TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

IDN70LT

Urmas Kaarn 132556IAPMM

ANDMEKAEVE REEGLISÜSTEEMI LÄBIMISE AJAKULU OPTIMEERIMINE

Magistritöö

Juhendaja: Rein Kuusik

Professor

Tallinn 2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Urmas Kaarn

09.05.2016

Annotatsioon

Töö eesmärgiks on leida tundmatut objekti kattev reegel võimalikult minimaalse ajaga. Aeg on sõltuv nii reeglitest endist kui ka tehnilisel tasemel toimuvast optimeerimisest.

Töös on uuritud atribuutide arvu ja ridade arvu mõju reeglisüsteemile . Samuti on käsitletud, kuidas mõjutab reeglite järjestamist reegli pikkus või sagedus.

Töö olulisemaiks tulemuseks on see, et kui reeglitest ehitada sageduste puu, siis võib kiiruste vahe olla suurem kui 10 korda. Tööd on võimalik kiirendada, kui näiteks reeglisüsteemi atribuutide järgi tükeldada ja neid blokke panna tööle üksteisega paralleelselt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 6 peatükki, 11 joonist, 12 tabelit.

Abstract

Data mining rule system passing time consumption optimization

Thesis target is to find methods what can make rule system to work faster. As the rule system can be part of bigger system where time consumption by every part of system is very critical.

Thesis deals with how effects if we sort rules with their length or frequency. Also looks how effects how many data rows we use in rule system generating and how affects result that how many attributes we are going to use in generating system. In generated rule system we can find system is going to check many times exactly same conditions even if we know that nothing changed between two checking times.

Main result were if we build our rule system as tree structure with frequency then there is possible to get rule system more than 10 times faster. Also speed improvements are possible if we split input table attributes into sub systems and put these sub systems work parallel way.

The thesis is in Estonian and contains 44 pages of text, 6 chapters, 11 figures, 12 tables.

Lühendite ja mõistete sõnastik

MONSA	*Monotone System Algorithm,* Monotoonse süsteemi algoritm
T	*True,* tõeväärtus tõde
F	*False***,** tõeväärtus väär

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust ja probleem

Töö eesmärgiks on tegeleda reeglisüsteemi optimeerimisega, et leida tundmatut objekti kattev reegel võimalikult minimaalse ajaga. Suure andmetabeli põhjal genereeritud reeglite hulk kipub minema üsna suureks ja arvuti protsessori seisukohalt ei ole hea, kui tingimuslauseid on liiga palju järjest. Kui genereeritav süsteem on üheks osaks üsna suure koormusega süsteemis, siis võib see osutuda liiga ressursi nõudlikuks protsessiks, et leida sobiv reegel, mis määraks väljundi.

1.2 Ülesande püstitus

Eesmärk on leida erinevaid ideid kasutades viis, kuidas vähendada reeglisüsteemi läbimisele kuluvat aega. Reeglisüsteemi optimeeritakse arvutiressursside (operatsioonisüsteem, protsessor) kasutuse seisukohast.

1.3 Metoodika

Algne reeglisüsteemi leitakse algoritmiperekonna MONSA abil, kuna algoritmiga on töö autoril kõige rohkem kogemusi. Põhimõtteliselt võib aluseks võtta suvalise meetodi reeglisüsteemi genereerimiseks, töös esitatav käsitlus toimib ka nende korral. Seejuures tuleb ainult lahendada reeglite esituse problemaatika, et loodud tarkvara edukalt töötaks. Reeglisüsteemi genereerimise algoritmide ühine omadus on, et nad ei vaatle oma tulemuse optimeeritust tehnilise tasandi suhtes. Optimeeritud reeglisüsteem aja kasutuse mõistes leitakse erinevate kriteeriumite alusel järjestades atribuute, millest võiks loota, et see viib optimaalsema tulemuse poole.

1.4 Ülevaade tööst

Teises peatükis antakse ülevaade töö teoreetilistest alustest.

Kolmandas peatükis esitatakse optimeerimise ideid ja analüüsitakse nende mõju reeglisüsteemile.

Neljandas peatükis antakse infot testkeskkonna ja realisatsiooni kohta.

Viiendas peatükis esitletakse võimalusi edasi arendamiseks.

2 Teoreetiline taust

Antud töös käsitletakse reeglisüsteemi optimeerimist tundmatu objekti jaoks seda katva reegli leidmiseks etteantud reeglite kogumist selle nurga alt, et see toimiks võimalikult kiiresti reaalses süsteemis. Mistõttu on oluline saada ülevaade lähtepunkti andvast reeglisüsteemi genereerivast algoritmist, aga samas uurida ka, mida suudab arvuti oma üldiste optimeerimistega saavutada, kasutades ära operatsiooni-süsteemi ja protsessori omadusi..

2.1 Reeglisüsteemi genereerimine

Reeglisüsteemi genereerimises on aluseks võetud MONSA perekonna algoritm MONSAMIN [1]. Algoritmi ülesanne on leida andmekogumist üles seaduspärasused, mis kirjeldavad uuritava väljundi seoseid sisenditega. Algoritmi tööks on vajalik, et andmed oleksid diskreetsed või siis, et neid on võimalik diskreetseks konverteerida.

2.1.1 MONSAMIN algoritm

MONSAMIN vaatleb andmekogumit kui tabelit, kus iga rida on andmeobjekt ja igas veerus on muutuja ehk atribuut. Igal objektil peab olema üks kindel atribuudi väärtus. Atribuuti, mille suhtes seoseid otsitakse, nimetatakse klassiks.

Algoritmi töö käigus arvutatakse andmekogumile sagedustabeleid, mille abil leitakse mustreid, mis kirjeldaksid mingit kindlat klassi. Selliseid mustreid nimetatakse reegliteks ning nende kogumit, mis klassi kohta leitakse, nimetatakse omakorda reeglite kogumikuks. Reegel kujutab endast sisulises mõttes tingimuslauset. Kui on atribuudid t1, t2 ja 3, siis reegel on näiteks:

t1=1 & t2=2 -> t3=3 Loetav kui t1=1 ja t2=2 siis t3=3.

Iga reegel võib omada ainult ühte kindlat klassi väärtust, kuid võib omada üht või rohkemat atribuudi ja selle väärtuse paari. Reegel katab ühte või rohkemat objekti. Reeglite leidmise MONSAMIN algoritmi kirjeldus on esitatud Joonis 1.

S0. t:=0 R_t :=∅
S1. Leia kõik sagedused ja juhul, kui tase ei ole 0, siis peegelda nullid alla
S2 Iga atribuudi väärtus A, mille sagedus on mõnes klassis C võrdne andmehulgaga
    Väljasta reegel $\{R_i\}$&A→C, i=0,...,t
    A←0
S3 Kui ei ole vabu atribuudi väärtusi, et teha väljavõtt siis
    Kui t=0 mine LOPP
    Vastasel juhul t:=t–1; mine S3
S4 Vali väikseima sagedusega atribuudi väärtus
    R_t←0; t:=t+1;
    Tee väljavõtt objektidest, mis sisaldavad R_t
    mine S1
LOPP. Reeglid on leitud

Joonis 1. MONSAMIN algoritm. [1]

2.1.2 MONSAMIN algoritmi näide ja selgitus

Näite andmestikus (Tabel 1) on valitud klassiks atribuut t3. Nagu eelpool kirjeldatud, reeglisüsteem üritab kirjeldada klassi teiste atribuutide kaudu.

Tabel 1. Näidisandmestik.

Atribuut t1	Atribuut t2	Atribuut t3
1	1	2
1	1	2
2	1	1
2	1	1
2	1	1
2	1	1
2	3	2
3	1	3
3	1	3
3	1	3

Algoritmis (Joonis 1) samm S1 järgi tuleb koostada sagedustabel (Tabel 2), kus märgitakse ära mitu korda atribuut-väärtuse paare andmestikus esines. Atribuut t3 antud tabelis ei kajastu, kuna tema on valitud klassiks, mille suhtes reegleid antud hetkel otsitakse.

Tabel 2. Näidisandmestiku üldine sagedustabel.

	Atribuut t1	Atribuut t2
Väärtus 1	2	9
Väärtus 2	5	0
Väärtus 3	3	1

Lisaks on vaja sagedustabeleid igale klassile eraldi (Tabel 3). Need tabelid on sarnased eelmisega, ainukeseks erinevuseks on see, et iga klassi puhul on arvestatud ainult neid algtabeli ridu, mis kuuluvad vastavasse klassi.

Samm 2 järgi leitakse sagedustabelist (Tabel 3) vastavad väärtused, kus üldises sagedustabelis sagedus on võrdne sama atribuudi väärtusega mõnes klassis. Antud näites leitakse järgnevad seosed:

- t1=1, sellest järeldub klass t3 = 2, kuna klass 2 ja üldises sagedustabelis on mõlemal juhul t1=1 kohal sageduseks märgitud 2.
- t2=3, sellest järeldub klass t3 = 2 , kuna klass 2 sagedustabelis ja üldises sagedustabelis on sagedus 1.
- t3=3 ning klass = 3.

Tabel 3. Näidisandmestiku sagedustabelid klassidele.

