MATEMATIKOS IR INFORMATIKOS INSTITUTAS
VYTAUTO DIDŽIOJO UNIVERSITETAS

Romanas Tumasonis

DAŽNŲ SEKŲ PAIEŠKA DIDELIUOSE DUOMENŲ MASYVUOSE

Daktaro disertacija

Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, Sistemų teorija (P 175)

Vilnius, 2007

Disertacija rengta 2002–2006 metais Matematikos ir informatikos institute.

Doktorantūros teisė suteikta kartu su Vytauto Didžiojo universitetu 2004 m. gruodžio 13 d. Lietuvos Respublikos Vyriausybės nutarimu Nr. 1285.

Mokslinis vadovas:

prof. habil. dr. Gintautas Dzemyda (Matematikos ir informatikos institutas, fiziniai mokslai, informatika, 09 P).

TURINYS

Žodynėlis

Angliškai	*Lietuviškai*
data mining	Duomenų gavyba
knowledge discovery in databases	Žinių (iš)gavimas iš duomenų bazės
online analytical processing	Operatyvus analitinis duomenų apdorojimas
pattern	Šablonas
hidden knowledge	Paslėptos žinios
itemset	Rinkinys
support	Dažnumas
association rule	Susietumo taisyklė
confidence	Tikrumas
strong	Griežta (taisyklė)
lattice	Gardelė
semilattice	Pusinė gardelė
minimum support	Minimalus dažnumas
breadth-first search	Paieška platyn
depth-first search	Paieška gilyn

Žymėjimai ir trumpiniai

Žymėjimas	*Paaiškinimas*
GSP	*Generate Sequence Pattern* (Algoritmas)
$\sigma(X)$	Rinkinio *X* dažnumas
min_sup	Minimalus dažnumas
F_k	Aibė dažnų *k*-rinkinių
$A \Rightarrow B$	Susietumo taisyklė tarp rinkinio *A* ir *B*
min_conf	Minimalus tikrumas
$X \sqsubset Y$	Aibė *X* padengta *Y*
$\top$	Viršutinis elementas
$\perp$	Apatinis elementas
$\equiv$	Ekvivalentumo sąryšis
$\aleph_k$	Aibė visų pseudoklasių
$(\alpha_1 \mapsto \alpha_2 \mapsto ... \mapsto \alpha_q)$	Seka α
$T_1 \mapsto T_2 \mapsto ... \mapsto T_n$	Transakcijų sąrašas
Θ_k	Ekvivalentumo sąryšis pagal *k*
$[X]_{\Theta_k}$	Ekvivalentumo klasė, sukurta pagal ekvivalentumo sąryšį Θ_k
$\alpha \subset \beta$	α yra sekos β būdingas posekis
ProMFS	Naujai pateiktas algoritmas (*probabilistic algorithm for mining frequent sequences*)
$\widetilde{C}$	Modelinė seka
$P(i_j) = \frac{V(i_j)}{VS}$	Elemento i_j pasirodymo tikimybė
$P(i_j \mid i_v)$	Sąlyginė tikimybė, kad elementas i_v pasirodys po elemento i_j
$D(i_j \mid i_v)$	Atstumas tarp elemento i_j ir i_v
$\widehat{A}$	Atstumų vidurkių matrica
$Q(i_j, c_r)$	Skaitinė charakteristika (matrica)
$\widehat{F}$	Dažniausių atstumų matrica
DNR	Deoksiribonukleorūgštis

1. ĮVADAS

1.1. Tyrimo objektas

Disertacijos tyrimo objektas yra duomenų gavybos (angl. *data mining*) technologijos algoritmai, skirti nustatyti dažnus fragmentus. Nagrinėjamas atskiras duomenų gavybos atvejis, kai duomenys yra paprasti simboliai. Disertacijoje aprašytuose tyrimuose duomenys yra didelės apimties tekstiniai failai, kuriuose bus ieškoma dažnų sekų.

1.2. Naujumas

Disertacijoje sukurtas, išnagrinėtas ir realizuotas naujas tikimybinis algoritmas, kuris remiasi elementų pasirodymo duomenyse tikimybinėmis charakteristikomis. Šis algoritmas yra apytikslis, tačiau veikia žymiai greičiau nei tikslieji algoritmai. Tokiu būdu jis yra naudojamas tais atvejais, kai dėl ženkliai didesnio pakankamai gerų rezultatų gavimo greičio, galime paaukoti tikslumą.

1.3. Tyrimo tikslai ir uždaviniai

Pagrindinis mokslinio tyrimo tikslas yra išnagrinėti kelis populiariausius dažnų sekų nustatymo algoritmus ir juos modifikuoti, o taip pat sukurti naują algoritmą specialiems uždaviniams spręsti. Tam tikrais atvejais tenka „aukoti“ tikslumą tam, kad gautume kuo greičiau pakankamai gerus rezultatus. Tokiu būdu pagrindinis disertacijos tikslas yra pasiūlyti apytikslį, bet greitą algoritmą, kurio pagalba galima būtų surasti dažnas sekas bei nustatyti tų sekų išsamumą.

Siekiant įgyvendinti suformuluotus tikslus, reikia išspręsti tokius uždavinius:

1) išnagrinėti dažniausiai naudojamus dažnų sekų nustatymo būdus bei algoritmus;

2) realizuoti GSP (*Generate Sequence Pattern*) algoritmą ir išanalizuoti jo veikimą su testiniais duomenimis;

3) modifikuoti realizuotą GSP algoritmą. Išanalizuoti šias modifikacijas su testiniais duomenimis;

4) patikrinti algoritmo modifikacijų greičio priklausomumą nuo duomenų dydžio, simbolių aibės duomenyse, o taip pat priklausomumą nuo tam tikrų dažnas sekas apsprendžiamų parametrų (pvz. minimalaus dažnumo);

5) sukurti ir realizuoti naują apytikslį algoritmą, kuris remiasi elementų pasirodymo duomenyse tikimybinėmis charakteristikomis;

6) palyginti šį algoritmą su tiksliuoju algoritmu (mūsų atveju modifikuotu GSP algoritmu);

7) išnagrinėti šio algoritmo tikslumo priklausomybę nuo įvairių pačio algoritmo ir duomenų charakteristikų;

8) realizuoti kitą naujojo algoritmo variantą ir išnagrinėti jo privalumus bei trukumus;

9) išnagrinėti realius duomenis su naujai pasiūlytu ir modifikuotu GSP algoritmu. Palyginti nagrinėjimo rezultatus.

1.4. Aktualumas

Pastaruoju metu, sparčiai vystantis informacinėms technologijoms bei greitoms ir efektyvioms duomenų saugojimo bei įrašymo formoms, visuomenę „užplūdo“ didžiuliai informacijos bei duomenų srautai iš pačių įvairiausių sričių. Dabar kiekvienos įmonės (komercinės, gamybinės, medicininės, mokslinės ir t.t.) veikloje vyksta įvairiausių įrašų registracija ir įvairiapusis informacijos apie įmonės veiklą saugojimas bei kaupimas. Be produktyvaus šios informacijos apdorojimo, šie duomenys su laiku gali pavirsti į visiškai beverčią „šiukšlių“ krūvą. Visi šie duomenys gali pasižymėti šiomis savybėmis:

- duomenys gali turėti neribotą dydį;
- duomenys gali būti įvairių tipų (loginiai, skaitiniai, tekstiniai);
- rezultatai turi būti konkretūs ir aiškūs;
- programiniai įrankiai, apdorojantys šiuos duomenis, turi būti paprasti naudojime.

Tokiu būdu labai aktualu rasti tarp šių didelių duomenų masyvų mums svarbią informaciją, kurią būtų galima panaudoti ateityje. Kitais žodžiais tariant, iš didelės masės duomenų reikia atrinkti informacijos „perlus“. Tai ir yra pagrindinis duomenų gavybos tikslas. Vienas iš svarbiausių jos tikslų yra dažnų pasikartojamumų radimas. Dar visai neseniai bet kuriai informacijos apdorojimo sistemai pakakdavo spręsti įvairius paieškos (surasti, kur ir kiek kartų pasikartoja nurodytas įrašas) arba statistinius uždavinius: koks yra vidutinis avaringumas (gimstamumas, nusikalstamumas) respublikoje, duotame rajone, per kažkokį laikotarpį ir t.t. Duomenų gavybos technologijos nagrinėja šiuos duomenys žymiai sudėtingiau ir pateikia gana detalius šios analizės rezultatus. Duomenų gavyba leidžia atsakyti į klausimus, kokie dažniausiai pasikartoja žodžiai tekste, kokių kriterijų visuma turi įtaką avaringumui (gimstamumui, nusikalstamumui) respublikoje, duotame rajone ar per kažkokį

laiko tarpą. Ne ką mažesnę svarbą duomenų gavyba turi medicinoje, nustatant žmogaus geno kodą ar kriterijus, pagal kuriuos žmonės serga viena ar kita liga. Taip pat duomenų gavyba yra populiari bankininkystėje, nustatant kokius kriterijus atitinka žmonės, kurie negali grąžinti paskolų. Prekybininkai irgi savo veikloje naudoja (arba gali naudoti) duomenų gavybos metodus, nustatant populiariausių prekių „krepšelius“, kurie buvo įsigyjami pirkėjų vieno apsipirkimo metu.

Galingos kompiuterinės sistemos, saugojančios ir apdorojančios didžiules duomenų bazes, tapo neatskiriamu veiklos atributu ne tik stambioms korporacijoms, bet ir smulkesnėms įmonėms. Reikia mokėti transformuoti „sausus“ duomenis (angl. *raw data*) į mums naudingą informaciją tam, kad galėtumėme priimti sau naudingus sprendimus. Tuo tikslu ir buvo sukurtos *Data mining* technologijos. Pramonėje ir versle labai aktualūs tapo šie klausimai:

- Kokias prekes pasiūlyti pirkėjui?
- Kokia tikimybė, kad duota pirkėjų grupė sureaguos į pateiktą reklamą?
- Ar galima pasirinkti optimalią strategiją, vykdant pirkimo-pardavimo operacijas vertybinių popierių biržoje?
- Ar gali bankas išduoti kreditą nurodytam klientui?
- Kokią diagnozę galima nustatyti duotam pacientui?
- Kaip prognozuoti didžiausią telefoninių ar elektros tinklų apkrovimą?
- Dėl ko atsiranda produkcijos brokas?

Atsakymai į šiuos ir daugelį kitų klausimų slypi duomenų bazių megabaituose. Paslėptų pasikartojamumų radimas, tarpusavio ryšiai tarp įvairių duomenų bazių įrašų, sudėtingų sistemų modeliavimas ir sudėtingų sistemų, priklausomų nuo jų elgsenos istorijos, analizė ir yra pagrindinis *data mining* tyrimų objektas.

Angliška terminą *data mining* galima būtų išversti kaip „duomenų gavyba“ arba „duomenų kasyba“. Tačiau nei vienas iš šių terminų nėra pakankamai išsamus ir iki galo neatspindi sąvokos prasmės. Dažnai su minėtu terminu yra siejamas terminas *žinių (iš)gavimas iš duomenų bazės* (angl. *knowledge discovery in databases*) ir *intelektinė duomenų analizė*. Šiuos terminus galima laikyti *data mining* sinonimais. Šių terminų atsiradimą sąlygojo nauji duomenų analizės metodai.

Tradicinė matematinė statistika ilgą laiką pretendavo į pagrindinį duomenų analizavimo instrumentą. Tačiau šiuo atveju ji tapo bejėge. Pagrindinė to priežastis – *vidurkinimo išrinkimo koncepcija*, pagal kurią visos operacijos yra vykdomos su fiktyviomis reikšmėmis (vidutinis pacientų skaičius ligoninėse, namo aukštų gatvėje vidurkis, vidutinė

paros temperatūra ir t.t.). Matematinės statistikos metodai buvo efektyvus tik tais atvejais, kai reikėjo patikrinti iš anksto suformuluotas hipotezes (angl. *verification-driven data mining*) arba atlikti „gilią“ žvalgybinę analizę, kurios pagrindą sudaro operatyvus analitinis duomenų apdorojimas (angl. *online analytical processing*, *OLAP*).

Šiuolaikinių *data mining* technologijų (angl. *discovery driven data mining*) pagrindą sudaro šablonai (angl. *pattern*), kuriuose atsispindi daugiafunkciniai duomenų tarpusavio santykiai. Šie šablonai savyje turi tam tikras duomenų išdėstymo taisykles, kurios gali būti lengvai suprantamos vartotojui. Šablonų paieška nesusieta su išankstinėmis duomenų struktūrų hipotezėmis. 1 lentelėje galima palyginti į kokius klausimus atsako statistiniai ir *data mining* metodai:

1 **lentelė**. Tradicinių ir *data mining* sprendžiamų klausimų palyginimas

Statistiniai metodai	*Data mining* metodai
Kokie vidutiniai traumatizmo atvejai pastebėti tarp rūkančių ir nerūkančių žmonių?	Ar yra tam tikri bendri bruožai tarp žmonių, kuriems yra padidintas traumatizmas?
Kokie vidutiniai sąskaitų dydžiai esamų klientų ir buvusių klientų (telefoninių paslaugų įmonėms)?	Ar yra tam tikri bendri požymiai apie klientus, kurie potencialiai gali atsisakyti telefoninės kompanijos paslaugų?
Kokia vidutinė pirkinių suma su pavogtomis kreditinėmis kortelėmis ir su nepavogtomis?	Ar egzistuoja stereotipiniai pirkinių požymiai, leidžiantys nustatyti sukčiavimo atvejus su kreditinėmis kortelėmis?

Vienas iš pagrindinių *data mining* aspektų yra netrivialūs šablonų paieškos metodai. Tai reiškia, kad atrasti šablonai yra neakivaizdūs, kad aptiktas netikėtas (angl. *unexpected*) reguliarumas duomenyse, kurie turi savyje paslėptas žinias (angl. *hidden knowledge*). Atsirado supratimas, kad tarp didelio kiekio „sausų“ duomenų yra dideli žinių sluoksniai, kuriuos ištyrinėjus galima rasti tikrus „deimantus“. *Data mining* – tai procesas, sugebantis atrasti „sausuose“ duomenyse šiuos šablonus. Labai dažnai visi tokio tipo uždaviniai suvedami į dažnų sekų paiešką didelėse duomenų bazėse.

1.5. Rezultatai

Disertacijoje buvo nagrinėjama dažnų sekų paieška dideliuose duomenų masyvuose. Buvo atlikti tokie darbai:

- Modifikuotas GSP dažnų sekų nustatymo algoritmas.
- Išnagrinėtas GSP algoritmo realizavimo būdas, panaudojant rekursinį dažnų sekų nustatymo būdą „gilyn“.
- Algoritmai palyginti pagal laiko ir atminties sąnaudas.
- Algoritmai palyginti pagal nagrinėjamos duomenų bazės pobūdį.
- Realizuotas naujas tikimybinis algoritmas ProMFS su dviem jo modifikacijomis: naudojant vidurkių matricą bei dažniausių atstumų matricą.
- Pateikti tikimybinio algoritmo su dviem modifikacijomis ir tikslaus GSP algoritmo palyginimai pagal laiko sąnaudas, tikslumą ir šių charakteristikų priklausomybę nuo failų dydžio bei elementų kiekį nagrinėjamame faile.
- Buvo išnagrinėti realūs duomenis su naujai pasiūlytu ir modifikuotu GSP algoritmu bei nagrinėjimo rezultatai.

1.6. Publikacijos:

Darbo rezultatai buvo publikuojami šiuose užsienyje leidžiamuose recenzuojamose leidiniuose:

R. Tumasonis, G. Dzemyda. *Analysis of Statistical Characteristics in Mining of Frequent Sequences*. Intelligent Information Processing and Data Mining. Proceedings of the International IIS: IIPWM’05 Conference. Gransk, Poland, June 13-16, 2005. Springer Berlin Heidelberg New York, ISSN 1615-3871, 2005, p. 377–387.

R. Tumasonis, G. Dzemyda. *The Probabilistic Algorihm for Mining Frequent Sequences*. Proceedings ADBIS’04 Eight East-European Conference on Advances in Databases and Information Systems, ISBN 963311358X, p. 89–98.

R. Tumasonis, G. Dzemyda. *The Statistical Characteristics in Probabilistic Algorithms for Mining Frequent Sequence*s. Datarzinatne. Serija 5, Sejums 22. Scientific Proceedings of the Riga Technical University, ISSN 1407-7493, Izdevnieciba "RTU", Riga, 2005, p. 85–94.

Darbo rezultatai buvo publikuojami šiame Lietuvos recenzuojamame leidinyje:

R. Tumasonis, G. Dzemyda. *Atmintis problema ieškant dažnų sekų didelėse duomenų bazėse*. Informacijos mokslai, Vilnius, 2003, p. 193–199.

Darbo rezultatai buvo publikuojami kitose Lietuvos leidiniuose:

R. Tumasonis, G. Dzemyda. *Analitiniai duomenų gavimo būdai šiuolaikinėse informacinėse sistemose*. Informacinės technologijos 2005: Aktualijos ir perspektyvos. IV mokslinė praktinė konferencija, ISBN 9955-9779-0-9, Alytus, 2005, p. 180–186.

R. Tumasonis, G. Dzemyda. *Statistinių charakteristikų analizė dažnų sekų paieškoje*. INFORMACINĖS TECHNOLOGIJOS 2005, Kauno technologijos universitetas, 2005, p. 108–114.

R. Tumasonis, G. Dzemyda. *Dažnų sekų paieškos tikimybinis algoritmas*. INFORMACINĖS TECHNOLOGIJOS 2004, Kauno technologijos universitetas, 2004, p. 14–20.

R. Tumasonis, G. Dzemyda. *Dažnų sekų nustatymo didelėse duomenų bazėse algoritmų analizė*. INFORMACINĖS TECHNOLOGIJOS 2003, Kauno technologijos universitetas, ISBN 9955-09-335-8, 2003, p. II-6-II-13.

Pranešimai buvo skaitomi šiose tarptautinėse konferencijose:

R. Tumasonis, G. Dzemyda. *The Probablistic Algorithm for Mining Frequent Sequences*. ADBIS'04 Eight East-European Conference on Advances in Databases and Information Systems. 2004 m. rugsėjo 22–25, Budapeštas, Vengrija.

R. Tumasonis, G. Dzemyda. *Analysis of the Statistical Charakteristics in Mining Frequent Sequences*. Intelligent Information Processing and Web Mining IIPWM'05 Conference. 2005 m. birželio 13–16, Gdanskas, Lenkija.

R. Tumasonis, G. Dzemyda. *The Statistical Characteristics in Probabilistic Algorithms for Mining Frequent Sequence*s. Thirteenth International Conference on Information Systems Development Methods and Tools, Theory and Practice. 2004 m. rugsėjo 9–11, Vilnius.

Pranešimai buvo skaitomi šiose respublikinėse konferencijose:

R. Tumasonis, G. Dzemyda. *Tikimybinių charakteristikų panaudojimas dažnų sekų paieškoje*. Lietuvos matematikų draugijos XLV konferencija. 2003 m. birželio 17–18, Kaunas, LŽŪU.

R. Tumasonis, G. Dzemyda. *Analitiniai duomenų gavimo būdai šiuolaikinėse informacinėse sistemose*. Informacinės technologijos 2005: aktualijos ir perspektyvos. IV mokslinė praktinė konferencija, 2005 m. balandžio 15–17, Alytus.

R. Tumasonis, G. Dzemyda. *Statistinių charakteristikų analizė dažnų sekų paieškoje*. INFORMACINĖS TECHNOLOGIJOS 2005, 2005 m. sausio 27–29, Kaunas, KTU.

R. Tumasonis, G. Dzemyda. *Dažnų sekų paieškos tikimybinis algoritmas*. INFORMACINĖS TECHNOLOGIJOS 2004, 2004 m. sausio 27–29, Kaunas, KTU.

R. Tumasonis, G. Dzemyda. *Dažnų sekų nustatymo didelėse duomenų bazėse algoritmų analizė*. INFORMACINĖS TECHNOLOGIJOS 2003, 2003 m. sausio 26–28, Kaunas, KTU.

R. Tumasonis, G. Dzemyda. *Atmintis problema ieškant dažnų sekų didelėse duomenų bazėse*. Vienuoliktoji mokslinė kompiuterininkų konferencija. 2003 m. rugpjūčio 28–30, Vilnius.

2. DAŽNŲ POAIBIŲ RADIMAS

2.1. Įvadas

Šiuo metu labai aktualu rasti tarp didelių duomenų masyvų mums svarbią informaciją, kurią būtų galima panaudoti ateityje. Tai ir yra pagrindinis duomenų gavybos tikslas. Vienas iš svarbiausių jos tikslų yra dažnų pasikartojamumų radimas. Dar visai neseniai bet kuriai informacijos apdorojimo sistemai pakakdavo spręsti įvairius paieškos (surasti, kur ir kiek kartų pasikartoja nurodytas įrašas) arba statistinius uždavinius: koks yra vidutinis avaringumas (gimstamumas, nusikalstamumas) respublikoje, duotame rajone, per kažkokį laikotarpį ir t.t. Duomenų gavybos technologijos nagrinėja šiuos duomenys žymiai sudėtingiau ir pateikia gana detalius šios analizės rezultatus. Duomenų gavyba leidžia atsakyti į klausimus, kokie žodžiai dažniausiai pasikartoja tekste, kokių kriterijų visuma turi įtaką avaringumui (gimstamumui, nusikalstamumui) respublikoje, duotame rajone ar per kažkokį laiko tarpą. Ne ką mažesnę svarbą duomenų gavyba turi medicinoje, nustatant žmogaus geno kodą ar kriterijus, pagal kuriuos žmonės serga viena ar kita liga. Taip pat duomenų gavyba yra populiari bankininkystėje, nustatant kokius kriterijus atitinka žmonės, kurie negali grąžinti paskolų. Prekybininkai irgi savo veikloje naudoja (arba gali naudoti) duomenų gavybos metodus, nustatant populiariausių prekių „krepšelius“, kurie buvo įsigyjami pirkėjų vieno apsipirkimo metu. Jeigu mūsų duomenys yra sudaryti iš tam tikrų aibės elementų, tai labai svarbu yra nustatyti dažnus tų elementų didžiausius poaibius duotoje duomenų bazėje.

2.2. Susietumo taisyklių nustatymas

Susietumo (asociacijos) nustatymo arba suradimo problema yra ta, kad reikia surasti aibę požymių ar atributų, kurie yra atskirti dideliu objektų rinkiniu duotoje duomenų bazėje. Jeigu mes peržiūrėtumėme komercinę knygyno duomenų bazę, kurioje objektai apibrėžia klientus, o atributai apibrėžia knygas ir autorius, tai šablonų radimo užduotis susiveda į radimą knygų, kurias pirkėjai dažniausiai pirko kartu, t.y. vieno apsipirkimo metu. Rezultatas galėtų būti toks: 40% žmonių, kurie pirko A. Diuma knygą „Trys muškietininkai“ kartu nusipirko ir V. Skoto knygą „Aivengas“. Parduotuvė galėtų pritaikyti šias gautas žinias reklamai, lentynų išdėstymui ir t.t. Yra labai daug technologinių sričių, kuriose galima naudoti pasikartojamumo radimo taisykles. Tokios sritys gali būti katalogų sudarymas, klientų segmentavimas, telekomunikacinių pavojaus signalų diagnostika bei daugelis kitų [1].

Šiuo metu visų dažnų susietumų radimas didelėse duomenų bazėse yra labai populiari ir viliojanti užduotis. Pagrindinė problema, nustatant dažnus susietumus, yra kiek galima minimizuoti įvedimo-išvedimo operacijos, kurios yra labai lėtos. Tačiau iš kitos pusės visiškai to išvengti nepavyks [2].

Tarkime turime aibę elementų. Pažymėkime šią aibę *I*. Duomenų bazė *D* sudaryta iš transakcijų. Kiekviena transakcija turi savo unikalų numerį *tid* ir yra sudaryta iš tam tikros aibės elementų, priklausančių *I*. Elementų seką pavadinkime *rinkiniu* (angl. *itemset*). Rinkinys, sudarytas iš *k* elementų vadinamas *k-rinkiniu*. Rinkinio *X dažnumas* (angl. *support*), žymimas $\sigma(X)$, yra skaičius transakcijų, kuriose šis rinkinys yra transakcijų poaibis. Poaibis, kurio dydis *k*, vadinamas *k-poaibiu*. Rinkinys vadinamas *maksimaliu*, jei jis nėra jokio kito rinkinio poaibis. Rinkinys vadinamas *dažnu*, jei jo pasikartojamumas yra nemažesnis už vartotojo nurodytą *minimalų dažnumą* (*min_sup*). Aibę dažnų *k*-rinkinių žymėsime F_k [3, 4].

Susietumo taisykle (angl. *association rule*) vadinsime išraišką $A \Rightarrow B$, kur *A* ir *B* yra rinkiniai. Dažnumas šioje taisyklėje žymimas kaip $\sigma(A \cup B)$, o *tikrumas* (angl. *confidence*) kaip $\frac{\sigma(A \cup B)}{\sigma(A)}$ (t.y. sąlyginė tikimybė, kad transakcijoje yra *B* su sąlyga, kad joje yra ir *A*). Taisyklė yra *griežta* (angl. *strong*), jeigu tikrumas yra nemažesnis nei vartotojo nurodytas minimalus tikrumas *min_conf* [5, 6, 7].

Duomenų paieškos užduotis yra sugeneruoti visas susietumo taisykles duomenų bazėje, kurios turi dažnumą didesnį už minimalų dažnumą (*min_sup*), t.y sugeneruoti dažnas taisykles. Taisyklės taip pat turi turėti tikrumą nemažesnį už minimalų tikrumą (*min_conf*), t.y. taisyklės yra griežtos. Ši užduotis yra padalinta į du žingsnius:

1. Visų dažnų rinkinių radimas. Šis žingsnis reikalauja labai daug skaičiavimų bei įvedimo-išvedimo operacijų. Jeigu iš viso yra duota *m* elementų, tai potencialiai gali būti 2^m dažnų rinkinių. Reikalingi efektyvūs metodai, kurie sumažina rinkinių paieškos erdvę. Dažnų rinkinių radimas ir yra pagrindinis tikslas.
2. Griežtų taisyklių generavimas. Šis žingsnis yra sąlyginai tiesinis. $X \backslash Y \rightarrow Y$ formos (čia $Y \subset X$) taisyklės, kur taisyklės yra generuojamos visiems dažniems *X* rinkiniams, jeigu taisyklė turi nemažesnį nei minimalų tikrumą [8, 9].

Paimkime kaip pavyzdį knygyno duomenų bazę, kuri parodyta 1 paveiksle.

Elementai

Aleksandras Diuma	A
Agata Kristi	C
Artūras Konan Doilis	D
Markas Tvenas	T
Gelbertas Velsas	W

Duomenų bazė

Transakcijos	Elementai
1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

1 **paveikslas**. Knygyno duomenų bazė

Šiame pavyzdyje turime penkis skirtingus elementus (autorių vardus, kurių knygas turi knygynas), t.y. $I=\{A, C, D, T, W\}$, ir duomenų bazę, kurioje pavaizduoti šešių pirkėjų pirkiniai.

Dažni rinkiniai (*min_sup*=50%)

Dažnumas	Rinkiniai
100% (6)	C
83% (5)	W, CW
67% (4)	A, D, T, AC, AW, CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW, **CDW**, CTW, **ACTW**

Maksimalūs dažni rinkiniai yra **CDW** ir **ACTW**

Susietumo taisyklės (*min_conf*=100%)

$A \Rightarrow C$ (4/4)	$AC \Rightarrow W$(4/4)	$TW \Rightarrow C$(3/3)
$A \Rightarrow W$(4/4)	$AT \Rightarrow C$(3/3)	$AT \Rightarrow CW$(3/3)
$A \Rightarrow CW$(4/4)	$AT \Rightarrow W$(3/3)	$TW \Rightarrow AC$(3/3)
$D \Rightarrow C$(4/4)	$AW \Rightarrow C$(4/4)	$ACT \Rightarrow W$(3/3)
$T \Rightarrow C$(4/4)	$DW \Rightarrow C$(3/3)	$ATW \Rightarrow C$(3/3)
$W \Rightarrow C$(5/5)	$TW \Rightarrow A$(3/3)	$CTW \Rightarrow A$(3/3)

2 **paveikslas**. Dažni rinkiniai ir susietumo taisyklės

2 paveiksle pavaizduoti visi dažni rinkiniai, kurie pasikartoja nemažiau kaip trijose transakcijose (t.y. minimalus dažnumas *min_sup* yra lygus 50%). Ten taip pat pavaizduota aibė visų susietumo taisyklių su minimaliu tikrumu *min_conf*=100% [10].

Rinkiniai ACTW ir CDW yra maksimalūs dažni rinkiniai. Kadangi visi kiti dažni rinkiniai yra gaunami iš maksimalių dažnų rinkinių, tai problemos sprendimą galime sutrumpinti, nurodant tik maksimalius dažnus rinkinius. Iš kitos pusės, norint nustatyti visas tikras susietumo taisykles, reikia žinoti visų dažnų rinkinių dažnumus. Tai galime pasiekti, kai bus identifikuoti maksimalūs elementai, atliekant papildomą duomenų bazės apėjimą ir nustatant dažnumą visų nepaskaičiuotų posekių.

2.3. Rinkinių nustatymas gardelių metodu

Prieš tyrinėjant algoritmus būtina apibrėžti tam tikras sąvokas ir pateikti kai kurias teoremas [plačiau 17].

1 apibrėžimas

Tarkime P yra aibė. ***Dalinis sutvarkymas*** *aibėje P yra toks binarinis sąryšis* $\leq$*, kad visiems* $X, Y, Z \in P$*, galioja šios savybės:*

1) *refleksyvumas:* $X \leq X$;
2) *antisimetriškumas (komutatyvumas) : jei* $X \leq Y$ *ir* $Y \leq X$*, tai* $X = Y$;
3) *tranzityvumas: jei* $X \leq Y$ *ir* $Y \leq Z$*, tai* $X \leq Z$.

Aibė P su sąryšiu $\leq$ *vadinama* ***sutvarkyta aibe****.*

2 apibrėžimas

Tarkime P yra sutvarkyta aibė ir tegul $X, Y, Z \in P$*. Sakysime, kad X yra* ***padengta*** *Y (žymėsime* $X \sqsubset Y$*), jeigu iš* $X < Y$ *ir* $X \leq Z < Y$ *išplaukia* $Z = X$*, t.y. jei nėra elemento* $Z \in P$*, tokio, kad* $X < Z < Y$.

3 apibrėžimas

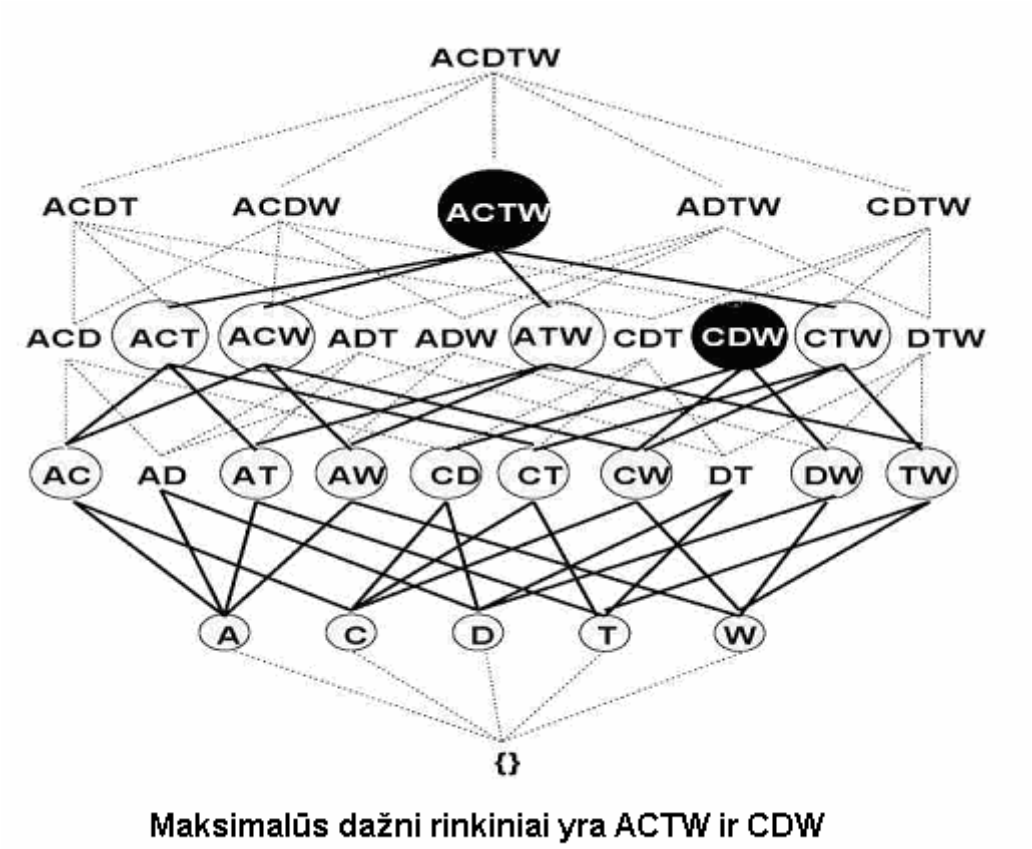
Tarkime P yra sutvarkyta aibė ir tegul $S \subseteq P$*. Elementas* $X \in P$ *yra aibės S* **viršutinė riba** (**apatinė riba**)*, jeigu* $s \leq X$ $(s \geq X)$ *visiems* $s \in S$*. Mažiausia viršutinė riba, vadinama* **sujungimu** *aibėje S ir žymima* $\vee S$*. Didžiausia apatinė riba, vadinama* **susikirtimu** *aibėje S ir žymima* $\wedge S$*. Didžiausias elementas aibėje P, žymimas* $\top$*, yra vadinamas* **viršutiniu elementu**, *o mažiausias elementas aibėje P, žymimas* $\bot$*, vadinamas* **apatiniu elementu.**

4 apibrėžimas

Tarkime L yra sutvarkyta aibė. L vadinama ***surišta*** *(sudurta)* ***pusinė gardelė*** *(*angl. *semilattice), jeigu* $X \vee Y$ *(*$X \wedge Y$*) egzistuoja visiems* $X, Y \in L$*. L vadinama* ***gardele*** *(*angl. *lattice), jei ji yra surišta (sudurta) pusinė gardelė, t.y. jeigu visiems* $X, Y \in L$ *egzistuoja* $X \vee Y$ *ir* $X \wedge Y$*. Aibė L yra* ***išsami gardelė****, jeigu visiems* $S \subseteq L$ *egzistuoja* $\vee S$ *ir* $\wedge S$*. Sutvarkyta aibė* $M \subset L$ *vadinama aibės L* ***daline gardele****, jei, kai* $X, Y \in L$*, tai ir* $X \vee Y \in M$ *bei* $X \wedge Y \in M$*.*

Aibės S galia yra sutvarkyta aibė *P(S)*, kuri yra išsami gardelė, kurioje sujungimai ir susikirtimai išreiškiami sąjunga ir sankirta atitinkamai:

$$\vee\{A_i \mid i \in I\} = \bigcup_{i \in I} A_i \qquad \wedge\{A_i \mid i \in I\} = \bigcap_{i \in I} A_i$$



3 **paveikslas**. Išsami gardelė *P*(*I*)

Viršutinis gardelės *P*(*S*) elementas yra $\top = S$, o apatinis *P*(*S*) elementas yra $\perp = \{\}$ (tuščia aibė). Bet kuri $L \subseteq P(S)$ vadinamas *aibių gardele*, jeigu ji yra uždara baigtinių sąjungų ir sankirtų operacijų atžvilgiu, t.y. $(L, \subseteq)$ yra gardelė su daline tvarka, nusakyta poaibio sąryšiu $\subseteq$ ir $X \vee Y = X \cup Y$ bei $X \wedge Y = X \cap Y$ [18].

