KLAIPĖDOS UNIVERSITETAS
GAMTOS IR MATEMATIKOS MOKLSŲ FAKULTETAS
INFORMATIKOS KATEDRA

EDVARDAS RIMKUS
InfMag04 magistrantas

ALGORITMŲ INTELEKTUALAUS PROGRAMINIO AGENTO BŪSENAI ATPAŽINTI TYRIMAS

Magistrinis darbas

Mokslinio darbo vadovas prof. habil. dr. Antanas Andrius Bielskis
Darbo kuratorė asistentė Dalia Baziukaitė

Klaipeda, 2006

Reziumė

Adaptyvios intelektualios mokymo(si) aplinkos (VMA) kontekste analizuojama konceptualios klasterizacijos algoritmų taikymo galimybė, siekiant užtikrinti programinio agento gebėjimą ”jausti” kintančią aplinką ir atpažinti būsenas, kuriose jis atsiduria. Agento aplinka suprantama kaip VMA sąsaja su vartotoju ir VMA bazėje išsaugotas besimokančiojo modelis, kuris nuolat kinta mokymo(si) proceso eigoje. Agento gebėjimas "jausti" suvokiamas kaip agento savybė klasifikuoti besimokančiuosius, atsižvelgiant į jų turimą žinių lygį, kuris taip pat kinta mokymo(si) proceso eigoje. Taikant literatūroje aprašytus konceptualios klasterizacijos algoritmus, siekiama atrinkti geriausiai tinkantį probleminės srities specifikai, pritaikant realiems duomenims modeliuoti.

RAKTINIAI ŽODŽIAI: klasterizacija, grupavimas, konceptuali klasterizacija, algoritmas, virtuali mokymo(si) aplinka, intelektualus programinis agentas

Abstract

In the context of adaptive intellectual learning environment (VLE) possibility of using conceptual clustering algorithms is analyzed, trying to accomplish the ability of software agent to "feel" the changing environment and recognise the states it is in. Agent environment is understood as interface between the user of VLE and the students model, which is stored in the VLE and is constantly changing. Agents ability to "feel" is understood as agents ability to classify students, based on their knowledge level, which changes in the learning process. Using conceptual clustering algorithms found in the literature, we are trying to choose one which is most suited for the problem area, modifying it to model real data.

KEYWORDS: clustering, grouping, conceptual clustering, algorithm, virtual learning environment, intellectual software agent

Turinys

Iliustracijų sąrašas

Lentelės

ĮVADAS

Virtuali mokymo(si) aplinka (VMA) yra tinklinė aplikacija, kuri sudaro sąlygas mokytis ir pateikti mokomąją medžiagą nesant auditorijoje. Dauguma dabar naudojamų mokymo sistemų nėra adaptyvios. Adaptyviomis mokymo(si) sistemomis laikome prie besimokančiojo poreikių gebančias prisitaikyti sistemas. Jose adaptyvumo lygmuo gali būti labai įvairus. Pavyzdžiui mokymosi medžiaga gali būti pateikiama skirtingais pavidalais, gali skirtis jos išdėstymo lygis. Sistema gali sekti jau išmoktą medžiagą ir besimokančiajam vizualiai nurodyti kokia temą reikėtu toliau mokytis, arba grąžinti besimokantįjį kartoti ankstesnės temos. Medžiagos išdėstymo lygio sudėtingumas gali priklausyti nuo besimokančiojo žynių lygio. Aplinką, kuri pati sugeba priimti sprendimus besimokančiojo atžvilgiu ir keisti savo elgesį pakitus mokymo(si) aplinkos tam tikriems parametrams, vadinama adaptyvia intelektualia aplinka.

Tam tikri duomenys apie besimokančiuosius, kurie fiksuojami VMA duomenų bazėje (besimokančiojo modelis) [6], yra vertinga informacija [2, 3], kurios dėka galime besimokančiuosius suskirstyti į tam tikrą žinių lygį turinčias grupes, o vėliau gautus naujus duomenis priskirti į tam tikrą grupę (diskriminuoti). Intelektikoje, o konkrečiau matematinėje mokymosi teorijoje, toks procesas vadinamas konceptualia klasterizacija. Kai kalbame apie sistemą, gebančia savarankiškai priimti sprendimus besimokančiojo atžvilgiu, ją abstrahuojame pasinaudodami agento paradigma. Ši idėja suformuluota ir aptarta D. Baziukaitės straipsnyje [1]. Siekiant apmokyti agentą tam tikroje būsenoje priimti geriausią sprendimą (parinkti geriausiai tinkantį veiksmą) [14], sistema formalizuota, kaip Markovo sprendimų priėmimo procesas. Agento būsena nustatoma pagal tai kokiai grupei (klasteriui) priskiriamas besimokantysis, kurio atžvilgiu turi būti priimtas sprendimas. Tuo tikslu virtualios mokymo(si) sistemos priemonėse turi egzistuoti metodai užtikrinantys besimokančiųjų sugrupavimą ir diskriminavimą. Čia puikiai praverčia konceptualios klasterizacijos algoritmai [5]. Šiame darbe apžvelgiami keli klasterizacijos algoritmai, tokie kaip taisyklėmis pagrįstas, sprendimo medžio ir hibridinis [4].

Taisyklėmis paremti algoritmai stengiasi išskirstyti duomenis į klases. Pagrindiniai šio tipo algoritmų privalumai yra tie, kad jo sukurtos taisyklės yra suprantamos žmonėms, bei jas galima apjungti, jei jos yra nepriklausomos bei kiekvieną taisyklę galima patikrinti ar ji turi prasmę. Didžiausias šio algoritmo trūkumas yra jo nesugebėjimas atskleisti sąryšių tarp taisyklių.

Sprendimų medžiai pirma stengiasi ne ieškoti taisyklių, o surasti labiausiai išskirtinius atributus iš pateiktų duomenų. Šie algoritmai savo prasme labai panašūs į ”skaldyk ir valdyk” principą. Kadangi duomenys dažniausiai turi daugiau nei vieną savybę, tai pagal kurią savybę duomenis bus skaidomi apsprendžia entropija (tvarka sistemoje) [8] ir informacijos kiekio padidėjimas. Didžiausios problemos su sprendimo medžiais yra negalima jų mokyti nuosekliai, t.y. kiekvienam naujam duomeniui medis yra kuriamas iš naujo, bei augant medžio sudėtingumui iš jo gaunamų taisyklių kiekis labai didėja ir jos tampa nesuprantamos.

Hibridiniai algoritmai apjungia geriausias abiejų ankstesnių algoritmų savybes. Šie algoritmai priešingai nei sprendimo medžiai skaido duomenis keliais skirtingais būdais ir iš jų pasi-

renkamas geriausiai tinkantį. To pasėkoje nebereikia saugoti viso medžio atmintyje ir sudaromos taisyklės yra daug kompaktiškesnės, todėl yra sutaupomas didelis atminties kiekis. Kitas šio algoritmo pranašumas yra tai, kad jis greitas nes nereikia skaičiuoti entropijų.

Klasterizacijai mokymo imties duomenų tinkami įvairūs algoritmai. Vienas iš jų yra K-vidurkių algoritmas. Pagrindinis šio algoritmo privalumas yra: skaičiavimo greitumas, didėjant imčiai jo sparta palyginus su hierarchiniais algoritmais lėtėja ne taip greitai. Kadangi šiam algoritmui reikia žinoti į kiek rinkinių reikia skaidyti duomenis, todėl apdorojus duomenis reikia patikrinti gautą suskirstymą klasteriais, tai patikrinama pasinaudojant skirstymo koeficiento ir skirstymo entropijos metodais.

Šiame darbe didžiausią dėmesį skiriame konceptualios klasterizacijos algoritmams, siekdami atrinkti aukščiau suformuluotai problemai spręsti geriausiai tinkantį algoritmą. Tai atlikus bus išrinktas efektyviausias algoritmas [7] turintis geriausią kombinaciją veiksnių įtakojančių jo greitį, efektyvumą ir užimamos atminties kiekį. To dėka tampa įmanoma išspręsti intelektualiam programiniam agentui keliamas užduotis.

Darbo tikslas:atrinkti probleminės srities specifikai tinkančius algoritmus, pritaikant juos realiems duomenims modeliuoti.

Uždaviniai:

1. Išdėstyti, apibendrinti ir palyginti pasirinktus konceptualios klasterizacijos algoritmus.
2. Aprašyti ir palyginti algoritmus skirtus duomenų diskretizacijai.
3. Patikrinti ir įvertinti diskretizacijos algoritmų darbo efektyvumą testiniams probleminės srities duomenims.
4. Realiems probleminės srities duomenims pritaikius pasirinktą konceptualios klasterizacijos algoritmą, gauti taisykles, kurios leistų klasifikuoti besimokančiuosius.

1. Literatūros apžvalga

Žinių radimas (knowledge discovery) apibrėžiamas kaip "Netrivialios ankščiau nežinotos ir galbūt naudingos informacijos išgavimas iš duomenų" [4, 11, 23]. Visas žinių radimo procesas turi savyje pradinius duomenis, apibrėžtą žingsnių kiekį duomenų apdorojimui, tokių kaip: duomenų išrinkimas, savybių verčių parinkimas, transformacija, susiejimas su jau žinomais dalykais ir interpretacija. Žinių radimo proceso pagrindas metodas, kuris išgauna ir analizuoja duomenų modelius (pattern), santykius tarp grupių sprendžiamos problemos plotmėje. Duomenų modelių atradimui naudojami euristiniai metodai, kurie pagrinde remiamasi statistine analize, skaičių sistematika ir konceptualios klasterizacijos algoritmais. Konceptualios klasterizacijos algoritmai gali būti taikomi mišriems duomenims susidedantiems iš skaičiuojamųjų, simbolinių reikšmių. Mūsų atveju žinių radimo proceso pačioje pradžioje yra duomenys gauti iš VMA sistemos (testų rezultatai ir kt.), o pabaigoje informacija/taisyklės kaip skirstyti studentus į klasterius.

1.1. K-vidurkių metodas

K-vidurkių klasterizavimo metodas geriausiai nusakomas kaip skirstymo metodas. Priešingai nei hierarchiniai klasterizacijos metodai, k-vidurkių klasterizacijos metodas nesukuria medžio struktūros aprašančios elementus, o sukuria vieno lygio klasterius [12]. Kitas K-vidurkių klasterizacijos skirtumas yra tas, kad jis naudoja realius objektus, o ne jų aplinką, tai įgalina K-vidurkių metodo panaudojimą esant labai dideliems duomenų kiekiams.

K-vidurkių metodas kiekvieną stebėjimą vertina kaip atskirą objektą turintį savo buvimo vietą. Jo tikslas yra rasti skirsnius, kuriuose objektai klasterių viduje yra kaip galima arčiau vienas kito ir tuo pačių kaip galima toliau nuo objektų esančių kituose klasteriuose.

Kiekvienas klasteris skirstinyje yra aprašomas objektais esančiais jame ir jo centru. Kiekvieno klasterio centras yra taškas iki kurio visų objektų atstumų suma yra minimali. K-vidurkių metodas naudoja iteratyvų algoritmą, kuris minimizuoja atstumų sumą nuo kiekvieno iki jo klasterio centro visuose klasteriuose. Šis algoritmas perkėlinėja objektus iš klasterio į klasterį tol kol atstumų suma negali būti labiau minimizuota. To pasekmė yra klasteriai, kurie yra maksimaliai kompaktiški ir atskirti vienas nuo kito kiek tai įmanoma.

1.2. ITERATE algoritmas

Buvo pademonstruota, kad skirstant remiantis skaičiuojamais klasteriais, skirtingi pradiniai klasteriai gali sukurti visiškai skirtingus klasterizacijos algoritmus, ypač jei naudojama kvadratinė paklaida konverguoja į lokalų minimumą, o tai būdinga jeigu grupės nėra pakankamai atskirtos. Todėl "geri" pradiniai klasteriai yra pirmo svarbumo objektai norint pasiekti gerus klasterizacijos rezultatus [5]. Buvo pasiūlyta naudoti hierarchinės klasterizacijos algoritmus šių pradinių klaste-

rių parinkimui. Šiuo pasiūlymu pasinaudojant galima gauti pradinius klasterius iš klasifikacinio medžio ir tada pritaikyti iteratyvų operatorių toliau tobulinti susidariusią struktūrą. ITERATE algoritmas panaudoja šį požiūrį apjungdamas hierarchinį ir skaidymo kontrolės metodus siekdamas sugeneruoti maksimaliai skirtingus ir surištus klasterius. Šis algoritmas turi tris žingsnius:

1. gauti klasifikavimo medį, pasinaudojant kategorizavimo įrankiu kaip kriterijus funkcija pavyzdžių grupavimui.

2. išgauti "gerus" pradinius duomenų skirsnius iš klasifikavimo medžio. Panaudojami kaip starto taškai ieškant pageidaujamų klasterių.

3. Iteratyvus duomenų objektų perskirstymas tarp grupių siekiant gauti maksimaliai atskirtus klasterius.

Kadangi šis algoritmas pirmiausiai skirtas duomenų gavimui (data minning) [5], yra priimamos keturios svarbios sąlygos:

1. didelis kiekis atributų aprašančių duomenų objektus yra vardiniai;

2. nėra išankstinės tiriamų duomenų klasifikacijos;

3. visi klasifikuojami objektai yra žinomi prieš algoritmo taikymo pradžią;

4. duomenų bazė yra pakankamai didelė, kad skaičiavimo efektyvumas būtų kritinis faktorius.

1.2.1. Klasifikavimo medžio sukūrimas

Pasinaudojant hierarchiniu klasterizacijos metodu galima valdyti pradinių skirsnių kūrimą. Taikant požiūrį galima sukurti klasifikacijos medį, kaip patį pirmą žingsnį "gerų" pradinių skirstinių paieškoje. Algoritmas naudojamas klasifikacijos medžio kūrimui apibendrintai atrodo taip:

Inicializacija: Priskirti L = N_1
O_1 = klasterizuojamų objektų rinkinys
Loop kol L tuščias
    Paimti pirmą elementą iš L, sakykime N_k
    Jei O_k turi daugiau nei vieną objektą
        Rušiuoti objektus pasinaudojant ADO metodu
        Loop kiekvienam O_k objektui
          Jei pirmasis O_k objektas
                Sukurti nauja mazgą N_{k+c} kaip N_k vaiką
                Padėti objektą į mazgą N_{k+c}
          Arba
                (i) Padėti objektą į visus N_k mazgus-vaikus vieną po kito ir apskaičiuoti

skirstinių vertes kiekvienam

(ii) Padėti objektą kaip naują $N_k$mazgą vaiką ir apskaičiuoti skirstinio vertę.

Priskirti objektą mazgui, kurio skirstinio vertė yra didžiausia ir atnaujinti $A_i = V_{ij}$ mazgui.

End Loop

Sudėti naujus vaikus į L.

End Jei

End Loop

N_i vaizduoja mazgą klasifikacijos medyje, o O_i apibrėžia duomenų objektų aibę mazge N_i. N_1 yra medžio šaknis, O_1 visa aibė duomenų kurie turi būti klasterizuoti. Klasifikacijos medžio kūrimas naudoja paprastą skaidymo metodą skirstant duomenis į sub klases, pradedant nuo medžio šaknies. Algoritmas naudoja skirsnio rezultatą (iš kur gaunamas reiks aprašyti aukščiau) norėdamas nustatyti ar pasirinktą objektą priskirti į vieną iš esamų klasių ar kurti naują klasę. Medis auginamas pilnai suklasifikuojant to lygio duomenis prieš pereinant į žemesnį lygmenį.