	Tabel klass t3=1		Tabel klass t3=2		Tabel klass t3=3	
	Atribuut t1	Atribuut t2	Atribuut t1	Atribuut t2	Atribuut t1	Atribuut t2
Väärtus 1	0	4	2	2	0	3
Väärtus 2	4	0	1	0	0	0
Väärtus 3	0	0	0	1	3	0

Järgmise sammuna tuleb teha samm S4, kuna alles on veel vabu atribuudi väärtuse paare. Väljavõtt tuleb teha vastavalt sagedustabelile (Tabel 2), võttes sealt kõige väiksema sageduse, milleks antud juhul on 5. Teised väärtused, mis sagedustabelis on väiksemad, on juba eelnevalt reeglites kasutatud ja seega need ei ole enam vabad atribuudi väärtuse paarid. Järgnevalt on esitatud t1=2 väljavõtu sagedustabel (Tabel 4). Kuna tase ei ole enam 0, siis tuleb nullid alla peegeldada, ehk antud juhul muutub sagedus 0-ks t2=3 korral.

Tabel 4. Sagedustabel väljavõtt atribuudi t1 järgi väärtuse 2 korral.

t1=2	Atribuut t2
Väärtus 1	4
Väärtus 2	0
Väärtus 3	~~1~~ 0

Lisaks tuleb taas koostada ka igale klassile oma sagedustabel (Tabel 5). Antud tabelist saame t1=2 ja t2=1 korral klass t3 = 1.

Tabel 5. Klasside sagedustabel sama väljavõtu korral

t1=2	Tabel klass t3=1	Tabel klass t3=2	Tabel klass t3=3
	Atribuut t2	Atribuut t2	Atribuut t2
Väärtus 1	4	0	0
Väärtus 2	0	0	0
Väärtus 3	0	~~1~~ 0	0

Selle järel tabelis vabu atribuudi väärtusi ei ole ning tagurdatakse tagasi ja nüüd on vaja väljavõtt teha ainsa vaba väärtuse t2=1 korral (Tabel 2). Selles väljavõtus paraku pole ühtegi vaba atribuudi-väärtuse paari, mistõttu toimub kohe tagasi tagurdamine ja nüüd pole enam ka esimese taseme sagedustabelis enam vabu atribuudi-väärtuse paare, ja järelikult on algoritm töö lõpetanud. Leitud reegliteks on:

- t1=1 -> t3=2
- t2=3 1 -> t3=2
- t1=1 -> t3=3
- t1=2 & t2=1 -> t3=1

2.2 Arvutiprotsessor

Kuna antud töös on oluline, et tulemus töötaks võimalikult kiiresti, siis järgnevalt uuritakse arvuti enda poolseid optimeerimisvõimalusi ja protsessori tööpõhimõtet.

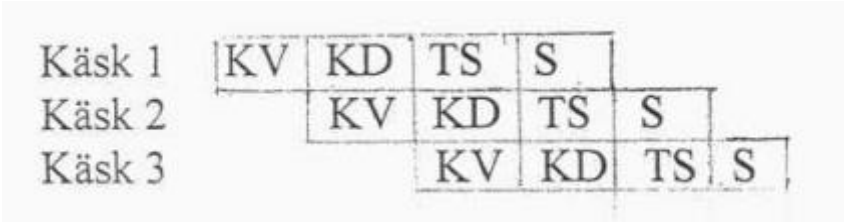
Programmid koosnevad käskude jadast. Protsessori käsustik koosneb kolme eri tüüpi käskudest: [2]

- Andmeedastuskäsud – operandide liigutamiseks registrite ja mälu vahel
- Andmetöötluskäsud – aritmeetika- ja loogikatehted
- Siirdekäsud – käsud, millega määratakse käskude täitmise järjekord

Iga käsu täitmine jaotub järgnevateks osadeks: [2]

- Käsu lugemine mälust
- Käsu dekodeerimine ja operandide aadresside dekodeerimine
- Operandide lugemine mälust registrisse
- Tehte sooritamine aritmeetika-loogikaüksuses
- Tulemuse salvestamine

Protsessori töö kiirendamiseks rakendatakse konveieri meetodit (Joonis 2). Kui esimene käsk on mälust loetud, siis samal ajal kui esimest dekodeeritakse, saab lugeda teise käsu mälust.



Joonis 2. Protsessori käskude konveier. Käsuvõtt(KV), käsu dekodeerimine(KD), Tehte sooritamine(TS), salvestamine(S). [2]

Kui kahe esimese käsutüübi puhul üldjuhul optimeerimisega probleeme ei ole, sest tavaliselt saab käskude järjestusega tagada, et kui üks käsk vajab teise käsu tulemust, siis järjekorda muutes tekib olukord, et eelneva käsu tulemus on selleks ajaks olemas ja seisakut ei teki. Küll aga tekitavad probleeme siirdekäsud (Joonis 3), kuna siirdekäsu

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
Aste 1	K1v	K2v	K3v	K4v	K5v	K6v	K1v	K2v	K3v	K4v
Aste 2		K1d	K2d	K3d	K4d	K5d		K1d	K2d	K3d
Aste 3			K1ts	K2ts	K3ts	K4ts			K1ts	K2ts
Aste 4				K1s	K2s	K3s	K4s			K1s

Joonis 3. Siirdekäsu mõju käskude konveierile [7]. Käsk1(K1), käsuvõtt(V), käsu dekodeerimine(D), Tehte sooritamine(TS), salvestamine(S). [2]

korral sõltub tulemusest, et milline on siirdekäsule järgnev käsk. Siirdekäskude puhul püütakse ennustamise abiga sisse lugeda konveierile õige käsk.

2.2.1 Siirdekäskude ennustamine

Igor Ostrovsky [3] on uurinud siirdekäskude ennustamist mõningate lihtsate tingimuslausetega, kus ta kasutas järgnevat koodijuppi (Joonis 4):

```
for (int i = 0; i < max; i++)
      if (<tingimus>)
            sum++;
```

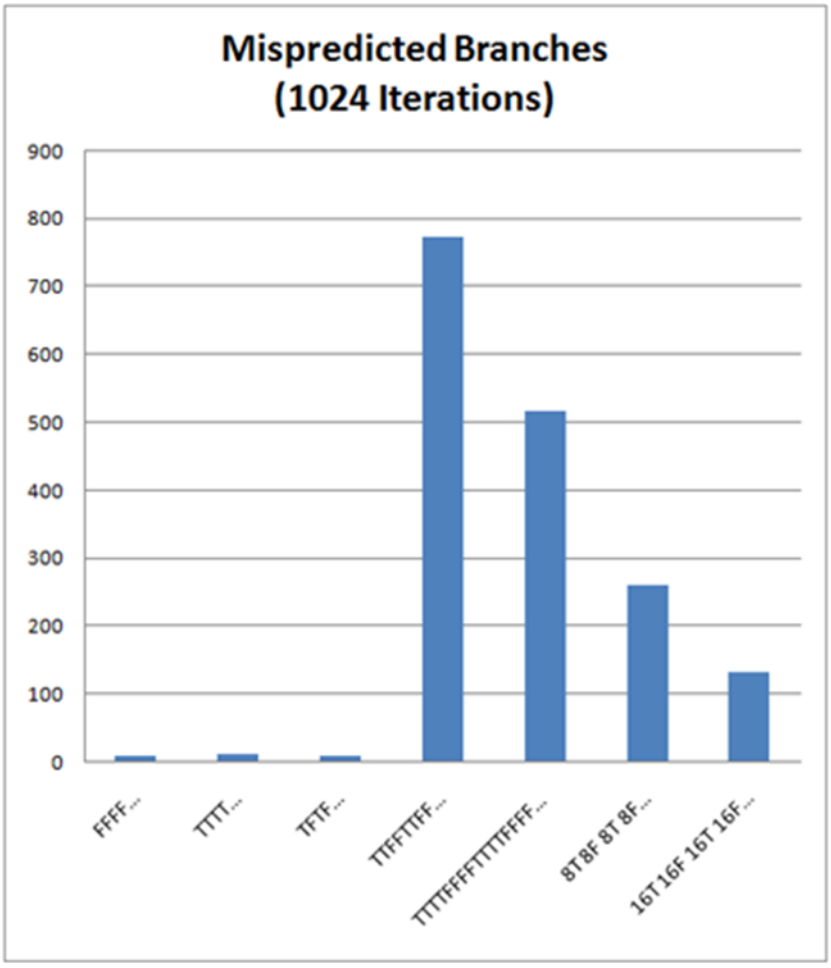
Joonis 4. Siirdekäskude hindamiskood [3]

Koodis kasutatud tingimusi võib näha tabelist (Tabel 6).

Tabel 6. Tingimuslausete ja aja seos [3].

Tingimus	**Muster**	**Aeg (ms)**
(i & 0x80000000) == 0	T lõputult	322
(i & 0xffffffff) == 0	F lõputult	276
(i & 1) == 0	TF vaheldumisi	760
(i & 3) == 0	TFFFTFFF…	513
(i & 2) == 0	TTFFTTFF…	1675
(i & 4) == 0	TTTTFFFFTTTTFFFF…	1275
(i & 8) == 0	8T 8F 8T 8F …	752
(i & 16) == 0	16T 16F 16T 16F …	490

Diagrammilt (Joonis 5) võib näha, et ennustamise juures ei valmista raskusi, kui siirdekäsk andis kogu aeg sama tulemuse või siirduti kordamööda kumbagi harru. Keerukamate mustrite puhul on näha, et ennustamine enam nii hästi ei läinud, kuid samas on näha, et mida rohkem sama tulemus järjest tuleb, siis seda täpsemaks on muutunud



Joonis 5. Vääralt ennustatud siirdekäsud(1024 iteratsiooni) [3].

ennustamine. Graafikult võib näha, et mustri TTFF puhul on ennustamine olnud üsna halb, sest palju parema tulemuse oleks saanud, kui ennustada, et kogu aeg tuleb T. Millest võiks järeldada, et tingimuslausete puhul tuleks püüda vältida olukorda, kus tingimuse tulemuseks on 50% T ja 50% F. Pigem tuleks eelistada 50%-st märksa väiksemaid või suuremaid protsente. Väiksemad protsendid on head selles suhtes, et kui näiteks T tõenäosus on 25%, siis tegelikult on see ju sama, mis F tõenäosus 75%.

