VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Daugiamačių sekų šablonų analizė

Atliko: 2 kurso, 9 grupės studentas
Klaidas Ivaškevičius (parašas)

Darbo vadovas:
Dr. Linas Bukauskas (parašas)

Vilnius
2012

Turinys

Anotacija

Pagrindinis šio magistro baigiamojo darbo tikslas buvo apžvelgti kai kurių algoritmų ir jų kombinacijų pritaikymą daugiamačiams sekų šablonams analizuoti ir įgyvendinti algoritmą, gebantį tai atlikti. Buvo aprašyta FP-Tree medžio struktūra, kuri yra skirta kompaktiškai saugoti kritiniams (pvz., dažnai pasikartojantiems) duomenims, pateiktas FP-Growth algoritmas, galintis analizuoti tokią duomenų struktūrą ir rezultate pateikiantis visų dažnų elementų šablonų aibę. Pristatyta modifikuotų FP-Growth ir PrefixSpan algoritmų kombinacija – MD-PS-FPG algoritmas, pateikti kai kurių atliktų testavimų rezultatai, tolimesnių darbų pagrindiniai tikslai ir pan.

Summary

The main goal of this master final work was to present some of the algorithms and their combinations for the multidimensional sequence pattern mining and implement an algorithm, that is capable of doing that. FP-Tree, that is used to store critical (for example, often repeated) data, was described. FP-Growth algorithm, that can analyze FP-Tree structure and give frequent pattern set as a result, was presented. MD-PS-FPG algorithm – a combination of modified FP-Growth and PrefixSpan algorithms – was introduced. The results of some tests, further work objectives and other things were also presented.

Įvadas

Duomenų analizė kaip mokslo šaka oficialiai buvo pristatyta tik 1990 metais, tačiau ši disciplina sparčiai populiarėja daugelyje sričių, pradedant paprasčiausiais klasikiniais statistiniais skaičiavimais, pritaikomais bet kurioje kasdieninėje rinkoje, baigiant sudėtingiausiais dirbtinio intelekto ir pan. sprendimais bioinformatikoje, karo pramonėje ar astronautikoje.

Duomenys gali būti saugomi įvairiais pavidalais bei būdais. Turint didelius duomenų rinkinius ir norint juos analizuoti, reikia turėti ir tam skirtus metodus bei įrankius tam atlikti. Vienas dažniau analizuojamų atvejų yra duomenų analizė sekų duomenų rinkiniams. Algoritmų tiek sekoms, tiek kitokioms duomenų struktūroms analizuoti yra pasiūlyta įvairiausių, kiekvienas turi savų privalumų bei trūkumų, yra geresnis vienokiems ar kitokiems duomenims ir pan. Taip pat neužtenka turėti algoritmų tik vieno ar dviejų matmenų duomenims analizuoti, kadangi duomenys yra tarpusavyje įvairiai susiję, yra skaidomi pagal tam tikrus kriterijus ir t.t.

Pagrindinis šio darbo tikslas – tiek mažesnių, tiek didesnių dimensijų aibių ir sekų šablonų analizės algoritmų pristatymas bei praktinis modifikuotų PrefixSpan ir FP-Growth algoritmų kombinacijos (pavadintos MD-PS-FPG) įgyvendinimas. Pirmojoje darbo dalyje apžvelgiami populiariausi aibių bei sekų šablonų analizės algoritmai, skirti mažesnių dimensijų duomenų rinkiniams analizuoti. Taip pat aprašyti pagrindiniai asociatyvumo taisyklių analizės apibrėžimai, pateikta sekų šablonų analizės algoritmų tipizacija. Antrojoje darbo dalyje pateikti kai kurie populiaresni algoritmai, skirti didesnių dimensijų duomenų rinkinių analizei. Trečiojoje darbo dalyje pristatoma FP-Tree duomenų struktūra, skirta kompaktiškai saugoti kritiniams (pvz., dažnai pasikartojantiems) duomenims. Taip pat pateikiamas FP-Growth algoritmas, kuris moka analizuoti tokią duomenų struktūrą ir rezultate pateikia visų duomenų rinkinio dažnų elementų šablonų aibę. Toliau aprašytas praktinis MD-PS-FPG algoritmo įgyvendinimas, atliktos modifikacijos bei papildomai atlikti darbai, pateikti kai kurių atliktų testavimų rezultatai, tolimesnių darbų pasiūlymai ir pan.

Rašant darbą buvo remtasi H. Pinto, R. Agrawal, R. Srikant, Y. Chen, T. Huang, Y. Yin, R. Mao bei kitų autorių moksliniais straipsniais ir knygomis. Daugiausia nagrinėta literatūra: H. Pinto MULTI-DIMENSIONAL SEQUENTIAL PATTERN MINING, 1998; J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, Mei-Chun Hsu, Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach, 2004.

1. Aibių bei sekų šablonų analizės metodai

Duomenų augimas yra viena pagrindinių šiuolaikinių informacinių technologijų problemų, tačiau tiesiog turėti duomenis neužtenka. Kyla poreikis duomenis analizuoti įvairiais aspektais, gauti iš jų kažkokią naudą. Čia susiduriama su kitomis problemomis – kaip analizuoti didelius duomenų kiekius, kokius metodus naudoti, kokie duomenys yra svarbesni, kokie gali būti daugiau ar mažiau ignoruojami ir t.t. Tokių duomenų analizėje pagrinde vyrauja algoritmai, kurie yra skirti vienos arba dviejų dimensijų duomenų rinkiniams analizuoti (pvz., Apriori [AS94], GSP [AS95], PSP [MCP98], SPADE [Zak01], FreeSpan [HPM+00], PrefixSpan [PHM+04] bei kiti algoritmai).

1.1. Asociatyvumo taisyklių analizė

Dideliuose duomenų kiekiuose ryšiams tarp kintamųjų rasti yra naudojamas populiarus ir gan gerai išnagrinėtas duomenų analizės metodas – asociatyvumo taisyklių analizė. Tarkime, turime elementų aibę $I = \{i_1, i_2, \ldots, i_n\}$ ir duomenų rinkinį sudarančią transakcijų aibę $DB = \{t_1, t_2, \ldots, t_n\}$, kuri yra sudaryta iš porų $T(t,i)$, kur $t \in$ – unikalus transakcijos ID ir $i \in$ – elementų aibės poaibis.

Implikacija – tai loginė operacija, kai iš teiginių A ir B sudaromas teiginys „jei A, tai B“, žymima: $A \Rightarrow$ [DGJ11]. Taisyklė apibrėžiama kaip implikacija $X \Rightarrow$, kur $X, Y \subseteq$ ir $X \cap Y = \emptyset$ [AIS93].

Elementų aibės dažnumo galia (angl. *support*) – tai santykis tarp skaičiaus, kiek įrašų turi tą aibę, ir visų įrašų skaičiaus (žymima $supp(X)$).

Taisyklės patikimumas (angl. *confidence*) yra apibrėžiamas, kaip santykis $conf(X \Rightarrow) = \frac{supp(X \cup)}{supp(X)}$.

Asociatyvumo taisyklių analizės algoritmams dažniausia yra nurodomi du parametrai: minimali aibės galia ir minimalus patikimumas. Taisyklių generavimas dažniausia skaidomas į du žingsnius: pirmiausia surandamos dažnos elementų aibės, panaudojant aibės galią; tada tinkamos aibės ir nustatytasis patikimumas yra panaudojami taisyklėms formuoti. Kadangi, norint surasti visas dažnas aibes, reikia pereiti per apskritai visas elementų aibes (kas yra gan sudėtingas uždavinys), tai yra išnaudojamos dažnų ir nedažnų aibių antimonotoniškumo savybės: visi dažnos elementų aibės poaibiai yra taip pat dažni ir kiekvienos nedažnos elementų aibės viršaibiai yra taip pat nedažni.

1.2. Apriori algoritmas

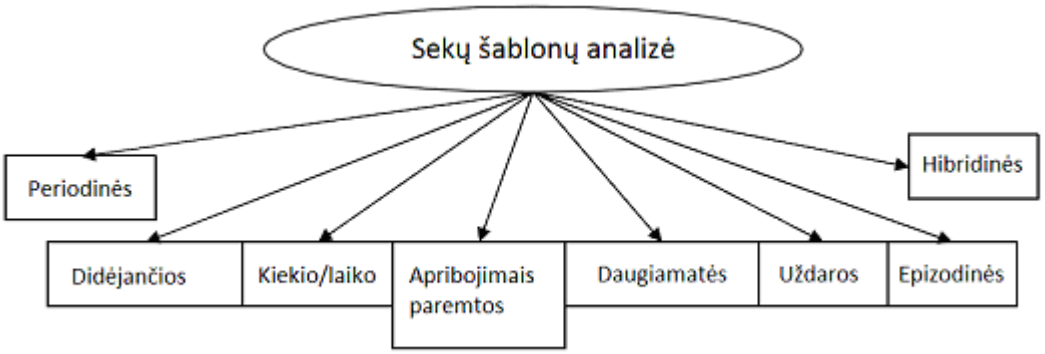
Klasikinis algoritmas, skirtas asociatyvumo taisyklėms formuoti, yra Apriori algoritmas. Šis algoritmas neretai naudojamas kaip kitų algoritmų dažnoms aibėms ar sekoms rasti pagrindas. Apriori naudoja kandidatų generavimą ir skaičiuoja elementų aibių galias dažnų elementų aibių radimui. Kandidatams generuoti yra remiamasi „iš apačios į viršų“ idėja: sukuriama kandidatų aibė iš vieno elemento aibių; netenkinantys numatytosios galios kandidatai yra atmetami; generuojami kandidatai iš dviejų elementų bei atmetami netinkantys; sekantys analogiški veiksmai vykdomi iki tol, kol nebegalima sugeneruoti naujų kandidatų. [AS94]

1.3. Algoritmai sekoms

Praktikoje dažniausiai yra svarbus elementų išsidėstymas duomenų rinkinyje. Pvz., norima žinoti, kaip dažnai tam tikros parduotuvės klientai, įsigiję vieną daiktą, po to įsigyja kitą; arba kaip dažnai tam tikros interneto svetainės lankytojai eina iš vieno puslapio į kitą, iš pastarojo į dar kitą ir t.t. Duomenims įvairiais norimais aspektais analizuoti tenka atsižvelgti į duomenis, saugomus kaip sekas. Sekų šablonų analizės tikslas yra sekų duomenų rinkinyje rasti visus dažnus sekų posekius kaip šablonus, kurie atitinka nustatytą dažnumo galią.