1.2.2. Kategorijų nauda

Tyrinėdami kognityviai pageidautinus kategorizacijos lygius, Gluck ir Corter pritaikė tikimybės pritaikymo strategiją apibrėždami kategorijos naudingumą. Jie apibrėžė klasės C_k kategorijos naudingumą (CU), kaip:

$$CU_k = P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2 \tag{1}$$

Kur $P(A_i = V_{ij})$ yra tikimybė, kad savybė A_i bus lygi V_{ij} ir $P(A_i = V_{ij}|C_k)$ yra sąlyginė tikimybė, kad $A_i = V_{ij}$ priklausys klasei C_k. Tai vaizduoja savybių verčių, kurios gali būti teisingai nuspėtos, skaičiaus didėjimą klasei C_k $\left(P(A_i = V_{ij}|C_k)^2\right)$, per tam tikrą teisingų spėjimų skaičių, jei jokios informacijos apie klases nėra $[P(A_i = V_{ij})^2]$. Skirstinio rezultatas, t.y. skirstinio struktūros susidedančios ir K klasių naudingumas yra apibrėžiamas kaip CU vidurkis per K klases:

$$\frac{\sum_{k=1}^{K} CU_k}{K} \tag{2}$$

Gluck ir Cortel [15] pademonstravo kategorijos naudos efektyvumą nuspėjant pageidautina kategorizacijos lygį, esant jau egzistuojančiai klasifikacijos hierarchijai.

Gerai ištyrinėta godžių inkrementinių algoritmų savybė yra jų priklausomybė nuo duomenų sekos. Jie generuoja skirtingus klasifikacinius medžius pateikiant duomenis skirtinga eilės tvarka. Pavyzdžiu, CU funkcija yra kompromisas tarp tarp dydžio $P(C_k)$ ir darnos Co arba savybių verčių $\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2$ spėjimo tikslumo klasėje arba kategorijoje. $P(C_k$

sukelia šališkumą prie didelių kategorijų ir duomenų pateikimas su pasikartojančiomis vienodomis arba labai panašiomis grupėmis duomenų objektų gali iškreipti skirsnio struktūrą. Tai reiškia, kad sporadiškai gali atsirasti tarpiniai mazgai klasifikavimo medyje, kurie gali sukelti nereikalingą fragmentaciją galutiniuose skirstiniuose. Fragmentacija apsunkina naudingos informacijos išgavimą ir skirstinio struktūros.

Skirstinio rezultatų analizė ekstremaliai iškreiptų duomenų atveju atskleidžia šališkumo prigimtį. Įsivaizduokime situaciją, kurioje m+n atvejų suklasifikuota naujai atsiradusiame skirstinyje, sukuriant n+1 klasę, kurioje n klasių yra vienetinės klasės (glaudumas Co_{vien}) ir viena klasė turi m objektų (glaudumas Co_m). Ši situacija pavaizduoja ekstremaliai nevienodą klasių augimą. Tokioje situacijoje galima parodyti kad:

$$(m+n+2)\,CO_{ms} = Co_{vien} \tag{3}$$

Kaip tikėtasi didesniosios klasės dydis yra aiškesnis naujų objektų pridėjimui į tą klasę. Bet didesnioji klasė yra $(m+1)+(n+1)$kartų labiau vertinama. Intuityviai matas turi šališkumą apriboti klasių skaičių, netgi tada kai naujas objektas turi mažai panašių savybių su didesniąja klase.

Kuriant ITERATE buvo pritaikytas kitoks požiūris [5]. Nebuvo bandyta išvengti priklausomybės nuo eiliškumo efektų ir padidinti skirstinio rezultatus (vidutinę kategorijų reikšmę), šis fenomenas buvo panaudotas kaip galimybė apdoroti iškreiptus duomenų rinkinius ir iškreiptos eiles tvarkos duomenis. Į tai yra atsižvelgiama manipuliuojant duomenų eilės tvarką taip, kad glaudumas turėtų didesnę reikšmę ankstyvojoje skirstinių formavimo fazėje.

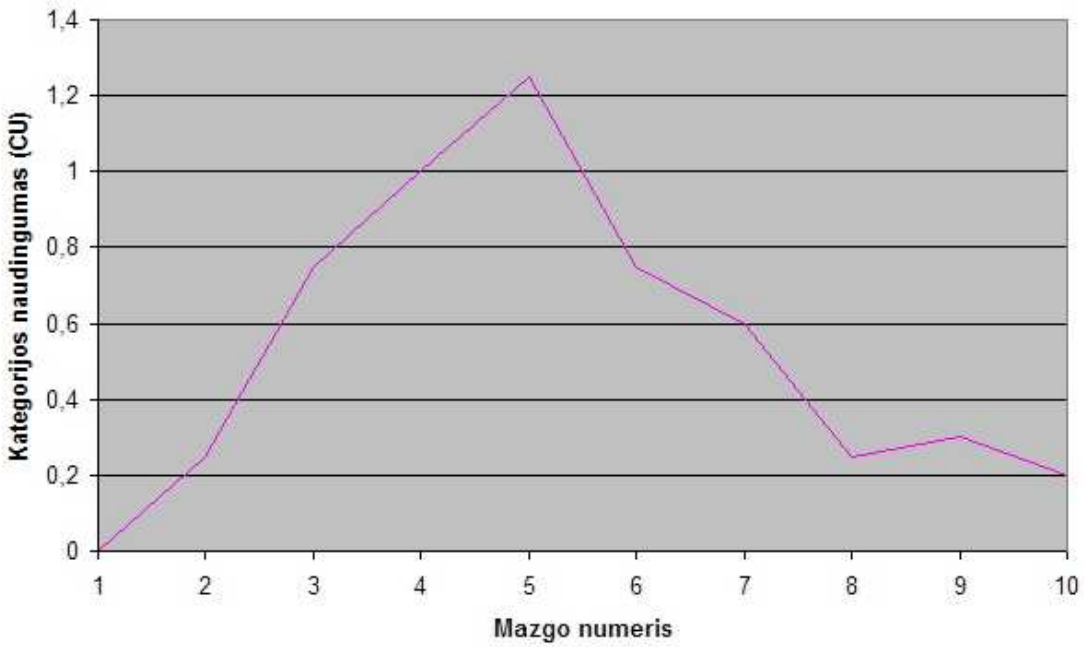
Norint panaudoti CU funkcijos šališkumą dydžiui, duomenų objektai yra surūšiuojami remiantis didžiausių skirtumų rūšiavimo (ADO) algoritmu. Objektas pasirenkamas būti sekančiu eilės tvarkoje, yra tas kuris labiausiai padidina Manhattan'o atstumą tarp jo ir ankščiau einančių n objektų. Manhattan'o atstumas tarp dviejų objektų yra apibrėžiamas kaip atributų-reikšmių skirtumų skaičius. Dydis n yra apibrėžiamas vartotojo, bet empyriškai atitinka tikėtiną klasių skaičių. Pirmasis objektas pasirenkamas toks, kuris labiausiai skiriasi nuo prototipinio duomens mazge.

1.2.3. Pradinių skirsnių išgavimas

Pradinė skirsnių struktūra yra išgaunama lyginant CU klasių arba mazgų reikšmę einant klasifikacijos medžiu. Einant bet kuriuo keliu (šaka) nuo medžio šakninės viršūnės iki lapų CU reikšmė didėja, o po to krenta. Klasės kurių reikšmės yra žemesnės nei maksimali CU reikšmė toje šakoje nepakankamai tiksliai apibrėžia duomenis ir todėl nėra naudingos klasterizacijos užduočiai.

Naudojant šituo kaip pagrindu, galima apibendrinti algoritmą:

Inicializacija: priskirti mazgą N = šaknis
Priskirti sąrašas = $\{N\}$



1 pav.: CU vertės klasifikaciniame medyje [5]

Loop kol sąrašas tuščias
Paimti vaikai(N)
Jei $CU_N > CU_c \forall c \in$vaikai($N$) N apibrėžia grupę pradiniame skirsnyje.
Arba
$\forall c$ toks, kad $c \in$vaikai(N) ir $CU_N \leq CU_c$
Pridėti c į sąrašą
$\forall c$ tokiems, kad $c \in$vaikai$N$ ir $CU_N > CU_c$
sukurti grupę iš c pradiniame skirstinyje
End Jei
Pašalinti N iš sąrašo
Priskirti N = pirmasis sąrašo elementas
End loop.

Pirmas algoritmo žingsnis garantuoja, kad mazgas, kuriame CU_k pasiekia savo maksimumą, bus pradiniame skirsnyje. Šis algoritmas yra pakankamai konservatyvus, todėl jis labiau mėgsta bendresnius klasterius pradinių skirstinių formavime. Einant bet kuriuo keliu nuo šaknies iki lapo, jei mazgo vaiko CU vertė didesnė nei mazgo, mazgas nepakliūva į pradinį skirstinį. Antras žingsnis. Jei CU vertė krenta kitiems mazgams vaikams šito mazgo, tai tie vaikai yra priskiriami pradiniam skirstiniui. Trečias žingsnis. Kai mazgas maršrute yra pasirenkamas į pradinį skirstinį, jokie kiti mazgai esantys tame kelyje žemiau nei jis nepakliūva į pradinį skirstinį. Todėl šis algoritmas pasirūpina, kad konceptai neapimtų vieni kitų. Šito konservatyvaus požiūrio pasėkoje sėklos paimtos iš medžio gilumos vaizduoja labiau specifinius koncertus ir jų atributų reikšmės yra būna labiau darnios. Jei šitie konceptai yra reikšmingi jie lieka galutiniame skirstinyje, priešingu atveju

jie gali būti įjungti į kitas grupes iteratyvaus perskirstymo metu.

1.2.4. Iteratyvus perskirstymas: skirstinių struktūros gerinimas

Iteratyvaus perskirstymo operatorius yra naudojamas norint padidinti atskirų klasių koherentiškumą skirstinyje. Perskirstymo operatorius priskiria objektą d klasei k, kurios atitikimas kategorijai CM_{dk} yra maksimumas. Objektas lieka savo klasėje jei yra kelios lygiavertės klasės ir klasė kurioje jis yra dabar yra tarp jų. Priešingu atveju ryšiai yra nutraukiami. Perskirstymo iteracija susideda iš kiekvieno objekto priskyrimo nustatymo ir pasiskirstymo atnaujinimo remiantis priskyrimais.

Skirstomieji metodai kaip ISODATA [13] pritaiko kvadratinio nuokrypio funkciją kaip kriterijų pasinaudojant Euklido atstumu skaičiuojant objektų ir klasterių centro atstumą. Duomenų objektai yra perskirstomi norint optimizuoti pasirinktą kriterijų funkciją, dažniausiai kvadratinių vidurkių paklaidą. Atitikimas tarp nominaliai įvertintų objektų d ir klasės k yra apibrėžtas tikimybiniu panašumo matu:

$$CM_{kd} = P(C_k) \sum_{i,j \in \{A_i\}_d} (P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2). \tag{4}$$

C_i yra aprašoma kaip sąlyginės tikimybės visų savybės reikšmių klasėje pasiskirstymas, $P(A_i = V_{ij}|C_k)$, $\forall$ i, j priklausantiems klasei C_k. Taip pat kategorijos atitikimo kategorijai matai daro prielaidą, kad objektas turi tik vieną reikšmę kiekvienam atributui (apibrėžtas $j \in \{A_i\}_d$). Atitikimas kategorijai matuoja tikėtino nuspėjamumo C_k padidėjimą klasės duomenų objekto d atributų reikšmėms.

1.2.5. Galutinių skirstinių vertinimas

Skirstinių gerumui aprašyti naudojami du tikimybiniai matai:

1. Maksimaliai glaudžiai surišti blokiniai (elementų panašumas blokinyje)
2. Pasiektas maksimalus atskirtumas (elementų skirtumai blokinyje)

Surištumas yra matuojamas padidėjusiu kiekvieno objekto savybės nuspėjimu, kai žinoma kokiai klasei jis yra priskirtas. Nuspėjamumo padidėjimas M_{dk} objektui d priskirtam klasei k yra apibrėžiamas taip:

$$\sum_{i,j \in \{A_i\}_d} \left(P\left(A_i = V_{ij}|C_k\right)^2 - P\left(A_i = V_{ij}\right)^2 \right) \tag{5}$$

Skirstinio surištumo struktūra yra matuojama kaip M_{dk} reikšmių suma visiems objektams rinkinyje. Tai galima interpretuoti kaip padidėjusį panašumą tarp duomenų objekto ir blokinio, kuriam jis yra priskirtas negu duomenų objekto panašumas duomenų rinkinio prototipui.

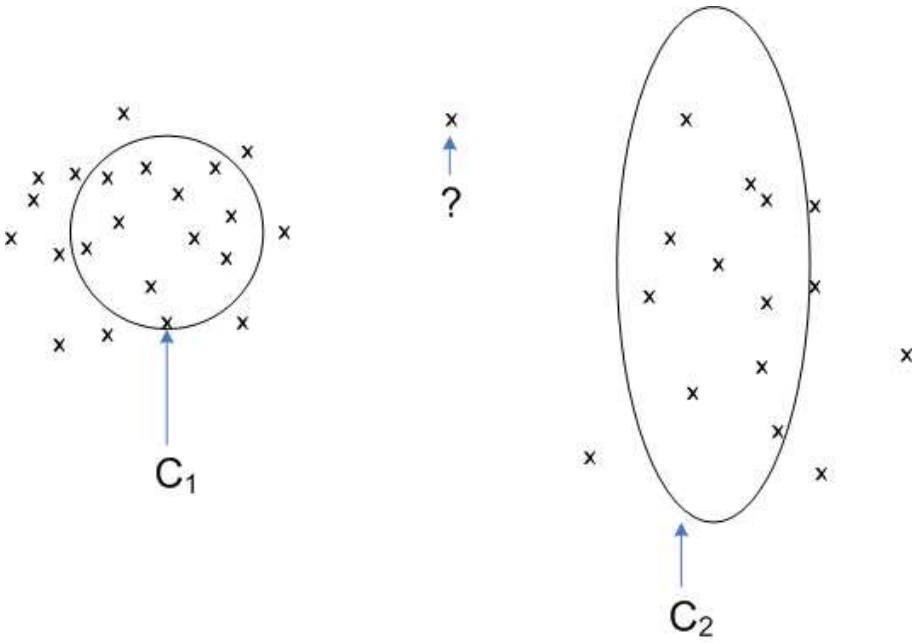
Blokinių atskirtumas galutiniame paskirstyme yra matuojamas kaip tarpklasinis skirtumas [5] naudojant tikimybinį panašumo matavimą pavadintą pasiskirstymo sutapimo laisvumu. Pasiskirstymo sutapimo laisvumas tarp k ir l duotajame skirstinyje yra matuojama kaip:

$$\frac{1}{n}\sum_{i}^{n}\sum_{j}\left(P\left(A_i = V_{ij}|C_k\right) - P\left(A_i = V_{ij}|C_l\right)\right)^2 \tag{6}$$

Ši vertė yra didesnį kuo dvi klasės yra skirtingesnės. Lyginant du skirstinius, reiktų pasirinktą tą particiją, kurios vidutinis laisvumas tarp klasių yra didesnis, nes klasės šiame skirstinyje atvaizduoja labiau išsiskiriančius konceptus.

1.3. Klasterizacija naudojant Renyi entropiją

Šis klasterizacijos algoritmas priskirdamas naujus narius blokiniui, kurio vidinė entropija pasikeičia mažiausiai [9, 21]. Norint įvertinti grupavimo kokybę yra įvedamas tarp-blokininė entropija, kuri remiasi Renyi entropija [8]. Šis metodas sugeba rasti bet kokios formos blokinius iš anksto nežinodamas jų kieko.



