets for association rules. *Database Theory — ICDT'99,* Springer Berlin, Heidelberg, 1999, sivut 398–416. [Myös http://www.springerlink.com/content/h15rjbc9mukltywf/, 10.3.2008]

PBT99b Pasquier, N., Bastide, Y., Taouil, R. ja Lakhal, L., Efficient mining of associations rules using closed itemset lattices. Information systems Vol. 24, No. 1, 1999, sivut 25–46., [Myös http://www.i3s.unice.fr/~pasquier/aigaion2/index.php/publications/show/46, 12.3.2008]