3 paveiksle pavaizduota išsami gardelė *P(I)*, kuri sudaryta iš mūsų duomenų bazės elementų $I=\{A, C, D, T, W\}$. Jame taip pat pavaizduoti dažni rinkiniai (pilki skrituliai) ir maksimalūs dažni rinkiniai (juodi skrituliai). Galima pastebėti, kad aibė visų dažnų rinkinių formuojama iš pusinių gardelių, ir jame esantys dažni rinkiniai taip pat yra iš apatinės pusinės gardelės, t.y. esant bet kokiems dažniems rinkiniams *X* ir *Y*, jų sankirta $X \cap Y$ taip pat bus dažna. Iš kitos pusės žiūrint, jei *X* ir *Y* yra dažni rinkiniai, tai jų sąjunga $X \cup Y$ nebūtinai yra dažna. Galima tikėtis, kad visi nedažni rinkiniai formuoja savo nedažną dalinę gardelę [18].

1 lema.

Visi dažnų rinkinių poaibiai yra dažni [11,12,13].

Ši lema leidžia sumažinti galimas paieškas bei sutaupyti įvedimo-išvedimo laiką. Iš jos galima daryti išvadą, kad visi rinkiniai, gaunami iš nedažno rinkinio, yra nedažni. Pagrindinė strategija, kuria remiantis bus vykdoma tolimesnė dažnų rinkinių paieška, yra ta, kad dabar galimus dažnus rinkinius galim formuoti tik iš dažnų prieš tai buvusių rinkinių aibės. Tai akivaizdžiai yra matoma iš pateiktos gardelės.

2 lema.

Maksimalūs dažni rinkiniai vienareikšmiškai nusako visus dažnus rinkinius. [14,15]

Teiginys pasako, kad pagrindinis tikslas yra nustatyti procedūrą, kuri suranda maksimalius dažnus rinkinius.

2.4. Dažnumų skaičiavimas

Šiame skyrelyje įsivesime keletą apibrėžimų ir lemų, skirtų dažnumo paskaičiavimui. Šis dažnumo paskaičiavimas remiasi poaibių sąjungos bei sankirtos nustatymams. Norint apskaičiuoti tam tikro poaibio dažnumą, mums nereikės peržiūrėti visą duomenų bazę.

5 apibrėžimas

Gardelė L vadinama **paskirstomąja**, *jeigu visiems* $X, Y, Z \in L$ *galioja*

$X \wedge (Y \wedge Z) = (X \wedge Y) \vee (X \wedge Z)$ [19, 20, 21].

6 apibrėžimas

Tarkime L yra gardelė su apatiniu elementu . Tada $X \in L$ *yra vadinamas* ***atomu***, *jeigu* $\perp \sqsubset X$, *t.y. X padengia* $\perp$. *Gardelės L atomų aibė žymima A(L)* [19, 20, 21].

7 apibrėžimas

Gardelė L vadinama ***logine gardele****, jeigu:*

1) *ji yra paskirstomoji;*
2) *turi* $\perp$ *ir* $\top$ *elementus;*
3) *kiekvienas gardelės narys turi papildinį* [19, 20, 21].

Galima pastebėti, kad duomenų bazės elementų *I* išsamioji gardelė *P(I)* yra loginė gardelė su papildiniu $X \in L$, gaunamu kaip $I \backslash X$. Gardelės atomų aibė atitinka elementų aibę, t.y. $A(P(I))=I$. Su kiekvienu duomenų bazės atomu *X* susiejamas jo transakcijų sąrašas *tid-list* (pavadinkime jį *L(I)*), kuriame nurodomos tos transakcijos, kuriose yra tas atomas. 4 paveiksle parodytas pavyzdžio transakcijų paskirstymas atomams. Pavyzdžiui, atomas *A* pasirodo duomenų bazės 1, 3, 4 ir 5 transakcijose [17].

A	C	D	T	W
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

4 paveikslas. Atomų transakcijų sąrašas

3 lema

Baigtinei loginei gardelei L su $X \in L$ *galioja* $X = \vee \{Y \in A(L) | Y \leq X\}$ [16].

Kitaip sakant, kiekvienas loginės gardelės elementas yra gaunamas kaip atomų aibės poaibio sujungimas. Kadangi išsamioji gardelė *P(I)* yra loginė gardelė su sujungimo operaciją, atitinkančią aibių sąjungos operaciją, tai gaunama 4 lema [17].

4 lema.

Tegul $J=\{Y \in A(P(I)) \mid Y \leq X\}$ *kiekvienam* $X \in P(I)$. *Tada*

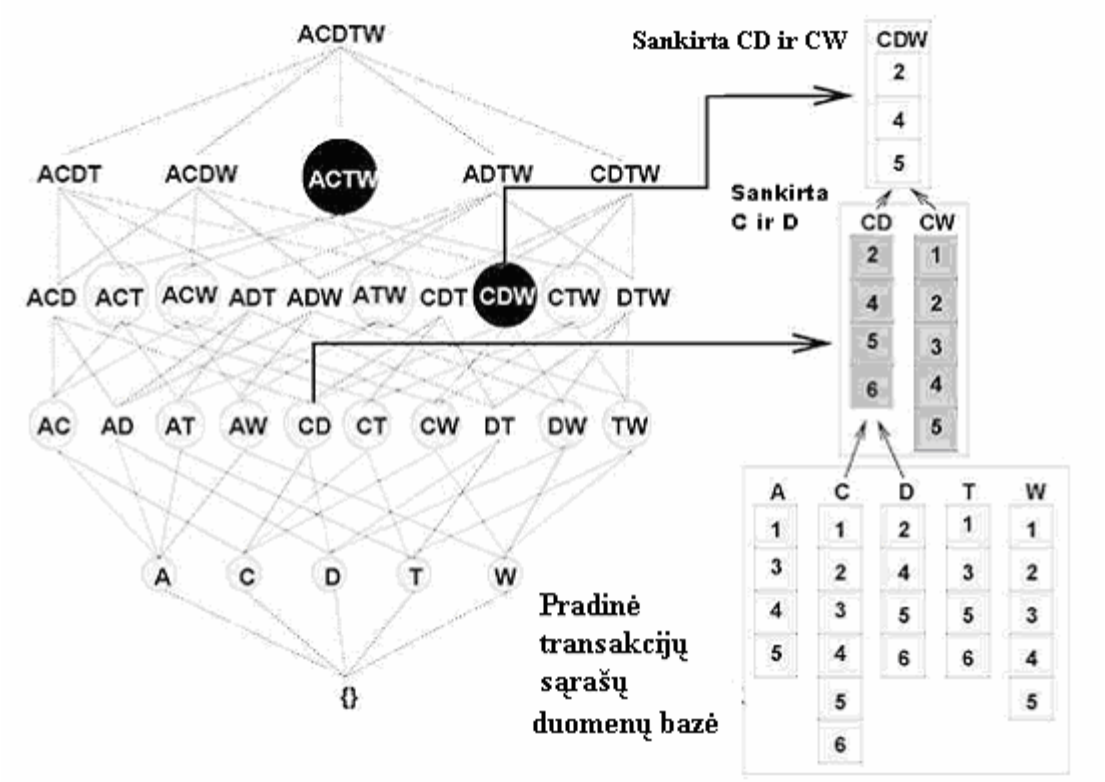
$X = \bigcup_{Y \in J} Y$ *ir* $\sigma(x) = \left| \bigcap_{Y \in J} L(Y) \right|$ [16].

Ši lema pasako, kad bet koks rinkinys gali būti gautas apjungiant keletą gardelės atomų, o rinkinio dažnumas gali būti gautas kaip atomų transakcijų sąrašų sankirta. Galima apibendrinti šią lemą [17].

5 lema.

Tegul $X = \bigcup_{Y \in J} Y$ *kiekvienam* $X \in P(I)$. *Tada* $\sigma(x) = |\bigcap_{Y \in J} L(Y)|$.

Ši lema sako, kad jeigu koks nors rinkinys pateiktas kaip rinkinių iš *J* sąjunga, tai dažnumas gaunamas iš *tid_list* elementų sankirtos. Atskiru atveju galima apskaičiuoti kiekvieno *k*-rinkinio dažnumą, paprasčiausiai nustatant bet kokių dviejų (*k*-1) ilgio rinkinių iš transakcijų sąrašo sankirtas. Paprastas elementų kiekio patikrinimas rezultato transakcijų sąraše leis nustatyti, ar rinkinys yra dažnas ar ne. Tokia situacija pavaizduota 5 paveiksle [17].



5 paveikslas. Rinkinių dažnumo paskaičiavimas pagal transakcijų sąrašų sankirtas

Čia pateikta duomenų bazė su transakcijų sąrašu kiekvienam elementui (t.y. atomui). Tarpinis *CD* transakcijų sąrašas yra gaunamas iš *C* ir *D* transakcijų sąrašų sankirtos, t.y. $L(CD)=L(C)\cap L(D)$. Analogiškai $L(CDW)=L(CD)\cap L(DW)$ ir t.t. Taigi, tiktai leksikografiškai du pirmieji poaibiai iš prieš tai buvusio lygmens reikalingi rinkinio dažnumui bet kuriame kitame lygmenyje apskaičiuoti.

6 lema.

Tegul X ir Y yra du rinkiniai ir $X \subseteq Y$. Tada $L(X) \supseteq L(Y)$.

Ši lema teigia, jeigu *X* yra *Y* poaibis, tai *Y* elementų kiekis (t.y. jo dažnumas) transakcijų sąraše *tid_list* gali būti nedidesnis nei *X*. Pagal šią lemą galima padaryti išvadą, jog transakcijų sąrašo elementų kiekis visada mažėja, kai „kylama“ gardele aukštyn. Tokiu būdu galima gana greitai apskaičiuoti rinkinių sankirtas ir dažnumus [22, 23].

2.5. Gardelės suskaidymas. Priesagų klasės

Jeigu būtų pakankamai pagrindinės atminties, tai būtų galima išvardinti ir nustatyti visus dažnus rinkinius, pereinant per visą tinklelį. Tačiau praktiškai tai padaryti mums trukdo pagrindinės atminties ribotumas, todėl visi tarpiniai transakcijų sąrašai negalės būti saugomi ir talpinami pagrindinėje atmintyje. Kyla logiškas klausimas ar negalima būtų suskaidyti gardelę į dalis ir toliau tas dalis nagrinėti nepriklausomai viena nuo kitos [24, 25].

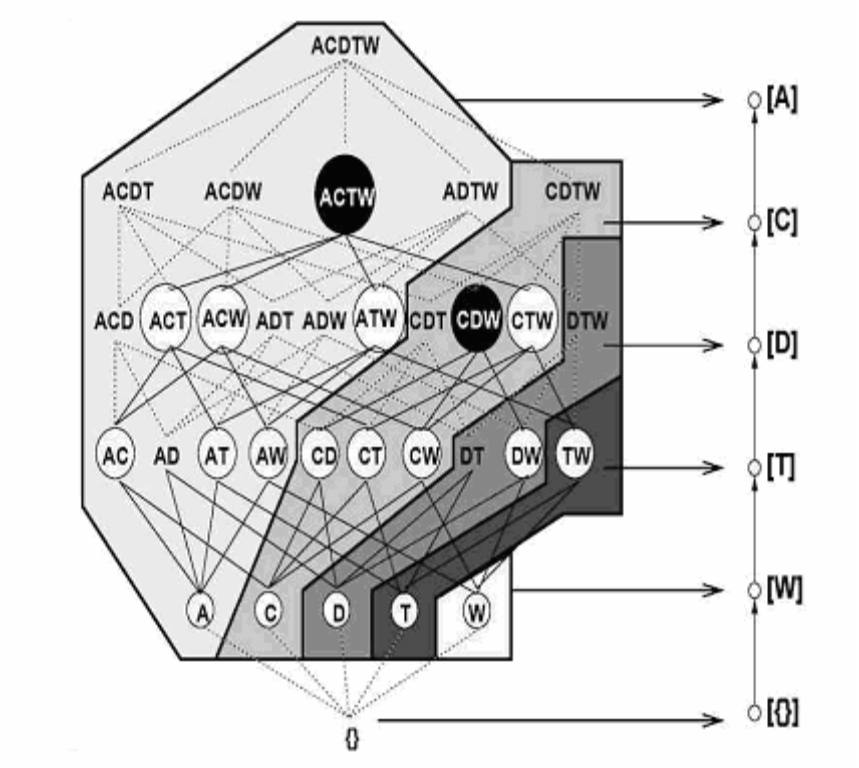
8 apibrėžimas

Tegul P yra aibė. ***Ekvivalentumo sąryšis*** *aibėje P yra toks binarinis ryšys $\equiv$, kad visiems X, Y, $Z \in P$, galioja šios savybės:*

1) *refleksyvumas: $X \equiv X$;*
2) *simetriškumas: jei $X \equiv Y$, tai $Y \equiv X$;*
3) *tranzityvumas: jei $X \equiv Y$ ir $Y \equiv Z$, tai $X \equiv Z$.*

Ekvivalentumo sąryšis suskaido aibę P į nesusikertančius poaibius, kurie yra vadinami ekvivalentumo klasėmis. Ekvivalentumo klasė elementui $X \in P$ yra gaunama kaip $[X]=\{Y \in P \mid X \equiv Y\}$ [24, 25].

Apibrėžkime funkciją *p*: *P(I)* $\mapsto$ *P(I)*, čia $p(X, k) \equiv X[1: k]$ ir *k* yra *X* priesagos ilgis. Apibrėžkime ekvivalentumo ryšį θ_k gardelėje *P(I)* šitaip: $\forall X, Y \in P(I),\ X \equiv_{\theta_k} Y \Leftrightarrow p(x,k) = p(Y,k)$. Kitaip tariant, du rinkiniai priklauso tai pačiai klasei, jeigu jie turi tą pačią *k* ilgio priesagą. Todėl θ_k vadinama priesaginiu ekvivalentumo ryšiu [17, 24, 25].



6 paveikslas. Aibės *P(I)* ekvivalentumo klasės pagal θ_1

6 paveiksle pavaizduota gardelė, suskirstyta pagal ekvivalentumo ryšį θ_1 iš *P*(*I*). Visi rinkiniai su bendra 1 ilgio priesaga sujungti į ekvivalentumo klasę. Galutinė ekvivalentumo klasių aibė yra {[*A*], [*C*], [*D*], [*T*], [*W*]}.

7 lema.

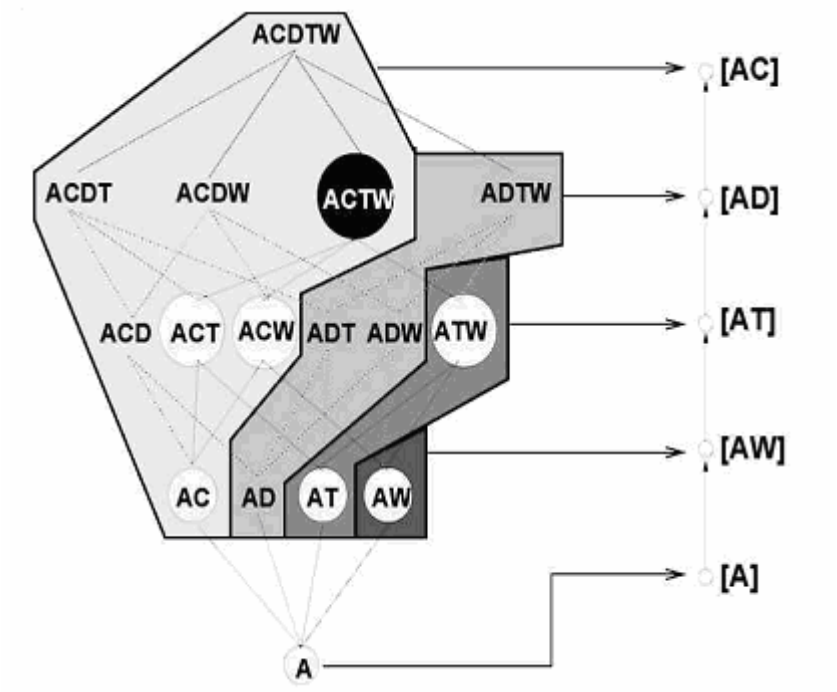
Kiekviena ekvivalentumo klasė $[X]_{\theta_k}$ *, gaunama iš ekvivalentumo ryšio* θ_k *, yra gardelės P(I) dalis.*

Įrodymas. Tegul *U* ir *V* yra bet kokie du elementai klasėje [*X*], t.y. *U* ir *V* turi bendrą priesagą *X*. *Iš* $U \vee V = U \cup V \supseteq X$ *išplaukia, kad* $U \vee V \in [X]$, *o iš* $U \wedge V = U \cap V \supseteq X$ *išplaukia, kad* $U \wedge V \in [X]$. Dėl to $[X]_{\theta_k}$ yra gardelės *P(I) dalinė gardelė* [18].

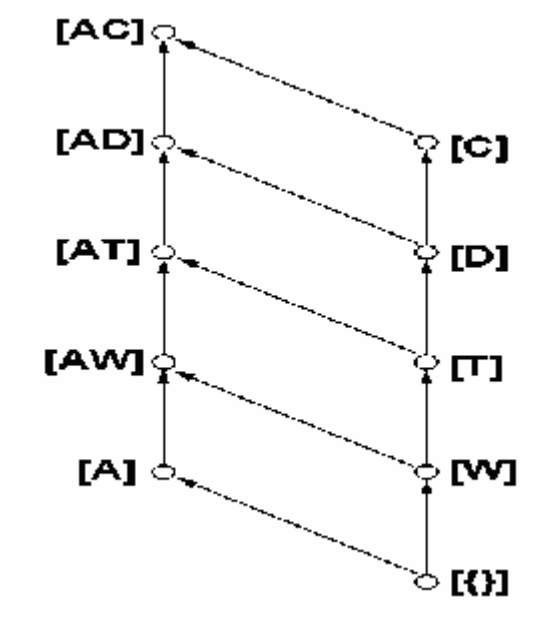
Kiekviena $[X]_{\theta_1}$ pati yra loginė gardelė su savo nuosavu atomų rinkiniu. Pavyzdžiui, klasės $[A]_{\theta_1}$ atomai yra {*AC, AD, AT, AW*}, o viršutinis ir apatinis elementas yra atitinkamai $\top$=*ACDTW* ir $\perp$=*A*. Remiantis 4 ir 5 lema, galima generuoti visus rinkinių dažnumus kiekvienai klasei (gardelės daliai), randant atomų transakcijų sąrašų arba bet kurių ankstesniojo lygmens poaibio sankirtas. Jeigu yra pakankamai atminties saugoti laikinuosius *tid_list* kiekvienai klasei, tai mes galime jas nagrinėti atskirai ir nepriklausomai. Kita įdomi ekvivalentumo klasių savybė yra tai, kad ryšiai tarp klasių nurodo jų priklausomybes. Tuo

norima pasakyti, kad galima pašalinti rinkinį, kuris turi nors vieną nedažną poaibį (žr. 2 lemą) ir tuo pačiu galima apdoroti klases tam tikra tvarka. Reikia nagrinėti visas klases nuo apačios iki viršaus atvirkščia leksikografine tvarka, t.y. pirmiausia nagrinėjama klasė [*W*], paskui [*T*], po to [*D*], [*C*] ir galiausiai [*A*]. Tai garantuos, kad visa informacija bus prieinama, norint pašalinti nedažnus poaibius [1, 2].

Praktiškai mes nustatėme, kad vieno lygmens dekompozicija gaunama su θ_1 yra pakankama. Tačiau iš kitos pusės klasė gali būti pakankamai didelė, kad galėtų būti patalpinama į pagrindinę atmintį. Tokiu būdu galima atlikti dar vieną rekursinę dekompoziciją. Tarkime klasė [*A*] yra per didelė būti patalpinama į pagrindinę atmintį. Kadangi klasė [*A*] irgi yra loginė gardelė, tai galima atlikti dekompoziciją, panaudojant θ_2. 7 paveiksle parodyta ekvivalentumo klasė, gauta taikant θ_2 klasei [*A*]. Galutinė klasių aibė yra {|*AC*|, |*AD*|, |*AT*|, |*AW*|}. Ir kiekviena ši ekvivalentumo klasė gali būti nagrinėjama atskirai. Dabar reikia rasti vieno lygmens dekompozicija. Galutinė klasių aibė yra gaunama taikant θ_1 aibei *P(I)* ir θ_2 klasei $[A]_{\theta_1}$. Kaip anksčiau buvo pastebėta, tarp klasių egzistuoja ryšiai, pagal kuriuos galima vykdyti nedažnų rinkinių atmetimą. Todėl optimaliausia klasių nagrinėjimo tvarka yra parodyta 8 paveiksle. Priklausomai nuo pagrindinės atminties kiekio, mes galime rekursiškai suskaidyti visas klases iki tinkamo dydžio [1, 2].



7 paveikslas. Aibės $[A]_{\theta_1}$ ekvivalentumo klasės pagal θ_2



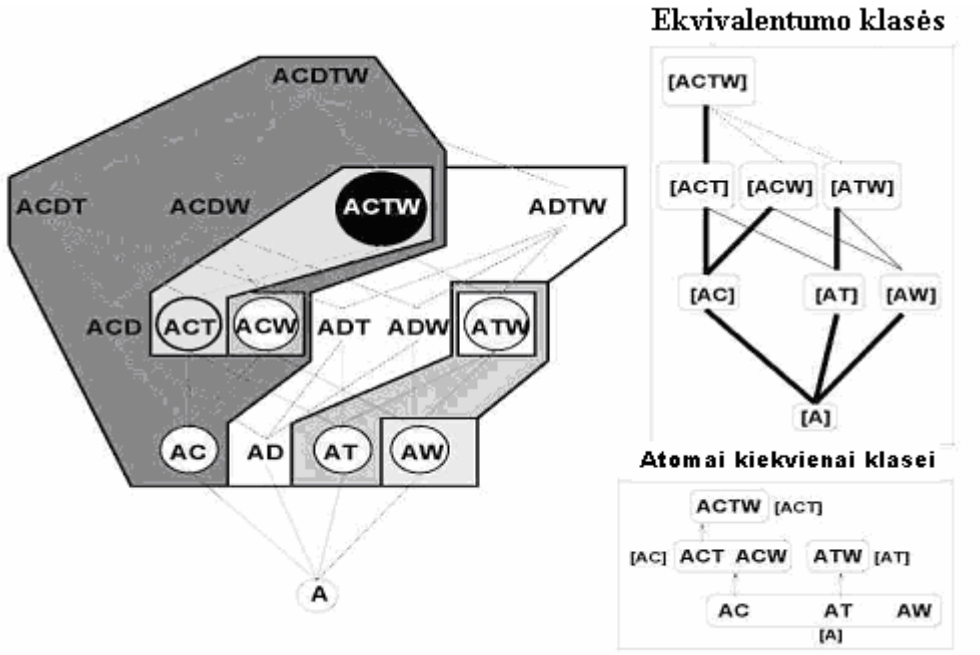
8 paveikslas Galutinė nepriklausomų klasių gardelė

2.6. Dažnų rinkinių paieška

Šioje dalyje pateikiami efektyvūs dažnų rinkinių paieškos būdai.

Paieška iš apačios į viršų

Paieškos iš apačios į viršų pagrindą sudaro kiekvienos klasės rekursinė dekompozicija į mažesnes klases, pasinaudojant θ_k ryšiu. 9 paveiksle parodyta klasės $[A]_{\theta_1}$ dekompozicija į mažesnes klases ir gauta ekvivalentumo klasių gardelė.

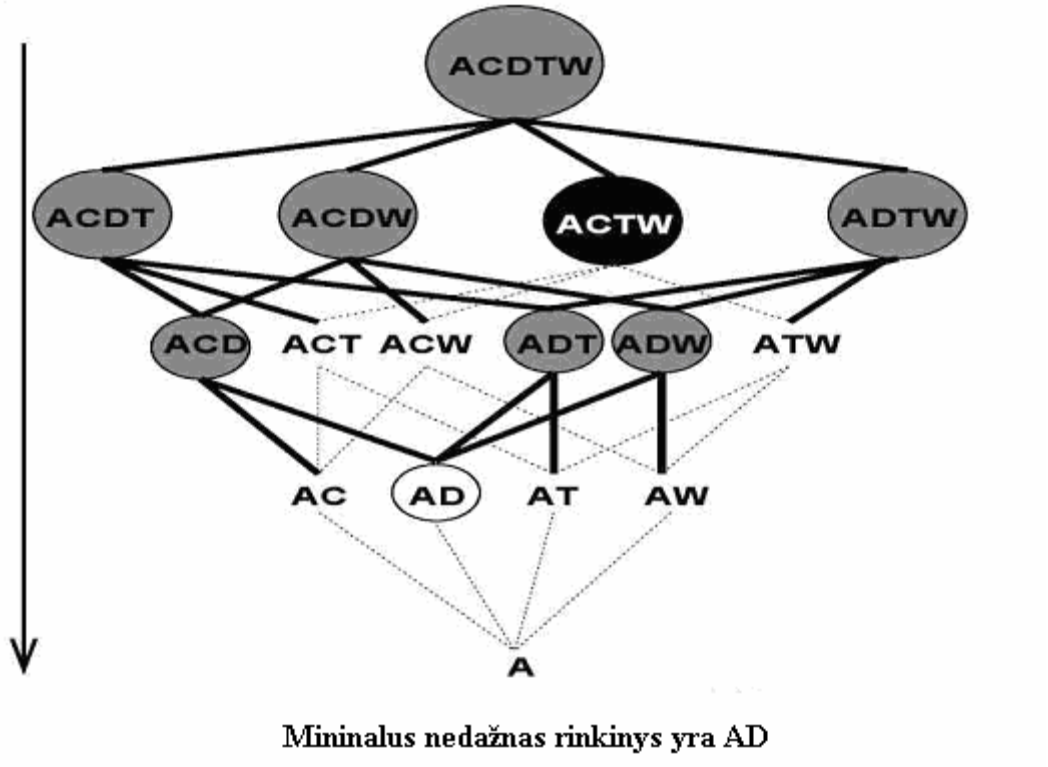


9 paveikslas. Paieška iš apačios į viršų

Čia taip pat parodyti kiekvienos klasės atomai, pagal kuriuos bus galima nustatyti visus likusius klasės elementus. Klasės gardelė gali būti nagrinėjama tiek „į gylį“, tiek ir „į plotį“. Čia parodyti tik apėjimo iš apačios į viršų ir pagal plotį rezultatai. Kitaip tariant, pirmiausia nagrinėjamos klasės {[*AC*], [*AT*], [*AW*]}, po to klasės {[*ACT*], [*ACW*], [*ATW*]} ir, galiausiai, [*ACTW*]. Norint nustatyti bet kokio rinkinio dažnumą, tiesiog imama ankstesniojo lygmens dviejų poaibių transakcijos sąrašų sankirta. Kadangi paieška vykdoma iš apačios į viršų, tai tokiu būdu galima gauti visus dažnus rinkinius [26, 27, 28].

Paieška iš viršaus žemyn

Pagal šį būdą rinkiniai nagrinėjami pradedant nuo pačio viršutinio elemento gardelėje. Jo dažnumas randamas, peržiūrint visus transakcijų sąrašo *tid_list* elementus ir atliekant jų sankirtą. Tokiu būdu, reikia atlikti *k* sankirtų, jeigu viršutinis elementas yra *k*-rinkinys. Pagrindinis metodo privalumas yra tas, kad, jeigu tas elementas yra dažnas, tai jo poaibiai irgi yra dažni (žr. 4 lemą). Paieška prasideda nuo paties viršutinio elemento. Jei jis yra dažnas, tai tuo paiešką ir baigiama (kadangi visi jo poaibiai irgi bus dažni). Jei ne, tai toliau tikrinami visi žemesnio lygmens rinkiniai. Procesas tęsiasi tol, kol surandami visi minimalūs nedažni rinkiniai. 10 paveiksle pavaizduotas šis paieškos būdas [29,30, 31].

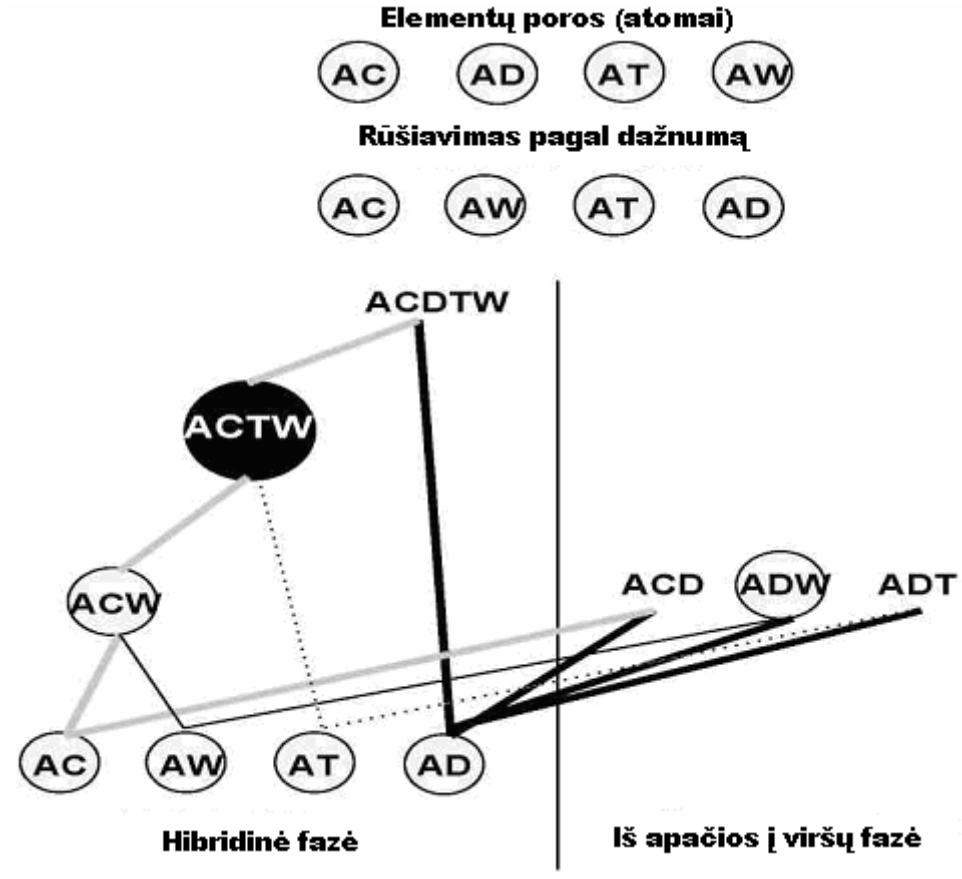


10 paveikslas. Paieška iš viršaus į apačią

Šis paieškos būdas randa tik kiekvienos dalinės gardelės maksimalius dažnus rinkinius. Tačiau maksimalus dalinės gardelės elementas gali ir nebūti maksimaliu globalioje gardelėje. Tokiu būdu, gali būti generuojami ir nemaksimalūs dažni rinkiniai [29, 30, 31].

Hibridinė paieška

Hibridinė paieškos schema remiasi intuityvia prielaida, kad didžiausią dažnumą turintis dažnas rinkinys tikėtinai yra dalis ilgesnio dažno rinkinio. Šį būdą sudaro dvi dalys. Pirmiausia visi klasės atomai surikiuojami pagal jų dažnumą. Pirma hibridinės paieškos fazė prasideda nuo pačio dažniausio atomo, po to vis didinant ir didinant rinkinio ilgį. Šis procesas sustoja, kai po kurio nors išplėtimo randamas nedažnas rinkinys. Antroje fazėje naudojamas iš apačios į viršų metodas. Likę atomai yra kombinuojami su atomais iš pirmos dalies tam, kad vykdyti paiešką į plotį ir tokiu būdu gauti visus likusius dažnus rinkinius. Hibridinė paieška pavaizduota 11 paveiksle (kad būtų aiškesnis algoritmas, o ypač fazė iš apačios į viršų, tarkime, kad *AD* ir *ADW* taip pat yra dažni rinkiniai) [32, 33, 34].