2 pav.: Duomenų priskirimas klasteriui [8]

Pažiūrėjus į situaciją pavaizduotą 2 paveikslėlyje. Erdvėje pasiskirstę objektai. Pradžioje poaibis šių objektų buvo priskirtas C_1 arba C_2 klasterius. Jie pažymėti kaip apskritimai. Klasterizacijos problema yra kuriam klasteriui priskirti pažymėtą objektą.

Šio metodo autoriai [8] siūlo objektą x priskirti kuriam nors klasteriui remiantis paprastu pastebėjimu. Jei x klaidingai priskiriamas C_1, darna arba entropija, padidės daugiau nei C_2, jei x bus priskirtas šiam klasteriui.

Todėl, turint pradinius klasterius C_k, kur k=1,…,K, x priskiriamas klasteriui C_i jei:

$$H\left(C_i + x\right) - H\left(C_i\right) < H\left(C_k + x\right) - H\left(C_k\right) \tag{7}$$

kai k=1,…,K, k≠i, kur $H\left(c_k\right)$ žymi klasterio c_k entropiją. Šis metodas autorių vadinamas skirtuminiu entropijos klasterizavimu. Šiuo metu kyla sekantys klausimai:

1. Kaip nuspėti entropiją tiesiai iš duomenų?
2. Kaip sukurti pradinius klasterius?
3. Kaip nustatyti kurį x klasterizuoti sekantį?

1.3.1. Entropija klasterio viduje

Ankščiau nustatyti entropiją tiesiogiai iš duomenų buvo sudėtingas procesas [21] nepriimant nerealių prielaidų apie duomenis. Neseniai buvo atrasta, kad entropijos matas pasiūlytas Renyi lengvai leidžia be parametrų nustatyti entropiją tiesiai iš duomenų.

Renyi entropija stochastiniam kintamajam X su tikimybinio tankio funkcija (ttf) [25] f_X yra apibrėžiamas kaip:

$$H_R\left(X\right) = \frac{1}{1-\alpha} log \int f_X^{\alpha} dx, \alpha > 1, \alpha \neq 1 \tag{8}$$

Specifiškai $\alpha = 2$ gauname lygtį

$$H_R\left(X\right) = -log \int f_X^2 dx, \tag{9}$$

vadinama Renyi kvadratine entropija.

Ši išraiška gali būti įvertinta tiesiogiai iš duomenų naudojant Parzen lango tenkino įvertinimą su daugiadimensiniu Gauso lango funkcija. Tarkime, kad klasteris C_k susideda iš rinkinio diskrečių taškų x_i, $i = 1 \ldots, N_k$. tada ttf spėjimas remiantis duomenų taškais esančiais $C_k$yra parašomas taip:

$$\widehat{f}_X = \frac{1}{N_k} \sum_{i=1}^{N_k} G\left(x - x_i, \sigma^2 I\right) \tag{10}$$

kur N_k yra taškų skaičius C_k ir autoriai naudojo simetrinį Gauso branduolį su kovariacine matrica [26] $\Sigma = \sigma^2 I$. Įstačius 10 i 9 ir pasinaudojant Gauso branduoliu, gaunamas C_k entropijos artinys

$$H\left(C_k\right) = -logV\left(C_k\right), \tag{11}$$

kur

$$V\left(C_k\right) = \frac{1}{N_k^2} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} G\left(X_i - x_j, 2\sigma^2 I\right). \tag{12}$$

Kadangi entropija yra skaičiuojama remiantis taškais priskirtais tam pačiam klasteriui, todėl 12 formulė vadinama entropijos klasterio viduje [8].

visi $G\left(X_i - x_j, 2\sigma^2 I\right), i, j = 1, \ldots, N$, kur N yra objektų skaičius rinkinyje, yra apskaičiuojami ir išsaugomi N^2 simetrinėje artumo matricoje. Tai reiškia kad apskaičiavus artumų matricą visi kiti skaičiavimai yra tik manipuliacijos matricomis. Tai gali šiek tiek riboti klasterizuojamų objektų skaičių. Kita problema, kurią nurodo autoriai, yra σ, norint gauti geriausiu klasterizacijos rezultatus reikia pasirinkti σ tokią, kad Parzen ttf artinys yra pakankamai tikslus. σ pasirinkimas todėl yra viena iš bendresnių problemų neparametrinėms ttf. Duomenys naudoti eksperimentams buvo normalizuoti kiekvienam atributui atskirai taip, kad pakliūtų į diapazoną [-1,1] ir jų vidurkis būtų lygus 0.

1.3.2. Klasterių inicializacija

Pradžioje yra sukuriami K_{init} klasteriai duomenų rinkinyje. Tai yra atliekama pirmiausia atsitiktiniu būdu pasirenkant K_{init} objektų. Po to taškas esantis arčiausiai bet kurio klasterio nario yra įtraukiamas į klasterį, kol priskiriama pradžioje numatytas kiekis kiekvienam klasteriui. Grupavimas atliekamas taip, o ne pasirenkant N_{init} objektų arčiausiai pradinio taško, kadangi tai daro duomenų grupavimą jautresnį duomenų struktūrai.

1.3.3. Kito objekto klasterizavimui pasirinkimas

Kitas objektas klasterizavimui gali būti pasirinktas keliais būdais.

- Atsitiktinai - šio būdo trūkumas yra tas, kad klasterizacijos pradžioje atsitiktinai paimtas taškas esantis toli nuo pradinių klasterių gali sukelti klasterizacijos nestabilumą vėlesnėse klasterizacijos fazėse.
- renkantis objektą arčiausiai klasterio prototipo - šiuo būdu sudaryti klasteriai yra labiau stabilesni.

Kadangi mes nežinome [8] tikrojo klasterių skaičiaus iš pradžių, o jei ir žinotumėme negalima būti tikriems, kad pradiniai taškai bus tikruosiuose klasteriuose.

Kaip galima šios problemos sprendimą [8] autoriai siūlo sukurti pradžioje daug pradinių klasterių [24], sužymėti visus taškus, ir perskirstyti taškus priklausiančius "blogiausiam klasteriui". Taškų priklausančių "blogiausiems klasteriams" peržymėjimas yra atliekamas pasinaudojant skirtumine klasterizacijos entropija. Ši procedūra yra kartojama ir kiekviename žingsnyje yra pašalinamas vienas klasteris.

1.3.4. Entropijos tarp klasterių vertinimas

Vietoje to kad skaičiuoti entropija kiekvienam klasteriui pasinaudojant 12 formule, [8, 10] siūlo naudoti dvigubą sumą, kuri skaičiuotų visų taškų sumą ir turėtų savyje nario f-ja, $M\left(x_{ij}\right)$, kurios reikšmė lygi vienam jei x_i ir x_j priklauso skirtingiems klasteriams arba 0 kai priklauso tam pačiam. Gauta lygtis buvo pavadinta tarpklasterine entropija

$$H\left(C_1,\ldots,C_K\right) = -logV\left(C_1,\ldots,C_K\right), \tag{13}$$

Kur

$$V\left(C_1,\ldots,C_k\right) = \frac{1}{2\prod_{k=1}^{K} N_k} \sum_{i=1}^{N}\sum_{j=1}^{N} M\left(x_{ij}\right) G\left(x_i - x_j, 2\sigma^2 I\right) \tag{14}$$

klasteriams $C_k, k = 1,\ldots,K$

Jei klasteriai gerai atskirti $V\left(C_1,\ldots.C_K\right)$ bus maža vertė, to pasėkoje $H\left(C_1,\ldots,C_K\right)$ bus didelė vertė. Remiantis tuo galima atskirti blogus klasterius.

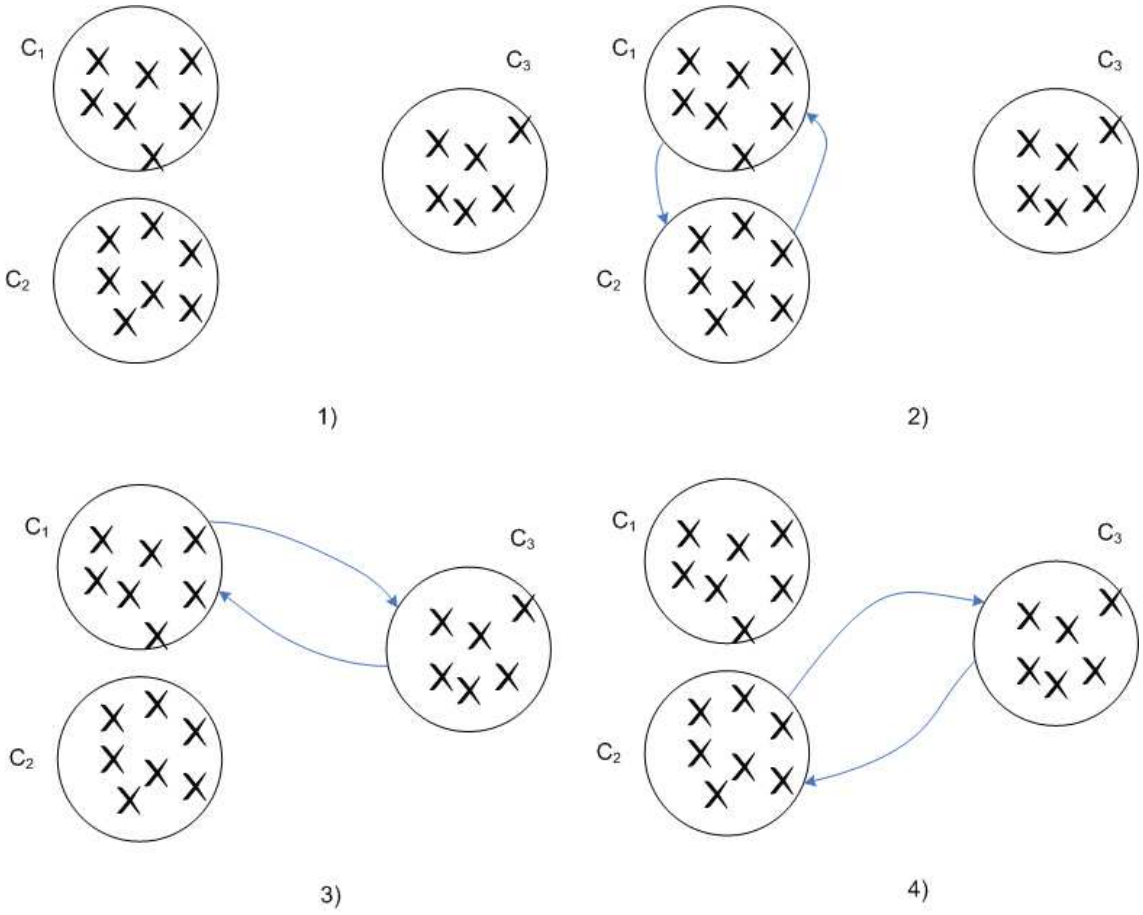
1.3.5. "Blogiausio klasterio" suradimas

Paveiksle 3 pavaizduota kaip tarpklasterine entropija gali būti panaudota nustatyti "blogiausią klasterį". Pirmojoje paveikslėlio dalyje pavaizduota duomenų suskirstymas į tris klasterius. Skaičiuojant entropiją tarp dviejų klasterių, visų kitų klasterių narių f-jos yra prilyginamos nuliui. Pavyzdžiui antrojoje piešinėlio dalyje klasteris C_3 yra pašalinamas ir $H\left(C_1, C_2\right)$ apskaičiuojamas. Taip pat atliekama ir trečioje ir ketvirtose piešinuko dalyse.

Palyginus visus klasterius poromis yra išrenkamas "blogiausias klasteris", kuris bus šalinamas remiantis didžiausia tarpklasterine entropija, tai reiškia, kad likę klasteriai yra labiausiai atskirti klasterai. Aukščiau pavaizduotame paveiksle "blogiausiu klasteriu" išrenkamas arba C_1 arba C_2 klasteris.

1.3.6. Galutinių klasterių aibės pasirinkimas

Prieš kiekvieną klasterių kiekio mažinimą vienetu ir jo narių perskirstymą, visi nariai senuosiuose klasteriuse yra įsimenami [10]. Taigi kiekviename žingsnyje egzistuoja klasterių ai-



3 pav.: "Blogiausio klasterio" nustatymas naudojantis tarpklasterine entropija. Šiame piešinyje tai būtų C_1 arba C_2 yra "blogiausias klasteris" [8]

bė. Kadangi klasterių narių sąrašai yra saugomi kiekviename žingsnyje mes gauname priskyrimų klasteriams hierarchiją. Tai yra kartojama kol lieka tik du klasteriai. Baigus klasterizaciją kyla klausimas, kuriame hierarchijos lygyje yra mūsų galutiniai klasteriai.

Autoriai pastebėjo [8], kad skaičiuojant tarpkasterinę entropiją kiekviename žingsnyje galima pamatyti, kaip šios reikšmės drastiškai padidėja sumažinus klasterių skaičių iki mažesnio nei yra klasterių.

Taigi stebint tarpklasterinės entropijos pokytį prieš ir po perklasterizavimo galima pasirinkti galutinių klasterių aibę.

Kadangi pradiniai klasteriai yra kuriami atsitiktiniu būdu, klasterizavimo rezultatai gali skirtis atliekant tų pačių duomenų klasterizaciją. Tai labai dažnai pasitaiko su duomenimis, kuriuose yra labai netolygiai pasiskirstę klasteriai. Tarpklasterinė etropija paskaičiuota skirtingiems galutiniams klasteriams leidžia pasirinkti geriausią klasterizacijos inicializacijos rinkinį.

1.4. CLIP3

CLIP3 algoritmas yra hibridinis algoritmas [4]. Jis kombinuodamas sprendimų medžių ir taisyklių algoritmus išsprendžia per didelio prisitaikymo problemą esant triukšmingiems duome-

nims. Triukšmingais duomenimis yra vadinami tokie duomenų rinkiniai, kuriuose yra netikslumų. Algoritmai galintis sugeneruoti geras taisykles esant triukšmams duomenyse vadinami triukšmams atspariais algoritmais. CLIP3 dalina duomenis po panašių duomenų poaibius. Triukšmų apkarpymai atliekami CLIP3 algoritme triukšmo ribos pagalba. Po to kai CLIP3 suskaido duomenis į skirsnius neturinčius triukšmo yra generuojamos taisyklės kiekvienam poaibiui.

CLIP3 pradeda savo veikimai skaidydamas duomenis kaip sprendimų medžio algoritmas. Bet priešingai nei kiti algoritmai jis neskaičiuoja jokių duomenų skaidymo gerumo matų, tokių kaip entropija. Jis pasiima savybes ir generuoja taisykles spręsdamas sveikojo programavimo uždavinį (IP). CLIP3 naudoja mokomuosius duomenis kurdama IP uždavinį ir tada panaudoja paprastą IP programą jam išspęsti. CLIP3 didžiausias skirtumas nuo sprendimo medžių algoritmų yra tas, kad jis neskaido duomenų "geriausiu" būdu, o skaido juos keliais būdais. Priedo nėra poreikio saugoti viso CLIP3 sprendimo medžio. Jis saugo tik medžio lapus (medis neegzistuoja). To pasekmė yra paprastesnės taisyklės, mažesnis taisyklių skaičius ir dideli atminties naudojimo sutaupymai. Kitas CLIP3 pranašumas yra jo greitis, duomenų skaidymas IP modelio pagalba yra daug greitesnis palyginus su entropijų skaičiavimu. IP modelio sprendiniai parodo svarbiausius bruožus, kurie bus naudojami taisyklių kūrimui.

CLIP3 algoritmo supratimo pagrindas yra dvejetainių matricų kūrimas ir skaidymas į poaibius naudojantis IP uždavinio sprendiniais [16]. Ši operacija yra atliekama daug kartų algoritmo eigoje. Po kiekvieno teigiamų pavyzdžių skaidumo į poaibius, pertekliniai poaibiais (saugomi kaip matricos) yra pašalinami. Perteklinis poaibis yra toks poaibis kuris yra kito poaibio dalis arba identiškas kitam poaibiui. Ši operacija pagreitina taisyklių generavimo procesą ir formą kuria yra pateikiamos galutinės taisyklės. IP uždavinys taip pat sumažina galutinių poaibių skaičių iš kurių yra generuojamos galutinės taisyklės. Toliau pateikiamas algoritmas pseudo kodu [4, 16].

