Assotsiatsioonireeglite leidmine suurtest andmehulkadest

Asko Tiidumaa
Asko.Tiidumaa@ut.ee

Andmekaevandamise uurimisseminar MTAT.03.169.
Arvutiteaduse instituut, Tartu Ülikool
Detsember 2003, lk. 1–15

Kokkuvõte

Assotsiatsioonireeglite analüüsi eesmärgiks on omavahel püsival ja mõistlikul viisil seotud nähtuste, sündmuste, objektide vms leidmine. Sellised seotud nähtused võivad olla näiteks sageli koos külastatavad veebilehed või tähelepanek, et mingit kindlat kaupa ostvad kliendid võtavad küllalt suure tõenäosusega ka ühe teise kindla toote. Käesoleva artikli ülesandeks on anda ülevaade assotsiatsioonireeglite leidmisega seotud problemaatikast, lahendustest ning võimalikest rakendustest. Assotsiatsioonireeglite leidmiseks vaadeldakse siin algoritmi Apriori koos kiire implementatsiooni ideega.

2 Sissejuhatus

Infotehnoloogia efektiivne ning suuremahuline rakendamine on endaga toonud kaasa kogutava infomahu plahvatusliku kasvu. Samas oluline pole mitte niivõrd kogunev andmemaht ise kuivõrd hoopis neid andmeid tekitavate protsesside mõistmine.

Vaatleme näiteks suure toidupoe müügiprotsessi. Siin on meil tegemist suure hulga erinevate kaubaartiklitega. Samuti on siin kliendid, kes kaubavalikust ühtteist enda ostukorvi panevad, ning seejärel kassas ostu sooritavad. See fakt kajastub toidupoe andmebaasis nn ostukorvina. Siit koguneb toidupoele väärtuslik infovaramu näiteks turundusosakonnale, kelle ülesandeks on korraldada erinevaid kampaaniaid.

Ostukorvide pealt on võimalik leida seoseid erinevate kaupade ühte ostukorvi sattumiste kohta. Üheks selliseks on sageli erialastes artiklites käsitletav tähelepanek, et sageli ostetakse koos õlut ning beebimähkmeid [TC00]. Sellisel juhul võib vaadelda implikatsiooni *beer*⇒*diapers* [1] kui assotsiatsioonireeglit üle õlut ning mähkmeid sisaldavate ostukorvide. Assotsiatsioonireeglite analüüsi ülesandeks on

[1] *beer* tähistab selles reeglis õlut ning *diapers* beebimähkmeid

leida analoogilisi seoseid andmestust — kui ostukorvis sisalduvad teatud kaubad, millised kaubad veel samas ostukorvis tõenäoliselt esinevad?

Vastavaid reegleid võib tulla suhteliselt palju, üldjoontes võib need jagada järgmisse nelja gruppi [ibm98]:

- **Juhuslikud seosed**
  Õlu ja mähkmed võisid täiesti juhuslikult olla samal päeval poole hinnaga müügis ja seetõttu neil omavahel mingit põhjuslikku seost ei ole.

- **Juba teada olevad seosed**
  Õlle ja küüslauguleivakeste vaheline seos võib varasemast olla juba tuntud.

- **Uued ent mitteolulised seosed**
  Punast ja sinist värvi pulgakommide vaheline seos võib olla küll uus kuid antud ülesande jaoks olematu praktilise tähtsusega.

- **Uued ja olulised seosed**
  Õlu ja mähkmete vaheline seos võib olla täiesti uus ja praktiliselt kasutatav, näiteks:

  – Õlu ja mähkmed võimalikult lähestikku paigutada — et klient kindlasti ühte korvi pannes ka teise kauba korvi tõstaks
  – Õlu ja mähkmed võimalikult kaugele teineteisest paigutada — et klient õlut korvi pannes peaks võimalikult paljude teiste kauvbariiulite vahelt mähkmeteni jalutama
  – Hetkel 15% soodsamat õlut korvi tõstes ei pane õnnelik klient ehk tähelegi, et samal päeval on mähkmed 15% kallimad...

Algoritmiline pool on vaid nende reeglite tuvastamine, reeglite kvaliteedi ning praktilisuse otsustamine jääb paraku ikkagi inimese otsustada. Suureks probleemiks on siin reeglite suur hulk, näiteks mõne tuhande kaubaartikliga poes võib olla miljardeid reegleid [BK02]. Kuna sellise hulga reeglite kontroll on väga töömahukas on vaja efektiivseid algoritme. Efektiivsus tähendab siin kontrollitavate reeglite arvu piiramist, jättes võimaluse korral siiski huvitavad reeglid alles.

Assotsiatsioonireeglite kaevandamise juured on jaekaubanduse ostude analüüsil [AIS93] ning seetõttu vastavate seoste leidmist andmestust nimetatakse sageli ostukorviaanalüüsiks(ik *market-basket analysis*). Samas ei ole see piiratud ainult ostukorvide analüüsiga. Palju huvitavamaid ning ärilisest seisukohast paremini rakendatavaid tulemusi võib saada ka ostukorvide andmestu uurimisel koos klientide demograafi liste andmetega. See võib uurija viia tulemuseni, et 25–30 aastane universaalkerega autoga abielus meesterahvas ostab suure tõenäosusega koos õllega ka lapsemähkmeid. Samuti ei pruugi ühe ostukorvi mõttes tegu korraga ostetud kaupadega vaid võib vaadelda ka ühe kliendi oste mingi kindla perioodi jooksul, näiteks veebiraamatupoe või plaadipoe ostud [AIS93].