Sekų šablonų analizės algoritmai gali būti skirstomi pagal įvairius kriterijus. Pvz.:

- pagal algoritmo pagrindą:
    - Apriori algoritmo pagrindu paremti algoritmai (GSP, SPADE);
    - šablonų auginimo pagrindu paremti algoritmai (FreeSpan, PrefixSpan);
    - grafų, maišos ir pan. pagrindu paremti algoritmai (DSG, HPA);
- pagal laiko intervalus:
    - algoritmai be laiko intervalų (GSP, PrefixSpan);
    - algoritmai su tiksliai apibrėžtais laiko intervalais (I-Apriori, I-PrefixSpan, MI-Apriori, MI-PrefixSpan);
    - algoritmai su netiksliai apibrėžtais laiko intervalais (FTI-Apriori, FTI-PrefixSpan, FuzzMI-Apriori, FuzzMI-PrefixSpan); [Hua10]
- pagal duomenų rinkinių savybes:
    - algoritmai duomenų srautams analizuoti;
    - algoritmai vienamačiams ar daugiamačiams duomenų rinkiniams analizuoti;
    - algoritmai uždarų aibių ar sekų šablonams analizuoti;
    - ir t.t.



1.3.1 pav. Sekų šablonų analizė pagal sekų tipus [Hua95]

1.3.1. Periodinių sekų šablonų analizė

Periodinių sekų šablonų paieškos tikslas yra rasti visus duomenų rinkinyje pasikartojančius sekų šablonus, vertinant laiko momentus. Tokio tipo sekų šablonai gali būti laikomi sekų šablonų analizės plėtiniu, atsižvelgiant į laikotarpius kaip į padalintų sekų aibę. Periodinių sekų šablonų uždaviniai gali būti klasifikuojami pagal dvi pagrindines kategorijas:

1. Užbaigtų/pilnų periodinių sekų šablonų analizė, kur kiekvienas laiko momentas turi įtakos bendram cikliniam laikotarpių režimui;
2. Neužbaigtų/dalinių periodinių sekų šablonų analizė, kur tik kai kurie (ne visi) laiko momentai turi įtakos bendram periodiniam laikotarpių režimui. [Hua95]

1.3.2. Didėjančių sekų šablonų analizė

Didėjančių sekų šablonų analizė yra išskirtinė tuo, kad pagrindinis dėmesys sutelktas į duomenų kitimą bėgant laikui. Paprastu atveju, kai duomenys atsinaujina retai, užtenka perskaičiuoti sekų šablonus visam duomenų rinkiniui, tačiau to neužtenka didėjančių sekų šablonų analizei, kur duomenys atsinaujina dažnai. Todėl tokiems atvejams spręsti yra taikomi kitokie metodai. [Hua95]

1.3.3. Kiekio/laiko sekų šablonų analizė

Duomenų rinkiniuose neretai vyrauja skaitiniai ar laiką apibrėžiantys duomenys. Kai kurie sekų šablonų algoritmai analizuoja sekų su laiko intervalais duomenų rinkinius, kur svarbi ne tik elementų išsidėstymo tvarka, bet ir laiko intervalai tarp tų elementų. Kiti algoritmai prijungia kiekio svarbą elementų sekose (svorius) ir pan. [Hua95]

1.3.4. Apribojimais paremtų sekų šablonų analizė

Minimalus galios slenkstis yra svarbus sekų šablonų analizės parametras. Neretai domina ne apskritai visi, bet tik tam tikri sekų šablonai. Tam yra pasiūlyta algoritmų, kuriems galima nurodyti kitus specifinius parametrus, kurie į rezultatų aibę atrenka tik tam tikrus norimus sekų šablonus. [Hua95]

1.3.5. Uždarų sekų šablonų analizė

Uždarų sekų šablonas yra sekų šablonas, neįeinantis į jokį kitą sekų šabloną, kuris turi tokią pačią dažnumo galią. Pagrindinė tokių šablonų analizės idėja yra ne tik surasti kompaktiškesnę rezultatų aibę, bet ir efektyvumas. Vieni žinomesnių pasiūlytų uždarų sekų šablonų analizės algoritmų yra CloSpan, BIDE, TSP. [Hua95]

1.3.6. Epizodinių sekų šablonų analizė

Epizodinių sekų šablonų analizės algoritmai analizuoja sekas, kurios yra sudarytos iš įvykių pagal tam tikrą struktūrą ir kurios yra santykinai artimos viena kitai laiko atžvilgiu pagal nustatytą slenkstį. Tokių sekų šablonai suteikia gan plačias galimybes analizuoti duomenis, susijusius su laiku. Kai kurie algoritmai taip pat siūlo nurodyti įvairius parametrus, kad būtų atrinkti tik tam tikrą norimą informaciją turintys sekų šablonai. [Hua95]

1.3.7. Hibridinių sekų šablonų analizė

Hibridinių sekų šablonų analizės algoritmai analizuoja sekas, kurios apjungia tolydžius ir netolydžius sekų šablonus. Tolydus sekų šablonas yra apibrėžiamas kaip šablonas, esantis iš eilės einančiose sekų transakcijų pozicijose, o netolydus, priešingai, yra apibrėžiamas kaip šablonas, nesantis iš eilės einančiose sekų transakcijų pozicijose. [Hua95]

1.4. GSP algoritmas

Seka duomenų rinkinyje gali būti dviejų duomenų formatų:

- horizontalaus duomenų formato;
- vertikalaus duomenų formato.

Pirmasis naudoja įprastinį duomenų rinkinio vaizdavimą: <sekos_id : objektų seka>, antrasis naudoja vertikalų vaizdavimą: <objektas : (sekos_id, laikas)>, kuris gali būti gautas transformacijos būdu iš horizontalaus formato sekų duomenų rinkinio. Kaip ir aibėms, sekų šablonams taip pat galioja antimonotoniškumo savybės: jei seka nėra dažna, tai joks jos viršsekis irgi nėra dažnas; kiekvienas netuščias sekų šablono posekis yra sekų šablonas.

GSP (angl. *Generalized Sequential Patterns*) algoritmas buvo sukurtas praplečiant Apriori algoritmą ir pritaikant jį sekoms. Vienu duomenų rinkinio skanavimu algoritmas sugeneruoja visas potencialių kandidatų sekas ir suskaičiuoja jų galias. GSP algoritmas pradžioje ieško dažnų vieno elemento posekių ir juos saugo kandidatų sekų aibėje, kuri skirta dviejų elementų posekių generavimui. Nustatytos galios netenkinantys kandidatai yra atmetami. Likę dažni dviejų elementų posekiai naudojami kitam skanavimui ir t.t. Kandidatų k-ilgio sekos yra generuojamos jungiant dažnus k-1 ilgio posekius. Seka s1 jungiasi su seka s2, jei posekis, gautas pašalinus pirmąjį elementą iš sekos s1, yra lygus poaibiui, gautam pašalinus paskutinįjį elementą iš sekos s2. GSP algoritmas skaito duomenis tol, kol nebegalima sugeneruoti kandidatų. [AS95], [Pin98]

1.5. PSP algoritmas

Tarkime, $\alpha = < e_1, e_2, \ldots, e_n >$ yra seka. Seka $\beta = < e'_1, e'_2, \ldots, e'_m >$ (m ≤ n) yra vadinama α prefiksu tada ir tik tada, jeigu:

1. $\forall \leq \eta - 1 : e'_i = {}_i$;
2. $e'_m \subseteq {}_\eta$;
3. Visi $(e_m - {}'_m)$ elementai po e'_m yra surūšiuoti alfabetiškai.

Tarkime, α yra β posekis (t.y., $\beta \subseteq$). α posekis α ($\alpha' \subseteq$) yra vadinamas α projekcija β atžvilgiu, tada ir tik tada, jeigu:

1. α turi prefiksą β;
2. nėra tokių α viršsekių α (t.y., $\alpha \subseteq$, bet $\alpha \neq \chi$), kad α būtų α posekis ir turėtų prefiksą β. [Pin98]

PSP (angl. *Prefix-tree for Sequential Patterns*) algoritmas buvo sukurtas patobulinti GSP algoritmo kandidatų saugojimo būdui. PSP algoritmas generuoja prefiksų medį, kur kiekviena briauna yra seka kandidatė, o lapo viršūnėje yra saugoma sekos dažnumo galia. Tam reikia mažiau atminties, kandidatų tikrinimas taip pat tampa paprastesnis nei GSP atveju. [Pin98]

1.6. SPADE algoritmas

SPADE (angl. *Sequential PAttern Discovery using Equivalent Class*) algoritmas, priešingai nei GSP, naudoja vertikalų duomenų formatą. Vietoje nuolat skanuojamų duomenų sekų aibėms rasti, šis metodas naudoja gardelės (angl. *lattice*) paieškos metodus ir paprastas jungimo operacijas. Visų pirma yra nuskanuojami duomenys ir sukuriamas vertikalus sąrašas. EID (angl. *Event ID*) – laiko momentas kiekvienam elementui. Kiekvienas elementas yra sudarytas iš poros: EID ir aibės SID (angl. *Sequence ID*) – kurioje sekoje yra elementas. [HPY05]

Dažni vieno elemento posekiai yra randami vienu ID sąrašo nuskanavimu. Dviejų elementų posekiai gali būti išskaičiuojami paverčiant dažnų elementų formatą iš vertikalaus į horizontalų. Formuojant dviejų elementų posekius, apjungiami tą patį SID turintys vieno elemento posekių EID. [Pin98] Procesas vyksta panašiai kaip Apriori algoritme, tačiau vertikalūs sąrašai leidžia atlikti greitesnę kiekvieno posekio paiešką ir atrinkimą. Visi k elementų posekiai gali būti randami pagal laikinus sujungimus, kurie sudaryti iš jau rastų k-1 elementų posekių. Procesas tęsiamas, kol neberandama dažnų sekų arba nebegalima sudaryti naujų sekų, apjungiant turimas.

