T A R T U Ü L I K O O L

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Tarkvarasüsteemide õppetool

Informaatika eriala

Jüri Reimand

Geenide funktsionaalsuse ennustamine ontoloogiate abil

Bakalaureusetöö (10AP)

Juhendaja: Jaak Vilo, PhD

Autor: ..... “.....” mai 2005

Juhendaja: ..... “.....” mai 2005

Õppetooli juhataja: ..... “.....” 2005

TARTU 2005

Sisukord

Lühendid

- **BP** - Biological Process
- **CC** - Cellular Component
- **DNA** - Deoxyribonucleic Acid
- **GO** - Gene Ontologies
- **GOST** - Gene Ontologies Statistics
- **HTTP** - HyperText Transfer Protocol
- **MF** - Molecular Function
- **ND** - No Biological Data Available
- **ORF** - Open Reading Frame
- **PERL** - Practical Extraction and Report Language
- **PHP** - PHP: Hypertext Preprocessor
- **SGD** - *Saccharomyces cerevisiae* Genome Database
- **SPEXS** - Sequence Pattern EXhaustive Search
- **SQL** - Structured Query language

1 Sissejuhatus

Andmekaeveks (ingl. k. *Knowledge Discovery in Databases, Data Mining*) nimetatakse mittetriviaalset protsessi, mille käigus eraldatakse suurtest andmekogudest senitundmatut, kuid potentsiaalselt huvitavat informatsiooni [FPSM92]. Andmekaeve on uus ja arenev uurimisvaldkond, mis kombineerib muuhulgas tekstialgoritme, statistikat ja tehisintellekti tehnikaid.

Andmekaeve oluliseks rakendusvaldkonnaks on geneetika. Seoses viimastel aastatel kiirenenud genoomide kaardistamisega on tekkinud vajadus kiirete arvutusmeetodite järgi, mille abil oleks võimalik mahukatest andmekogudest seoseid leida ning leidude olulisust hinnata.

Geeniontoloogiate Konsortsium[1] (ingl. k. *The Gene Ontology Consortium*)) on projekt, mille ülesandeks on geenide ja nende funktsioonidega seotud mõistete struktuuri arendamine [HJA+04]. Ontoloogiate eesmärk on geenide rollide ja toimimispiirkondade standardiseerimine organismist sõltumatult. Tänaseks on geeniontoloogia mõistetega seotud paljude organismide genoomibaasid, muuhulgas on annoteeritud ka inimese geene.

Geeniontoloogiates leiduv informatsioon pakub mitmeid huvitavaid ülesandepüstitusi. Näiteks võib tundmatuid elemente sisaldavale geenide hulgale ühiseid mõisteid otsides ennustada geenide omadusi. Kuna võimalikke geenide ja mõistete paare tekib palju, ei saa kõiki seoseid lugeda oluliseks. Antud küsimusele võib vastuseid otsida andmekaeve meetodite ja statistilise lähenemise abil.

Geeniontoloogiate kaevandamise saab taandada täisarvuhulkadel põhinevatele tehetele. Unikaalne täisarv sobib identifikaatoriks, seega on arvuhulk efektiivne viis geenide või mõistete säilitamiseks. Operatsioonid täisarvudega on realiseeritud madalal tasemel ning toimivad kiirelt. Suuri hulki on sageli kasulik säilitada ja töödelda bitivektoritena, kus iga bitt näitab vastava elemendi kuuluvust hulka.

[1] http://www.geneontology.org/

1.1 Ülesandepüstitus

Käesoleva bakalaureusetöö võib jagada neljaks osaks.

Esimeses osas uurin täisarvuhulkade realiseerimise võimalusi ja efektiivsust salvestusruumi ja hulgaoperatsioonide osas. Olen edasi arendanud varasemas valminud hulgatehete teeki LibSets ja binaarset salvestusformaati BinSets, mis kasutavad kiireid andmestruktuure ja tehteid. Samuti võrdlen LibSets teegil põhinevate programmide ja andmebaasi efektiivsust.

Töö teises osas uurin lähemalt geeniontoloogiate struktuuri ning olemust. Geeniontoloogiaid vaatlen pagaripärmi *Saccharomyces cerevisiae* genoomiga seotud annotatsioonide suhtes.

Töö kolmas osa on loomult kõige praktilisem. Tulemusena on valminud LibSets teegil põhinev geeniontoloogiate kaevandamise töövahend GOST, mille abil saab antud geenigrupile leida statistiliselt olulisi mõisteid. GOST on loodud nii käsureal kasutamiseks kui veebiliidesena. Samuti olen sooritanud mõningad arvutused pagaripärmi genoomil, mille abil on võimalik edasises statistilisi parameetreid paremini määrata.

Neljandas osas on sooritatud mahukaid arvutusi, mis leiavad pagaripärmi geenidest moodustatud maatriksil tähtsamaid geenigruppe. Tulemusena on saadud tuhatkond väga kõrgete headushinnangutega geenigruppi.

2 Hulgad ja realiseerimine

Hulk on matemaatikas üks lihtsamaid ja üldisemaid mõisteid. Mitteformaalselt võib hulgale viidata kui mingite objektide või mõistete kogule, millel erinevalt ülejäänud objektidest või mõistetest mingi ühine omadus. Hulkadest rääkides ei saa mööda universaalhulga ja tühihulga mõistetest. Universaalhulk U ehk domeen sisaldab kõiki elemente antud kontekstis ning tühihulk $\emptyset$ ei sisalda ühtegi elementi. Edasises näeme, et hulkade realiseerimises on domeeni mõistel tähtis roll.

Tabel 1 sisaldab mõningaid olulisi hulgamõisteid ja tehteid.

$x \in A$	Element x kuulub hulka A
$\|A\|$	hulga A võimsus ehk temas leiduvate elementide arv
$\emptyset$	Tühi hulk, $\|\emptyset\| = 0$
U	Universaalhulk, domeen
$den(A)$	$\|A\|/\|U\|$, hulga tihedus (hõredus)
$A \cup B$	$\{x \| x \in A \vee x \in B\}$, hulkade A ja B ühend
$A \cap B$	$\{x \| x \in A \wedge x \in B\}$, hulkade A ja B ühisosa
$A \setminus B$	$\{x \| x \in A \wedge x \notin B\}$, hulkade A ja B vahe
$A \triangle B$	$((A \cup B) \setminus (A \cap B))$, hulkade A ja B sümmeetriline vahe
A^-	$(U \setminus A)$, hulga A täiend
$A = B$	$\forall x \in U(x \in A \iff x \in B)$

Tabel 1: Hulkade mõisted ja operatsioonid

Hulgad võivad olla nii lõplikud kui lõpmatud. Lõpmatute hulkade korral võib veel rääkida loenduvatest ning mitteloenduvatest hulkadest. Käesoleva ülesandepüstituse raames tegeleme vaid lõplike hulkadega. Hulkade puhul on oluliseks omaduseks elementide unikaalsus, st. hulk ei tohi sisaldada korduvaid elemente. Üldiselt on hulgad võrdsed järjestuse täpsusega. Hulgaelemendid ei ole hulgas

teineteise suhtes järjekorras.

Programmeerimise ja andmekaeve seisukohast on väga loomulikud positiivsete täisarvude hulgad. Arv võib tähistada näiteks andmebaasikirje võtit, viita mingile mälupiirkonnale, positsiooni jadas või stringis. Säilitades hulkades vaid viitasid või tegeliku info võtmeid, hoiame kokku nii hulkade kui ka vastavate operatsioonide kiiruse ning mäluvajaduse pealt. Samuti saame tagada hulgaelementide unikaalsuse nõude täitmise, kuna kirje võti või positsioon jadas on juba definitsiooni poolest unikaalne.

Täisarvude kasutamise hulgaelemendina teeb lihtsaks asjaolu, et programmeerimiskeeltes on täisarv erinevalt kirjest või stringist primitiivne andmetüüp, seega on elementide võrdsuse-võrratuse testimine lihtne ja kiiresti teostatav.

Järgnevas tutvume hulgaoperatsioonide teegiga LibSets ning hulkade kasutamise võimalustega andmebaasiplatvormil.

2.1 Hulgaoperatsioonide teek LibSets

Eelneva töö käigus on valminud hulgateek LibSets [Rei04], kuhu on koondatud hulkade realiseerimiseks, säilitamiseks ning operatsioonideks vajalikud funktsioonid. Hulgateek põhineb suures osas SPEXS tarkvara [Vil02] lähtekoodil ning on kirjutatud keeles C. Hulgateek LibSets toetab hulkade realiseerimist kahel erineval viisil, täisarvude massiivide ning bitivektorite kujul. Võimalikud on teisendamised ning tehted erinevat tüüpi hulkade vahel, samuti hulkade statistilise olulisuse hindamine, hulkade salvestamine ning lugemine failist.

2.1.1 Täisarvude massiivid

Lihtsaimaks ja intuitiivseimaks hulga realiseerimise viisiks on täisarvude massiiv. Keeles C saavad massiivielemendid alguse indeksist `0`. Hulga `S i`-nda elemendi poole pöördumiseks kasutame süntaksit `S[i]`. Viit massiivile `*S` on samaväär-

ne selle esimese elemendiga `S[0]`. Teegi LibSets funktsioone kasutades toimub täisarvude hulga loomine näiteks järgnevalt.

```
L_Sets S(100);
int **mySet = S.create("5<L:,1,8,15,72,99>");
```

Teegi LibSets realisatsioonis tähistab massiivi `S[0]`-element elementide arvu. Hulgateek LibSets hoiab elemente massiivides mittekahanevas järjekorras ning selline eeldus seatakse ka sisseloetavatele hulkadele. Mittekahanev järjekord võimaldab jagada ülesannet osadeks, rakendada kiireid algoritme elemendi otsimiseks, hulkade ühendamiseks ning teisteks operatsioonideks.

Hulkade realiseerimine massiividena on loomulik ja selgesti loetav. Tuleb siiski tähele panna ka sellise struktuuri puuduseid. Nimelt on järjekorda seatud hulgaelementide korral lisamise ning eemaldamise tehted võrdväärsed terve massiivi loomisega. Teiseks kulub iga hulgaelemendi salvestamiseks täisarvu jagu bitte, s.t vaadeldaval platvormil tüüpiliselt 32 bitti ehk 4 baiti. Edasises näitame, et hulki saab realiseerida ka märksa väiksema ruumikuluga ning ilma loetletud puudusteta.

2.1.2 Bitivektorid

Hulgaloogika fundamentaalseim predikaat on elemendi hulka kuuluvuse kohta. Mingi domeeni U iga elemendi k ja iga hulga S kohta on üheselt määratud, kas element on selle hulga liige või mitte. Osutub, et hulka saab kujutada bitivektorina B ning elemendi k hulka kuuluvuse määrab bitivektori bitt b_k.

$$B = \{b_1, b_2 \dots b_n\},\ n = |U|$$

$$b_k = \texttt{true} \leftrightarrow k \in S$$

$$b_k = \texttt{false} \leftrightarrow k \notin S$$

Teegis Libsets on bitivektorid BA_SETS realiseeritud samuti täisarvude massiivina, kuid siinkohal kujutab iga täisarv 32 võimalikku hulgaelementi. Massiivi 0-element on samuti reserveeritud elementide arvu jaoks, kuid bitivektori korral on see antud negatiivsena, võimaldades programmisisest bitivektorite ja massiivide eristamist.

```
BA_Sets S(100);
int **mySet = S.create("-5<B:,1,8,15,72,99>");
```

Bitivektorite vahelised tehted sooritatakse terve täisarvu ehk 32 hulgaelemendi (biti) kaupa, kasutades C loogikatehteid (vt. tabel 2), mis on implementeeritud suhteliselt madalal tasemel ning peaks andma tunduvalt võitu töökiiruses. Elemendi lisamine ning kustutamine hulgast on sisuliselt biti väärtuse vahetamine.

&, &=	Loogiline $\cap$, bitikaupa korrutamine ja omistamine
|, |=	Loogiline $\cup$, bitikaupa liitmine ja omistamine
$\wedge$, $\wedge$=	Loogiline $\triangle$, bitikaupa välistav liitmine ja omistamine
», »=	Bitikaupa paremale nihutamine ja omistamine
«, «=	Bitikaupa vasakule nihutamine ja omistamine

Tabel 2: Loogikaoperaatorid keeles C

Bitivektorite realisatsioon põhineb kahel väga olulisel detailil. Esiteks tuleb arvestada sellega, et mälu tuleb reserveerida ka hulgast puuduvatele elementidele, kirjutades vastavasse bitti väärtuseks null.

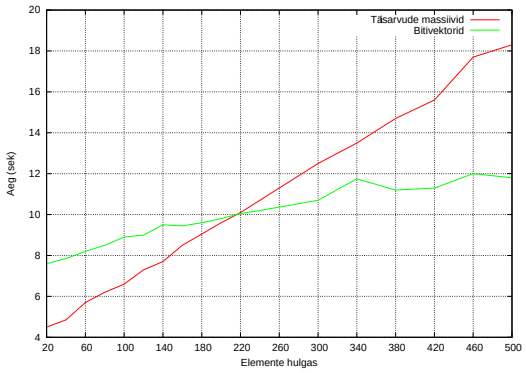
Teiseks peame juba bitivektori loomisel ette teadma antud domeeni maksimaalse elemendi väärtust. Nendest asjaoludest johtuvalt peame alati mälu reserveerima universaalhulga täies ulatuses, mis aga mõningatel juhtudel võib osutuda ressursside raiskamiseks.

2.1.3 Massiivide ja bitivektorite efektiivsus

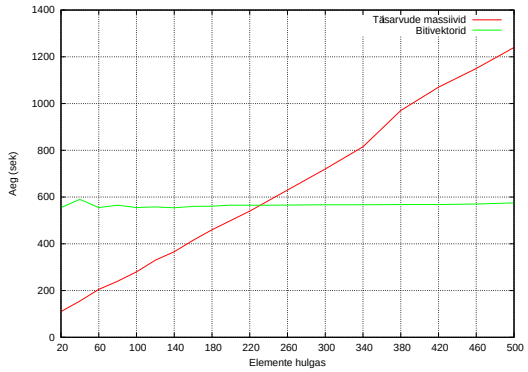
Võib öelda, et hulkade realiseerimisel massiividena või bitivektoritena on omad head ja halvad küljed. Ilmselt on ka efektiivsus erineva iseloomuga sisendandmete korral erinev. Teoreetiliseks efektiivsuspiiriks on $\frac{1}{32}$, millest suurema tihedusega hulki on mõistlikum töödelda bitivektorite kujul, väiksemaid aga massiividena.

Varasemas töös [Rei04] olen bitivektoritel ja massiividel põhinevate erineva struktuuriga hulkade tööaja ja sisselugemise efektiivsust võrrelnud ning üldjoontes kinnitavad saadud andmed teoreetilist piiri.

Kokkuvõtvad tulemused on esitatud joonistel 1 ja 2. Vasakpoolsel joonisel on mõõdetud 6000 hulga sisendist lugemise aega. Paremal joonisel on mõõdetud kõikide hulkade omavaheliste ühisosade leidmiseks kulunud aega.



Joonis 1: Hulkade sisendi kiirus



Joonis 2: Ühisosa leidmise kiirus

Arvestades teoreetilist hinnangut ning seda kinnitavaid testandmeid, on soovitatav kasutada erinevaid realisatsioone olenevalt vaadeldava hulga tihedusest. Seega on teegis LibSets võimalik luua üldistatud hulgastruktuur `Sets`, mille korral otsutatakse antud hulga tiheduse alusel automaatselt ühe või teise struktuuri kasuks ning edasine hulkadega opereerimine toimib kasutajale tüübist sõltumatult.