11 paveikslas. Hibridinė paieška

Kaip ir paieškoje iš apačios į viršų, šis būdas turi 2 persikirtimo kelius. Metodas suranda „ilgą“ maksimalią dažną seką hibridinėje fazėje, o taip randa ir nemaksimalius dažnus rinkinius iš apačios į viršų žingsnyje.

2.7. Mažesnių klasių generavimas

Šioje dalyje bus parodyta, kaip sukurti mažesnes dalines gardeles arba ekvivalentumo klases. Bus palyginta su tiesioginiu sukūrimu, remiantis priesagomis ir naudojant papildomą informaciją. Mažesnės gardelės turi savyje nedaug atomų ir tuo pačiu gali saugoti nereikalingas sankirtas. Tarkime, kad yra *k* atomų, tai tada būtina atlikti $\binom{k}{2}$ sankirtų kitam lygiui, naudojant iš apačios į viršų metodą. Mažai atomų sumažina sankirtų kiekį, vykdant paiešką nuo apačios iki viršaus. Mažai atomų mažina sankirtų skaičių ir hibridinėje paieškoje. Bendru atveju mažesnis atomų skaičius sumažina tyrinėjimo laiką [35, 36, 37].

9 apibrėžimas

Tegul P yra aibė. ***Pseudoekvivalentumo*** *sąryšis aibėje P yra toks binarinis ryšys* $\equiv$*, kad visiems X, Y, Z* $\in$ *P, galioja šios savybės:*

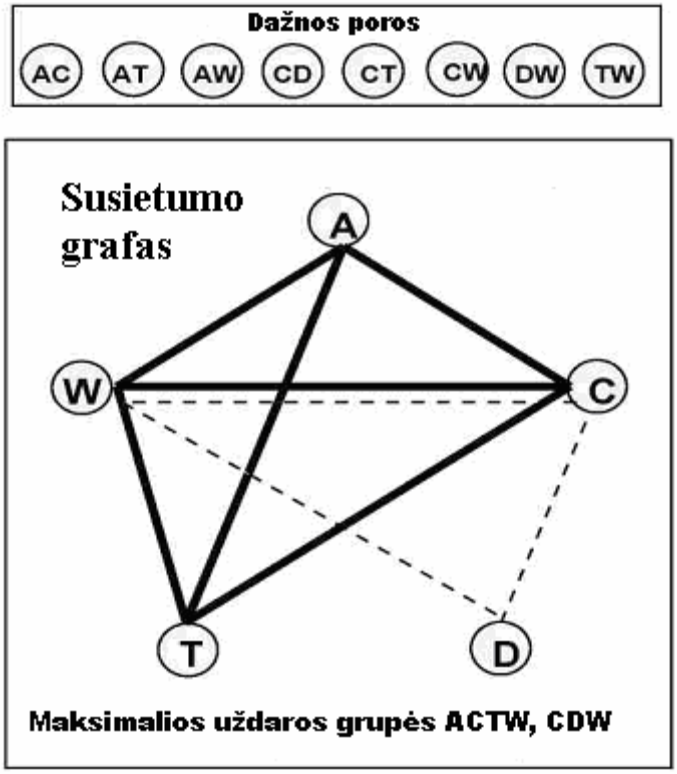
1) *refleksyvumas:* $X \equiv X$*;*
2) *simetriškumas: jei* $X \equiv Y$*, tai* $Y \equiv X$*.*

Pseudoekvivalentumo sąryšis suskaido aibę P į nesusikertančius poaibius, kurie yra vadinami pseudoekvivalentumo klasėmis [35, 36].

10 apibrėžimas

Grafas yra sudarytas iš aibės elementų V, kurie vadinami grafo **viršūnėmis,** *ir aibės tiesių, kurios sujungia tas viršūnes, kurios vadinamos grafo* **briaunomis**. *Grafas vadinamas* **pilnu**, *jei kiekviena jo viršūnė yra sujungta su kiekviena kita viršūne. Pilnas grafo pografis vadinamas* **uždara grupe** [35, 36, 37].

Tarkime F_k yra dažnų *k*-rinkinių aibė. Apibrėžkime *k*-susietumo grafą, kuris bus žymimas kaip $G_k=(V, E)$, su viršūnių aibe $V=\{X \mid X \in F_1\}$ ir briaunų aibe $E=\{(X, Y) \mid X, Y \in V$ *ir* $\exists\, Z \in F_{(k+1)}$ tokie, kad $X, Y \subset Z\}$. Tarkime M_k yra maksimalių uždarų grupių grafe G_k aibė. 12 paveiksle pavaizduotas susietumo grafas G_1 ir jo maksimalių uždarų grupių aibė $M_1=\{ACTW, CDW\}$.



12 paveikslas. Maksimalios uždaros grupės susietumo grafe

Apibrėžkime pseudoekvivalentumo sąryšį ϕ_k gardelei *P(I)* taip: $\forall X, Y \in P(I)$, $X \equiv_{\phi_k} Y \Leftrightarrow \exists C \in M_k$, toks kad X, $Y \subseteq C$ ir $p(X, k)=p(Y, k)$. Tai yra du rinkiniai X ir Y yra susieti pseudoekvivalentumo sąryšiu. Kitaip sakant, du rinkiniai yra susieti, t. y. jie yra toje pačioje pseudoklasėje, jeigu šie rinkiniai yra tos pačios maksimalios uždaros grupės poaibiai ir turi tą pačią priesagą, kurios dydis yra *k*. Sąryšis ϕ_k yra vadinamas *maksimaliu uždaros grupės* pseudoekvivalentumo sąryšiu [35, 36, 37].

8 lema.

Kiekviena pseudoklasė $[X]_{\phi_k}$ *, sukurta pseudoekvivalentumo ryšiu* ϕ_k *, yra gardelės P(I) dalinė gardelė.*

Įrodymas:

Tegul U ir V yra bet kokie du elementai klasėje [X], t. y. jie turi bendrą priesagą X ir egzistuoja maksimali uždara grupė $C \in M$, tokia, kad $U, V \in C$. Aišku, kad $U \cup V \subseteq C$ ir $U \cap V \subseteq C$. Be to, iš $U \vee V = U \cup V \supseteq X$ išplaukia, kad $U \vee V \in [X]$, o iš $U \wedge V = U \cap V \supseteq X$ išplaukia, kad $U \wedge V \in [X]$.

Dar galime pastebėti, kad kiekviena pseudoklasė $[X]_{\phi_k}$ yra loginė gardelė ir visų gardelės elementų dažnumai gali būti randami, taikant 4 ir 5 lemas atomams ir naudojant bent kurią iš anksčiau išvardintų 3-jų strategijų [38, 39, 40].

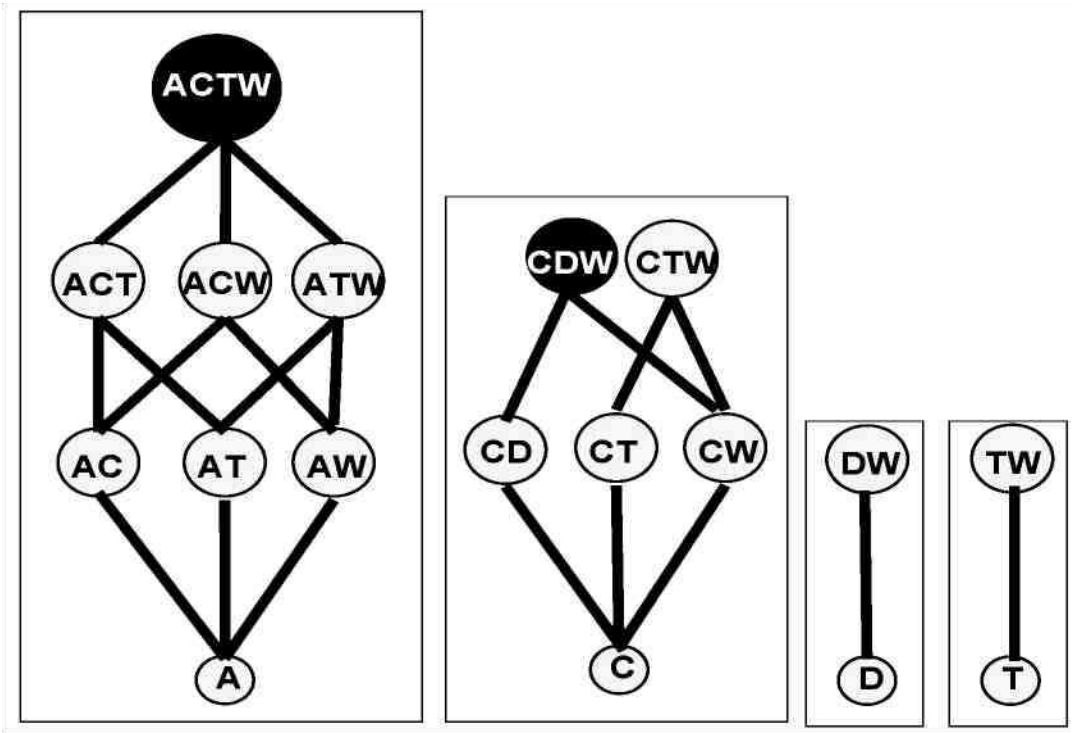
9 lema.

Apibrėžkime $\aleph_k$ *kaip aibę visų pseudoklasių, gautų iš maksimalios uždaros grupės sąryšio* ϕ_k. *Kiekviena pseudoklasė* $[Y]_{\phi_k}$ *gauta pagal priesaginį ryšį* ϕ_k, *yra tam tikros klasės* $[X]_{\theta_k}$ *poaibis, gaunamos iš ekvivalentumo sąryšio* θ_k. *Ir atvirkščiai, kiekviena klasė* $[X]_{\theta_k}$ *yra pseudoklasių* ψ, *gautų kaip* $[X]_{\theta_k} = \bigcup\{[Z]_{\phi_k} \mid Z \in \psi \subseteq \aleph_k\}$ *aibių sąjunga* [41, 42, 43].

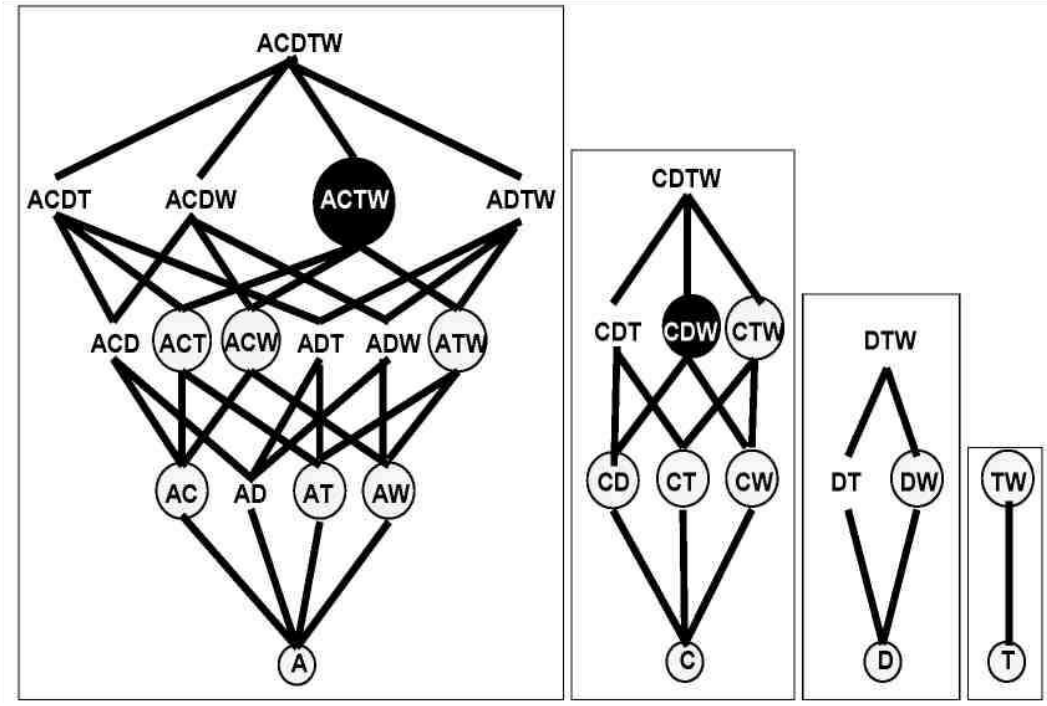
Įrodymas.

Tarkime, kad $\Gamma(X)$ žymi grafo G_k kaimynus X. Tada $[X]_{\theta_k} = \{Z \mid X \subseteq Z \subseteq \{X, \Gamma(X)\}\}$. Kitaip tariant, [$X$] sudarytas iš elementų su priesaga X ir praplėstas iki visų įmanomų grafo G_k kaimynų X poaibių. Kadangi bet kokia uždara grupė Y yra posekis $\{Y, \Gamma(Y)\}$, turime, kad $[Y]_{\phi_k} \subseteq [X]_{\theta_k}$, čia Y yra X priesaga. Iš kitos pusės paprasta parodyti, kad $[X]_{\theta_k} = \bigcup\{[Y]_{\phi_k} \mid Y$ yra X priesaga$\}$.

Pagal šią lemą gaunasi, kad kiekviena pseudoklasė iš ϕ_k yra patobulinta, o tiksliau mažesnė nei atitinkama klasė iš θ_k. Naudojant ryšį ϕ_k vietoj ryšio θ_k, galima generuoti mažesnes dalines gardeles. Jos naudoja mažiau atminties ir gali būti apdorojamos nepriklausomai, naudojant bet kurią iš trijų anksčiau išvardintų strategijų. 13 ir 14 paveiksluose parodyti pagrindiniai sąryšių θ_k ir ϕ_k skirtumai. Iš jų matyti, kad sąryšis ϕ_k sukuria mažesnes gardeles [39, 40, 42].



13 paveikslas. Maksimalios uždaros grupės dalinės gardelės pagal ϕ_k



14 paveikslas. Priesaginės dalinės gardelės pagal θ_k

2.8. Algoritmų konstravimas ir įgyvendinimas

Šioje dalyje nagrinėjama keletas algoritmų su kuriais galima apskaičiuoti dažnus rinkinius. Pirmame žingsnyje mes apskaičiuojami ir randami dažni elementai ir 2-rinkiniai (t.y. rinkiniai, kurių ilgis 2). Antrame žingsnyje iš dažnų 2-rinkinių aibės F_2 generuojamos dalinės gardelės (klasės) arba naudojant priesaginį ekvivalentumo sąryšį θ_1, arba naudojantis maksimaliu uždaros grupės sąryšiu ϕ_k. Dalinės gardelės po to yra nagrinėjamos atvirkščia leksikografine tvarka pagrindinėje atmintyje pagal bet kurią iš ankščiau paminėtų būdų: nuo apačios iki viršaus, nuo viršaus iki apačios ir hibridiniu [44, 45, 46].

1-rinkinių (elementų) ir 2-rinkinių dažnumo apskaičiavimas

Dauguma susietumo algoritmų naudoją horizontalų duomenų išdėstymą, pavaizduotą 1 paveiksle. Čia duomenys yra sudaryti iš transakcijų, ir kiekviena transakcija turi savyje elementų sąrašą. Šią duomenų bazę galime pertvarkyti į vertikalų formatą, parodytą 2 paveiksle. Šiuo formatu nurodoma kiekvieno elemento transakcijų sąrašas. Dažnumai randami, atliekant paprastas transakcijų sąrašų sankirtas.

***F₁* apskaičiavimas**. Jei yra vertikali duomenų bazė, tai visus dažnus elementus galima surasti vienu duomenų bazės apėjimu. Kiekvienam elementui tiesiog nuskaitomas jo transakcijų sąrašas iš disko į atmintį. Po to peržiūrimas transakcijų sąrašas, padidinant vienetu kiekvieno rasto elemento dažnumą.

***F₂* apskaičiavimas**. Tarkime $N = |I|$ yra dažnų elementų skaičius, o A yra transakcijų sąrašo (*id_list*) dydžio, išreikšto baitais, vidurkis. Būtu naivu skaičiuoti visus dažnus 2-rinkinius tiesioginiu būdu, nes tai reikalaus $\binom{N}{2}$ sankirtų visoms elementų poroms. Perskaitytų duomenų kiekis bus lygus $A \cdot N \cdot (N-1)/2$ kas atitinka maždaug $N/2$ duomenų peržiūrų. Tai yra aiškiai neefektyvu. Galima pasinaudoti tokiu alternatyviu sprendimu [47, 48, 49, 50].

Duomenų bazė transformuojama iš vertikalaus formato į horizontalų. Tai padaryti yra pakankamai lengva. Kiekvienam elementui galima priskirti masyvą atmintyje, kuriame bus talpinamas ir peržiūrimas jo pasirodymo transakcijose sąrašas. Šis būdas gali truputi persidengti, t.y. turėti nedidelę perteklinę informaciją [51, 52].

2.9. Paieškos realizavimas

Paieška iš apačios į viršų

15 paveiksle pavaizduotas paieškos iš apačios į viršų pseudokodas. Šioje procedūroje įvedimo duomenys yra visi gardelės *S* atomai. Dažni rinkiniai generuojami, atliekant visų skirtingų atomų porų transakcijų sąrašų (*tid-list*) sankirtas ir tikrinant gauto transakcijų sąrašo elementų kiekį. Tolimesniems rinkiniams apdoroti rekursinė procedūra kviečiama tik tiems sąrašo rinkiniams, kurie yra dažni einamajame lygmenyje. Šis procesas yra kartojamas tol, kol visi dažni rinkiniai bus apskaičiuoti. Atmintyje mums pakanka laikyti tik dviejų lygmenų sąrašus (einamojo ir prieš tai buvusio lygmens). Kai tik visi dažni rinkiniai iš einamojo lygmens bus surasti, tai galima bus pašalinti prieš tai buvusio lygmens dažnus rinkinius.

Kadangi visa dalinė gardelė yra apdorojama atvirkščia leksikografine tvarka, tai yra prieinama visa informacija iki tų rinkinių, kurie yra dažni. Todėl jie nenagrinėjami ir nauji rinkiniai nekuriami [53, 54, 55].

Bottom-Up(S):
for visiems atomams $A_i \in S$ **do**
  $T_i = \varnothing$;
  for visiems atomams $A_j \in S$, kur $j>i$ **do**
    $R = A_i \cup A_j$ **;**
    $L(R) = L(A_i) \cap L(A_j)$ **;**
    if $\sigma(R) \geq min\_sup$ **then**
      $T_i = T_i \cup \{R\}; F_{|R|} = F_{|R|} \cup \{R\}$;
  end
end;
for visiems $T_i \neq \varnothing$ **do** Bottom-Up(T_i);

15 paveikslas. Paieškos iš apačios į viršų pseudokodas

Paieška iš viršaus į apačią

Paieškos iš viršaus į apačią pseudokodas pavaizduotas 16 paveiksle. Paieška prasideda nuo didžiausio elemento *R* dalinėje gardelėje *S*. Tikrinant, eliminuojami elementai, apie kuriuos žinoma, kad jie yra dažni. Pirmiausia tikrinamas *k* lygmuo. Jei jame esantys rinkiniai yra dažni, tai algoritmas baigia savo darbą. Visi likę dažni rinkiniai bus gaunami kaip didžiausio rinkinio poaibiai. Jei didžiausias rinkinys yra nedažnas, tai rekursyviai tikrinamas (*k*-1) lygmuo lygiai pagal tas pačias taisykles. Taip pat naudojama speciali maišos lentelė (*HT*

hash table), kurioje saugomi visi ankstesnio rekursinio patikrinimo nedažni rinkiniai. Šitaip išvengiama pakartotinio transakcijų sąrašų apdorojimo [56, 57, 58].

Top-Down(S):
$R = \bigcup\{A_i \in S\}$;
if $R \notin F_{|R|}$ **then**
  $L(R) = \bigcap\{L(A_i) | A_i \in S\}$;
  if $\sigma(R) \geq min\_sup$ **then**
    $F_{|R|} = F_{|R|} \cup \{R\}$
  else
    for visiems $Y \subset R$, su $|Y|=|R|-1$ **do**
      if $Y \notin HT$ **then**
        Top-Down($\{A_j | A_j \in Y\}$);
      if $\sigma(Y) < min\_sup$ **then** $HT=HT \cup \{Y\}$;
    end;
end;

16 paveikslas. Paieškos iš viršaus į apačią pseudokodas

Hybrid (S surušiuota pagal dažnumą):
$R = A_1; S_1 = \{A_1\}$;
for visiems $A_i \in S$, $i > 1$ **do** /*Maksimumo fazė*/
  $R = R \cup A_i$; $L(R) = L(R) \cap L(A_i)$;
  if $\sigma(R) \geq min\_sup$ **then**
    $S_1 = S_1 \cup \{A_i\}$; $F_{|R|} = F_{|R|} \cup \{R\}$;
  else break;
end;
$S_2 = S - S_1$;
for visiems $B_i \in S_2$ **do** /*Iš apačios į viršų fazė */
    $T_i = \{X_j | \sigma(X_j) \geq min\_sup,\ L(X_j) = L(B_i) \cap L(A_j), \forall A_j \in S_1\}$;
    $S_1 = S_1 \cup \{B_i\}$;
    if $T_i \neq \varnothing$ **then** Bottom-Up(T_i);
end;

17 paveikslas. Hibridinės paieškos pseudokodas

Hibridinė paieška

Hibridinės paieškos algoritmo pseudokodas pavaizduotas 17 paveiksle. Įvedimo duomenys – tai surikiuoti pagal dažnumą aibės *S* atomai. Maksimali fazė pradedama, atliekant

atomų sankirtas po vieną, kol joks plėtinys yra nedažnas. Visi atomai, nagrinėjami šioje fazėje, saugomi aibėje S_1. Likę atomai $S_2=S\backslash S_1$ yra nagrinėjami antroje fazėje iš apačios į viršų. Kiekvienas atomas iš S_2 perkertamas (t.y. randama sankirta) su kiekvienu atomu iš S_1. Dažni rinkiniai naujoje gardelėje yra nustatomi pasinaudojus, paieška iš apačios į viršų. Procesas yra kartojamas su visais kitais atomais iš S_2. Maksimali fazė reikalauja atminties tik atomams, o paieškos nuo apačios aukštyn fazė reikalauja saugoti daugiausia dviejų paskutinių lygmenų rinkinius [59, 60, 61].

Uždaros grupės algoritmai

Žemiau pateikti algoritmai skiriasi paieškos strategija ir sąryšiais naudojamais generuojant nepriklausomas dalines gardeles.

1. *Eclat*. Algoritmas naudoja priesagų ekvivalentumo sąryšį θ_1 ir taiko paieškos iš apačios į viršų metodą. Tokiu būdu surandami visi dažni rinkiniai [62, 63, 64].
2. *MaxEclat*. Algoritmas naudoja priesagų ekvivalentumo sąryšį θ_1 ir taiko hibridinį paieškos metodą. Jis suranda ilgus maksimalius dažnus rinkinius ir kai kuriuos nemaksimalius dažnus rinkinius [65, 66, 67].
3. *Clique*. Algoritmas naudoja maksimalų uždarų grupių pseudoekvivalentumo sąryšį ϕ_1 ir taiko paieškos iš apačios į viršų metodą. Randa visus dažnus rinkinius [68, 69, 70].
4. *MaxClique.* Šis algoritmas naudoja maksimalų uždarų grupių pseudoekvivalentumo sąryšį ϕ_1 ir hibridinį paieškos metodą. Jis suranda ilgus maksimalius dažnus rinkinius ir kai kuriuos nemaksimalius dažnus rinkinius [71, 72, 73].
5. *TopDown.* Naudoja maksimalų uždarų grupių pseudoekvivalentumo ryšį ϕ_1 ir taiko paieškos iš viršaus į apačią metodą. Suranda tik maksimalius dažnus rinkinius [74, 75, 76].
6. *AprClique.* Algoritmas naudoja maksimalų uždarų grupių pseudoekvivalentumo ryšį ϕ_1. Naudoja horizontalų duomenų bazės išdėstymą. Jį galima padalinti į du etapus:
    a) Visi įmanomi maksimalaus elemento poaibiai kiekvienoje dalinėje gardelėje yra generuojami ir saugomi specialiuose maišos medžiuose (angl. *hash trees*), išvengiant dublikatų. Visiems *k*-rinkiniams yra skiriamas atskiras medis C_k. Vidinis *d* gylio medžio mazgas turi lentelę, kurios

elementai rodo į (d+1)-ąjį lygį. Visi rinkiniai yra saugomi medžio lapuose. Įterpimo į medį procedūra startuoja nuo medžio šaknies ir įterpia visus kandidatus į lapus.

b) Dažnumo paskaičiavimo žingsnis yra panašus kaip *Apriori* algoritme. Kiekvienai transakcijai $t \in D$ formuojami visi įmanomi k-poaibiai. Po to ieškomas tas poaibis medyje C_k ir, radus, atnaujinamas skaitiklis [77, 78, 79].

Taigi, duomenų bazė yra peržiūrima tik vieną kartą, generuojami visi dažni rinkiniai. Algoritmo pseudokodas pavaizduotas 18 paveiksle.

AprClique():
for visoms dalinėms gardelėms S_i, gautiems pagal sąryšį ϕ_1 **do**
  $R = \bigcup\{A_j \in S_i\}$**;**
  for visiems k>2 ir $k \le |R|$ **do**
    Įterpti kiekvieną k-poaibį i C_k*;*
end;
for visoms transakcijoms $t \in D$ **do**
  for visiems k-poaibiams s iš t, kai k>2 ir $k \le |t|$ **do**
    if $(s \in C_k)$ $s.count++$;
end;
$F_k = \{c \in C_k \mid c.count \ge min\_sup\}$;
Aibė visų dažnų rinkinių = $\bigcup_k F_k$;

18 paveikslas. *AprClique* algoritmo pseudokodas

***Apriori* ir *Partition* algoritmai**

***Apriori* algoritmas**. Šis algoritmas yra iteratyvus, kuris skaičiuoja tam tikro dydžio rinkinius duoto perėjimo per duomenų bazę metu. Procesas pradedamas, peržiūrint visas duomenų bazės transakcijas ir suradus visus dažnus elementus (1-rinkinius). Kiti potencialūs dažnų 2-rinkinių kandidatai generuojami iš prieš tai buvusių dažnų elementų. Kita duomenų bazės peržiūra nustato ir jų dažnumus. Rasti dažni 2-rinkiniai naudojami generuojant kito lygmens dažnų rinkinių kandidatus. Algoritmo pseudokodas pavaizduotas 19 paveiksle [80, 81, 82].

Šis algoritmas turi 3 pagrindinius žingsnius:

1) Generuojami *k*-dydžio kandidatai iš dažnų (*k*-1)-ojo dydžio rinkinių, apjungiant juos (vykdant sąjungą) pagal F_{k-1}. Pavyzdžiui, jeigu F_2={*AB, AC, AD, AE, BC, BD, BE*}, tai C_3={*ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE*}.
2) Pašalinami visi kandidatai, turintys nors vieną nedažną poaibį. Pavyzdžiui *ACD* bus pašalintas, kadangi *CD* yra nedažnas rinkinys. Po pašalinimo C_3={*ABC, ABD, ABE*}.
3) Peržiūrimos visos transakcijos, norint nustatyti kandidatų dažnumą. Kandidatai yra saugomi medyje, norint greičiau apskaičiuoti jų dažnumus.

F_1={dažni 1-rinkiniai};
for $(k = 2; F_{k-1} \neq \varnothing; k++)$ **do**
  C_k=aibė naujų kandidatų;
  for visoms transakcijoms $t \in D$ **do**
    for visiems *k*-poaibiams *s* iš *t* **do**
      if $(s \in C_k)$ *s.count*++;
  $F_k = \{c \in C_k \mid c.count \geq min\_sup\}$;
end;
Aibė visų dažnų rinkinių=$\bigcup_k F_k$;

19 paveikslas. *Apriori* algoritmo pseudokodas

***Partition* algoritmas**. Šis algoritmas logiškai suskaido horizontalią duomenų bazę į tam tikrą skaičių tarpusavyje nepersidengiančių dalių. Dalies dydis yra toks, kad jis tilptų į atmintį. Kiekviena dalis yra perskaitoma, kiekvienam elementui formuojamas vertikalus transakcijų sąrašas, t.y. visų transakcijų, kuriame šis elementas pasirodo, sąrašą. Tada, panaudojant transakcijų sąrašų sankirtas, generuojami visi lokaliai dažni rinkiniai. Visi lokaliai dažni rinkiniai apjungiami ir toliau atliekama visų dalių antroji peržiūra. Duomenų bazė vėl konvertuojama į vertikalią būseną ir atliekamas globalus visų išrinktų rinkinių apskaičiavimas. Taigi, duomenų bazė peržiūrima tiktai du kartus. Esmė yra ta, kad globaliai dažnas rinkinys privalo būti lokaliai dažnu bent vienoje dalyje. Šis algoritmas garantuoja visų dažnų rinkinių radimą [83, 84, 85].

2.9. Išvados

Ieškant dažnus poaibius, visa duomenų bazė yra suskaidoma į tarpusavyje nesusietas ir nepriklausomas dalines gardeles. Šiose gardelėse lygiagrečiai galima vykdyti dažnų poaibių paiešką iš apačios į viršų, iš viršaus į apačią bei hibridiniu būdu. Populiariausi paieškos algoritmai *Eclat*, *MaxEclat, Clique*, *MaxClique, TopDown*, *AprClique*, *Apriori* bei *Partition* naudoja vieną iš šių paieškos metodų. Pagrindinė dažnų poaibių radimo taisyklė yra ta, kad kiekvieno dažno poaibio visi vidiniai poaibiai irgi yra dažni. Tuo pačiu galime pastebėti, kad, jeigu poaibyje yra nors vienas nedažnas vidinis poaibis, tai mūsų poaibis irgi bus nedažnas. Ši taisyklė leidžia mums eliminuoti daugelį nedažnų poaibių net jų netikrinus.

3. Dažnų sekų radimas

3.1. Įvadas

Šiame skyriuje panagrinėsime dažnų sekų radimo pagrindinius principus. Pirmiausia apibrėžkime problemą su kuria susiduriama.

Tarkime $I = \{i_1, i_2, ..., i_m\}$ yra aibė *m* skirtingų objektų, kuriuos pavadinsime *elementais*. *Rinkinys* yra netuščia nesutvarkyta elementų sankaupa (neprarandant bendrumo, laikysime, kad elementai yra surikiuoti leksikografine tvarka). Rinkinys *i* yra apibrėžiamas kaip $(i_{j_1} i_{j_2} ... i_{j_k})$, kur i_{j_p} yra elementas. Rinkinys, turintis *k* elementų, vadinamas *k*-rinkiniu. *Seka* α yra apibrėžiama kaip sutvarkytas rinkinių sąrašas $(\alpha_1 \mapsto \alpha_2 \mapsto ... \mapsto \alpha_q)$, kur kiekvienas elementas α_j yra rinkinys. Seka $(\alpha_1 \mapsto \alpha_2 \mapsto ... \mapsto \alpha_q)$ vadinama *k*-seka, jeigu jos dydis lygus *k*, t.y. $k = \sum_j |\alpha_j|$. Pavyzdžiui, seka $(B \mapsto AC)$ yra 3-seka. Elementas rinkinyje gali pasirodyti tik vieną kartą, bet gali daug kartų pasirodyti kituose sekos rinkiniuose [86, 87].

Seka $\alpha = (\alpha_1 \mapsto \alpha_2 \mapsto .. \mapsto .\alpha_n)$ yra kitos sekos $\beta = (\beta_1 \mapsto \beta_2 \mapsto ... \mapsto \beta_m)$ *posekis* (žymima $\alpha \preccurlyeq \beta$), jeigu egzistuoja tokie sveiki skaičiai $i_1 < i_2 < ... < i_n$, kad $\alpha_j \subseteq \beta_{i_j}$ visiems α_j. Pavyzdžiui seka $(B \mapsto AC)$ yra sekos $(AB \mapsto E \mapsto ACD)$ posekis, kadangi pradinės sekos elementai $B \subseteq AB$ ir $AC \subseteq ACD$. Antra vertus, seka $(AB \mapsto E)$ nėra sekos (*ABE*) posekis. Sakoma, kad α yra sekos β *taisyklingas posekis* (žymima $\alpha \prec \beta$, jeigu $\alpha \preccurlyeq \beta$ ir $\beta \not\preccurlyeq \alpha$). Seka vadinama *maksimalia*, jeigu ji nėra jokios kitos sekos posekis. Posekis, turintis dydį *k*, vadinamas *k*-posekiu [88,89].

Transakcija *T* turi unikalų identifikatorių ir turi savyje elementų aibę, t.y. $T \subseteq I$. Vartotojas *C* turi unikalų identifikatorių ir yra susietas su transakcijų sąrašu $\{T_1, T_2, ..., T_n\}$. Laikoma, kad joks vartotojas vienu laiko momentu negali turėti daugiau nei vieną transakciją. Taigi transakcijos laiką galima laikyti transakcijas identifikatoriumi. Laikoma taip pat, kad vartotojų transakcijų sąrašas yra surikiuotas pagal transakcijų laiką ir žymimas $T_1 \mapsto T_2 \mapsto ... \mapsto T_n$, kurį vadinsime *vartotojų seka*. Duomenų bazė *D* ir yra sudaryta iš tokių vartotojų sekų [90, 91].