Duota: TEIgiami ir NEIgiami mokomieji duomenys, ribos duomenys.

1 pakopa

Visiems NEIgiamiems pavyzdžiams:

1. Sugeneruoti dvejetainę matricą palyginant TEIgimą matricą su neg_i, kur $i = , \ldots, N$ ir N yra NEIgiamų pavyzdžių skaičius.
2. Išspręsti atitinkamą IP uždavinį
3. Suskaidyti TEIgiamų pavyzdžių matricą remiantis savybėmis parodytomis IP uždavinio sprendiniais ir pašalinti perteklines matricas

2 pakopa

1. Sugeneruoti ŠM (šabloninę matricą) dvejetainę matricą
2. Išspręsti ŠM dvejetainę matricą pasinaudojant IP

3. Atstatyti matricas pasirinktas 2 žingsnyje

4. Paversti kiekvieną gautą matricą į IP modelį ir išspręsti jį

5. Sugeneruoti taisykles remiantis IP rezultatais

3 pakopa

1. Surasti geriausią taisyklę

2. Pašalinti teigiamus pavyzdžius, kuriuos apima geriausia taisyklė ir pereiti prie 1 pakopos

Rezultatas: Taisyklių rinkinys apimantis visus teigiamus pavyzdžius ir nevieno neigiamo.

CLIP3 turi tris ribas. Šių ribų paskirtis yra kontroliuoti kuriamų taisyklių sudėtingumą ir nuspręsti, kada sustoti generuoti taisykles. IP uždavinio sprendinys sudarytas iš vienetų ir nulių, kuo daugiau vienetų eilutėje tuo sudėtingesnis sprendinys.

Triukšmo riba (NT): Ji apsprendžia kurios šakos (galinčios turėti triukšmingų teigiamų pavyzdžių) remiantis IP sprendiniu pirmoje pakopoje bus pašalintos. Jei NT=0% jokios šakos nebus šalinamos.

Geriausios taisyklės riba (BRT): Taisyklės sugeneruotos antroje pakopoje taps geriausiomis taisyklėmis tik tada jei jos apima nemažiau nei BRT procentų teigiamų arba likusių teigiamų pavyzdžių.

Sustojimo riba (ST): Ji sustabdo algoritmo veikimą jei lieka mažiau nei ST procentų taisyklėmis neaprašytų teigiamų pavyzdžių.

Tiek ST, tiek BRT gali sustabdyti algoritmo veikimą.

1.5. Diskretizacija

Diskretizacija yra procesas, kurio metu nuoseklūs duomenys yra suskaidomi į baigtinį intervalų skaičių, kurių kiekvienas apima visas reikšmes pakliūvančias į jį [4]. Discretizacija gali būti reikalingas pradinis apdorojimo žingsnis prieš klasterizacijos algoritmo vykdymą, nes daugeliui jų reikia kad duomenys būtų pateikti diskrečia forma, nominalūs (pvz. baltas, geltonas, juodas) arba skaitmeniniai (pvz. 1, 2, 3,4).

Daugelis autorių pabrėžia, kad nuo diskretizacijos rezultatų kokybės labai priklauso ir vėliau vykdomi klasterizacijos algoritmų rezultatų kokybė, nors iš dalies ta kokybė priklauso ir nuo algoritmo savybių. Yra du požiūriai į duomenų diskretizacijos atlikimą. Vienas yra integruoti diskretizacijos algoritmą į patį klasterizacijos algoritmą. Kitas požiūris, kuris yra daug labiau paplitęs, yra atlikti duomenų diskretizaciją kaip atskirą procesą dar prieš patį klasterizacijos algoritmą. Kadangi ankščiau aptarti algoritmai neturi savyje integruotų duomenų diskretizavimo savybių, mes dėmesį skirsime antrajam požiūriui.

Remiantis kriterijais diskretizacijos algoritmus galima suskirstyti į:

- Prižiūrimus ir neprižiūrimus. Pagrindinis skirtumas tarp jų yra sprendimo atributo naudojimas. Neprižiūrimi algoritmai nenaudoja jo.
- Globalūs ir lokalūs. Globalūs algoritmai skirsto kiekvieną atributą į intervalus neatsižvelgdamas į kitus atributus. Lokalūs algoritmai generuoja intervalus iš skirsnių lokalizuotose objektų erdvės vietose. Tokio algoritmo pavyzdys gali būti diskretizacija sprendimo medžio pagalba.
- Statiniai ir dinaminiai. Statiniai algoritmai diskretizuoja kiekvieną objekto savybę viena iteracija norint sužinoti intervalų skaičių, neatsižvelgdami į kitas savybes. Dinaminiai algoritmai ieško visų galimų intervalų visoms savybėms tuo pat metu.

Toliau pateikti algoritmai remiantis klasifikacija pateikta aukščiau yra neprižiūrimi, globalūs ir statiniai.

Diskretizacijos procesas susideda iš dviejų dalių:

1. Diskrečių intervalų skaičiaus pasirinkimas, kuris dažniausia atliekamas vartotojo.
2. Pasirinktų intervalo pločio pasirinkimas, kuris dažniausia yra atliekamas diskretizacijos algoritmo.

Intervalų skaičius ir tuo pačiu ir įverčių skaičius, kuriuos gali turėti savybė, turi didelį efektą klasterizacijos algoritmų tikslumui ir veikimo spartai. Todėl bet koks diskretizacijos algoritmas, naudojamas kaip pradinio apdorojimo fazė klasterizacijos algoritmui, turėtų generuoti kaip galima mažesnį kiekį diskrečių intervalų. Tai yra pakankamai sudėtingas reikalavimas, nes kuo daugiau intervalų yra, tuo daugiau pradinės informacijos yra išsaugoma. Todėl reikia pasirnikti kompromisą ir naudoti kokį nors euristinį metodą padėti vartotojui pasirinkti intervalų skaičių. Vienas iš tokių metodų yra, kad intervalų skaičius turi būti didesnis nei norima aprašyti klasių, arba klasifikuoti. Kitas metodas yra apskaičiuoti intervalų skaičių n_{Fi}, kiekvienam atributui $F_i\,(i = 1, \ldots,)$ pasinaudojant formule:

$$n_{Fi} = \frac{M}{3C}, \tag{15}$$

kur M yra bendras diskretizuojamų narių skaičius, C - klasių skaičius.

Toliau patiekiami keli diskretizavimo algoritmų pavyzdžiai.

1.5.1. Vienodo pločio diskretizacija

Vienodo pločio diskretizacijos algoritmas pirmiausiai randa mažiausią ir didžiausias reikšmes kiekvienai pateiktai savybei F_i ir padalina šitą ištisinį atributų intervalą į vartotojo nurodytą, n_{Fi}, skaičių vienodo pločio diskrečių intervalų. Daugelio klasterizacijos algoritmų intervalų

skaičius stipriai įtakoja jų naudingumą, nes kuo didesnis intervalų skaičiuos tuo didesnė tikimybė, kad bus per daug prisiderinama. kitu atveju, kai yra per mažai intervalų, informacijos tikslumas mažėja ir tai gali paslėpti ryšį tarp klasės kintamojo ir intervalo kintamojo. Tai gali labai stipriai pasireikšti, kai duomenys yra netolygiai pasiskirstę, tokiu atveju didžiuliai informacijos kiekiai gali būti prarasti.

1.5.2. Vienodo dažnio diskretizacija

Vienodo dažnio diskretizacijos algoritmas pirmiausia surūšiuoja visus kiekvienos savybės duomenis didėjimo tvarka ir padalinus juos į vartotojo nurodytą intervalų, n_{Fi}, skaičių. Tokiu būdu gaunami intervalai kuriuose yra po vienodą skaičių surūšiuotų savybės įverčių.

1.5.3. Vieno lygio sprendimo medžio diskretizacija

Vieno lygio sprendimo medžio algoritmas vadinamas vienos taisykles diskretizatoriumi [17]. Jis godžiai dalina visą reikšmių aibę į intervalus, naudojant apribojimą, kad kiekvienas intervalas turi turėti nemažiau verčių nei vartotojo nurodytas skaičius. Jis pirmiausia sukuria pradinius skirstinius, kuriuose yra minimalus reikšmių skaičius. Po to kiekvieno intervalo ribos yra praplečiamos pridedant naujas reikšmes, taip kad kiekvienas intervalas turi didelę daugumą objektų iš vienos sprendimo klasės. Žemiau pateikiamas algoritmo pseudo kodas:

1. mokomajame duomenų rinkinyje apskaičiuoti kiek objektų klasėje C turi atributo A įverčius V: išsaugoti šią informaciją trimačiam masyve, $COUNT[C, V, A]$.

2. Bazinė klasė yra tokia klasė, kuriai priklauso daugiausiai objektų mokomajame rinkinyje. Bazinės klasės tikslumas yra gaunamas padalinus objektų skaičių esančių šioje klasėje iš bendro objektų skaičiaus visame rinkinyje.

3. Kiekvienam skaitmeniniam atributui A, sukuriama nominali jo versija apibrėžiant baigtinį įverčių intervalų skaičių. Šie intervalai tampa nominalaus A vertėmis. Pavyzdžiui, jei A skaitmeninės vertės yra padalinamos tris intervalus, tada nominali A versija turės įverčius "$Intevalas1$", "$Intevalas2$", "$Intevalas3$". $COUNT[C, V, A]$ ataspindi šią transformaciją: $COUNT[C, "IntervalasI", A]$ bus lygus sumai visų $COUNT[C, V, A]$, kurių V pakliūvą į intervalą I.

   Apibrėžimai:

   Klasė C yra optimali atributo A įverčiui V jei $COUNT[C, V, A]$ yra maksimalus.

   Klasė C yra optimali atributo A intervalui $IntervalasI$ jei $COUNT[C, "IntervalasI", A]$ yra maksimalus.

Įverčiai yra skirstomi į intervalus taip, kad kiekvienas intervalas patenkintų šiuos apribojimus:

(a) Yra mažiausiai viena klasė, kuri yra optimali daugiau nei vartotojo apibrėžtam skaičiui [18] įverčių tame intervale. Šis ribojimas negalioja dešiniausiam intervalui.

(b) Jei $V[I]$ yra mažiausia atributo A vertė mokymosi duomenyse, kurie yra didesni nei vertės intervale I tada nėra klasės C, kuri būtų optimali $V[I]$ ir intervalui I

4. Kiekvienam atributui A, naudoti nominalią versiją skaitmeniniams atributams:

   (a) Sukurti hipotezę turinčią savyje atributą A pasirenkant optimalią klasę kiekvienam A įverčiui V. Jei kelios klasės yra optimalios įverčiui pasirinkti vieną iš jų atsitiktinai.

   (b) pridėti sukurtą hipotezę prie aibės HIPOTEZĖS. Šis aibė turės savyje po viena hipotezę kiekvienam atributui.

5. Pasirinkti taisyklę, kurios tikslumas didžiausias mokomojoje aibėje, iš HIPOTEZIŲ aibės. Jei yra kelios vienodai tikslios taisyklės vieną iš jų reikia pasirinkti atsitiktiniu būdu.

Iš aukščiau aprašyto algoritmo matosi, kad norint atlikti vieno lygio sprendimo medžio diskretizaciją reikia prieš jį atliekant žinoti kokiai klasei priklauso vienas ar kitas įrašas. O atsižvelgiant į mūsų turimus duomenis, mes tokios informacijos neturime. Norint sėkmingai atlikti šį algoritmą reikia pirmiausia apdoroti duomenis vienu iš kitų čia aprašytų diskretizavimo algoritmų. Tai būtų neefektyvu laiko ir resursų požiūriu. Todėl šis algoritmas toliau nebus tyrinėjamas.

1.5.4. K-vidurkių klasterizacijos diskretizavimas

Kaip buvo aprašyta ankščiau k-vidurkių metodas remiasi našumo indekso, apibrėžiamo kaip visų vektorių kvadratinis atstumas iki klasterio centro, minimizavimu. Jis gali būti perapibrėžtas specialiai vienos savybės intervalų radimui [4].

Duota: Duomenų rinkinys susidedantis iš M objektų ir C klasių, vartotojo nurodytas intervalų skaičius n_{Fi} savybei F_i.

1. Kiekvienai klasei $c_j, j = 1, \ldots, C$ daryti

2. Pasirinkti $K = n_{Fi}$ kaip pradinius klasterių centrus. Pradžioje galima paimti pirmas K savybės reikšmes, kaip klasterių centrus.

3. Paskirstyti likusias savybės reikšmes tarp K klasterių centrų, kaip kriterijų naudojant minimalų atstumą. To pasėkoje savybės reikšmės susirenka apie atnaujintus K klasterių centrus.

4. Apskaičiuoti K naujų klasterių centrų, taip kad kvadratinio atstumo suma tarp naujo centro ir visų klasterio objektų būtų minimali.

5. Patikrinti ar atnaujinti K klasterių centrai sutampa su senaisiais centrais, jei taip eiti į pirmą žingsnį, jei ne pereiti prie trečio žingsnio.

Rezultatai: Galutinės savybių ribos susidės iš minimalios savybės reikšmės, vidutinės reikšmės tarp šalia esančių klasterių prototipų visoms klasėms ir didžiausios savybės reikšmės.

Jei informacijos apie klasę nėra tada galima vykdyti aukščiau pateiktą algoritmą naudojant $j = 1$ kiekvienai savybei ir gaunamas intervalų skaičius bus lygus n_{Fi}. K-vidurkių metodas yra labai veikiamas klasterių kieko pasirinkimu, pradinių klasterių centrų pasirinkimo, duomenų pateikimo eilės tvarkos, geometrinių duomenų savybių. Prieš nurodant intervalų skaičių galima pabraižyti dažnių lentelę, nes neteisingai nurodžius intervalų skaičių gaunamas neatitinkantis realybės suskirstymas. Šitai problemai išspręsti galima atlikti suskirstymą kelis kartus naudojant skirtingus intervalų skaičius ir atliekant gerumo skaičiavimus, vienas jų galėtų būti entropijos apskaičiavimas kiekvienam variantui.

Tyrimui atlikti pasirinktas lyginamosios analizės metodas.

Lyginamosios analizės paskirtis yra palyginti kiekvieno diskretizavimo bei klasterizacijos algoritmų darbo efektyvumą tam tikrų kriterijų atžvilgiu ir parodyti, kuris iš algoritmų atlieka savo darbą geriau nei kiti mus dominančių kriterijų atžvilgiu. Šis palyginimas remiasi literatūros apžvalgoje aprašytų algoritmų detalia analize, kurioje kiekvienas algoritmas yra analizuojamas atskirai neatsižvelgiant į kitus algoritmus. Šio naudojamo metodo pagrindinė paskirtis yra nustatyti pranašumus ir trūkumus kiekvieno algoritmo tam tikrose srityse taip, kad būtų nustatyti pagrindiniai algoritmo naudojimo kompromisai.

Norint išskirti kurį nors algoritmą kaip geriausią reikia apibrėžti kriterijus, kuriais remiantis bus vertinami algoritmai. Algoritmai bus vertinami žemiau pateiktais kriterijais:

- Darbo sparta;
- Sunaudojami procesoriaus resursai;
- Sunaudojami atminties resursai;
- Algoritmo laiko sąnaudų pokytis pakitus objektų skaičiui;
- Algoritmo elgesys atsiradus naujam objektui;
- Rezultatų pastovumas;
- Atnaujinimo dažnis;

Norint išvengti atsitiktinių klaidų visi eksperimentai bus atliekami $N = 5$ kartų.