3 Reeglisüsteemi optimeerimine

Reeglisüsteem on oma olemuselt üks suur tingimuslausete süsteem, ja nagu eelnevast näha, võib tingimuslausetest tingitud siirdekäsud olla arvuti jaoks üsna suur probleem. Kuidas neid täita võimalikult efektiivselt, kuna meil pole ühtki alust eeldada, et kaks järjestikust päringut reeglisüsteemile oleksid omavahel kuidagi seotud. Näiteks omada sarnaseid atribuutide väärtuseid. Seetõttu ei saa eeldada, et iseenesest võiks tekkida olukord, kus järjestikused tingimuslausete töötlemised sarnaneksid mõnele mustrile, mida arvuti suudab efektiivselt ennustada (vt Joonis 3 lk 16).

3.1 Reeglisüsteemi väljund ja tema minimeerimine

MONSAMIN suudab genereerida tohutu hulga reegleid, mille hulgas on suurel hulgal reegleid, mis katavad üksteist. Antud näites on reeglisüsteem genereeritud 1984 aasta USA kongressi hääletustulemuste alusel, kus sisendatribuutideks on poolt või vastu hääletamine erinevatele poliitilistele otsustele ning klassiks on, et kas tegu on demokraadi või vabariiklasega. MONSAMINi poolt genereeritud reeglite arv on 57 550, mis on ilmselgelt liiga suur hulk, et seda käsitsi analüüsida ja leida sealt üleliigseid reegleid.

Reeglite hulga vähendamiseks kasutatakse algoritmi (Joonis 6), mille tulemusena jäi järgi 28 reeglit. Algoritmi eesmärk on leida minimaalne kate.

```
S1 Otsi objekte, mis oleks kaetud ainult ühe reegli poolt.
      Lisa leitud reeglid minimeeritud reeglite nimekirja,
S2 Otsi vanast reeglite listist üles kõik reeglid, mis katavad ainult neid
objekte, mis on juba uues reeglite nimekirjas olevate reeglite poolt kaetud.
S3 leia vanade reeglite hulgast enim katmata objekte kattev reegel.
      Lisa see reegel minimeeritud reeglite hulka
      Kui on reegleid vanas nimekirjas Mine S1
```

Joonis 6. Reeglite hulga vähendamise algoritm.

Järgnevalt näeme (Tabel 7) reeglisüsteemi sellisena nagu MONSAMIN ja tema järel integreeritud minimeerimisalgoritm selle järjestanud on.

Tabel 7. Reeglisüsteemi minimeeritud väljund.

	Reegel	**Objektide arv**
1	t2=1 & t3=2->t16=1	219
2	t3=0 & t14=0 & t10=1 & t2=0->t16=0	104
3	t4=0 & t6=1 & t3=0->t16=0	32
4	t3=2 & t10=2->t16=1	113
5	t3=0 & t8=0 & t9=0 & t2=0->t16=0	67
6	t10=1 & t15=0 & t9=0 & t3=0->t16=0	48
7	t11=1 & t5=0 & t8=1->t16=1	50
8	t2=1 & t6=0 & t10=2->t16=1	29
9	t2=0 & t11=2->t16=0	10
10	t12=0 & t7=0 & t5=0 & t3=0 & t6=0 & t15=0 & t9=1 & t2=0->t16=0	22
11	t3=2 & t12=0->t16=1	58
12	t8=2 & t2=2 & t9=2->t16=0	2
13	t0=2 & t2=2->t16=1	2
14	t5=0 & t12=2 & t15=2->t16=1	2
15	t3=0 & t11=0 & t10=2 & t0=2 & t15=2->t16=0	2
16	t13=0 & t1=1 & t12=2 & t2=0 & t5=1->t16=0	5
17	t8=1 & t13=2 & t9=2->t16=1	1
18	t0=1 & t11=0 & t2=2->t16=0	1
19	t4=0 & t3=1->t16=1	2
20	t11=1 & t7=0 & t12=1->t16=1	1
21	t3=0 & t14=1 & t15=2->t16=0	2
22	t0=0 & t1=1 & t10=2 & t15=2->t16=1	2
23	t12=0 & t1=0 & t9=1 & t0=0 & t10=2 & t15=2->t16=0	1
24	t1=0 & t3=0 & t0=2 & t2=0 & t15=1->t16=1	1
25	t2=1 & t7=0 & t15=1->t16=1	8
26	t2=0 & t5=1 & t3=0 & t10=2->t16=1	1
27	t2=1 & t5=1 & t3=0 & t10=2->t16=0	1
28	t13=1 & t2=0 & t5=1->t16=1	4

Eelnevast tabelist võib näha, et reeglites kontrollitakse korduvalt samu atribuut-väärtus paare. Tabelist (Tabel 8) on näha, et näiteks t3=0 esineb reeglisüsteemis 10 korda, mis tähendab, et arvuti kontrollib 10 korda täpselt ühte ja sedasama ning iga kontrolliga kaasneb omakorda risk, et siirdekäsk ennustati valesti, ja lisaks siirdekäsk tuleb täita, mistõttu on ka iga siirdekäsk ise ajakulu vähemalt 1 protsessori takt.

Tabel 8. Atribuut- väärtus paaride esinemissagedused reeglisüsteemis.

Tingimus	**Sagedus**	**Tingimus**	**Sagedus**	**Tingimus**	**Sagedus**	**Tingimus**	**Sagedus**
t3=0	10	t5=0	3	t15=1	2	t13=0	1
t2=0	8	t7=0	3	t4=0	2	t13=1	1
t10=2	7	t0=0	2	t6=0	2	t13=2	1
t15=2	5	t1=0	2	t8=1	2	t14=0	1
t2=1	4	t1=1	2	t9=0	2	t14=1	1
t5=1	4	t10=1	2	t9=1	2	t3=1	1
t0=2	3	t11=0	2	t9=2	2	t6=1	1
t12=0	3	t11=1	2	t0=1	1	t8=0	1
t2=2	3	t12=2	2	t11=2	1	t8=2	1
t3=2	3	t15=0	2	t12=1	1		

3.2 Erinevad ideed optimeerimiseks

Eelmises punktis olevast väljundist on näha, et esimeste reeglite hulka on sattunud reegleid, mis katavad palju erinevaid objekte. Samas on ka näha, et nende hulka on sattunud reegel, mis katab vaid 32 objekti, olles suurusjärgu võrra väiksema esinemissagedusega kui talle eelnev ja järgnev reegel.

3.2.1 Reeglite järjestus pikkuse järgi

Kuna protsessori jaoks on reeglis iga atribuudi väärtuse kontroll omaette siirdekäsk, siis on esimeseks katseks reeglite sorteerimine pikkuse järgi. Kui kaks reeglit on ühepikkused, siis eelistada reeglit, mis katab rohkem objekte.

Idee mõte on järgnev: kuna lühemate reeglite kontrollimine on kiirem, siis võib kaasneda ajalist võitu, kui sooritada lühemate reeglite kontroll varem (vt. Hääletuse andmestiku väljundit, Lisa 4).

3.2.2 Reeglite järjestamine sageduse alusel

Reeglisüsteemi väljundist (Tabel 7) on näha, et suurema sagedusega reeglid on küll eespool, aga siiski mitte sorteeritult. Reeglisüsteem toimib loogikal, et kui leitakse kirjeldav reegel, siis väljutakse reeglisüsteemist.

Seega idee on sageduse järgi järjestada, et siis ehk enamikel juhtudel leitakse sobiv reegel natukene kiiremini üles (vt. Hääletuse andmestiku väljundit, Lisa 3).

3.2.3 Reeglite ühendamine ja järjestamine sageduse alusel

Reeglites esineb mõnda atribuut-väärtuse paari korduvalt (vt Tabel 8, lk 21). Idee on esimesena kontrollida sellist atribuut-väärtuse paari, mis esines kõige rohkemates objektides. Järgmisena leida selle atribuudi väärtuse paariga seotud reeglite hulgast omakorda kõige sagedasem atribuudi väärtuse paar. Sageduse puu koostamiseks kasutatud algoritm on esitatud Joonisel 7.

```
OptimeeriSageduseAlusel(reeglid){
  Koosta kõikides reeglites leiduvate atribuudi väärtuste list. Iga atribuudi
väärtuse kohta on teada esinemissagedus kõigi reeglite peale ning reeglid,
millega ta on seotud.
  reegliSüsteem =""
  while(sagedaseim atribuudiväärtus listist != NULL){
    if(atribuudi väärtus ei ole kaetud){
      reegliSüsteem += "if(atribuut = väärtus)"
      märgi atribuudiväärtus kaetuks
      reegliSüsteem += OptimeeriSageduseAlusel(atribuudiväärtuse reeglid)
      märgi kõik atribuudid, mis ei ole kaetud reeglitega seotud mittekaetuks
    }
  }
  if (katmata atribuudi väärtuseid ei olnud)
    märgi reegel kaetuks
    return "return väljundKlass"
  return reegliSüsteem
}
```

Joonis 7. Sageduse alusel järjestamise algoritm pseudokoodis.