Nors SPADE algoritmas sumažina duomenų skanavimų skaičių, tačiau pagrindinis paieškos principas yra panašus į GSP algoritmo. Pagrindiniai GSP minusai pasikartoja ir SPADE algoritme. [HPY05]

1.7. FreeSpan algoritmas

FreeSpan (angl. *Frequent pattern-projected Sequential pattern mining*) algoritmas buvo sukurtas daugkartinių Apriori algoritmo kandidatų generavimo ir atrinkimo operacijų skaičiui sumažinti, išlaikant pagrindinius Apriori principus. FreeSpan algoritmas rekursyviai projektuoja sekų duomenų rinkinį į duomenų rinkinių poaibius, kuriuose saugomi posekių fragmentai. Kiekviena projekcija suskirsto duomenis į palaipsniui mažesnius ir paprasčiau valdomus vienetus. [HPY05]

Duomenų rinkinių poaibiams kurti visų pirma dažni posekiai yra surūšiuojami mažėjimo tvarka pagal galią. Pradedant nuo mažiausią galią turinčio elemento formuojamas duomenų rinkinys, į kurį įeina tik tokios sekos, kurios turi tą elementą, atmetant visus elementus, kurie turi mažesnę galią nei einamasis elementas. Taip suprojektuotos sekos bus ilgiausios einamajame duomenų rinkinyje, o tai reiškia, kad duomenų rinkinys turi mažiausią skaičių sekų apskritai. Rekursijos pagalba yra sumažinamas duomenų bazės dydis, nes nelieka dubliuotų duomenų. Tiesa, tai priklauso nuo pradinių duomenų, dėl kurių kai kuriais atvejais galima gauti nepageidaujamus rezultatus. [PHM+04]

1.8. PrefixSpan algoritmas

PrefixSpan [PHM+04] (angl. *Prefix-projected Sequential pattern mining*) algoritmas buvo sukurtas FreeSpan algoritmo kaštams sumažinti. Jo pagrindinė idėja yra rasti dažnus sekų posekius kaip šablonus, projektuojant duomenų rinkinius ir saugant juos prefiksų pavidalu, nes bet koks dažnas posekis visada gali būti randamas auginant dažną prefiksą.

TId	Naršymo seka
1	(f) (a c) (a c d g) i (m c p)
2	a (b c) (a c f) (a c l) m o
3	(a b) (a d f) h j (a d o)
4	(b c) (k s p)
5	(a c) (a c f) (e l p m) n

1.8.1 pav. Svetainės X puslapių naršymo transakcijų pavyzdys

1.8.1 pav. stulpelyje „Naršymo seka“ pateiktos svetainės X puslapių naršymo transakcijos – sekos su lankytais puslapiais, sugrupuotais pagal posekius, kurie reiškia, kad puslapiai buvo lankyti to paties vartotojo vienu metu, sekantys puslapiai – kitu metu, dar sekantys – dar kitu metu (pvz., „(f) (a c) (a c d g) i (m c p)“ reiškia, kad iš pradžių buvo aplankytas puslapis f, po to – a ir c, tada – a, c, d ir g, dar po to – i ir galų gale m, c bei p puslapiai).

Tarkime, kad minimali dažnumo galia yra lygi 3. PrefixSpan algoritmas visų pirma ieško 1-ilgio (sudaryto iš vieno elemento) dažnų posekių, vieną kartą nuskaitydamas duomenų rinkinį. Pateiktame pavyzdyje tokie posekiai su savo galiomis yra: (a) : 4, (c) : 4, (f) : 4, (b) : 3, (m) : 3, (p) : 3. Toliau yra siaurinama paieškos aibė: rezultate galės būti tik posekiai, kurių prefiksai prasideda (a), (b), (c), (f), (m) arba (p). Tada kiekvienam šiam prefiksui yra projektuojami duomenų rinkiniai. Pvz., ieškoma dažnų posekių su prefiksu (a) (imamas tik posekis nuo pirmojo (a)): < (_ c) (a c d g) i (m c p) >; < (b c) (a c f) (a c l) m o >; < (_ b) (a d f) h j (a d o) >; < (_ c) (a c f) (e l p m) n >. Tokia aibė yra vadinama (a) projekcijos duomenų rinkiniu (angl. *(a)-projected database*). Sekančiu žingsniu yra skanuojamas (a) projekcijos duomenų rinkinys ir atrenkami minimalią galią tenkinantys posekiai: (a) : 4, (c) : 3, (f) : 3, (m) : 3. Iš čia randami 2-ilgio dažni posekiai: (a a) : 4, (a c) : 3, (a f) : 3, (a m) : 3. Toliau konstruojami atitinkamai (a a), (a c), (a f) bei (a m) projekcijų duomenų rinkiniai ir tokia procedūra vykdoma rekursiškai, kol nebegalima sugeneruoti dažnų posekių. Po to imamas (b) prefiksas ir ieškoma dažnų posekių su prefiksu (b). Ir taip toliau, kol pereinama per visus prefiksus.

PrefixSpan našumas gali būti patobulintas alternatyvaus lygio arba pseudo projekcija – procesu, kuris išnaudoja savybę, kad kiekvienas vaikinis suprojektuotas duomenų rinkinys yra tėvo duomenų rinkinio poaibis. Pvz., suprojektuotos sekos <ab> būtinai turės ir suprojektuotas

sekas <a>. Vietoje abiejų suprojektuotų sekų laikymo atmintyje, pseudo projekcija saugo tik tėvines sekas ir sekos numerį bei poslinkį kiekvienai vaikinei sekai. Tai žymiai sumažina suprojektuotus duomenų rinkinius tiek vietos atžvilgiu, tiek duomenų nuskaitymui reikalingas operacijas. [Pin98]

Palyginimui su GSP ir FreeSpan algoritmais, PrefixSpan algoritmas yra efektyvesnis dėl šių priežasčių:

- nereikalingas kandidatų sekų generavimas. Priešingai nei GSP, PrefixSpan algoritmas tik augina ilgesnius sekų šablonus iš dažnų trumpesnių. Nėra nei generuojami, nei testuojami neegzistuojantys suprojektuotų duomenų rinkinių kandidatai. Šiuo aspektu yra skanuojama žymiai mažesnė duomenų erdvė;
- projektuojami duomenų rinkiniai mažėja.

Didžiausia laiko dalis PrefixSpan algoritme atitenka duomenų rinkiniams projektuoti. Blogiausiu atveju gali būti konstruojamas suprojektuotas duomenų rinkinys kiekvienam sekų šablonui. Tiesa, tai gali būti sušvelninta anksčiau aprašytų pseudo projekcijų pagalba. [PHM+04]

1.9. Kiti algoritmai

Algoritmų sekų duomenų rinkiniams analizuoti yra sukurta įvairių ir vis atsiranda naujų, kurie yra geresni vienokio ar kitokio tipo duomenims, nustatytoms dažnumo galioms bei kitiems parametrams. Toliau pateikta keletas populiaresnių algoritmų, kurie yra dažniau naudojami praktikoje ir jais taip pat neretai remiamasi kuriant naujus algoritmus.

I-Apriori algoritmas, kuris yra paremtas Apriori algoritmu, yra skirtas sekoms su laiko intervalais analizuoti. Nuo paprasto Apriori algoritmo šis algoritmas skiriasi metodu kandidatams generuoti bei metodu, skaičiuojančiu sekų kandidatų galias. Kandidatams saugoti yra naudojamas medis, kuris yra sudarytas iš elemento pavadinimo ir laiko intervalo reikšmės. Skanuojamas visas duomenų rinkinys ir su kiekviena transakcija yra pereinama per kandidatų medį bei paskaičiuojamos kandidatų galios.

I-PrefixSpan algoritmas yra praplėstas PrefixSpan algoritmas, taip pat galintis analizuoti sekų su laiko intervalais duomenų rinkinius.

FTI-Apriori (angl. *fuzzy time-interval Apriori*) bei FTI-PrefixSpan (angl. *fuzzy time-interval PrefixSpan*) algoritmai taip pat paremti atitinkamai Apriori bei PrefixSpan algoritmais ir yra skirti sekų su laiko intervalais analizei. Nuo I-Apriori bei I-PrefixSpan jie skiriasi tuo, kad jų tikslas yra analizuoti sekų duomenų rinkinius su ne tiksliai apibrėžtais laiko intervalais. Algoritmų išskirtinumas yra tas, kad juose naudojami kalbiniai elementai (angl. *linguistic terms*), pvz., *trumpas*, *vidutinis*, *ilgas*, kurie nusako laiko intervalo ilgumą. [CH05]

SPAM (angl. *Sequential PAttern Mining*) algoritmas naudoja vertikalų duomenų formatą. Žymėjimui, ar elementas yra einamojoje transakcijoje, ar ne, yra naudojamas bitų masyvas. [AGY+02]

DISC algoritmas iš pradžių generuoja tam tikrą skaičių sekų kiekvienai transakcijai. Tuomet duomenų rinkinys yra surūšiuojamas pagal pastarąsias transakcijas. Dažnos sekos yra randamos paeiliui tikrinant pirmuosius surūšiuoto duomenų rinkinio įrašų elementus. Jeigu pirmasis įrašo elementas sutampa su elementu, esančiu i-oje vietoje (kur i - minimali sekos dažnumo galia), tai seka yra dažna. [CWC04]

2. Didesnių dimensijų duomenų rinkinių šablonų analizės algoritmai

Sekų šablonų analizėje pagrinde vyrauja algoritmai, kurie yra skirti vienos arba dviejų dimensijų duomenų rinkiniams analizuoti. Tačiau šiuolaikinių sistemų modeliai retai apsiriboja vienmate ar dvimate erdve. Pavyzdžiui, įvairios šiuolaikinės žiniatinklio sistemos siūlo daugybę paslaugų su didele įvairove tarpusavyje susijusių skirtingų technologijų. Vartotojas prisijungia prie sistemos, jam sukuriama sesija, kurios metu jis naršo įvairius svetainės puslapius. Puslapių aibė priklauso sesijų aibei, kuri savo ruožtu priklauso laikotarpių (dienų, savaičių ar pan.) aibei. Gauname jau tris dimensijas. Esant reikalui, sesija gali būti skaidoma į transakcijas, kurios savyje saugo dar kažkokią papildomą informaciją ir t.t. Puslapio lankytojas gali būti neregistruotas, gali būti registruotas ir priklausyti tam tikrai vartotojų grupei; IP adresas, iš kurio prisijungta, gali būti priskirtas tam tikram rėžiui (šaliai, miestui ar pan.); tam tikra naršyklės informacija bei daugybė kitų įvairių duomenų gali smarkiai išplėsti dimensijų kiekį. Taip gauname gan sudėtingus duomenų rinkinius, kuriems analizuoti nėra trivialių algoritmų, kadangi mažesnių dimensijų duomenų rinkinių analizės algoritmai didesnėms dimensijoms netinka (jie neatsižvelgia į dimensijų tarpusavio sąsajų informaciją). Taigi kyla poreikis turėti algoritmus didesnių dimensijų duomenų rinkiniams analizuoti.

