U N I V E R S I T Y O F T A R T U
Faculty of Mathematics and Computer Science
Institute of Computer Science

Jüri Reimand

Gene Ontology mining tool GOSt

Master's Thesis

Supervisor: Jaak Vilo, PhD

TARTU 2006

Contents

Introduction

Recent advancements in technology have changed research course of molecular biology. Fully sequenced genomes of versatile organisms become available at great pace. A modern biologist no longer needs to concentrate on a single gene. High-thoughput technologies such as microarrays produce groups with hundreds of interesting genes proven similar in some sense. A common question to ask in this case is whether there is something else similar to the group besides expression values.

New issues arise with the ever-growing amount of available data. Genetic information is scattered in various databases and often accompanied with terminological confusion. Gene Ontology (GO) was established to address this problem. GO consists of controlled hierarchical vocabularies of genetic domain. GO is structured as a Directed Acyclic Graph. Many research communities use GO terms for annotating genes to known processes, components and functions.

This Master's Thesis investigates methods for mining Gene Ontology and respective gene annotations, to determine common annotations to a group of genes. First chapter gives a brief introduction to biological matters, and proposes a model for combining GO data with biological pathway databases. Second chapter investigates statistical means for ontology mining, and proposes thresholds for recognising significant results from random matches. Third chapter describes fast mining algorithms, and introduces novel concepts of analysing ordered gene lists and mining significant subgraphs.

Practical result of this Thesis is GOSt, a Gene Ontology mining tool. Fourth chapter is dedicated to usage and features of GOSt. It describes common options, sample output, and expression data analysis pipeline. Visualisation is often the key for understanding and interpreting biological data. We have put great effort in making complex results understandable to users, and consider visualisation to be among the strengths of GOSt.

Chapter 1

Gene Ontology (GO)

This chapter starts off with the biological background of our work. We give a brief overview of the controlled biological vocabularies maintained by Gene Ontology Consortium, followed by an introduction to biological pathway databases and related gene annotations. We propose a model for combining the two datasources, and finally provide a motivating example for largescale analysis of genetic data with GO and KEGG vocabularies.

1.1 Biological background

Recent developments in molecular biology have brought along an explosion in the amount of available data. Sequencing race began in 1996 with the release of the genome of *Saccharomyces cerevisiae*, a higher model organism commonly known as baker's yeast. The completely sequenced human genome was announced in 2003, containing approximately 3 billion base pairs and 25000 genes [NHG]. At the time, numerous other animal genomes are fully available, while others are still at different stages of completeness.

Information gained from completed sequences of various genomes suggests that there exists a single finite superset of genes and proteins, most of which are conserved in many or all living cells. This recognition has lead to unification of biology. Known properties and functions of genes and proteins in a specific genome contribute to the general knowledge base, as it is likely that similar functions are expressed in homologous genes of many other diverse organisms [ABB$^+$00].

Relevant information can be extracted from previously proven results with well-known model organisms, and used for studying more complex genomes.

Unfortunately, the pace of at which new genomes are sequenced often exceeds the speed of organising and cross-referencing existing data. Biological information concerning genes, proteins and their functions is primarily available in numerous genome-specific databases and maintained by different organisations. Moreover, there is often no clear understanding or agreement concerning common genetic terminology and functional descriptions of biological objects. Therefore, the task of finding relevant genes of similar function may be quite challenging.

Gene Ontology (GO) Consortium was established to address the above problems. The primary goal of the consortium is to maintain and develop a controlled and organism-independent vocabulary of the molecular biology domain. Such vocabulary provides a hierarchical collection of terms, to describe general as well as specific molecular functions, cellular components and biological processes [ABB+00, ABB+01, ABB+04]. GO project was initially a collaboration between the Saccharomyces Genome Database [SGD] of baker's yeast, Mouse Genome Informatics [MGI] of common house mouse, and Flybase [FLY], the database of fruitfly. More databases joined the consortium later, and many other data sources are using the ontologies today for identifying genes and proteins by their functionality. Vocabularies and gene annotations are freely available at the GO Consortium web site [GO].

1.2 Ontology design and implementation

The concept of ontologies in computer science was first introduced in research related to Artificial Intelligence and Knowledge-Based Systems with purpose of sharing knowledge and improving communication between independent systems [Gru93].

According to Genesereth and Nilsson, a body of formally represented knowledge is based on *conceptualisation*: objects, concepts and other entities that are assumed to exist in some area if interest, and relationships that hold among them [GN87]. The specific area of interest is often referred to as *domain*, and objects are known as *terms*.