Kuigi assotsiatsioonireegli all käesoleva ülevaate mõttes peame silmas lihtsalt hulkadevahelist seost, on loodud meetodeid ka järjestusega assotsiatsioonireeglite

leidmiseks [AS95, AFS93, SA96] . Sellist tüüpi assotsiatsioonireegleid esineb näiteks geenisekventsiandmebaaside [Chi96, YWYH02] ja veebikohtade külastuste analüüsis [BBR01, CSM97, MJHS96, MA] .

Üks huvitav valdkond on negatiivsete assotsiatsioonide leidmine. Näiteks üks selline reegel võiks kõlada järgnevalt: «75% abielus meestest kes ostavad õlut ei osta pakipiima». Sedasorti probleeme lahendavaid algoritme on juba ka loodud ning autori väiteil on nad küllalt efektiivsed [SON98].

Käesoleva artikli eesmärgiks on anda ülevaade assotsiatsioonireeglitega seotud problemaatikast, milleni jõudmiseks tuuakse kõigepealt ära probleemi esitus. Illustreerimaks lahendust eelpool kirjeldatud probleemile on toodud ära algoritm Apriori ning tema efektiivne implementatsioon. Lühidalt vaadeldakse ka assotsiatsioonireeglite analüüsi rakendusvaldkondi ning lahendamist ootavaid probleeme.

3 Formaalne esitus

Olgu $I = \{I_1, I_2, ..., I_m\}$ hulk binaarseid tunnuseid, ehk kaupu [AIS93]. Olgu T ostude andmebaas, kus igat ostu t kirjeldab kahendvektor, milles $t[k] = 1$ kui ostukorvis sisaldus kaup I_k, ning $t[k] = 0$ kui ei olnud. Iga ostu kohta on andmebaasis T üks korteeþ. Olgu $X \subseteq I$. Ütleme, et ost t rahuldab X kui iga ostu I_k jaoks hulgas X nii, et $t[k] = 1$.

Assotsiatsioonireegli all peame edaspidi silmas implikatsiooni kujul $X \Rightarrow I_j$, kus $X \subset I$ ning $I_j \in I \wedge I_j \notin X$. Reegel $X \Rightarrow I_j$ on täidetud baasil T usaldusteguriga (ik *confidence*) $0 \leq c \leq 1$ kui vähemalt $c\%$ ostudest baasis T, mis sisaldavad hulka X, sisaldavad ka I_j. Edaspidi tähistame vastavat olukorda kujul $X \Rightarrow I_j \mid c$, et reeglil $X \Rightarrow I_j$ on usaldustegur c. Ütleme, et reegli $X \Rightarrow I_j$ tugi (ik *support*) baasil T on s kui $s\%$ ostudest baasis T rahuldab $X \cup I_j$. Hulga a tuge võime väljendada ka kujul $support(a)$. Sageli vaadeldakse I_j asemel ka hulka $I^\star$ nii, et $I^\star \subseteq I$. Kui usaldustegur mõõdab reegli tugevust siis tuge võib vaadelda kui olulisust statistilises mõttes [AIS93, Fio02, BMS97].

Ülesanne on leida kõik assotsiatsioonireeglid, millel on tugi suurem kui eelnevalt seatud minimaalne tugi (edaspidi *minsup*) ning usaldustegur suurem kui eelnevalt seatud minimaalne usaldustegur (edaspidi *minconf*). On võimalik seada ka mitmeid täiendavaid kitsendusi soovitud assotsiatsioonireeglitele [AIS93]:

- **Süntaktilised kitsendused**
  Sedasorti kitsendused seavad piiranguid ostudele mis võivad reeglis esineda. Võib näiteks nõuda, et reegli paremal või vasakul poolel esineks kaup I_x või siis teda kindlasti seal ei oleks. Kuigi selliseid reegleid võib hiljem kõigist leitud reeglitest välja filtreerida võib nende kitsenduste integreerimine algoritmi tulemuse leidmist oluliselt kiirendada [SVA97].

- **Kitsendused toele**
  Sellised kitsendused seavad piiri vastavat reeglit sisaldavate ostude arvule.

Näiteks ärilises mõttes ei ole mõtet tegeleda seostega, mis kehtivad liiga väikesele osale juhtumitest.

Antud probleemi esituse korral võib assotsiatsioonireeglite leidmise protsessi jagada kaheks [AIS93, AS94]:

1. Leida kõik kaupade kombinatsioonid mille tugi on suurem kui *minsup*. Nimetame kõiki neid kombinatsioone suurteks kaupadehulkadeks (ik *large itemset*) ning kõiki ülejäänuid, mille tugi jääb allapoole *minsup* väärtust, väikesteks kaupadehulkadeks (ik *small itemset*). Kaupadehulki, mille vastavust kitsendustele pole veel jõutud kontrollida, nimetame edaspidi kandidaathulkadeks. Süntaktiliste kitsenduste olemasolul peame arvestama ka nendega.