Järgnevas näites (Joonis 8) on näha genereeritud tulemus hääletamise andmekogumi reeglite põhjal. Esimene *if* blokk kujutab endast kokku ühendatud reegleid 1, 4 ja 11 (vt Tabel 7, lk 20) ja pikemalt on väljund esitatud Lisas 2.

```
if(physicianFeeFreeze.equals("n")){
	if(adoptionOfTheBudgetResolution.equals("y")){
		return "democrat";
	}if(synfuels_corporation_cutback.equals("y")){
		return "democrat";
	}if(superfund_right_to_sue.equals("y")){
		return "democrat";}
	}if(physicianFeeFreeze.equals("y")){
		if(adoptionOfTheBudgetResolution.equals("n")){
			if(duty_free_exports.equals("n")){
				if(synfuels_corporation_cutback.equals("n")){
					return "republican";}
				}if(immigration.equals("y")){
					if(mx_missile.equals("n")){
						return "republican";}
	...
```

Joonis 8. genereeritud reeglisüsteemi algus hääletuste andmekogumi näitel.

3.3 Tulemuste võrdlus

Töös on kasutatud nelja erinevat andmekogumit [4-7], et hinnata optimeerimiste tulemusi ja analüüsida mõju. Tabelist (Tabel 9) on näha vastavate andmekogumite kirjeldused.

Tabel 9. Andmestike andmed.

	atribuute	**objekte**	**klasse**
Bridges [4]	13	108	4
Car [5]	6	1728	4
Nursery [6]	8	12960	5
Votes [7]	16	435	2

Järgnevalt (vt.

		Voting	***Nursery***	***Car***	***Bridges***
Minimeeritud reeglisüsteemi väljund	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	115	5082	1103	198

Pikkuse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	161	5960	1435	201
Sageduse järgi koostatud puu	Siirdekäskude arv	75	939	378	82
	Ajakulu(ms)	110	511	86	129
Sageduse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu(ms)	125	5960	1433	229

Tabel 10) andmekogumite puhul saadud tulemusi vaadates on võimalik märgata, et sageduste järgi järjestatud reeglid on saanud ajakulu suhtes kehvema tulemuse kui minimeeritud reeglisüsteemi väljund, kus reeglid olid suures osas sageduse alusel järjestatud, aga siiski osad reeglid olid sageduse mõistes vales kohas (vt Tabel 7, lk 20). Siinkohal saab tekkida küsimus, et miks? Kui vaadata joonist (Joonis 5, lk 18) võib eeldada, et need kõige halvemad ennustamise kohad reaalselt tekivad, kui T ja F tõenäosus läheneb 50%. Reeglisüsteemis suurema sagedusega reeglid ongi, väiksema sagedusega võrreldes, lähemal sellele 50%-le ja seega kehvemini ennustatavad, et kumba harru siirdekäsu puhul minema peab. Samas võib eeldada, et vastupidi keeratud järjekord annaks veelgi halvema tulemuse, kuna sel juhul käiks arvuti üle suurest hulgast reeglitest, mille puhul on ette teada, et enamiku puhul saab tulemuseks F ja iga üle käidud reegel on reaalne ajakulu.

Tabel 10. Erinevate ideede ajakulu ja siirdekäskude võrdlus.

		Voting	***Nursery***	***Car***	***Bridges***
Minimeeritud reeglisüsteemi väljund	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	115	5082	1103	198
Pikkuse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	161	5960	1435	201
Sageduse järgi koostatud puu	Siirdekäskude arv	75	939	378	82
	Ajakulu(ms)	110	511	86	129
Sageduse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu(ms)	125	5960	1433	229

Lisaks võib märgata, et pikkuse järgi sorteerides on tulemus väga sarnane sageduse järgi sorteeritule. See on seletatav asjaoluga, et lühemad reeglid kipuvad omama suuremat sagedust, mistõttu on probleem sisuliselt sama, mis sageduse järgi sorteeritul.

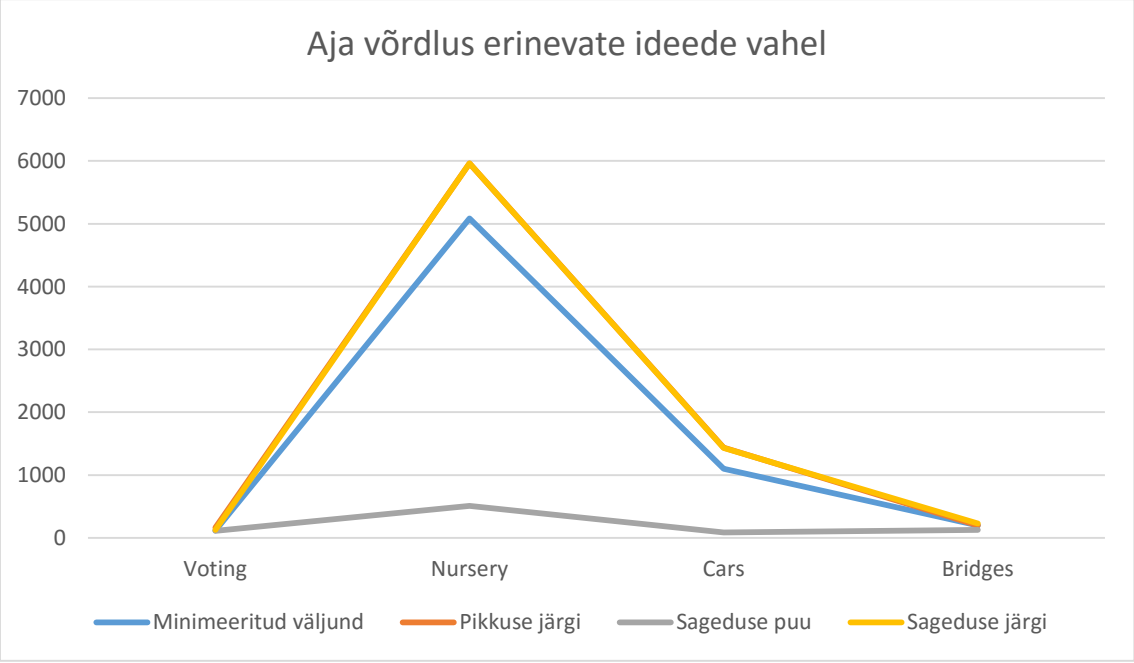
Kiiruse suhtes võib näha tabelist (

Tabel 10), et Sageduste puu on saavutanud kahe andmekogumi puhul väga suuri

		Voting	***Nursery***	***Car***	***Bridges***
Minimeeritud reeglisüsteemi väljund	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	115	5082	1103	198
Pikkuse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu (ms)	161	5960	1435	201
Sageduse järgi koostatud puu	Siirdekäskude arv	75	939	378	82
	Ajakulu(ms)	110	511	86	129
Sageduse järgi järjestus	Siirdekäskude arv	98	3249	1350	106
	Ajakulu(ms)	125	5960	1433	229

erinevusi. Sageduste puu korral on näha, et kui siirdekäske on üsna palju, siis on õnnestunud üsna palju ka reegleid omavahel kokku ühendada ja seetõttu vähendada korduvalt samade atribuut-väärtuste paaride kontrollimist.

Järgnevalt graafikult (Joonis 9) võib näha, et erinevate ideede mõju aja suhtes on säilitanud iga idee juures sama kuju.



Joonis 9. Ideede võrdlus ajakasutuse järgi.