2.1 pav. pateiktas svetainės X puslapių naršymo transakcijų duomenų rinkinio pavyzdys, kur *TId* – transakcijos ID, *Grupė* – vartotojo grupė (gN – neprisijungęs, gA – administratorius, gP – paprastas vartotojas ir pan.), *Miestas* – miestas, iš kurio prisijungta (mVln – Vilnius, mKns – Kaunas, mNzn – nežinomas ir pan.), *Naršyklė* – naršyklė, su kuria prisijungta (n1 – naršyklė nr. 1, n2 – naršyklė nr. 2 ir pan.), *Naršymo seka* – seka puslapių, kuriuose lankėsi vartotojas (kiekvienas skirtingas simbolis atitinka skirtingą puslapį).

TId	Grupė	Miestas	Naršyklė	Naršymo seka (unikalūs psl.)	Naršymo seka (pilna)
1	gN	mKns	n3	f a c d g i m p	(f) (a c) (a c d g) i (m c p)
2	gA	mNzn	n2	a b c f l m o	a (b c) (a c f) (a c l) m o
3	gN	mNzn	n3	b f h j o	(a b) (a d f) h j (a d o)
4	gP	mVln	n1	b c k s p	(b c) (k s p)
5	gA	mNzn	n3	a f c e l p m n	(a c) (a c f) (e l p m) n

2.1 pav. Svetainės puslapių naršymo transakcijų pavyzdys

Analizuojant pateikto pavyzdžio duomenų rinkinį galima spręsti, kokie puslapiai daugiausia lankomi skirtingų lankytojų, iš kokio miesto jų dažniausiai sulaukiama, ar daugiau lankosi prisijungę (ir kokios jų grupės) ar neprisijungę vartotojai ir t.t. Toliau atitinkamai galima daryti išvadas, kaip galima tobulinti svetainės navigaciją, kur plėsti rinkodarą, ar naudoti kažkokias

privilegijas prisiregistravusiems ir pan. Toliau pateikiami kai kurie populiaresni algoritmai tokių duomenų rinkinių analizei bei šablonų radimui.

2.1. AprioriMD algoritmas

AprioriMD algoritmas yra modifikuotas Apriori algoritmas, skirtas daugiamačių sekų šablonų analizei. Pagrindinės algoritmo savybės: metodas kandidatų sekoms generuoti; metodas kandidatų sekų galioms skaičiuoti. Kadangi elementai gali būti skirtingose dimensijose, tai elementų posekiams reikia generuoti visas galimas poras iš visų dimensijų. Šiuo atveju galima taikyti antimonotoniškumo savybę, kad visi dažnos k ilgio sekos k-1 posekiai taip pat bus dažni. Taip sumažinama kandidatų aibė. Kandidatų galioms skaičiuoti yra naudojamas kandidatų medis, kurio mazgai sudaryti iš elemento pavadinimo ir dimensijos reikšmės. [YC05]

2.2. PrefixMDSpan algoritmas

PrefixMDSpan algoritmas yra modifikuotas PrefixSpan algoritmas. Pagrindinis skirtumas nuo tikrojo PrefixSpan algoritmo yra tas, kad tarp dažno elemento suprojektuotame rinkinyje ir jo tėvinio rinkinio paskutinio elemento egzistuoja dimensijos srities reikšmė. Paprasto PrefixSpan algoritmo nepakanka, nes bus prarandamas ryšys tarp dimensijų. Tam yra naudojama lentelės struktūra, vadinama IDM (ID matrica), kurios stulpeliuose yra saugomas elementas, o eilutėse – dimensijos srities reikšmė. Turint IDM lentelę, galima rasti dažnus sekų šablonus rekursyviai pereinant per suprojektuotus duomenų rinkinius. [YC05]

2.3. MI-Apriori bei MI-PrefixSpan algoritmai

MI-Apriori bei MI-PrefixSpan algoritmai vėlgi yra labai panašūs atitinkamai į Apriori bei PrefixSpan algoritmus. MI-Apriori algoritme skiriasi metodai kandidatams generuoti bei jų galioms skaičiuoti, o MI-PrefixSpan algoritme skiriasi metodai projektuojamiems duomenų rinkiniams konstruoti, kandidatų sekų šablonams bei jų galių skaičiavimui. [HHY+09]

2.4. FuzzMI-Apriori bei FuzzMI-PrefixSpan algoritmai

FuzzMI-Apriori bei FuzzMI-PrefixSpan algoritmai taip pat yra sukurti panaudojant atitinkamai Apriori bei PrefixSpan algoritmus. Esminis skirtumas yra tas, kad šie algoritmai yra naudojami sekų su daugiamačiais laiko intervalais šablonų analizei. Kaip ir FTI-Apriori bei

FTI-PrefixSpan algoritmų atveju, šiuose algoritmuose taip pat yra naudojami kalbiniai elementai intervalų neapibrėžtumui nusakyti. [Hua10]

2.5. AMGMSP algoritmas

AMGMSP (angl. *Approximate Mining of Global Multidimensional Sequential Patterns*) algoritmas yra skirtas daugiamačių sekų šablonų analizei dideliems duomenų rinkiniams paskirstytoje aplinkoje (angl. *distributed environment*). Visų pirma daugiamatė informacija yra įterpiama į atitinkamas sekas, kad dingtų daugiamatiškumas. Tada sekos yra sugrupuojamos, susumuojamos bei analizuojamos paskirstytose vietose kaip lokalūs šablonai. Globalūs daugiamačių sekų šablonai gali būti analizuojami surinkus visus lokalius vienos paskirstytos vietos šablonus. [HZC07]

2.6. DFSM algoritmas

DFSM (angl. *Divide-and-conquer fuzzy sequential mining*) algoritmas skirtas analizuoti daugiamačiams kiekio sekų šablonams (pvz., svetainės klientas, įsigijęs 3-6 detektyvines knygas, grįš ir nusipirks 5-8 mokslinės fantastikos knygas bei 1-3 romanus). Kiekio neapibrėžtumui naudojami lingvistiniai elementai, pvz., *mažai*, *vidutiniškai*, *daug*. [CH06]

2.7. Kiti algoritmai

UniSeq algoritmo pagrindinė idėja yra į sekas pridėti daugiamatę informaciją, panaudojant PrefixSpan algoritmą. DimSeq algoritmas iš pradžių ieško daugiamačių šablonų, tada kiekvienam šablonui sukuria suprojektuotą duomenų rinkinį, kuriuose vyksta analizė pagal dažnumo galią ir pan. SeqDim algoritmas, priešingai, iš pradžių ieško sekų šablonų, tada kiekvienai sekai projektuoja duomenų rinkinį, kur ieškoma daugiamačių šablonų. [PHP+01]

Mokymo sistemų vartotojų veiksmų analizei yra pasiūlytas praplėsto Hirate-Yamana algoritmo, randančio bei sugrupuojančio visas laiko atžvilgiu pratęstas sekas, ir SeqDim algoritmo junginys. [FNM08] MobilePrefixSpan algoritmas yra skirtas mobilių vartotojų judėjimo šablonams rasti. Dar kiti algoritmai yra skirti specifiškai interneto puslapių vartotojų veiksmų šablonams ar hierarchiniams sąryšiams analizuoti. [PLL+10]

[PHP+01] pateikti algoritmų variantai (UniSeq, DimSeq, SeqDim), naudojantys PrefixSpan algoritmą bei kitus algoritmus (BUC, H-Cubing ir pan.) daugiamačiams duomenų rinkiniams analizuoti.

Vienas naujausių pristatytų (2011 metų antroje pusėje) daugiamačių sekų šablonų analizės algoritmų yra [HZR11a] straipsnyje pateiktas Seq-Cmp algoritmas, naudojantis H-Forest duomenų struktūrą. Tačiau šiame straipsnyje pasigesta tikslesnių pateiktų duomenų struktūrų aprašymų, todėl algoritmo įgyvendinimas nėra trivialus. Kitame tų pačių autorių straipsnyje [HZR11b] pateiktas algoritmas ExtSeq-MIDim yra skirtas būtent žiniatinklio puslapių dažniems apsilankymų šablonams analizuoti.

2.8. Algoritmų kombinacijos

Daugiamačių sekų šablonams rasti taip pat galima naudoti ir įvairias algoritmų kombinacijas. Tokie metodai remiasi idėja, kad pagrindinė (sekų) duomenų rinkinio dimensija yra analizuojama sekų šablonų analizės metodų (pvz., PrefixSpan) pagalba, o likusios dimensijų reikšmės yra saugomos ir analizuojamos, pvz., Apriori algoritmo pagalba.