3.2. Sekos taisyklės

Sakoma, kad vartotojo seka *C turi savyje* seką α, jeigu $\alpha \preccurlyeq C$, t. y. jeigu α yra vartotojų sekos *C* posekis. Sekos *pasikartojamumas*, arba *dažnumas* $\sigma(\alpha)$ yra bendras vartotojų, kurie turi šią seką, skaičius. Iš anksto nustatytas slenkstis vadinamas *minimaliu dažnumu* (angl. *minimum support*) ir žymimas *min_sup*. Laikoma, kad seka yra dažna, jei ji pasikartoja daugiau negu *min_sup* kartų. Aibė dažnų *k*-sekų žymima F_k [92, 93].

Duomenų bazė

Vartotojo identifikatorius	Transakcijos laikas	Elementai
1	10	*C D*
1	15	*A B C*
1	20	*A B F*
1	25	*A C D F*
2	15	*A B F*
2	20	*E*
3	10	*A B F*
4	10	*D G H*
4	20	*B F*
4	25	*A G H*

Dažnos sekos

Dažnos 1-sekos

A	4
B	4
D	2
F	4

Dažnos 2-sekos

AB	3
AF	3
$B \mapsto A$	2
BF	4
$D \mapsto A$	2
$D \mapsto B$	2
$D \mapsto F$	2
$F \mapsto A$	2

Dažnos 3-sekos

ABF	3
$BF \mapsto A$	2
$D \mapsto BF$	2
$D \mapsto B \mapsto A$	2
$D \mapsto F \mapsto A$	2

Dažnos 4-sekos

$D \mapsto BF \mapsto A$	2

20 paveikslas. Pradinė vartotojų sekų duomenų bazė *D*

Vartotojų sekų duomenų bazėje *D* su duotu minimaliu dažnumu *min_sup* reikia ištirti sekų šablonus, t. y. rasti visas dažnas sekas duomenų bazėje. 20 paveiksle pateiktoje duomenų bazėje (ji bus naudojama kaip pavyzdys ir toliau) yra aštuoni elementai (nuo *A* iki *H*), keturi vartotojai ir dešimt transakcijų visiems vartotojams. Paveiksle taip pat parodytos visos dažnos sekos, esant minimaliam dažnumai 50% arba pasirodymui 2-uose vartotojuose. Šiame pavyzdyje unikali maksimali dažna seka yra $D \to BF \to A$.

Apibrėžkime pagrindinį taisyklių generavimo algoritmą, kurio pagalba galėsime generuoti visas sąlygas, tenkinančias tam tikras taisykles [94, 95].

RuleGen(*F*, *min_conf*):
  for visoms dažnoms sekoms $\beta \in F$ **do**
    for visiems posekiams $\alpha \prec \beta$ **do**
      $conf = fr(\beta) / fr(\alpha)$;
      if ($conf \geq min\_conf$) **then**
        rezultatas yra taisyklė $\alpha \Rightarrow \beta$ ir *conf*;

21 paveikslas. Taisyklių generavimo algoritmas

Kai tik dažnos sekos yra žinomos, galima nustatyti tam tikras taisykles, kurios aprašo santykius tarp skirtingų elementų sekose. Pavyzdžiui, seka (*BF*) pasirodo keturiuose vartotojuose, kai tuo tarpu (*ABF*) tik trijuose. Taigi, galima sakyti, kad, pasirodžius kartu *BF*, yra 75% galimybė kartu pasirodyti ir *A*. Kitaip tariant, sakoma, kad susietumo taisyklė $(BF) \Rightarrow (BFA)$ turi 75% tikrumą (angl. *confidence*). Kita pavyzdžio susietumo taisyklė $(D \mapsto BF) \Rightarrow (D \mapsto BF \mapsto A)$ turi 100% tikrumą. Turint iš anksto nustatytą minimalų tikrumą *min_conf*, galima generuoti visas sąlygas, tenkinančias taisykles, pasinaudojant paprastu algoritmu pavaizduotu 21 paveiksle [42].

3.3. Sekų nustatymas gardelės metodu

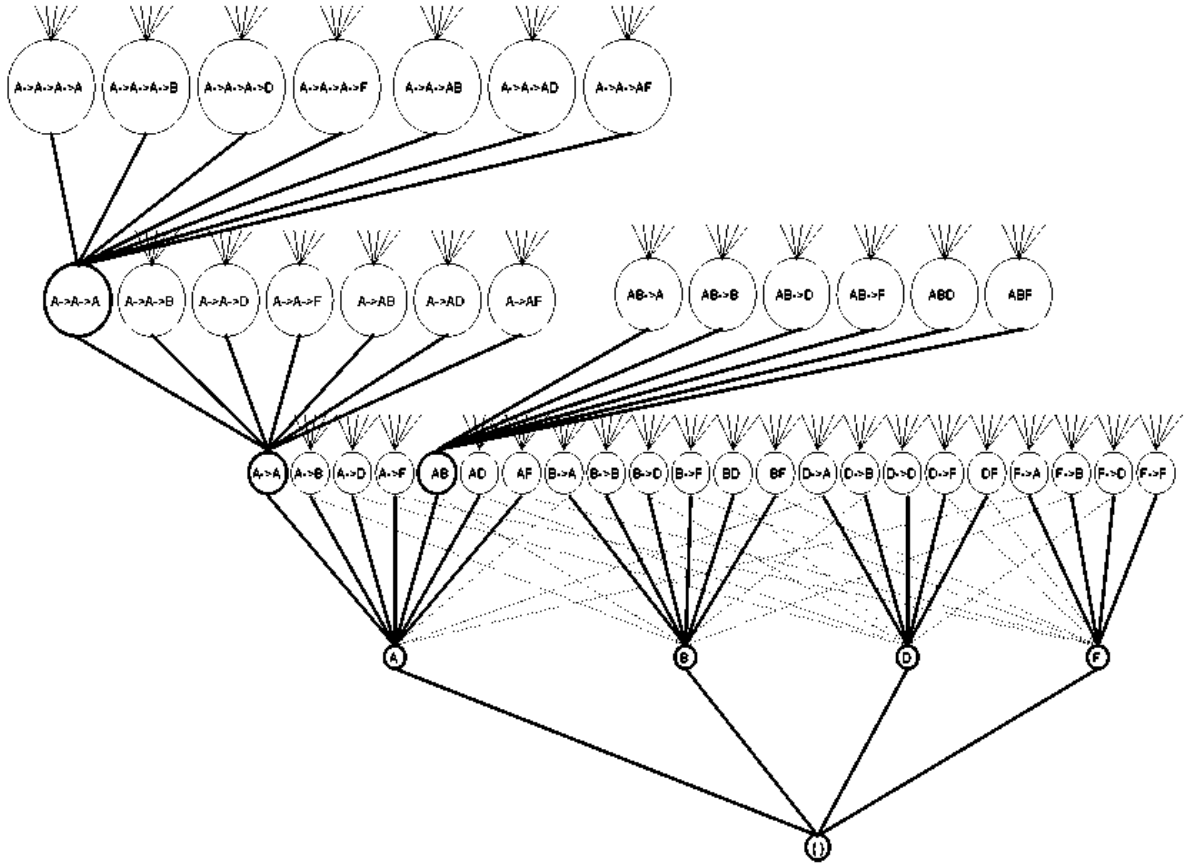
10 lema. *Tegul S yra visų sekų, sudarytų iš I elementų, aibė. Tada posekių sąryšis* $\preccurlyeq$ *apibrėžia dalinį sutvarkymą aibėje S [96].*

1 teorema. *Jei I yra elementų aibė, tai visų įmanomų sekų, sudarytų iš I elementų, sutvarkyta aibė S yra išsami gardelė, kurioje sujungimai ir susikirtimai gaunami per sąjungas ir sankirtas atitinkamai:*

$$\vee\{A_i \mid i \in I\} = \bigcup_{i \in I} A_i \text{ ir } \wedge\{A_i \mid i \in I\} = \bigcap_{i \in I} A_i$$

Nagrinėjamame pavyzdyje $F_1 = \{A, B, D, F\}$. 22 paveiksle pavaizduota sekų gardelė *S*, kuri yra sudaryta pagal 4-ių elementų, kurie yra atomai, posekių santykius. Pats apatinis sekų gardelės elementas yra $\perp$={}, o viršutinis elementas yra neapibrėžtas, taigi abstrakčiu lygmeniu galima teigti, kad šis tinklelis yra begalinis. Paveiksle pavaizduota išsami 2-sekų aibė, tos 3-sekos, kurios gali būti sugeneruotos iš $A \mapsto A$ bei *AB*, ir visos įmanomos 4-sekos, kurios gali būti sugeneruotos iš sekos $A \mapsto A \mapsto A$. Rekursinė kombinatorinė gardelės struktūra yra akivaizdi. Pavyzdžiui, panagrinėkime aibę sekų, sugeneruotų iš elemento *A*, ir

seką $A \mapsto A$. Šios abi aibės yra identiškos, išskyrus papildomą priešdėlį $A \mapsto$ antroje aibėje [97].



22 paveikslas. Išsami sekų gardelė

11 lema. *Tarkime n yra visų dažnų elementų kiekis. Tada bendras k-sekų kiekis bus lygus* $\sum_{i_1=1}^{k}\binom{n}{i_1}\sum_{i_2=1}^{k-n_i}\binom{n}{i_2}\ldots\sum_{i_k=1}^{k-i_1-\ldots-i_{k-1}}\binom{n}{i_k}$ [98].

Įrodymas.

Apskaičiuojamas kelių, kuriais gali būti sukonstruota *k*-seka, kiekis ir po to kiekvienai kombinacijai paskiriami elementai. Kelių, kuriais gali būti sukonstruota *k*-seka, skaičius yra lygus skaičiui būdų, kuriais gali būti gautas skaičius *k* kaip sveikųjų skaičių suma. 2 lentelėje parodyti būdai, kuriais gaunamas skaičius 4 [99].

2 lentelė. 4-sekos gavimo variantai

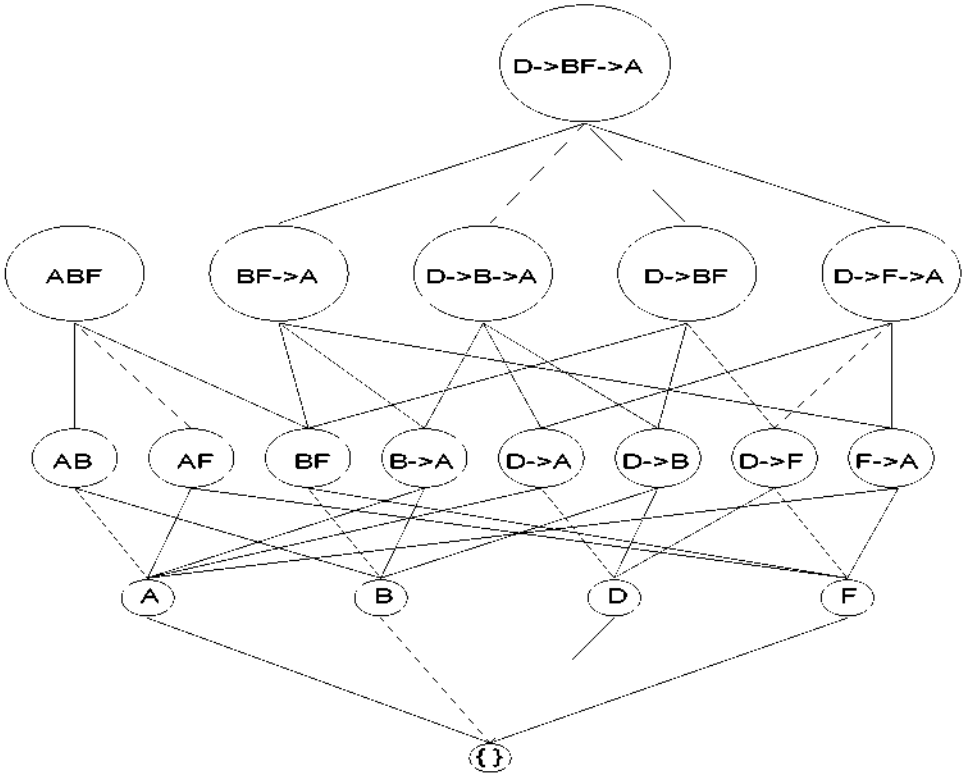
i_1	i_2	i_3	i_4
1	1	1	1
1	1	2	
1	2	1	
1	3		
2	1	1	
2	2		
3	1		
4			

Sumos sveikieji dėmenys yra interpretuojami kaip rinkinių, sudarančių *k* ilgio seką, dydžiai. Po to kiekvienam tokiam rinkiniui priskiriami elementai. Rinkiniui, kurio dydis *i*, yra $\binom{n}{i}$ tokių priskyrimų. Dauginant pasirinkimus kiekvienam variantui ir sudedant variantus, gaunamas bendras *k*-sekų kiekis [46].

Kaip buvo minėta aukščiau, abstrakcijos lygiu gardelė yra begalinė. Laimei, praktiškai ji yra apribota. Sekos elementų (rinkinių) skaičius yra apribotas iš viršaus maksimaliu transakcijų skaičiumi vartotojui (tarkime *C*). Kadangi rinkinio dydis irgi yra apribotas iš viršaus maksimaliu transakcijos elementų kiekiu (tarkime *T*), tai seka turės daugiausia $C \cdot T$ elementų. Taigi posekių gardelė neviršys $C \cdot T$ elementų. Mūsų pavyzdyje *C*=4 ir *T*=4, vadinasi maksimaliai įmanoma seka turės 16 elementų.

Visais praktiniais atvejais ne tik gardelės dydis yra apribotas, bet ir pati dažnų sekų aibė yra labai reta (priklausomai nuo *min_sup* reikšmės). Pavyzdžiui, 23 paveiksle pavaizduota gardelė, sukurta pagal dažną maksimalią seką $D \mapsto BF \mapsto A$. Pažymėkime aibę atomų A, kuri yra sudaryta iš dažnų elementų {*A, B, D, F*}. Akivaizdu, kad visų dažnų sekų aibė formuoja susikirtimų pusinę gardelę, kadangi ji yra uždara susikirtimų operacijos požiūriu, t. y., jeigu *X* ir *Y* yra dažnos sekos, tai $X \cap Y$ irgi yra dažnas. Tačiau iš kitos pusės negalima teigti, kad, jeigu *X* ir *Y* yra dažnos sekos, tai ir $X \cup Y$ bus dažnas. Uždarumas susikirtimo operacijai leidžia formuluoti tokią lemą [100].

20 lema. *Visi dažnos sekos posekiai yra dažni.*



23 paveikslas. Sekų gardelė pagal maksimalią dažnumo seką $D \mapsto BF \mapsto A$

Ši lema nurodo, jog reikėtų sutelkti dėmesį tik į tas sekas, kurių posekiai yra dažni. Tai bus naudinga ateityje, kai reikės eliminuoti iš galimų dažnų sekų tas, kurios turės nors vieną nedažną posekį [85, 86].

3.4. Dažnumo skaičiavimas

Su kiekvienu atomu *X* yra susiejamas jo identifikacinis sąrašas (*id_list*), kuris žymimas dar ir *L(X)*. Šį sąrašą sudaro vartotojo sąrašo (*cid*) ir transakcijų sąrašo (*tid*) elementų poros. 24 paveiksle pavaizduotos šios poros [101].

A	
cid	*tid*
1	15
1	20
1	25
2	15
3	10
4	25

B	
cid	*tid*
1	15
1	20
2	15
3	10
4	20

D	
cid	*tid*
1	10
1	25
4	10

F	
cid	*tid*
1	20
1	25
2	15
3	10
4	20

24 paveikslas. Atomų identifikaciniai sąrašai

Kaip matosi iš 20 paveikslo, atomas *D* pasirodo šiose transakcijų ir vartotojų porose: {(1, 10), (1, 25), (4, 10)}. Ši ir yra atomo *D* porų sąrašas.

21 lema.

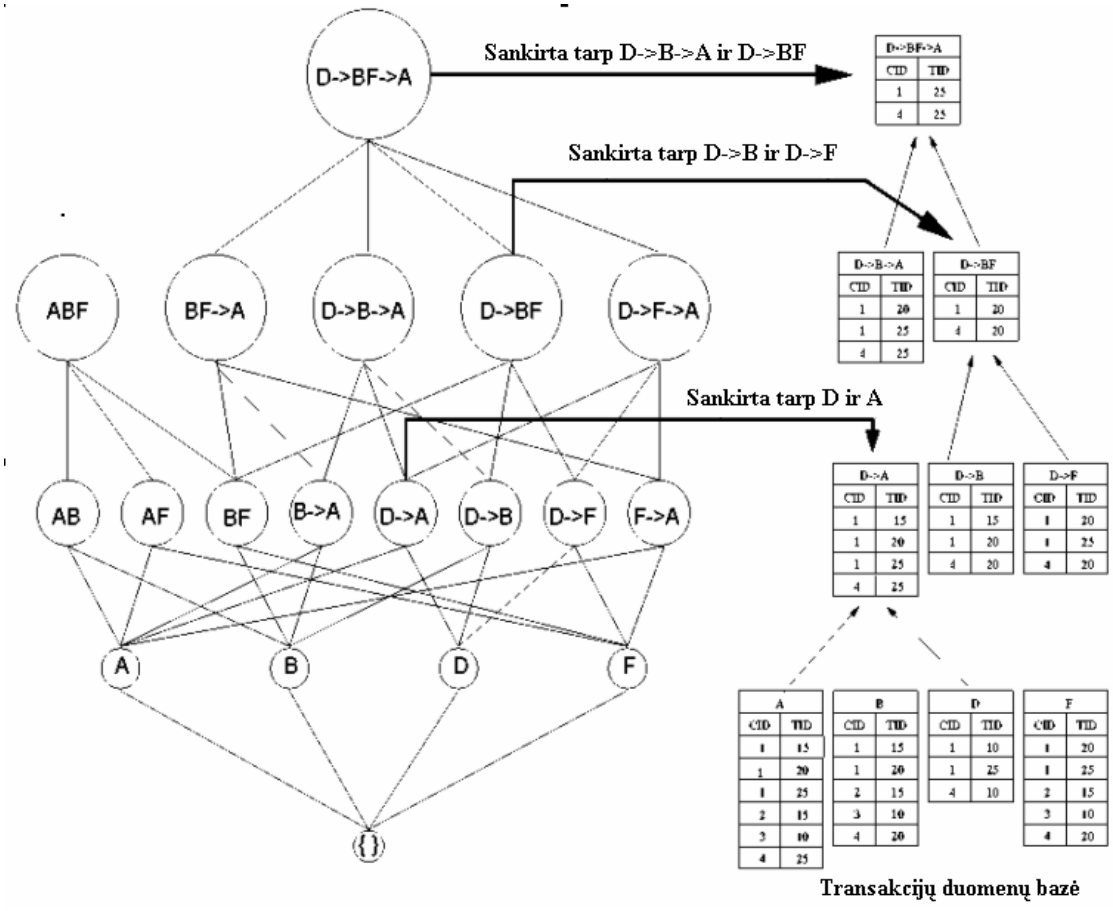
Tegul $J = \{Y \in A(S) \mid Y \leq X\}$ *kiekvienam* $X \in S$. *Tada* $X = \bigcup_{Y \in J} Y$ ir $\sigma(X) = |\bigcap_{Y \in J} L(Y)|$.

Remiantis šia lema galime teigti, kad kiekvieną seką iš *S* galima gauti, apjungiant tam tikrus gardelės atomus, o sekos dažnumas gali būti nustatomas, atliekant tų atomų transakcijų sąrašų sankirtas. Ši lema skirta tik gardelės tinklelio atomams. Apibendrinimą sekų aibėms išreiškia 22 lema.

22 lema.

Tegul $X = \bigcup_{Y \in J} Y$ *kiekvienam* $X \in S$. *Tada* $\sigma(X) = |\bigcap_{Y \in J} L(Y)|$ [95].

Ši lema sako, kad, jeigu *X* duota kaip sekų iš *Y* aibių sąjunga, tai dažnumas nustatomas, atliekant elementų iš *Y* transakcijų sąrašų sankirtas.



25 **paveikslas.** Dažnumo skaičiavimas per transakcijų sąrašų sankirtas

Praktiškai galima nustatyti kiekvienos *k*-sekos dažnumą, atliekant bet kurių dviejų (*k*-1) ilgio posekių sankirtą. Patikrinus rezultatą, galima bus nuspręsti, ar seka yra dažna ar ne. Šis procesas pavaizduotas 25 paveiksle. Jame yra vertikali (transakcijų) duomenų bazę su transakcijų sąrašu kiekvienam atomui. Tarpinis transakcijos sąrašas elementui $D \mapsto A$ yra gaunamas, atliekant atomo D ir atomo A transakcijų sąrašų sankirtą, t.y. $L(D \mapsto A) = L(D) \cap L(A)$.

Analogiškai apskaičiuojama $L(D \mapsto BF \mapsto A) = L(D \mapsto BF) \cap L(D \mapsto B \mapsto A)$ ir t.t. Tokiu būdu, tik leksikografiškai du pirmieji prieš tai buvusio lygio posekiai reikalingi, norint apskaičiuoti duotojo lygio sekos dažnumą.

23 lema.

Tarkime X ir Y yra dvi sekos ir $X \preccurlyeq Y$. *Tada* $L(X) \supseteq L(Y)$ [95,96].

Ši lema sako, kad, jei seka *X* yra sekos *Y* posekis, tai sekos *Y* dažnumas transakcijų duomenų bazėje yra mažesnis arba lygus sekos *X* dažnumui.

Tokiu būdu visada galima gana greitai nustatyti einamojo lygmens (sekų ilgis *k*) dažnas sekas, pasinaudojant posekių dažnumu iš prieš tai buvusio lygmens (sekų ilgis *k*-1).

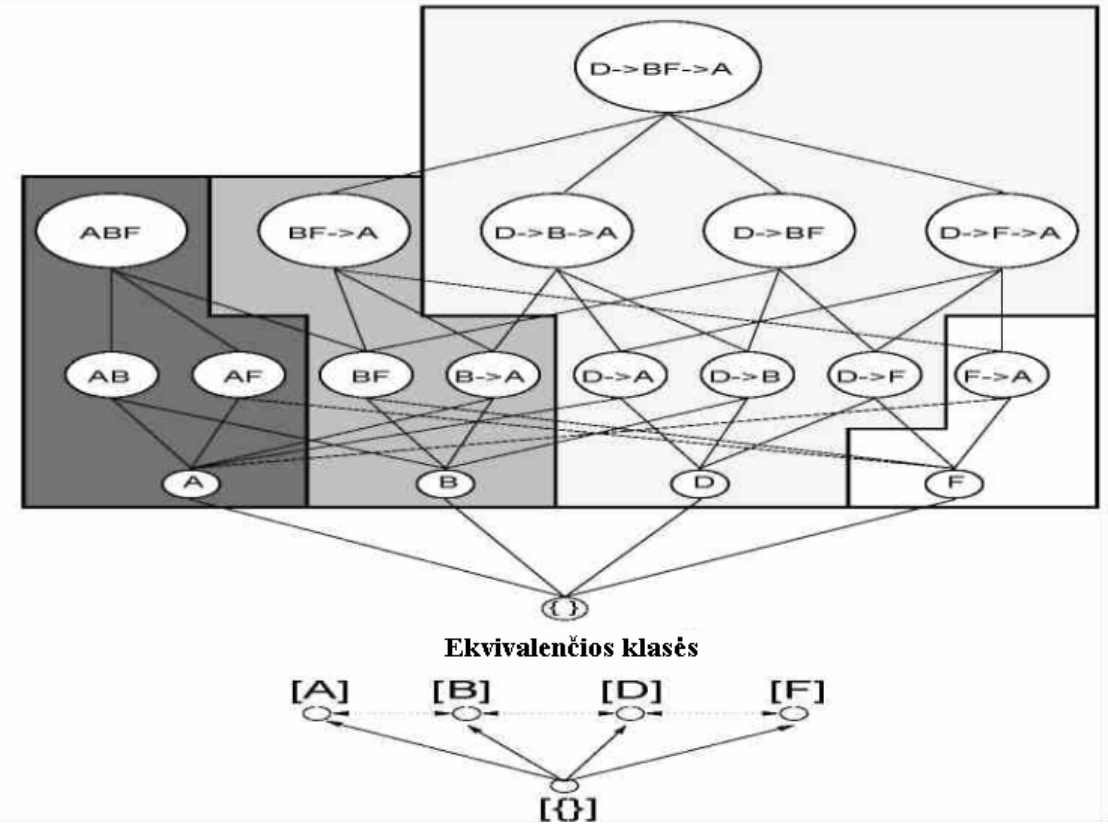
3.5. Gardelių dekompozicija. Priesaginės klasės

Jeigu būtų pakankamai atminties, būtų galima sunumeruoti visas dažnas sekas, pereinant per gardelę ir, atliekant sankirtas, nustatyti sekų dažnumą. Tačiau praktiškai yra pagrindinės atminties ribojimai ir patalpinti visas įmanomus sankirtas ir tarpinius dažnumus yra neįmanoma. Tokiu atveju kyla natūralus klausimas: ar galima suskaidyti mūsų pradinę gardelę į kelias smulkesnes dalis taip, kad jas galima būtų saugoti pagrindinėje atmintyje ir apdoroti (tikrinti dažnumą) nepriklausomai viena nuo kitos. Toliau tai ir nagrinėjama.

Apibrėžkime funkciją $p: S \to S$, čia $p(X, k)=X[1:k]$. Kitaip sakant, funkcija $p(X, k)$ grąžina sekos *X* *k* ilgio priesagą. Apibrėžkime ekvivalentumo sąryšį Θ_k gardelėje *S*, šitaip: $\forall X, Y \in S$ sakoma, kad *X* susietas su *Y* pagal Θ_k ir žymima $X \equiv_{\Theta_k} Y$, tada ir tik tada, jeigu $p(X, k)=p(Y, k)$. T.y. dvi sekos priklauso tai pačiai klasei, jei jos turi tą pačią *k* dydžio priesagą [102].

26 paveiksle pavaizduota gardelė pagal ekvivalentumo sąryšį Θ_1. Jame visos sekos yra suskirstytos pagal priesagas į ekvivalentumo klases. To suskirstymo rezultatas yra

ekvivalentumo klasių aibė {[*A*], [*B*], [*D*], [*F*]}. Paveikslo apačioje taip pat parodyti ryšiai tarp šių keturių klasių.



26 paveikslas. Gardelės *S* ekvivalentumo klasės pagal sąryšį Θ_1

Šie ryšiai rodo atmetimo informaciją. Kitaip tariant, norint atmesti seką (atmetimas vyksta tada, kai ji turi nors vieną nedažną posekį) reikia turėti informaciją apie klasių persikirtimą. Tai bus nagrinėjama vėliau.

24 lema.

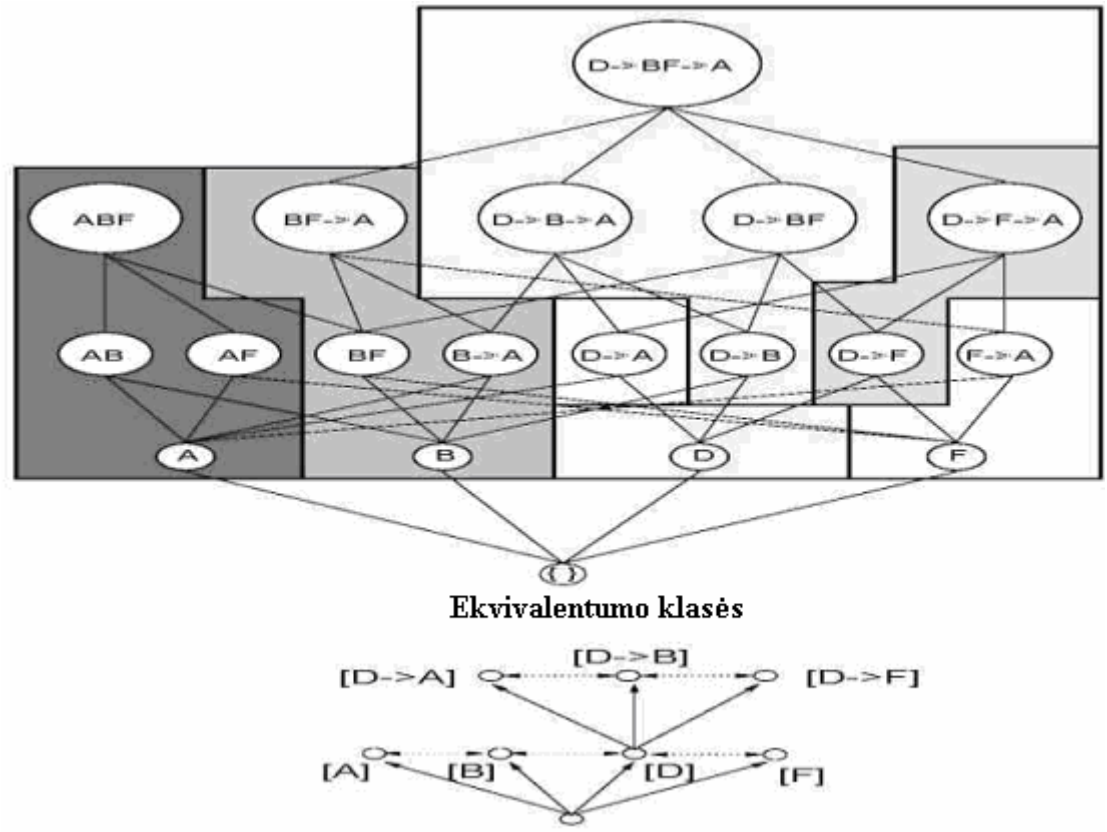
Kiekviena ekvivalentumo klasė $[X]_{\Theta_k}$, *sukurta pagal ekvivalentumo sąryšį* Θ_k, *yra gardelės S dalinė gardelė* [55,56].

Įrodymas.

Tarkime *U* ir *V* yra bet kokie du elementai klasėje [*X*], t. y. jie abu turi tą pačią priesagą *X*. Iš $U \vee V = U \cup V \supseteq X$ išplaukia, kad $U \vee V \in [X]$, o iš $U \wedge V = U \cap V \supseteq X$ išplaukia, kad $U \wedge V \in [X]$. Vadinasi, $[X]_{\Theta_k}$ yra gardelės *S* dalinė gardelė [55, 56].

Kiekviena ekvivalentumo klasė $[X]_{\Theta_1}$ yra gardelė su savo nuosava atomų aibe. Pavyzdžiui, $[D]_{\Theta_1}$ atomai yra $\{D \mapsto A, D \mapsto B, D \mapsto F\}$, o apatinis elementas $\perp = D$. Pagal 21 ir 22 lemas galima generuoti kiekvienos klasės (dalinės gardelės) visus sekų dažnumus,

atliekant atomų transakcijų sąrašų arba bet kurių dviejų ankstesniojo lygmens posekių sankirtas. Jeigu yra pakankamai atminties saugoti laikinus transakcijų sąrašus kiekvienai klasei, tai kiekvieną $[X]_{\Theta_1}$ galima apdoroti nepriklausomai.



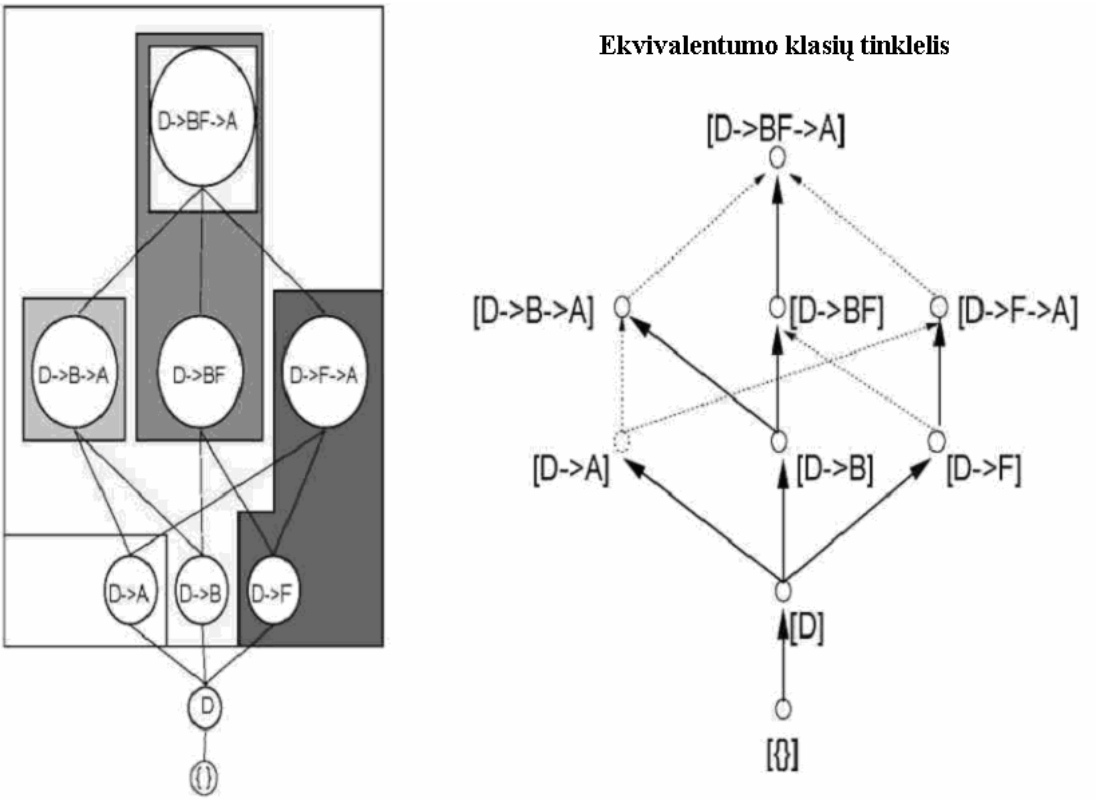
27 **paveikslas.** Gardelės $[D]_{\Theta_1}$ ekvivalentumo klasės pagal sąryšį Θ_2

Praktiškai buvo nustatyta, kad vieno lygmens dekompozicija pagal sąryšį Θ_1 yra pakanka. Tačiau iš kitos pusės gali gautis, kad pirmojo lygmens klasė gali būti per didelė talpinti į pagrindinę atmintį. Tokiu atveju pagal tą patį principą gali būti taikoma rekursinė dekompozicija. Tarkime, kad klasė [*D*] (gauta, kai gardelei *S* buvo pritaikytas sąryšis Θ_1) yra per didelė. Kadangi ji irgi yra gardelė, galima atlikti jos dekompoziciją, naudojant sąryšį Θ_2. 27 paveikslas vaizduoja klases, gautas atlikus klasės [*D*] dekompoziciją pagal sąryšį Θ_2. Kiekviena iš gautų 6 klasių [*A*], [*B*], $[D \mapsto A]$, $[D \mapsto B]$, $[D \mapsto F]$, [*F*] gali būti nagrinėjama nepriklausomai viena nuo kitos. Tokiu būdu galima skaidyti visas klases į mažesnis tol, kol kiekviena klasė bus pakankamai maža, kad tilptų į pagrindinę atmintį.