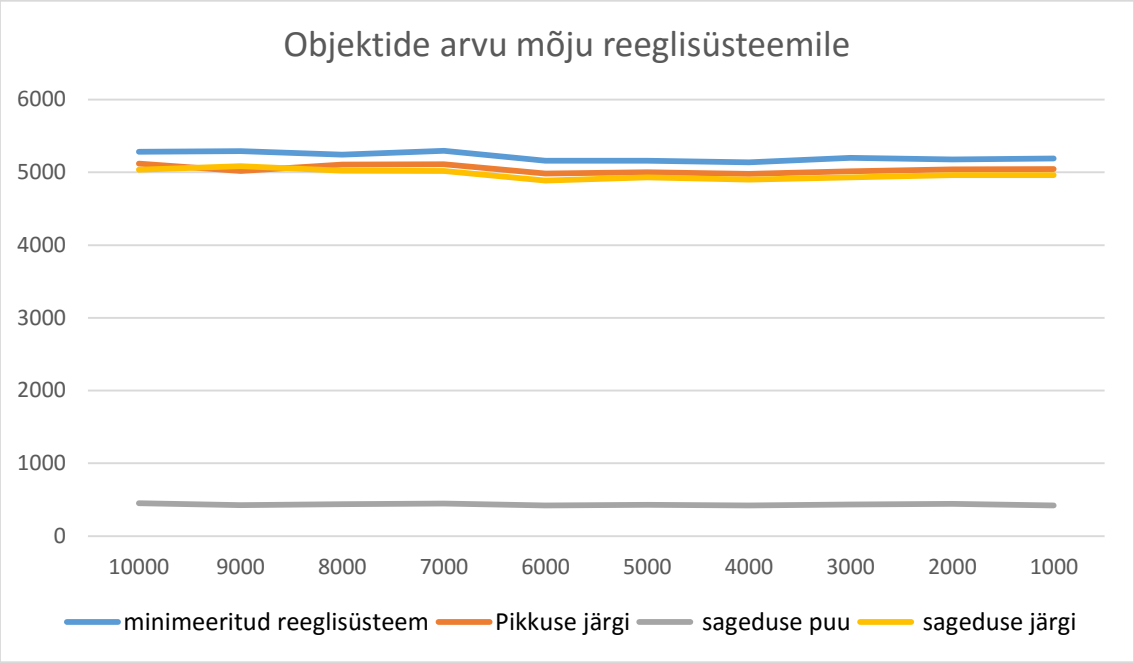
3.4 Ridade arvu mõju

Reeglisüsteemi genereerimisel võib väga kergelt tekkida küsimus, et kuidas mõjutab tabeli ridade (objektide) arv reeglisüsteemi läbimise kiirust. Loogilisena võib tunduda, et mida rohkem objekte kaasata, siis seda rohkem leiab reeglisüsteemi generaator ka detailsemaid seoseid klassi ja teiste atribuutide vahel ning seetõttu suureneb ka siirdekäskude arv ja sellest tulenevalt ka reeglisüsteemi ajakulu suureneb. Järgnevast (Tabel 11) võib näha, et siirdekäskude arvu ja objektide arvu vahel on seos tõepoolest olemas, aga aja suhtes nii kindel olla ei saa. Aluseks on võetud *Nursery* andmekogum [6].

Tabel 11. Objektide arvu mõju reeglisüsteemile.

Objektide arv	**Reeglite arv**	**Minimeeritud reeglisüsteemi siirdekäskude arv**	**Sageduse puu siirdekäskude arv**	**Minimeeritud reeglisüsteem (ms)**	**Pikkuse järgi (ms)**	**Sageduse puu (ms)**	**Sageduse järgi (ms)**
10000	446	2540	797	5282	5120	453	5038
9000	385	2133	679	5293	5019	424	5085
8000	331	1816	564	5242	5106	437	5023
7000	287	1543	511	5298	5113	449	5021
6000	248	1294	440	5161	4982	420	4885
5000	231	1192	400	5161	5001	431	4930
4000	215	1091	330	5137	4978	421	4899
3000	125	629	198	5198	5015	434	4930
2000	53	244	83	5179	5040	444	4961
1000	51	228	87	5189	5045	423	4964

Järgnevalt (Joonis 10) on näha, et graafik on kergelt kõver, aga objektide arv 10 000 kuni 1000 juures püsib kiirus sisuliselt muutumatuna. Arvesse tuleb siinkohal võtta, et mõõtmistulemusi mõjutavad ka arvuti muud protsessid, mistõttu on loomulik, et mõõtmistulemused saavad ka natuke kõikuda. Kui eelneva tabeli korral võis olla küsimus,



Joonis 10. Objektide arvu mõju reeglisüsteemile.

et kas äkki on väike kaldenurk nende kõikumiste vahel peidus, siis graafikult on näha, et erinevus on siiski nii väike, et tegemist on pigem mõõtevea piires toimuva varieerumisega. Ridade arvu suurendamisest võib näha, et reeglisüsteem saab reegleid juurde ja kvaliteet sellest tõuseb, aga samas ajalises mõttes kvaliteetsema tulemuse kasutamisele eriti juurde maksta ei ole vaja.

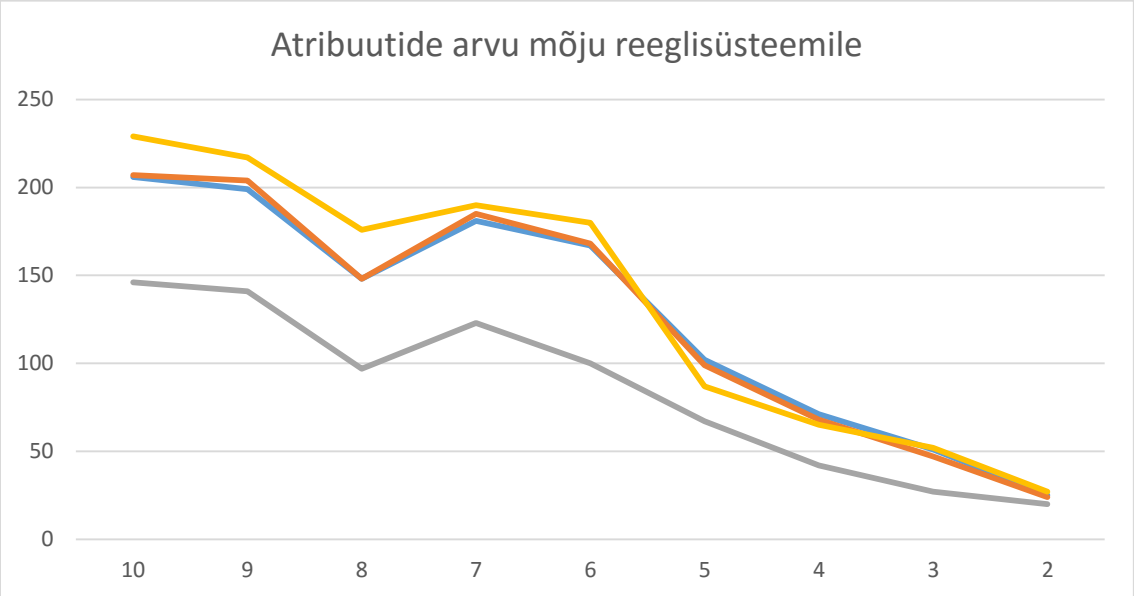
3.5 Atribuutide arvu mõju

Reeglisüsteemi genereerides tekib küsimus, et kui suur mõju on atribuutide arvu ja aja kasutuse vahel. Järgnevalt (Tabel 12) on näha erinevate atribuutide arvu mõju ajale. Tabelist võib näha, et ajaliselt töömaht eriti ei muutuks, kui näiteks 10 atribuudiga tabelit lammutades kaheks 5 atribuudiga tabeliks. Arvestada tuleb siinkohal, et 10 atribuudi kaheks lammutamisel tuleb liita 5 + 6 atribuuti, sest esimese viie atribuudi võtmisel tuleb koostada neile klass, mida nad kirjeldama hakkavad ja see klass liitub teise viiese atribuudi komplektile juba kuuenda atribuudina.

Tabel 12. Atribuutide arvu mõju reeglisüsteemile.

Objektide arv	Reeglite arv	Minimeeritud reeglisüsteemi siirdekäskude arv	Sageduse puu siirdekäskude arv	Minimeeritud reeglisüsteem (ms)	Pikkuse järgi (ms)	Sageduse puu (ms)	Sageduse järgi (ms)
10	37	106	82	206	207	146	229
9	37	107	79	199	204	141	217
8	32	96	71	148	148	97	176
7	30	83	63	181	185	123	190
6	24	68	51	167	168	100	180
5	13	30	25	102	99	67	87
4	11	25	19	71	68	42	65
3	7	13	10	51	47	27	52
2	3	6	5	25	24	20	27

Nagu tabelist võis märgata ja graafikult (Joonis 11) selgelt paistab, et atribuutide arvu 8 korral on üldisest loogikast toimunud mingisugune kõrvalekalle. Kui võrrelda reeglite arvu ja siirdekäskude arvu, siis pole sealt näha, et oleks toimunud järsk siirdekäskude vähenemine. Samas kui graafikult vaadata aega väärtuste 7 ja 9 korral ja nad ühendada mõttelise sirgega, siis graafiku kuju järgi on näha, et see sirge sobib sinna ja seetõttu võib eeldada, et antud juhul on tegu pigem erandjuhtumiga, kus juhuslikult generaatoril



Joonis 11. Atribuutide arvu mõju reeglisüsteemile.

õnnestus genereerida mõned suure katvusega reeglid, mistõttu õnnestub antud juhul otsitav klass reeglisüsteemiga kiiremini leida kui tavaliselt.

Kuigi katsetest võib näha, et ajalist võitu pole tükeldamisega eriti lootust saada, kui tükke rakendada jadamisi. Võib oletada, et kui tükke suuta panna tööle paralleelselt, siis on võimalik ajaliselt kokku hoida küll.

4 Reeglisüsteemi generaatori realisatsioon ja testkeskkond

4.1 Testkeskkond

- Protsessor Intel Core i3-4030U 1,90GHz
- 4GB DDR3 mälu
- Java versioon 1.8.0_45

Testjuhud on koostatud nii, et kontrollitakse 1000 erineva sisendiga 1000 korda järjest ehk siis reeglisüsteem saab korraga miljon päringut. Suur päringute arv on tehtud eesmärgiga, et taandada välja süsteemi käivitamise järel tehtud siirdekäskude valesti ennustamise mõju. Näiteks esimese tuhande päringuga on reaalne saada ajaline kulu neli korda suurem, kui märksa hilisemate päringutega. Reaalses süsteemis on arvutil kogunenud siirdekäskude kohta statistilist infot, mille abiga teostatakse siirdekäskude ennustamisi.