2.1.4 Hulkade sisend-väljund ning efektiivne salvestamine

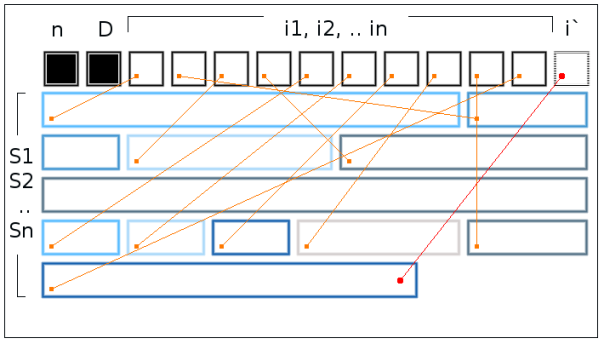
Hulkade salvestamiseks ja failisüsteemist lugemiseks on kõige kergem kasutada tekstiformaati. Niisugusel juhul saab salvestamiseks kasutada eelnevas mainitud funktsioone ja suunata programmi väljund soovitud tekstifaili. Tekst on inimesele loetav ning vajadusel hõlpsasti muudetav. Viimane eelis ei kehti bitivektoritena säilitatud hulkade korral, mis on reeglina raskesti arusaadavad.

Tekstiformaadi korral on lihtne näha, et tegemist on väga ebaefektiivse salvestusruumi kasutamisega. Nimelt kasutatakse iga sümboli hoidmiseks 1 baiti. Kuna täisarvud on eraldatud komaga ning suurim arv võib olla kuni 10-kohaline, siis kulub hulgaelemendi salvestamiseks kuni 11 baiti.

Arvestades eelnevalt toodud puudusi ning edasises praktilises ja eksperimentaalses osas vajaminevaid mahukaid andmehulki, on käesoleva töö käigus realiseeritud formaat BinSets hulkade binaarseks salvestamiseks ja sisendist lugemiseks.

Formaat BinSets lähtub salvestamisel 4-baidisest täisarvutüübist `INTEGER`. Iga hulk salvestatakse vastavalt efektiivsuspiirile $\frac{1}{32}$ bitivektori või massiivina. Sisend ja väljund toimib hulkade massiivina. Hulkade massiiv $S_1, S_2 \ldots S_n$ salvestatakse faili jadamisi ning ühekordse läbimisega. Faili algus sisaldab hulkade massiivi kohta kehtivat metainformatsiooni ning indeksit hulkade positsioonide määramiseks. BinSets struktuurist annab ülevaate joonis 3. Järgnevas loetelus on toodud BinSets faili komponendid esinemise järjekorras.

- Arv n hulkade arvu märkimiseks,
- Arv D domeenisuuruse ehk suurima võimaliku elemendi märkimiseks,
- Indeks $P_1, P_2 \ldots P_n$ hulkade $S_1, S_2 \ldots S_n$ alguspositsioonide näitamiseks,
- Arv P' viimase hulga viimase elemendi positsiooni näitamiseks,
- Hulkade $S_1, S_2 \ldots S_n$ elemendid.



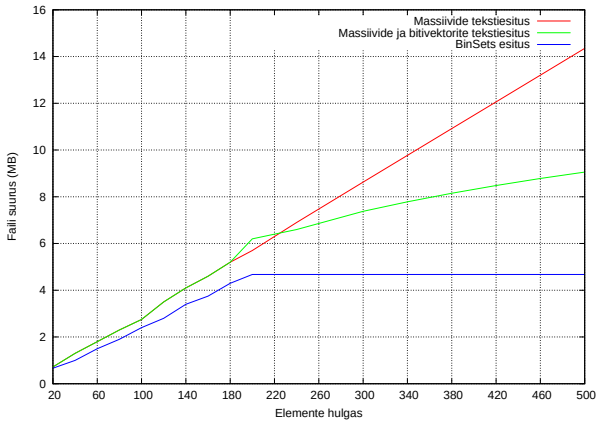
Joonis 3: Binaarne salvestusformaat BinSets

Salvestusruumi vajadusest ülevaate saamiseks võrdlesin ruumivajadust teksti- ja binaarfailidena bioloogiliselt korreleeritud hulkade korral. Andmekomplekti aluseks on 6221 rea ja veeruga ruutmaatriks, mille elementideks on pärmi geenide täisarvulised identifikaatorid. Maatriksi ridadest on $2 \ldots 70$ esimese elemendi kaupa loodud hulgad. Saadud on seega 6221 kaheelemendist hulka, 6221 neljaelemendist hulka, jne. Võrdluseks kasutasin allpool toodud kolme tüüpi hulki. Kahe viimase puhul otsustatakse realisatsiooni kasuks vastavalt efektiivsuspiirile $\frac{1}{32}$.

- Massiivides hulgad tekstina
- Massiivides või bitivektorites hulgad tekstina
- Massiivides või bitivektorites hulgad BinSets formaadis

Nagu võib näha joonisel 4, on mahuline erinevus väikeste hulkade korral praktiliselt olematu, kuid hulkade kasvades kasvab oluliselt BinSets efektiivsus. Tekstina salvestatud massiivide maht kasvab konstantselt ning failisuurused muutuvad kiiresti üle jõu käivaks. Alates tihedusest $\frac{1}{32}$ muutub BinSets-failide suurus täielikult konstantseks, kuna hulkasid hakatakse säilitama bitivektorites. Märgatavalt aeglustub kasv tekstina salvestatud bitivektorite korral, kuid ei peatu täielikult, kuna elementide lisandudes pikenevad mõningal määral bitivektorite arvkujud. Ilm-

selt koondub bitivektorite tekstiesituse maht hilisemas samuti konstantseks, kuna varem või hiljem täituvad kõik bitivektori arvkuju pikkust mõjutavad bitid.



Joonis 4: Salvestusruumi võrdlus

Võrdluseks toon ära failide suurused suuruse 3000 juures, kus esindatud on pooled domeeni elementidest ning eeldatavasti on tekstikujul bitivektorid samuti konstantseks koondunud. Massiivina salvestatud tekstifaili suurus on 90 MB, tekstikujul bitivektorite faili suurus 13 MB ning BinSets formaadis salvestatud faili maht on 4.7 MB. Seega võib öelda, et binaarne formaat BinSets on tavalisest tekstisalvestusest salvestusruumi osas ligikaudu 2.75 korda kasulikum.

Edaspidi on programmides kasutatud formaati BinSets ning bitiesituse ja massiivesituse vahel valimist vastavalt piirile $\frac{1}{32}$.

2.2 Hulkade realiseerimine andmebaasis

Alternatiivina loodud hulgaoperatsioonide teegile LibSets vaatlen järgnevas täisarvuhulkade realiseerimist relatsioonilises andmebaasis. Relatsioonilises mudelis säilitatakse andmeid ning nendevahelisi seoseid relatsioonidena ehk tabelites. Tabelid koosnevad ridadest ehk kirjetest ning veergudest ehk vastavate kirjete atri-

buutidest. Atribuudid võivad olla erinevatest tüüpidest, näiteks täisarvulised atribuudid, tõeväärtused, stringid, jne. Kirjetega on seotud unikaalsed võtmed, mille abil on võimalik andmetevaheliste seoste kujutamine.

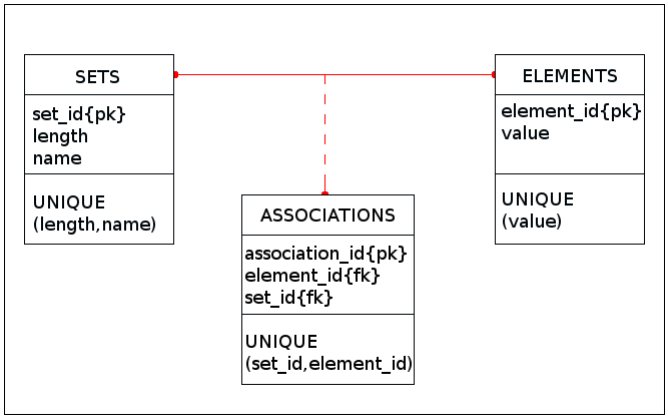
Võrreldes kirjeldatud hulgateegiga on andmebaasil mitmeid eeliseid. Andmete töötlemiseks ja pärimiseks on loodud standardne struktuurpäringukeel (ingl. k. *Structured Query Language, SQL*). SQLi näol on tegemist keelega, milles on vahendid nii andmetüüpide defineerimiseks kui andmete valimiseks ja väljastamiseks. Keele abil on võimalik lihtne hulkade kuvamine, lisamine, muutmine ning tulemustega manipuleerimine.

Andmebaasitarkvaraks olen võrdlevalt valinud vaba lähtekoodiga andmebaasi PostgreSQL versiooni 7.4.7 [pos05] ja MySQL versiooni 4.0.27 [mys05], mis töötavad vabavaralisel Debian Linux [deb05] operatsioonisüsteemil. Programmi MySQL näol on tegemist õhukese ja lihtsatele rakendustele orienteeritud andmebaasiga. Teine andmebaas PostgreSQL sisaldab rohkem lisavõimalusi ning on ka ressursside suhtes nõudlikum.

2.2.1 Hulkade relatsiooniline mudel ja andmebaasiskeem

Hulkade kujutamiseks andmebaasis on sobiv näiteks skeem, mida kujutatud joonistel 5 ja 6. Hulgad ning elemendid on kujutatud eraldi tabelites `Sets` ja `Elements` ning iga kirjega on seotud antud tabelis unikaalne võti `set_id` ja `element_id`. Hulkade ja elementide vahel kehtib mitu-mitmele seos ($m - n$ seos), seega on iga elemendiga võimalik siduda $0 \ldots n$ hulka ning igas hulgas võib olla $0 \ldots m$ elementi, kus m ja n on suvalised täisarvud. Seose realiseerimiseks kasutatakse abitabelit `Associations`, milles säilitatakse hulkade ja elementide võtmete paare.

Kirjeldatud skeemi eeliseks on asjaolu, et tabelitele välju lisades on nii hulkade kui elementide kohta võimalik säilitada mitmesugust lisainformatsiooni. Eri-



Joonis 5: Relatsiooniline mudel - diagramm

nevalt hulgateegile LibSets ei sea eelnevas kirjeldatud relatsiooniline skeem praktiliselt piiranguid hulkade või nendesse kuuluvate elementide arvule. Kuna hulga pikkus on oluline otsinguparameeter, on loodud hulga pikkust sisaldav väli `length`. Arvestades, et tabelid ei muutu sageli, ei ole mõtet igakordsel otsingul elementide loendust rakendada.

Andmebaasiskeemis toodud unikaalsuse nõuetega (võtmesõna `UNIQUE`) on loodud hulkade kasutamiseks korrektsed tingimused - hulgad ei saa sisaldada korduvaid elemente, samuti ei ole lubatud mitmed sama väärtusega elemendid või samanimelised hulgad.

Paljude elementidega tabelite korral on oluline, et päringule vastavate andmete leidmiseks ei oleks vaja sooritada väga ebaefektiivseid terve ridade ruumi läbivaatusi. Läbivaatuste vältimiseks ja töö kiirendamiseks on andmebaasides sisse viidud indeksi mõiste. Indeks on andmestruktuur, mille abil on võimalik andmeridu välja väärtuse järgi väga kiiresti leida. Tavaliselt on soovitatav indeksid luua vähemalt tabelite võtmeväljadele ning andmebaas PostgreSQL teeb seda automaatselt.

Indeksite loomine kiirendab andmete otsingut, kuid muudab tunduvalt aeglasemaks ridade lisamise, muutmise ja kustutamise operatsioonid. Praeguses üles-

```
CREATE TABLE sets (
 set_id  INTEGER PRIMARY KEY AUTO_INCREMENT,
 length  INTEGER NOT NULL,
 name    INTEGER NOT NULL,
 UNIQUE (length,name));
CREATE TABLE elements (
 element_id   INTEGER PRIMARY KEY AUTO_INCREMENT,
 value        INTEGER UNIQUE NOT NULL),
 UNIQUE(value));
CREATE TABLE associations (
 association_id  INTEGER PRIMARY KEY AUTO_INCREMENT,
 element_id      INTEGER NOT NULL,
 set_id          INTEGER NOT NULL,
 UNIQUE (element_id,set_id));
```

Joonis 6: Relatsiooniline mudel - tabelite loomine andmebaasi

andes on peamine hulkadevaheliste seoste arvutamine ning andmete lisamine on ühekordne tegevus. Seega on loodud kõik vajalikud indeksstruktuurid.

2.2.2 Hulgaoperatsioonid andmebaasis

Andmebaasiplatvorm annab struktuurpäringukeele SQL näol loomulikud vahendid hulkade kuvamiseks ja hulgatehete sooritamiseks. Antud relatsioonilises skeemis tuleb näiteks 25. hulga kuvamiseks selle elemendid assotsiatsioonide tabeli võõrvõtmete kaudu siduda. Struktuurpäringukeeles on selleks `WHERE`-lause (päring joonisel 7).

Ühendi ja ühisosa leidmiseks saab `WHERE`-lause tingimusi siduda binaarsete operaatorite `AND` ja `OR` abil filtreerimaks välja vastavatesse alamhulkadesse kuu-

```
SELECT s.set_id, e.value
FROM elements e, associations a, sets s
WHERE e.element_id=a.element_id
 AND s.set_id=a.set_id AND s.set_id=25;
```

Joonis 7: Päring hulga kuvamiseks

luvate elementide võtmeid. Järgneva lause abil (joonis 8) saab leida 25. ja 26. hulga ühisosa.

```
SELECT s1.name, s2.name, e1.value
FROM sets s1, sets s2, elements e1,
 associations a1, associations a2,
WHERE s1.set_id=25  AND s2.set_id = 26
 AND s1.set_id=a1.set_id AND s2.set_id=a2.set_id
 AND a1.element_id=a2.element_id
 AND e1.element_id=a1.element_id
```

Joonis 8: Päring kahe hulga ühendi kuvamiseks

Lisaks on struktuurpäringukeeles defineeritud spetsiaalsed funktsioonid ridade ühendi, ühisosa ja vahe leidmiseks, kuid võrdlusesse valitud tarkvarast toetab neid vaid PostgreSQL. Funktsioonid on toodud tabelis 3. Siinkohal tähistab (ingl. k. *val*) hulgaelemendi väärtust, `sets` hulkade tabelit ning `id` hulga nimetust . Üldiselt tagab PostgreSQL hulgatehetes `UNION`, `INTERSECT`, `EXCEPT` automaatselt elemendi unikaalsuse tingimuse. Kui seda ei soovita, tuleks kasutada võtmesõna `SELECT ALL`.

Edasises efektiivsusvõrdluses leiab rakendust koondpäring (joonis 9), milles leian ühe hulga ühisosa kõigi teiste hulkadega, millel on esimesega sama pikkus.

$A \cup B$	`SELECT val FROM sets WHERE id='A'`
	`UNION`
	`SELECT val FROM sets WHERE id='B'`

$A \cap B$	`SELECT val FROM sets WHERE id='A'`
	`INTERSECT`
	`SELECT val FROM sets WHERE id='B'`

$A \setminus B$	`SELECT val FROM sets WHERE id='A'`
	`EXCEPT`
	`SELECT val FROM sets WHERE id='B'`

Tabel 3: Hulgatehted SQLis

2.2.3 Hulgaoperatsioonide efektiivsus

Järgnevalt uurin hulgaoperatsioonide efektiivsust andmebaaside PostgreSQL ja MySQL ning hulgateegi LibSets vahel. Võrreldavateks operatsioonideks on valitud hulkade väljatrükk ning ühisosa leidmine.