Kriterijų pagrindimas:

Darbo sparta apibūdina laiko tarpu sekundėmis, per kurį algoritmas atlieka savo užduotį su tam tikru duomenų rinkiniu. Duomenų rinkinys visais atvejais vienodas ir yra poaibis visų turimų duomenų. Darbo spartos tikslumui padidinti eksperimentas kartojamas N kartus ir galutinė sparta yra apskaičiuojamai remiantis formule:

$$ds = \frac{\sum_{n=1}^{N} el_n}{N}, \tag{16}$$

kur ds - darbo sparta, o el_n - laiko tarpas per kurį atliktas n-asis eksperimentas.

Sunaudoti procesoriaus resursai parodo kiek yra apkraunamas vidutiniškai procesorius apdorojant duomenų rinkinį ir yra matuojamas procentais. Sunaudotų procesoriaus resursų tikslumui padidinti eksperimentas kartojamas N kartus ir galutiniai sunaudoti procesoriaus resursai yra apskaičiuojamai remiantis formule:

$$pr = \frac{\sum_{k=1}^{N} epr_k}{N}, \tag{17}$$

kur pr - sunaudoti procesoriaus resursai, o epr_k - vidutinis procesoriaus apkrovimas atliekant k-ąjį eksperimentą.

Sunaudoti atminties resursai parodo kiek kompiuterio darbinės atminties algoritmas sunaudoja apdorodamas duomenų rinkinį ir yra matuojamas baitais. Sunaudotų atminties resursų tikslumui padidinti eksperimentas kartojamas N kartus ir galutinis resursų sunaudojimas apskaičiuojamai remiantis formule:

$$ar = \frac{\sum_{l=1}^{N} ear_l}{N}, \tag{18}$$

kur ar - sunaudoti atminties kiekis megabaitais, o ear_l - sunaudojama atmintis atliekant l-ąjį eksperimentą.

Algoritmo laiko sąnaudų pokytis pakitus objektų skaičiui parodo kiek kartų pakintą laiko tarpas, kol algoritmas atlieka veiksmus su duomenų rinkiniu, kai duomenų rinkinys padidėja 2 kartus. Laiko sąnaudų pokyčio tikslumui padidinti eksperimentas kartojamas N kartus ir galutinis resursų sunaudojimas apskaičiuojamai remiantis formule:

$$lp2d = \frac{\sum_{m=1}^{N} elp2d_m}{N \cdot ds}, \tag{19}$$

kur $lp2d$ - laiko sąnaudų pokytis, ds - darbo sparta apskaičiuota remiantis 16 formule, $elp2d_m$ - laiko tarpas per kurį atliktas m-asis eksperimentas.

Algoritmo elgesys atsiradus naujam objektui parodo kiek kartų pakintą laiko tarpas, kol algoritmas atlieka veiksmus su duomenų rinkiniu, kai duomenų rinkinys padidėja 1 objektu. Laiko sąnaudų pokyčio tikslumui padidinti eksperimentas kartojamas N kartus ir galutinis resursų sunaudojimas apskaičiuojamai remiantis formule:

$$lp1o = \frac{\sum_{i=1}^{N} elp1o_i}{N \cdot ds}, \tag{20}$$

kur $lp1o$ - laiko sąnaudų pokytis, ds - darbo sparta apskaičiuota remiantis 16 formule, $elp1o_m$ - laiko tarpas per kurį atliktas m-asis eksperimentas (duomenų rinkinys padidėjo 1 objektu).

Rezultatų pastovumas parodo kokia dalis objektų iš pradinio rinkinio lieka tose pačiose klasėse po kiekvieno eksperimento. Pastovumui apskaičiuoti atliekamas kontrolinis klasifikavimas ir $N - 1$ testinių klasifikavimų. Galutinis rezultatų pastovumas paskaičiuojamas remiantis formule:

$$rp = \frac{\sum_{j=1}^{N-1} (os - erp_j)}{(N-1) \cdot os}, \tag{21}$$

kur rp - rezultatų pastovumas, erp_j - objektų skaičius, kurie pakliūva į kitą klasę nei kontroliniame klasifikavime, atlikus j-ąjį eksperimentą, os - pradinėje imtyje esančių objektų kiekis.

Atnaujinimo dažnis parodo santykį tarp atnaujinimų skaičiaus ir įdėtų naujų elementų, kad algoritmo rezultatas vykdant jį po kiekvieno elemento įdėjimo palyginus po atitinkamo elementų skaičiaus įdėjimo vienu kartu skirtųsi nedaugiau nei $rp = 95\%$. Atnaujinimo dažnio algoritmas:

Įsiminti pradinę aibę A;
$A_1 := A$;
Atlikti testuojamąjį algoritmą;
count:=0;
$A^* := \oslash$;
$cont := TRUE$;
$rp = \oslash$;
KARTOTI kol $cont = TRUE$
    Pridėti prie aidės A_1 vieną elementą A';
    $count := count + 1$;
    $A^* := A^* \cup A'$;
    Atlikti testuojamą algoritmą;
    Jei $count\ mod\ 5 = 0$
        Tada
        $A_2 := A \cup A^*$;
        Atliekame testuojamą algoritmą;
        Apskaičiuojame rp';
        $rp = rp \cup rp'$;
        Jei $rp' < 95\%$
            Tada
            Grąžinti $ad := \frac{1}{count-5}$;
            $cont := FALSE$;
        Baigti;
    Baigti;
Baigti

Diskretizacijos algoritmams palyginti gali taip pat būti naudojami visi kriterijai, bet rezultatų pastovumo kriterijus prasmingas tik tiems algoritmams, kuriuose naudojamas atsitiktinumas. Visi aukščiau išvardinti kriterijai tinka klasterizacijos algoritmų įvertinimui. Rezultatų pastovumo kriterijus yra labai svarbus kriterijus algoritmams, kurie prasideda atsitiktinių taškų pasirinkimu ir yra jautrūs lokaliems minimumams.

Idealus algoritmas turėtų atitikti tokį kriterijų įverčių rinkinį:

- $ds < 300s$;
- $pr < 20\%$;
- $ar < 1Mb$;
- $lp2d < 2,5$;

- $lp1o < 1,1$;
- $rp > 90\%$;
- $ad < \frac{1}{50}$;

Preliminariai galime numanyti, jog nevisi skyriuje 1.5 aprašyti diskretizacijos algoritmai tenkins aukščiau pateiktą sąlygų rinkinį. Todėl remiantis gautais rezultatais parinksime tą algoritmą, kurio savybių rinkinys labiausiai atitiks aprašytam rinkiniui.

Nuo diskretizacijos rezultatų kokybės labai priklauso ir vėliau vykdomi klasterizacijos algoritmų rezultatų kokybė, nors iš dalies ta kokybė priklauso ir nuo algoritmo savybių. Todėl toliau darbe didžiausias dėmesys bus skirtas diskretizacijos algoritmams.

Tolimesniame skyriuje pateikiame diskretizavimo algoritmų bandymo rezultatus.

3. Rezultatai

Norint visapusiškai atlikti diskretizacijos algoritmų testavimą buvo pasirinkti trys skirtingo dydžio ir skirtingomis savybėmis pasižymintys duomenų rinkiniai.

- Pirmasis rinkinys sudarytas iš 14 savybių ir 90 objektų. Visi objektai turi pilnus savybių rinkinius. Testavimui buvo naudojami šio rinkinio poaibiai sudaryti iš 45, 46 ir 90 pirmųjų rinkinio objektų. ad kriterijaus testavimas prasidėjo su 45 pirmaisiais rinkinio objektais. Žemiau pateiktose lentelėse šis duomenų rinkinys identifikuojamas kaip $CID13$.

- Antrasis rinkinys sudarytas iš 25 savybių ir 86 objektų. Dalis objektų turi nežinomas savybes, šios savybės duomenų rinkinyje pažymėtos -1. Rinkinyje trūksta 1567 arba 72,88% objektų savybių. Testavimui buvo naudoti šio rinkinio poaibiai sudaryti iš 43, 44, 86 pirmųjų rinkinio objektų. ad kriterijaus testavimas prasidėjo su 43 pirmaisiais rinkinio objektais. Žemiau pateiktose lentelėse šis duomenų rinkinys identifikuojamas kaip $CID02$.

- Trečiasis duomenų rinkinį sudaro 100 savybių ir 10000 objektų. Dalis objektų turi nežinomas savybes, šios savybės duomenų rinkinyje pažymėtos -1. Rinkinyje trūksta 29927 arba 2,9927% objektų savybių. Testavimui buvo naudoti šio rinkinio poaibiai sudaryti iš 5000, 5001, 10000 pirmųjų rinkinio objektų. ad kriterijaus testavimas prasidėjo su 5000 pirmaisiais rinkinio objektais. Žemiau pateiktose lentelėse šis duomenų rinkinys identifikuojamas kaip $Matrica2$.

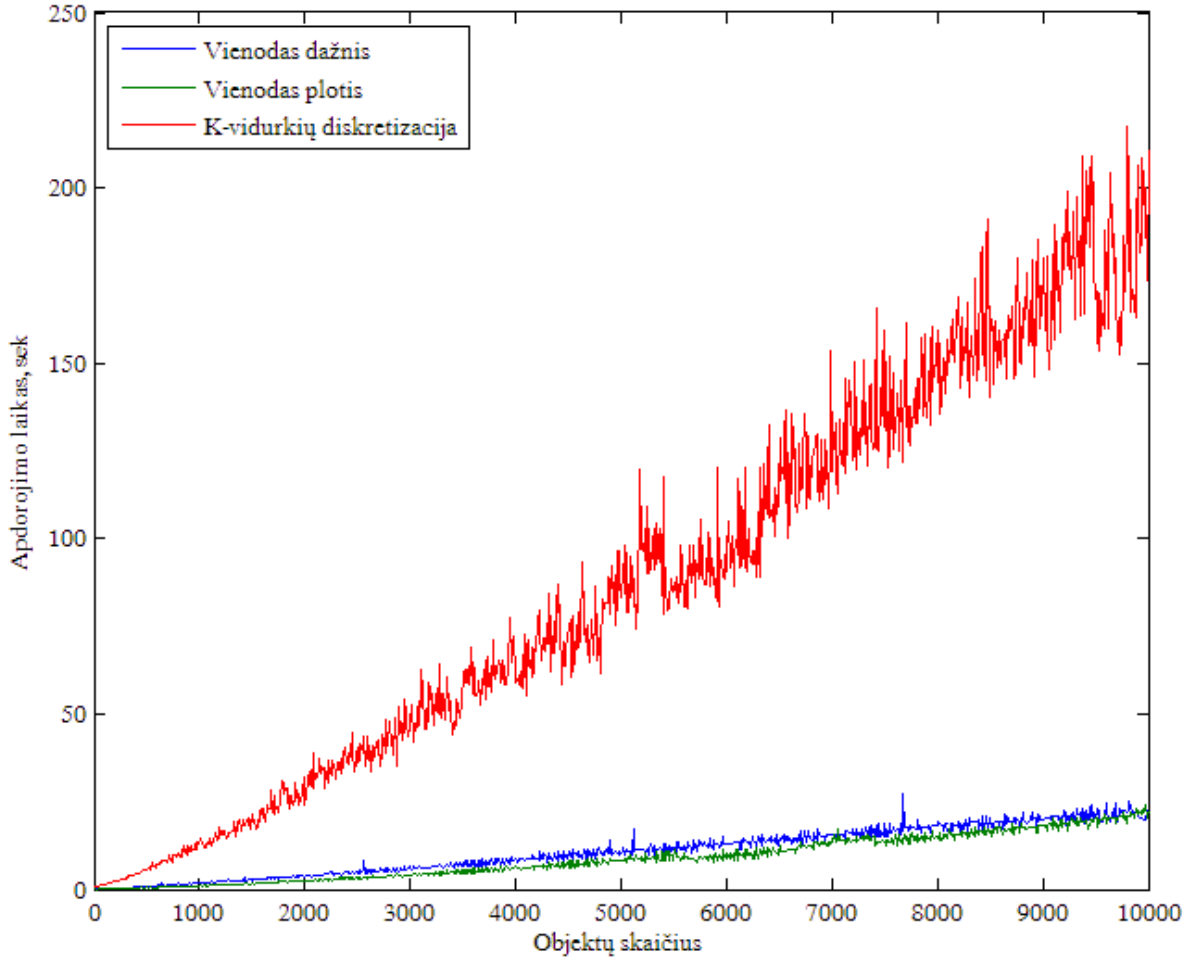
Pirmieji du rinkiniai yra realūs duomenys paimti iš veikiančios sistemos. Trečiasis duomenų rinkinys yra sugeneruotas. Pagrindinė jo paskirtis yra parodyti kaip algoritmai elgiasi esant dideliems duomenų kiekiams.

Visi testuojami algoritmai buvo realizuoti MathWorks MATLAB®7.1 (R14 SP3) matematiniu paketu. Buvo naudojami MATLAB®bazinis paketas ir Statistics toolbox. Paketas buvo įdiegtas į tą patį diską kaip ir operacinė sistema. Algoritmai buvo iš dalies optimizuoti M-Lint [20] rezultatais ir MathWorks kompanijos rekomendacijomis [19].

Visų testų atlikimui buvo naudojamas kompiuteris su Intel®Pentium®4 procesoriumi su NorthWood branduoliu ir 512 kb spartinančiosios atminties, kurio taktinis dažnis 2,8 GHz, darbinės atminties kiekis 1024 Mb (atminties tipas DDR-400), atmintis dirbo dviejų kanalų režimu, motininės plokštės mikroschemų rinkinys turėjo aktyvuotą PAT[1] technologijos palaikymą, operacinė sistemą Microsoft®Windows®XP SP2. Skaičiavimams skirti duomenys ir rezultatai buvo saugomi atskirame fiziniame diske nuo sistemos ir kitos programinės įrangos skirtos testavimui. Po kiekvieno testavimų etapo tarpiniai rezultatai buvo perkeliami į kitą diską, kietasis diskas defragmentuojamas ir perkraunamas kompiuteris, norint išvengti MATLAB®bandymų paspartinti vykdymą talpinant funkcijas į tarpinę saugyklą.