Ontology is an explicit specification of conceptualisation. An ontology con-

sists of a set of terms represented in a given domain, and relationships that hold between terms. Knowledge concerning objects and relations is stored in a representational *vocabulary* [Gru93]. In addition to objects and relations, ontology holds respective human-readable descriptions, and formal axioms that constrain interpretation and use of objects and relations.

Gene Ontology vocabularies are structured in a form of *Directed Acyclic Graphs* (DAG), directed graphs with no path starting and ending at the same vertex. A vertex of GO graph corresponds to a biological term, and a directed edge between two terms shows that one term is hierarchically related to the other. Such graph represents hierarchical structure resembling a tree, except that each child vertex may have more than one parent vertices. The situation that a specific term is a child of multiple broad terms, captures well the biological reality [ABB+01].

Two types of parent-child relationships are defined in GO. Relation type *is_a* describes the fact that child term is an instance of parent, while relation type *part_of* denotes that child term is a component of parent. A child term may have different classes of relationships with its parents. Every term has a unique identifier (`GO:0000001`) and a name. Besides these, a number of optional properties may be defined.

There are a few GO rules and guidelines to be followed. *True Path Rule* is the most relevant guideline in the context of this work. It states that for any given child term, the path to its top-level parent must always be true. Therefore, every parent term means of the union of its child terms. In case of multiple parents, all paths from a term to top hierarchy have to be verified. An example of GO hierarchy is available in Figure 1.1.

1.3 Three ontologies of GO

As stated in above definitions, an ontology represents knowledge of a specific domain or an area of knowledge. Gene Ontology maintains vocabularies of three domains, Molecular Function, Biological Process and Cellular Component. These particular classifications were chosen because they represent information sets that are common to all living organisms. Vocabularies are developed for a generic eukaryotic cell; specialised organs and body parts are not represented [ABB+01].

It is correct to say that Gene Ontology consists of three independent vocabu-

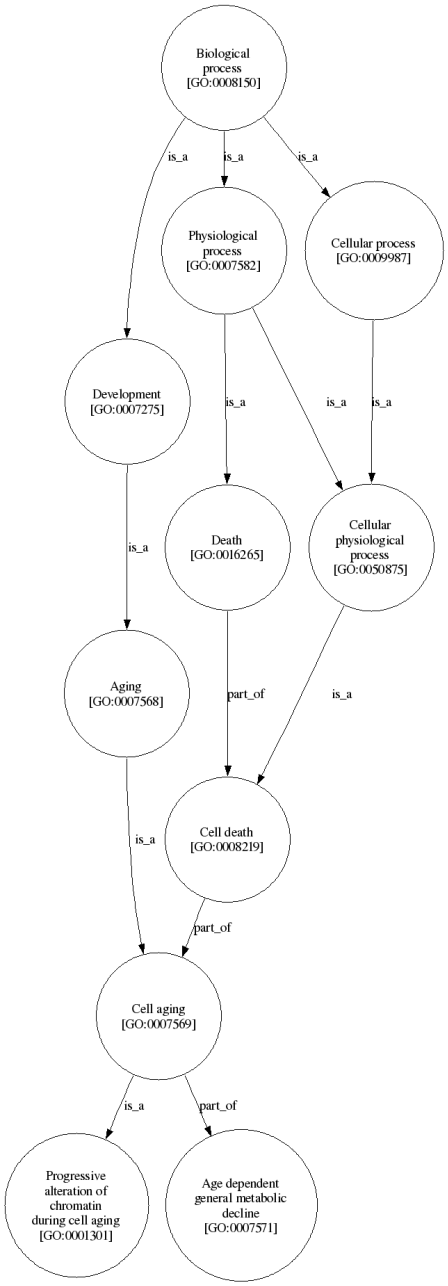

Figure 1.1: A fragment of GO hierarchical vocabulary demonstrating different relation types, multiple parents and True Path Rule. The concept of *cell aging* is a part of *cell death* and at the same time, a more specific term of *aging*. All 8 ancestors of *cell aging* must always be true when the term itself is true.

laries of different domains. Each vocabulary has one root term and there are no parent-child relations linking vertices of different ontologies.

- *Molecular function* (`GO:0008639`, MF or F) is defined as what a gene product does at biochemical level. Domain terms only specify function, location and time of event remain undefined within ontology.

- *Biological process* (`GO:0008150`, BP or P) refers to biological objective to which a gene product contributes. A process is accomplished by through one or more ordered assemblies of functions, often involving transformation of biological matter.

- *Cellular component* (`GO:0005575`, CC or C) refers to place in cell or extracellular region where a gene product is found or where the product is active.