Katseandmetena on kasutatud sarnaselt varasematele võrdlustele bioloogiliselt korreleeritud hulki domeeniga 6221. Andmekomplekti aluseks on 6221 korda 6221 ruutmaatriks, mille ridadest on $2 \ldots 70$ esimese elemendi kaupa loodud hulgad. Seega sisaldab elementide tabel 6221 elementi, hulkade tabel 217735 hulka ning assotsiatsioonide tabel 7838460 hulk-element seost.

Efektiivsusvõrdluse esimeses osas selgitan välja antud operatsioonide ja andmete jaoks kiirema andmebaasitarkvara ning päringute sooritamise viisi. Testpäring leiab ühe 20-elemendilise hulga ühisosa kõigi teiste sama suurte hulkadega.

Päringute sooritamiseks on kasutatud kolme erinevat lähenemist. Ühe võimalusena olen skriptitsüklite abil koostanud päringute faili, kus iga päring leiab ühe hulkade paari vahelise ühisosa tavalise `WHERE`-lause abil. Teise võimalusena olen

```
SELECT s1.name, s2.name, e1.value
FROM sets s1, sets s2, elements e1,
 associations a1, associations a2
WHERE s1.set_id=25 AND s1.set_id!=s2.set_id
 AND s1.length=10 AND s1.length=s2.length
 AND s1.set_id=a1.set_id AND s2.set_id=a2.set_id
 AND a1.element_id=a2.element_id
 AND e1.element_id=a1.element_id
```

Joonis 9: Päring ühendite leidmiseks antud hulga ja ülejäänud hulkade vahel

koostanud analoogse päringufaili `INTERSECT`-võtmesõnu kasutades. Seda mõõtmist saan rakendada ainult PostgreSQL tarkvara korral. Kolmandaks võimaluseks on eelneva alajaotuse lõpul toodud koondpäring. Erinevate meetodite ja andmebaaside ajalised tulemused on toodud tabelis 4.

Tarkvara	Päring	Aeg
MySQL	`WHERE`-laused failist	3h 46m
PostgreSQL	`WHERE`-laused failist	21s
PostgreSQL	`INTERSECT`-laused failist	48s
MySQL	koondpäring	2.88s
PostgreSQL	koondpäring	0.32s
LibSets		0.71s

Tabel 4: MySQL, PostGreSQL ja LibSets ühisosade leidmise võrdlus

Tabelist 4 on näha, et niisuguse mastaabiga arvutuste juures on PostgreSQL andmebaas MySQL tarkvarast kordi kiirem. Eriti paistab see silma päringufailidega töötamisel. Faili `file.sql` sisendisse lugemiseks on PostgreSQL baasi

klientprogrammil spetsiaalne süntaks (käsureaviit `-f`). Samas MySQL korral tuleb kasutada päringufailide lugemiseks toru `<`, milles päringud söödetakse kliendile ette ridahaaval. Seega lisandub iga MySQL päringule konstantne aeg kliendi laadimiseks.

```
psql -f file.sql;    mysql < file.sql
```

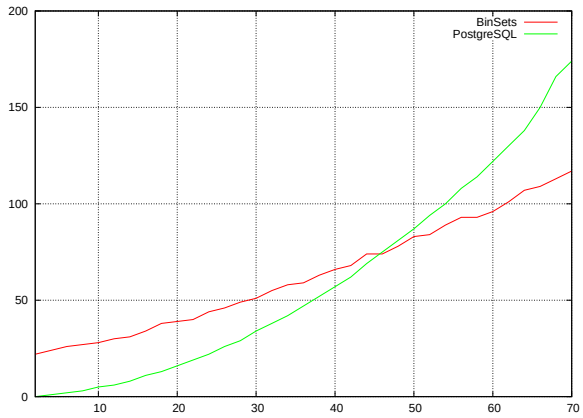
Efektiivsusvõrdluse teises osas vaatlen PostgreSQL ja LibSets käitumist ühisosade leidmisel. Andmekomplekt on sarnaselt eelnevale bioloogiline maatriks, kuid vaatlen kõikide sama pikkusega hulgapaaride vaheliste ühisosade leidmist, kusjuures kasutan kõiki 35 hulgakomplekte pikkustega $2 \ldots 70$. Iga pikkuse juures leitakse ühisosad temast järgnevate ridadega. Nii väldin paaride ühisosade korduvat leidmist. Antud andmebaasiskeemis tähistab maatriksi rea numbrit väli `name`. Kõikide 20-elemendiliste hulkade ühisosade leidmiseks sobib järgnev koondpäring (joonis 10).

```
SELECT s1.name, s2.name, e1.value
FROM sets s1, sets s2, elements e1,
 associations a1, associations a2
WHERE s1.set_id!=s2.set_id AND s1.name<s2.name
 AND s1.length=s2.length AND s1.length=20
 AND s1.set_id=a1.set_id AND s2.set_id=a2.set_id
 AND a1.element_id=a2.element_id
 AND e1.element_id=a1.element_id
```

Joonis 10: Päring ühendite leidmiseks kõikide hulkade vahel

Teegi LibSets ja andmebaasitarkvara PostgreSQL ühisosade leidmise võrdluse tulemused on nähtaval joonisel 11. Näeme, et PostgreSQL on efektiivsem, kui hulgad on väiksemad, kuid mahtude suurenedes kasvab andmebaasi töö ajakulu

järjest kiiremini. Samal ajal on LibSets koodi ajafunktsioon suhteliselt lineaarselt kasvav. Samuti teame, et seoses bitivektorite rakendamisega alates hulgatihedusest $\frac{1}{32}$ ehk ligikaudu pikkusest 194 muutub hulkade leidmise ajakulu praktiliselt konstantseks.



Joonis 11: LibSets ja PostgreSQL ühisosade leidmise võrdlus

3 Geeniontoloogiad

Geeniontoloogiate konsortsium GO on koostööprojekt, mille eesmärgiks on bioloogilise info integreerimine, läbivate standardite loomine ja praktikas rakendamine [HJA+04, MBB00]. Organismide genoomide uurimine on viinud olukorrani, kus erinevate töörühmade tulemusi ja genoomide andmeid säilitatakse erinevates andmebaasides. Geeniontoloogiad loovad geenidele ja funktsionaalsusele organismist sõltumatu vaate, mille abil on võimalik bioloogilisi andmekogusid integreerida.

Esiteks on GO näol tegemist molekulaarbioloogia domeene, geenide ning geeniproduktide omadusi ja bioloogilisi järjestusi kirjeldavate ontoloogiatega. Ontoloogiaks nimetatakse siin ja edaspidi hierarhilise struktuuriga sõnaraamatut ning termiks ehk mõisteks nimetatakse antud struktuuri ühte tippu. Ontoloogiad loovad standardi erinevate organismide geenide ning geeniproduktide funktsionaalsuse ühtseks klassifitseerimiseks ning on ühenduslüliks erinevate organismide genoomide vahel.

Teiseks GO projekti eesmärgiks on nimetatud mõistete hierarhia rakendamine erinevate organismide geenide annoteerimisel. Annoteerimise all on silmas peetud geenile vastava ontoloogiamõiste määramist. Erinevate organismide geenide annoteerimine ühtsesse võrku aitab vältida terminoloogilist segadust ning pakub mitmeid huvitavaid küsimusi geneetika universaalsuse seisukohast.

Käesolevas peatükis annan ülevaate geeniontoloogiate, nendega seotud annotatsioonide ning katseandmetena kasutatud geenigruppide struktuurist ja iseloomust.

3.1 GO ontoloogiad

Ontoloogia on hierarhiline struktuur, täpsemalt suunatud atsükliline graaf, mille tippudeks on mõisted. Ontoloogiagraafis on semantiliselt erineva alluvussuhte tähistamiseks võimalik kasutada `part-of` ja `is-a` relatsioone. Relatsiooni `is-a` abil tähistatakse suhet, milles alluv tipp on vanemast spetsiifilisem mõiste. Relatsioon `part-of` tähistab asjaolu, et alluv tipp moodustab osa vanemast. Näiteks mõiste *telomeer* moodustab osa mõistest *kromosoom*.

Juure tasemel jaguneb ontoloogiastruktuur kolmeks omavahel mittelõikuvaks haruks, mis kirjeldavad molekulaarsete funktsioonide (MF), bioloogiliste protsesside (BP) ja rakukomponentide (CC) domeene. Iga mõistega on seotud unikaalne võti (näiteks `GO:0003993`) ning tekstiline kirjeldus, näiteks *happelise fosfataasi aktiivsus*. Igal juurest erineval tipul võib olla üks või mitu vanemat ning null või enam alluvat.

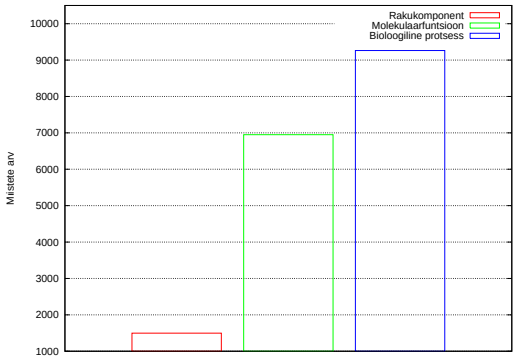
Molekulaarsed funktsioonid kirjeldavad geenidega seotud protsesse molekulaartasemel, kusjuures üldise lähenemise huvides jäetakse täpsustamata protsessi kontekst ning selles osalevad komponendid. Molekulaarprotsessi näiteks sobib väga üldine term *kinaasi aktiivsus* ning selle alamtüüp *6-fosfofruktokinaasi aktiivsus*.

Bioloogilised protsessid kirjeldavad molekulaarsete funktsioonide või nende jadade tulemusena toimuvaid bioloogilisi muutusi. Bioloogiliseks protsessiks on näiteks rakusurm *raku surm*, millel võib olla nii alamtüüpe (*apoptoos*) kui ka otseseid alamprotsesse (*apoptootiline kromosoomi kondenseerumine*).

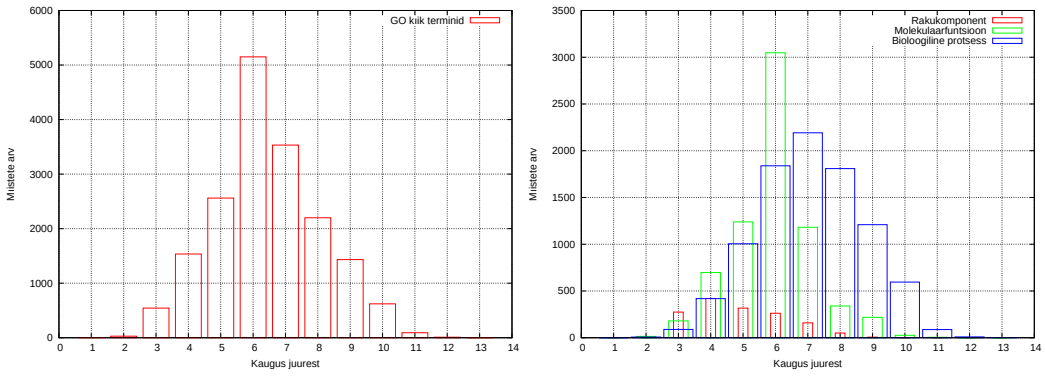
Rakukomponendid kirjeldavad geenide funktsionaalsusega seotud asukohti makromolekulaarsete komplekside ja rakusiseste komponentide tasemel. Näiteks on rakukomponentideks *tuuma sisemembraan* ning *ubikvitiini ligaasi kompleks*. Mõlema kompleksiga on seotud hulk alamtüüpe.

Geeniontoloogiates on GO veebilehekülje [go05] andmetel 09.aprilli 2005 sei-

suga 18696 mõistet, kusjuures 93.8% mõistetetele on lisatud definitsioon. Nendest 9263 kirjeldavad bioloogilisi protsesse ning 6949 molekulaarseid funktsioone. Rakukomponente kirjeldab 1496 mõistet. Kõikidest mõistetest on 988 on märgitud vananenuks (ingl. k. *obsolete*) ning neid ei ole käesolevates numbrites kajastatud. Visuaalse ülevaate mõistete domeenide vahelisest jaotusest annab joonis 12.



Joonis 12: GO mõisted domeenide lõikes



Joonis 13: GO mõistete jaotused ontoloogiagraafi tasemetel

Ontoloogiagraafi sügavuseks ehk pikimaks teeks lehest juurtippu on 13. Arvestades, et tegemist on suunatud atsüklilise graafiga, võib igal vahetipul olla roh-

kem kui üks vanem. Graafi konstrueerisin laiuti alates juurtippudest ehk domeenidest, võttes null-tasemeks domeenid (MF,BP,CC) ning esimeseks tasemeks tipud, millel ei ole märgitud ühtegi vanemat ehk seost `is_a` või `part_of`. Seejärel valisin järgnevaks tasemeks tipud, mille vanem oli eelmisel tasemel. Siinkohal on iga tipp lisatud varaseimale sobivale tasemele, seega on leitud iga tipu jaoks lühim tee juurtipuni. Jooniselt 13 saab ülevaate mõistete jaotusest graafi tasemete ja erinevate domeenide lõikes.

Visuaalsel vaatlusel jaotuvad ontoloogiad üldiselt normaaljaotuse järgi, kusjuures jaotused on omavahel veidi nihkes vastavalt alamate koguarvule. Esimesel tasemel on graafis kolm tippu, nendeks on bioloogiliste protsesside (`GO:0008150`), molekulaarsete funktsioonide (`GO:0003674`) ja rakukomponentide (`GO:0005575`) domeenid.

3.2 GO annotatsioonid

Geeniontoloogiate projekt sai alguse 1998. aastal kolme organismi, äädikakärbse, pagaripärmi ja hiire genoomide andmebaaside koostööst. Tänaseks on GO internetilehekülje [go05] andmeil geeniontoloogiatesse annoteeritud geene ligi neljakümnest erinevast genoomi või teadusprojekti andmebaasist.

Genoomide annotatsioone säilitatakse tabulaatoriga eraldatud failides. Iga rida tähistab ühte annotatsiooni, kusjuures geeni ja mõiste paarile võib vastata mitu erinevate parameetritega annotatsiooni. Iga annotatsiooniga on seotud viisteist välja. Antud töö raames olulised väljad on toodud tabelis 5. Iga geeniga võib olla seotud üks või mitu annotatsiooni. Iga annotatsiooni juures on toodud allikas, milleks võib olla viide kirjandusele, andmebaasile või arvutusliku analüüsi tulemusele.

Iga annotatsiooni juures peab olema toodud tõendikood (ingl. k. *evidence code*), mis näitab, mis laadi eksperimentide tulemusel on geen vastavalt annoteeritud. Standardsed geeniontoloogiates kasutusel olevad koodid koos selgitustega on

toodud tabelites 6 ja 7. Tõendikoodide juures tuleb märkida, et seni ei ole määratud kindlat koodide headuse järjekorda. Tõendikoodide peamiseks eesmärgiks on kasutajapoolse valiku hõlbustamine.