3.6. Dažnų sekų paieška

Šioje dalyje panagrinėjamos efektyvios paieškos strategijos, kai išvardinamos visos dažnos sekos kiekvienai klasei. Nagrinėjamos dvi strategijos: *paieška platyn* (angl. *breadth-*

first search) ir *paieška gilyn* (angl. *depth-first search*). Abu šie metodai taiko rekursinę dekompoziciją skaidant klases į smulkesnes pagal ekvivalentumo sąryšį Θ_k. 28 paveiksle pavaizduota klasės $[D]_{\Theta_1}$ dekompozicija į smulkesnes klases ir galutinė ekvivalentumo klasės gardelė [57, 58].



28 paveikslas. Rekursyvinė klasės [*D*] dekompozicija pagal sąryšį Θ_k

Paieška platyn. Šios strategijos pagrindinė idėja yra ta, kad ekvivalentumo klasių gardelė yra generuojama rekursyviai taikant Θ_k sąryšį, pradedant nuo apačios ir einant į viršų. Apdorojamos visos einamojo lygmens klasės ir tik po to pereinama prie kito lygmens. 28 paveiksle iš pradžių patikrinamos klasės $\{[D \mapsto A], [D \mapsto B], [D \mapsto F]\}$, o tik po to pereinama į aukštesnio lygmens klases $\{[D \mapsto B \mapsto A], [D \mapsto BF], [D \mapsto F \mapsto A]\}$ ir t. t. [57, 58].

Paieška gilyn. Šios strategijos pagrindinė idėja yra, ta, kad apdorojimas vyksta visų tos pačios klasės elemento sekų ir tik tada pereinama prie kitos to paties lygmens sekos. Mūsų pavyzdyje sekos bus analizuojamos tokia tvarka: $[D \mapsto A]$, $[D \mapsto B]$, $[D \mapsto B \mapsto A]$, $[D \mapsto BF]$, $[D \mapsto BF \mapsto A]$ ir t. t.

Paieškos platyn privalumas prieš paiešką gilyn yra tas, kad čia gaunama daugiau informacija nedažnoms sekoms atmesti. Pavyzdžiui, yra žinomos visos 2-sekas prieš

konstruojant 3-sekas, o paieškoje gilyn ši informacija yra nežinoma. Antra vertus, paieškos platyn metodas reikalauja daugiau atminties [59, 60].

3.7. Išvados

Šiame skyriuje buvo apibrėžta sekos sąvoka bei išaiškinti populiariausi dažnų sekų radimo būdai: paieška platyn ir paieška gilyn. Ieškant dažnas sekas būtina žinoti vartotojo nustatytą pradinį dažnumo „slenksnį“ *min_sup*. Laikoma, kad seka yra dažna tada ir tik tada, kai jos dažnumas didesnis už minimalų dažnumą *min_sup*. Viena pagrindinė dažnų sekų radimo taisyklė yra ta, kad kiekvienos dažnos sekos visi posekiai irgi yra dažni. Tuo pačiu galime pastebėti, kad, jeigu sekoje yra nors vienas nedažnas vidinis posekis, tai mūsų seka irgi bus nedažnas. Ši taisyklė leidžia mums eliminuoti daugelį nedažnų sekų net jų netikrinus.

4. Duomenų paieškos algoritmai

4.1. Įvadas

Mūsų pagrindinė nagrinėjama problema yra dažnų sekų paieška dideliuose duomenų masyvuose–tekstuose. Tai atskiras atvejis iš visos *data mining* sprendžiamos problemos.

Labai dažnai reikia nustatyti, kokios simbolių sekos dažniausiai pasitaiko duomenų bazėje. Ši problema yra aktuali daugeliu atvejų. Tarkime, turime populiarų interneto puslapį, kuriame kiekvienas vartotojas privalo užsiregistruoti, pateikdamas sistemai tam tikrus duomenis apie save. Tokiu atveju labai pravartu sužinoti, kokios simbolių sekos yra dažniausiai pasikartojančios tam, kad vėliau padaryti puslapį labiau lankstesniu arba automatizuoti tam tikras jo funkcijas, kurios padarytų jį patogesniu vartotojui. Kitu pavyzdžiu gali būti dažniausiai pasikartojančių IP adresų ar adresų grupės, kuriais vartotojai jungiasi prie vieno ar kito internetinio portalo, paieška. Tokiu būdu galėtumėme sužinoti ir analizuoti puslapį aplankančių vartotojų geografiją. Be to, tokia sekų paieška yra labai naudinga analizuojant prekių srautus, nustatant tam tikrų prekių rūšių paklausą ar jų perkamumą per tam tikrą laiko intervalą. Ateityje galima būtų nagrinėti literatūrinius kūrinius, siekiant pagal dažniausiai pasikartojančius žodžius ar žodžių junginių nustatyti jų autorystę.

Tarkime turime m simbolių. Tegul simbolių seka yra sudaryta iš k elementų. Tada potencialių paieškos sekų gali būti m^k. Tai pakankamai didelis kiekis. Todėl, nustatant dažnas sekas yra labai svarbu analitiškai atmesti tą sekų grupę, kuri tikrai bus nedažna.

4.2. Sekų struktūros

Priminsime, kad $L = \{i_1, i_2, ..., i_m\}$ yra aibė sudaryta iš m skirtingų elementų (angl. *items*). Tam tikras netuščias elementų kombinacijas pavadinsime **rinkiniais** (angl. *itemset*). Rinkinį žymėsime kaip $(i_{j_1} i_{j_2} ... i_{j_k})$, čia i_j yra elementai. Laikoma, kad visi elementai rinkiniuose atsiranda tuo pačiu metu.

Tam tikra sutvarkyta rinkinių grupė vadinama **seka** (angl. *sequence*). Seką α žymėsime kaip $(a_1 \mapsto a_2 \mapsto ... \mapsto a_q)$, čia kiekvienas sekos narys α_j yra rinkinys. Seka sudaryta iš k elementų $(k = \sum_j |\alpha_j|)$ vadinama *k-seka*. Pavyzdžiui, $(B \mapsto ABC)$ yra 4-seka. Toliau bus nagrinėjami atskiri sekų atvejai, kai $|\alpha_j| = 1$ visiem j ir elementų aibė yra simbolių aibė [11,12, 13].

Seka $\alpha = (a_1 \to a_2 \to ... \to a_n)$ yra kitos sekos $\beta = (\beta_1 \mapsto \beta_2 \mapsto ... \mapsto \beta_m)$ *posekis* (žymima $\alpha \subseteq \beta$), jei egzistuoja sveiki $l_1 < l_2 < l_3 < ... l_n$, tokie, kad $\alpha_j \subseteq \beta_{i_j}$ visiems α_j. Pavyzdžiui, seka $(B \mapsto AC)$ yra posekė sekos $(AB \mapsto E \mapsto ACD)$ posekis, kadangi $B \subseteq AB$, o $AC \subseteq ACD$. O seka ($AB \mapsto C$) nėra sekos(*ABC*) posekis. Sakoma, kad seka α yra sekos β *būdingas posekis* (žymima $\alpha \subset \beta$), jei $\alpha \subseteq \beta$ ir $\beta \not\subset \alpha$. Seka yra *maksimaliai būdinga*, jei ji nėra kokios nors kitos sekos posekis [15, 17, 21].

Sakysime, kad vartotojo seka *C* turi savyje seką α, jeigu $\alpha \subseteq C$, t.y. jeigu α yra vartotojo sekos *C* posekis. Pasirodymo dažnumas, žymimas $\sigma(\alpha)$, yra vartotojų, kurie turi šią seką, bendras skaičius. Iš anksto apibrėžtas skaičius, nusakantis *minimalų dažnumą***,** žymimas *min_sup*.

Toliau bus nagrinėjami atskiri sekų atvejai, kai $|\alpha_j| = 1$ visiem *j* ir elementų aibė yra simbolių aibė.

Pavyzdžiui, tarkime, kad aibę sudaro du elementai: simboliai *A* ir *B*, t.y. *L=(A,B)*. Tarkime, kad yra dvi šių elementų kombinacijos (*AAB)* ir (*ABA)*. Nors elementų kiekis bei patys elementai sutampa, bet yra skirtingas jų išsidėstymas, todėl tai yra dvi skirtingos sekos.

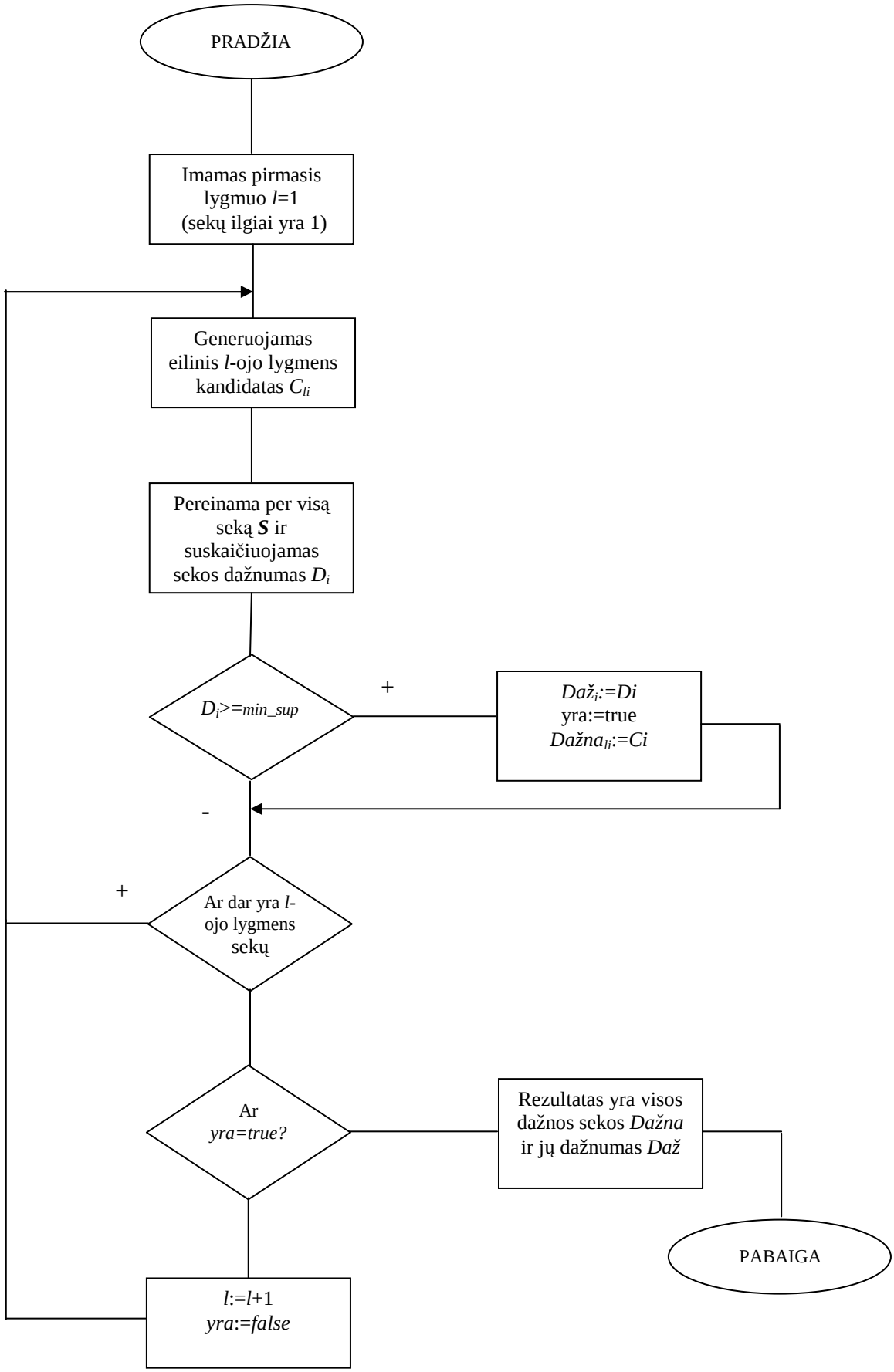
Seka vadinama dažna, jei ji pasikartoja nagrinėjamoje elementų (mūsų atveju simbolių) aibėje nemažiau kartų už duotą konstantą *min_sup*. Šį konstanta yra viena iš pradinių paieškos sąlygų. Pavyzdžiui, sakysime, kad seka yra dažna, jei ji pasikartoje nemažiau kaip **6** kartus. Vadinasi, mūsų minimalus dažnumas *min_sup=6*.

Sakysime, kad seka yra *k* ilgio arba *k*-ojo lygmens, jei ji sudaryta iš *k* elementų. Pavyzdžiui seka (*ABAA)* turi ilgį 4, t.y. yra 4-ojo lygmens.

Panagrinėsime kelis algoritmus, kurie ieško dažnų simbolių sekų, didelėse duomenų bazėse.

4.3. Tiesioginės paieškos algoritmas

Tarkime turime duomenų bazę *S*, sudaryta iš simbolių. Mūsų tikslas yra surasti visas simbolių sekas, kurios yra dažnos. Minimalus dažnumas *min_sup* yra sveikasis skaičius, kuris bus traktuojamas, kaip pradinis duomuo. Tiesioginės paieškos algoritmo schema pavaizduota 29 paveiksle.



29 paveikslas. Tiesioginės paieškos algoritmo blokinė schema

Dažnų sekų paieška pradedama nuo pirmoje lygmens, t.y. nagrinėjamos sekos, kurių ilgis lygus 1. Pavyzdžiui, *A*, *B*, *C* ir t.t. Paėmus pirmojo lygmens (l=1) seką, pereinama per visą duomenų bazę ir nustatomas jos dažnumas. Jei seka yra dažna (jos pasirodymo dažnumas nemažesnis nei duotas pradinis dažnumas), fiksuojama, kad šiame lygmenyje yra dažnų sekų, todėl privalu nagrinėti ir kito aukštesnio lygmens sekas. Išnagrinėjus vieną einamojo lygmens seką, pereinama prie kitos ir t.t. Išnagrinėjus visas einamojo lygmens sekas, jei buvo nors viena dažna seka, tai pereinama prie kito lygmens ((l+1)-ojo lygmens). Taip tęsiama, kol surandamas lygmuo, kuriame nėra dažnų sekų. Sutartiniai žymėjimai:

S – duomenų bazė;

L – lygmuo;

C_{li} – esamo lygmens nagrinėjama seka (kandidatas);

Di – einamosios sekos dažnumas;

min_sup – minimalus dažnumas;

$Daž_i$ – einamosios sekos dažnučasi;

$Dažna_i$ – dažnų sekų masyvas.

Panagrinėkime konkretų pavyzdį ir paieškokime dažnų sekų duomenų bazėje.

Tarkime, kad duomenų bazėje yra simboliai **ABAABBABBABBABBBBABB**. Reikia nustatyti dažnas seka. Pasikartojančios sekos yra šios:

A –7 kartai

B- 13 kartų

AA – 1 kartas

AB – 6 kartai

BA –5 kartai

BB – 7 kartai

AAA – 0 kartų

AAB-1 kartas

ABA –1 kartas

ABB-5 kartai

BAA – 1 kartas

BAB – 4 kartai

BBA – 4 kartai

BBB – 2 kartas

……..

ABAABBABBABBABBBBABB – 1 kartas

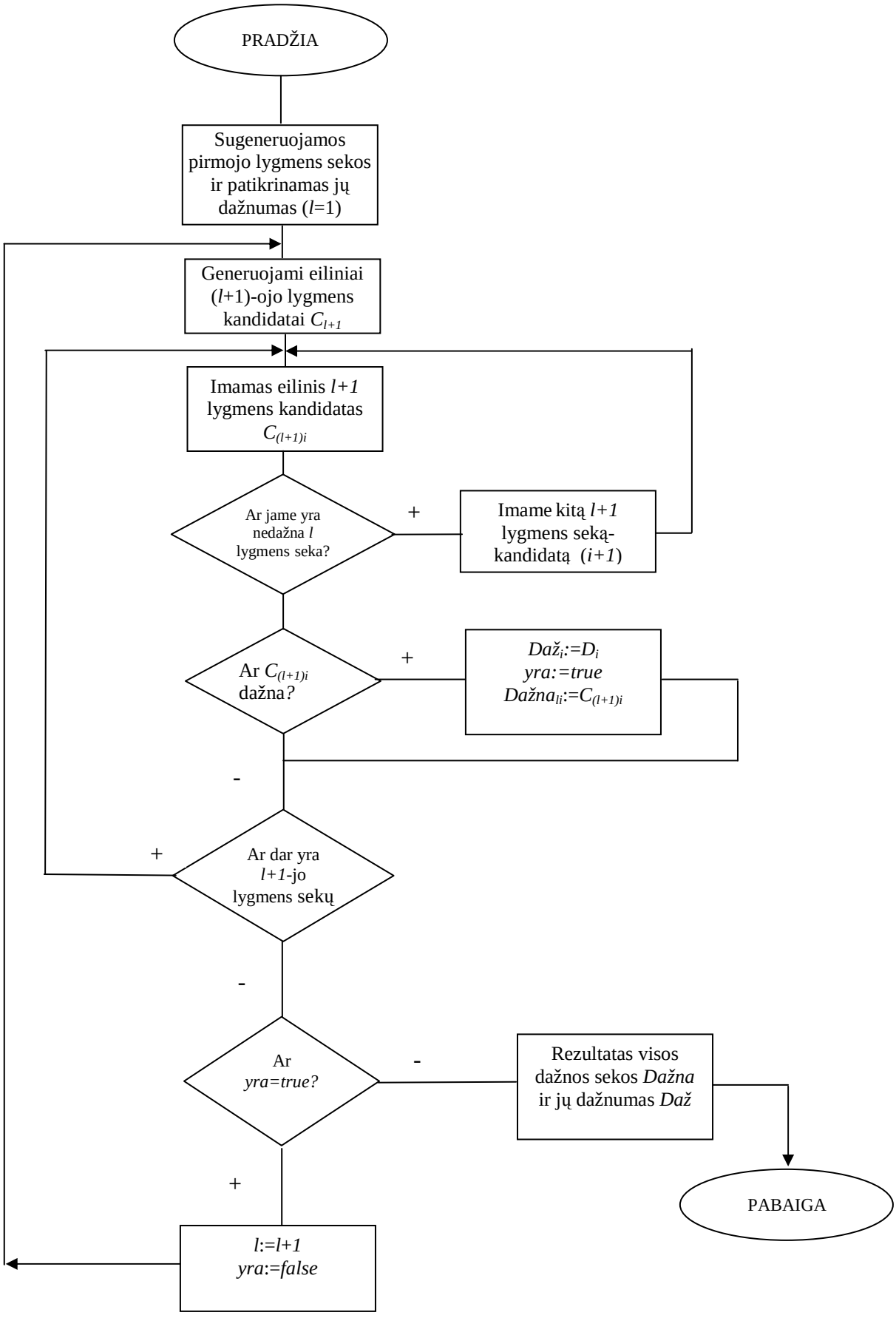
Mūsų atveju pirmojo lygmens sekos yra *A* ir *B*, antrojo – *AA*, *AB*, *BA*, *BB*, trečiojo *AAA*, *AAB*, *ABA*, *ABB*, *BAA*, *BAB*, *BBA*, *BBB* ir t.t. Bendru atveju, esant dviems elementams *l* lygmenyje, bus 2^l sekų. Vadinasi, *l* lygmenyje bus tikrinamos $\sum_{i=1}^{l} 2^i$ kombinacijos. Bendru atveju, kai elementų bus *n*, tikrinamos $\sum_{i=1}^{l} n^i$ kombinacijos. Jeigu nagrinėjamas tekstas yra pakankamai didelis (o taip ir bus), tai kiekvienos sekos peržiūrėjimas užims labai daug laiko. Taigi, tiesioginės paieškos algoritmas yra neefektyvus.

4.4. GSP (*Generate Sequence Pattern*) algoritmas

Šio algoritmo pagrindinis uždavinys nustatyti, kokios sekos yra TIKRAI nedažnos ir jų toliau netikrinti.

Pirmiausia tokie pastebėjimai. Jeigu seka *K* yra dažna tai visi jos galimi posekiai R_i irgi yra dažni ($K = \bigcup R_i$). Pavyzdžiui, jei seka *AABA* yra dažna, tai visi jos posekiai *A*, *B*, *AA*, *AB*, *BA*, *AAB*, *ABA* yra dažni. Remiantis šiuo teiginiu, galima padaryti išvada, kad jei sekoje yra nors vienas nedažnas posekis, tai seka yra nedažna. Akivaizdu, kad jei seka nedažna, visos kitos iš jos gaunamos naujos pridėtinės, aukštesnio lygmens, sekos irgi bus nedažnos. Pavyzdžiui, jei seka *AA* nedažna, tai visos sekos *AAB*, *AAA*, *AAAB* ir t.t. bus nedažnos.

Tarkime, kad yra aibė $L=\{i_1, i_2, \ldots, i_m\}$, sudarytą iš *m* elementų. Nagrinėjama duomenų bazė yra *Q*, kuri sudaryta iš įvairių aibės *L* elementų kombinacijų. Reikia rasti dažnas sekas. Pirmiausia tikrinamos pirmojo lygmens sekos. Tokių sekų yra *m:* ($i_1, i_2, \ldots, i_m$). Nustačius jų dažnius, pereinama tikrinti 2-jo lygmens sekas. Jų jau bus m^2: ($i_1i_1, i_1i_2, \ldots i_1i_m, i_2i_1, \ldots i_2i_m, \ldots, i_mi_1, \ldots i_mi_m$). Tačiau dabar tikrinamos nevisos sekos. Ką tikrinti, sprendžiama pagal prieš tai buvusį lygmenį. Jeigu į antrojo lygmens seką įeina nedažna pirmojo lygmens seka, tai pagal mūsų padarytą išvadą antrojo lygmens seka irgi yra nedažna ir ją galima atmesti toliau netikrinant. Tarkime, kad pirmajame lygmenyje nedažnų sekų buvo *p*, tai 2-ojo lygmens nedažnų sekų bus žymiai daugiau.



30 paveikslas. GSP algoritmo blokinė schema

Taip pereinama prie kito lygmens, kuris buvo sukurtas iš prieš tai buvusio antrojo lygmens. Trečiajame lygmenyje bus jau m^3 kombinacijų, bet vėl gi tikrinamos ne visos sekos, o tik tos kuriose **nėra nedažnų antrojo lygmens posekių** [89,92]. Vadinasi, bus tikrinamos ne visos $\sum_{i=1}^{m} l^i$ kombinacijas, bet $\sum_{i=1}^{n} (l^i - p_{i-1})$ kombinacijos, kur $p_{i\text{-}1}$ yra prieš tai buvusio lygmens nedažnų sekų posekiai. GSP algoritmo blokinė schema pavaizduota 30 paveiksle.

Panagrinėkime pavyzdį. Tarkime duomenų bazėje yra tie patys simboliai **ABAABBABBABBABBBBABB.**

Sakysime, kas seka yra dažna tada ir tik tada, kai ji pasirodo nemažiau kaip 6 kartus. T.y. minimalus dažnumas yra 6.

3 lentelė. Pirmasis lygmuo

Lygmuo	Seka	Dažnumas	Ar dažna?
1	A	7	+
1	B	13	+

Kaip matosi, visos sekos priklausančios pirmajam lygmeniui, yra dažnos. Pagal jas formuojamas antrasis lygmuo.

4 lentelė. Antrasis lygmuo

Lygmuo	Seka	Ar tikriname?	Dažnumas	Ar dažna?
2	AA	+	1	-
2	AB	+	6	+
2	BA	+	5	-
2	BB	+	7	+

Čia tikrinamos visos sekos, nes jos neturi nedažnų prieš tai buvusio lygmens posekių. Formuojamas trečiasis lygmuo.

5 lentelė. Trečiasis lygmuo

Lygmuo	Seka	Ar tikriname?	Dažnumas	Ar dažna?
3	AAA	-	-	-
3	AAB	-	-	-
3	ABA	-	-	-
3	ABB	+	5	-
3	BAA	-	-	-
3	BAB	-	2	-
3	BBA	-	-	-
3	BBB	+	2	-

Iš naujai gautų trečiojo lygmens sekų netikrinamos penkios: *AAA*, *AAB*, *ABA*, *BAA*, *BBA*, nes į jas įeina nedažni prieš tai buvusio lygmens posekiai. Sumažėjęs tikrinimų skaičius didina algoritmo efektyvumą laiko atžvilgiu. Kadangi trečiajame lygmenyje dažnų sekų nebėra, ketvirtasis lygmuo neformuojamas ir laikoma, kad algoritmo darbas baigtas.

4.5. GSP algoritmas. Atminties problema

GSP algoritmas gali būti realizuotas dvejopai. Pirmas standartinis realizacijos variantas neatsižvelgia į atminties taupymą. Naujai pasiūlytas realizacijos variantas atsižvelgia į atminties taupymą ir stengiasi jį minimizuoti. Formuojant eilinio lygmens sekas, pirmiausiai generuojamos visos įmanomos kombinacijos ir tik po to žiūrima, ar reikia tikrinti jų dažnumą. T.y. ar juose nėra prieš tai buvusio lygmens nedažnų sekų. Algoritmą galima patobulinti, visai negeneruojant naujų sekų iš ankstesniojo lygmens nedažnų sekų, nes jos vis tiek bus nedažnos.

Šio algoritmo blokinė schema pavaizduota 31 paveiksle.

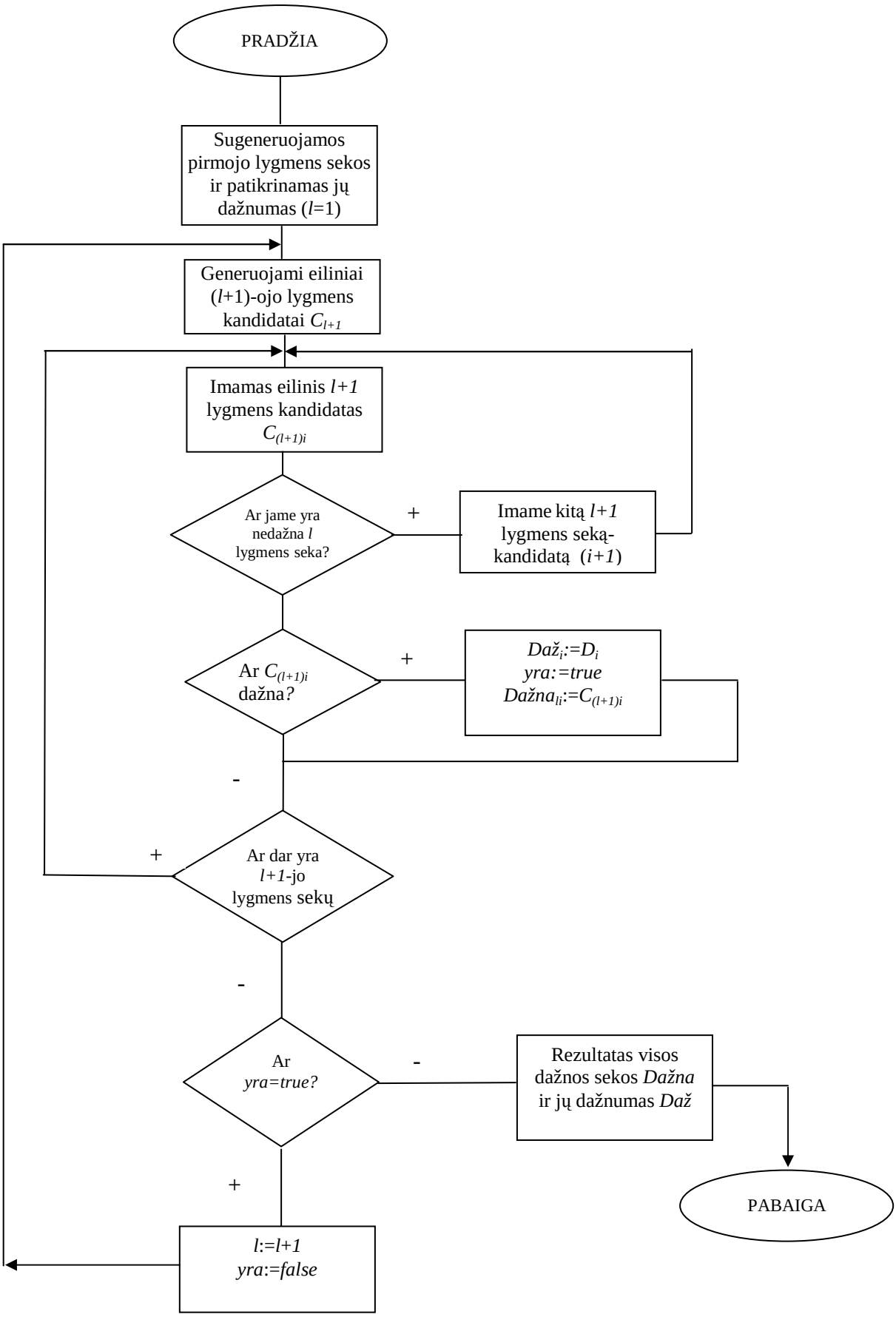
Grįžkime prie mūsų ankstesnio pavyzdžio, kurį nagrinėjome GSP algoritmo pirmoje versijoje. Pirmieji du lygmenys atrodo vienodai, tačiau trečiajame lygmenyje yra skirtumų. Dabar formuojant šį lygmenį, bus atsižvelgta į prieš tai einančio lygmens sekų dažnumą.

Trečiasis lygmuo pateiktas 7 lentelėje.

7 lentelė. Trečiasis lygmuo (GSP2)

Lygmuo	Seka	Ar tikriname?	Dažnumas	Ar dažna?
3	ABA	-		-
3	ABB	+	5	-
3	BBA	-		-
3	BBB	+	2	-

Realizuojant anksčiau išvardintus GSP algoritmus, susiduriama su operatyviosios atminties problema. Šią problemą galima iš dalies išspręsti. Detaliau panagrinėjus šiuos algoritmus, galima pastebėti, kad nebūtina saugoti visų sugeneruotų sekų atmintyje. Pakanka saugoti tik dažnas sekas ir dviejų lygmenų (einamojo ir prieš tai buvusio) sekas.



31 paveikslas. GSP algoritmo blokinė schema: realizacija, atsižvelgianti į atminties taupymą

4.6. Rekursinis algoritmas

Nereikalingų sekų saugojimas operatyvioje atmintyje mažina algoritmo efektyvumą. Pateiksime rekursinį algoritmą, kuris generuos naujas sekas ne „platyn“, kaip buvo parodyta GSP algoritme, bet „gilyn“. Pirmajame žingsnyje imama tuščia seka ir iš jos generuojamos sekos-kandidatės $\{i_1, i_2, \dots, i_n\}$. Iš šių sekų generuojamos antrojo lygmens sekos $\{i_1i_1, i_1\ i_2, \dots, i_1\ i_n, i_2i_1, i_2\ i_2, \dots, i_2\ i_n, \dots, i_ni_1, i_n\ i_2, \dots, i_n\ i_n\}$. Skirtingai nei GSP, sekų generavimas čia vyksta „į gylį“. Kiekvienai sugeneruotai sekai nustatomas dažnumas. Jei seka yra nedažna, tai daugiau sekų-kandidačių iš jos negeneruojama. Žemiau pateikiamas pseudokodu parašytas rekursinis algoritmas:

```
type sekos=string;
procedure naujos_sekos_generavimas (var seka: sekos);
var   simbolis: char;
      dažnumas: longint;
      seka_kandidatė: string;
begin
   for simbolis:=i1 to in do
      begin
         seka_kandidatė:=seka+simbolis;
         ieskomas_dažnumas(seka_kandidatė,dažnumas);
         if daznumas>=Min_daznumas then
            begin
               index:=index+1;
               dažnos_sekos[index].sekos_daznumas:=dazžumas;
               dažnos_sekos[index].seka:= seka_kandidatė;
               naujos_sekos_generavimas (seka_kandidatė);
            end;
      end;
end;
```

Vėl panagrinėkime ankstesnį pavyzdį, kai duomenų bazė yra tekstas ABAABBABBABBABBBBABB ir minimalus dažnumas lygus 6. Algoritmo veikimas parodytas 6 lentelėje.