Two types of terms deserve further attention. Every domain has a an *unknown* term just below root; it is meant to hold genes and gene products that have been investigated, but no knowledge of the domain has been revealed. From time to time, some terms are marked as *obsolete* as biological knowledge evolves; these terms are removed from active vocabularies and placed under *obsolete* term of the domain in question.

GO Consortium explicitly states that Biological Process domain is not equivalent to a biological pathway and describing a pathway through the necessary dynamics and dependencies between processes and functions is beyond the scope of the GO project [ABB$^+$01]. GO Consortium recognises that there exists a biological relationship between a series of molecular functions in a biological process, that unfold in a certain component of a cell. This means that there are in fact numerous interconnections between three independent domains. Even though GO could be logically expanded to reflect states, operations and components of cells, current goal of the project is to concentrate on the development of three independent and precise collections of terms [ABB$^+$01].

In current work, we give a brief introduction into biological pathways and then propose a simple model for integrating knowledge from Gene Ontology vocabularies with pathway databases such as KEGG.

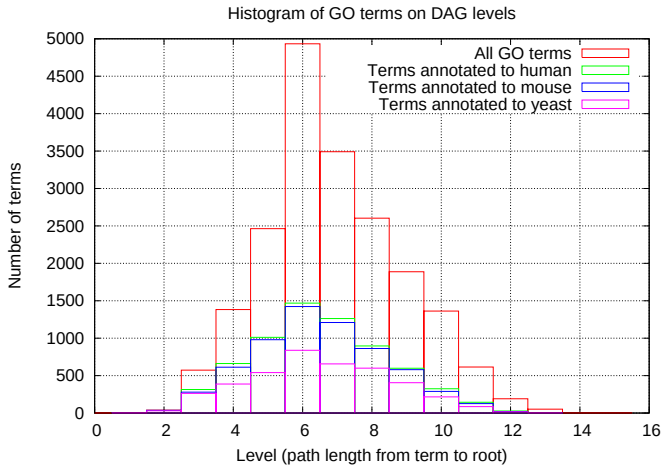

Figure 1.2: Histogram of GO terms for different DAG levels. Terms are shown only if annotations of fixed genomes exist. Out of nearly 20000 GO terms, 8785 are either annotated to human (*Homo sapiens*, 6754 terms), house mouse (*Mus musculus*, 6424 terms), or baker's yeast (*Saccharomyces cerevisiae*, 4041 terms). More than 2600 terms are common to all three organisms, while nearly 84% of mouse terms are also annotated in human.

On March 26^{th}, 2006, GO vocabularies consisted of 19604 terms, including 10517 terms of Biological Process, 1688 terms of Cellular Component and 7399 terms of Molecular Function [GO]. There were 1001 obsolete terms not included in the above statistics. The longest path from child to root involves 15 edges, but most of the terms are normally distributed on middle levels (Figure 1.2). Ontologies are by no means complete and continuously expanded through collaboration of many organism-specific databases. Gene Ontology website [GO] releases a new vocabularies snapshot in every 30 minutes.

1.4 Gene annotations

In addition to three ontologies of Molecular Function, Biological Process and Cellular Component, Gene Ontology involves links between GO terms and genes or gene products. These links are commonly referred to as *annotations* or some-

times as *associations*. Annotations are maintained and developed independent of GO [ABB+00].

GO Consortium maintains annotations of three model organisms of founding members, namely baker's yeast *Saccharomyces cerevisiae,* common house mouse *Mus musculus* and fruitfly *Drosophila melanogaster*. GO annotations of numerous other genomes, including human, are now available from respective organism-specific reseach communities.

Every annotation to GO is attributed to a source, which may be literature reference, another database or computational analysis. Annotation must also indicate the type of evidence, provided by the source to support association between given genetic entity and GO term. A standard set of *evidence codes* is available for qualifying annotations with respect to different types of experimental conditions [ABB+04].

Evidence codes provide means to describe a range of different experiments varying from *in vitro*[1] techniques as direct assay, to purely *in silico*[2] methods for determining sequence alignments and string similarity. GO web site includes a comprehensive annotation guide for evidence codes and proposes a loose order of decreasing reliability [GO]. A histogram of evidence codes across yeast, mouse and human gene annotations is available in Figure 1.3.

Two evidence codes deserve further discussion. Evidence code *Inferred from Electronic Annotation* (IEA) is meant to describe results that are gained from completely unsupervised computation. Such annotations and conclusions from these should therefore be treated with care.