2. Iga suure kaupadehulga $Y = I_1 I_2 ... I_k, k \geq 2$ jaoks koostada reeglid mis kasutavad kaupu valikust $I_1, I_2, ..., I_k$. Vastava assotsiatsioonireegli vasakuks pooleks saab alamhulk $X \subset Y$ nii, et hulgal X on $k-1$ elementi, ning reegli paremaks pooleks saab $Y - X$. Et leida reeglit $X \Rightarrow I_j \mid c$ nii, et $X = I_1 I_2 ... I_{j-1} I_{j+1} ... I_k$, tuleb võtta hulga X tugi ning jagada see I_j toega. Kui tulemus on suurem kui *minconf* siis võime vastava reegli alles jätta [AS94]. Tasub märkida, et kui kaupadehulk Y on suur, siis on suur ka iga Y alamhulk ning setõttu peab meil iga kaupadehulgaga olema teada ka tema tugi. Samuti peavad kõik kaupadehulgast Y tuletatud reeglid rahuldama toele seatud kitsendusi sest Y ise seda teeb, ning ta on samas kõikide nende reeglite paremate ja vasakute poolte ühend.

Kuna teise alamülesande lahendamine on suhteliselt elementaarne siis järgmises jaotises vaatleme lähemalt algoritmi Apriori, kus põhirõhk on esimese ülesande lahendamisel.

4 Algoritm Apriori

4.1 Suurte kaupadehulkade leidmine

Tavaliselt kasutavad suurte kaupadehulkade leidmise algoritmid andmestu T korduvat läbivaatust. Esimese läbivaatusega leitakse üksikute kaupade tugi ning otsustatakse, millised neist on toele seatud kitsenduste mõttes suured. Iga algoritmi samm kasutab eelmisel sammul leitud suuri kaupadehulki leidmaks võimalikke kandidaathulki. Seejärel leitakse andmestu T läbivaatusega toed neile kandidaathulkadele ning toele seatud kitsendust rahuldavad hulgad loetakse suurteks, ning seega järgmise sammu lähteandmeteks. Protsessi korratakse kuni rohkem suuri kaupadehulki ei leita.

Probleemiks on siin andmestu läbivaatamiseks kulutatav ressursside maht ning samuti kandidaathulkade suur arv. Üheks lahenduseks on siin vähemoluliste kandidaathulkade võimalikult varajane kõrvalejätmine ning andmestu läbivaatuste arvu minimeerimine.

Eelpoolkirjeldatud skeemi järgib küllalt laialdaselt levinud algoritmide pere Apriori [AS94]. Sellele algoritmide perele on iseloomulik uute kandidaathulkade koostamine vaid eelmiselt sammult saadud suuri kaupadehulki kasutades. Seda teha lubab teadmine, et iga suure kaupadehulga alamhulk peab samuti olema suur. Seetõttu võime koostada uusi k elemendilisi kandidaathulki $k-1$ elemendilistest suurtest kaupadehulkadest. Neist omakorda võib välja jätta need hulgad mille ükskõik milline alamhulk pole toele seatud kitsenduste mõttes suur. Nii tehes on tulemuseks märksa väiksem arv kandidaathulki, andes nii ajalises kui ka mäluvajaduse mõttes palju paremaid tulemusi.

Edaspidi nimetame kaupade arvu hulgas tema suuruseks ning suurusega k kaupadehulka k-kaupadehulgaks. Kaupu hulgas hoitakse leksikograafi lises järjestuses. Notatsiooni $c[1] \cdot c[2] \cdot ... \cdot c[k]$ kasutame kujutamaks k-kaupadehulka c mis koosneb kaupadest $c[1], c[2], ..., c[k]$, kus $c[1] < c[2] < ... < c[k]$. Kui $c = X \cdot Y$ ja Y on m-kaupadehulk siis ütleme, et Y on X m-laiendus. Iga kaupadehulgaga c on seotud loendurväli *c.count* säilitamaks tema tuge üle andmestu. Selle välja väärtuseks saab hulga loomisel 0.

Vaatleme nüüd algoritmi Apriori (joonis 1). Algoritmi esimene samm loeb kokku üksikute kaupade esinemiste arvu baasis T leidmaks suuri 1-kaupadehulki. Iga järgmine samm k koosneb kahest osast. Kõigepealt leitakse funktsiooni AprioriGen abil eelmiselt sammult saadud suurtest kaupadehulkadest L_{k-1} kandidaathulgad C_k. Seejärel leitakse baasist T tugi neile hulkadele ning jäetakse alles vaid suured hulgad. Et vähendada andmebaasi läbivaatuste arvu kasutatakse kiireks loendamiseks funktsiooni AprioriSubset (kirjeldatakse allpool).

Funktsioon AprioriGen võtab argumendina sisse suurte kaupadehulkade hulga L_{k-1} ning tagastab kõikide suurte k-kaupadehulkade ülemhulga C_k. Algoritm töötab kahe sammuna [AS94]: kõigepealt ühendatakse liitmissammus L_{k-1} hulgaga L_{k-1} (joonis 2)

ning seejärel eemaldatakse harvendamissammus (joonis 3) kõik kaupadehulgad $c \subset C_k$ mille ükskõik milline $(k-1)$-alamhulk ei sisaldu L_{k-1} sees.