Tõendikoodid võimaldavad annotatsioone erinevalt hinnata. Nii on GO kuraatorite poolt kirjanduse põhjal soovitatud ja eksperimentaalsete andmetega kinnitatud annotatsioonid ilmselt usaldusväärsemad kui sellised, mille ainsaks tõendusmaterjaliks on elektrooniliste andmete analüüs või simulatsioon.

Samas on ontoloogiate üheks peamiseks eesmärgiks paindlikkus ja universaalsus genoomide suhtes. Erinevate tõendikoodide abil on võimalik annoteerida ka näiteks inimese genoomi andmeid, millel on senini vähe tuge reaalsete eksperimentide näol, kuid on sooritatud hulganisti arvutuslikke katseid.

#	Nimetus	Kirjeldus
1.	DB	Andmebaasi tunnus, millest annotatsioon pärineb.
5.	GOID	GO identifikaator geenile või geeniproduktile annoteeritud mõistele.
7.	Evidence	Tõendikood, vt. tabelid 6 ja 7.
9.	Aspect	Annotatsiooni domeeni tunnus, bioloogiline protsess (`b`), molekulaarne funktsioon (`f`) või rakukomponent (`c`).
10.	DB_Object_Name	Geeni või geeniprodukti tekstiline nimetus.
11.	DB_Object_Synonym	Geeni või geeniprodukti identifikaator. Esimesel kohal on primaarne identifikaator, järgneda võivad püstkriipsuga eraldatud aliased.
12.	DB_Object_Type	Väli kirjeldab annoteeritava objekti tüüpi. Käesoleva töö raames vaadeldakse kirjeid, mille tüübiks on märgitud `gene`.
14.	Date	Annoteerimise kuupäev.

Tabel 5: Annotatsioonide andmeväljad

IC	Kuraatori järeldus (ingl. k. *Inferred by curator*) - Annotatsiooni kohta pole otseseid tõendeid, kuid järelduse on teinud GO kuraator tõenditega kinnitatud analoogiliste annotatsioonide alusel.
IDA	Järeldus otsesest analüüsist (ingl. k. *Inferred from direct assay*) - Annotatsiooni tõendavad tulemused bioloogilistest eksperimentidest, näiteks ensüümianalüüsist, *in vitro* transkriptsioonist.
IEA	Järeldus elektroonilisest annotatsioonist (ingl. k. *Inferred from electronic annotation*) - Annotatsiooni tõendid on saadud järjestuste või andmebaasikirjete sarnasuse (elektroonilise) analüüsi tulemusena. Tulemused on saadud mitte GO kuraatori poolt, vaid kolmanda osapoole vahendusel.
IEP	Järeldus ekspressioonimustrist (ingl. k. *Inferred from expression pattern*) - Annotatsioon on järeldus geeni ekspressiooni toimumisaega või kohta analüüsides.
IGI	Järeldus geneetilisest interaktsioonist (ingl. k. *Inferred from genetic interaction*) - Annotatsiooni tõendiks on eksperimendid, milles on muudetud rohkem kui ühe geeni või geeniprodukti ekspressioonitaset või esinemisjärjekorda.
IMP	Järeldus mutantsest fenotüübist (ingl. k. *Inferred from mutant phenotype*) - Annotatsiooni tõendiks on eksperimendid, milles on muudetud (muteeritud) täpselt ühe geeni või geeniprodukti avaldumist või järjekorda.
IPI	Järeldus füüsilisest interaktsioonist (ingl. k. *Inferred from physical interaction*) - Annotatsiooni tõendab vaadeldava produkti füüsiline side mõne muu molekuliga, näiteks valkude või aminohapete sidumise korral.

Tabel 6: GO tõendikoodid

ISS	Järeldus järjestuse või struktuuri sarnasusest (ingl. k. *Inferred from sequence or structural similarity*) - Annotatsiooni tõendiks on sekventsijoonduse analüüs või struktuuri jm. omaduste võrdlus.
RCA	Järeldus valitud arvutuslikust analüüsist(ingl. k. *Inferred from reviewed computational analysis*) - Annotatsioon on tehtud arvuste põhjal, mis ei puuduta järjestuste analüüsi. Selle koodiga tähistatakse näiteks suurtes andmehulkades tuvastatud seoseid ja tekstide automaatsest analüüsist (n.n. tekstikaevandamisest) saadud tulemusi. Selle tõendikoodiga tähistatud eksperimendid peavad olema GO kuraatorite poolt kinnitatud.
NAS	Tundmatu autori allikas (ingl. k. *Non-traceable author statement*) - Selle tõendikoodiga tähistatakse nii puuduvate kirjanduslike allikatega annotatsioone, kui neid, mille kohta ei ole tuvastatud alternatiivseid allikaid lisaks viidatule.
ND	Bioloogilised andmed puuduvad (ingl. k. *No biological data available*) - Selle tõendikoodiga tähistatakse tundmatu molekulaarse funktsiooni (GO:0005554), bioloogilise protsessi (GO:0000004) või rakukomponendiga (GO:0008372) annotatsioone. Tõendikood näitab, et kuraator on tutvunud produkti puudutavate allikatega, kuid pole ühegi mõiste jaoks kinnitust leidnud.
TAS	Teadaoleva autori allikas(ingl. k. *Traceable author statement*) - Annotatsiooni tõendab originaaleksperimente puudutav ülevaateartikkel või üldiste teadmiste hulka loetav õppematerjal, sõnastik jne.
NR	Salvestamata(ingl. k. *Not recorded*) - Annotatsioon on lisatud enne tõendikoodide kasutuselevõttu. Kood esineb pagaripärmi ja äädikakärbse annotatsioonide arhiivides ning uute annotatsioonide lisamisel seda ei kasutata.

Tabel 7: GO tõendikoodid jätk

3.3 Annotatsioonid pagaripärmi genoomil

Käesolev töö keskendub andmeanalüüsis hariliku pagaripärmi ehk *Saccharomyces cerevisiae* genoomi andmebaasile SGD [yea05] ja vastavatele GO annotatsioonidele. Pagaripärm sekveneeriti täielikult aastaks 1996 ja on seega üks enim uuritud genoomiga organisme. Pagaripärmi genoom on väike, sellel on 16 kromosoomi ja genoomi pikkuseks on üle 12 miljoni aluspaari. Pagaripärmi andmete maht on piisavalt kompaktne, et suure osa vajalikest analüüsiarvutustest saab läbi viia personaalarvutil. Samuti on pagaripärmi kohta saadaval rohkelt varasemaid tulemusi ja kirjandust.

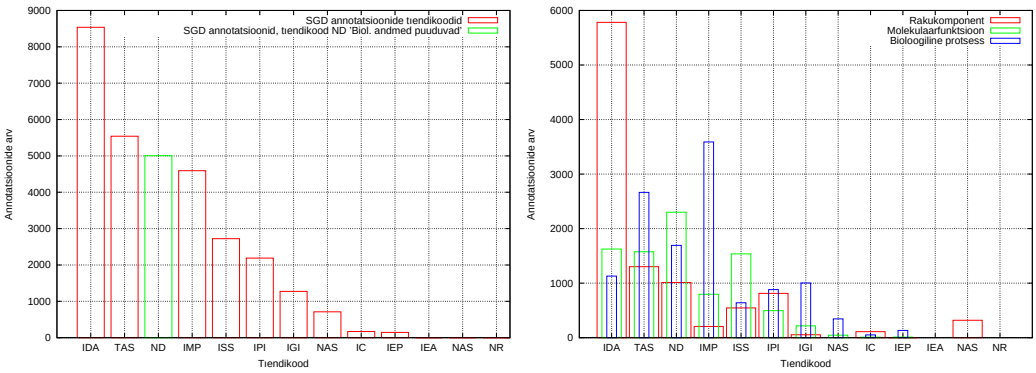
2005. aasta 9. aprilli seisuga on SGD andmebaasis 6591 ORFi (ingl. k. *Open Reading Frame*). ORFiks nimetatakse DNA lõiku, mis kindlate algus- ja lõpumarkerite olemasolu tõttu võib osutuda valke kodeerivaks geeniks. Harilikult ei loeta ORFi geeniks enne, kui selleks on leitud eksperimentaalsel teel kinnitust. Käesolevas töös ei ole selguse mõttes ORFe ja geene terminoloogiliselt eristatud, kuna elementide geenide või ORFide hulka kuulumine ei ole sageli selgelt määratav.

Kõikidele pagaripärmi ORFidele on antud unikaalsed nimetused (näiteks kujul `YBR085W-C`), kus `Y` tähistab pagaripärmi (ingl. k. *yeast*), `B` tähistab kromosoomi järjekorranumbrit, milles vaadeldav ORF asub (`A`=1, `B`=2, . . . `P`=16). Järgnev sümbol (`L` või `R`) näitab, kas ORF asub kromosoomi vasakul või paremal õlal. Kolmekohaline number näitab ORFi järjekorda kromosoomi õlal ning järgnev täht `W` või `C` seda, kas kromosoom asub DNAs Watson- või Crick-ahelal. Kui leitakse ORF, mille nimi on juba kasutuses, võetakse tavaliselt varem leitud ORFi nimi ning liidetakse sellele sufiksiks veel üks täht (`A`, `B` . . .) vältimaks varasemate nimetuste ümbernummerdamist.

Annotatsioonifailis on märgitud 6281 geeni või geeniprodukti, nendest 5863 kannavad ORFi standardkujul nimetust. Geenid on seotud 2530 erineva geeniontoloogia mõistega. Kokku on annotatsioonifailis 25992 unikaalset SGD produkti

ja GO mõiste paari, nendest 24377 on seotud ORFidega.

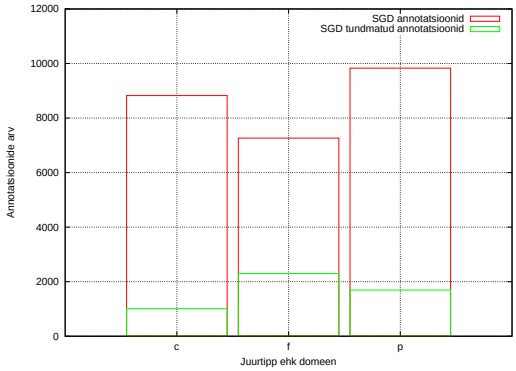
Esimese analüüsina olen uurinud pagaripärmi geenide annotatsioonide jaotusi tõendite järgi. Analüüsi tulemused on nähtaval joonistel 14, kusjuures vasakpoolsel graafikul on eraldi välja toodud kood `ND` - 'bioloogilised andmed puuduvad' (ingl. k. *No biological data available*). Tõendikoodi `ND` 5005 esinemist annab küllalt olulise tulemuse, nimelt tunnistavad üle 20% kõigist annotatsioonidest asjaolu, et annoteeritava geeni või produkti roll või toimimispiirkond on teadmata. Nagu on näha tabelist 7, viitavad mainitud koodiga annotatsioonid ühele kolmest tundmatust kategooriast (`GO:0005554`, `GO:0000004` või `GO:0008372`).



Joonis 14: SGD tõendikoodid

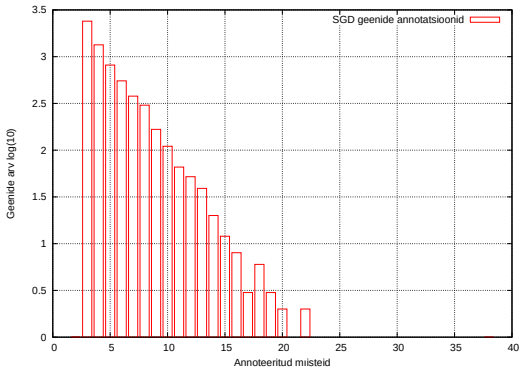
Järgmisena uurin loomulikku küsimust tundmatut tähistava koodi `ND` osakaalust pagaripärmi rakukomponentide (c), bioloogiliste funktsioonide (f) ja molekulaarsete funktsioonide (p) lõikes. Arvulised andmed ja visuaalne tulemus on näha joonise 3.3 tabelis ja graafikul. Eriti suur on `ND`-koodide osa molekulaarsete funktsioonide juures, kus iga ligi kolmas annotatsioon viitab tundmatule kategooriale. Arvestades, et `ND` määratakse geenile vaid andmete täielikul puudumisel ning järelikult ei ole sellele teisi antud domeeni mõisteid nimetatud, võib öelda, et ligi 37% 6281-st SGD baasi annoteeritud geenist on tundmatu molekulaartaseme funktsiooniga.

	Kõik	ND
c	8826	1012
f	7265	2300
p	9831	1693
+	25922	5005

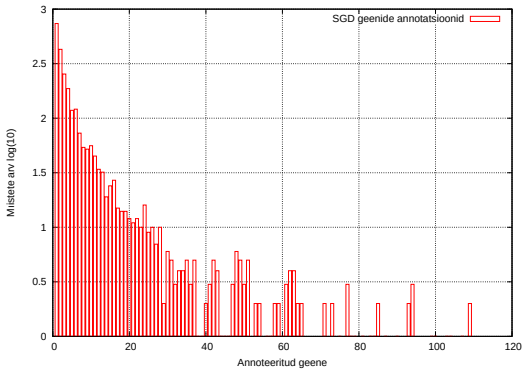


Joonis 15: SGD annotatsioonid ja kood ND domeenide järgi

Samuti võrdlen SGD geenide ja geeniontoloogia mõistete seotust. Teatavasti mõistete ja geenide vahel tegemist `m-n` ehk mitu-mitmele seosega. Seega tekivad küsimused, mitu geeni või geeniprodukti on seotud GO mõistetega ning mitme GO kirjeldava mõistega on geenid enamasti seotud. Vastavad tulemused on toodud graafikutel 17 ja 16. Tabelis 8 on antud kõige rohkem kasutatud mõisted ja enim annoteeritud geenid.

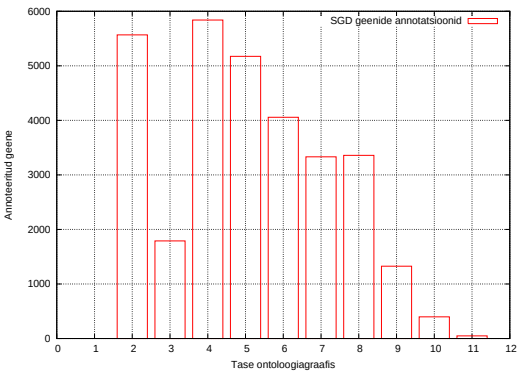


Joonis 16: Mõisteid geenide kohta



Joonis 17: Geene mõistete kohta

Nagu näha, on enimkasutatud mõistete järjestuses tundmatute mõistete kategooriad. Arvestades, et praktiliselt iga annoteeritud geeniga on seotud vähemalt



Joonis 18: SGD geenid ontoloogiagraafi tasemetel

kolm mõistet, võime järeldada, et suure osa annotatsioonides moodustavad geenid, millele on leitud sobiv mõiste ühes domeenis ning seejärel on vaadeldav geen lisatud teiste domeenide juurde tundmatusse kategooriasse. Järeldust kinnitavad ka pistelised katsed annotatsioonifailiga.