4.2 Realisatsiooni kommentaarid

Lähtetabeli sissevõtmisel realisatsioon kontrollib, et andmete hulgas ei oleks vasturääkivusi, et samade atribuutide väärtuste korral, kui esineb kaks erinevat klassi väärtust, et sel juhul määratakse klassiks see, mida esines enim selliste atribuudi väärtuste korral.

Realisatsioonis kasutatakse suhteliselt lihtsaid andmeobjekte nagu massiivid ja Javasse sisse ehitatud listid.

Realisatsiooni kood on saadaval aadressil: https://github.com/urmaska/magister

5 Ideed edasiseks arenduseks

Eelnevalt vaadeldud optimeerimine on eelkõige kasutatav ühe arvuti piires. Kuna arvuti ressurssidel on piirid, siis teatud suurusest reeglisüsteemi ei ole enam võimalik niipalju optimeerida, et seda oleks võimalik töös hoida, kasutades ainult ühe arvuti ressursse.

Keerukamates süsteemides tuleb arvestada lisaks siirdekäskudele veel ka seda, kui palju nad omavahel üle võrgu suhtlevad ja mismoodi on arvutite struktuur üles ehitatud.

5.1 Mitme arvuti kasutamine

Mitme arvuti kasutamise puhul tekib küsimus, et kas minna lihtsalt süsteemi dubleerimise teed, et hajutada koormust või ehitada mitme arvuti koostöös toimiv süsteem.

Esimese variandi korral saab suhteliselt lihtsalt koormust tõsta, aga samas ei pruugi sellega suuta piisavalt vähendada ajalist kulu.

Teises variandi puhul tuleb luua mitmetasandiline süsteem, kus on vaja välja mõelda loogika, mis suudaks süsteemi võrdseteks osadeks jaotada teises tasemes. Võrdseteks osadeks jagamine aga ei ole ilmselt kõige lihtsam ülesanne, sest lühikesed reeglid katavad palju objekte, mistõttu on nendel kõrge esinemissagedus, aga pikkade reeglite korral on rohkem siirdekäske. Siirdekäskude kaudu hakkab rolli mängima protsessori tasemel optimeerimine.

5.2 Süsteemi keerukus

Reeglisüsteemi loomise käigus on peamisi ülesandeid leida võimalikult optimaalne lahendus. Kui sisendiks on 20 erinevat atribuuti, aga reaalselt klassi määrab ära reegel, mis vajas ainult kahe atribuudi väärtust, siis tegelikult oli reeglisüsteemile ülejäänud 18 atribuudi väärtuse edastamine üleliigne tegevus. Osad atribuudid võivad näiteks pärineda andmebaasist.

Süsteem võib käivituda sel viisil, et klient täidab ära veebivormi ja täidetud andmete alusel võtab süsteem andmeid lisaks veel andmebaasist. Näiteks pank loob laenu andmise süsteemi ja üks atribuut küsib väärtuse andmebaasist ja andmebaas tagastab väärtuse „krooniline võlglane“. Antud juhul võib eeldada, et selliseid andmeid andmebaasis

otseselt ei ole ehk pigem on tegemist tuletatava tulemusega näiteks tehingute ajaloost. Samas tahetakse saada sellist laenu, kus panga ärireeglid ei näe probleemi, kui isegi tegemist on probleemse isikuga. Mistõttu tuleneks situatsioon, kus see mahukas andmebaasi päring osutuks ressursside raiskamiseks, sest tegelikult seda fakti ei vajatudki.

Eelnevast tuleneb, et kui reeglisüsteemil on vaja saada osade atribuutide väärtused andmebaasist või mõnest välisest süsteemist, siis oleks vajalik tekitada selline reeglisüsteemi ehitus, et vastavaid andmeid küsitaks ainult juhul, kui neid tõesti vaja läheb. Antud juhul on probleemiks, et kuidas reeglid selliste atribuutidega piisavalt efektiivselt eraldada teistest. Alternatiiv on kasutada optimeerimisel ennustamist nagu seda teeb protsessor.

Lisaks eksisteerib ka reaalseid süsteeme, kus reeglisüsteemi osad paiknevad geograafiliselt erinevates kohtades. Sellistes oludes tuleb ka arvestada, et reeglisüsteemi eri osade vaheline suhtlus tuleks viia võimalikult minimaalseks. [8]

6 Kokkuvõte

Töö peamiseks eesmärgiks oli reeglisüsteemi optimeerimine arvuti seisukohast vaadatuna, et objekti tuvastamine etteantud reeglisüsteemis toimiks võimalikult kiirelt. Selleks kasutati operatsioonisüsteemi ja protsessori tehnilisi omadusi. Probleemiks on, et kas saaks reegleid selleks paremini järjestada? Lisaks reeglisüsteemis sees on probleemiks, et mitmeid väärtusi kontrollitakse erinevate reeglite koosseisus üsna palju kordi. Näiteks käesolevas töös, kus reeglisüsteem koosnes 28-st reeglist ja üks atribuut-väärtuse paar esines kümnes reeglis.

Katsed näitasid, et kui sorteerida reegleid kui tervikuid sageduse või pikkuse alusel, siis võitu ei tulnud. Aga kui reegleid kokku ühendada samade atribuudi väärtuste kaudu, siis oli võimalik saavutada üsna suuri ajalisi võite. Näiteks andmestiku *Car* puhul ajaline võit oli rohkem kui 10-kordne.

Mida rohkem on atribuute, siis seda rohkem kulub aega, aga töös teostatud eksperimentidest on näha, et ajalise kulu poolest on võimalik reeglisüsteemi tükeldada ja üritada eri osad panna toimima paralleelselt.

Kui andmeid anda rohkem tabeli ridade mõttes ja seetõttu reeglisüsteemi generaator leiab detailsemad seosed, siis loogilisena tundub, et see võiks kaasa tuua suurema ajakulu. Katsete tulemusena aga selgus, et sellist seost siiski ei eksisteerinud.

Võimalike edasiarenduste osas tasuks liikuda selles suunas, et kasutada samaaegselt mitut arvutit.

Lisaks on probleemiks ka see, et suhteliselt paljusid objekte katavad üsna lühikesed reeglid. Lühikesed reeglid tähendavad seda, et suur osa esitatud andmetest olid tegelikult sobiva klassi leidmise seisukohalt üleliigsed. Reaalses süsteemis ei oleks mõttekas kõiki andmed kohe võtta, osa andmeid võiks kaasata andmebaasist mahuka päringu tulemusel või siis välisest süsteemist. Variantidena saaks olla, et kas saab kuidagi neid atribuute, mis enim aega võtavad, reeglisüsteemis tahapoole lükata või siis võtta eeskuju protsessori tööpõhimõttest ja luua vastav ennustamise algoritm.

Kasutatud kirjandus

[1] P. Roosmann, L. Võhandu, R. Kuusik, T. Treier ja G. Lind, „Monotone Systems approach in Inductive,“ *International Journal of Applied Mathematics and Informatics,* kd. 2, nr 2, pp. 47-56, 2008.

[2] H. Mägi, „Protsessorid,“ TTÜ, Tallinn, 2007.

[3] I. Ostrovsky, 15 5 2010. [Võrgumaterjal]. Available: http://igoro.com/archive/fast-and-slow-if-statements-branch-prediction-in-modern-processors/. [Kasutatud 18 04 2016].

[4] Y. R. &. S. J. Fenves, „Pittsburgh bridges,“ 1 August 1990. [Võrgumaterjal]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/bridges/. [Kasutatud 30 Aprill 2016].

[5] M. Bohanec, „Car Evaluation Database,“ [Võrgumaterjal]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/car/. [Kasutatud 30 Aprill 2016].

[6] V. Rajkovic, „Nursery Database,“ Juuni 1997. [Võrgumaterjal]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/nursery/. [Kasutatud 30 Aprill 2016].

[7] Congressional Quarterly Almanac, „1984 United States Congressional Voting Records Database,“ 27 Aprill 1987. [Võrgumaterjal]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/. [Kasutatud 20 Detsember 2015].

[8] M. Z. Ashrafi, D. Taniar ja K. Smith, „ODAM: An Optimized Distributed Association,“ [Võrgumaterjal]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1285877. [Kasutatud 20 Märts 2016].