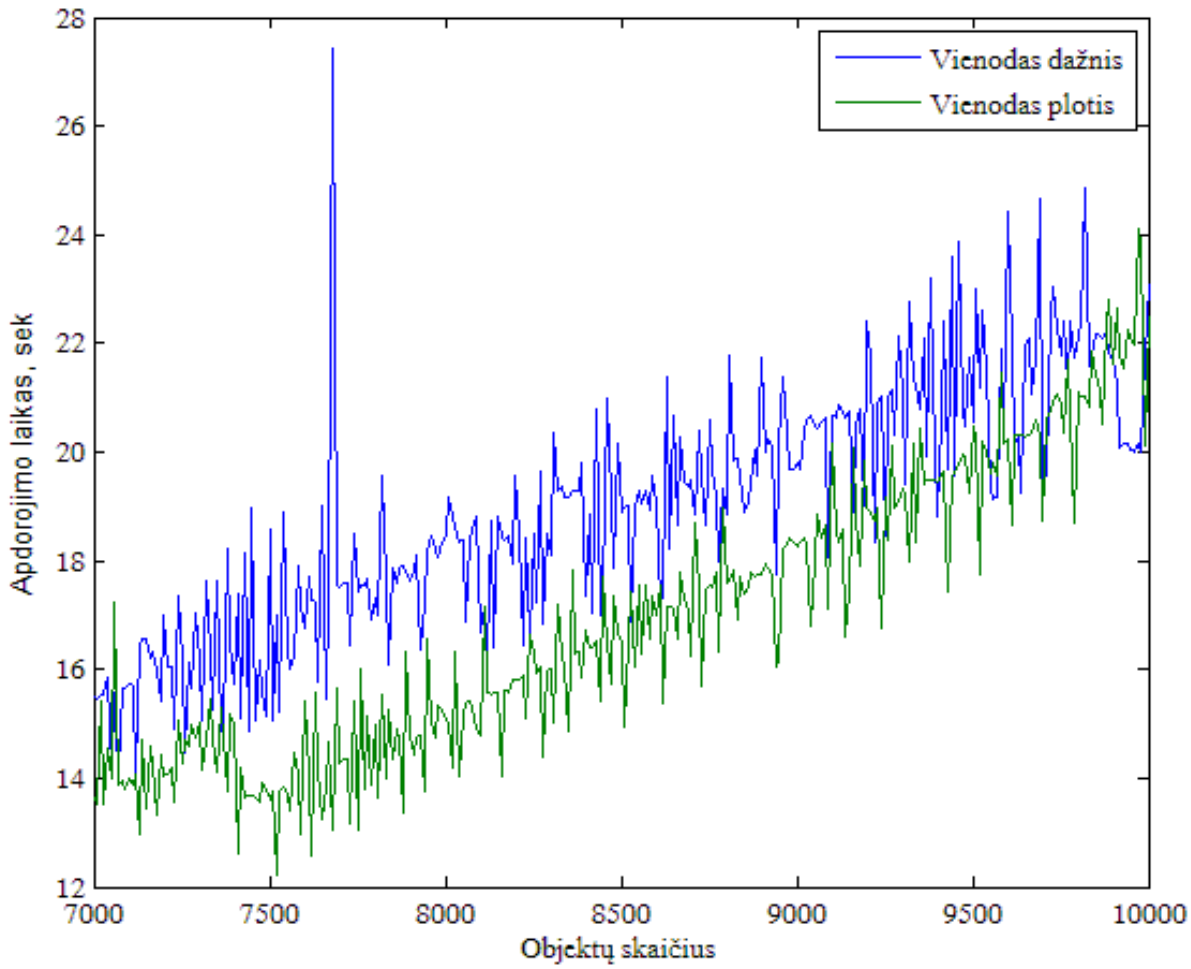
[1] PAT (Performance Acceleration Technology) - našumo didinimo technologija

Atliekant tyrimą buvo analizuojami ne tik tie algoritmų aspektai, kurie aprašyti tyrimo metodų dalyje, bet ir pažiūrėta kaip elgiasi algoritmai vykdymo laiko požiūrių pradedant nuo mažiuko rinkinio ir baigiant dideliu. Šiam eksperimentui atlikti buvo pasirinktas $Matrica2$ duomenų rinkinys. Buvo stebima kaip kinta kiekvienos iteracijos vykdymo laikas didinant objektų skaičių 10 objektų.



4 pav.: Laiko kaštai kintant objektų skaičiui

Kaip matyti iš grafiko pavaizduoto 4 paveiksle vienodo dažnio ir vienodo pločio diskretizacijos algoritmai atlieka diskretizaciją pakankamai greitai ir laiko pokytis didėjant elementų skaičiui didėja nežymiai palyginti su k-vidurkių klasterizacijos diskretizacija. Grafike labai aiškiai matyti dideli diskretizavimo laiko svyravimai k-vidurkių klasterizacijos diskretizacijos algoritmo šakoje, tai gali sukelti sunkumų prognozuojant reikalingą laiko tarpą atlikti diskretizacijai. 5 paveiksle pavaizduota 4 paveikslo detalizacija (pavaizduoti tik du greičiausiai dirbantys diskretizacijos algoritmai, vienodo dažnio ir vienodo pločio), kai objektų skaičius yra tarp 7000 ir 10000. Šiame paveiksle gerai matosi, kad abiejų diskretizacijos algoritmų duomenų apdorojimo laikas tolygiai ilgėja. Beveik viso našumo testo metu vienodo pločio diskretizacijos algoritmas buvo sparčiausias, vidutiniškai artimiausią spartos požiūriu algoritmą (vienodo dažnio diskretizacijos) lenkdamas apie



5 pav.: Vienodo pločio ir vienodo dažnio diskretizacijos algoritmų darbo sparta esant dideliam objektų skaičiui

20%, o k-vidurkių klasterizacijos diskretizacijos algoritmas lenkiamas darbo spartos požiūriu net 880%.

Žemiau pateikiami kiekvieno testuojamo diskretizacijos algoritmo savybės tyrimo rezultatai.

3.1. Vienodo dažnio diskretizacijos algoritmas

Atlikus testavimą profiliavimo įrankio pagalba [19] buvo pastebėta, kad apdorojant visus tris duomenų rinkinius daugiausiai laiko (atitinkamai 71%, $48,8\%$ ir $56,1\%$) yra skiriama eilučių rūšiavimui didėjimo tvarka prieš atliekant pačią diskretizaciją. Rūšiavimui buvo naudojama integruota f-ja $sortrows$, atlikus šios funkcijos analizę gal būtų galima parašyti optimizuotą funkciją, kuri veiktų sparčiau mūsų atveju. Analizuojamo algoritmo tiriamos savybės pavaizduotos 1, 2, 3 lentelėse. Kaip matyti 1, 2 lentelių $lp1o$ eilutes savybės rezultato stulpelyje gautos reikšmes yra mažesnės už vieną (atitinkamai $0,871387$ ir $0,973036$) to priežastis visų pirma yra labai trumpi vykdymo laikai (atitinkamai trumpesni nei $\frac{1}{56}$ ir $\frac{1}{42}$ sekundės dalys) ir bandymo platformos opti-

1 lentelė: Vienodo dažnio diskretizacijos rezultatai $CID13$ duomenų rinkiniui, kur ds - darbo sparta, lp1o - algoritmo laiko sąnaudų pokytis atsiradus vienam naujam objektui, lp2d - algoritmo laiko sąnaudų pokytis padidėjus objektų skaičiui du kartus, pr - sunaudojami procesoriaus resursai, rp - rezultatų pastovumas, ar - sunaudojami atminties resursai, ad - atnaujinimo dažnis.

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,015897$	$0,017267$	$0,02637$	$0,024520$	$0,016752$	–	$0,022161$
$lp1o$	$0,015637$	$0,015561$	$0,018238$	$0,020069$	$0,018336$	$0,017568$	$0,871387$
$lp2d$	$0,023688$	$0,025529$	$0,019764$	$0,018764$	$0,030007$	$0,0235564$	$1,168403$
pr	5%	5%	5%	5%	5%	–	5%
rp	0	0	0	0	–	0	100%
ar	20752	20752	20752	20752	20752	–	20752
ad	–	–	–	–	–	–	$\frac{1}{45}$

2 lentelė: Vienodo dažnio diskretizacijos rezultatai $CID02$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,032651$	$0,021357$	$0,022321$	$0,021566$	$0,024341$	–	$0,024447$
$lp1o$	$0,024900$	$0,029756$	$0,021435$	$0,021651$	$0,021198$	$0,023788$	$0,973036$
$lp2d$	$0,033838$	$0,031169$	$0,027345$	$0,027482$	$0,022980$	$0,028563$	$1,168465$
pr	14%	14%	14%	14%	14%	–	14%
rp	0	0	0	0	–	0	100%
ar	35064	35064	35064	35064	35064	–	35064
ad	–	–	–	–	–	–	$\frac{1}{43}$

mizacijos pasekmė, kurių buvo bandyta išvengti. Maži procesoriaus apkrovos (pr) rodikliai šioje lentelėse susiję taip pat su labai trumpalaikiu šių algoritmų veikimu. 3 lentelėje $lp1o$ savybės kriterijus taip pat mažesnis už vieną, bet labai nežymiai (nuokrypis mažesnis nei $0,014\%$), todėl galima laikyti, kad objektų rinkinio dydžiui pakitus vienu objektu vykdymo laikas beveik nesikeičia. Visose lentelėse vaizduojančiose šio diskretizacijos algoritmo testavimo rezultatus rp savybė lygi 100% ir ad savybės vertė lygi likusių rinkinio objektų skaičiui, nes šis algoritmas neturi jokio atsitiktinumo faktoriaus savyje.

3.2. Vienodo pločio diskretizacija

Atlikus testavimą profiliavimo įrankio pagalba [19] buvo pastebėta, kad apdorojant pirmuosius du duomenų rinkinius ($CID13$ ir $CID02$) daugiausiai laiko (atitinkamai $51,6\%$ ir $48,2\%$) yra sunaudojamą duomenų saugojimui. Apdorojant $Matrica2$ duomenų rinkinį daugiausiai laiko ($45,1\%$) sugaištama išrenkant iš duomenų rinkinio neegzistuojančias reikšmes ir pasiruošiant

3 lentelė: Vienodo dažnio diskretizacijos rezultatai $Matrica2$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$9,303926$	$9,314896$	$9,273600$	$9,268855$	$9,393906$	–	$9,311037$
$lp1o$	$9,192614$	$9,178257$	$9,152766$	$9,179629$	$9,216890$	$9,184031$	$0,986360$
$lp2d$	$19,036156$	$19,051533$	$18,909160$	$18,965618$	$18,912656$	$18,975025$	$2,037907$
pr	49%	50%	49%	49%	49%	–	$49,2\%$
rp	0	0	0	0	–	0	100%
ar	16040920	16040920	16040920	16040920	16040920	–	16040920
ad	–	–	–	–	–	–	$\frac{1}{5000}$

4 lentelė: Vienodo pločio diskretizacijos rezultatai $CID13$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,009746$	$0,014274$	$0,011209$	$0,012323$	$0,011693$	–	$0,011849$
$lp1o$	$0,015078$	$0,009898$	$0,011797$	$0,028836$	$0,015340$	$0,016190$	$1,366343$
$lp2d$	$0,014943$	$0,017281$	$0,021153$	$0,016972$	$0,017121$	$0,017494$	$1,476412$
pr	12%	12%	12%	12%	12%	–	12%
rp	0	0	0	0	–	0	100%
ar	20600	20600	20600	20600	20600	–	20600
ad	–	–	–	–	–	–	$\frac{1}{45}$

kiekvienos savybės minimalios ir maksimalios reikšmės paieškai. Remiantis šiais skaičiais galima daryti išvadą, kad šis algoritmas gerai tinka darbui su didesniais duomenų rinkiniais. Analizuojamo algoritmo tiriamos savybės pavaizduotos 4, 5, 6 lentelėse. $lp1o$ reikšmė $Matrica2$ duomenų rinkinio lentelėje ir $lp2d$ duomenų rinkinio $CID02$ tyrimų rezultatų lentelėje yra mažesni nei vienas (nuokrypis lygus atitinkamai $0,036639\%$ ir $0,010495\%$ duomenų rinkiniams). $CID02$ duomenų rinkinio atveju trumpas algoritmo veikimo laiko tarpas ir mažas duomenų rinkinys kartu su sistemos fone atliekamomis funkcijomis galėjo sąlygoti tokius rezultatus (nors duomenų rinkinys padidėjo du kartus, jo objektų skaičius padidėjo tik 43 objektais). 6 lentelės $lp1o$ savybės galutinio rezultato reikšmė gali buti įtakota sistemos atliekamos funkcijos foniniame režime atliekant ds savybės skaičiavimus. pr reikšmės pavaizduotos 4 ir 5 lentelėse yra didesnės nei tų pačių duomenų rinkinio pr savybės reikšmės vienodo dažnio diskretizacijos algoritmo, tai susiję su tuo kad realizuojant šį algoritmą nebuvo naudojamos jokios išorinės funkcijos. Visose lentelėse vaizduojančiose šio diskretizacijos algoritmo testavimo rezultatus rp savybė lygi 100% ir ad savybės vertė lygi likusių rinkinio objektų skaičiui, nes šis algoritmas neturi jokio atsitiktinumo faktoriaus savyje.

5 lentelė: Vienodo pločio diskretizacijos rezultatai $CID02$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,013038$	$0,024171$	$0,023205$	$0,023559$	$0,024743$	–	$0,021745$
$lp1o$	$0,012910$	$0,023195$	$0,023795$	$0,023980$	$0,026457$	$0,022067$	$1,014817$
$lp2d$	$0,022702$	$0,026839$	$0,018635$	$0,017813$	$0,021597$	$0,0215172$	$0,989515$
pr	16%	16%	16%	16%	16%	–	16%
rp	0	0	0	0	–	0	100%
ar	36704	36704	36704	36704	36704	–	36704
ad	–	–	–	–	–	–	$\frac{1}{43}$

6 lentelė: Vienodo pločio diskretizacijos rezultatai $Matrica2$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$12,516904$	$13,180708$	$12,683876$	$11,971433$	$12,519481$	–	$12,574480$
$lp1o$	$12,469880$	$11,698598$	$12,364461$	$11,691241$	$12,344668$	$12,113770$	$0,963361$
$lp2d$	$17,539462$	$17,067718$	$16,928938$	$16,844765$	$16,951138$	$17,066404$	$1,357225$
pr	48%	50%	49%	50%	50%	–	$49,4\%$
rp	0	0	0	0	–	0	100%
ar	16002504	16002504	16002504	16002504	16002504	–	16002504
ad	–	–	–	–	–	–	$\frac{1}{5000}$

3.3. K-vidurkių klasterizacijos diskretizacija

Atlikus testavimą profiliavimo įrankio pagalba [19] buvo pastebėta, kad apdorojant visus tris duomenų rinkinius daugiausiai laiko (atitinkamai $90,7\%$, $90,7\%$ ir $96,5\%$) yra skiriama išorinės funkcijos $kmeans$ kvietimui, kuri atlieka diskretizaciją, ši funkcija yra Statistics toolbox dalis [12]. Analizuojamo algoritmo tiriamos savybės pavaizduotos 7, 8, 9 lentelėse. savybės $lp1o$ reikšmės 8 ir 9 lentelėse yra nežymiai mažesnės už 1 (nukrypimas lygus $0,018808\%$ ir $0,003092\%$ atitinkamai), todėl į šį nukrypimą galima nekreipti dėmesio. Kaip parodė testavimas visų duomenų rinkinių atžvilgiu rp yra nedidelis ir mažėja didėjant duomenų rinkiniui. Atlikus $CID13$ ir $CID02$ klasterizacijos duomenų analizę buvo pastebėta, kad didelė dalis objektų savybių grupelių lieka kartu atliekant pradinio duomenų rinkinio diskretizaciją, bet pakliūva į kitą klasę. Tokie rezultatai rodo, kad k-vidurkių klasterizacijos diskretizacijos algoritmas gerai aptinka objektų grupavimąsi duomenų rinkinyje tik su minimaliomis paklaidomis. Bet geresnių rp savybės įvertė gauti trukdo šio algoritmo atsitiktinė pradžia, t.y. pradedant diskretizaciją visa objektų rinkinio savybės erdvė yra atsitiktiniu būdu suskaidoma į mūsų nurodytą intervalų skaičių, o po to tik susidariusios klasės optimizuojamos. Šitokių rezultatų priežastis yra pats algoritmas, kuris pačioje savo darbo pradžioje visus rinkinio objektus atsitiktiniu būdu suskirsto į klases. Atnaujinimo dažnis (ad)

7 lentelė: K-vidurkių klasterizacijos diskretizacijos rezultatai $CID13$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,078393$	$0,073257$	$0,057237$	$0,064162$	$0,063111$	–	$0,067232$
$lp1o$	$0,083372$	$0,070879$	$0,079034$	$0,077217$	$0,077404$	$0,0775812$	$1,153933$
$lp2d$	$0,088558$	$0,079290$	$0,089207$	$0,076348$	$0,083131$	$0,0833068$	$1,239094$
pr	10%	14%	12%	13%	12%	–	$12,2\%$
rp	347	298	307	233	–	$296,25$	$52,976191\%$
ar	20252	20252	20252	20252	20252	–	20252
ad	–	–	–	–	–	–	1

8 lentelė: K-vidurkių klasterizacijos diskretizacijos rezultatai $CID02$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,114715$	$0,104673$	$0,110860$	$0,110100$	$0,110336$	–	$0,110137$
$lp1o$	$0,125581$	$0,105652$	$0,098941$	$0,101113$	$0,109040$	$0,108065$	$0,981192$
$lp2d$	$0,132390$	$0,137425$	$0,131076$	$0,128085$	$0,125322$	$0,130860$	$1,188155$
pr	30%	35%	37%	34%	36%	–	$34,4\%$
rp	1415	1392	1390	1353	–	$1387,5$	$35,465116\%$
ar	34492	34492	34492	34492	34492	–	34492
ad	–	–	–	–	–	–	1

visų testuotų duomenų rinkinių atžvilgiu lygus 1, tai susiję su rp reikšmėmis. Esant mažoms rp reikšmėms pakartotinis algoritmo vykdymas ant tų pačių duomenų gražina pakankamai skirtinga diskretizuotą aibę, kuri savo ruožtu netenkina sąlygų apibrėžtų ad kriterijuje. Atlikus k-vidurkių klasterizacijos diskretizacijos algoritmo ilgalaikį testavimą galimos ad reikšmės galėtų būti gautos $\frac{1}{5}$ ir $\frac{1}{10}$, bet atlikus dar vieną pakartotinį bandymą ad reikšmė vėl padidėtų iki 1.