Alajaotuse lõpetuseks annan veel tulemuse geenide jaotusest ontoloogiagraafi tasemetele. Graafik 18 kinnitab juba sõnastatud järeldusi - oluline osa annotatsioonidest on teisel tasemel ehk tundmatu funktsionaalsusega kategooriate juures.

allikas	kood	#	nimetus / alias
GO	GO:005554	2300	tundmatu molekulaarne funktsioon (MF)
GO	GO:000004	1693	tundmatu bioloogiline protsess (BP)
GO	GO:005732	1434	väike tuuma ribonukleovalgu kompleks (CC)
GO	GO:005634	1225	tuum (CC)
GO	GO:008372	1012	tundmatu rakukomponent (CC)
GO	GO:005739	931	mitokonder (CC)
SGD	YFL039C	38	aktiin-513
SGD	YNL079C	22	tropomüosiin
SGD	YIL138C	22	tropomüosiin
SGD	YML085C	20	-
SGD	YML124C	20	-
SGD	YER133W	19	DIS2S1

Tabel 8: Enimmärgitud mõisted ja geenid

4 Geeniontoloogiate kaevandamine

Geeniontoloogiate, pagaripärmi genoomi annotatsioonide ja ekspressiooniandmete põhjal koostatud geeniklastrite edasiseks analüüsiks on mitmeid motiveerivaid põhjuseid.

Ontoloogiate analüüsiks on vaja vahendit, millega saaks annoteeritud geene ja tundmatuid geene grupeerida, leida statistiliselt sobivaid geeniontoloogiate mõisteid ning selle abil tundmatutele geenidele sobivaid mõisteid välja pakkuda.

Selles peatükis annan ülevaate pagaripärmi genoomi andmebaasi ning vastavate geeniontoloogiate annotatsioonide kaevandamiseks tehtud praktilisest tööst. Peatüki esimeses pooles kirjutan hulkade headuse hindamise funktsioonidest ja juhuslike hulkade hindamisest. Seejärel tutvustan valminud geeniontoloogiate kaevandamise programme *GOSTminer*, *GOST in the Shell* ja *GOST on the Web*.

4.1 Hulkade olulisuse hindamine

Edasises analüüsis on tähtis roll funktsioonidel, mille abil oleks võimalik saadud tulemusi statistilise olulisuse alusel järjestada. Töös olen hulkade ühisosade olulisuse hindamiseks kasutanud paralleelselt kolme erinevat olulisusmõõtu - hulkade sümmeetrilist vahet ning binoom- ja hüpergeomeetriliste jaotuste tõenäosusi. [vH05].

4.1.1 Ühisosa ülekate

Hulkade A ja B ühisosa $A \cap B$ olulisuse hindamiseks A ja B suhtes on lihtsaimaks ja intuitiivseimaks mõõduks ülekate $c_{A,B}$. Ülekate põhineb ühisosa suuruse ja vastavate hulkade suuruste suhtel. Ülekate on arvutatav järgnevast valemist.

$$c_{A,B} = \frac{|A \cap B|}{|A \cup B|}$$

Arvutuslikult on ilmselt otstarbekam kasutada eelmise valemiga võrdväärset kuju, milles ühendi suurus on avaldatud ühisosa suuruse kaudu.

$$c_{A,B} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Ülekate jääb vahemikku $[0, 1]$ ning annab arvuliselt suuremaid ning semantiliselt olulisemaid tulemusi juhul, kui ühisosa moodustab mõlemast hulgast silmapaistva osa. Semantilise tähenduse all pean silmas sisulisi seoseid andmete vahel, mille leidmiseks on vastavad programmid loodud. Ülekatte tulemus on 0, kui hulkade ühisosa on tühihulk, ning 1, kui hulgad on võrdsed.

Hulgateegis LibSets on ühisosa ülekatte leidmine praktiliselt samaväärne ühisosa leidmisega ning lisatööd teha ei tule, kuna ühisosa arvutamisel on alghulkade ja tulemuseks saadud ühisosa suurused määratud nende salvestusruumiks kasutatud massiivide suurusega. Seega toimib ülekatte arvutamine väga kiiresti.

Ülekatte mõõdu miinuseks on asjaolu, et arvesse ei võeta üksikute hulgaelementide esinemise tõenäosusi hulkades ja terves domeenis. Nii annavad kaks suhteliselt suurt hulka enamasti häid ülekatteid, kuna mõlemas hulgas on märgatav osa kõikvõimalikest elementidest, mistõttu on ka ühisosa elementide arv kõrge. Samas ei pruugi niisugused tulemused olla andmeanalüüsi seisukohalt huvitavad, kuna iga element on eraldi võttes ühisosas tühise osakaaluga. Järgnevas alajaotuses vaatlen alternatiivsete olulisusmõõtude leidmiseks binoomjaotust ja hüpergeomeetrilist jaotust.

4.1.2 Binoomjaotus

Klassikalises tõenäosusteoorias kirjeldatakse binoomjaotust järgneva probleemi abil. Olgu meil antud sündmus, näiteks kirja saamine mündiviskel, ning selle toimumise tõenäosus p ühel katsel. Milline on tõenäosus P_b, et n järjestikusel katsel toimub sündmus täpselt $X = k$ korda? Antud tõenäosus on arvutatav järgnevast

valemist.

$$P_b(X = k, p) = \frac{n!}{k!(n-k)!}p^k(1-p)^{n-k} = C_k^n p^k(1-p)^{n-k}$$

Nagu näha, leitakse tõenäosus kolme komponendi korrutisena, ehk toimunud sündmuste, vastandsündmuste (ehk toimumata sündmuste) ja nende kombinatsioonide korrutisena. Samuti on selge, et üksiku sündmuse tõenäosus püsib konstantsena, see tähendab, et seda ei mõjuta juba eelnevalt toimunud sündmuste ahel.

Binoomjaotuse tõenäosust võib laiendada tervele vahemikule $k \ldots n$. Niisugusel juhul otsime tõenäosust vähemalt k sündmuse toimumiseks. Vahemiku tõenäosust kirjeldab järgnev valem.

$$P_b(X \geq k, p) = \sum_{i=k}^{n} C_i^n p^i(1-p)^{n-i}$$

Hulkade ühisosa olulisuse hindamise ülesandes saab antud tõenäosuse sõnastada järgnevalt. Kui suur on tõenäosus P, et n elemendi korral hulgast A kuulub k elementi ka hulka B, kui suvalise elemendi hulka B kuulumise tõenäosus p on hulga B elementide arvu suhe domeenisuurusesse ehk kõikvõimalike elementide arvu. Sellisel juhul omandavad eelnevas toodud valemi parameetrid alljärgnevad väärtused. Hulga U all on silmas peetud universaalhulka ehk kõiki elemente sisaldavat hulka.

$$n = |A|; \quad k = |A \cap B|; \quad p = \frac{|B|}{|U|}$$

Nagu näha, võetakse käesolevas lähenemises arvesse ka üksiku hulgaelemendi esinemise tõenäosust. Binoomjaotus jääb vahemikku $[0, 1]$ ning analoogselt eelnevale tähistab 0 tühja ühisosa ning 1 võrdseid hulki.

Tulemusi järjestades on olulisemad sellised alamhulgad, mille juhuslikult tekkimise tõenäosus on nullist erinev, kuid võimalikult väike. Siin kasutan seisukohta, et mida väiksem on leitud alamhulga juhuslikult tekkimise tõenäosus, seda tõenäolisemalt on vaadeldavad elemendid omavahel semantiliselt seotud ning seega

on tegemist olulise hulgaga. Sama kehtib ka vastupidi - kui ühisosa tekkimise tõenäosus on suhteliselt suur, näiteks suurte hulkade ühendamise korral, siis vastavate elementide semantiline side on nõrgem ja juhusliku kokkusattumuse tõenäosus suurem.

4.1.3 Hüpergeomeetriline jaotus

Hüpergeomeetrilise tõenäosusjaotuse kirjeldamiseks sobib näiteks alljärgnev ülesanne. Olgu meil urn N kuuliga, millest K kuuli on valged. Milline on tõenäosus P_h, et n kuuli võtmisel saame täpselt $X = k$ valget kuuli ja seega vastavalt $n - k$ muud kuuli? Jaotust kirjeldab järgmine valem.

$$P_h(X = k) = \frac{C_k^K C_{n-k}^{N-K}}{C_n^N}$$

Hüpergeomeetrilist tõenäosust võib samuti laiendada tervele vahemikule $k \ldots n$. Niisugusel juhul otsime tõenäosust vähemalt k valge kuuli saamiseks N kuuli seast, millest K on valged. Vahemiku tõenäosust kirjeldab järgnev valem.

$$P_h(X \geq k) = \sum_{i=k}^{K} \frac{C_i^K C_{n-i}^{N-K}}{C_n^N}$$

Jaotus on sisult sarnane eelnevale binoomjaotusele, kuid erinevus seisneb selles, et iga järgneva sündmuse toimumise või mittetoimumise tõenäosus oleneb juba toimunud sündmuste jadast.Käesolevas alamhulkade olulisuse hindamise ülesandes omandavad valemi parameetrid järgnevad väärtused.

$$n = |A|; \quad k = |A \cap B|; \quad N = |U|; \quad K = |B|$$

Hüpergeomeetriline jaotus on samuti vahemikus $[0..1]$ ning analoogselt on semantiliselt olulisemad väikese nullist erineva esinemistõenäosusega alamhulgad.

Mõnes mõttes on hüpergeomeetrilise jaotuse kasutamine alamhulkade olulisuse hindamisel õigustatum kui binoomjaotuse kasutamine, kuna hüpergeomeetriline tõenäosus arvestab juba ühisosadesse kuuluvaid elemente. Kuna definitsiooni

järgi ei ole hulkade korduvad elemendid lubatud, tundub loomulikum, et järgneva elemendi ühisosasse kuulumise tõenäosus on juba kuuluvate elementide võrra väiksem.

4.2 Juhuslike hulkade headuse hindamine

Eelmises peatükis toodud geeniontoloogiate ja pagaripärmi annotatsioonide analüüs näitab, et peaaegu kõikide geenidega on seotud vähemalt kolm erinevat geeniontoloogiate mõistet, kusjuures olulise osa annotatsioonidest moodustavad koodiga ND tähistatud suurte elementide arvuga tundmatud kategooriad. Samal ajal on umbes 70% kõikidest mõistetest seotud kuni viie geeniga. Sellest järeldub, vähegi pikemate geenipäringute korral saame tulemuseks palju erineva headushinnanguga geeniontoloogia mõisteid, kusjuures ei ole selge, milline headushinnang on oluline ning milline juhuslik.

Järgnevas kirjeldatud arvutuste eesmärgiks on erinevate päringupikkuste jaoks künniste leidmine, mille abil saaks hinnata, millisest piirist alates võivad tulemused olla tekkinud juhuslikult. Künnise olen leidnud kolme headusfunktsiooni jaoks päringupikkustel $1 \ldots 6220$. Selleks olen katsetanud erinevate juhuslike geenihulkadega ning valinud parima tulemuse usalduspiiriga 95%. Arvutuste käik on toodud algoritmiskeemil 19.

Tasub tähele panna, et algoritm leiab parimatest parimad juhuslikud tulemused. Esmalt annab iga kindla pikkusega juhuslikult koostatud geenide hulk üle 2440 geeniontoloogia mõiste mingi arvu tulemusi. Neist valitakse välja parim hinnang. Mitmete juhuslike hulkadega katset korrates saab palju tulemusi, millest omakorda valitakse niisugune, millest vaid 5% juhtudest on juhuslik hulk andnud paremaid hinnanguid.

Tulemuseks saadud künnised on seotud GOST kasutajaliidestega ning katsed näitavad, et piiridest allapoole jäävaid mõisteid välja filtreerides on väheneb kuva-

Algorithm 1 Juhuslike tõenäosuste hindamine

{antud olgu headusfunktsioon, ontoloogiafail, usalduspiir, katsete arv}

Require: $\exists gofile, \exists function, \exists percent, \exists repeat$;

{iga antud pikkuse jaoks}

for $size = 1$ to $repeat$ **do**

$bestresults \Leftarrow \emptyset$;

{korrata erinevate juhuslike hulkadega}

while $i < 1000$ **do**

{luua antud pikkusega juhuslik hulk}

$randset \Leftarrow create\_random(size)$;

{leida kattuvad ontoloogiamõisted, nendest parim tulemus}

$results \Leftarrow gostminer(gofile, randset, function)$;

$result \Leftarrow find\_best(results)$;

$result \rightarrow bestresults$;

end while

{leida tulemustest parim, arvestades usalduspiiri}

{sorteerida tulemused parimad ettepoole, vastavalt headusfunktsioonile}

$sort(bestresults, function)$;

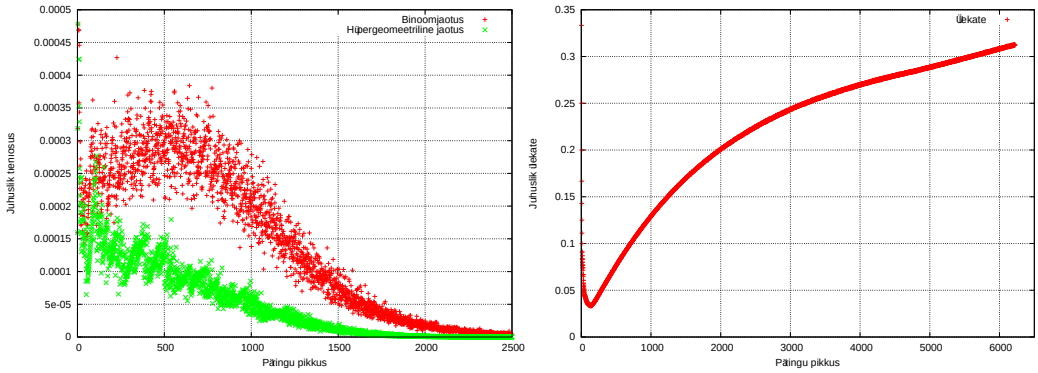
{leida usalduspiirile vastav tulemus}

$index \Leftarrow repeat - repeat/100 * limit$

$print(size, bestresults[index])$;

end for

tavate tulemuste arv märgatavalt, eriti just üldisemate ja arvukamalt annoteeritud mõistete arvelt. Künniste graafikud on toodud joonisel 19.



Joonis 19: Juhuslike hulkade künnised tõenäosuste (vasakul) ja ülekatte jaoks

Graafikute juures tuleb esmalt tähele panna ülekatte ja tõenäosusmõõtude sisulist erinevust - ülekatte korral on mittejuhuslikud ehk olulised tulemused künnisest arvuliselt suuremad, samas kui tõenäosusfunktsioonide korral on olulised tulemused, mille korral on funktsiooni väärtus väiksem kui toodud lävi. Seega on graafikud sisu poolest horisontaaljoone suhtes peeglis.

On näha, et tõenäosushinnangute ja ülekatte funktsioonid on kujult sarnased, kuid tõenäosused sisaldavad eriti just ekstreemumpiirkonnas rohkelt müra, s.t erinevusi lähinaabrite tulemustes. Niisiis tasub geenipäringute koostamisel proovida kergelt varieeruvate päringupikkustega.

Väga kõrged ülekatted ja madalad tõenäosused päringupikkustel $1 \ldots 20$ on põhjendatavad geeniontoloogiate ja pagaripärmi genoomi struktuuri eripäradega, kus 70% pagaripärmiga seotud mõistetest kirjeldavad viit või vähemat geeni. Seega omandavad väga lühikesed päringud ka juhuslikult suuri hinnanguid. Järelikult ei pruugi lühikesed päringud anda adekvaatseid tulemusi ja huvi võiksid pakkuda eelkõige suuremad päringud veidi üldisemate kategooriate leidmiseks.

4.3 Hulkade kaevandamise programm GOST

Käesoleva bakalaureusetöö olulisemaks praktiliseks rakenduseks on geeniontoloogiate kaevandamise vahend GOST. Nimetus ‘GOST‘ on akronüüm sõnadest *Gene Ontologies STatistics*.