Teoriškai galima kombinuoti įvairius algoritmus: PrefixSpan + Apriori, FreeSpan + Apriori ir t.t. Tačiau skirsis galutinių rezultatų gavimo efektyvumas, nes vienamačių algoritmų savybės pereina į daugiamatį algoritmą. Taip pat yra reikalingos papildomos algoritmų modifikacijos, kad būtų išsaugotas dimensijų tarpusavio ryšys ir t.t.

3. MD-PS-FPG algoritmas ir jo įgyvendinimas

Vienas šio darbo tikslų buvo įgyvendinti MD-PS-FPG algoritmą, kuris paremtas modifikuotų PrefixSpan bei FP-Growth algoritmų kombinacija. Pasirinkta programavimo kalba – C#. Programuota Microsoft Visual Studio 2008 aplinkoje su .NET 3.5 komponentu (angl. *framework*). Testavimams skirti duomenų rinkiniai saugojami failinėje sistemoje arba Microsoft SQL Server 2008 duomenų bazėje.

3.1. FP-Tree duomenų struktūra

J. Han bei kiti [HPY+04] pristato medžio pavidalo duomenų struktūrą FP-Tree (angl. *Frequent-Pattern Tree*) – tai kompaktiška duomenų struktūra, skirta kritiniams (dažnai pasikartojantiems) duomenims saugoti.

FP-Tree duomenų struktūra yra sudaryta iš:

- šaknies (angl. *root*) „*null*“;
- elementų prefiksų pomedžių (EPP) aibės;
- dažnų aibių *header* lentelės.

Kiekvienas EPP mazgas turi:

- elemento pavadinimą (*name*) – nurodo, kuris tai elementas;
- transakcijų sumą (*count*) – skaičius, kiek transakcijų šią dalį bendrai dalinasi;
- mazgo nuorodą (*link*) – rodo į sekantį mazgą, kuris turi tokį patį elemento pavadinimą; jei tokio nėra – tada rodo į „*null*“ (ne į šaknį, o tiesiog į neapibrėžtą objektą).

Kiekvienas FP-Tree įrašas susideda iš:

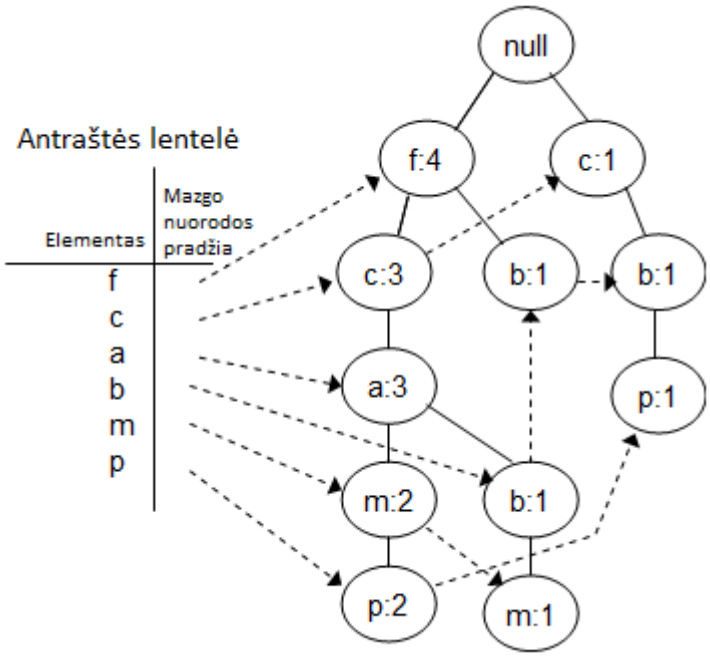
- elemento pavadinimo;
- mazgo nuorodos pradžios (*head*) – nuoroda į pirmąjį FP-Tree mazgą, kuris turi atitinkamą elemento pavadinimą.

3.1.1 pav. pateiktas svetainės X puslapių naršymo transakcijų duomenų rinkinio pavyzdys su paruoštais (išrūšiuotais) dažnų elementų duomenimis FP-Tree konstravimui (visi ne dažni elementai yra atmesti). Šiuo atveju minimali aibės galia yra lygi 3.

TId	Naršymo seka	Dažni elementai (išrūšiuoti)
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, l, m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

3.1.1 pav. Svetainės puslapių naršymo transakcijų pavyzdys su išrūšiuotais dažnais elementais

3.1.2 pav. pateiktas vizualus FP-Tree pavyzdys, sudarytas iš aukšiau pateikto svetainės puslapių naršymo transakcijų pavyzdžio. Reikia pastebėti, kad čia lankyti puslapiai nesikartoja, tačiau galima į šį pavyzdį žiūrėti kaip į unikalių puslapių lankymą per vartotoją, jo grupę, miestą ir kitus parametrus. Yra pasiūlyta FP-Tree patobulinimų, kurie tinka sekoms (STMFP [SSS+08], mWAP [PG07] ir pan.), tačiau pats FP-Tree yra tinkamas, pvz., saugoti dimensijų reikšmėms (vartotojų grupėms, miestams ir pan.), o pačias sekas galima analizuoti kitais algoritmais (PrefixSpan ar pan.). Kiek modifikavus algoritmus ir juos apjungus, atsiranda galimybė analizuoti daugiamatę informaciją, neprarandant dimensijų tarpusavio informacijos.



3.1.2 pav. FP-Tree vizualus pavyzdys

3.2. FP-Tree konstravimas

Pirmuoju duomenų rinkinio skanavimu yra randami dažni aibių elementai ir jų galios. Elementai yra išrūšiuojami mažėjančia tvarka galios atžvilgiu. Sukuriama FP-Tree medžio šaknis

ir ji pažymima „*null*“. Antruoju duomenų rinkinio skanavimu rekursiškai suterpiami dažni elementai, kiekvieno pasikartojančio FP-Tree prefikso mazgo transakcijų sumos reikšmę padidinant vienetu, t.y., jei mazgas su tokiu elemento pavadinimu šakoje (prefikse) jau egzistuoja, tai padidinama jo *count* reikšmė, priešingu atveju yra sukuriamas naujas mazgas (su *count* = 1).

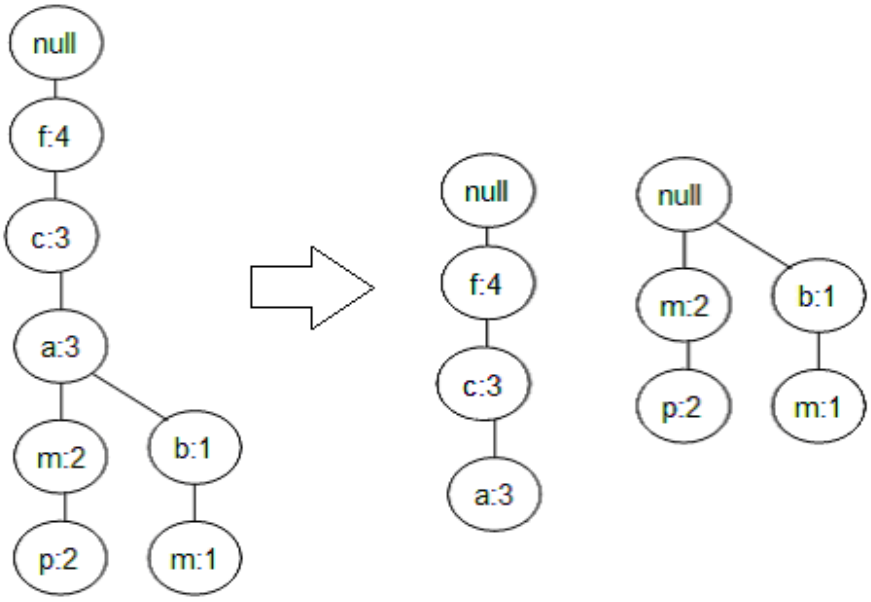
Kuriant naują mazgą (įterpiant naują elementą) taip pat yra saugomas jo pavadinimas bei mazgo nuoroda. Be to, yra konstruojama *header* lentelė su dažnų elementų mazgų nuorodų pradžiomis.

3.3. FP-Growth algoritmas

[HPY+04] straipsnyje taip pat pateiktas FP-Growth algoritmas, kuris yra skirtas FP-Tree duomenų struktūroje saugojamiems duomenims analizuoti. Šio algoritmo rezultatas yra visų dažnų elementų šablonų aibė.

FP-Tree elemento visų prefiksų kelių (angl. *path*) aibė yra vadinama sąlyginių šablonų baze (angl. *conditional pattern-base*). Pvz., pateikto svetainės X puslapių naršymo transakcijų pavyzdžio elemento p sąlyginių šablonų bazė yra sudaryta iš dviejų prefiksų kelių: {f c a m} ir {c b}.

Jei FP-Tree turi atkarpą, kuri sudaryta tik iš vieno prefikso kelio, kuris tęsiasi nuo šaknies iki pirmojo išsišakojančio mazgo (turinčio daugiau nei vieną vaiką), tai toks FP-Tree medis yra vadinamas vieno prefikso kelio FP-Tree.



3.3.1 pav. Vieno prefikso kelio FP-Tree ir jo išskaidymas

Jei FP-Tree yra vieno prefikso kelio FP-Tree, tai FP-Growth algoritme jis skaidomas į dvi dalis: į vieno prefikso kelio dalį ir į likusią dalį, kurios pirmasis mazgas yra pažymimas kaip „*null*“ šaknis. Vieno prefikso kelio FP-Tree (jei toks yra) visų mazgų, kurie tenkina minimalios aibės galios taisyklę, kombinacijos tarpusavyje yra pridedamos į rezultatų aibę. Kiekvienam likusiosios FP-Tree dalies mazgui yra generuojamas naujas FP-Tree iš sąlyginių šablonų bazės ir, jei toks FP-Tree yra sudarytas ne vien iš šaknies, rekursiškai atliekami aukščiau minėti veiksmai. Tokiu būdu gaunama visų duomenų rinkinio dažnų elementų šablonų aibė.

3.4. FP-Tree ir FP-Growth įgyvendinimas

Programuojant FP-Tree duomenų struktūrą buvo įgyvendintas [HPY+04] straipsnyje pateiktas FP-Tree medžio konstravimo algoritmas, naudojant apsirašytas klases *Node* bei *FPTree*.