Evidence code *No Biological Data Available* (ND) indicates that a gene or product has been researched, but no information to support a GO term was discovered. Genes with ND evidence code are annotated to the special *unknown* terms described in the previous section. It is common that ND in one vocabulary is used in conjunction of one or more 'normal' evidence codes in other vocabularies. For example, it may be known that a certain gene product is activated inside cell nucleus, but nothing is known about its molecular function or involved biological process.

[1]Latin: "within glass"; biological experiments performed in a test tube, or generally outside a living organism or cell.

[2]Latin: "within silicon"; a general term for any computational means in biology.

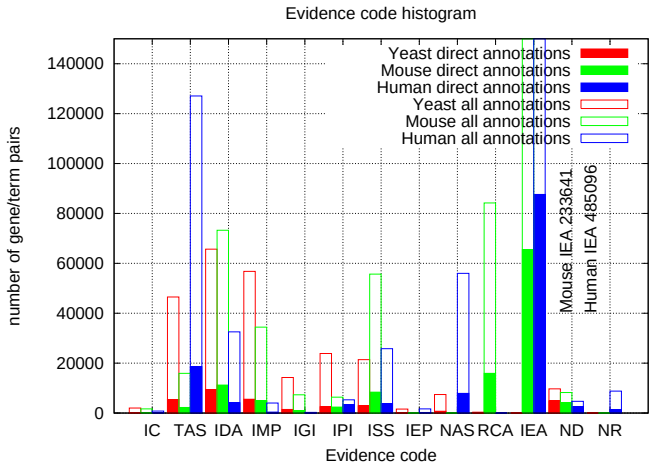

Figure 1.3: GO annotations histogram for different evidence codes of yeast, mouse and human genomes. Filled boxes represent direct annotations, while the transparent area above filled boxes represents annotations via True Path Rule. Also see Table 4.1.

One of the most important guidelines of GO is the previously described True Path Rule. In the context of annotations, the guideline is interpreted as follows. Every gene or gene product that is annotated to a specific term in Gene Ontology, is always annotated to all term's parents up to top-level parent, using all possible paths from term to root [ABB$^+$01].

Such indirect True Path annotations are not provided in GO datasets and therefore are to be added explicitly. True Path Rule also explains the need for storing several evidence codes for any gene-term pair. Besides the fact that different experimental results may support exactly the same annotation, terms located in top hierarchy get repeated indirect annotations of same genes via different paths.

Figure 1.4 displays a histogram for term sizes in sense of annotated genes. For every organism, there numerous highly specialised terms with only a few annotated genes, while larger groups are more uncommon. Largest groups on the right side of the figure represent root terms, each of these containing the union of its descendants' annotations.

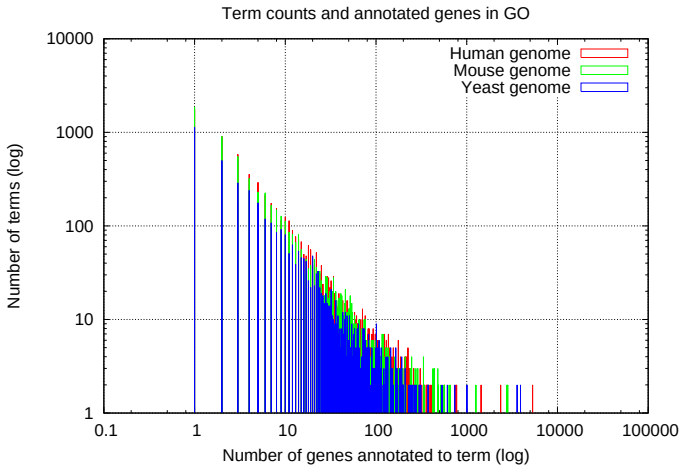

Figure 1.4: Histogram of annotations per GO term, for human, mouse, and yeast genomes.

1.5 Biological pathways

According to Karp, a *pathway* is a linked set of biochemical reactions, where a product of one reaction is a reactant of, or an enzyme that catalyses, a subsequent reaction [Kar01]. In other words, a pathway is a biochemical process that can be partitioned into component steps [Sch04]. Small metabolic processes with just a few reactants, as well as macroprocesses involving hundreds of molecular components with the cooperation of multiple cells, are commonly described as pathways [Kar01].

Metabolic networks are currently the most well-studied biological pathways. A *metabolic network* is essentially a chemical processing factory within each cell, that enables the organism to convert small molecules from the environment into building blocks of its own structures, and to extract energy from those molecules [Kar01].