Programmi GOST sisendiks on geenidest koosnev päring ning väljundina tagastatakse geeniontoloogiate mõisted, mille annotatsioonidel on ühisosa antud geenipäringuga. Põhimõtteliselt on programm rakendatav mistahes sisuga alamhulkade kaevandamiseks.

Tehniliselt koosneb GOST mitmest omavahel seotud moodulist, mis on kirjutatud keeltes Perl ja C. Andmehulkade kaevandamiseks ja kogu süsteemi tuumaks on C programm GOSTminer, mis kasutab hulgaoperatsioonide teeki LibSets [Rei04].

Skeem 2 kirjeldab süsteemi tuumaks oleva algoritmi funktsionaalsust.

Algoritmi GOSTminer realiseerimise juures on lähtutud peamiselt kompaktsuse ja kiiruse nõuetest. Algoritm kasutab kaevandamiseks täisarvuhulki, mida säilitatakse BinSets-formaadis failides. Etteantud päring tuleb samuti eelnevalt täisarvuhulgaks teisendada. Nagu eelnevas näidatud, annab binaarkujul hulkade kasutamine võitu nii töökiiruses kui salvestusruumis.

Eelnimetatud põhjustel on GOSTminer’i mugavamaks kasutamiseks kirjutatud käsurea- kui ka veebiliides ning programm indeksstruktuuride loomiseks.

4.3.1 Käsurealiides GOST in the Shell

Hulkade kaevandamise programmi GOSTminer kasutamiseks käsureal on kirjutatud programm *GOST in the Shell*. Käsureaprogrammi loomisel on silmas peetud võimalust paljude päringute automaatseks sooritamiseks skriptide abil ning tulemuste suunamist torudega.

Programm on realiseeritud Perl moodulina `Gost.pm`. Programmi sisendiks

Algorithm 2 GOSTminer oluliste ühisosade leidmine

{Hulgafailid peavad olema loetavad, antud peab olema hinnangufunktsioon}

Require: $\exists queryfile, \exists gofile, \exists function$;

{Lugeda päring ja GO annotatsioonid failidest}

$geneset \Leftarrow read\_binsets(queryfile)$; $gosets \Leftarrow read\_binsets(gofile)$;

$gonumber \Leftarrow 0$;

for all go, $go \in gosets$ **do**

{Iga annotatsioonihulga jaoks leida ühisosa geenipäringuga}

$subset \Leftarrow intersect(geneset, go)$;

if $\exists subset$ **then**

{leida parameetrid hinnangufunktsioonile, arvutada ühisosa olulisus}

$n \Leftarrow size(geneset)$; $k \Leftarrow size(subset)$;

$N \Leftarrow domain\_size(gosets)$; $K \Leftarrow size(goset)$; $p \Leftarrow K/N$;

{Hinnangufunktsioon on valitud vastavalt algparameetrile}

if $function$ is $BINOM$ **then**

$ranking \Leftarrow binomial\_probability(n, k, p)$;

else if $function$ is $HYPGM$ **then**

$ranking \Leftarrow hypergeometrical\_probability(n, k, N, K)$;

else

{Vaikimisi leida ülekate}

$ranking \Leftarrow cover(go, geneset, subset)$;

end if

{Trükkida leitud ühisosad väljundisse}

print go, $ranking$;

end if

$gonumber++$;

end for

on geeninimedest koosnev päring, mis võib olla antud nii failina kui stringina käsurealt. Käsurealiidese peamised ülesanded on toodud järgnevas loetelus.

1. Geenipäringu ettevalmistus ja teisendamine BinSets formaati;
2. Programmi GOSTminer käivitamine ja saadud tulemuste sisselugemine;
3. tulemuste järjestamine vastavalt headusfunktsioonile;
4. tulemuste sidumine vastavate GO nimetustega;
5. juhuslike hulkade headuse piiri määramine vastavalt päringupikkusele;
6. tulemuste kuvamine standardväljundisse.

Oluliseimaks käsureaparameetriks on headuse funktsiooni valimine viida `-r` abil. Vaikimisi väärtuseks on ülekate `c`, valida saab hüpergeomeetrilise `-rh` ja binoomtõenäosuse `-rb` vahel.

Programmi väljundisse tuuakse ridadena leitud GO mõisted. Esimesel kohal on leitud GO mõiste nimetus, järgneb geenide päringule vastav string. Stringile järgneb saadud tulemuse headusfunktsiooni väärtus. Kui väärtusele eelneb tilde `,. ,.o~o~`, siis on sellise väärtusega hulki saadud ka juhuslikult ning järelikult ei ole tulemus oluline. Viimased kaks numbrit näitavad selle mõistega seotud geenide arvu (mõiste hulga suurust) ning päringu ja mõiste ühisosa suurust.

```
GO:0005737 -?--S-M---?-D?-?DA-?-D-D-?-  0.122141 1303 7
```

Stringis tähistab iga sümbol üht päringu geeni. Tähed `O`, `A`, `a`, `D`, `G`, `S`, `X`, `E`, `M`, `P`, `C`, `R` näitavad, et leitud on geeni ja mõiste vaheline seos, ning tähistavad erinevaid tõendikoode. Sümbol `-` märgib seose puudumist. Sümbol `?` näitab, et vaadeldavat geeni ei ole üldse annoteeritud. Sümboli # abil on tähistatud mitme erineva tõendikoodi leidmine.

Programmis on võimalik väljundi taset muuta. Niinimetatud jutukas väljund (ingl. k. *verbose mode*) aktiveeritakse viidaga `-v` ning selles kuvatakse mitmesugust lisainfot programmi töö käigu kohta. Kompaktne väljund `-c` jätab tulemustes kuvamata geenidele vastava stringi.

Programmis on loodud võimalused mitmete annotatsioonikomplektide poole pöördumiseks käsureaviida `-g` abil. Nii on võimalik tulevikus laieneda ka teiste organismide genoomide või laiemalt suvaliste hulkade kaevandamiseks. Annotatsioonide asukohad ja identifikaatorid asuvad vastavas indeksis. Praeguseks kasutatavateks väärtusteks `-g1` ja `-g2`, mis viitavad kahele erineva kuupäevaga SGD annotatsioonifailile.

Programm GOST kasutab täisarvuhulkade ja objektide sidumiseks indeksifaile, milles on toodud geenide ja ontoloogiamõistete identifikaatorid. Praeguses relatsioonis on selleks kasutatud harilikke tekstifaile, kuid edasistes arendustes on plaanis üle minna andmebaasile. Indeksite ettevalmistamiseks olen loonud komplekti Perl skripte *Annotations2Binary*.

4.3.2 Veebiliides GOST on the Web

Käsureaprogramm *GOST on the Shell* on mugav mitmesuguste programmeeritavate ülesannete lahendamiseks. Samal ajal ei ole tekstipõhine väljund parim visuaalseks ülevaateks, mis on suurte andmehulkade korral väga oluline.

Mugavamaks interaktiivseks kasutamiseks olen loonud veebiliidese *GOST on the Web*, mille graafiline vorm annab tulemustest parema visuaalse ülevaate. Veebiliidese ekraan mahutab rohkem infot ning erinevate värvide kasutamine väljundis võimaldab kiiremini tabada tulemuste trende. Veebiliidese peamised täiustused võrreldes käsurealiidesega on toodud järgnevas loetelus.

- Tõendikoodide filtreerimine väljundis;

- Väljastavate ridade filtreerimine vastavalt headusfunktsioonile;
- Viit kirjeldusele GO mõistete andmebaasi AmiGO [ami05];
- Mõiste kauguse kuvamine juurtipuni;
- Mõiste domeeni ja nimetuse kuvamine;
- Teksti-, kompakt- ja graafikaväljundi valimise võimalus.

Veebiliides on loodud keeles PHP (versioon 4.3.10) [php05] MySQL andmebaasi (versioon 4.1.11) ja Apache HTTP veebiserveri (versioon 2.0.54) [apa05] kihtide peale. Valitud tehnoloogia eeliseks on eelkõige vabavaralisus ja arendusprotsessi kiirus.

Veebiliides kutsub välja käsureaprogrammi ning kasutab selle väljundit. Seega ei ole kahe GOST'i väljundites suuri erinevusi. Veebiliideses on rea algusesse lisatud mõiste kaugus juurtipust (näiteks `(L7)`). Rea lõppu on lisatud mõiste domeeni tähistav sümbol (`MF`, `BP`, `CC`) ning mõiste nimetus. Olulised tulemused, mille hinnangud ületavad juhuslike hulkade hinnangut, on väljundis toodud tumedama raamiga.

Saadud mõisteid on võimalik filtreerida vastavalt headusfunktsiooni väärtusele ja esinenud tõendikoodidele. Headusfunktsiooni filtri jaoks on võimalik valida erinevaid juhuslike hulkade headuse piiriga seotud väärtusi. Esinenud tõendikoodide järgi filtreerimine toimib praeguses versioonis mittevälistaval põhimõttel. See tähendab, et mõiste kuvatakse, kui sellega on seotud vähemalt üks soovitud tõendikoodidest, teiste tõendikoodide esinemine antud real filtrit ei mõjuta.

5 Geenieksпressioonide maatriksi kaevandamine

Käesoleva bakalaureusetöö viimases peatükis tutvustan geeniontoloogiate analüüsiks loodud vahendite ja analüüsitulemuste toel valminud arvutusi, mille eesmärgiks on pagaripärmi geenidest koostatud ekspressioonimaatriksil oluliste geenigruppide ja ridade arvutamine.

Siinkohal toon välja hüpoteesi, et sarnase ekspressiooniga ehk avaldumismustriga geenidel võib olla teisigi ühiseid omadusi. Seega teades, et ekspressioonimaatriksi näol on tegemist bioloogiliselt korreleeruvate geenihulkadega, võib maatriksist erinevaid alamhulki kombineerides ning geeniontoloogiatega võrreldes õppida ontoloogiate statistilise struktuuri kohta.

Teiseks on huvitav uurida ekspressioonimaatriksi ridade olulistust geeniontoloogiates erinevate pikkuste kaupa. Maatriksis on iga rea esimeseks elemendiks üks geen, järgnevad geenid on kahaneva sarnasuse järjekorras. Seega sisaldab maatriks mitmesuguse tihedusega ridu, kusjuures geeniontoloogiate abil saaks ridu võrrelda ja hinnata, millised maatriksi read on paremad.

Peatüki esimeses pooles tutvustan sisendandmete iseloomu ja nende saamiseks sooritatud eksperimenti [VBJ+00, BV00]. Töö teises pooles kirjeldan GOSTminer'i abil läbi viidud arvutusi ning tulemuseks saadud olulisi geenigruppe. Analüüsiks olen kasutanud paralleelarvutusi, millest kirjutan käesoleva bakalaureuse töö lisas 1.

5.1 Katseandmed pärmi genoomil

Varasemas hulgaoperatsioonide peatükis olen korduvalt maininud ja võrdlemiseks kasutanud bioloogiliselt korreleeritud täisarvuhulki. Olles andnud sissejuhatuse geeniontoloogiate ning pagaripärmi genoomi andmebaasi olemusse, on aeg lähemalt tutvustada Vilo, Brazma, Jonasseni, Robinsoni ja Ukkoneni [VBJ+00] poolt

läbi viidud analüüsi pagaripärmi genoomil ja selle tulemusena saadud eksperimentaalseid andmeid, mida olen kasutanud bakalaureusetöö peamise andmehulgana.

Erinevatel katsetingumustel läbi viidud geenide avaldumise mõõtmiste abil saab defineerida vastavate geenide ekspressiooniprofiili. On püstitatud hüpotees, et sarnaste ekpressiooniprofiilidega geenid ehk geenid, mis avalduvad koos, võivad omada sarnaseid regulatsioonimehhanisme. Eukaürootsete organismide geeniregulatsioonil on oluline roll kindlatel valkudel, niinimetatud transkriptsioonifaktoritel, mis seonduvad promootorregioonides DNA külge ja mõjutavad geeni transkriptsiooni. Pagaripärmi korral seonduvad transkriptsioonifaktorid enamasti geeni vahetusse lähedusse, tüüpiliselt kuni tuhande aluspaari võrra ülesvoolu enne geeni algust.

Vilo et al. demonstreerivad oma töös oluliste geeniregulatsioonist märku andvate kandidaatmustrite leidmist pagaripärmi genoomis. Automatiseeritud analüüs koosneb üldjoontes geenide grupeerimisest nende ekspressiooniprofiilide järgi, sekventsimustrite otsimisest saadud geenigruppide võimalikes promootorpiirkondades, leitud oluliste sekventside grupeerimisest ning võrdlemisest olemasolevate andmebaasikirjetega.

Sekventside analüüs ning kandidaatmustrite tuvastamine jääb käesoleva töö ulatusest välja, seega ei uuri ma edasises geenide regulatsioonimehhanisme, vaid vaatlen sarnasust üldisemas mõttes. Laiendan püstitatud hüpoteesi, oletades, et sarnase ekspressiooniprofiiliga geenidel on tõenäoliselt teisigi sarnaseid jooni, see tähendab, et geenidele võib olla annoteeritud samasid või sarnaseid geeniontoloogiate mõisteid. Järelikult pakuvad edasises huvi eksperimendi esimene etapp ja selle tulemused, s.t. ekspressiooniprofiilide järgi grupeeritud geenid.

5.2 Pärmi geenide klasteranalüüs

Vilo, Brazma, Jonasseni, Robinsoni ja Ukkoneni [VBJ+00] poolt läbi viidud analüüsi sisendiks on avalikult kättesaadavad tulemused Stanfordi Ülikoolis sooritatud pagaripärmi geenide ekspressiooni eksperimentidest [ESBB98]. Andmehulk koosneb 6221 pagaripärmi geenist, millega on seotud ekspressioonitasemed 80 erineva eksperimentaalse tingimuse lõikes.

Grupeerimisel on kasutatud k-keskmise järgi klasterdamist, millest saab hea ülevaate Meelis Kulli klasteranalüüsi puudutavast magistritööst [Kul04]. Klasterdamiseks nimetatakse üldiselt andmete või objektide grupeerimist klassidesse nii, et iga klassi ehk klastri objektid on omavahel väga sarnased ning teiste klastrite objektidest võimalikult erinevad. Objektidevahelised erinevused põhinevad objektide atribuutide väärtustel ning avaldatakse kauguste mõõtudena.

Klasterdamine k-keskmise järgi on üheks lihtsamaks ja levinumaks meetodiks. Algoritm vajab sisendiks klastrite arvu k ning jaotab n objekti mingi sarnasusmõõdu d alusel k klastri vahel nii, et klastrisisene objektide sarnasus oleks võimalikult suur, terviklike klastrite omavaheline sarnasus aga võimalikult väike. Algsed klastrid määratakse sisendandmete hulgast juhuslikult või mingi valikukriteeriumi alusel. Oluline on märkida, et algoritmile tuleb ette anda klastrite arv k. K-keskmise klasterdamise algoritm koosneb kokkuvõtvalt kolmest osast.

1. Leida iga objekti jaoks kaugused kõikide klastriteni. Lisada objekt klastrisse, mille kaugus on vähim.

2. Leida igale klastrile uus keskpunkt, milleks saab selle klastri objektide ariaritmeetilineskmine.

3. Korrata, tagasi 1. punkti juurde.

Algoritmi kolmas samm ei ole kohustuslik, kuid selle abil on võimalik saa-

dud tulemuste kvaliteeti parandada. Reeglina rakendatakse iteratsioonisammu seni, kuni klastrid enam ei muutu.