Näiteks olgu L_3 meil $\{\{abc\}, \{abd\}, \{acd\}, \{ace\}, \{bcd\}\}$. Pärast liitmissammu on C_4 sees kaupadehulgad $\{abcd\}$ ja $\{acde\}$. Harvendamissammu käigus eemaldatakse kaupadehulk $\{acde\}$ sest tema alamhulk $\{ade\}$ ei sisaldu hulgas L_3. Seega jääb C_4 sisse alles ainult $\{abcd\}$.

Funktsiooni AprioriSubset ülesandeks on leida kõik kaupadehulgad mis sisalduvad nii C_k kui ka hetkel loendamissammu poolt käsitletavas ostus t.

4.2 Reeglite leidmine sagedaste hulkade alusel

Olgu meil suur kaupadehulk l. Vastavate reeglite leidmiseks tuleb lihtsalt leida selle kõik mittetühjad alamhulgad. Iga sellise alamhulga a kohta leiame reegli kujul $a \Rightarrow (l-a)$ kui *support*(l)/*support*(a)$\geq$*minconf*.

Kirjeldatud skeemi suure kaupadehulga alamhulkade leidmiseks saab sügavuti rekursiooniga kiirendada. Näiteks hulga $ABCD$ korral vaatleme kõigepealt hulka ABC, seejärel AB etc. Kui suure kaupadehulga l mingist alamhulgas a ei tule

proc **Apriori**

- **Sisend**:
  - ***T***: ostude baas
  - ***minsup***: kaupadehulga suuruse kriteerium
- **Väljund**:
  - ***vastus***: suured kaupadehulgad

1: $L_1 = \{$*suured 1-kaupadehulgad*$\}$
2: **for** $k = 2;\ L_{k-1} \neq \emptyset;\ k$++ **do**
3: $C_k =$*AprioriGen*(L_{k-1})
4: **for all** $tehingud\ t \in T$ **do**
5: $C_t =$*AprioriSubset*(C_k, t)
6: **for all** *kandidaathulgad* $c \in C_t$ **do**
7: *c.count*=*c.count*+1
8: **end for**
9: **end for**
10: $L_k = \{c \in C_k\ |$*c.count*$\geq minsup\}$
11: **end for**
12: *vastus*$= \bigcup_k L_k$

Joonis 1: Algoritm Apriori

reeglit (toele seatud kitsenduste mõttes), siis võib öelda, et ka l alamhulki ei maksa enam reeglite leidmisel arvestada. Näiteks kui reegel $ABC \Rightarrow D$ ei ole eelnevalt seatud usaldusteguri kitsenduste mõttes oluline, siis ei ole vaja ka kontrollida enam reeglit $AB \Rightarrow CD$. Nii ei lähe ükski reegel kaotsi, sest iga a alamhulga $\bar{a}$ tugi peab olema vähemalt sama suur kui hulgal a. Seetõttu ei saa ka $\bar{a} \Rightarrow (l - \bar{a})$ usaldustegur olla suurem kui $a \Rightarrow (l - a)$ usaldustegur. Seega, kui a kasutamisel ei teki reeglit kõigi l elementidega kus a oleks reegli paremal poolel, ei tee seda ka $\bar{a}$. Selle idee võtab kokku algoritm joonisel 4.

5 Implementatsioon

Kõigi elementide kombinatsioonide esinemiste loendamine üle terve andmestu on küllalt suuremahuline töö. Seetõttu on oluline vähendada andmestu läbivaatuste arvu. Christian Borgelt ja Rudolf Kruse on välja pakkunud algoritmi Apriori kiire implementatsiooni [BK02], mis põhineb prefi ksipuu kasutamise abil kandidaathulkade genereerimisest loobumises [HPY00].

Leidmaks sagedasi kaupadehulki tuleb üle lugeda kõik ostukorvid kus nad esinevad. Kirjeldatava implementatsiooni ideeks on kaupadehulkade loendurite esitus

proc **AprioriGenJoin**

- **Sisend**:
  - L_{k-1}: $(k-1)$-kaupadehulgad
- **Väljund**:
  - C_k: k-kaupadehulgad

1: $C_k = \emptyset$
2: **for all** $p, q \in L_{k-1}$ **do**
3: **if** $p.kaup_1 = q.kaup_1 \wedge ... \wedge p.kaup_{k-2} = q.kaup_{k-2} \wedge p.kaup_{k-1} < q.kaup_{k-1}$ **then**
4: $C_k = C_k \cup \{p.kaup_1, p.kaup_2, ..., p.kaup_{k-1}, q.kaup_{k-1}\}$
5: **end if**
6: **end for**

Joonis 2: Funktsiooni AprioriGen liitmissamm

erilise prefiksipuuna (vt joonis 5) mis mitte ainult ei hoia kokku mälu vaid lubab ka efektiivselt töödelda ostukorve ning hiljem esitada reegleid. Iga n_S kujutab endast loendurit kaupadehulga S jaoks. Servade lipikud juurtipust mõne lehttipuni tähistavad ühisosa neile loenduritele mis antud tipus paiknevad. Kujutatud puu pole balansseeritud kuna leksikograafilise järjestuse nõude tõttu loeme n_{abc} samaks mis n_{bca} ning seetõttu vajatakse vaid üht neist loenduritest.