6 lentelė. Rekursinio algoritmo veikimas

Žingsnis	Seka	σ(Seka)	Dažna ar ne?
1	A	7	+
2	AA	1	-
3	AB	6	+
4	ABA	1	-
5	ABB	5	-
6	B	13	+
7	BA	5	-
8	BB	7	+
9	BBA	4	-
10	BBB	2	-

Tokiu būdu pagrindinėje atmintyje nebereikia saugoti visų einamojo lygmens sekų. Šiuo atveju rekursija žymiai efektyviau ir greičiau atlieka apdorojimo operacijas.

Panagrinėkime kitą pavyzdį su GSP algoritmu. Šis pavyzdys mums bus reikalingas aprašant tolimesnius tyrimus.

Tarkime pagrindinė seka *S* yra tokia:

S = ABCCCBBCABCABCABCBABCCCABCAABABCABC, o minimalus dažnumas (*min_sup*) yra lygus 4.

Visos pirmojo lygmens sekos yra dažnos (žr. 7 lentelę). Remiantis šiomis sekomis, generuojamas antras lygmuo (žr. 8 lentelę). Po antrojo lygmens patikrinimo generuojamas trečias lygmuo (žr. 9 lentelę).

7 lentelė. Pirmasis lygmuo

Lygmuo	Seka	Dažnumas	Ar dažna?
1	*A*	10	+
1	*B*	13	+
1	*C*	13	+

8 lentelė. Antras lygmuo

Lygmuo	Seka	Ar tikriname?	Dažnumas	Ar dažna?
2	*AA*	+	1	-
2	*AB*	+	9	+
2	*AC*	+	0	-
2	*BA*	+	2	-
2	*BB*	+	1	-
2	*BC*	+	9	+
2	*CA*	+	6	+
2	*CB*	+	2	-
2	*CC*	+	4	+

9 lentelė. Trečias lygmuo

Lygmuo	Seka	Ar tikriname?	Dažnumas	Ar dažna?
3	*ABA*	-	-	-
3	*ABC*	+	8	+
3	*ABB*	-	-	-
3	*BCA*	+	5	+
3	*BCB*	-	-	-
3	*BCC*	+	2	-
3	*CAB*	+	5	+
3	*CAA*	-	-	-
3	*CAC*	-	-	-
3	*CCA*	+	1	-
3	*CCB*	-	-	-
3	*CCC*	+	2	-

Nebus tikrinamos šešios naujos trečiojo lygmens sekos: *ABA*, *ABB*, *BCB*, *CAA*, *CAC*, *CCB* (žr. 9 lentelę), kadangi jose yra nedažni posekiai iš prieš tai buvusio (antrojo) lygmens. Ketvirtojo lygmens visos naujos sekos turės nedažnus posekius, vadinasi bus irgi nedažnos. Todėl laikysime, kad algoritmas baigė darbą. Surastos dažnos sekos yra šios: *A*, *B*, *C*, *AB*, *BC*, *CA*, *CC*, *ABC*, *BCA*, *CAB.*

4.7. Tikimybinis dažnų sekų nustatymo algoritmas ProMFS (*probabilistic algorithm for mining frequent sequences*)

Bet kuriuo atveju laiko sąnaudos, kurias naudoja aukščiau išvardinti algoritmai, yra pakankamai didelės, kadangi reikia daug kartų peržiūrėti visą duomenų bazę, esančią diske. Kyla klausimas, ar negalime būtų sumažinti kreipinių į diską skaičių, o dažnas sekas nustatyti

pagal tam tikras charakteristikas. Žinoma, tokie metodai gali turėti paklaidas ir netikslumas. Todėl jie tik apytiksliai nustatys dažnas sekas. Tokiu būdu būsime priversti aukoti tikslumą dėl greičio. Toliau pristatysime naująjį tikimybinį apytikslį algoritmą ProMFS.

Naujasis tikimybinis dažnų sekų nustatymo algoritmas remiasi šiomis statistinėmis pagrindinės sekos charakteristikomis:

- elemento pasirodymo sekoje tikimybė;
- tikimybe, kad vienas elementas eis po kito;
- atstumo tarp dviejų elementų pagrindinėje sekoje vidurkio.

Pagrindinė algoritmo idėja yra tokia:

1) tikimybinės charakteristikos apibūdina elementų pozicijas pagrindinėje sekoje;
2) remiantis šiomis charakteristikomis, generuojama nauja žymiai trumpesnė modelinė seka $\tilde{C}$;
3) nauja seka analizuojama GSP algoritmu (arba kokiu nors kitu tiksliu algoritmu);
4) GSP algoritmu gauti dažni posekiai modelinėje sekoje bus dažni posekiai ir pagrindinėje sekoje.

Pažymėkime:

1) $P(i_j) = \frac{V(i_j)}{VS}$ yra elemento i_j pasirodymo tikimybė pagrindinėje sekoje, čia $i_j \in L,\ j = 1,...,m$; $L=\{i_1, i_2, \ldots, i_m\}$ yra aibė, sudaryta iš m skirtingų elementų; $V(i_j)$ yra elemento i_j pasirodymo tikimybė pagrindinėje sekoje S; VS yra sekos ilgis. Galima pastebėti, kad $\sum_{j=1}^{m} P(i_j) = 1$.

2) $P(i_j \mid i_v)$ yra sąlyginė tikimybė, kad elementas i_v pasirodys po elemento i_j, kur $i_j, i_v \in L,\ j, v = 1,...,m$. Pastebėkime, kad $\sum_{v=1}^{m} P(i_j \mid i_v) = 1$ visiems $j = 1,...,m$.

3) $D(i_j \mid i_v)$ yra atstumas tarp elemento i_j ir i_v, kur $i_j, i_v \in L,\ j, v = 1,...,m$. Kitaip tariant, $D(i_j \mid i_v)$ yra skaičius elementų tarp i_j ir pirmojo surasto i_v, ieškant nuo i_j iki pagrindinės sekos pabaigos. Atstumas tarp dviejų kaimyninių elementų sekoje yra lygus vienam.

4) $\widehat{A}$ yra atstumų vidurkių matrica. Jos elementai yra šie: $a_{jv} = Average\,(D(i_j \mid i_v),\ \ i_j, i_v \in L),\ \ j, v = 1,..., m.$

Visos šios charakteristikos gaunamos vieną kartą peržiūrėjus pagrindinę seką. Remiantis šiomis charakteristikomis, sudaroma žymiai trumpesnė seka (vadinama modeline seka) $\widetilde{C}$, kurios ilgis bus lygus *l*. Pažymėkime jos elementus c_r, $r = 1,...,l$. Modelinė seka $\widetilde{C}$ turės visus elementus iš *L*: $i_j \in L$, $j = 1,...,m$. Kiekvienam jos elementui c_r apibrėžkime skaitinę charakteristiką $Q(i_j, c_r)$, $r = 1,...,l$, $j = 1,...,m$. Pradžioje $Q(i_j, c_r)$ yra matrica su nulinėmis reikšmėmis. Vėliau jos reikšmės bus nustatytos po statistinės pagrindinės sekos analizės. Papildome algoritmą dar viena funkcija $\rho(c_r, a_{rj})$. Ši funkcija padidina charakteristikų $Q(i_j, c_r)$ reikšmes vienetu. Pirmasis elementas c_1 modelinėje sekoje $\widetilde{C}$ yra iš *L* ir nustatomas pagal maksimalią reikšmę $\max(P(i_j))$, $i_j \in L$. Pagal c_1 yra aktyvuojama funkcija $\rho(c_1, a_{1j}) \Rightarrow Q(i_j, 1 + a_{1j}) = Q(i_j, 1 + a_{1j}) + 1$, $j = 1,..., m$. Likę elementai c_r, $r = 2,...,l$, yra parenkami tokiu būdu. Tarkime, kad norime nustatyti *r*-ąjį modelinės sekos $\widetilde{C}$ elementą c_r. Sprendimas, kuris simbolis iš *L* bus parinktas elementu c_r, nustatomas po $\max(Q(i_j, c_r))$, $i_j \in L$ paskaičiavimo. Jeigu tam tikriems *p* ir *t* gaunama, kad $Q(i_p, c_r) = Q(i_t, c_r)$, tada elementas c_r bus parinktas pagal maksimalią sąlyginių tikimybių reikšmę, t.y. $\max(P(c_{(r-1)} \mid i_p), P(c_{(r-1)} \mid i_t))$ šitaip $c_r = i_p$, jeigu $P(c_{(r-1)} \mid i_p) > P(c_{(r-1)} \mid i_t)$, ir $c_r = i_t$ jeigu $P(c_{(r-1)} \mid i_p) < P(c_{(r-1)} \mid i_t)$. Jeigu jos yra lygios, t.y. $P(c_{(r-1)} \mid i_p) = P(c_{(r-1)} \mid i_t)$, tada c_r bus parinktas nustatant $\max(P(i_p), P(\mid i_t))$. Po c_r reikšmės pasirinkimo aktyvuojama funkcija $\rho(c_r, a_{rj}) \Rightarrow Q(i_j, r + a_{rj}) = Q(i_j, r + a_{rj}) + 1$. Visi šie veiksmai bus atliekami visiems $r = 2,...,l$. Tokiu būdu gaunama modelinė seka $\widetilde{C}$, kuri yra žymiai mažesnė už mūsų nagrinėjamą pagrindinę seką. Po to modelinė seka analizuojama GSP algoritmu. Taigi laiko sąnaudos ženkliai sumažinamos.

Panagrinėkime prieš tai buvusiame pavyzdyje seką (1) *L*={*A, B, C*}, t. y. *m*=3, $i_1 = A$, $i_2 = B$, $i_3 = C$. Sekos *VS*=35, t.y. seka sudaryta iš 35 elementų.

Po vieno sekos patikrinimo apskaičiuojamos tikimybinės charakteristikos:

$P(A) = \frac{10}{35} \approx 0{,}2857$, $P(B) = \frac{12}{35} \approx 0{,}3429$, $P(C) = \frac{13}{35} \approx 0{,}3714$, $P(A \mid A) = 0{,}1$,

$P(A|B)=0.9$, $P(A|C)=0$, $P(B|A)\approx 0{,}1667$, $P(B|B)=0{,}0833$,

$P(B|C)\approx 0{,}7500$, $P(C|A)\approx 0{,}4615$, $P(C|B)=0{,}1538$, $P(C|C)\approx 0{,}3077$.

Mūsų pavyzdžio atstumų vidurkio matrica $\hat{A}$ pateikta 10 lentelėje.

10 lentelė. Atstumų vidurkių matrica $\hat{A}$

	A	*B*	*C*
A	3,58	1,10	2,50
B	2,64	2,91	1,42
C	2,33	2,25	2,67

Pasirinkime modelinės sekos $\tilde{C}$ ilgį *l*=8. Iš pradžių seka $\tilde{C}$ yra tuščia ir $Q(i_j, c_r)=0$, $r=1,...,l$, $j=1,...,m$:

r	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0
Modelinė seka $\tilde{C}$	-	-	-	-	-	-	-	-

Pirmasis elementas $\tilde{C}$ yra gaunamas, apskaičiuojant maksimalią $P(i_j)$ reikšmę. Mūsų pavyzdyje tai bus raidė *C*, taigi $c_1=C$. Perskaičiuojame $Q(i_j, c_1)$, $j=1,2,3$, pasinaudojus atstumų vidurkiais. Situacija bus tokia:

r	1	2	3	4	5	6	7	8
A	0	0	**1**	0	0	0	0	0
B	0	0	**1**	0	0	0	0	0
C	0	0	0	**1**	0	0	0	0
Modelinė seka $\tilde{C}$	***C***							

c_2 pasirinkimas. Visos trys reikšmės $Q(i_j, c_1)$, $j=1,2,3$, yra vienodos ir lygios 0. Tada c_2 bus nustatyta pagal maksimalią elementų pasirodymo tikimybių reikšmę. Max(*P*(*C*|*A*), *P*(*C*|*B*), *P*(*C*|*C*)=*P*(*C*|*A*)=0,4615.

Vadinasi $c_2 = A$. Perskaičiuojame $Q(i_j, c_2)$, $j = 1,2,3$, pasinaudodami atstumų vidurkiais. Situacija bus tokia:

r	1	2	3	4	5	6	7	8
A	0	0	1	0	0	**1**	0	0
B	0	0	**2**	0	0	0	0	0
C	0	0	0	1	**1**	0	0	0
Modelinė seka $\tilde{C}$	*C*	***A***						

Kiti trys žingsniai pavaizduoti žemiau:

r	1	2	3	4	5	6	7	8
A	0	0	1	0	0	**2**	0	0
B	0	0	2	0	0	**1**	0	0
C	0	0	0	1	**2**	0	0	0
Modelinė seka $\tilde{C}$	*C*	*A*	***B***					

r	1	2	2	3	4	4	6	7
A	0	0	1	0	0	**3**	0	0
B	0	0	2	0	0	**2**	0	0
C	0	0	0	1	2	0	**1**	0
Modelinė seka $\tilde{C}$	*C*	*A*	*B*	***C***				

r	1	2	3	4	5	6	**7**	8
A	0	0	1	0	0	3	**1**	0
B	0	0	2	0	0	2	**1**	0
C	0	0	0	1	2	0	1	**1**
Modelinė seka $\tilde{C}$	*C*	*A*	*B*	*C*	***C***			

Galutinai gaunama tokia modelinė seka $\tilde{C}$ = *CABCCABC*. GSP algoritmu nustatome, kad dažniausia seka modelinėje sekoje yra *ABC,* kai minimalus dažnumas lygus 2. Kita dažna seka yra seka *CAB.* Pagrindinėje sekoje šios sekos irgi yra dažnos (sekos *ABC* dažnumas lygus 8, o sekos *CAB* – 5). Tačiau seka *BCA*, kurios dažnumas lygus 5, yra nenustatyta ir, vadinasi, prarasta.

Šį algoritmą galime modifikuoti, pakeičiant kai kurias tikimybines charakteristikas kitomis. Pakeisime 3-ią charakteristiką (atstumo tarp dviejų elementų pagrindinėje sekoje vidurkį) į dažniausią atstumą tarp dviejų elementų pagrindinėje sekoje. Vidurkių nustatymo charakteristika nevisada bus tiksli, o dažniausias atstumas tarp dviejų elementų padarytų šį naujai pasiūlytą algoritmą tikslesniu. Pavadinsime jį ProMFS su dažniausių atstumų matrica.

$\hat{F}$ yra dažniausių atstumų matrica. Jos elementai yra tokie: $a_{jv} = Frequent\ (D(i_j \mid i_v),\ \ i_j, i_v \in L),\ \ j, v = 1, ..., m.$

Mūsų pavyzdžio atveju dažniausi atstumai tarp elementų bus šie:
$D(A|A)$ = (8, 3, 3, 4, 5, 3, 1, 2, 3), $D(A|B)$ = (1, 1, 1, 1, 1, 1, 2, 1, 1, 1),

$D(A|C)$ = (2, 2, 2, 2, 2, 2, 5, 4, 2, 2), $D(B|A)$ = (7, 3, 2, 2, 2, 3, 1, 4, 2, 1, 2),

$D(B|B)$ = (4, 1, 3, 3, 3, 3, 2, 2, 5, 4, 2, 3), $D(B|C)$ = (1, 2, 1, 1, 1, 1, 3, 1, 3, 1, 1),

$D(C|A)$ = (6, 5, 4, 1, 1, 1, 2, 3, 2, 1, 1, 1), $D(C|B)$ = (3, 2, 1, 2, 2, 2, 1, 4, 3, 2, 3, 2),

$D(C|C)$ = (1, 1, 3, 3, 3, 3, 4, 1, 1, 3, 6, 3).

Gauta dažniausių atstumų matrica $\hat{F}$ pateikta 11 lentelėje.

Lentelė 11. Dažniausių atstumų matrica $\hat{F}$.

	A	*B*	*C*
A	3	1	2
B	2	3	1
C	1	2	3

Pasinaudojant tomis pačiomis tikimybinėmis charakteristikomis bei pakeičiant vidurkių matricą $\hat{A}$ dažniausių atstumų matrica $\hat{F}$, apskaičiuokime modelinę seką kitu būdu. Laikysime, kad modelinės sekos $\tilde{C}$ ilgis *l*=8. Pradžioje seka $\tilde{C}$ yra tuščia ir $Q(i_j, c_r) = 0$, $r = 1,...,l$, $j = 1,...,m$:

r	1	2	3	4	5	6	7	8
A	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0
Modelinė seka $\tilde{C}$	-	-	-	-	-	-	-	-

Pirmasis sekos $\tilde{C}$ elementas apskaičiuojamas pagal didžiausią tikimybinę reikšmę $P(i_j)$. Mūsų pavyzdyje tai yra *C*, taigi $c_1 = C$. Perskaičiuojame $Q(i_j, c_1)$, $j = 1,2,3$, pasinaudojus dažniausių atstumų matrica. Situacija po šio perskaičiavimo atrodys taip:

r	1	2	3	4	5	6	7	8
A	0	**1**	0	0	0	0	0	0
B	0	0	**1**	0	0	0	0	0
C	0	0	0	**1**	0	0	0	0
Modelinė seka $\tilde{C}$	***C***							

Dabar pasirenkame c_2. Elementas *A* turi didžiausią reikšmę lygią 1. Perskaičiuojame $Q(i_j, c_2)$, $j = 1,2,3$, pasinaudojus dažniausių atstumų matrica. Situacija po šio perskaičiavimo atrodys taip:

r	1	2	3	4	5	6	7	8
A	0	1	0	0	**1**	0	0	0
B	0	0	**2**	0	0	0	0	0
C	0	0	0	**2**	0	0	0	0
Modelinė seka $\tilde{C}$	*C*	***A***						

Kiti žingsniai, formuojant modelinę seką atrodys taip:

r	1	2	3	4	5	6	7	8
A	0	1	0	0	**2**	0	0	0
B	0	0	2	0	0	**1**	0	0
C	0	0	0	**3**	0	0	0	0
Modelinė seka $\tilde{C}$	*C*	*A*	***B***					

r	1	2	2	3	4	4	6	7
A	0	1	0	0	**3**	0	0	0
B	0	0	2	0	0	**2**	0	0
C	0	0	0	3	0	0	**1**	0
Modelinė seka $\tilde{C}$	*C*	*A*	*B*	***C***				

r	1	2	3	4	5	6	**7**	8
A	0	1	1	0	3	0	0	**1**
B	0	0	2	0	0	**3**	0	0
C	0	0	0	3	0	0	**2**	0
Modelinė seka $\tilde{C}$	*C*	*A*	*B*	*C*	***A***			

r	1	2	3	4	5	6	**7**	8
A	0	1	1	0	3	0	0	**2**
B	0	0	2	0	0	3	0	0
C	0	0	0	3	0	0	**3**	0
Modelinė seka $\tilde{C}$	*C*	*A*	*B*	*C*	*A*	***B***		

Galutinai gaunama modelinė seka $\tilde{C}$ = *CABCABCA*. Tiksliu GSP algoritmu nustatoma, kad didžiausios dažnos sekos modelinėje sekoje yra *ABC*, *BCA* ir *CAB*, kai minimalus dažnumas yra nustatytas 2. Pagrindinėje sekoje *S* sekos *ABC* dažnumas lygus 8, *BCA* – 5, o sekos *CAB* dažnumas yra 5. Toks pat rezultatas gaunamas ir su tiksliu GSP algoritmu. Skirtingai nei prieš tai pateiktame tikimybinio algoritmo variante, čia praradimų neužfiksuota.

4.8. Išvados

Šiame skyriuje yra realizuotas ir patobulintas standartinis GSP algoritmas, skirtas dažnoms sekoms rasti dideliuose duomenų masyvuose. Taip pat buvo realizuota tikslaus dažnų sekų nustatymo algoritmo GSP modifikacija, kurios metu iš nedažnų sekų nebuvo generuojamos kito lygmens sekos-kandidatės. GSP algoritmas buvo taip pat realizuotas, panaudojant rekursinį dažnų sekų nustatymo būdą „gilyn“.

Pateiktas naujas tikimybinis algoritmas ProMFS su dviem jo modifikacijomis: naudojant vidurkių matricą bei dažniausių atstumų matricą. Paprastu pavyzdžiu išnagrinėti šių dviejų variantų veikimo skirtumai.

5. Tyrimų rezultatai

5.1. Įvadas

Šiame skyriuje aprašyti algoritmų efektyvumo (laiko sąnaudų bei atminties panaudojimo) palyginimai su dirbtiniais bei realiais duomenimis. Tai pat pateikta naujai pasiūlyto tikimybinio ProMFS algoritmo tikslumo bei efektyvumo analizė, lyginant jį su tiksliuoju GSP algoritmu. Buvo atlikti šie praktiniai darbai:

a) standartinio bei patobulinto GSP algoritmų laiko bei atminties sąnaudų palyginimas;
b) skirtingų patobulinto GSP algoritmo realizacijų palyginimas, priklausomai nuo nagrinėjamame faile simbolių išdėstymo tvarkos;
c) tikimybinio ProMFS algoritmo tikslumo bei laiko sąnaudų palyginimas su tiksliuoju GSP algoritmu, priklausomai nuo failo dydžio, skirtingų simbolių kiekio. Palyginimai atliekami su abiejomis ProMFS algoritmo realizacijomis (su vidurkių ir dažniausių atstumų matricom);
d) ProMFS ir GSP algoritmų efektyvumas palygintas su realiais genetiniais duomenimis.

Visi eksperimentai buvo atlikti su 2400 Mhz dažnio ir 512 MB operatyviosios atminties (RAM) PENTIUM III kompiuteriu. Algoritmams realizuoti buvo panaudota *Free Pascal* programavimo kalba.

5.2. GSP algoritmų realizacijų palyginimas

Tarkime turime failą, kurio turinį sudaro 90000 simbolių. Kiekvienas simbolis tai arba A arba B raidė. Mūsų tikslas apskaičiuoti dažnas sekas. Algoritmo GSP realizacijos, neatsižvelgiant į atminties taupymą, ir realizacijos, atsižvelgiant į atminties taupymą, rezultatai pavaizduoti 12 lentelėje.

Šioje lentelėje yra tokie laukai:

Minimalus dažnumas – yra minimalus ieškomų sekų dažnumas;

Laikas I – laiko sąnaudos, reikalingos GSP algoritmo realizacijai be atminties taupymo;

Laikas II – laiko sąnaudos, reikalingos GSP algoritmo realizacijai su atminties taupymu;

MAX – maksimalus dažnos sekos ilgis;

Atmintis I – atminties kiekis, reikalingas GSP algoritmo realizacijai be atminties taupymo;

Atmintis II – atminties kiekis, reikalingas GSP algoritmo realizacijai su atminties taupymu;

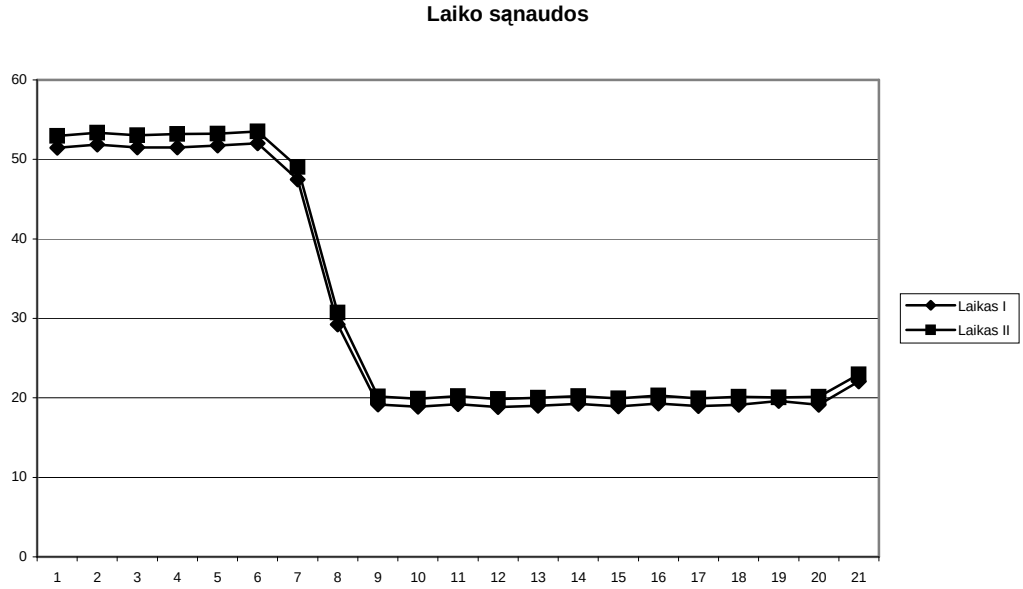
Rasta dažnų sekų – dažnų sekų kiekis;

Kreipiniai į diską – kiek kartų buvo kreiptasi į diską.

12 lentelė. Algoritmo GSP realizacijų rezultatai

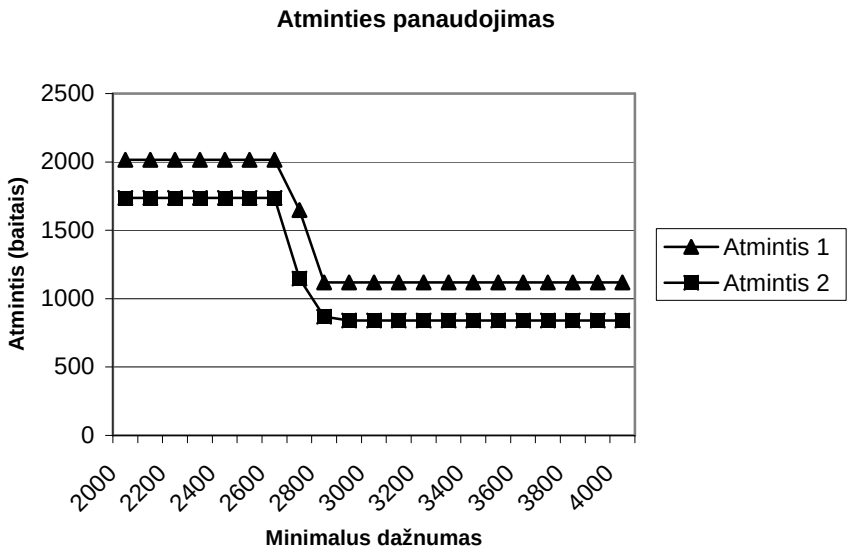
Minimalus dažnumas	*Laikas* I	*Laikas* II	*MAX*	*Atmintis* I (B)	*Atmintis* II (B)	*Rasta dažnų sekų*	*Kreipiniai į diską*
2000	51.47	52.97	6	6692	2400	114	178
2100	51.85	53.35	6	6692	2400	114	178
2200	51.52	53.02	6	6692	2400	114	178
2300	51.52	53.19	6	6692	2400	114	178
2400	51.74	53.24	6	6692	2400	114	178
2500	52.01	53.51	6	6692	2400	114	178
2600	47.47	49.04	6	6412	2400	109	168
2700	29.22	30.72	6	4676	2400	78	114
2800	19.17	20.17	5	3164	800	52	84
2900	18.9	19.9	5	3164	800	52	84
3000	19.22	20.22	5	3164	800	52	84
3100	18.84	19.84	5	3164	800	52	84
3200	19.01	20.01	5	3164	800	52	84
3300	19.23	20.23	5	3164	800	52	84
3400	18.94	19.94	5	3164	800	52	84
3500	19.28	20.28	5	3164	800	52	84
3600	18.95	19.95	5	3164	800	52	84
3700	19.12	20.12	5	3164	800	52	84
3800	19.6	20.06	5	3164	800	52	84
3900	19.12	20.12	5	3164	800	52	84
4000	22.08	22.95	5	3164	800	52	84

Pateiktos lentelės abiejų GSP realizacijų laiko ir atminties sąnaudos grafiškai iliustruojamos 27 ir 28 paveiksluose. Iš 27 paveikslo matyti, kad laiko sąnaudos beveik vienodos.



27 paveikslas. GSP algoritmo realizacijų laiko sąnaudos

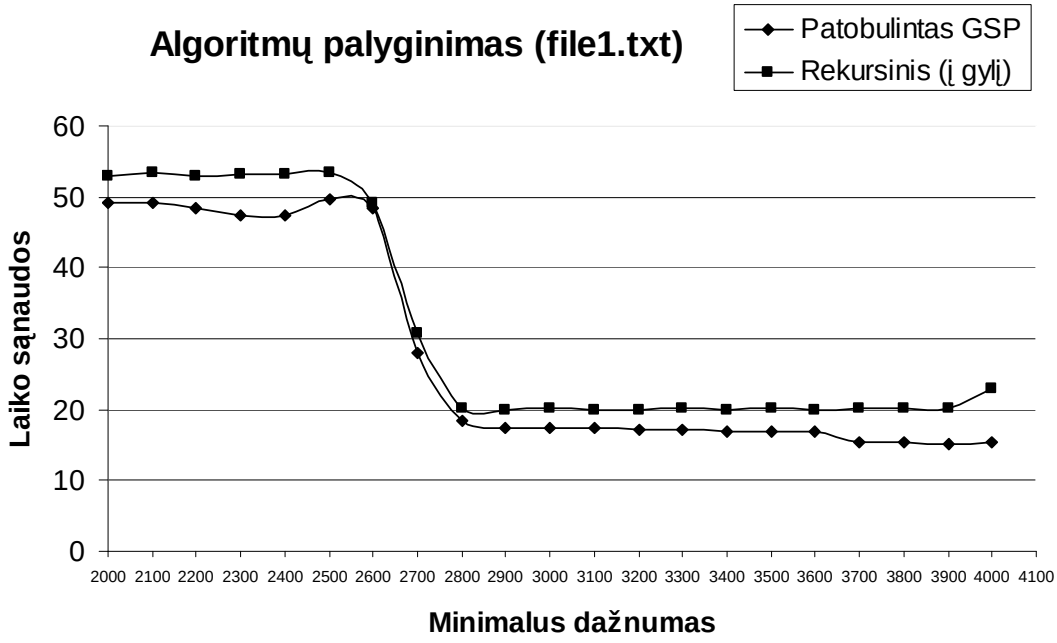
Iš 28 paveikslo matyti, kad skiriasi naudojamos operatyviosios atminties dydis ir GSP algoritmo realizacija taupant atmintį yra efektyvesnė. Galime padaryti išvadą, kad pirmuoju algoritmu mums reikės atminties nedaugiau kaip $k\sum_{i=1}^{n}2^i$, čia k yra atminties kiekis, reikalingas saugoti vieną simbolių eilutę. Tuo tarpu antrojo algoritmo atminties poreikis yra tik $k(2^{n-1}+2^2)$.



28 paveikslas. Algoritmų atminties panaudojimo palyginimas

5.3. Patobulinto GSP algoritmo bei rekursinio algoritmo veikimo rezultatai

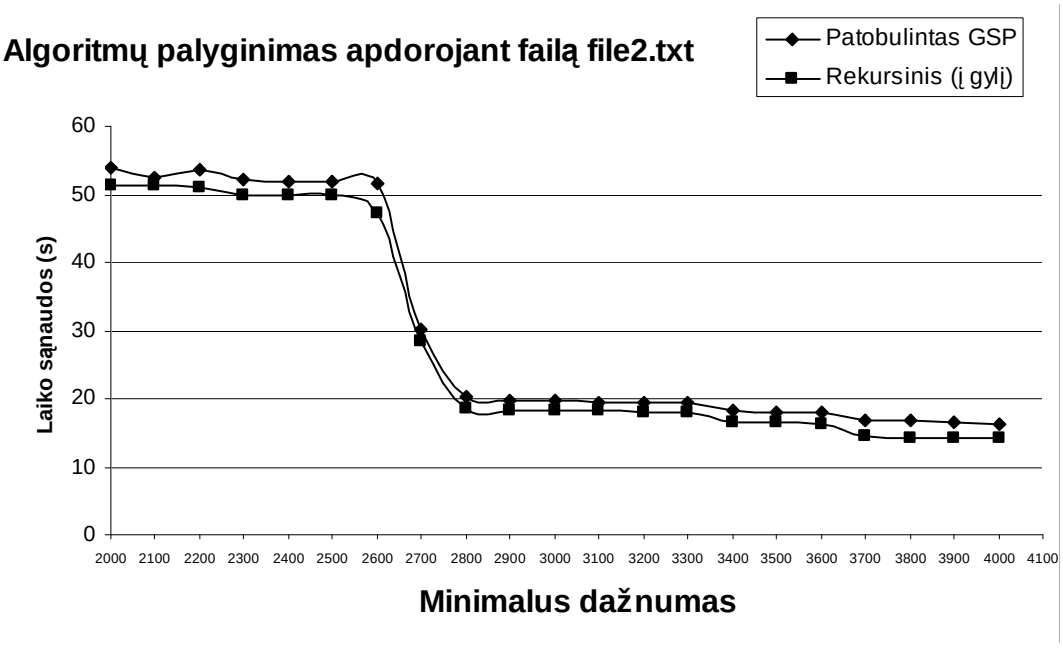
Nagrinėjant algoritmų efektyvumą, rezultatai labai priklauso nuo testuojamų duomenų. Skirtingai sugeneravus duomenys, galime gauti skirtingus rezultatus. Sugeneruokime du failus, sudarytus iš simbolių A, B ir C taip, kad viename faile (pavadinkime jį *file1.txt*) visos dažnos sekos neturėtų iš eiles einančių vienodų simbolių (pvz.: ABCACBACBA, BACACBABAC ir t.t.), o kitame faile (pavadinkime jį *file2.txt*) eitų daug vienodų iš eilės einančių simbolių (pvz., AAAAABBBCCC, CCCAAABBB, BBBCCCAA ir t.t.). Panagrinėkime patobulinto GSP algoritmo (su atminties taupymu) ir rekursinio algoritmo, kuris naudoja paieška gilyn, laiko sąnaudas.