Lisa 1 – Minimeeritud reeglisüsteemi väljund andmekogumile Voting

```
	public static String reegliSystem(String handicappedInfants, String
waterProjectCostSharing, String adoptionOfTheBudgetResolution, String
physicianFeeFreeze, String elSalvadorAid, String religiousGroupsInSchools,
String antiSatelliteTestBan, String aidToNicaraguan_contras, String
mx_missile, String immigration, String synfuels_corporation_cutback, String
education_spending, String superfund_right_to_sue, String crime, String
duty_free_exports, String export_administration_act_south_africa){
	if(adoptionOfTheBudgetResolution.equals("y") &&
physicianFeeFreeze.equals("n"))
		return "democrat";
	else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("n") && synfuels_corporation_cutback.equals("n") &&
adoptionOfTheBudgetResolution.equals("n"))
		return "republican";
	else if(elSalvadorAid.equals("y") && antiSatelliteTestBan.equals("y")
&& physicianFeeFreeze.equals("y"))
		return "republican";
	else if(physicianFeeFreeze.equals("n") &&
synfuels_corporation_cutback.equals("y"))
		return "democrat";
	else if(physicianFeeFreeze.equals("y") && mx_missile.equals("n") &&
immigration.equals("y") && adoptionOfTheBudgetResolution.equals("n"))
		return "republican";
	else if(synfuels_corporation_cutback.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("y")
&& physicianFeeFreeze.equals("y"))
		return "republican";
	else if(education_spending.equals("n") &&
religiousGroupsInSchools.equals("y") && mx_missile.equals("y"))
		return "democrat";
	else if(adoptionOfTheBudgetResolution.equals("y") &&
antiSatelliteTestBan.equals("n") && synfuels_corporation_cutback.equals("y"))
		return "democrat";
	else if(adoptionOfTheBudgetResolution.equals("n") &&
education_spending.equals("?"))
		return "republican";
	else if(superfund_right_to_sue.equals("y") &&
aidToNicaraguan_contras.equals("n") && religiousGroupsInSchools.equals("y")
&& physicianFeeFreeze.equals("y") && antiSatelliteTestBan.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("n")
&& adoptionOfTheBudgetResolution.equals("n"))
		return "republican";
	else if(physicianFeeFreeze.equals("n") &&
superfund_right_to_sue.equals("y"))
		return "democrat";
	else if(mx_missile.equals("?") &&
adoptionOfTheBudgetResolution.equals("?") && immigration.equals("?"))
		return "republican";
```

```
      else if(handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("?"))
              return "democrat";
      else if(religiousGroupsInSchools.equals("y") &&
superfund_right_to_sue.equals("n") &&
export_administration_act_south_africa.equals("n"))
              return "democrat";
      else if(physicianFeeFreeze.equals("y") &&
education_spending.equals("y") && synfuels_corporation_cutback.equals("y") &&
handicappedInfants.equals("y") &&
export_administration_act_south_africa.equals("n"))
              return "republican";
      else if(crime.equals("y") && waterProjectCostSharing.equals("n") &&
superfund_right_to_sue.equals("n") &&
adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n"))
              return "republican";
      else if(mx_missile.equals("y") && crime.equals("?") &&
immigration.equals("?"))
              return "democrat";
      else if(handicappedInfants.equals("?") &&
education_spending.equals("y") && adoptionOfTheBudgetResolution.equals("?"))
              return "republican";
      else if(elSalvadorAid.equals("y") && physicianFeeFreeze.equals("?"))
              return "democrat";
      else if(education_spending.equals("n") &&
aidToNicaraguan_contras.equals("n") && superfund_right_to_sue.equals("?"))
              return "democrat";
      else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("y") &&
export_administration_act_south_africa.equals("n"))
              return "republican";
      else if(handicappedInfants.equals("n") &&
waterProjectCostSharing.equals("n") &&
synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
              return "democrat";
      else if(superfund_right_to_sue.equals("y") &&
waterProjectCostSharing.equals("y") && immigration.equals("n") &&
handicappedInfants.equals("n") && synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
              return "republican";
      else if(waterProjectCostSharing.equals("y") &&
physicianFeeFreeze.equals("y") && handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("n") &&
export_administration_act_south_africa.equals("?"))
              return "democrat";
      else if(adoptionOfTheBudgetResolution.equals("y") &&
aidToNicaraguan_contras.equals("n") &&
export_administration_act_south_africa.equals("?"))
              return "democrat";
      else if(adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
              return "democrat";
```

```
	else if(adoptionOfTheBudgetResolution.equals("y") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
		return "republican";
	else if(crime.equals("n") && adoptionOfTheBudgetResolution.equals("n")
&& religiousGroupsInSchools.equals("n"))
		return "democrat";
	return "";
	}
```

Lisa 2 – Sageduse puu andmekogumile Voting

```
	public static String sagedaseimTunnus(String handicappedInfants,
String waterProjectCostSharing, String adoptionOfTheBudgetResolution, String
physicianFeeFreeze, String elSalvadorAid, String religiousGroupsInSchools,
String antiSatelliteTestBan, String aidToNicaraguan_contras, String
mx_missile, String immigration, String synfuels_corporation_cutback, String
education_spending, String superfund_right_to_sue, String crime, String
duty_free_exports, String export_administration_act_south_africa){

if(physicianFeeFreeze.equals("n")){
	if(adoptionOfTheBudgetResolution.equals("y")){
		return "democrat";
	}if(synfuels_corporation_cutback.equals("y")){
		return "democrat";
	}if(superfund_right_to_sue.equals("y")){
		return "democrat";}
}if(physicianFeeFreeze.equals("y")){
	if(adoptionOfTheBudgetResolution.equals("n")){
		if(duty_free_exports.equals("n")){
			if(synfuels_corporation_cutback.equals("n")){
				return "republican";}
			}if(immigration.equals("y")){
				if(mx_missile.equals("n")){
					return "republican";}
			}if(antiSatelliteTestBan.equals("n")){
				if(superfund_right_to_sue.equals("y")){
					if(aidToNicaraguan_contras.equals("n")){

	if(religiousGroupsInSchools.equals("y")){

	if(export_administration_act_south_africa.equals("y")){

	if(immigration.equals("n")){

	return "republican";}}}}}

	}if(export_administration_act_south_africa.equals("?")){
					if(waterProjectCostSharing.equals("y")){
						if(handicappedInfants.equals("y")){
							return "democrat";}}
				}if(synfuels_corporation_cutback.equals("y")){
					if(religiousGroupsInSchools.equals("n")){
						return "democrat";}}
			}if(export_administration_act_south_africa.equals("y")){
				if(synfuels_corporation_cutback.equals("n")){
					if(immigration.equals("y")){
						return "republican";}}
			}if(elSalvadorAid.equals("y")){
				if(antiSatelliteTestBan.equals("y")){
```

```
                                    return "republican";}
                    }if(export_administration_act_south_africa.equals("n")){
                           if(education_spending.equals("y")){

if(synfuels_corporation_cutback.equals("y")){
                                    if(handicappedInfants.equals("y")){
                                                  return "republican";}}
                           }if(duty_free_exports.equals("y")){
                                    return "republican";}
                    }if(adoptionOfTheBudgetResolution.equals("y")){
                           if(religiousGroupsInSchools.equals("n")){

if(synfuels_corporation_cutback.equals("y")){
                                           return "republican";}}}
             }if(religiousGroupsInSchools.equals("y")){
                    if(education_spending.equals("n")){
                           if(mx_missile.equals("y")){
                                    return "democrat";}
                    }if(superfund_right_to_sue.equals("n")){

if(export_administration_act_south_africa.equals("n")){
                                    return "democrat";}}
             }if(antiSatelliteTestBan.equals("n")){
                    if(adoptionOfTheBudgetResolution.equals("y")){
                           if(synfuels_corporation_cutback.equals("y")){
                                    return "democrat";}}
             }if(religiousGroupsInSchools.equals("n")){
                    if(adoptionOfTheBudgetResolution.equals("n")){
                           if(crime.equals("y")){
                                  if(waterProjectCostSharing.equals("n")){

if(superfund_right_to_sue.equals("n")){
                                                  return "republican";}}
                           }if(crime.equals("n")){
                                    return "democrat";}}
             }if(education_spending.equals("?")){
                    if(adoptionOfTheBudgetResolution.equals("n")){
                                  return "republican";}
             }if(handicappedInfants.equals("y")){
                    if(adoptionOfTheBudgetResolution.equals("?")){
                                  return "democrat";}
             }if(adoptionOfTheBudgetResolution.equals("?")){
                    if(mx_missile.equals("?")){
                           if(immigration.equals("?")){
                                    return "republican";}
                    }if(handicappedInfants.equals("?")){
                           if(education_spending.equals("y")){
                                    return "republican";}}
             }if(handicappedInfants.equals("n")){
                    if(synfuels_corporation_cutback.equals("y")){
```

```
      if(export_administration_act_south_africa.equals("n")){
                              if(waterProjectCostSharing.equals("n")){
                                          return "democrat";
                              }if(immigration.equals("n")){

      if(superfund_right_to_sue.equals("y")){

      if(waterProjectCostSharing.equals("y")){
                                                                  return
"republican";}}}}}
            }if(physicianFeeFreeze.equals("?")){
                  if(elSalvadorAid.equals("y")){
                              return "democrat";}
            }if(crime.equals("?")){
                  if(mx_missile.equals("y")){
                        if(immigration.equals("?")){
                                    return "democrat";}}
            }if(superfund_right_to_sue.equals("?")){
                  if(education_spending.equals("n")){
                        if(aidToNicaraguan_contras.equals("n")){
                                    return "democrat";}}}
            if(adoptionOfTheBudgetResolution.equals("y")){
                  if(aidToNicaraguan_contras.equals("n")){

      if(export_administration_act_south_africa.equals("?")){
                                    return "democrat";}}}
            return "";
      }
```