Apriori algoritmi esimese sammuna luuakse eelpoolkirjeldatud puu tase taseme haaval. Esimese baasi läbivaatusega luuakse 1-kaupadehulgad, teise läbivaatusega 2-hulgad jne. Puu selline ehitusviis lubab meil järjest välja jätta neid tippe mille tugi ei vastanud eelnevalt toele seatud kitsendustele — tänu apriori eeldusele on meil teada, milline puu haru sisaldab endas sagedasi kaupadehulki ja milline kindlasti mitte.

Puu tippudes olevate loendurite hoidmiseks on algselt välja pakutud kolm võimalust:

1. Täisarvude vektor, kus iga element tähistab ühe kaupadehulga tuge. Selle struktuuri juures ei ole meil otseselt viidet konkreetsele kaupadehulgale vaid igale kaupadehulgale vastab vektori konkreetne element. Seetõttu tuleb säilitada loendureid ka nende kaupadehulkade jaoks mille kohta eelmisest sammust on teada, et need on kindlalt väikesed — vektoris ei ole tühimikud lubatud.

2. Vektor kaupadehulkade identifikaatoritest ja vastavatest loenduritest, järjestatuna kaupadehulga identifikaatori järgi.

3. Paisktabel. Siin on aga probleemiks efektiivse paiskfunktsiooni leidmine. Seetõttu ei paku ta piisavalt kiirusevõitu küllalt suure mälukasutuse kõrvalt

proc **AprioriGenPrune**

- **Sisend**:
  - C_k: k-kaupadehulgad
- **Väljund**:
  - C_k: harvendatud k-kaupadehulgad

1: **for all** $kaupadehulgad\ c \in C_k$ **do**
2: **for all** $c\ (k-1) - alamhulgad\ s$ **do**
3: **if** $s \notin L_{k-1}$ **then**
4: $C_k = C_k - c$
5: **end if**
6: **end for**
7: **end for**

Joonis 3: Funktsiooni AprioriGen harvendamissamm

ja edaspidi me seda võimalust ei vaatle.

On näha, et esimene viis loendurite säilitamiseks on kõige mälunõudlikum aga ka kõige kiirem. Samuti saab tippude jaoks valida, millist struktuuri loendurite säilitamiseks kasutada. Kui suhteliselt suur osa võimalikest kaupadehulkadest on suured siis on mõtet kasutada esimest meetodit. Tühikute arvu saab vähendada ka kaupadele sobivate märgiste valimisega, näiteks järjestades nad esinemissageduste järgi.

Ostude andmestu T töötlemine, ehk sagedaste hulkade loendamine, on eelpool kirjeldatud puu korral suhteliselt lihtne. Eeldades, et kaubad on konkreetses ostus leksikograafi liselt järjestatud, võib puu lihtsalt rekursiivselt läbi käia ning vastavalt igas tipus olevat loendurit suurendada või uusi tippe luua:

1. leida ostu esimesele kaubale vastav alamtipp ning rakendada ülejäänud kaupadehulka selles ostus rekursiivselt sellele alamtipule
2. jätta ostu esimene kaup kõrvale ning rakendada allesäänud kaupadehulka sellele tipule

. Iga sammu juures tuleb suurendada ka vastavaid loendureid. Kui me just lisasime puule uue taseme, siis kasvatame selle uue taseme loendurit ning edasi ei liigu. Sedasi tagatakse, et kõikide ostu alla kuuluvate kaupadehulkade loendurid on korrektselt suurendatud.

Kirjeldatud implementatsiooni puuduseks on süntaktiliste kitsendustega mitte-arvestamine kuigi nende lisamine ei ole kuigi keeruline. Samuti võib tööd jätkata selle implementatsiooni realiseerimisel SQL põhisena.

1: **for all** *suured kaupadehulgad* $l_k, k \geq 2$ **do**
2: call GenRules(l_k, l_k)
3: **end for**

proc **GenRules**

- **Sisend**:
  - l_k: suur k-kaupadehulk
  - a_m: suur m-kaupadehulk

1: $A = \{(m-1) - kaupadehulgad\, a_{m-1} \mid a_{m-1} \subset a_m\}$
2: **for all** $a_{m-1} \in A$ **do**
3: $conf = support(l_k)/support(a_{m-1})$
4: **if** $conf \geq minconf$ **then**
5: väljasta reegel $a_{m-1} \rightarrow (l_k - a_{m-1})$, usaldusteguriga $conf$ ja toega $support(l_k)$
6: **if** $m - 1 > 1$ **then**
7: call GenRules(l_k, a_{m-1})
8: **end if**
9: **end if**
10: **end for**