Sequence of steps comprising a pathway is rarely a simple linear sequence, as a single reaction often requires multiple inputs and creates multiple outputs. A pathway may contain redundancy, as multiple parallel series of events produce the same biochemical result. On the other hand, a single molecular component can be multifunctional and involved in multiple pathways with different goals. Pathways

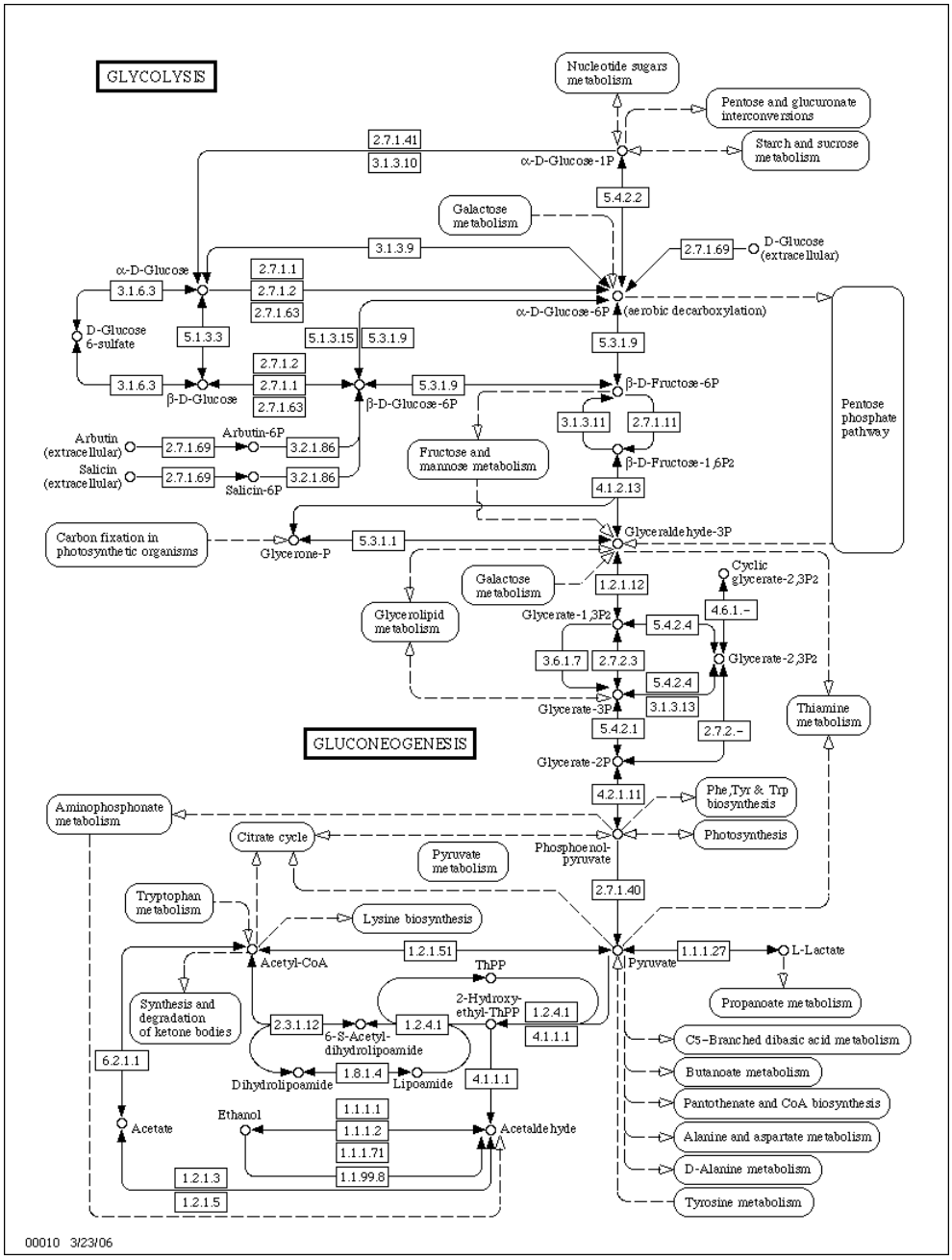

Figure 1.5: KEGG pathway `00010` for *glycolysis and glyconeogenesis*. The pathway involves 48 genes in yeast, 55 genes in mouse, and 63 human genes. The genes have significant overlaps with other genes in pathways and GO Biological Process terms.

may also be competitive; activities of one pathway may render the other pathway inactive, as the first one consumes, binds or deactivates some resource on which the second pathway depends [Sch04].

A mathematical representation for a pathway is a directed graph, that at a high level displays the cause-effect dependencies among the components. It has been more common to display molecular components as nodes of graph and underlying events (reaction, modification, translocation, transcription) as edges between nodes [Sch04].