Apibendrinant bandymų rezultatus pavaizduotus 1, 2, 3, 4, 5, 6, 7, 8, 9 lentelėse sudaryta 10 lentelė, kuri parodo visų tirtų diskretizacijos algoritmų savybes skirtingų duomenų rinkinių atžvilgiu.

3.4. Renyi entropijos klasterizacijos algoritmas

Atlikus testavimą profiliavimo įrankio pagalba [19] buvo pastebėta, kad apdorojant duomenų rinkinį CID13 daugiausiai laiko buvo sugaišta duomenų išsaugojimui ir Gauso simetrinio branduolio apskaičiavimui. Analizuojamo algoritmo tiriamos savybės pavaizduotos 11, 12 lentelėse. Savybės $lp1o$ reikšmės yra nežymiai mažesnės už 1 ($0,993171\%$ ir $0,958759\%$), todėl į šį nukrypimą galima nekreipti dėmesio. Kaip parodė testavimas visų duomenų rinkinių atžvilgiu rp yra geras. pr kriterijus netenkina keliamų reikalavimų, tai reiškia kad vykdomas algoritmas su-

9 lentelė: K-vidurkių klasterizacijos diskretizacijos rezultatai $Matrica2$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$76,697130$	$80,234864$	$72,324608$	$78,266760$	$77,850186$	–	$77,074710$
$lp1o$	$69,987920$	$85,162754$	$71,628754$	$72,995884$	$84,406543$	$76,836371$	$0,996908$
$lp2d$	$188,877550$	$162,498500$	$155,306590$	$161,574180$	$172,605740$	$168,172511$	$2,181942$
pr	50%	50%	50%	50%	50%	–	50%
rp	432090	437235	438312	440198	–	$436958,75$	$12,608250\%$
ar	16000092	16000092	16000092	16000092	16000092	–	16000092
ad	–	–	–	–	–	–	1

10 lentelė: Duomenų rinkinių ir diskretizacijos algoritmų tyrimo kriterijų suvestinė

Duomenų rinkinys	Diskretizacijos algoritmas	Kriterijus						
		ds	$lp1o$	$lp2d$	pr	rp	ar	ad
$CID13$	Vienodo dažnio	$0,022161$	$0,871387$	$1,1638403$	5%	100%	20752	$\frac{1}{45}$
	Vienodo pločio	$0,011849$	$1,366343$	$1,476412$	12%	100%	20600	$\frac{1}{45}$
	k-vidurkių	$0,067232$	$1,153933$	$1,239094$	$12,2\%$	$52,976191\%$	20252	1
$CID02$	Vienodo dažnio	$0,02447$	$0,973036$	$1,168465$	14%	100%	35064	$\frac{1}{43}$
	Vienodo pločio	$0,021745$	$1,014817$	$0,989515$	16%	100%	36704	$\frac{1}{43}$
	k-vidurkių	$0,110137$	$0,981192$	$1,188155$	$34,4\%$	$35,465116\%$	34492	1
$Matrica2$	Vienodo dažnio	$9,311037$	$0,986360$	$2,037907$	$49,2\%$	100%	16040920	$\frac{1}{5000}$
	Vienodo pločio	$12,574480$	$0,963361$	$1,357225$	$49,4\%$	100%	16002504	$\frac{1}{5000}$
	k-vidurkių	$77,074710$	$0,996908$	$2,181942$	50%	$12,6080250\%$	16000092	1

naudoja per daug sistemos resursų ir gali trukdyti kitų vartotojų darbui. Šita problema gali būti išspręsta optimizuojant algoritmo kodą.

Atlikus tyrimus su $CID13$ duomenų rinkiniu buvo gauti tokie klasteriai:

- 9, 25, 40
- 1, 4, 8, 13, 14, 22, 35, 36, 41, 44
- 10, 20, 37
- 2, 3, 5, 6, 7, 11, 12, 15, 16, 17, 18, 19, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 38, 39, 42, 43, 45

Išanalizavus šiuos klasterius buvo sudarytos tokios taisyklės.

1-as klasteris:

- Savybė3 = 1;

11 lentelė: Renyi entropijos diskretizacijos rezultatai $CID13$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,539630$	$0,512143$	$0,538414$	$0,528552$	$0,521472$	–	$0,528042$
$lp1o$	$0,538197$	$0,543752$	$0,516248$	$0,513554$	$0,510429$	$0,524436$	$0,993171\%$
$lp2d$	$0,600836$	$0,563915$	$0,568682$	$0,537157$	$0,556677$	$0,565453$	$1,070849\%$
pr	30%	28%	31%	30%	29%	–	$29,6\%$
rp	1	3	1	2	–	$1,75$	$3,8888\%$
ar	74696	83216	79140	81282	80370	–	79741
ad	–	–	–	–	–	–	$\frac{1}{20}$

12 lentelė: Renyi entropijos diskretizacijos rezultatai $CID02$ duomenų rinkiniui

Kriterijus	Eksperimento numeris					Tarpinis rezultatas	Savybės rezultatas
	1	2	3	4	5		
ds	$0,569139$	$0,578710$	$0,567964$	$0,691630$	$0,559026$	–	$0,593295$
$lp1o$	$0,593976$	$0,549836$	$0,557281$	$0,562281$	$0,580762$	$0,568827$	$0,958759\%$
$lp2d$	$0,818652$	$0,699575$	$0,674955$	$0,662395$	$0,718079$	$0,714731$	$1,204681\%$
pr	33%	29%	36%	29%	29%	–	$31,2\%$
rp	4	3	2	2	–	$2,75$	$6,39535\%$
ar	76096	81216	79840	81282	80970	–	79881
ad	–	–	–	–	–	–	$\frac{1}{19}$

- Savybė5 = 1;
- Savybė7 = 2;
- Savybė8 = 1;
- Savybė9 = 3;
- Savybė10 = 1;
- Savybė11 = 1;
- Savybė12 = 2;
- Savybė13 = 1;
- Savybė14 = 1.

3-as klasteris:

- Savybė2 = 1;

- Savybė3 = 1;
- Savybė4 = 1;
- Savybė10 = 1;
- Savybė11 = 1;
- Savybė12 = 2;
- Savybė13 = 1;
- Savybė14 = 1.

Antram ir ketvirtam klasteriui nepavyko išvesti taisyklės, kuri būtų tikusi visiems to klasterio objektams. Tai gali būti susiję su tuo, kad priskiriant ieškant "Blogiausio klasterio" naudojamas Gauso branduolys bei logaritmai.

4. Išvados

Diskretizacijos algoritmai yra svarbi klasterizacijos proceso sudedamoji dalis, leidžianti paspartinti algoritmų veikimą. Dalis klasterizacijos algoritmų turi savyje integruotus diskretizacijos algoritmus, kitiems algoritmams reikalingas išankstinis duomenų apdorojimas diskretizacijos algoritmais.

Darbe buvo apžvelgti vienodo pločio, vienodo dažnio ir k-vidurkių klasterizacijos diskretizacijos algoritmai. Atlikus šių algoritmų testavimą su probleminės srities duomenimis (realiais ir sugeneruotais) buvo nustatyta, kad tinkamiausias algoritmas yra vienodo pločio diskretizavimo algoritmas dėl jo mažo atminties naudojimo, lėtesnio nei tiesinis laiko pokyčio didėjant objektų skaičiui, bei didelio rezultatų pastovumo. Kaip alternatyvą šiam algoritmui galima naudoti vienodo dažnio diskretizavimo algoritmą, kuris turi daugelį vienodo pločio diskretizavimo algoritmo savybių, bet laiko sąnaudos didėja sparčiau didėjant objektų skaičiui.

Atlikus literatūros analizę kaip perspektyviausias iš apžvelgtų klasterizacijos algoritmų buvo pasirinktas Renyi entropijos klasterizacijos algoritmas, nes jo realizacija KU VMA neturėtų sukelti problemų, dėl algoritmo naudojamų funkcijų. Buvo atlikta šio algoritmo realizacija ir pirminis testavimas su realiais duomenimis. Remiantis testavimo rezultatais buvo pastebėta, kad algoritmo pradiniu klasterių mazgų pasirinkimas atsitiktiniu būdu apsunkina pastovių taisyklių išvedimą. Taisyklių išvedimą iš sudarytų klasterių pavyko dalinai, buvo išvestos taisyklės dviems klasteriams iš keturių. Atlikus šio diskretizacijos algoritmo testavimą buvo nustatyta kad jis tenkina daugelį iškeltų kriterijų, todėl jo tolimesnis naudojimas ir galimas diegimas į VMA gali būti realizuotas su mažais optimizavimais.

Neišspręstos problemos ir ateities darbai:

- Rasti efektyvų būdą taisyklių ištraukimui iš klasterių.
- Išanalizuoti diskretizacijos ir klasterizacijos algoritmų įdiegimo galimybes į Klaipėdos universiteto naudojamą VMA.
- Realizuoti pasirinktą diskretizacijos ir klasterizacijos algoritmus VMA.

Literatūra

[1] D. Baziukaitė, Making Virtual Learning Enviroment More Intelligent: Aplication of Markov Decision Process, Liet. Mat. Rink. 2005 vasario mėn.

[2] Y. Kondratoff and G. Tecuci, Learning Based on Conceptual Distance, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL 10, NO 6, NOVEMBER 1988

[3] Sanjiv K. Bhatia and Jitender S. Deogun, Conceptual Clustering in Information Retrieval, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS, VOL. 28, NO. 3, JUNE 1998

[4] K. Cios, Witold Pedrycz, Roman Swiniarski, Data Minning Methods for Knowledge Discovery, Kluwer Academic Publishers, 2000.

[5] G. Biswas, Jerry B. Weinberg, and Douglas H. Fisher, ITERATE: A Conceptual Clustering for Data Mining, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEWS, VOL. 28, NO. 2, MAY 1998

[6] Y. Quintana, Cognitive Approaches to Intelligent Information Retrieval, CCECE'97

[7] M. Sato-Ilk, On Evaluation of Clustering Homogeneity Analysis, Systems, Man, and Cybernetics, 2000 IEEE International Conference, Volume 5, 8-11 Oct. 2000 Page(s):3588 - 3593 vol.5

[8] R. Jenssen, Kenneth E. Hild II, Deniz Erdogmus, Jose C. Principe and Torbjorn Eltoft Clustering using Renyi's Entropy, Neural Networks, 2003. Proceedings of the International Joint Conference, Volume 1, 20-24 July 2003 Page(s):523 - 528 vol.1

[9] Z. Rached, F. Alajaji, L.Lorne Campbell, Renyi's Divergence and Entropy Rates for Finite Alphabet Markov Sources, IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 47, NO. 4, MAY 2001

[10] C. Wu, C. Yu, S. Lee, An Entropy-based Evaluation Function for Conceptual Clustering, Systems, Man and Cybernetics, 1995. 'Intelligent Systems for the 21st Century'., IEEE International Conference Volume 5, 22-25 Oct. 1995 Page(s):4307 - 4312 vol.5

[11] Syed Sibte Raza Abidi, Jason Ong, A Data Minning Strategy for Inductive Data Clustering: A Synergy Between Self-Organising Neural Networks and K-Means Clustering Techniques, IEEE, 2000

[12] K-means Clustering (http://www.mathworks.com/access/helpdesk/help/toolbox/stats/f74589.html), žiūrėta 2006.01.10

[13] G. H. Ball, Data analysis in the social sciences: What about the details, Proc. AFIPS Fall Joint Comput. Conf., 1965

[14] W Bialek, I Nemenman, and N Tishby. Predictability, complexity, and learning. Neur. Comp., 13:2409-2463, 2001

[15] M. Gluck, J. Corter, Information, uncertainty, and the utility of categories, Proc. 7th Ann. Conf. Cognitive Sci. Soc., 1985

[16] Cios K.J., Wedding D.K. and Liu N, CLIP3: cover learning using integer programming, Kybernetes (invited paper), 26(4-5), 1997

[17] Robert C. Holte, Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning 11:63-91, 1993

[18] Robert C. Holte, Liane E. Acker, Bruce W. Porter, Concept Learning and the Problem of Small Disjuncts, Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pp. 813-818, 1989

[19] MATLAB Programming, MathWorks, June 2004

[20] MATLAB Desktop Tools and Development Enviroment, Mathworks, June 2004

[21] A. Antos and I. Kontoyiannis. Estimating the entropy of discrete distributions. IEEE International Symposium on Information Theory, 2001.