Geenide klasterdamisel ekspressioonitasemete järgi on sarnasusmõõduna kasutatud Eukleidese kaugust. Kuna geenidega on seotud 80 erineva katse käigus saadud ekspressioonitasemed, võib igat geeni vaadata kui vektorit 80-mõõtmelises ruumis. Eukleidilise kauguse d arvutamiseks n-mõõtmelises ruumis punktide x ja y vahel sobib järgmine valem.

$$d_{x,y} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

On teada, et klasterdamise tulemus ja kvaliteet oleneb suuresti valitud meetodist ja algparameetrite väärtustest. Vilo et al on oma eksperimendis kasutanud k-keskmise algparameetri k varieerimist, uurinud paljusid klastreid ja algalgsete tsentroididenfiguratsioone paralleelselt ning hinnanud saadud klastrite headust formaalsel teel.

Objekti i klastrisse A klassifitseerimise headust saab mõõta siluettväärtusega, mida on kirjeldanud Rousseeuw allikas [Rou87]. Siluettväärtus $s(i)$ on defineeritud järgnevalt. Olgu $d(i,j)$ kaugus objektide i ja j vahel ning $|A|$ klastrisse A kuuluvate objektide hulk. Siis objekti i jaoks klastris A saab defineerida tema keskmise kauguse $a(i)$ teistest sama klastri objektidest.

$$a(i) = \frac{1}{|A|-1}\sum_{j \in A, j \neq i} d(i,j)$$

Tähistagu $d(i,C)$ objekti i kaugust klastrist A erineva klastri C jaoks ning $b(i)$ objekti i keskmist kaugust lähima teise klastri elementideni.

$$d(i,C) = \frac{1}{|C|}\sum_{j \in C} d(i,j)$$

$$b(i) = min_{C \neq A}\{d(i,C)\}$$

Objekti i siluettväärtus $s(i)$ defineeritakse järgnevalt.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

Objekti i siluettväärtus on vahemikus $[-1; 1]$. Kui $s(i) = 1$, siis on objekt i hästi klaklassifitseeritudui $s(i) < 0$, siis on i halvasti klassifitseeritud ning sarnaneb pigem mõne teise klastri objektidele. Kõikide klastri objektide keskväärtust saab klasutad klastri headuse hindamisel. Keskmine siluettväärtus iseloomustab hästi nii vaadeldava klastri tihedust kui tema elementide kaugust lähimast klastrist.

Vilo et al eksperimendis on k-keskmise parameetrit k varieeritud $2 \ldots 1000$ ning korratud vastavaid katseid 10 korda juhuslike hulkadega. Saadud klastritele on arvutatud siluettväärtused ning selle abil valitud välja parimad klastrid.

Edasises kirjeldatud arvutuste sisendiks on niisiis pagaripärmi geenidest koosnev maatriks, milles iga rea esimeseks elemendiks on järjekordne pagaripärmi geen. Sellele järgnevad 6220 geeni on järjestatud esimese geeni suhtes mittekahaneva kauguse järgi. Kaugused on arvutatud eelnevas kirjeldatud Eukleidese mõõdu järgi klasterdamise tulemustest. Maatriks koosneb 6221 reast ja veerust, kusjuures paljud ridadest on omavahel tugevas korrelatsioonis. Maatriksit veergude kaupa alamhulkadeks tükeldades saab erineva headusega klastreid, mida on antud töös korduvalt rakendatud hulkade analüüsiks ja optimaalsemate tõenäosusparameetrite hindamiseks.

5.3 Ekspressioonimaatriksi kaevandamine

Selles alajaotuses kirjeldan eksperimenti, milles on rakendatud programmi GOST ja eelnevate analüüside tulemusi. Analüüs on küllalt töömahukas ning selleks on kasutatud paralleelarvutuste meetodeid ja klient-server arhitektuuri, millest annan lühiülevaate töö lisas 1.

Vaadeldava analüüsi sisendiks on Vilo et al uurimuses [VBJ+00] kasutatud pagaripärmi geenidest koosnev $6221 * 6221$ maatriks. Maatriksi iga rida on järjestatud esimese elemendi ekspressiooni sarnasuse järgi järgnevate elementide suhtes. Niisugune maatriks sisaldab ridades mitmesuguse pikkuse ja statistilise olulisusega geenigruppe.

5.3.1 Kaevandamise meetod

Analüüsi eesmärgiks on ekspressioonimaatriksist selliste geenigruppide leidmine, mis annaksid geeniontoloogiate mõistete suhtes võimalikult huvitavaid tulemusi. Täpsemalt on ekspressioonimaatriksist otsitud igale pagaripärmiga seotud geeniontoloogiate mõistele grupp, mille headushinnang on parim.

Parimaid hinnanguid olen otsinud nii hüpergeomeetrilise jaotuse, binoomjaotuse kui ka ülekatte osas. Seega on tulemusteks geenigrupid, mille esinemise tõenäosus on võimalikult väike või mille ülekate mõne geeniontoloogia mõistega on võimalikult suur.

Maatriksist geenigruppide valimiseks olen kasutanud lihtsat reapikkuse suurendamise taktikat. Kuna on teada, et read on bioloogiliselt järjestatud, on selge, et parima korrelatsiooniga read saavad olla ainult alguses ning ridade järjekorra kombineerimiseks pole vajadust. Alustan otsingut ridadest, mis sisaldavad vaid esimest elementi, seejärel vaatlen esimese ja teise elemendiga ridasid, jne.

Skeem 3 annab ülevaate töö käigust. Siinkohal on tegemist lihtustatud versiooniga, mis jätab kirjeldamata paralleelse töötluse ja ekspressioonimaatriksi tükeldamisega seotud detailid.

5.3.2 Kaevandamise tulemused

Analüüsi tulemusel on saadud 2431 geeniontoloogiate mõistele vastavat parimat gruppi. Kokku on uuritud annotatsioonifaili märgitud seoseid 2540 mõistega, kuid

Algorithm 3 Ekspressioonimaatriksi kaevandamine

{antud olgu ontoloogiafail ja soovitud headusfunktsioon}

Require: $\exists gofile, \exists func$;

$bestgroups \Leftarrow \emptyset$; {Andmestruktuur parimate säilitamiseks, alguses tühi}

{Vaadelda kõikvõimalikke reapikkusi}

for all $linelength$, $linelength \in \{1 \ldots 6220\}$ **do**

$submatrix \Leftarrow getmatrix(linelength)$;

{leida kattuvad ontoloogiamõisted GOSTminer programmiga}

$results \Leftarrow gostminer(gofile, submatrix, func)$;

{iga leitud tulemuse jaoks}

for all r, $r \in results$ **do**

$(r.goid, r.rank) \Leftarrow r$; {tulemus koosneb goid ja hinnangu atribuutidest}

{lisada koos parimate hulka, kui seni ei ole mõistele parimat leitud}

if $\nexists bestgroups[r.goid]$ **then**

$(goid, rank, linelength) \rightarrow bestgroups$;

else

$oldr \Leftarrow bestgoups[r.goid]$; {võrrelda, kui mõistega on seotud tulemus}

{suurem ülekate on parem, väiksem tõenäosus on parem}

if $oldr.rank \leq r.rank \cap func$ is $COVER$ **then**

$(r.goid, r.rank, linelength) \rightarrow bestgroups$;

else if $oldr.rank \geq r.rank \cap (func$ is $BINOM \cup func$ is $HYPGM)$ **then**

$(r.goid, r.rank, linelength) \rightarrow bestgroups$;

end if

end if

end for

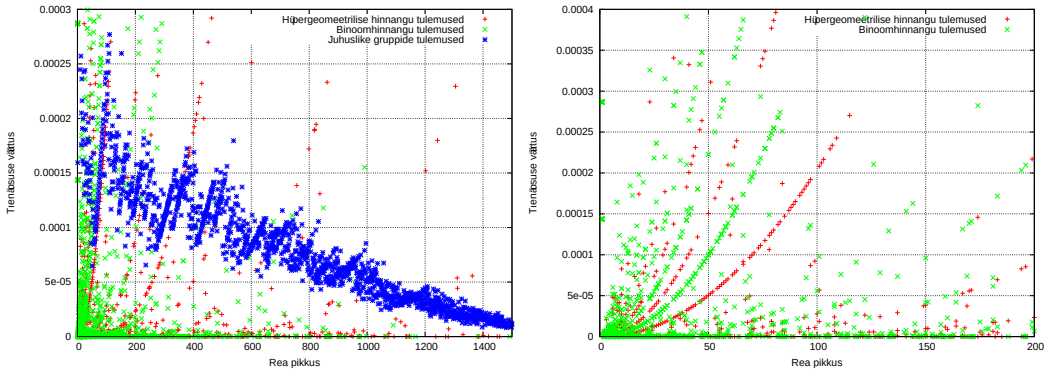
end for

print $bestgroups$;

puuduvate mõistete jaoks ei leitud ühtegi sobivat ahelat. Põhjus on ilmne - ekspressioonimaatriksi andmed pärinevad mitme aasta tagusest uurimusest ning sellest ajast on geeninimetusi lisatud ja kustutatud. Mõisted on leitud kokku 1652 erinevast reast.

Joonisel 20 on toodud analüüsi esmased tulemused maatriksist päritud ridade ja vastavate hüpergeomeetrilise ja binoomjaotuse hinnanagute järgi. Vasakpoolsel joonisel on toodud ka eelmises peatükis leitud juhuslike hulkade hüpergeomeetrilise hinnangu künnis. Joonisele on selguse ja kompaktsuse mõttes kandmata jäetud binoomjaotuse künnis, kuna hüpergeomeetriline künnis on rangem ja annab tulemustest piisava info.

Künnise asetsemine demonstreerib saadud tulemuste üldist kõrget taset, enamik saadud tulemustest on oluliselt parema hinnanguga kui sama pikkusega juhuslikud hulgad.



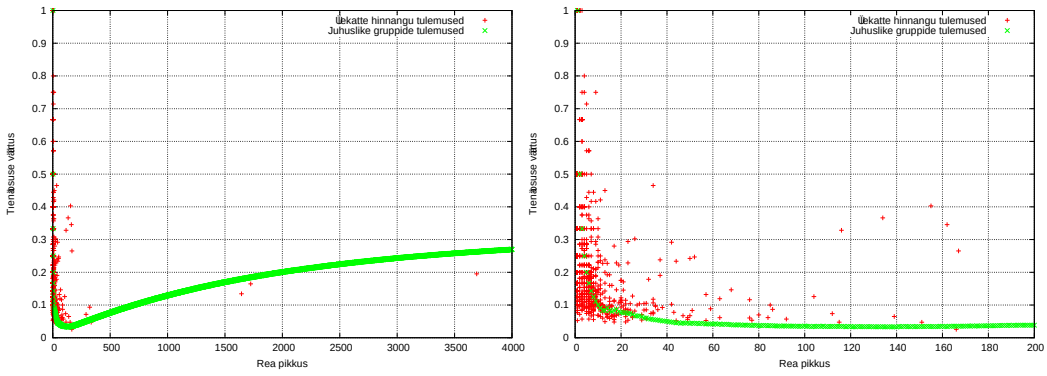
Joonis 20: Ekspressioonimaatriksi parimad tulemused geeniontoloogiatelt

Parempoolsel graafikul on toodud vasakpoolse graafiku huvipakkuvam vahemik päringu pikkust tähistava x-telje algusosast, kuhu on koondunud lõviosa tulemustest. Näha on huvitavad korrelatsioonid tõenäosuse ja päringupikkuse vahel - moodustunud on mitu erineva tihedusega jada, mille punktid paiknevad tõenäosuse ja pikkuse suhtes järjestikku. Esmapilgul tuleks niisugust korrelatsioo-

ni seostada eelkõige sisendandmete iseloomuga. On selge, et tihedamad geenide järjestused korduvad paljudes erinevates ekspressioonimaatriksi ridades. Antud korrelatsioone tuleks edasises lähemalt uurida.

Korrelatsioonid esinevad nii hüpergeomeetrilise kui binoomjaotuse juures, kuid alati on vastavad jadad teineteise suhtes pikkuseteljel nihkes, reeglina on hüpergeomeetrilise tõenäosuse järgi järjestatud ahelad tihedamad ning suuremate pikkustega.

Edasises analüüsis olen selguse mõttes uurinud hüpergeomeetrilise hinnangu järgi saadud tulemusi, võrreldes neid vajadusel teiste hinnangute tulemustega. Hüpergeomeetrilise hinnangu eelistamist kinnitab ka jaotuse statistiline sisu, mis arvestab hulga tõenäosuse arvutamisel juba eelnevalt hulka lisatud elementide tõenäosust, välistades nii korduvate elementide kaasamist.



Joonis 21: Ekspressioonimaatriksi parimad tulemused geeniontoloogiatelt

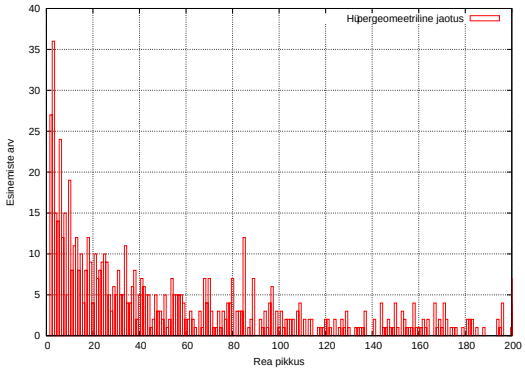
Võrdlevalt olen joonisel 21 toonud katse tulemused ekspressioonimaatriksi kaevandamisel ülekatte mõõdu järgi. On näha, et ülekatte järgi järjestatud parimad tulemused on üldiselt palju lühema pikkusega ning koonduvad juhusliku künnise ümber. Järeldusena võib öelda, et puhtalt ülekatte mõõdu järgi saadud tulemused ei paku edasises analüüsis huvi.

Ligikaudu pooled leitud gruppidest, täpsemalt 1205 gruppi, koosnevad vaid

ühest geenist. Niisuguseid gruppe ei ole edasises mõtet uurida, kuna teame, et iga geeni jaoks leidub rida, mis algab selle geeniga. Järelikult on ühe geeniga ontoloogiamõistete parimad read juba ette teada.

Juhuslikest hulkadest saadud tõenäosustega võrreldes osutus ebaoluliseks 448 leitud gruppi. Seega on ühest suuremate oluliste gruppide arvuks 1046.

Joonisel 22 on toodud oluliste hulkade pikkuste sagedustabelid kuni pikkuseni 200.



Joonis 22: Pikkuste sagedustabel

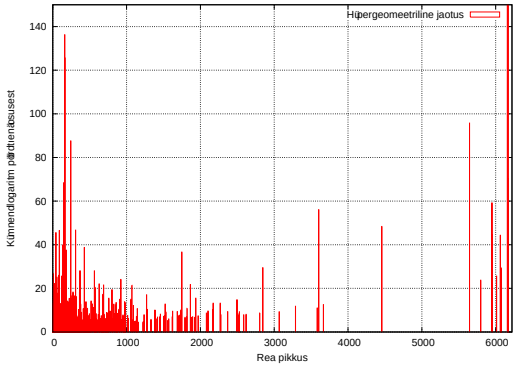
Sagedustabelite esimene ots on ootuspärane ja vastab geeniontoloogiate struktuurile, milles enamiku mõistetega on seotud paar-kolm geeni. Tundub loomulik, et parima headushinnangu annab sarnaselt lühike geenigrupp.