1.6 Pathway databases and integration with GO

One may argue that the most natural embodiment of pathway knowledge is the related set of biomedical literature. As most of this literature is available in electronic form, a lot of knowledge can be extracted automatically using techniques of text mining. However, there are serious limitations to this approach. Despite the many advantages in the field of natural language processing, the raw output of text mining is not sufficiently precise and and it takes great effort to complete even the simplest queries on the pathway data [Kar01].

Relational databases are nowadays the most common way of storing pathway data. Most pathway databases were initially created to describe metabolic pathways, but databases with signalling and genetic-regulatory pathways are appearing as well. Comparative analysis of some pathway databases is available in [Sch04]. In our current work we study the pathway data from the The Kyoto Encyclopedia of Genes and Genomes (KEGG). KEGG is a knowledge base for systematic analysis of gene functions in terms of networks of genes and molecules, that provides means of linking genomes to biological systems [OGS+99, KGH+06]. KEGG database is publicly available on their web site [KEG].

KEGG resource consists of 4 major components. GENES database is a collection of gene catalogues for all complete genomes and some partial genomes. LIGAND database describes building blocks of the biochemical space, such as enzymes, chemical compound structures, reactions and other substances in living cells, as well as a set of drug molecules. PATHWAY database consists of a collection of pathway maps, while BRITE database holds a collection of hierarchies and binary relations that correspond to rules governing the genome-environment

interactions in pathways [KGH+06].

In current work, we are most interested in the KEGG PATHWAY database, that holds a collection of manually drawn pathway maps for metabolism, genetic information processing, environmental information processing, various other cellular processes and human diseases [KGH+06]. Every pathway in KEGG is identified with a five-digit code (`00010`) and described with a name (*glycolysis and glyconeogenesis*). Organism-specific pathways are automatically generated based on the generic pathway maps by matching genes from the organism's catalogues. Pathways are partially distributed into classes and subclasses. For example, the broad class *metabolism* is divided into subclasses like *energy metabolism*, *nucleotide metabolism*, etc. Each of the subclasses holds a number of pathways, for example *sulfur metabolism* is a kind of energy metabolism.

Hierarchical relations within the set of KEGG pathways suggest a model for integrating pathway data into Gene Ontology model. Gene Ontology consists of three independent ontologies, namely Biological Process, Molecular Function and Cellular Component. We add KEGG pathway data into Gene Ontology data model as the 4^{th} independent ontology of Pathway (PW). The fourth ontology has the following GO-compliant properties.

- The set of terms in PW ontology is equal to the collection of available organism-independent pathways. All terms have a unique identifier. The identifier in our model includes the prefix 'KEGG:' (`KEGG:01150`) to distinguish it from GO terms.

- The set of gene annotations of a pathway is the collection of genes mapped to organism-specific version of the pathway.

- Terms and annotated genes of PW ontology are created independently, and hierarchically unrelated of the three remaining ontologies.

- Every parent term in the PW ontology explicitly includes all the annotations of its child terms.

- The top-level root term PW ontology *KEGG Pathways* with the identifier `KEGG:00000` holds all genes present in KEGG pathways. This term is a placeholder, as no such general term currently exists in KEGG database.

Proposed model makes it possible to analyse Gene Ontology terms and pathways simultaneously, and determine possible interconnections and correlation within related gene annotations. The model is not limited to KEGG database; other pathway databases as well as different types of knowledge, such as protein-protein interactions may be integrated.

The idea of combining GO with other sources of genetic knowledge is not new. Queries over KEGG Pathways are enabled in GO tools like DAVID [GDSH+03] and GFINDer [MMP04]. Knowledge from the protein families database Pfam [BCD+04] is available in these tools as well; the latter tool also includes data from OMIM, the catalogue of human genes and genetic disorders [HSA+02]. Further developments of this work involve studying above databases as well as other sources, and integrating those into our proposed data model.

On the one hand, we recognise that viewing a pathway as an unstructured set of annotated genes greatly simplifies the picture, as we disregard internal dependencies, chemical building blocks and internal rules of behaviour. On the other hand, a high-level overview of participating pathway genes with combined data of molecular functions, biological processes and cell locations may help to hypothesize about more general ideas of the biological domain.

1.7 Application of GO: Gene expression analysis

This subsection gives a brief introduction into gene expression and involved experiments in the context of Gene Ontology mining. We provide motivation for large-scale GO analysis of gene sets. However, detailed insight into the matter falls out of the scope of this work. The section is based on an introductory bioinformatics web material [BPSS01].