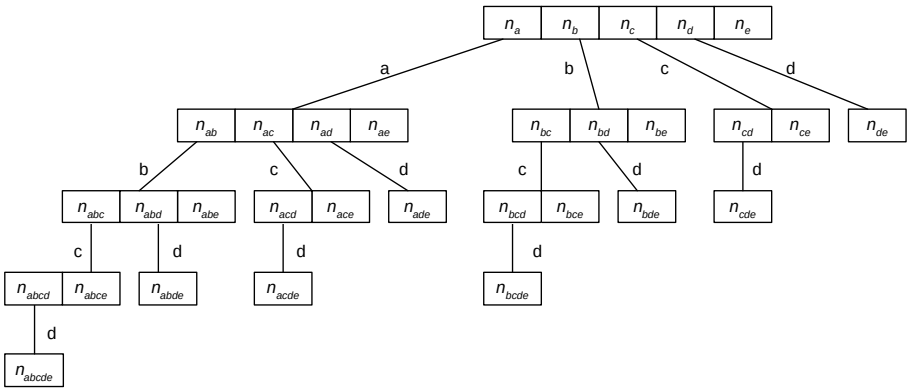
Joonis 4: Assotsiatsioonireeglite genereerimise algoritm

On tehtud töid ka assotsiatsioonireeglite leidmiseks paralleeltöötluse vormis [HKK97, JSGV00] ent sellest me käesoleva artikli piiratud mahu tõttu kahjuks ei räägi.

6 Rakendusi

Assotsiatsioonireeglite kaevandamise rakendusvaldkondi on palju rohkem kui jaekaubandus ning veebilehtede külastatavuse analüüs. USA relvajõududes kasutatav automaatne raketitõrjesüsteem Patriot, mis leidis laialdast kasutust Lahesõjas, kasutab assotsiatsioonireeglitel põhinevat otsustussüsteemi [Ros00]. Automaatreþiimil suudab see süsteem saamaegselt jälgida sadu lendavaid objekte ning otsustab teatud reeglite põhjal punktiarvestust pidades millist neist objektidest alla tulistada. Näiteks lennumasin, mis tuleb läbi eeldefi neeritud «turvakoridori», saab mõned punktid. Kui tema lennutrajektoor pole aga piisavalt paralleelne turvakoridori omaga siis ta kaotab mõned punktid. Kui aga kiirus on teatud piires siis on see hea. Lendobjekt, mis siseneb turvakoridori altpoolt radari nähtavusala (nn pop-up), saab jälle miinuspunkte. Kui tema raadioseadmed ei suuda identifi tseerida teda kui sõbralikku objekti, kaotab ta veel mõned punktid. Raketitõrjesüsteem otsustab objekti allatulistamise kasuks kui punktisumma on piisavalt madal.

Assotsiatsioonireeglite abil on võimalik leida reegleid ka dokumentidest, esita-



Joonis 5: Täielik kaupadehulkade puu kaupade a, b, c, d ja e jaoks

des dokumentides esinevaid (olulisi) sõnu kujul (d, w), kus d on dokumendi identifikaator ning w tähistab dokumendis esinevat sõna. Ehk tegemist on suure (ja ilmselt ka hõreda) maatriksiga $A(d, w)$, kus read tähistavad dokumente ning verud sõnu ning iga maatriksi element näitab kas sõna esinemise fakti dokumendis või siis tema esinemiste arvu [Man02].

Sarnaselt võib kujutada ka veebiserveri logifaili, kolmikuna (h, p, t), kus h on päringu esitanud tööjaama identifikaator, p on vaadeldud lehekülje identifikaator ning t on pöördumise aeg [Man02]. Jättes kõrvale aja, saame eelmises lõigus toodud näitega sarnase esituse.

Selliste esitusviiside tulemusena tekivad küllalt suurte mõõtmetega andmestud, dokumendiandmebaasidel on täheldatud sadu tuhandeid dimensioone [Man02]. Samas on need maatriksid suhteliselt hõredad, seetõttu tegelik dimensioonide arv on väiksem. Samas on siin selgelt näha vajadus dimensioonide vähendamise meetodite järele.

Üks huvitavaid rakendusi on ka sündmuste järjendites muudatuste leidmine, ajamõõdet järgides. Sellist lähenemist on rakendatud oluliste muudatuste leidmises demograafiliste andmete (sünniaeg, abielu ning püsiva partnerisuhte algus ja lõpp, laste sünniajad) seast [BFPB01].

Laialdast kasutamist on assotsiatsioonireeglid leidnud ka veebispetsiifiliste andmete töötlemisel [Tii02]. Üks võimalik rakendus on veebilehtede mugandamine konkreetse kasutaja vajadustele vastavaks [ora02]. Oletame, et veebilogist leidsime reegli $A, B \Rightarrow C \mid 80\%$, kus A, B ja C on veebilehed. Seda võib tõlgendada nii, et kui kasutaja on küsinud lehti A ja B siis on 80% tõenäosus, et ta küsib ka lehte C. Lehele C ei pruugi olla otsest linki lehtedelt A ja B. Selle reegli järgi võib selle lingi konkreetse kasutaja konkreetse sessiooni ajal dünaamiliselt luua.

7 Lahtised küsimused

Üks probleeme assotsiatsioonireeglite leidmisel on leitud reeglite seast huvitavate ning oluliste reeglite tuvastamine [ST96]. Samuti võiksid need reeglid olla hõlpsalt tõlgendatavad ning rakendatavad. Ühed võimalikest huvitavuse kriteeriumitest ongi erinevad kitsendused, nii süntaktilised kui ka toe ning usaldusteguri mõttes. Samas nii võib tekkida suur hulk küllalt tugevaid reegleid mis ei osutu aga reaalse elu taustal küllalt huvitavateks. On välja pakutud ka eeldefi neeritud mallidel põhinevaid reeglite valiku meetodeid [ST96] kui ka semantilisel lähenemisel põhinevaid meetode [BP01]. Samas on siin ikkagi jäetud küllalt suur osa tegevusest inimese kanda ning arenguruumi veel jätkub.