Lisa 3 – Sageduse järgi järjestatud reeglid andmekogumile Voting

```
      public static String sagedus(String handicappedInfants, String
waterProjectCostSharing, String adoptionOfTheBudgetResolution, String
physicianFeeFreeze, String elSalvadorAid, String religiousGroupsInSchools,
String antiSatelliteTestBan, String aidToNicaraguan_contras, String
mx_missile, String immigration, String synfuels_corporation_cutback, String
education_spending, String superfund_right_to_sue, String crime, String
duty_free_exports, String export_administration_act_south_africa){

             if(adoptionOfTheBudgetResolution.equals("y") &&
physicianFeeFreeze.equals("n"))
                    return "democrat";
              else if(physicianFeeFreeze.equals("n") &&
synfuels_corporation_cutback.equals("y"))
                    return "democrat";
              else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("n") && synfuels_corporation_cutback.equals("n") &&
adoptionOfTheBudgetResolution.equals("n"))
                    return "republican";
              else if(physicianFeeFreeze.equals("y") &&
mx_missile.equals("n") && immigration.equals("y") &&
adoptionOfTheBudgetResolution.equals("n"))
                    return "republican";
              else if(physicianFeeFreeze.equals("n") &&
superfund_right_to_sue.equals("y"))
                    return "democrat";
              else if(education_spending.equals("n") &&
religiousGroupsInSchools.equals("y") && mx_missile.equals("y"))
                    return "democrat";
              else if(synfuels_corporation_cutback.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("y")
&& physicianFeeFreeze.equals("y"))
                    return "republican";
              else if(elSalvadorAid.equals("y") &&
antiSatelliteTestBan.equals("y") && physicianFeeFreeze.equals("y"))
                    return "republican";
              else if(adoptionOfTheBudgetResolution.equals("y") &&
antiSatelliteTestBan.equals("n") && synfuels_corporation_cutback.equals("y"))
                    return "democrat";
              else if(superfund_right_to_sue.equals("y") &&
aidToNicaraguan_contras.equals("n") && religiousGroupsInSchools.equals("y")
&& physicianFeeFreeze.equals("y") && antiSatelliteTestBan.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("n")
&& adoptionOfTheBudgetResolution.equals("n"))
                    return "republican";
              else if(adoptionOfTheBudgetResolution.equals("n") &&
education_spending.equals("?"))
                    return "republican";
```

```
            else if(adoptionOfTheBudgetResolution.equals("y") &&
aidToNicaraguan_contras.equals("n") &&
export_administration_act_south_africa.equals("?"))
                    return "democrat";
            else if(crime.equals("y") &&
waterProjectCostSharing.equals("n") && superfund_right_to_sue.equals("n") &&
adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n"))
                    return "republican";
            else if(crime.equals("n") &&
adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n"))
                    return "democrat";
            else if(elSalvadorAid.equals("y") &&
physicianFeeFreeze.equals("?"))
                    return "democrat";
            else if(religiousGroupsInSchools.equals("y") &&
superfund_right_to_sue.equals("n") &&
export_administration_act_south_africa.equals("n"))
                    return "democrat";
            else if(mx_missile.equals("?") &&
adoptionOfTheBudgetResolution.equals("?") && immigration.equals("?"))
                    return "republican";
            else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("y") &&
export_administration_act_south_africa.equals("n"))
                    return "republican";
            else if(handicappedInfants.equals("n") &&
waterProjectCostSharing.equals("n") &&
synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
                    return "democrat";
            else if(handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("?"))
                    return "democrat";
            else if(physicianFeeFreeze.equals("y") &&
education_spending.equals("y") && synfuels_corporation_cutback.equals("y") &&
handicappedInfants.equals("y") &&
export_administration_act_south_africa.equals("n"))
                    return "republican";
            else if(mx_missile.equals("y") && crime.equals("?") &&
immigration.equals("?"))
                    return "democrat";
            else if(adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
                    return "democrat";
            else if(education_spending.equals("n") &&
aidToNicaraguan_contras.equals("n") && superfund_right_to_sue.equals("?"))
                    return "democrat";
            else if(superfund_right_to_sue.equals("y") &&
waterProjectCostSharing.equals("y") && immigration.equals("n") &&
handicappedInfants.equals("n") && synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
                    return "republican";
```

```
              else if(waterProjectCostSharing.equals("y") &&
physicianFeeFreeze.equals("y") && handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("n") &&
export_administration_act_south_africa.equals("?"))
                     return "democrat";
              else if(handicappedInfants.equals("?") &&
education_spending.equals("y") && adoptionOfTheBudgetResolution.equals("?"))
                     return "republican";
              else if(adoptionOfTheBudgetResolution.equals("y") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
                     return "republican";
             return "";
       }
```

Lisa 4 – Pikkuse järgi järjestatud reeglid andmekogumile Voting

```
public static String pikkus(String handicappedInfants, String
waterProjectCostSharing, String adoptionOfTheBudgetResolution, String
physicianFeeFreeze, String elSalvadorAid, String religiousGroupsInSchools,
String antiSatelliteTestBan, String aidToNicaraguan_contras, String
mx_missile, String immigration, String synfuels_corporation_cutback, String
education_spending, String superfund_right_to_sue, String crime, String
duty_free_exports, String export_administration_act_south_africa){

      if(adoptionOfTheBudgetResolution.equals("y") &&
physicianFeeFreeze.equals("n"))
              return "democrat";
       else if(physicianFeeFreeze.equals("n") &&
synfuels_corporation_cutback.equals("y"))
              return "democrat";
       else if(physicianFeeFreeze.equals("n") &&
superfund_right_to_sue.equals("y"))
              return "democrat";
       else if(adoptionOfTheBudgetResolution.equals("n") &&
education_spending.equals("?"))
              return "republican";
       else if(elSalvadorAid.equals("y") && physicianFeeFreeze.equals("?"))
              return "democrat";
       else if(handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("?"))
              return "democrat";
       else if(education_spending.equals("n") &&
religiousGroupsInSchools.equals("y") && mx_missile.equals("y"))
              return "democrat";
       else if(elSalvadorAid.equals("y") && antiSatelliteTestBan.equals("y")
&& physicianFeeFreeze.equals("y"))
              return "republican";
       else if(adoptionOfTheBudgetResolution.equals("y") &&
antiSatelliteTestBan.equals("n") && synfuels_corporation_cutback.equals("y"))
              return "democrat";
       else if(adoptionOfTheBudgetResolution.equals("y") &&
aidToNicaraguan_contras.equals("n") &&
export_administration_act_south_africa.equals("?"))
              return "democrat";
       else if(crime.equals("n") &&
adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n"))
              return "democrat";
       else if(religiousGroupsInSchools.equals("y") &&
superfund_right_to_sue.equals("n") &&
export_administration_act_south_africa.equals("n"))
              return "democrat";
       else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("y") &&
export_administration_act_south_africa.equals("n"))
```

```
            return "republican";
        else if(mx_missile.equals("?") &&
adoptionOfTheBudgetResolution.equals("?") && immigration.equals("?"))
            return "republican";
        else if(mx_missile.equals("y") && crime.equals("?") &&
immigration.equals("?"))
            return "democrat";
        else if(handicappedInfants.equals("?") &&
education_spending.equals("y") && adoptionOfTheBudgetResolution.equals("?"))
            return "republican";
        else if(education_spending.equals("n") &&
aidToNicaraguan_contras.equals("n") && superfund_right_to_sue.equals("?"))
            return "democrat";
        else if(physicianFeeFreeze.equals("y") &&
duty_free_exports.equals("n") && synfuels_corporation_cutback.equals("n") &&
adoptionOfTheBudgetResolution.equals("n"))
            return "republican";
        else if(physicianFeeFreeze.equals("y") && mx_missile.equals("n") &&
immigration.equals("y") && adoptionOfTheBudgetResolution.equals("n"))
            return "republican";
        else if(synfuels_corporation_cutback.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("y")
&& physicianFeeFreeze.equals("y"))
            return "republican";
        else if(handicappedInfants.equals("n") &&
waterProjectCostSharing.equals("n") &&
synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
            return "democrat";
        else if(adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
            return "democrat";
        else if(adoptionOfTheBudgetResolution.equals("y") &&
religiousGroupsInSchools.equals("n") && physicianFeeFreeze.equals("y") &&
synfuels_corporation_cutback.equals("y"))
            return "republican";
        else if(crime.equals("y") && waterProjectCostSharing.equals("n") &&
superfund_right_to_sue.equals("n") &&
adoptionOfTheBudgetResolution.equals("n") &&
religiousGroupsInSchools.equals("n"))
            return "republican";
        else if(physicianFeeFreeze.equals("y") &&
education_spending.equals("y") && synfuels_corporation_cutback.equals("y") &&
handicappedInfants.equals("y") &&
export_administration_act_south_africa.equals("n"))
            return "republican";
        else if(waterProjectCostSharing.equals("y") &&
physicianFeeFreeze.equals("y") && handicappedInfants.equals("y") &&
adoptionOfTheBudgetResolution.equals("n") &&
export_administration_act_south_africa.equals("?"))
            return "democrat";
```

```
        else if(superfund_right_to_sue.equals("y") &&
waterProjectCostSharing.equals("y") && immigration.equals("n") &&
handicappedInfants.equals("n") && synfuels_corporation_cutback.equals("y") &&
export_administration_act_south_africa.equals("n"))
                return "republican";
        else if(superfund_right_to_sue.equals("y") &&
aidToNicaraguan_contras.equals("n") && religiousGroupsInSchools.equals("y")
&& physicianFeeFreeze.equals("y") && antiSatelliteTestBan.equals("n") &&
export_administration_act_south_africa.equals("y") && immigration.equals("n")
&& adoptionOfTheBudgetResolution.equals("n"))
                return "republican";
       return "";
       }
}
```