Gene expression is generally the process of producing proteins from information stored in genes. Proteins are fundamental building blocks of known living organisms. Therefore, gene expression mechanisms, conditions and alternative variants are subject to some of the most relevant (and often unanswered) questions of life sciences.

Various examples of gene expression combined with GO analysis are available, for example *Zhang et al* study expression in mouse tissues, and construct corresponding groups of GO terms from Biological Process domain [ZMC+04].

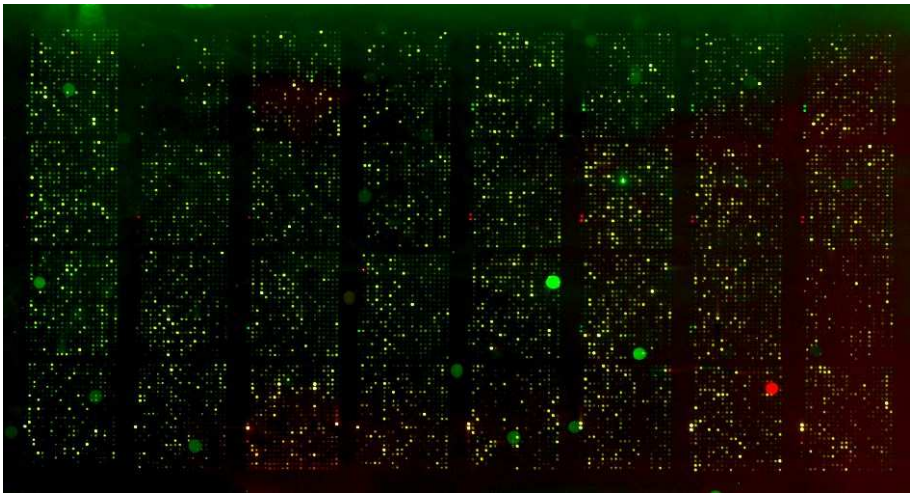

Figure 1.6: Mouse microarray slide with 15000 cDNA spots in an hybridisation experiment

At the time, *microarray technology* is the principal method for evaluating gene expression. The technology takes advantage of preferential binding of complementary single-stranded nucleic acid sequences. DNA is normally a double-stranded helix with connected opposite strands of complementary sequences. A single strand of DNA will therefore easily bind to its complementary sequence.

A *microarray* (Figure 1.6) is typically a square glass slide in size of a few centimetres, onto which single-stranded DNA molecules are attached at fixed *spots*. There are often tens of thousands spots on an array, and millions of DNA molecules on a spot. Such arrays are now widely produced by commercial institutions such as Affymetrix [AFF]. All molecules of a given spot ideally represent the sequences of the same gene, and all spots correspond to genes in the genome. However, not all molecules in spots are of same quality, and not all sequences on the array are actually known.

One of the most popular microarray applications involves comparison of expression in two different hybrid samples, for example, mRNA extracts from diseased and healthy instances of a specific cell type. Extracts are labelled with fluorescent dyes, usually green and red, and washed over the array. Due to preferential binding, extracts of both diseased and healthy cells bind to complementary

sequences on the array, and extracts' fluorescence indicates spots on array that have sequences actually activated in cells. One active condition dyes the spot in a shade of either red or green. If both of the conditions activate the spot, it appears as a yellow shade; if the gene is not expressed in either healthy or diseased cell, the spot remains black. Coloured spots are then scanned in with lasers as values of relative gene expression.

Second step of expression analysis combines single microarray experiments into groups. One common approach involves analysis of time series, where gene expression in a cell is measured in multiple consecutive timepoints τ; another approach gathers together several experimental conditions τ, for example mutations, disabled genes (knockouts), etc. Expression values e over genes in multiple datasets are normalised into an *expression matrix* E.

$$E = \begin{pmatrix} & \tau_{\mathbf{1}} & \tau_{\mathbf{2}} & \tau \ldots & \tau_{\mathbf{m}} \\ \mathbf{g_1} & e_{g_1,\tau_1} & e_{g_1,\tau_2} & \ldots & e_{g_1,\tau_m} \\ \mathbf{g_2} & e_{g_2,\tau_1} & e_{g_2,\tau_2} & \ldots & e_{g_2,\tau_m} \\ \mathbf{g}\vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{g_n} & e_{g_n,\tau_1} & e_{g_n,\tau_2} & \ldots & e_{g_n,\tau_m} \end{pmatrix}$$

Genes in the expression matrix may then be defined through their expression vector, $\mathbf{g} = (e_{\tau_1}, e_{\tau_2}, \ldots e_{\tau_m})$. Different measures, such as *Euclidean distance* or *correlation*, may then be used for calculating distances between genes. Distance defines the expression similarity of two genes; the more similar are the genes, the smaller is the distance between them.