29 **paveikslas.** Algoritmų palyginimas apdorojant failą *file1.txt*

29 paveiksle pavaizduotas laiko sąnaudų palyginimas patobulinto GSP algoritmo bei rekursinio algoritmo, kuris nagrinėja dažnas sekas į gylį. Failas *file1.txt* buvo sugeneruotas taip, kad visos dažnos sekos neturėtų iš eiles einančių vienodų simbolių. Matome, kad šiuo atveju greičiau (nors ir nedaug) veikia patobulintas GSP algoritmas.

30 paveiksle pavaizduotas laiko sąnaudų palyginimas patobulinto GSP algoritmo bei rekursinio algoritmo, kuris nagrinėja dažnas sekas į gylį. Failas *file2.txt* buvo sugeneruotas taip, kad visos dažnos sekos turėtų iš eilės einančių vienodų simbolių. Matome, kad šiuo atveju greičiau (nors irgi nedaug) veikia rekursinis algoritmas, nagrinėjantis sekas į gylį.



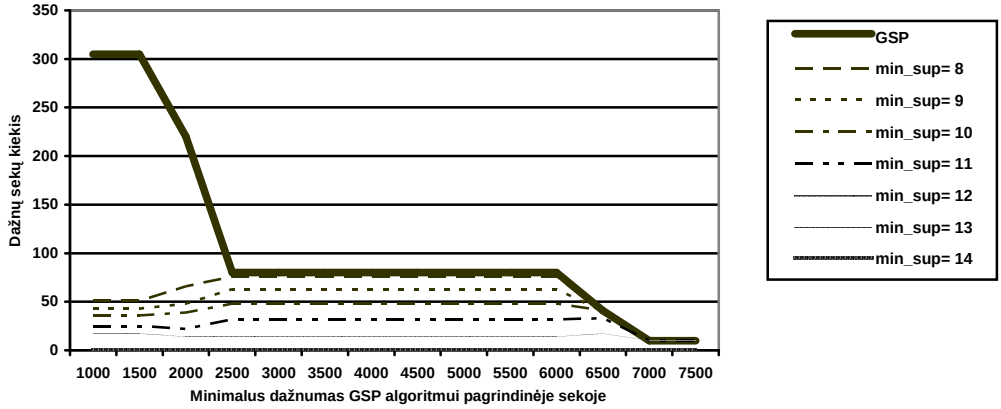
30 **paveikslas.** Algoritmų palyginimas apdorojant failą *file2.txt*

5.4.Tikimybinio algoritmo veikimo rezultatai

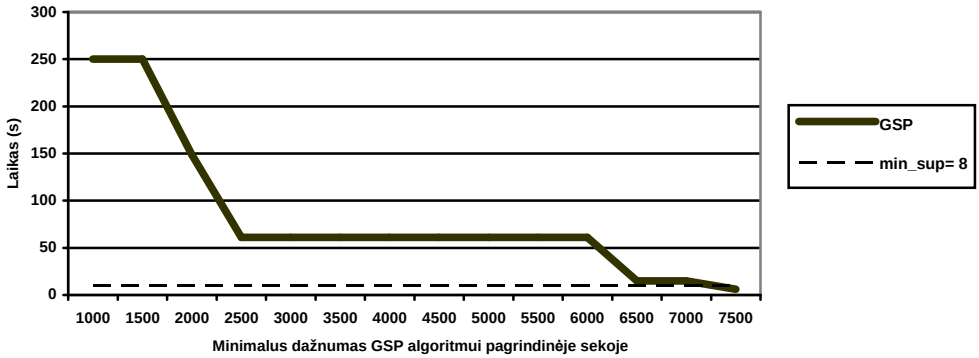
Čia tikimybinis dažnų sekų nustatymo algoritmas (ProMFS) su vidurkių matricų yra palyginamas su GSP algoritmu. Sugeneruotas tekstas turi 100000 raidžių (1000 eilučių, kuriose yra po 100 simbolių). $L=\{A, B, C\}$, t.y. m=3, $i_1 = A, i_2 = B, i_3 = C$. Į tekstą įdėta labai dažnai pasikartojanti seka *ABBC*. Ši seka pasikartoja 20 kartų kiekvienoje eilutėje. Kiti 20 simbolių parinkti atsitiktiniu būdu. Pirmiausia ištyrėme šią pagrindinę seką (100000 simbolių) GSP algoritmu. Rezultatai parodyti 31 ir 32 paveiksluose. Aptarsime rezultatus, kuriuos gavome su ProMFS algoritmu. ProMFS sugeneravo šią modelinę seką $\tilde{C}$, kurios ilgis l=40:

$\tilde{C}$ = *BBCABBCABBCABBCABBCABBCABBCABBCABBCABBCA*

Ši modelinė seka buvo išnagrinėta GSP algoritmu su šiais minimaliais dažnumais (*Ms*): 8, 9, 10, 11, 12, 13 ir 14. 31 paveiksle sulygintas GSP ir ProMFS algoritmų rastų dažnų sekų kiekis. 32 paveiksle sulyginamas GSP ir ProMFS algoritmų sugaištas laikas. 31 paveiksle matyti, kad esant sąlyginai nedideliam minimaliam dažnumui (mažesniam nei 1500), GSP algoritmas suranda žymiai daugiau dažnų posekių nei ProMFS. Tačiau esant didesniam minimaliam dažnumui (nuo 2500 iki 6000), rezultatai gaunami apytiksliai vienodi. Tačiau, jei palygintume algoritmų sugaištą laiką, tai pastebėtume, kad ProMFS algoritmas analizuoja seką žymiai greičiau. Vadinasi, galima padaryti išvadą, kad ProMFS algoritmas yra efektyvus (žymiai mažesnės laiko sąnaudos), esant sąlyginai dideliam minimaliam dažnumui.

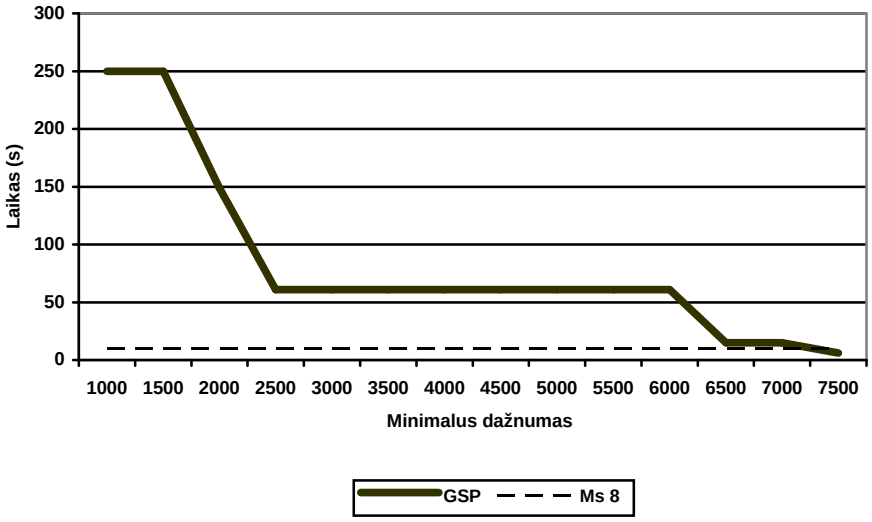


31 paveikslas. Rastų dažnų sekų kiekio tarp GSP ir ProMFS algoritmų palyginimas (minimalus dažnumas ProMFS yra lygus *min_sup*=8, …, 14)



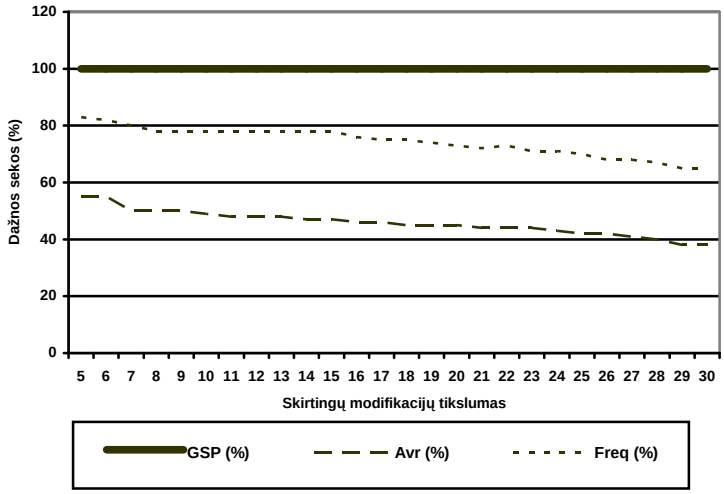
32 paveikslas. GSP ir ProMFS laiko sąnaudų palyginimas (minimalus dažnumas ProMFS lygus *min_sup*=8)

Panagrinėkime tikimybinio algoritmo modifikaciją, naudojant dažniausių atstumų matricą.



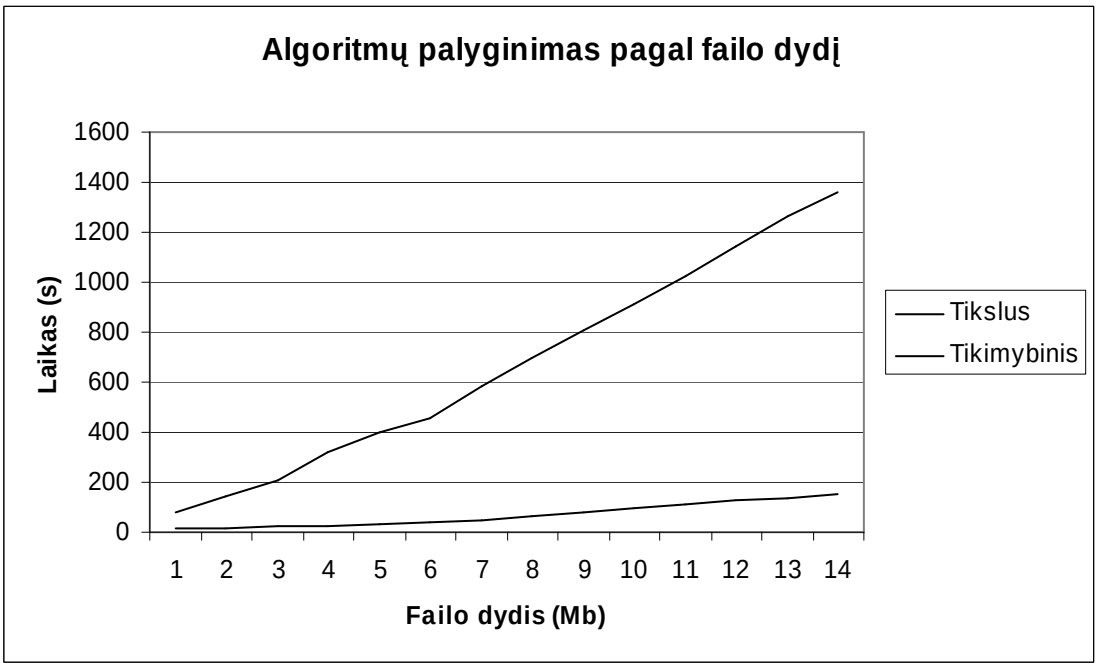
33 paveikslas. Tikimybinio algoritmo su dažniausių atstumų matrica palyginimas su GSP

Galima padaryti išvadą, kad tikimybinio algoritmo su vidurkių matrica bei tikimybinio algoritmo su dažniausių atstumų matrica veikimo laikas yra vienodas ir ženkliai geresnis nei GSP algoritmo.



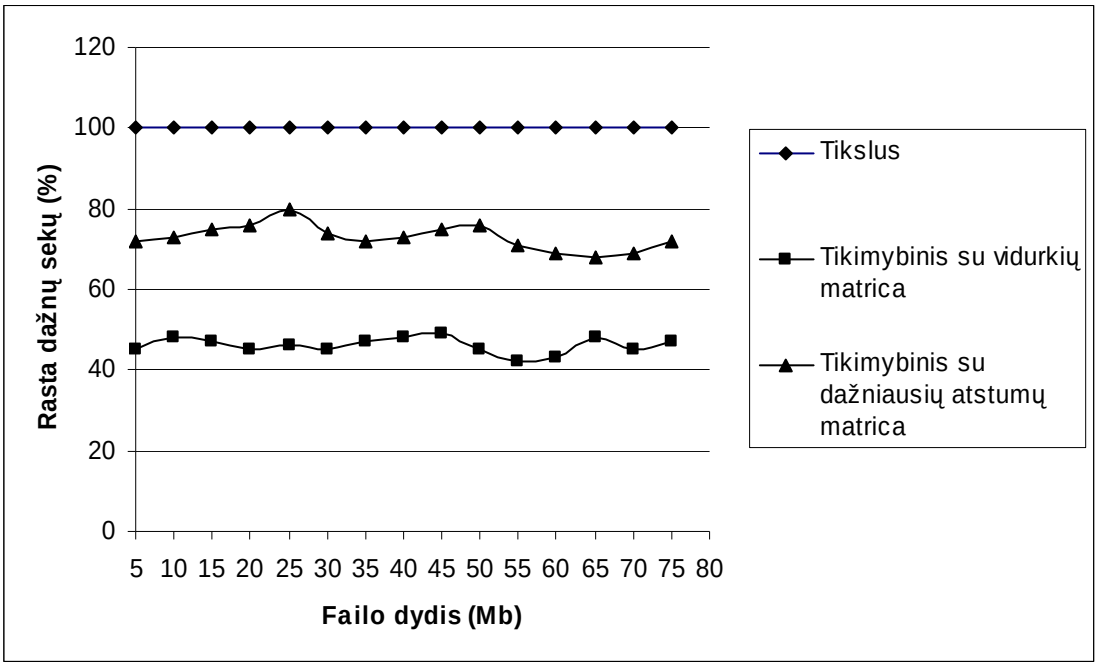
34 paveikslas Dviejų modifikacijų tikslumo palyginimas

34 paveiksle pavaizduotas praradimų palyginimas tarp tikimybinio algoritmo su vidurkių matrica ir tikimybinio algoritmo modifikacijos su dažniausių atstumų matrica. Matome, kad antroji modifikacija veikia geriau ir turi mažesnį praradimų kiekį.



35 paveikslas. Tikimybinio algoritmo laiko sąnaudų palyginimas su GSP, priklausomai nuo failo dydžio

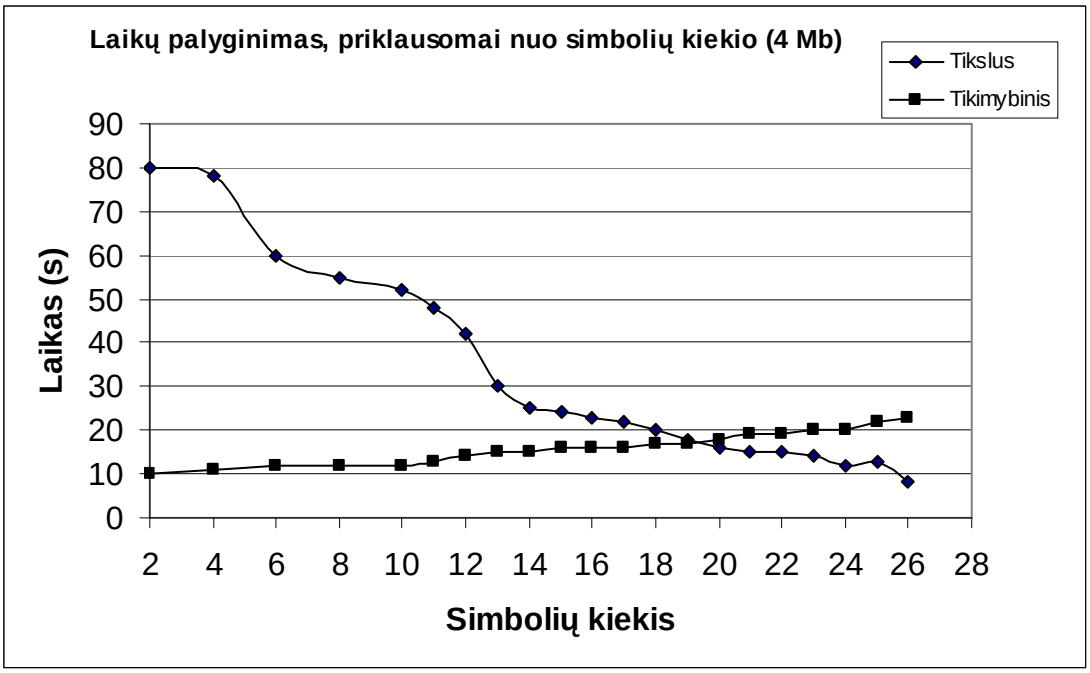
35 paveiksle pavaizduota, kaip keičiasi laiko sąnaudos, priklausomai nuo failo dydžio. Matome, kad GSP algoritmas yra pakankamai jautrus failo dydžio padidėjimui, ir laiko sąnaudos didėja labai greitai. Tuo tarpu tikimybinio algoritmo (su abiem modifikacijomis) laiko sąnaudos didėja nežymiai.



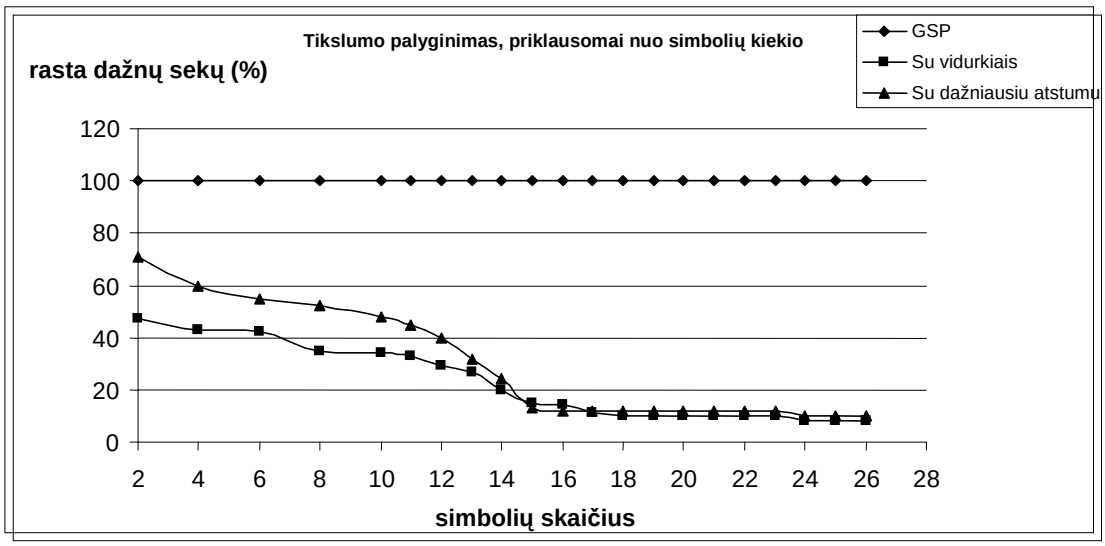
36 paveikslas. Tikimybinių algoritmų tikslumų palyginimas su GSP algoritmu pagal failo dydį.

Kaip matyti iš 36 paveikslo, priklausomai nuo failo dydžio padidėjimo tikimybinio algoritmų su skirtingom modifikacijom tikslumas yra sąlyginai pastovus, Taigi, darytina išvada, kad tikslumas nepriklauso nuo failo dydžio.

37 paveiksle matome tikimybinio algoritmo sąnaudų palyginimą su GSP algoritmų, priklausomai nuo simbolių kiekio. Padidinus simbolių kiekį tokio pačio dydžio faile, tikimybinio algoritmo efektyvumas (laiko atžvilgiu) sumažėja, o GSP atvirkščiai didėja (mažėja laiko sąnaudos).



37 p**aveikslas.** Tikimybinių algoritmų laiko sąnaudų palyginimas su GSP algoritmu pagal simbolių kiekį



38 paveikslas. Tikslumo palyginimas, priklausomai nuo simbolių kiekio

Iš 38 paveikslo matome, kad didėjant simbolių skaičiui tikimybinių algoritmų tikslumas mažėja. Taigi, galima daryti išvadą, kad abiejų modifikacijų tikimybiniai algoritmai ne visada veikia gerai.

Sugeneruokime failus su skirtingais simbolių (elementų) kiekiais kartu su viena dažna seka. Panagrinėkime juos su skirtingais modelinės sekos ilgiais. Rezultatai pateikti 13 lentelėje.

13 lentelė. ProMFS algoritmo eksperimentų rezultatai

Skirtingų simb.	d	v	ModSeka	Ar rado?
20	20	r	100	+
20	50	r	100	+
20	120	r	100	+
20	200	r	100	+
20	20	d	25	+
20	50	d	25	+
20	120	d	25	+
20	200	d	25	+
19	20	r	100	+
19	50	r	100	+
19	120	r	100	+
19	200	r	100	+
19	20	d	23	+
19	50	d	23	+
19	120	d	23	+
19	200	d	23	+
18	20	r	100	+
18	50	r	100	+
18	120	r	24	+
18	200	r	18	+
18	20	d	8	+
18	50	d	8	+
18	120	d	8	+
18	200	d	8	+
17	20	r	26	+
17	50	r	100	+
17	120	r	100	+
17	200	r	26	+
17	20	d	8	+
17	50	d	8	+
17	120	d	8	+
17	200	d	8	+
15	20	r	49	+
15	50	r	100	+
15	120	r	26	+

Skirtingų simb.	d	v	ModSeka	Ar rado?
15	200	r	26	+
15	20	d	100	+
15	50	d	100	+
15	120	d	100	+
15	200	d	100	+
10	20	r	24	+
10	50	r	11	+
10	120	r	4	+
10	200	r	1	+
10	20	d	100	+
10	50	d	100	+
10	120	d	100	+
10	200	d	100	+
5	20	r	12	+
5	50	r	1	+
5	120	r	11	+
5	200	r	9	+
5	20	d	4	+
5	50	d	4	+
5	120	d	4	+
5	200	d	4	+

Kaip matome, visais atvejais dažniausia seka yra randama. Iš to galima daryti išvadą, kad nors tikimybinis algoritmas su abejomis modifikacijomis ir turi dažnų sekų praradimus, tačiau tai niekaip neatsiliepia į dažniausios sekos radimą. Vadinasi, bet kuriuo atveju dažniausia seka yra surandama.

5.5. Dažnų sekų nustatymas genetinių duomenų bazėse

Dažnų sekų nustatymo algoritmai dažnokai naudojami bioinformatikoje. **Deoksiribonukleorūgštis** (**DNR**, seniau *dezoksiribonukleino rūgštis*) – tai nukleorūgštis, esanti kiekvienoje gyvoje ląstelėje, daugiausia jos branduolyje. Joje (jos azotinių bazių sekoje) yra tripletiniu kodu užkoduota genetinė informacija, kuri perduodama dalijimosi metu naujoms ląstelėms, o per lytines ląsteles - individo palikuonims.

Deoksiribonukleorūgštis susideda iš deoksiribonukleotidų. Deoksiribonukleotidas yra sudarytas iš heterociklinės azoto bazės ir angliavandenio deoksiribozės. Deoksinukleotidai skiriasi azoto bazėmis. Pagrindinės yra keturios: adeninas (A), guaninas (G), citozinas (C) ir

timinas (T). Esminiai duomenys susideda iš žmogaus ir kt. organizmų genomų ir proteomų, baltymų 3D struktūrų ir funkcijų, mikromatricos duomenų, metabolizmo ypatybių, ląstelių linijų, biologinės įvairovės ir pan. Biotechnologijos tyrimų metu gaunami didžiuliai informacijos kiekiai. Milžiniška duomenų gausybė bus naudinga tik tuomet, jei bus sukurti pažangūs apdorojimo įrankiai, priemonės. Didžiulių duomenų kiekių skaitmeninis kodavimas, indeksavimas, ieškojimas reikalauja ypač gerų technologijų ir modernių programinių sprendimų. Informacinės technologijos, ypač internetas, naudojamos rinkti, platinti, galimybe naudotis bet kada ir bet kur – auganti informacija, kuri vėliau analizuojama matematiniais ir statistiniais metodais. Bioinformatika yra nukleorūgščių (genų ir RNR) sekos, baltymų sekos ir struktūrinės informacijos įrašymas, anotavimas, saugojimas, analizavimas ir paieška/išrinkimas. Tai apima sekų ir struktūrinės informacijos duomenų bankus, o taip pat ir metodus pasiekti, ieškoti, atvaizduoti ir atrinkti informaciją. Dėl šių priežasčių bioinformatika naudojama nustatant ir analizuojant viso genomo seką, eksperimentinėje analizėje įtraukiant tūkstančius genų, analizuojant DNR fragmentus ir grandines, palyginamojoje analizėje tarp rūšių ir rūšies viduje, medicininiuose taikymuose, pavyzdžiui nustatant genetines ligas, teismuose, žemės ūkyje [103].

Genetinės duomenų bazės daugiausia yra sudarytos iš nukleorūgščių sekų ir iš jų gaunamų baltymų sekų. O tai sudaro milijonus nukleotidų, kurių saugojimas ir organizavimas tampa jau nebe tokia trivialia užduotimi.

Skiriami trys analizės lygiai.

1. Vieno geno (baltymo) sekos analizė. Pavyzdžiui:
    - ✓ Panašumas su kitais žinomais genais.
    - ✓ Filogenetiniai medžiai; evoliuciniai ryšiai.
    - ✓ Tiksliai nustatytų sekos sričių atpažinimas.
    - ✓ Sekos požymiai (fizinės savybės, praimerių prisitvirtinimo, mutavusios sekos).
    - ✓ Lokalizacijos prognozavimas subląsteliniu lygiui.
    - ✓ Antrinės ir tretinės struktūros prognozavimas.
2. Išbaigto genomo analizė. Pavyzdžiui:
    - ✓ Kurios genų šeimos yra, o kurių trūksta?
    - ✓ Genų padėtis chromosomoje, koreliacija su funkcija ar evoliucija.
    - ✓ Genų šeimų išplėtimas/dvigubinimas.
    - ✓ Trūkstamų enzimų atpažinimas.
    - ✓ Didelio masto įvykiai organizmo evoliucijoje.

3. Genų ir genomų analizė, siekiant gauti praktiškai panaudojamus duomenis. Pavyzdžiui:
    - ✓ Išraiškos analizė; mikromatricos duomenys.
    - ✓ Proteomika. Kovalentinė modifikacija.
    - ✓ Pakitusių fenotipų ir genotipų palyginimas.
    - ✓ Biocheminių kelių palyginimas ir analizė.
    - ✓ Esminių genų ar genų, įtraukiamų į specifinius procesus, atpažinimas.

Biologinė informacija yra kaupiama tarptautiniuose informaciniuose bankuose. Paprasčiausia duomenų bazė gali būti vienas failas, talpinantis daug įrašų, kurių kiekvienas turi tokią pačią informacijos struktūrą. Tačiau dažniausiai tai didelės apimties duomenų bazės, paprastai susietos su kompiuterine programa skirta atnaujinti, ieškoti, atrinkti duomenis, saugomus sistemos viduje.

Paprastai duomenų bazėms yra keliami tokie reikalavimai.

- ✓ kuo paprastesnis kreipimasis į duomenų bazę;
- ✓ metodas, kuris atrinktų geriausią informaciją pagal vartotojo (specifinius) užklausimus.

Duomenų bazės gali būti įvairių tipų:

- ✓ Viešos ir privačios (serveriai, paieškos programos, bioinformatikos įrankiai).
- ✓ Pirminės ir antrinės.
- ✓ Makromolekulių (DNR sekos, baltymų amino rūgščių sekos, baltymų trimatės struktūros) ir mažų molekulių.
- ✓ Orientuotos tekstui (žinių bazė).

Pastaruoju metu bioinformatikos specialistai daug laiko praleidžia kurdami genetinių duomenų bazes. Tokios duomenų bazės gali būti viešos, kaip *GenBank* (genų bankas), *PDB* (baltymų duomenų bankas) arba privačios, kuriomis naudojasi tik atskiri vartotojai, pavyzdžiui vykdantys tam tikrus projektus arba priklausantys konkrečioms biotechnologijų kompanijoms. Laisvam priėjimui prie tokių duomenų bazių keliami labai dideli reikalavimai jau vien dėl to, kad bioinformatikos duomenų vartotojai naudoja skirtingas kompiuterio platformas. DNR ir RNR yra baltymai, kurie saugo paveldimą organizmo informaciją.

Populiariausios duomenų bazės susideda iš ilgų nukleotidų (*guanine, adenine, thymine, cytosine* and *uracil*) ir/arba amino rūgščių (*threonine, serine, glycine,* t.t.) sekų. Kiekviena nukleotidų arba amino rūgščių seka atitinkamai simbolizuoja tam tikrą geną ar baltymą. Sekos sudaromos naudojant, vienos raidės pažymėjimus. Tokį informacijos kodavimą naudoja

daugelis duomenų bankų, nes tai sumažina saugomos informacijos kiekį ir pagreitina analizės vyksmą. Pavyzdžiui, DNR ir RNR sekos sudaromos, naudojant raides iš 1 lentelės [104].

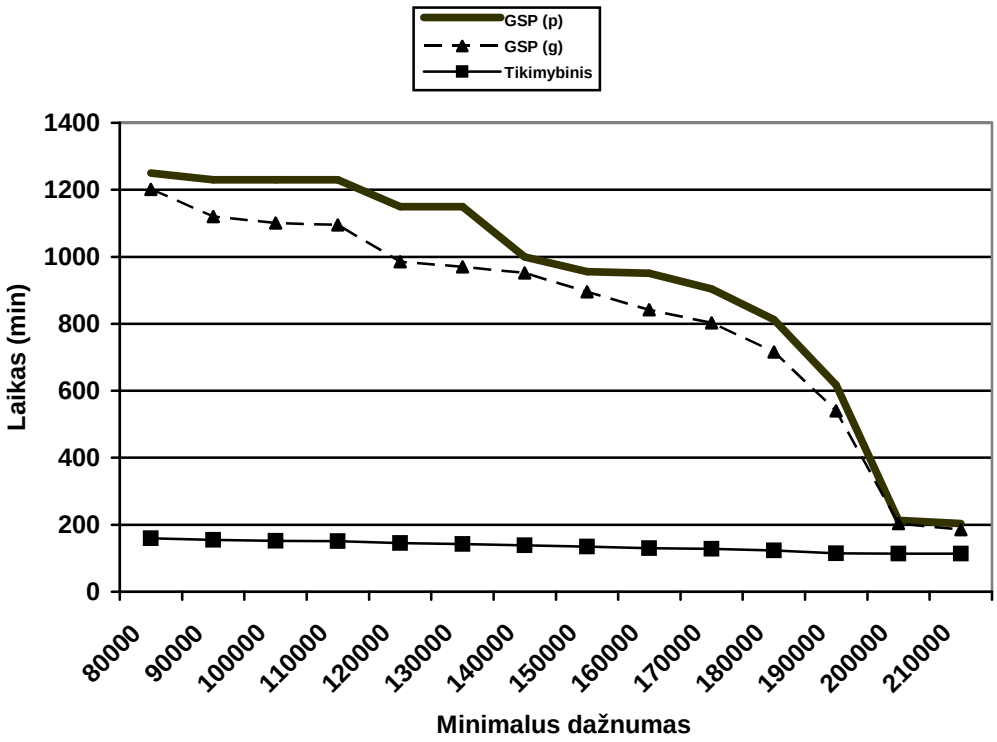
16 lentelė. Nukleotidai

DNR	*adenine*	*guanine*	*cytosine*	*thymine*
	A	**G**	**C**	**T/U**
RNR	*adenine*	*guanine*	*cytosine*	*uracil*

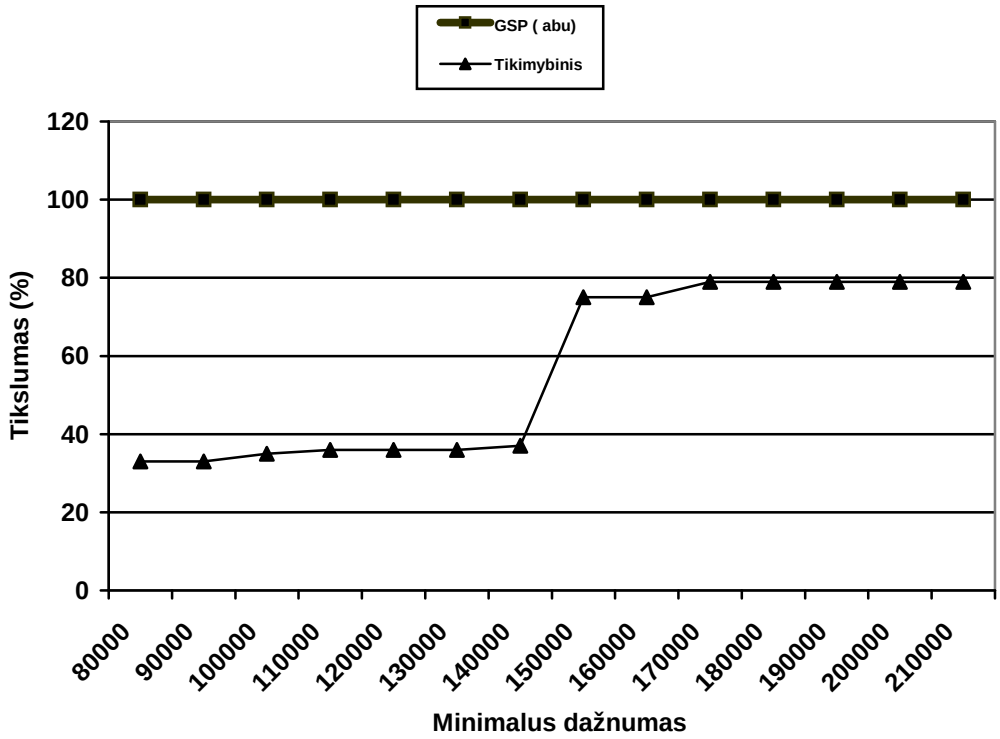
Savo algoritmams demonstruoti buvo paimti duomenys iš laisvai prieinamo genomo banko, esančio adresu ftp://ftp.ncbi.nih.gov/genbank/.