[22] I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. Adv. Neural Inf. Proc. Syst. 14, Cambridge, MA, 2002. MIT Press

[23] Ray J. Solomonoff. Progress in incremental machine learning. NIPS'02 Universal Learning workshop, 2002

[24] S. Still and W. Bialek. How many clusters? An information-theoretic perspective. Neural Comp., 16(12):2483-2506, 2004

[25] J. Principe and D. Xu. Information-theoretic learning using renyi's quadratic entropy. Proc. 1st Intl. Workshop on Independent Component Analysis (ICA '99), pages 407-412, Aussois, France, 1999

[26] Christopher J. Paciorek and Mark J. Schervish. Nonstationary Covariance Functions for Gaussian Process Regression. NIPS'03 Entropy estimation wirkshop, 2003

[27] Chris Fraley and Adrian Raftery. Model-based clustering, discriminant analysis, and density estimation. J. Am. Stat. Assoc., 97(458):611-641, 2002

A. Vienodo dažnio diskretizacijos algoritmo kodas

```
function [ klasteriai ] = vienodas_daznis( pr, inter, nr )
% Diskretizuoja pateiktus duomenis vienodo daznio algoritmu
%   funkcijai reikai gauti pradinius duomeis, kurie yra nurodomi ja
%   iskvieciant, rezultatai pateikiami kartu su pradiniais duomenimis
%   (pirma pateikiami pradiniai duomenys, o po ju rezultatai). Baigus
%   viekti funkcijai i konsole isvedamas laiko tarpas per kuri buvo atlikta
%   si funkcija.
%
% Pradinius duomenys:
%   pr - pradine iverciu matrica
%   inter - diskreciu intervalu skaicius
%tic; % laiko atskaitos pradzia
t1 = cputime;
dydis = size(pr); % randami gautu duomenu ismatavimai
k1 = dydis(1,1); % eiluciu skaicius
k2 = dydis(1,2); % stulpeliu skaicius
for i2 = 1:k2
    i = 1;
    for i1 = 1:k1
        if pr(i1,i2) >= 0
            i = i + 1;
        end;
    end;
    daznis(1,i2) = i / inter; % prelimenarus plotis. Nes didele tikimybe,
                              % kad bus reiksme po kablelio
    clear i;
end;
for i = 1:k1 % ikeliame papildoma stulpeli, reikalinga pirminei duomenu
             % sekai atstatyti
    pr(i,k2+1)=i;
end;
clear i;
for i2 = 1:k2
    pr = sortrows(pr, i2); % rusiavimas didejimo tvarka i2-ojo stulpelio
    i = 1; % klases numeris
    i3 = 0;
    for i1 = 1:k1
        if pr(i1,i2) < 0
            pr(i1,i2+k2+1) = -1;
        elseif i3 + 1 <= daznis(1,i2) * i % jei lelemanto eiles numeris
                % mazesnis nei tos iteracijos maksimalaus daznio pozicija
                % objektas priskiriams prie esamos klases
            pr(i1,i2+k2+1) = i;
            i3 = i3 + 1;
        else
```

```
46              i = i + 1; % didadame klases numeri
47              pr(i1,i2+k2+1) = i; %priskiriame naujai klasei
48              i3 = i3 + 1;
49          end;
50      end;
51  end;
52  pr = sortrows(pr, k2+1); % pradinio duomenu eiliskumo atstatymas
53  pr1=pr(1:k1,k2+2:2*k2+1); % tik rezultatu srities paruosimas isvedimui
54  vieta = ['vienodas_daznis' int2str(nr) '.txt'];
55  save (vieta, '-ascii', '-double', '-tabs', 'pr1');
56      % rezultatu saugojimas i faila
57  %klasteriai = pr1; % grazinamos reiksmes priskyrimas
58  klasteriai = cputime - t1;
59  %toc % laiko atskaitos pabaiga ir vykdymo laiko isvedimas i konsole
```

B. Vienodo pločio diskretizacijos algoritmo kodas

```
function [ klasteriai ] = vienodas_plotis_opt( pr, inter, nr )
% Diskretizuoja pateiktus duomenis vienodo plocio algoritmu
%   funkcijai reikai gauti pradinius duomeis, kurie yra nurodomi ja
%   iskvieciant, rezultatai pateikiami kartu su pradiniais duomenimis
%   (pirma pateikiami pradiniai duomenys, o po ju rezultatai). Baigus
%   viekti funkcijai i konsole isvedamas laiko tarpas per kuri buvo atlikta
%   si funkcija.
%
% Pradinius duomenys:
%   pr - pradine iverciu matrica
%   inter - diskreciu intervalu skaicius
tic; % laiko atskaitos pradzia
dydis = size(pr); % pradines matricos ismatavimu gavimas
k1 = dydis (1,1); % eiluciu skaicius
k2 = dydis (1,2); % stulpeliu skaicius
for i2 = 1:k2
    i = 1;
    for i1 = 1:k1
        if pr(i1,i2) >= 0
            a(i,1) = pr(i1,i2);
            i = i + 1;
        end;
    end;
    maxreiks(1,i2) = max (a); % kiekvieno stulpelio maksimalios
                              % reiksmes radimas
    minreiks(1,i2) = min (a); % kiekvieno stulpelio minimalios
                              % reiksmes radimas
    clear i;
    clear a;
end;
plotis = (maxreiks - minreiks) / (inter - 1); % intervalu plociu gavimas
% norint sumazinti sunaudojama laiko tarpa atlikti veiksmams butu galima
% vietoje siu kintamuju reiksmes traukti tiesiogiai is dydis kintamojo, bet
% to pasekme butu sunkiai iskaitomas kodas vidiniame FOR cikle.
for i1 = 1:k1 % eiluciu perejimas
    for i2 = 1:k2 % stulpeliu perejimas
        if pr(i1,i2) == 0 && plotis(1,i2) == 0 % apsauga nuo NaN (0 / 0)
            % jei narys ir plotis abu lygus 0 tada onbjektas
            % priskiriams 0 klasei
            pr(i1,i2+k2) = 0;
        elseif pr(i1,i2) > 0 % tikrinama ar reiksme teigiama
                             % (ne null po isorinio apdorojimo)
            pr(i1,i2+k2) = ceil(pr(i1,i2) / plotis(1,i2)); % reiksmes
                                                           %diskretizacija
        else
```

```
46              pr(i1,i2+k2) = pr(i1,i2); % null reiksme perkeliame
47            end;
48        end;
49    end;
50    pr1=pr(1:k1,k2+1:2*k2); % rezultatu isvedimo srities formavimas
51    vieta = ['vienodas_plotis' int2str(nr) '.txt'];
52    save (vieta, '-ascii', '-double', '-tabs', 'pr1');
53        % rezultatu saugojimas i faila
54    klasteriai = pr1; % grazinamos reiksmes priskyrimas
55    toc % laiko atskaitos pabaiga ir isvedimas i konsole
```

C. k-vidurkių klasterizacijos diskretizacijos algoritmo kodas

```
function [ klasteriai ] = kvid_disk( pr, inter, nr )
% Diskretizuoja pateiktus duomenis vienodo plocio algoritmu
%   funkcijai reikai gauti pradinius duomeis, kurie yra nurodomi ja
%   iskvieciant, rezultatai pateikiami kartu su pradiniais duomenimis
%   (pirma pateikiami pradiniai duomenys, o po ju rezultatai). Baigus
%   viekti funkcijai i konsole isvedamas laiko tarpas per kuri buvo atlikta
%   si funkcija.
%
% Pradinius duomenys:
%   pr - pradine iverciu matrica
%   inter - diskreciu intervalu skaicius
tic; % laiko atskaitos pradzia
dydis = size(pr); % pradines matricos ismatavimu gavimas
k1 = dydis (1,1); % eiluciu skaicius
k2 = dydis (1,2); % stulpeliu skaicius
for i2 = 1:k2 % stulpeliu perejimas
    temp = kmeans(pr(:,i2), inter, 'EmptyAction', 'drop','Display', 'off');
        % reiksmes diskretizacija susidariusius tuscius klasterius
        % saliname ir visus pranesimus apie tai atjungiame
    pr(1:k1,k2+i2)=temp;
    clear temp;
end;
for i1 = 1:k1
    for i2 = 1:k2
        if pr(i1,i2) == -1
            pr(i1,i2+k2) = -1;
        end;
    end;
end;
pr1=pr(1:k1,k2+1:2*k2); % rezultatu isvedimo srities formavimas
vieta = ['kvid_disk' int2str(nr) '.txt'];
save (vieta, '-ascii', '-double', '-tabs', 'pr1');
    % rezultatu saugojimas i faila
klasteriai = pr1; % grazinamos reiksmes priskyrimas
toc % laiko atskaitos pabaiga ir isvedimas i konsole
```

D. Renyi entropijos klasterizacijos algoritmo kodas

```
function [ klasteriai ] = renyi_entropija( pr, klsk, Ninit )
% klasterizuoja pateiktus duomenis Renyi entropijos algoritmu.
%   funkcijai reikai gauti pradinius duomenis, kurie yra nurodomi ja
%   iskvieciant, rezultatai grazinami pasibaigus f-jos veikimui. Baigus
%   viekti funkcijai i konsole isvedamas laiko tarpas per kuri buvo atlikta
%   si funkcija.
%
% Pradinius duomenys:
%   pr - pradine diskretizuota matrica
%   klsk - pradinis klasteriu skaicius
%   Ninit - objektu skaicius pradiniame klasteryje
%t1 = cputime;
sigma = 0.12;
tic; % laiko atskaitos pradzia
dydis = size(pr); % pradines matricos ismatavimu gavimas
k1 = dydis (1,1); % eiluciu skaicius
k2 = dydis (1,2); % stulpeliu skaicius
for i2 = 1:k2
    i = 1;
    for i1 = 1:k1
        if pr(i1,i2) >= 0
            a(i,1) = pr(i1,i2);
            i = i + 1;
        end;
    end;
    maxreiks(1,i2) = max (a); % kiekvieno stulpelio maksimalios reiksmes
                              % radimas
    minreiks(1,i2) = min (a); % kiekvieno stulpelio minimalios reiksmes
                              % radimas
    clear i;
    clear a;
end;

koefx = (maxreiks + minreiks) / 2;
koefy = maxreiks - koefx;

for i1 = 1:k1
    for i2 = 1:k2
        if koefy(i2) == 0
            norm_d(i1,i2)=0;
        else
            norm_d(i1,i2) = (pr(i1,i2) - koefx(i2)) / koefy(i2);
        end;
    end;
end;
```

```
46
47  % GMatricos skaiciavimas
48  IMatrica = eye(k1);
49  for i1 = 1:k1
50      for i2 = 1:k1
51          skirt1 = pr(i1,:)-pr(i2,:);
52          size(skirt1);
53          skirt2 = (sigma*sigma)*IMatrica(i1,1:k2);
54          size(skirt2);
55          GMatrica(i1,i2) = exp (-(sum((skirt1 - skirt2).^2))/(2*sigma^2));
56      end;
57  end;
58
59  % Startiniu tasku isrinkimas
60  for i=1:klsk
61      matrica(i,1,1) = 0;
62      matrica(i,2,1) = 0;
63      matrica(i,3,1) = 0;
64  end
65  matrica(1,k1,1)=0;
66  paimtiObjektai = 0;
67  tmp=0;
68  for i=1:klsk
69      kartoti = 1;
70      while kartoti == 1
71          index = round((k1-1)*rand)+1;
72          dydis1 = size(matrica);
73          skait=0;
74          for i1=1:dydis1(1,1)
75              if matrica(i1,3,1) == index
76                  skait = skait+1;
77              end;
78          end;
79          if skait < 1
80              matrica(i,3,1) = index;
81              matrica(i,1,1) = 1;
82              tmp=tmp+1;
83              paimtiObjektai(tmp,1) = index;
84              kartoti = 0;
85          end;
86      end;
87  end;
88  paimtiObjektai1 = sort(paimtiObjektai);
89  paimtiObjektai = paimtiObjektai1;
90  clear paimtiObjektai1;
91
92  % Ninit objektu priskyrimas i klasterius
```

```
% atstumu matricos generavimas
for i1 = 1:k1
    for i2 = 1:k1
        atstumuMatrica(i1,i2) = sum(pr(i1,:)-pr(i2,:)).^2;
    end;
end;
%maks1=max(atstumuMatrica);
%atst=max(maks1);
clear maks1;
objAts=0;
for i1=1:klsk
    maks1=max(atstumuMatrica);
    atst=max(maks1);
    for tir=1:matrica(i1,1,1)
        paimt=size(paimtiObjektai);
        j=matrica(i1,tir+2,1);
        for i2=1:k1
            if atstumuMatrica(j,i2) < atst && j~ =i2
                %patikrinimas ar nepaimtas
                pm=0;
                for i3=1:paimt(1,1)
                    if paimtiObjektai(i3,1) == i2
                        pm=1;
                    end;
                end;
                if pm == 0
                    objAts(2)=atstumuMatrica(j,i2);
                    objAts(1)=i2;
                    atst = atstumuMatrica(j,i2);
                end;
            end;
        end;

    end;
        matrica(i1,1,1)=matrica(i1,1,1)+1;
    matrica(i1,matrica(i1,1)+2,1)=objAts(1);
    paimtiObjektai(paimt(1,1)+1,1,1)=objAts(1);
    atst = objAts(2);
end;

% Laisvu objektu priskyrimas klasteriams
clear maks1;
clear objAts;
objAts=0;
for i1=1:k1
    objAts=0;
    maks1=max(atstumuMatrica);
```

```
140     atst=max(maks1)+1;
141     paimt=size(paimtiObjektai);
142     paimObj=0;
143     for i2=1:paimt(1,1)
144        if i1==paimtiObjektai(i2,1)
145           paimObj = 1;
146        end;
147     end;
148     if paimObj == 0
149        for i3=1:klsk
150           for i4=1:matrica(i3,1,1)
151              j=matrica(i3,i4+2,1);
152              if atstumuMatrica(j,i1) < atst && j~ =i1
153                 atst = atstumuMatrica(j,i1);
154                 objAts(1)=i1;
155                 objAts(2)=atstumuMatrica(j,i1);
156                 objAts(3)=i3;
157              end;
158           end;
159        end;
160        matrica(objAts(3),1,1)=matrica(objAts(3),1,1)+1;
161        matrica(objAts(3),matrica(objAts(3),1)+2,1)=objAts(1);
162        paimtiObjektai(paimt(1,1)+1,1)=objAts(1);
163        paimtiObjektai1 = sort(paimtiObjektai);
164        paimtiObjektai = paimtiObjektai1;
165        clear paimtiObjektai1;
166        atst = objAts(2);
167     end;
168  end;
169
170  %Blogiausio klasterio isrinkimas
171  suma=0;
172  klsk1=klsk;
173  while klsk1>1
174     klskid=klsk-klsk1+1;
175     klskid
176     for i1=1:klsk1
177        for i2=1:klsk1
178           suma(i1,i2,klsk1)=0;
179           if matrica(i1,1,klskid)>0 && matrica(i2,1,klskid) > 0
180           for i3=1:matrica(i1,1,klskid)
181              for i4=1:matrica(i2,1,klskid)
182                 j1=matrica(i1,i3+2,klskid);
183                 j2=matrica(i2,i4+2,klskid);
184                 j1;
185                 j2;
186                 suma(i1,i2,klsk1)=suma(i1,i2,klsk1)+GMatrica(j1,j2);
```

```
187                 end;
188             end;
189             end;
190             %suma(i1,i2,klsk1)=-log(suma(i1,i2,klsk1)/ ...
191             suma(i1,i2,klsk1)=(suma(i1,i2,klsk1)/ ...
192                 (2*matrica(i1,1)*matrica(i2,1)));
193         end;
194     end;
195     % tbd
196     maks1=min(suma(:,:,klsk1));
197     maks=min(maks1);
198     for i1=1:klsk1
199         for i2=1:klsk1
200             if suma(i1,i2,klsk1)>maks && i1~=i2
201                 obj(1)=i1;
202                 obj(2)=i2;
203                 maks=suma(i1,i2,klsk1);
204             end;
205         end;
206     end;
207     kl1=matrica(obj(1),1,klskid);
208     kl2=matrica(obj(2),1,klskid);
209     tarp=matrica(:,:,klskid);
210     for i3=1:kl2
211         kl1=tarp(i1,1);
212         tarp(i1,1)=tarp(i1,1)+1;
213         tarp(i1,kl1+3)=tarp(i2,i3+2);
214     end;
215     tarp(obj(2),:)=[];
216     ism= size(tarp);
217     matrica(1,ism(1,2),1)=0;
218     clear ism;
219     ind=size(tarp);
220     ind1=ind(1,1);
221     ind2=ind(1,2);
222     for i4=1:ind1
223         for i5=1:ind2
224             matrica(i4,i5, klskid+1)=tarp(i4,i5);
225         end;
226     end;
227     obj(1)
228     obj(2)
229     %matrica(:,:,klskid)
230     %matrica(:,:,klskid+1)
231
232     clear tarp;
233     klsk1=klsk1-1;
```

```
234 end;
235
236 vieta = ['renyi.txt'];
237 save (vieta, '-ascii', '-double', '-tabs', 'norm_d'); % rezultatu
238                                                     % saugojimas i faila
239
240 klasteriai = matrica; % grazinamos reiksmes priskyrimas
241 toc % laiko atskaitos pabaiga ir isvedimas i konsole
```

E. Kompaktinė plokštelė

Plokštelės turinys:

- Diplominio darbo tekstas (tex ir PDF formatu)
- Pristatymas
- Testavimui naudoti duomenų rinkiniai
- Realizuotų algoritmų kodai