Töö käib ka ajadimensiooni lisamisel assotsiatsioonireeglite analüüsi. Selle näiteks võime võtta teatud tüüpi reeglid, mis kehtivad vaid kindlal ajavahemikul, näiteks kohvi ja saiakesi müüakse koos enamasti hommikutundidel. Samuti on piparkoogitaigna müük suhteliselt harvaesinev nähtus, ent detsembrikuus võib täheldada jälle vastupidist. On uuritud näiteks tsüklilisi assotsiatsioonireegleid [ORS98] ent need jäävad hätta selliste reaalsest elust pärit väidetega, nagu «iga kuu esimene tööpäev», kuna tööpäevade vahelised kaugused pole konstantsed. On uuritud kalendripõhiste assotsiatsioonireeglitega seotud probleeme [LNWJ01, LLC01] ent seni pole veel päris selge, millise piirini on võimalik assotsiatsioonireegleid üldistada [Man02]. Ajamõõtmega tegelemisel üks huvitav lahendus on ka spetsiaalse mustrituvastusriistvara ning geneetiliste algoritmide kasutamine [HS02, SH].

Lahtine on ka kaubahierarhiatega tegelemise aspekt. Jaemüügis on palju valdkonnasisest teavet peidetud hierarhiliste struktuuride taha. Saku Tume ning A Le Coq Premium käivad õlle alla, õlu ja limonaad omakorda karastusjookide alla. Sedasorti üldistuste kohta käivad reeglid võivad pakkuda väga väärtuslikku üldise taseme informatsiooni. Kaubahierarhiate korral võib ülemisel tasemel olla suurem tugi ostude andmestul kui mõnel konkreetsel alamtasemel. Näiteks «Bock ja kartulikrõpsud» ei pruugi andmebaasis piisavalt tuge omada, ent «õlu ja krõpsud» jälle võib. Kuigi otseselt algoritmid taolisi hierarhiaid ei toeta on välja pakutud meetodeid selle probleemi lahendamiseks [HKMT95, DT01]. Üldjoontes on hierarhiatega tegelemine siiski meetodi lõppkasutaja (inimese) ülesandeks jäetud.

Probleeme tekitab ka tekkivate assotsiatsioonireeglite suur hulk, mis vajab inimese sekkumist kasulikumate reeglite identifi tseerimiseks. Võib ka öelda, et suurte kaupadehulkade leidmine on oluliselt lihtsam ülesanne kui oluliste reeglite tuvastamine [Man02].

Andmete säilitamise poolelt valmistab probleeme algoritmide vajadus andmete maatriksikujulisele esitusele. Sageli hoitakse andmeid siiski normaliseeritud kujul, kus ostukorvid on esitatud paaridena kujul (t, i). Siin t on ostukorvi identifi kaator ning i on kauba identifi kaator. Samuti võib olla eraldi seos konkreetse ostukorvi ja ostu sooritanud isiku, aja jms vahel. Samas on tehtud uurimistööd nn vertikaalse kaevandamise (ik *vertical mining*) vallas, kus andmestu ongi just sellisel normaliseeritud kujul [SHS$^+$00] ning püütakse ära kasutada ka andmebaasisüsteemide poolt pakutavaid standardmeetodeid [HKMT95] .

Hetkel käib ka töö leidmaks algoritme mis võimaldavad leida assotsiatsioone nii strktureeritud kui ka struktureerimata andmete põhjal [SSC97, SCH+98, LAS97].

8 Kokkuvõte

Vaatlesime assotsiatsioonireeglite kaevandamise ülesande püstitust ning algoritmi Apriori kui selle lahendust, samuti mõningaid rakendusi ning lahtisi probleeme.

Kuigi algoritm Apriori kasutamiste ja viitamiste hiilgeaeg paistab möödas olevat[2] on assotsiatsioonireeglite rakendamise valdkond küllalt lai, seda suuresti ka tänu kiirete implementatsioonide ning kommertstarkvara toe tõttu.

Käesoleva artikli autor loodab uurimistööd jätkata ajadimensiooni sisaldavate assotsiatsioonireeglite kaevandamisel ning samuti omab perspektiivi tekstiandmebaasidest trendide leidmine [LAS97], näiteks teadusartiklite kokkuvõtetest. Lähtudes praegusel hetkel levinud andmebaaside disainist tundub küllalt palju kasu olevat ka vertikaalse kaevandamise põhimõtetest lähtuvail algoritmidel. Arvestades asjaolu, et andmebaaside päringukeeled on viimasel ajal pidevalt arenenud, tuleks kaaluda ka SQL põhise algoritmi loomist.

Kuna andmete kasvutrend on hetkel eksponentsiaalne ning arvutusvõimsuse kasv ei suuda sellega sammu pidada, eriti mälu kiiruse osas, on oluline uurida ka assotsiatsioonireeglite paralleelse kaevandamise algoritme.