Kaip pavyzdį panagrinėkime vieną chromosomų seką (*Hs1_34565 Homo sapiens chromosome 1 genomic contig*), kurios dydis yra 250 Mb. Nagrinėsime trimis algoritmais: patobulintu GSP algoritmu „į plotį“, patobulintu GSP algoritmu „į gylį“ bei tikimybinių algoritmu, naudojančiu dažniausių atstumų matricą.

Rezultatai matomi 39 ir 40 paveiksle.



39 paveikslas. Chromosomos *Hs1*_34565 laiko sąnaudų palyginimas



40 paveikslas. Chromosomos *Hs1*_34565 tikslumo palyginimas

Iš rezultatų matosi, kad mažiausios sąnaudos yra tikimybinio algoritmo. Skirtumas gana didelis. Tačiau iš 40 paveikslo taip pat matosi, kad tikimybinis algoritmas praranda tam tikras dažnas sekas. Palyginus dvi GSP modifikacijas, paaiškėja, kad šiuo atveju metodas gylyn yra efektyvesnis. Pagal rezultatus dažniausios didžiausio ilgio sekos yra TTTTTTTTTTTTTTT ir AAAAAAAAAAAAAA. Iš bandymų buvo nustatyta, kad chromosomose vyrauja iš eilės einantys nukleiotidai adeninas ir timinas (T, A). Nukleotidas G (guaninas) pasitaiko rečiausiai. Jis 10 kartų retesnis nei adeninas ir timinas. Citozinas irgi yra sąlyginai retas. Jis du kartus dažnesnis nei guaninas, tačiau 5 kartus retesnis nei adeninas ar timinas.

5.6. Išvados

Šiame skyriuje buvo atlikti algoritmų efektyvumo (laiko sąnaudų bei atminties panaudojimo) palyginimai su dirbtiniais bei realiais duomenimis. Galima padaryti tokias išvadas:

1. Modifikuotas GSP algoritmas naudoja mažiau operatyviosios atminties, nei standartinis GSP algoritmas. Laiko sąnaudos skiriasi nedaug.
2. Pateiktas naujas rekursinis GSP algoritmo realizavimo būdas, tikrinantis sekas ne „į plotį“, kaip standartinis GSP algoritmas, o „į gylį“. Jo efektyvumas priklauso nuo duomenų bazės pobūdžio. Jei joje dominuoja dažnos sekos, kuriose iš eilės eina tie patys simboliai (pvz. AAAAABBBCCC, CCCAAABBB, BBBCCCAA ir t.t.), jis veikia efektyviau už standartinę GSP algoritmo paiešką „į plotį”. Tačiau jei duomenų bazėje dominuoja dažnos sekos, neturinčios vienodų simbolių (pvz. ABCACBACBA, BACACBABAC, ir t.t.), tai rekursinis algoritmas veikia lėčiau.
3. Tyrinėjant tikimybinio ProMFS algoritmo efektyvumą, eksperimento metu buvo išnagrinėtas tekstas iš 10000 raidžių. Gauti rezultatai buvo sulyginti ir nustatyta, kad ProMFS algoritmas veikia efektyviai tada, kai yra pakankamai didelis minimalus dažnumas. Greičio atveju tikimybinis algoritmas yra žymiai greitesnis už tikslųjį GSP algoritmą, kadangi jis visą duomenų bazę peržiūri tik vieną kartą ir suranda reikiamas tikimybines charakteristikas.
4. Tikimybinio algoritmo modifikacijos pagrindinė idėja yra ta, kad atstumų vidurkių matrica yra keičiama į dažniausių atstumų matricą. Atlikus tyrimus buvo nustatyta, kad ši modifikacija veikia tiksliau ir turi mažiau dažnų sekų praradimų. Laiko sąnaudos šių dviejų modifikacijų yra vienodos.
5. Didėjant nagrinėjamo failo dydžiui, laiko sąnaudos GSP algoritme didėja gana greitai, kai tuo tarpu tikimybinio algoritmo su abejomis modifikacijomis laiko sąnaudos didėja iš lėto. Taigi, GSP algoritmas yra labiau jautrus failo dydžiui negu tikimybinis algoritmas.
6. Didinant simbolių kiekį ir paliekant tą patį failo dydį, GSP laiko sąnaudos mažėja, kadangi automatiškai mažėja dažnų sekų dydis, o tikimybinio algoritmo sąnaudos didėja. Taigi, esant nedideliam failo dydžiui, bet dideliam skirtingų elementų kiekiui, tikimybinis algoritmas nepasiteisina. Jis turi ir daug praradimų (praradimų kiekis didėja, priklausomai nuo elementų padidėjimo bei dažnų sekų sumažėjimo), didėja ir laiko sąnaudos.

7. Tikimybinis algoritmas su abiem modifikacijomis labai efektyviai ir greitai suranda pačią dažniausią seką nagrinėjamoje duomenų bazėje. Modifikacija su dažniausių atstumų matricą veikia efektyviau. Bet kuriuo atveju, jei mūsų nagrinėjamoje duomenų bazėje yra labai dažna seka, ji yra surandama, o prarandamos tik tos sekos, kurių dažnumas yra sąlyginai nedidelis.
8. Norint efektyviai naudoti tikimybinį algoritmą, reikia pasirinkti pakankamai didelį minimalų dažnumą.

6. IŠVADOS

Disertacijoje buvo nagrinėjama dažnų sekų paieška dideliuose duomenų masyvuose. Tyrimo metu buvo išaiškinti populiariausi dažnų sekų radimo būdai: paieška platyn ir paieška gilyn ir hibridinė paieška. Ieškant dažnas sekas būtina žinoti vartotojo nustatytą pradinį dažnumo „slenkstį“ *min_sup*. Laikoma, kad seka yra dažna tada ir tik tada, kai jos dažnumas didesnis už minimalų dažnumą *min_sup*. Viena pagrindinė dažnų sekų radimo taisyklė yra ta, kad kiekvienos dažnos sekos visi posekiai irgi yra dažni. Tuo pačiu galime pastebėti, kad, jeigu sekoje yra nors vienas nedažnas vidinis posekis, tai mūsų seka irgi bus nedažna. Ši taisyklė leidžia mums eliminuoti daugelį nedažnų sekų, net jų netikrinus.

Disertacijoje yra realizuotas ir patobulintas standartinis GSP algoritmas, skirtas dažnų sekų radimui dideliuose duomenų masyvuose. Taip pat buvo realizuotas tikslaus dažnų sekų nustatymo algoritmo GSP modifikacija, krios metu iš nedažnų sekų nebuvo generuojamos kito lygmens sekos-kandidatės. GSP algoritmas buvo taip pat realizuotas, panaudojant rekursinį dažnų sekų nustatymo būdą „gilyn“.

Pateiktas naujas tikimybinis algoritmas ProMFS su dviem jo modifikacijomis: naudojant vidurkių matricą bei dažniausių atstumų matricą. Paprastu pavyzdžiu išnagrinėti šių dviejų variantų veikimo skirtumai.

Tyrimo metu buvo atlikti šie palyginimai:

- Buvo pateikti algoritmų palyginimai pagal laiko ir atminties sąnaudas;
- Buvo pateikti algoritmų palyginimai pagal nagrinėjamos duomenų bazės pobūdį;
- Pateikti tikimybinio algoritmo su dviem modifikacijomis ir tikslaus GSP algoritmo palyginimai pagal laiko sąnaudas, tikslumą ir šių charakteristikų priklausomybę nuo failų dydžio bei elementų kiekį nagrinėjamame faile;
- Buvo išnagrinėti realūs genetiniai duomenys su GSP ir ProMFS algoritmais.

Atlikus tyrimus, padarytos tokios išvados:

1. GSP kad generuoja potencialias sekas-kandidates, kurios gali būti dažnos, t.y. pakankamai dažnai pasikartoti duomenų bazėje. Sekų dažnumą apsprendžia tam tikras sveikas skaičius, kuris vadinamas minimaliu dažnumu. Visos sekos, kurių pasikartojimo skaičius didesnis už minimalų dažnumą, laikomos dažnomis. GSP algoritmas eliminuoja nedažnas sekas net jų netikrinant, tačiau vis tiek saugo operatyvioje atmintyje. T.y. jeigu atitinkamame lygmenyje pasitaiko dažnų sekų,

tai pirmiausia iš jų yra generuojamos kito lygmens sekos kandidatės ir tik po to atmetamos tos, kurios turi nedažnus posekius. Mūsų pasiūlytoje modifikacijoje iš nedažnų sekų net negeneruojamos naujo lygmens sekos-kandidatės ir yra eliminuojamos iš karto. Tai leidžia taupyti pagrindinės atminties kiekį.

2. Pateiktas naujas rekursinis GSP algoritmo realizavimo būdas, tikrinantis sekas ne „į plotį“, kaip standartinis GSP algoritmas, o „į gylį“. Jo efektyvumas priklauso nuo duomenų bazės pobūdžio. Jei joje dominuoja dažnos sekos, kuriose iš eilės eina tie patys simboliai (pvz. AAAAABBBCCC, CCCAAABBB, BBBCCCAA ir t.t.), jis veikia efektyviau už standartinę GSP algoritmo paiešką „į plotį“. Tačiau jei duomenų bazėje dominuoja dažnos sekos, neturinčios vienodų simbolių (pvz. ABCACBACBA, BACACBABAC, ir t.t.), tai rekursinis algoritmas veikia lėčiau.
3. Pagrindinė naujo tikimybinio algoritmo ProMFS idėja remiasi statistinėmis charakteristikomis: elemento pasirodymo tikimybe sekoje, tikimybe, kad vienas elementas eis po kito, atstumo vidurkio tarp dviejų elementų pagrindinėje sekoje. Remiantis šiomis charakteristikomis, generuojama nauja, žymiai trumpesnė modelinė seka $\widetilde{C}$, kuri po to analizuojama GSP algoritmu (arba kokiu nors kitu tiksliu algoritmu) ir daroma išvada, kad gauti GSP algoritmu dažni posekiai modelinėje sekoje, bus dažni posekiai ir pagrindinėje sekoje. Eksperimento metu buvo išnagrinėtas tekstas iš 10000 raidžių su GSP ir ProMFS algoritmu. Gauti rezultatai buvo sulyginti ir nustatyta, kad ProMFS algoritmas veikia efektyviai tada, kai yra pakankamai didelis minimalus dažnumas. Greičio atveju tikimybinis algoritmas yra žymiai greitesnis už tikslųjį GSP algoritmą, kadangi jis visą duomenų bazę tik vieną kartą peržiūri ir suranda reikiamas tikimybines charakteristikas.
4. Tikimybinio algoritmo modifikacijos pagrindinė idėja yra ta, kad atstumų vidurkių matrica yra keičiama į dažniausių atstumų matricą. Atlikus tyrimus buvo nustatyta, kad ši modifikacija veikia tiksliau ir turi mažiau dažnų sekų praradimų. Laiko sąnaudos šių dviejų modifikacijų yra vienodos.
5. Didėjant nagrinėjamo failo dydžiui, laiko sąnaudos GSP algoritme didėja gana greitai, kai tuo tarpu tikimybinio algoritmo su abiejomis modifikacijomis laiko sąnaudos didėja iš lėto. Tokiu būdu GSP algoritmas yra labiau jautrus failo dydžiui negu tikimybinis algoritmas.

6. Didinant simbolių kiekį ir paliekant tą patį failo dydį, GSP laiko sąnaudos mažėja, kadangi automatiškai mažėja dažnų sekų dydis, o tikimybinio algoritmo sąnaudos didėja. Taigi, esant nedideliam failo dydžiui, bet dideliam skirtingų elementų kiekiui, tikimybinis algoritmas nepasiteisina. Jis turi ir daug praradimų (praradimų kiekis didėja, priklausomai nuo elementų padidėjimo bei dažnų sekų sumažėjimo), didėja ir laiko sąnaudos.
7. Tikimybinis algoritmas su abiem modifikacijomis labai efektyviai ir greitai suranda pačią dažniausią seką nagrinėjamoje duomenų bazėje. Modifikacija su dažniausių atstumų matricą veikia efektyviau. Bet kuriuo atveju, jei mūsų nagrinėjamoje duomenų bazėje yra labai dažna seka, ji yra surandama, o prarandamos tik tos sekos, kurių dažnumas yra sąlyginai nedidelis.
8. Norint efektyviai naudoti tikimybinį algoritmą, reikia pasirinkti pakankamai didelį minimalų dažnumą.

Literatūros sąrašas

[1] Mohammed J. Zaki. Scalable Algorithms for Association Mining. *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, May/June, 2000, p. 372–390.

[2] Mohammed J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences, *Machine Learning*, Volume 42, Numbers 1-2 / January, 2001, p. 31–60.

[3] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, DC, May 26–28, 1993, p. 207–216.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, August 29–September 1, 1994.

[5] Yi-Hung Wu, Chia-Ming Chiang, Arbee L.P. Chen. Hiding Sensitive Association Rules with Limited Side Effects. *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, Jan., 2007, p. 29–42.

[6] Wen-Yang Lin, Ming-Cheng Tseng, Ja-Hwung Su. A Confidence-Lift Support Specification for Interesting Associations Mining. *Advances in Knowledge Discovery and Data Mining : 6th Pacific-Asia Conference, PAKDD 2002, Taipel, Taiwan, May 6–8, 2002. Proceedings*, Springer Berlin / Heidelberg, 2004, p. 148.

[7] Marcela Xavier Ribeiro and Marina Teresa Pires Vieira. A New Approach for Mining Association Rules in Data Warehouses. *Flexible Query Answering Systems*, Volume 3055/2004, Springer Berlin / Heidelberg 2004, p. 98–110.

[8] Jian Pei, Haixun Wang, Jian Liu, Ke Wang, Jianyong Wang, Yu P.S. Discovering Frequent Closed Partial Orders from Strings. *Knowledge and Data Engineering, IEEE Transactions* on Volume 18, Issue 11, Nov, 2006, p. 1467–1481.

[9] Congnan Luo, Anil L. Pereira and Soon M. Chung. Distributed Mining of Maximal Frequent Itemsets on a Data Grid System. *The Journal of Supercomputing*, Volume 37, Number 1 / July, 2006.

[10] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifer for database mining applications. *Proceedings of the 18th International Conference on Very Large Data Bases*, August, 1992, p. 560–573.

[11] Brock Barber, Howard J. Hamilton. Algorithms for Mining Share Frequent Itemsets Containing Infrequent Subsets. *Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD 2000, Lyon, France, September, 2000. Proceedings*, Springer Berlin / Heidelberg, 2004.

[12] Gösta Grahne and Jianfei Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. *Proceeding of the First IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (*FIMI'03*), Melbourne, FL, Nov, 2003.

[13] J.S.Park, M-S. Chen, P.S. YU. An effective hash based algorithm for mining association rules. *In M.J.Carey and D.A. Schneider, editors, Proceedings of the 1995 ACM SIG-MOD International Conference on Management of Data*, San Jose, California, 1995, p. 175–186.

[14] R. Agrawal and R.Srikant. Fast algorithms for mining association rules. *Proc. of Intl. Conf. On Very Large Databases (VLDB)*, Sept, 1994.

[15] R.Agrawal and R.Srikant. Mining sequential patterns. *In P.S.Yu and A.L.P. Chen, editors, Proc.11the Int. Conf. Data engineering. ICDE*, p. 3-14. IEEE pages 6-10, 1995.

[16] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifer for database mining applications. *Proceedings of the 18th International Conference on Very Large Data Bases*, August, 1992, p. 560–573.

[17] E. Woronowicz. Relations Defined on Sets. *Journal of Formalized Mathematics*, Volume 1, 1989, University of Bialystok.

[18] H. P. Sankappanavar. Congruence lattices of pseudocomplemented semilattices. *Algebra Universalis*. Birkhäuser Basel ISSN0002-5240 1420-8911 Volume 9, Number 1 / December, 2006, p. 304–316.

[19] R. Agrawa, H. Mannila, R. Srikant, H. Toivonen and A. Inkeri Verkamo. Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining* In U. Fayyad and et al editors, AAAI Press Menlo Park CA, p. 327-338.

[20] J. S. Park, M. Chen and P. S. Yu. An efective hash based algorithm for mining association rules. *In ACM SIGMOD Intl. Conf. Management of Data*, May, 1995.

[21] A. Savasere, E. Omiecinski and S. Navathe. An efficient algorithm for mining association rules in large databases. *In 21 st. VLDB Conf*. 1996.

[22] Karam Gouda and Mohammed J. Zaki. GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets. *Data Mining and Knowledge Discovery* Springer Netherlands, Volume 11, Number 3, November, 2005, p. 223–242.

[23] Olaf Owe and Ole-Johan Dahl. Generator induction in order sorted algebras. *Formal Aspects of Computing*. Springer London, (Online) Volume 3, Number 1, January, 1991, p. 2–20.

[24] Fuzan Chen and Minqiang Li. An Efficiently Algorithm Based on Itemsets-Lattice and Bitmap Index for Finding Frequent Itemsets. *Lecture Notes in Computer Science* Springer Berlin / Heidelberg Volume 3614/2005, *Fuzzy Systems and Knowledge Discovery*, p. 420–429.

[25] Zaki, M.J. Scalable algorithms for association mining. *Knowledge and Data Engineering, IEEE Transactions.* Volume 12, May/Jun 2000, p. 372–390.

[26] Jochen Hipp. Algorithms for Association Rule Mining : A General Survey and Comparison. *ACM SIGKDD*, 2000,Volume 2, Iss.1, p. 58–65.

[27] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. *Sequential Pattern Mining Using a Bitmap Representation. Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining*, 2002, p. 429–435.

[28] Zaki Mohammed. Mining Non-Redundant Association Rules. *Data Mining and Knowledge Discovery*, Volume 3, 2004, p. 223–248.

[29] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara and Wei Li. Parallel Algorithms for Discovery of Association Rules. *Data Mining and Knowledge Discovery*. Springer Netherlands, Volume 1, Number 4, December, 1997, p. 343–373.

[30] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi. Adaptive constraint pushing in frequent pattern mining. *Proc. PKDD'03*, Volume 2838 of *LNAI*, Springer-Verlag, 2003, p. 47–58.

[31] J.-F. Boulicaut and B. Jeudy. Using constraint for itemset mining: should we prune or not? *Proc. BDA'00*, 2000, p. 221–237.

[32] S. Nijssen and J. N. Kok. A quickstart in frequent structure mining can make a diference. *In Proceedings of the 10th International Conference on Knowledge Discovery and Data mining (KDD)*, New York, NY, USA, 2004. ACM Press pages 647–652.

[33] S. Parthasarathy, M. J. Zaki, M. Ogihara, and W. Li. Parallel data mining for association rules on shared-memory systems. *Knowledge and Information Systems*, volume 3, 2001, p. 1–29.

[34] E.H. Han, G. Karypis, and V. Kumar. Scalable Parallel Data Mining for Association Rules. *Proc. ACM Conf. Management of Data*, ACM Press, New York, 1997, p. 277–288.

[35] D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. *Proceedings of the 31st Int. Conference on Very large data bases* (VLDB'05), VLDB Endowment, 2005, p. 709–720.

[36] N. Vanetik and E. Gudes. Mining frequent labeled and partially labeled graph

patterns. *Proceedings of the 20th Int. Conference on Data Engineering*

(ICDE '04), IEEE Computer Society, 2004, p. 91.

[37] P. Valtchev and R. Missaoui. Building concept (galois) lattices from parts: Generalizing the incremental methods. *In 9th Int. Conference on Conceptual Structures*, 2001, p. 290–303.

[38] M. Sahami. Learning classification rules using lattices. *Proceedings of the 8th European Conference on Machine Learning*, 1995, p. 334–346.

[39] E.M. Nguifo and P. Njiwoua. IGLUE: A lattice-based constructive induction system. *Proceedings of the 9th Int. Conference on Tools with Artificial Intelligence*, 1997, p. 75–76.

[40] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. *Proceedings of the 15th Int. Conference on Advanced Databases*, 1999, p. 361–381.

[41] N. Vanetik and E. Gudes. Mining frequent labeled and partially labeled graph patterns. *Proceedings of the 20th Int. Conference on Data Engineering (ICDE '04)*, IEEE Computer Society, 2004, p. 91.

[42] X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. *Proceedings of the 9th ACM SIGKDD Int. Conference on Knowledge discovery and data mining (KDD'03)*, ACM Press, 2003, p. 286–295.

[43] P. Tzvetkov, X. Yan, and J. Han. TSP: Mining top-k closed sequential patterns. *Proceedings of the 3rd IEEE Int. Conference on Data Mining*, 2003, p. 347–358.

[44] Chung, S.M.; Congnan Luo. Parallel mining of maximal frequent itemsets from databases. *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference* on Volume , Issue , 3–5 Nov, 2003, p. 134–139.

[45] Mahesh V. Joshi, Eui-Hong (Sam) Han, George Karypis, Vipin Kumar. Efficient Parallel Algorithms for Mining Associations. *Large-Scale Parallel Data Mining*, Springer Berlin / Heidelberg, Volume 1759/2000, p. 83.

[46] Shenoy, P.; Haritsa, J.; Sudarshan, S.; Bhalotia, G.; Bawa, M.; Shah, D. VIPER: A Vertical Approach to Mining Association Rules, 2002, http://dbpubs.stanford.edu/pub/2000-2.

[47] Salvatore Orlando, Paolo Palmerini, Raffaele Perego, Fabrizio Silvestri. An Efficient Parallel and Distributed Algorithm for Counting Frequent Sets. *High Performance Computing for Computational Science - VECPAR 2002: 5th International Conference*, Porto, Portugal, June 26-28, 2002. Selected Papers and Invited Talks, p. 421–435.

[48] Ja-Hwung Su; Wen-Yang Lin. CBW: an efficient algorithm for frequent itemset mining. *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference* on Volume , Issue , 5–8 Jan, 2004, p. 9.

[49] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An Effective Hash Based Algorithm for Mining Association Rules. *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, California, 22-25 May, 1995, p. 175–186.

[50] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen and A. Inkeri Verkamo. Finding Interesting Rules From Large Sets of Discovered Association Rules. *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, November 1994, p. 401–407.

[51] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining Optimized Association Rules for Numeric Attributes. *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Montreal, Quebec, Canada, 3–5 June, 1996 ,p. 182–191.

[52] Bing Liu, Wynne Hsu and Yiming Ma. Mining Association Rules with Multiple Supports. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99)*, SanDiego, CA, USA, August 15–18, 2002.

[53] Grahne, G.; Zhu, J. Fast algorithms for frequent itemset mining using FP-trees. *Knowledge and Data Engineering*, IEEE Transactions on Volume 17, Issue 10, Oct, 2005 p. 1347–1362.

[54] J. Wang, J. Han, and J. Pei. Closet: Searching for the best strategies for mining frequent closed itemsets. *Proceedings of ACM SIGKDD'03*, Washington, DC, 2003.

[55] Petko Valtchev, Rokia Missaoui, Robert Godin, Mohamed Meridji. Generating frequent itemsets incrementally: two novel approaches based on Galois lattice theory. Journal of Experimental & Theoretical Artificial Intelligence, Volume 14, Numbers 2-3/April 01, 2002, p. 115–142.

[56] Frans Coenen, Graham Goulbourne and Paul Leng. Tree Structures for Mining Association Rules. *Data Mining and Knowledge Discovery*, Volume 8, Number 1 / January, 2004, p. 25–51.

[57] Raymond T. Ng, Laks V.S. Laksmanan, Jiawei Han and Alex Pang. Exploratory Mining and Pruning Optimization of Constarinted Association Rules. *ACM Sigmoid, 1998,* Seattle, WA, USA. p. 13–24.

[58] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases. *Proceedings of the 21nd International Conference on Very Large Databases*, Zurich, Swizerland, 1995, p. 432–444.

[59] Hannu Toivonen. Sampling Large Databases for Association Rules. *Proceedings of the 22nd International Conference on Very Large Databases*, Mumbai, India, 1996, p. 134–145.

[60] Roberto J. Bayardo Jr., Rakesh Agrawal, Dimitris Gunopulos. Constarint-Based Rule Mining in Large, Dense Databases. *Proceedings of the 15th International Conference on Data Engineering*, 23-26 March 1999, Sydney, Australia, p. 188–197.

[61] Sergey Brin, Rajeev Motwani, and Craig Silverstein, Beyond. Market Baskets: Generalizing Association Rules to Correlations. *Proceedings of the ACM SIGMOD Conference*, 1997, p. 265–276.

[62] Mohammed J. Zaki, Nagender Parimi, Nilanjana De, Feng Gao, Benjarath Phoophakdee, Joe Urban, Vineet Chaoji, Mohammad Al Hasan and Saeed Salem. Towards Generic Pattern Mining. *Formal Concept Analysis*. Springer Berlin / Heidelberg, Volume 3403/2005, p. 1–20.

[63] H.-C. Chang, C.-C. Hsu, and E. Chen. Mining Closed Frequent Itemsets for Incremental and Diminished Database with Lexicographic Tree Traversal**.** *Networks, Parallel and Distributed Processing, and Applications*, 2002, p. 368.

[64] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, May, 2000, p. 1–12.

[65] D. Lin and Z. M. Kedem. Pincer-search: An efficient algorithm for discovering the maximum frequent set. *IEEE Transactions on Knowledge and Data Engineering*,

14(3) 2002, p. 553–566.

[66] M. J. Zaki, S. Parthasarathy, M. Ogihara, andW. Li. New algorithms for fast discovery of association rules. In D. H. H. Mannila and D. Pregibon, editors, *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997*, AAAI Press, 1997, p. 283–286.

[67] J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4) 2002, p. 750–767.

[68] Lin, D. and Kedem, Z. M. Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set. *In Pnx. Of the Sixth European Cant on Extending Database Technology*.

[69] Bayardo, R. J. Brute-Force Mining of High-Confidence Classification Rules. *Proc. of the Third Int '1 Con. on Knowledge Discovery and Data Mining*, 1997, p. 123–126.

[70] Jaturon Chattratichat, John Darlington, Moustafa Ghanem, and et. al, Large Scale Data Mining: Challenges and Responses. *Proceedings of the 3th International Conference on Knowledge Discovery and Data Mining*, August, 1997, p. 143–146.

[71] Congnan Luo, Anil L. Pereira and Soon M. Chung. Distributed Mining of Maximal Frequent Itemsets on a Data Grid System. *The Journal of Supercomputing*, Springer Netherlands, Volume 37, Number 1 / July, 2006, p. 71–90.

[72] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1997 1(3), p. 259–289.

[73] Craig Silverstein, Sergey Brin, Rajeev Motwani, and Jeffrey D. Ullman. Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, 4(2/3), 2000, p. 163–192.

[74] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining - A general survey and comparison. *SIGKDD Explorations*, 2(2) 2000, p. 1–58.

[75] Salvatore Orlando, Claudio Lucchese, Paolo Palmerini, Raffaele Perego, and Fabrizio Silvestri. Kdci: a multi-strategy algorithm for mining frequent sets. In Bart Goethals and Mohammed J. Zaki, editors, *FIMI'03: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations*, November, 2003.

[76] David Wai-Lok Cheung, Vincent T. Ng, Ada Wai-Chee Fu, and Yongjian Fu. Efficient Mining of Association Rules in Distributed Databases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, December, 1996, p. 911–922.

[77] Ashok Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995, p. 432–443.

[77] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to knowledge Discovery: An Overview. *Advances in Knowledge Discovery and Data Mining*, Edited by Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padraic Smyth, and Ramasamy Uthurusamy, AAAI Press, 1996, p. 1–34.

[78] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen and A. Inkeri Verkamo. Finding Interesting Rules From Large Sets of Discovered Association Rules, *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, November, 1994, p. 401–407.

[79] Jun-Lin Lin and M. H. Dunham. Mining Association Rules: Anti-skew Algorithms. *Proceedings of the 14th IEEE International Conference on Data Engineering*, Olando, Florida, February, 1998.

[80] Akihiro Inokuchi, Takashi Washio, Hiroshi Motoda. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. *Lecture Notes In Computer Science; Vol. 1910 archive Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, Springer-Verlag, 2000, p. 13–23.

[81] Yoshikawa M., Terai H. Apriori, Association Rules, Data Mining,Frequent Itemsets Mining (FIM), Parallel Computing. *Software Engineering Research, Management and Applications, Fourth International Conference*, 2006, p. 95–100.

[82] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules Between Sets of Items in Large Databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington, D.C., May, 1993, p. 207–216.

[83] Chang-Hung Lee Ming-Syan Chen,Cheng-Ru Lin. Progressive Partition Miner: An Efficient Algorithm for Mining General Temporal Association Rules. *Knowledge and Data Engineering*, July/August, 2003 (Vol. 15, No. 4), p. 1004–1017.

[84] David Wai-Lok Cheung, Jiawei Han, Vincent Ng, Ada Wai-Chee Fu, and Yongjian Fu, A Fast Distributed Algorithm for Mining Association Rules. *Proceedings of PDIS*, 1996.

[85] D. W. Cheung, J. Han, V. T. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *Proceedings of the 12th IEEE International Conference on Data Engineering*, February, 1996, p. 106–114.

[86] Chung-Wen Cho, Yi-Hung Wu and Arbee L.P. Chen. Effective Database Transformation and Efficient Support Computation for Mining Sequential Patterns. *Database Systems for Advanced Applications*, Springer Berlin / Heidelberg, Volume 3453/2005, p. 163–174.

[87] Garofalakis M, Rastogi R. and Shim K. Mining Sequential Patterns with Regular Expression Constraints. *IEEE Transactions on Knowledge and Data Engineering*, Volume 14, nr. 3, 2002, p. 530–552.

[88] Zhong-Yang Xiong; Yu-Fang Zhang; Dai-Jie Cheng. The unified storage method of three kinds of data mining resultsė. *Machine Learning and Cybernetics*. International Conference on Volume 1, Issue , 2–5 Nov, 2003, p. 284–288.

[89] Zaki, M.J.; Parthasarathy, S.; Wei Li; Ogihara, M. Evaluation of sampling for data mining of association rules. Research Issues in Data Engineering, 1997. *Proceedings. Seventh International Workshop on Volume*, Issue , 7–8 Apr, 1997, p. 42–50.

[90] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl Jianyong Wang Helen Pinto Qiming Chen Umeshwar Dayal, Mei-Chun Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *Knowledge and data engineering*, Volume 16, No. 11, November, 2004, p. 1424–1440.

[91] René A. García-Hernández, José Fco. Martínez-Trinidad and Jesús Ariel Carrasco-Ochoa. A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text. *Progress in Pattern Recognition, Image Analysis and Applications*, Springer Berlin / Heidelberg, Volume 3287/2004, p. 478–486.

[92] Wang, J. and Han, J. BIDE: Efficient mining of frequent closed sequences. *In 20th International Conference on Data Engineering*, IEEE Press, 2004, p. 79–90.

[93] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. & Hsu, M.-C. Freespan: frequent pattern-projected sequential pattern mining. *In 6th ACM SIGKDD International Conference Proc. Fifth Australasian Data Mining Conference (AusDM2006) 133 on Knowledge Discovery and Data Mining*, ACM Press, Boston, MA, USA, 2000, p. 355–359.

[94] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. *Proc. 1998 European Symp. Principle of Data Mining and Knowledge Discovery (PKDD '98),* Sept, 1998, p. 176–184.

[95] R. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang. Exploratory Mining and Pruning Optimizations of Constrained Associations Rules. *Proc. 1998 ACM-SIGMOD Int'l Conf. Management of Data (SIGMOD '98),* June, 1998, p. 13–24.

[96] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. *Proc. 2000 ACM SIGKDD Int'l Conf. Knowledge Discovery in Databases (KDD '00),* Aug, 2000, p. 355–359.

[97] X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. *Proc. 2003 SIAM Int'l Conf. Data Mining (SDM '03),* May, 2003, p. 166–177.

[98] R.J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-Based Rule Mining on Large, Dense Data Sets. *Proc. 1999 Int'l Conf. Data Eng. (ICDE '99),* Apr. 1999, p. 188–197.

[99] J. R. Griggs, P. Hanlon, A. M. Odlyzko and M. S. Waterman. On the number of alignments of *k* sequences. *Computer Science and Mathematics and Statistics*. Springer Japan, Volume 6, Number 2 / June, 1990, p. 133–146.

[100] Koji Iwanuma, Ryuichi Ishihara, Yo Takano, Hidetomo Nabeshima, Extracting Frequent Subsequences from a Single Long Data Sequence: A Novel Anti-Monotonic Measure and a Simple On-Line Algorithm. *Fifth IEEE International Conference on Data Mining (ICDM'05), 2005*, p. 186–193.

[101] Linhui Jiang, Howard J. Hamilton. Methods for Mining Frequent Sequential Patterns. *Advances in Artificial Intelligence: 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2003, Halifax, Canada, June 11–13, 2003. Proceedings,* Springer Berlin / Heidelberg, Volume 2671/2003, p. 486–491.

[102] Mannila, H., Toivonen, H. and Verkamo, A. I. Discovering frequent episodes in sequences. *In U. M. Fayyad & R. Uthurusamy, eds, '1st International*

Conference on Knowledge Discovery and Data Mining (KDD-95), AAAI Press, Menlo Park, CA, USA, Montreal, Quebec, Canada, p. 210–215.

[103] http://www.neuro.wustl.edu/neuromuscular/mother/dnarep.htm

[104] http://www.library.csi.cuny.edu/~davis/molbiol/lecture_notes/ bioinformatics_genomics/bioinformaticsIntro.html