Id	Transaction
1	Facdgimp
2	Abcflmo
3	Bfhjo
4	Bcksp
5	Afcelpmn

3.4.1 pav. Duomenų bazės lentelės *Transaction* duomenys

Straipsnyje visus veiksmus, susijusius su elemento pavadinimu, siūloma atlikti su simbolių eilutėmis, tačiau vietoje elemento pavadinimo (*name*) buvo pasirinkta naudoti tiesiog elemento Id – transakcijos saugomos ne kaip simbolių eilutės (*string*), o sveikųjų skaičių tipu *int*.

Straipsnyje aprašytas algoritmas rūšiuoja elementus pagal jų galias, tačiau nėra tiksliai pasakyta, kaip turi išsidėstyti elementai, jei jų galios yra vienodos. Neatsižvelgiant į šį niuansą, norimas gauti rezultatas gali būti visiškai kitoks, nei tikimasi (nebus maksimaliai išnaudojama prefiksų savybė). Įgyvendinant algoritmą tokiems atvejams buvo pasirinktas rūšiavimas pagal elemento Id, dėl ko šiek tiek skiriasi galutinis sukurtos programos rezultatas nuo straipsnyje pateikto pavyzdžio rezultato (kaip gauti rezultatą, pateiktą straipsnyje – deja, lieka neaišku).

Taip pat buvo įgyvendintas straipsnyje aprašytas algoritmas FP-Growth. Sukurta klasė *FPGrowth*, įgyvendinanti algoritmą, bei prie pagrindinės FPTree klasės pridėta procedūra *addPrefixPath*, padedanti konstruoti FP-Tree pagal prefikso kelią iš sąlyginių šablonų bazės.

3.5. MD-PS-FPG algoritmas ir jo įgyvendinimas

Kadangi PrefixSpan algoritmas yra vienas efektyviausių mažesnių dimensijų sekų duomenų rinkiniams analizuoti, o FP-Growth algoritmas tinka ne pagrindinėms dimensijoms analizuoti (įskaitant kompaktišką duomenų struktūrą FP-Tree, skirtą joms saugoti), tai siekiamam tikslui – daugiamačių sekų dažnų posekių šablonų radimui – pasiekti buvo panaudoti būtent šių algoritmų modifikacijų kombinacija.

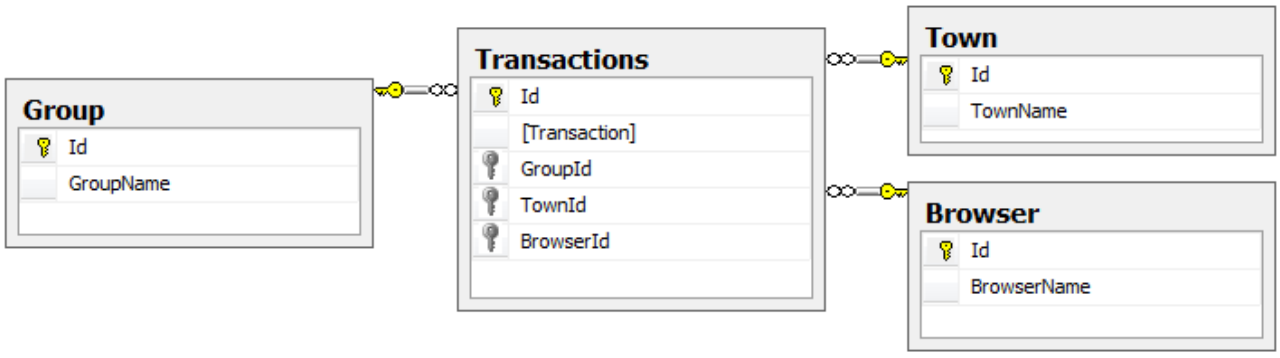
MD-PS-FPG (angl. *Multi-dimensional Prefix-projected Sequential pattern mining with Frequent Pattern Growth)* algoritmo pagrindinė idėja:

- kiekvienam PrefixSpan sugeneruotam projekcijos duomenų rinkiniui kuriama FP-Tree duomenų struktūra, sauganti likusiųjų dimensijų reikšmes;
- kai randamas dažnas posekis, jo FP-Tree medyje saugomos reikšmės yra analizuojamos FP-Growth algoritmu ir taip randami bendri (visų dimensijų) dažni posekiai.

Įgyvendinant MD-PS-FPG algoritmą, buvo sukurti du Microsoft Visual Studio 2008 projektai: MFPGrowth (anksčiau aprašyto FP-Growth algoritmo biblioteka) bei jį naudojantis konsolinis (angl. *console*) MPrefixSpan (modifikuoto PrefixSpan algoritmo) projektas.

Transakcijų duomenų rinkinio nuskaitymui sukurtas atskiras programinis sluoksnis, suteikiantis galimybę pasirinkti skirtingas duomenų rinkinio nuskaitymo strategijas. Viena jų (*MDSequenceFileDatabase*) yra skirta duomenų nuskaitymui iš tekstinio failo, kita (*MDSequenceSQLDatabase*) – iš Microsoft SQL Server 2008 duomenų bazės. Norint algoritmą pritaikyti kitokiam duomenų rinkinio tipui (skirtingos struktūros, skirtingo duomenų bazės serverio ir pan.) pakanka sukurti vaikinę klasę iš *MDSequenceDatabase* klasės ir įgyvendinti jos abstrakčius metodus.

Testinių duomenų (transakcijų), atitinkančių pateiktus 2.1 pav., saugojimui buvo sukurta Microsoft SQL Server 2008 duomenų bazė, kurios struktūra pateikta 3.5.1 pav.



3.5.1 pav. Duomenų bazės struktūra

2.1 pav. *TId* stulpelis atitinka duomenų bazės *Transactions* lentos *Id* lauką (*Transactions.Id*), *Grupė – Group.GroupName* lauką, *Miestas – Town.TownName* lauką, *Naršyklė – Browser.BrowserName* lauką, *Naršymo seka – Transactions.[Transaction]* lauką.

Reikiamų laukų paėmimui buvo sukurta SQL procedūra *GetTransactionData*, grąžinanti tik tolimesnei analizei reikalingus duomenis. Laukų pavadinimai ir jų kiekis – nesvarbus, išskyrus vieną lauką – *SequenceColumn*, kuris yra privalomas ir kuris nurodo pagrindinės dimensijos lauko pavadinimą (likę laukai laikomi ne pagrindinėmis dimensijomis). Kadangi algoritme operuojama sveikųjų skaičių tipu, o realūs duomenys yra simbolių eilutės, tai sąsajai (kartu ir rezultatų atvaizdavimui) yra naudojamos transakcijų nuskaitymo strategijose aprašytos procedūros *itemToId* ir *itemToString*.

Atlikus daugiau testavimų, buvo pastebėta, kad lėčiausiai veikia projekcijų duomenų rinkinių generavimas (užimdavo vidutiniškai apie 70% viso algoritmo laiko), kadangi tam reikalinga kiekvieno dažno posekio paieška tarp visų transakcijų. Todėl algoritmas buvo patobulintas, kiekvienam posekiui saugant sekų Id sąrašą, į kurias sekas tas posekis įeina. Tokiu būdu sekos, reikalingos projekcijų duomenų rinkiniams, yra iškart sužinomos vieną sykį nuskaičius visas transakcijas, taip projekcijų duomenų rinkinius galima generuoti daug efektyviau. Taip pat iš FP-Growth algoritmui paduodamo transakcijų sąrašo buvo išimtos transakcijos, kurios neturi elementų, kai kurie pasikartojantys skaičiavimai iškelti į vieną vietą ir pan.

3.6. Rezultatai

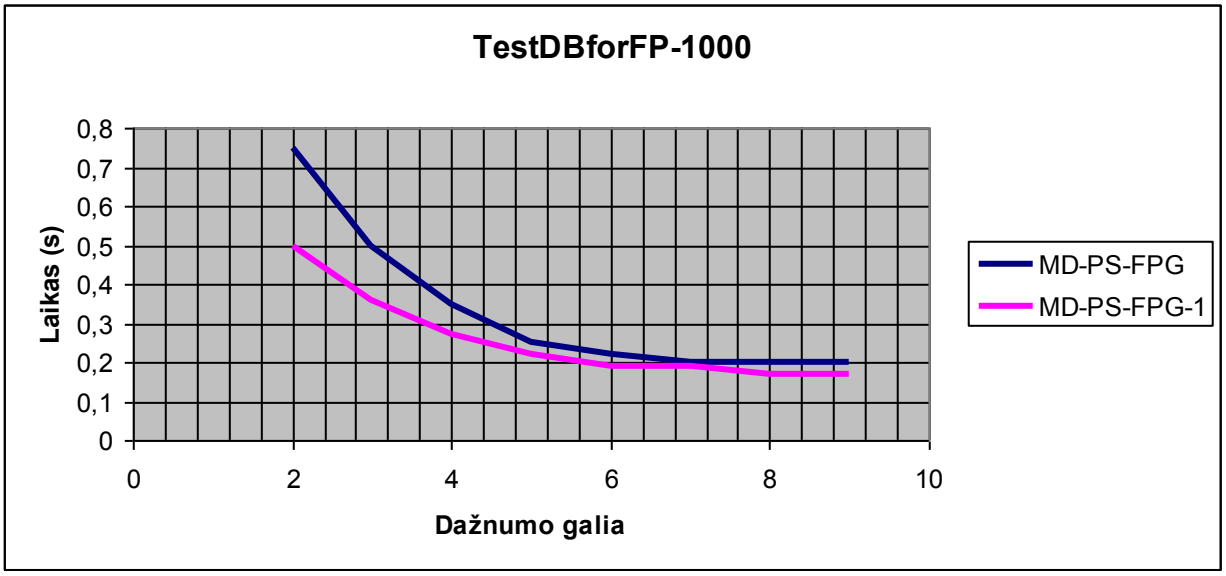
```
pattern:  [ 85 ]      << c )( c )( p >>
pattern:  [ 46 ]      << c )( c f )( l >>
pattern:  [ 46 7 ]    << c )( c f )( l >>
pattern:  [ 7 ]       << c )( c f )( l >>
pattern:  [ 46 ]      << c )( c )( l >>
pattern:  [ 46 7 ]    << c )( c )( l >>
pattern:  [ 7 ]       << c )( c )( l >>
pattern:  [ 46 ]      << c )( f )( l >>
pattern:  [ 46 7 ]    << c )( f )( l >>
pattern:  [ 7 ]       << c )( f )( l >>
pattern:  [ 46 ]      << b )( a f )( a >>
pattern:  [ 46 ]      << b )( a )( a >>
pattern:  [ 46 ]      << b )( f )( a >>

Execution time ~ 0,2640151 s

Total time ~ 0,6080348 s

Frequent sequences count : 267
```