References

[AFS93] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Effi cient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[2] Teadusartiklite kogu citeseer.nj.nec.com info [AS94] tsiteeringute kohta

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[BBR01] Pavel Berkhin, Jonathan D. Becher, and Dee Jay Randall. Interactive path analysis of web site traffi c. In *Knowledge Discovery and Data Mining*, pages 414–419, 2001.

[BFPB01] H. Blockeel, J. Fürnkranz, A. Prskawetz, and F. C. Billari. Detecting temporal change in event sequences: An application to demographic data. *Lecture Notes in Computer Science*, 2168:29–??, 2001.

[BK02] Christian Borgelt and Rudolf Kruse. Induction of association rules: A priori implementation, 2002.

[BMS97] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: generalizing association rules to correlations. pages 265–276, 1997.

[BP01] Catherine Blake and Wanda Pratt. Better rules, few features: A semantic approach to selecting features from text. In *ICDM*, pages 59–66, 2001.

[Chi96] G. Chirn. *Pattern discovery in sequence databases: Algorithms and applications to DNA/protein classification*. PhD thesis, Department of Computer and Information Science, New Jersey Institute of Technology, 1996.

[CSM97] R. Cooley, J. Srivastava, and B. Mobasher. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[DT01] Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 189–212. Springer-Verlag, 2001.

[Fio02] Fabio Di Fiore. Visualizing interestingness, 2002.

[HKK97] Eui-Hong Han, George Karypis, and Vipin Kumar. Scalable parallel data mining for association rules. pages 277–288, 1997.

[HKMT95] Marcel Holsheimer, Martin L. Kersten, Heikki Mannila, and Hannu Toivonen. A perspective on databases and data mining. In *128*, page 10. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 30 1995.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In Weidong Chen, Jeffrey Naughton, and Philip A. Bernstein, editors, *2000 ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press, 05 2000.

[HS02] Magnus Lie Hetland and Pål S£trom. Temporal rule discovery using genetic programming and specialized hardware. In *Proc. of the 4th Int. Conf. on Recent Advances in Soft Computing (RASC)*, 2002.

[ibm98] *Data Mining - An IBM Overview*, 1998.

[JSGV00] M. Joshi, E. Sam, H. George, and K. Vipin. Parallel algorithms for data mining, 2000.

[LAS97] Brian Lent, Rakesh Agrawal, and Ramakrishnan Srikant. Discovering trends in text databases. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 227–230. AAAI Press, 14–17 1997.

[LLC01] Chang-Hung Lee, Cheng-Ri Lin, and Ming-Syan Chen. On mining general temporal association rules in a publication database. In *ICDM*, pages 337–344, 2001.

[LNWJ01] Yingjiu Li, Peng Ning, Xiaoyang Sean Wang, and Sushil Jajodia. Discovering calendar-based temporal association rules. In *TIME*, pages 111–118, 2001.

[MA] Behzad Mortazavi-Asl. Discovering and mining user web-page traversal patterns.

[Man02] Heikki Mannila. Local and global methods in data mining: Basic techniques and open problems. In *ICALP 2002, 29th International Colloquium on Automata, Languages, and Programming*, Malaga, Spain, July 2002. Springer-Verlag.

[MJHS96] B. Mobasher, N. Jain, E. Han, and J. Srivastava. Web mining: Pattern discovery from world wide web transactions, 1996.

[ora02] *Oracle9i Data Mining Concepts. Release 2 (9.2)*, 2002.

[ORS98] Banu Ozden, Sridhar Ramaswamy, and Abraham Silberschatz. Cyclic association rules. In *ICDE*, pages 412–421, 1998.

[Ros00] Peter Ross. Data mining, 2000.

[SA96] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In Peter M. G.

Apers, Mokrane Bouzeghoub, and Georges Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[SCH+98] Lisa Singh, Bin Chen, Rebecca Haight, Peter Scheuermann, and Kiyoko Aoki. A robust system architecture for mining semi-structured data. In *Knowledge Discovery and Data Mining*, pages 329–333, 1998.

[SH] Pål S£trom and Magnus Lie Hetland. Unsupervised temporal rule mining with genetic programming and specialized hardware.

[SHS+00] Pradeep Shenoy, Jayant R. Haritsa, S. Sundarshan, Gaurav Bhalotia, Mayank Bawa, and Devavrat Shah. Turbo-charging vertical mining of large databases. pages 22–33, 2000.

[SON98] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. Mining for strong negative associations in a large database of customer transactions. In *ICDE*, pages 494–502, 1998.

[SSC97] Lisa Singh, Peter Scheuermann, and Bin Chen. Generating association rules from semi-structured documents using an extended concept hierarchy. In *CIKM*, pages 193–200, 1997.

[ST96] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *Ieee Trans. On Knowledge And Data Engineering*, 8:970–974, 1996.

[SVA97] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining, KDD*, pages 67–73. AAAI Press, 14–17 1997.

[TC00] Shiby Thomas and Sharma Chakravarthy. Incremental mining of constrained associations. In *HiPC*, pages 547–558, 2000.

[Tii02] Asko Tiidumaa. Knowledge discovery in large databases. Master's thesis, Institute of Computer Science, University of Tartu, 2002.

[YWYH02] Jiong Yang, Wei Wang, Philip S. Yu, and Jiawei Han. Mining long sequential patterns in a noisy environment. In *SIGMOD Conference*, 2002.