5.3.3 Tulemuste hindamine ülekatte abil

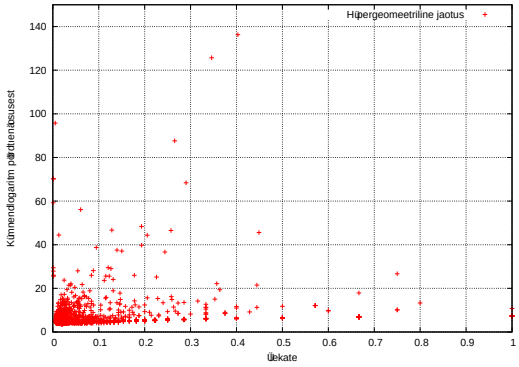
Graafikul 23 on kujutatud leitud gruppide tõenäosuste jaotust. Kuna headushinnangud on suuresti erinevad, on selgemaks ülevaateks esitatud pöördtõenäosuste kümnendlogaritm $log_{10}\frac{1}{p}$. Välja on jäetud ekstreemumid reapikkustega 6167 ja 6159 (vastavad p väärtused $3.64449 * 10^{-297}$ ja $2.74829 * 10^{-199}$). Viimatimainitud tulemused ei ole ilmselt olulised, kuna hõlmavad pea kõiki maatriksi geene ning seega sobivad enamiku maatriksi ridadega.

Märgatav osa 1046-st olulisest tulemusest on väga heade tõenäosustega. Nagu võib näha eelmises lõigus toodud headushinnangute graafiku ekstreemumväärtustest, ei pruugi ainult ühe tõenäosushinnangu järgi järjestamine tingimata õige olla.

Selleks, et leitud geenigruppidest huvitavamaid välja otsida, toon uuesti sisse ülekatte mõõdu ning leian ülekatted mõistete ja nende hüpergeomeetrilise hinnangu järgi leitud parimate gruppide vahel. Graafikul 24 on toodud tõenäosuse pöördväärtuse logaritmi ja ülekatte jaotus. Ülekatte mõõt kirjeldab hästi ja intuitiivselt leitud gruppide ja ontoloogiamõistete sobivust.



Joonis 23: Tõenäosused ja pikkused



Joonis 24: Tõenäosused ja ülekatted

Parempoolselt graafikult pakuvad ilmselt huvi tihedamast alast välja jäävad grupid, mille ülekate ja tõenäosuse pöördväärtus on võimalikult suured. Valik niisuguseid gruppe on toodud tabelis 9. Read on järjestatud tõenäosuse ja ülekatte järgi, eelduseks on vähemalt kümment geeni sisaldav rida. Valitud on 25 esimest tulemust.

goid	pikkus	Tõenäosus	1.geen	ülekate	ühisosa
GO:0003735	162	5.28994e-137	YGR027C	0.40293	110
GO:0006412	162	2.013e-126	YGR027C	0.345794	111
GO:0005842	243	2.23428e-88	YJL190C	0.265385	69
GO:0005843	147	3.90294e-69	YLR029C	0.290123	47
GO:0006511	89	3.01449e-47	YGL011C	0.257353	35
GO:0004175	42	2.60781e-46	YGR253C	0.44898	22
GO:0006096	22	6.96704e-23	YDR240C	0.357143	10
GO:0006094	10	4.12932e-20	YGL011C	0.363636	8
GO:0003678	21	6.38808e-17	YDR545W	0.258065	8
GO:0005656	15	8.83445e-16	YEL032W	0.352941	6
GO:0000722	21	1.25333e-15	YDR545W	0.259259	7
GO:0004003	15	6.6087e-15	YEL032W	0.315789	6
GO:0006268	15	2.90031e-14	YEL032W	0.285714	6
GO:0046933	13	5.37931e-14	YBL099W	0.272727	6
GO:0019773	10	2.14348e-12	YGR253C	0.4	4
GO:0006616	14	3.67497e-12	YJL034W	0.263158	5
GO:0005749	12	5.0525e-12	YOR065W	0.333333	4
GO:0006121	12	5.0525e-12	YOR065W	0.333333	4
GO:0000256	10	3.20969e-11	YIR027C	0.333333	4
GO:0005100	10	2.68917e-10	YDR431W	0.266667	4
GO:0006081	10	4.47808e-10	YJR155W	0.25	4
GO:0005756	11	2.93129e-09	YGL188C	0.272727	3

Tabel 9: Valik olulisi geenigruppe ekspressioonimaatriksist

6 Kokkuvõte

Andmekaeveks nimetatakse protsessi, mille eesmärgiks on suurtest andmekogudest mittetriviaalse ja senitundmatu informatsiooni leidmine. Andmekaeve meetodite tähtsaks rakendusvaldkonnaks on geneetika.

Geeniontoloogiate Konsortsium on projekt, mis haldab geenidega seotud mõistete hierarhilist struktuuri. Ontoloogiates sisalduv informatsioon pakub mitmeid mahukaid arvutuslikke ülesandeid.

Üheks huvitavaks ülesandeks on etteantud geenide grupile niisuguste funktsionaalsust kirjeldavate mõistete leidmine, mis iseloomustaks antud gruppi kõige paremini. Leitud mõistete abil on võimalik tundmatuid geene ennustavalt iseloomustada või geenidevahelisi seoseid hinnata.

Käesolevas bakalaureusetöös on uuritud ontoloogiate kaevandamiseks sobivaid hulgatehetel põhinevaid kiireid algoritme ja andmestruktuure ning võrreldud erinevate realisatsioonide efektiivsust. Töö tulemusel on loodud programmide kogu GOST, millega saab geeniontoloogiatest leida geenigruppidele olulisi seoseid.

Töö teises pooles on uuritud geeniontoloogiate ja pagaripärmi *Saccharomyces cerevisiae* annotatsioonide statistilist iseloomu. Valminud töövahendite abil on sooritatud seeria arvutusi, mis võimaldavad leitud mõistete olulisust paremini hinnata.

Samuti on GOST programme rakendatud pagaripärmi ekspressioonimaatriksist oluliste seoste kaevandamiseks. Tulemusena on saadud tuhatkond väga kõrgete headushinnangutega geenigruppi.

Tehtud töö edasiarendusena tuleks uurida senisest võimsamat oluliste tulemuste eristamist ning tulemuste mitmekesistamiseks siduda programmidega erinevaid headuse hindamise viise. Teiseks tähtsaks jätkuprojektiks on ekspressioonimaatriksilt leitud oluliste geenigruppide võrdlemine ja klasteranalüüs.

Estimating gene functionality through ontologies

Bachelor thesis

Jüri Reimand

Abstract

Data Mining is the process of finding nontrivial and potentially unknown information from large amounts of data. One important application field of Data Mining is genetics.

The Gene Ontologies Consortium maintains structured vocabularies from the domain of genetics. The information stored in the ontologies introduces a number of interesting computational tasks.

One important task of this kind involves finding terms in the vocabularies that best describe a group of genes. Through matching statistically relevant features that are common to the gene group as a whole, we may be able to determine unknown functionality of single members of the group.

In this bachelor's thesis, we first study and compare fast set-based algorithms and structures that can be applied for mining gene ontologies. The work results in GOST, a collection of tools used for finding relevant ontology terms to groups of genes.

In the second part, we investigate the statistical nature of the baker's yeast *Saccharomyces cerevisiae* genome and respective annotations to gene ontologies. We describe a number of calculations to find thresholds that distinguish statistically relevant matches from random coincidences.

GOST tools are then applied for mining important gene groups from the baker's yeast expression matrix. As a result, we have found more than a thousand gene groups with very high relevancy rankings.

Viited

[ami05] AmiGO ontology tool.
`http://www.godatabase.org/cgi-bin/amigo/go.cgi`,
14.05.2005.

[apa05] The Apache Software Foundation.
`http://www.apache.org`, 14.05.2005.

[BV00] Alvis Brazma and Jaak Vilo. Gene expression data analysis. *FEBS Letters*, pages 17–24, 2000.

[deb05] Debian open source operation system.
`http://www.debian.org`, 12.05.2005.

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences (PNAS)*, 95(25):14863–14868, December 1998.

[FPSM92] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases - an overview. *Ai Magazine*, 13:57–70, 1992.

[go05] The Gene Ontology Consortsium.
`http://www.geneontology.org`, 11.05.2005.

[HJA+04] M.A Harris, J.Clark, A.Ireland, J.Lomax, M.Ashburner, and R.Fougler et al. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32:258–261, 2004.

[Kul04] Meelis Kull. Fast clustering in metric spaces. Master's thesis, University of Tartu, Estonia, 2004.

[MBB00] M.Ashburner, C.A. Ball, and J.A. Blake. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[mpi05] MPI - Message Passing Interface. `http://www-unix.mcs.anl.gov/mpi`, 19.05.2005.

[mys05] MySQL open source database. `http://www.mysql.com`, 12.05.2005.

[php05] PHP: Hypertext Preprocessor. `http://www.php.net`, 14.05.2005.

[pos05] PostgreSQL open source database. `http://www.postgresql.org`, 12.05.2005.

[Rei04] Jüri Reimand. Kiire hulgaaritmeetika ja rakendused andmekaeves. Semestritöö, 2004.

[Rou87] Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.

[VBJ+00] J. Vilo, A. Brazma, I. Jonassen, A. Robinson, and E. Ukkonen. Mining for putative regulatory elements in the yeast genome using gene expression data. In *Proc. of Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB-2000)*, volume 8, pages 384–394, La Jolla, California, 2000. AAAI Press.

[vH05] Jacques van Helden. Statistics applied to bioinformatics, course materials. `http://www.scmbb.ulb.ac.be/~jvanheld`, 14.05.2005.

[Vil02] Jaak Vilo. *Pattern Discovery from Biosequences*. PhD thesis, 2002.

[yea05] The Yeast Genome.
`http://www.yeastgenome.org`, 13.05.2005.

A Paralleelarvutused ekspressioonimaatriksi kaevandamisel

Käesoleva bakalaureusetöö viiendas peatükis kirjeldatud pärmi geenide ekspressioonimaatriksi kaevandamiseks on rakendatud paralleelarvutuste meetodeid.

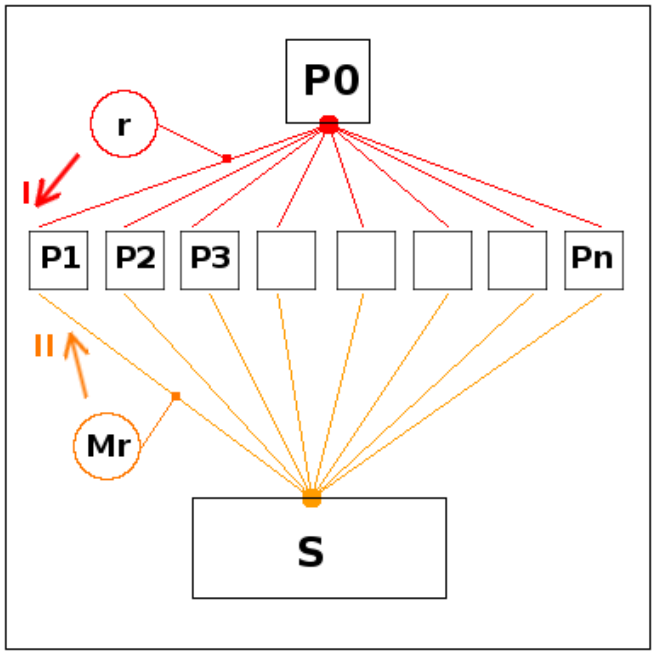
Maatriksi kaevandamiseks on kasutatud arvutiklastrit, mis koosneb 20 Linux operatsioonisüsteemil põhinevast personaalarvutist. Masinad on klastriks seotud paralleelarvutuste tarkvara MPI (ingl. k. *Message Passing Interface*) abil. MPI tarkvara kohta saab lisainfot vastavalt kodulehelt [mpi05].

Tarkvara MPI kujutab endast funktsioonide komplekti keelte C ja FORTRAN jaoks. Funktsioonid võimaldavad üht programmi paljudel arvutitel paralleelsete protsessidena käivitada. Protsesside vahel on võimalik teadete vahetamine funktsioonide `MPI_send` ja `MPI_recv` abil, teate saatmine kõigile protsessidele (ingl. k. *broadcast*) funktsiooniga `MPI_bcast` ja palju muud.

Geenimaatriksi kaevandamiseks loodud süsteem põhineb klient-server mudelil (vt joonis 25), milles arvutavad protsessid laevad serverist maatriksi tükke. Sisuliselt on tegemist kahe erineva serverprotsessiga. Protsess P_0 koordineerib andmete jagamist ning server S jagab andmed füüsiliselt laiali.

Vastavalt algoritmiskeemile 3 leiab iga protsess parimaid ontoloogiamõistete ja maatriksi ridade alamhulki. Seega saab iga protsess tsükli alguses BinSets faili, mis sisaldab teatud pikkusega maatriksi ridu. BinSets failist leitud tulemusi võrreldakse juba olemasolevate tulemustega ning säilitatakse parimad. Siin tuleb märkida, et võrdlus toimub ainult vaadeldava protsessi enda poolt leitud tulemustega.

Üks protsessidest on klastri juur P_0, mis koordineerib teistele protsessidele $P_1, P_2, \ldots P_n$ jagatavaid sisendväärtusi. Alamprotsess P_i pärib juurelt iga tsükli alguses järgmist sisendväärtust r. Juur haldab jooksvat reapikkust r' ning päringu



Joonis 25: Paralleelarvutused ekspressioonimaatriksi kaevandamisel

järel suurendab selle väärtust 1 võrra. Seejärel laeb P_i vastavalt saadud sisendväärtusele serverist S maatrisi tüki M_r.

Kui reapikkus on jõudnud suurima võimaliku väärtuseni, ootab juur ära kõigi alamprotsesside uued päringud ning vastab neile väärtusega `-1`, mille järel alamad lõpetavad töö. Summaarsete tulemuste sünkroniseerimine toimub väljaspool paralleelarvutuste süsteemi.

Loodud süsteemi struktuur on suuresti seotud töödeldavate andmete iseloomuga. Ridade pikkuse järgi tükeldatud geenimaatriks on ligi 30Gb suur ning selle jaotamine kõikide klastri arvutite vahel ei ole mõeldav. Samal ajal on BinSets formaadis tüki suurus maksimaalselt 5Mb, mille laadimine kiire kohtvõrgu korral ei ole kuigi aeganõudev.

Tasub tähele panna, et maatriks on protsesside vahel dünaamiliselt jaotatud.

Võrreldes staatilise jaotusega, kus igal protsessil on kindlaksmääratud reapikkuste vahemik, on kirjeldatud jaotus märgatavalt kiirem. Võit tuleneb sellest, et klastri arvutid on loomult erineva võimsusega ning mõned masinad on perioodiliti rohkem hõivatud.

Paralleelarvutuste kasutamine andis käesolevas töös kirjeldatud analüüsi sooritamiseks märgatava ajalise võidu. Hinnanguliselt oleks ühe arvutiga ühe headusparameetri jaoks parimate gruppide leidmine aega võtnud kolm nädalat. Paralleelse lähenemisega valmisid tulemused kahe päeva jooksul.

B Lisatud CD tarkvara ja andmetega

Bakalaureusetööga on kaasas CD, mis sisaldab valikut töö kaigus valminud programmidest ja andmetest. Lähem informatsioon CD sisu kohta on saadaval kataloogidele vastavates `README` failides.