3.6.1 pav. MD-PS-FPG algoritmo dalinio rezultato pavyzdys

3.6.1 pav. pateiktas pavyzdys su dalimi MD-PS-FPG algoritmo rezultato. Nors įvairioje literatūroje yra pateikta nemažai algoritmų, skirtų daugiamačiams duomenų rinkiniams analizuoti, tačiau veikiančių programiškai įgyvendintų algoritmų rasti nepavyko. Todėl

testavimai buvo atlikti su pirmine MD-PS-FPG bei patobulinta MD-PS-FPG-1 algoritmo versija. Testavimai buvo atlikti su įvairiais kiekiais duomenų ir įvairiomis dažnumo galiomis. Duomenys sugeneruoti naudojantis Red Gate SQL Data Generator 2 duomenų generatoriumi, pagrindinės dimensijos reikšmėms naudojant nuo 1 iki 5 posekių sekoje, kiekvienam posekiui turint nuo 1 iki 3 elementų, o kitų dimensijų reikšmes – intervale nuo 1 iki 99. Pagrindiniams testavimams buvo naudojamos duomenų bazės: TestDBforFP-1000 (1000 transakcijų), TestDBforFP-10000 (10000 transakcijų), TestDBforFP-50000 (50000 transakcijų) ir TestDBforFP-100000 (100000 transakcijų).

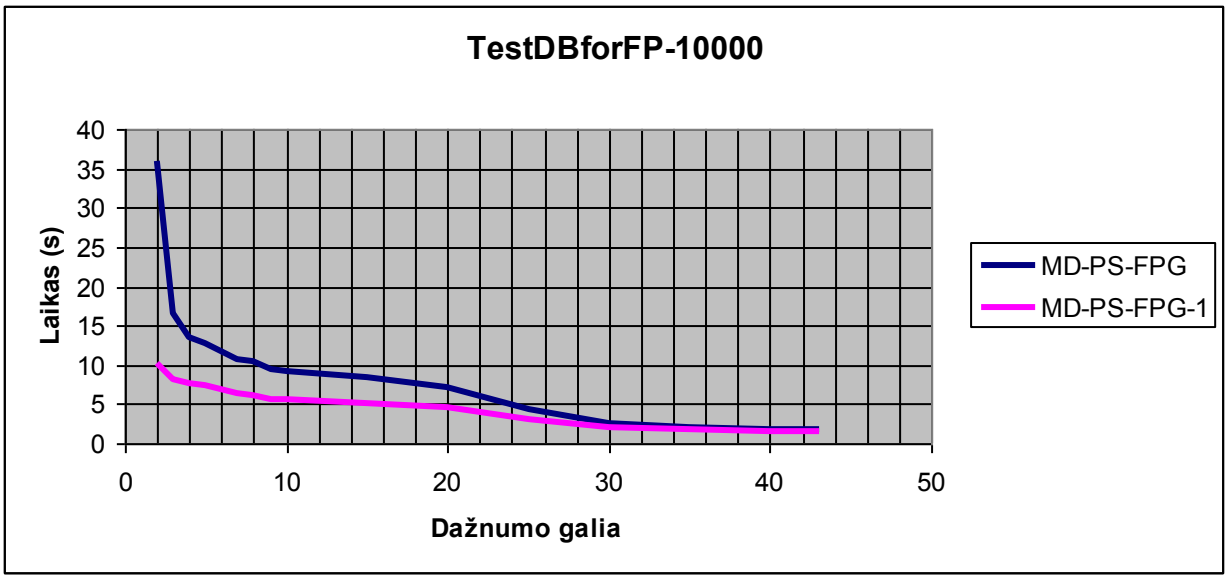
Testavimai buvo atliekami su tokiomis dažnumo galiomis, kai rezultate yra randamas bent vienas dažnas šablonas. Minimali sekos dažnumo galia – 2 (kadangi mažesnės neturi praktinės prasmės). Testavimai atlikti Intel Core 2 Duo 2.26 GHz kompiuteriu su 3 GB operatyviosios atminties (RAM).



3.6.2 pav. MD-PS-FPG ir MD-PS-FPG-1 algoritmų testavimo rezultatai su TestDBforFP-1000 duomenų bazės transakcijomis

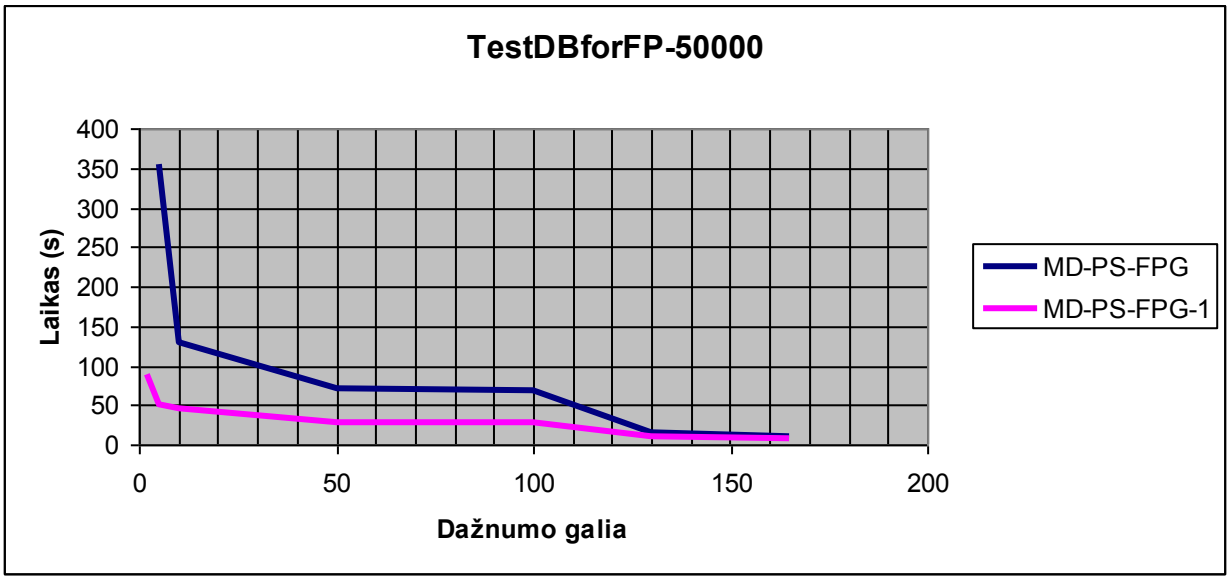
3.6.2 pav. pateikti testavimų su TestDBforFP-1000 duomenų bazės transakcijomis pradinės MD-PS-FPG algoritmo versijos ir patobulintos MD-PS-FPG-1 algoritmo versijos rezultatai laiko atžvilgiu. Rezultatuose atvaizduotas laikas yra grynasis algoritmo veikimo laikas (t.y., į jį nėra įtrauktas duomenų skaitymas iš duomenų bazės).

Abi algoritmo versijos veikia gan greitai, tačiau dažnumo galiai esant lygiai 2, patobulintasis algoritmas veikia apytiksliai pusantro karto greičiau už pradinį algoritmo variantą.



3.6.3 pav. MD-PS-FPG ir MD-PS-FPG-1 algoritmų testavimo rezultatai su TestDBforFP-10000 duomenų bazės transakcijomis

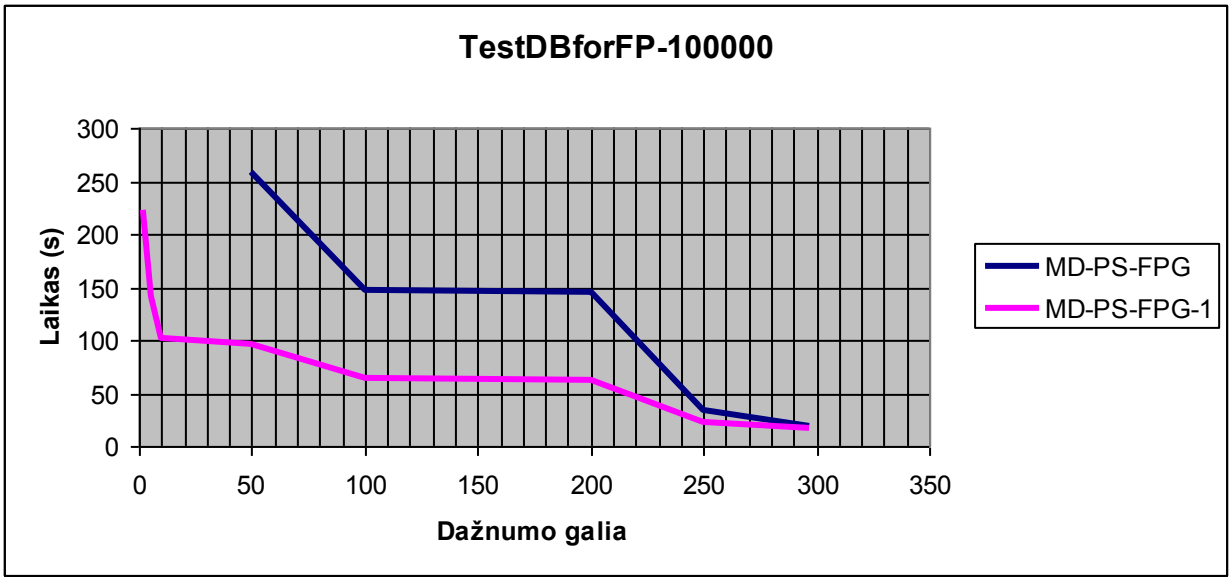
3.6.3 pav. pateikti testavimų su TestDBforFP-10000 duomenų bazės transakcijomis rezultatai. Dažnumo galiai esant didesnei už 15, algoritmai veikia daugmaž vienodai ilgai, tačiau kai dažnumo galia yra mažesnė – patobulintasis algoritmas veikia nuo apytiksliai pusantro karto iki pusketvirto karto greičiau už pradinį algoritmo variantą.



3.6.4 pav. MD-PS-FPG ir MD-PS-FPG-1 algoritmų testavimo rezultatai su TestDBforFP-50000 duomenų bazės transakcijomis

3.6.4 pav. pateikti testavimų su TestDBforFP-50000 duomenų bazės transakcijomis rezultatai. Dažnumo galiai esant didesnei už 130, algoritmai veikia daugmaž vienodai ilgai, tačiau kai dažnumo galia yra mažesnė – patobulintasis algoritmas veikia nuo apytiksliai pusantro

karto iki daugiau nei septynių kartų greičiau už pradinį algoritmo variantą. Pradinė algoritmo versija su dažnumo galia lygia 2 veikė ilgiau nei 7 minutes, todėl jos veikimas buvo nutrauktas ir rezultato nebelaukta. Patobulintoji versija su dažnumo galia lygia 2 veikė beveik taip pat ilgai kaip pradinė versija su dažnumo galia lygia 50.



3.6.5 pav. MD-PS-FPG ir MD-PS-FPG-1 algoritmų testavimo rezultatai su TestDBforFP-100000 duomenų bazės transakcijomis

3.6.5 pav. pateikti testavimų su TestDBforFP-100000 duomenų bazės transakcijomis rezultatai. Dažnumo galiai esant didesnei už 250, algoritmai veikia daugmaž vienodai ilgai, tačiau kai dažnumo galia yra mažesnė – patobulintasis algoritmas veikia bent kelis kartus greičiau. Pradinė algoritmo versija veikė ilgiau nei 7 minutes su dažnumo galia mažesne už 30, taigi veikimas tokiais atvejais taip pat buvo nutrauktas. Patobulintasis algoritmas iš 100000 transakcijų atrinko šablonus su dažnumo galia lygia 2 per 223,08 sekundės.

3.7. Tolimesni darbai

Šiame darbe pristatyta praktiškai įgyvendinta FP-Tree duomenų struktūra bei ją analizuojantis FP-Growth algoritmas ir šio algoritmo bei PrefixSpan algoritmo modifikacijų kombinacija MD-PS-FPG, skirta daugiamačių sekų šablonų analizei. Testavimai buvo atlikti su generuotais duomenimis, nesistengiant atkurti tam tikros realios situacijos ar pan.

Vienas iš siektinų ateities tikslų yra įgyvendinti (ar kitu būdu gauti) kai kuriuos kitus šiame darbe aprašytus algoritmus ir palyginti jų efektyvumą su MD-PS-FPG algoritmu. Taip pat būtų gerai išbandyti įgyvendintą algoritmą su duomenų rinkiniais, kurie būtų paimti iš realių sistemų,

arba susigeneruoti kiek įmanoma įvairesnius ir daugiau ar mažiau realią sistemą galinčius atitikti duomenis.

Taip pat siektina toliau tobulinti suprogramuotą MD-PS-FPG versiją, atsižvelgiant į tai, kad duomenų kiekiai gali būti daug kartų didesni nei testuota iki šiol, netilpti į atmintį ir pan. Reikėtų modifikuoti algoritmą, kad jis mokėtų saugoti jau apdorotas duomenų rinkinio transakcijas failinėje sistemoje ar pan., bet kartu ir mokėtų jas panaudoti bendroje visų transakcijų analizėje. Galbūt reikėtų dalį logikos iškelti į duomenų bazės procedūras, kurios galėtų atlikti dalinį duomenų apdorojimą, kas sumažintų reikiamą saugoti duomenų kiekį.

Turint kitokią duomenų rinkinio struktūrą, reikės įgyvendinti naują duomenų (transakcijų) nuskaitymo strategiją. Dar vienas iš siektinų tikslų yra tvarkingesnis rezultato išvedimas: būtų galima patobulinti atvaizdavimą, rodant ne dimensijos *Id* lauką, o dimensijų pavadinimus; priklausomai nuo tam tikro duomenų kiekio rėžio ar tiesiog parametro išvedimas galėtų būti saugomas į failą ar pan. Taip pat gali iškilti kitų nenumatytų atvejų, kai teks modifikuoti su tuo susijusį jau esamą programinį kodą.

Išvados ir rekomendacijos

Dideliems duomenų kiekiams analizuoti yra sukurta įvairių algoritmų, tačiau nėra universalių, kurie tiktų bet kokiems duomenims be išimties. Šiame darbe buvo pristatytas MD-PS-FPG algoritmas – modifikuotų FP-Growth (kompaktišką duomenų struktūrą FP-Tree galinčio analizuoti) ir PrefixSpan algoritmų kombinacija, kuri yra vienas variantų, kaip galima analizuoti daugiamačių sekų duomenų rinkinius.

Pirmojoje darbo dalyje buvo pateikti asociatyvumo taisyklių analizės apibrėžimai bei sekų šablonų analizės algoritmų tipizacija. Taip pat apžvelgti populiariausi mažesnių dimensijų aibių bei sekų šablonų analizės algoritmai.

Antrojoje darbo dalyje pateikti kai kurie populiaresni didesnių dimensijų duomenų rinkinių analizės metodai.

Trečiojoje darbo dalyje aprašytas praktinis MD-PS-FPG algoritmo įgyvendinimas, atliktos modifikacijos bei papildomai atlikti darbai, pateikti kai kurių atliktų testavimų rezultatai, FP-Growth algoritmo rezultatų palyginimas su straipsnyje esamais pavyzdžiais, skirtumų priežastys ir pan.

Ateities planuose numatyta toliau tobulinti MD-PS-FPG algoritmą bei palyginti jį su kitais įvairioje literatūroje aprašytais algoritmais, taip pat atlikti daugiau testavimų su įvairesniais, kiek įmanoma realias sistemas atitinkančiais, duomenimis.

Literatūros sąrašas

[AGY+02] J. Ayres, J. Gehrke, T. Yiu, J. Flannick, Sequential PAttern Mining using A Bitmap Representation, 2002.

[AIS93] R. Agrawal, T. Imielinski, A. Swami, Mining Association Rules Between Sets of Items in Large Databases, 1993, p. 207-216.

[AS94] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, 1994.

[AS95] R. Agrawal, R. Srikant, Mining sequential patterns, 1995, p. 3-14.

[CH05] Y. Chen, T. Huang, Discovering Fuzzy Time-Interval Sequential Patterns in Sequence Databases, 2005.

[CH06] Y. Chen, T. Huang, A new approach for discovering fuzzy quantitative sequential patterns in sequence databases, 2006.

[CWC04] D. Chiu, Y. Wu, A. L. P. Chen, An Efficient Algorithm for mining Frequent Sequences by a New Strategy without Support Count, 2004.

[DGJ11] V. Dagienė, G. Grigas, T. Jevsikova, Enciklopedinis kompiuterijos žodynas, URL: http://www.likit.lt/term/enc.html. 2012.05.26.

[YC05] C. Yu, Y. Chen, Mining Sequential Patterns from Multidimensional Sequence Data, 2005.

[FNM08] P. Fournier-Viger, R. Nkambou, E. Mephu Nguifo, A Knowledge Discovery Framework for Learning Task Models from User Interactions in Intelligent Tutoring Systems, 2008.

[HHY+09] Y. Hu, T. Huang, H. Yang, Y. Chen, On mining multi-time-interval sequential patterns, 2009.

[HPY+04] J. Han, J. Pei, Y. Yin, R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, 2004.

[HPY05] J. Han, J. Pei, X. Yan, Sequential Pattern Mining by Pattern-Growth: Principles and Extensions, 2005, p. 136-140.

[HPM+00] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M. Hsu, FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining, 2000.

[Hua10] T. Huang, Knowledge gathering of fuzzy multi-time-interval sequential patterns, 2010.

[Hua95] T. Huang, A Fuzzy Mining Process for Discovering Sequential Patterns, 1995.

[HZR11a] G. Huang, N. Zuo, J. Ren, A New Compound Method of Multi-dimensional Sequential Pattern Mining, 2011.

[HZR11b] G. Huang, N. Zuo, J. Ren, Mining Web Frequent Multi-dimensional Sequential Patterns, 2011.

[MCP98] F. Masseglia, F. Cathala, P. Poncelet, The PSP approach for mining sequential patterns, 1998.

[HZC07] K. Hu, C. Zhang, L. Chen, A SCALABLE METHOD OF MINING APPROXIMATE MULTIDIMENSIONAL SEQUENTIAL PATTERNS ON DISTRIBUTED SYSTEMTS, 2007.

[PG07] J. Parmar, S. Garg, Modified Web Access Pattern (mWAP) Approach for Sequential Pattern Mining, 2007.

[PHM+04] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, Mei-Chun Hsu, Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach, 2004.

[PHP+01] H. Pinto, J. Han, J. Pei, K. Wang, Multidimensional Sequential Pattern Mining, 2001.

[Pin98] H. Pinto, MULTI-DIMENSIONAL SEQUENTIAL PATTERN MINING, 1998, p. 3-28, 32-35, 40-41.

[PLL+10] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, Y. W. Choong, Mining Multidimensional and Multilevel Sequential Patterns, 2010.

[SSS+08] Y. Sui, F. Shao, R. Sun, J. Wang, A Sequential Pattern Mining Algorithm Based on Improved FP-tree, 2008.

[Zak01] M. J. Zaki, SPADE: An Efficient Algorithm for Mining Frequent Sequences, 2001, p. 31-60.