Similarity measures allow to split expression dataset into smaller groups of genes. For example, one may pick an interesting gene and find a number of its closest neighbours based on expression similarity. There are more sophisticated methods; *clustering*, for example, attempts to split the given dataset into meaningful groups - clusters - so that objects inside a cluster are very similar to each other, while objects of different clusters are very different. A good overview of various clustering techniques and a novel method for fast approximate clustering is given in [Kul04].

As expression datasets are large and there are various methods for splitting the data into smaller subsets, such experiments normally result in innumerable groups of genes that have displayed biological evidence to be "similar" in some

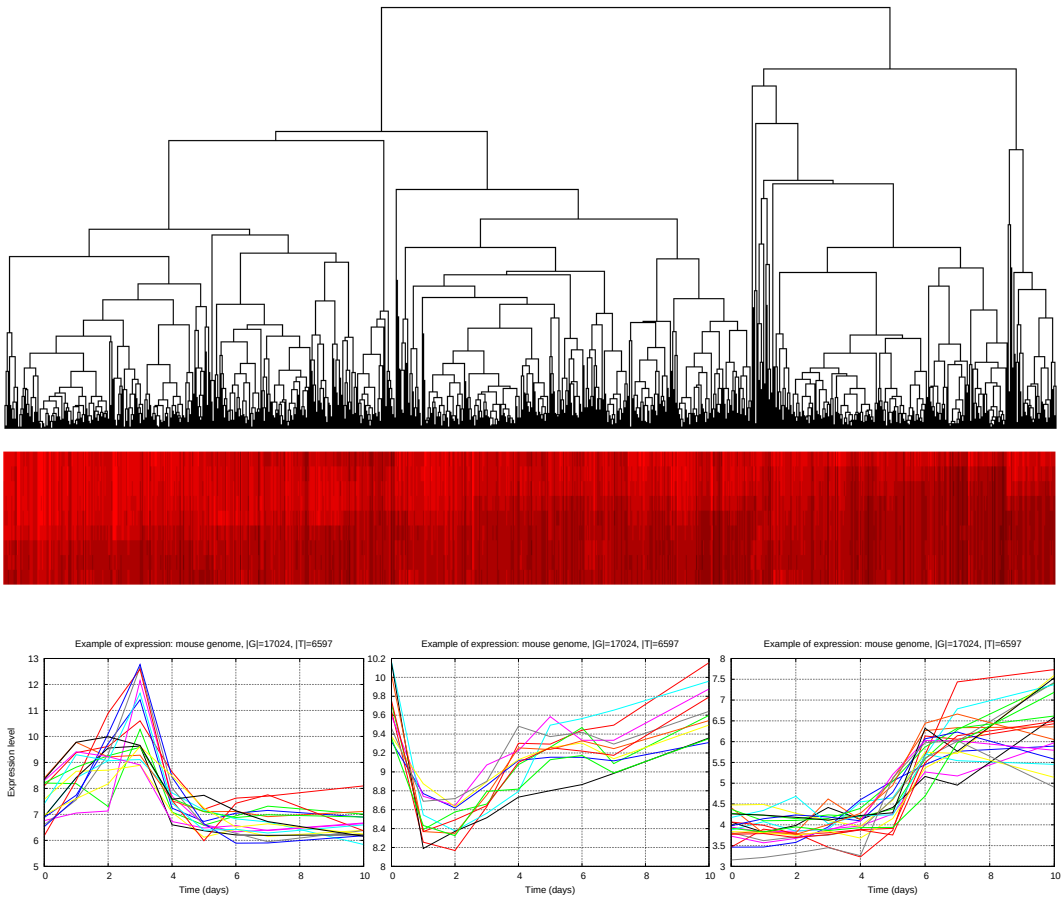

Figure 1.7: Figures display fragments of mouse embryonic stem cells expression data, involving 9 time points across 10 days. Genes are grouped with hierarchical clustering (top image), and by sorting closest similar genes (bottom images). The 724 genes in hierarchical clustering have a significant fraction annotated in *organ development* (`GO:0048513`). A large subset of the group on bottom leftmost plot is known to take part in *cell differentiation* (`GO:0030154`). This group also includes two previously unannotated sequences.

sense (Figure 1.7). The most common question in this case is whether there is anything else common to a group besides expression similarity.

This question may be answered with the help of controlled vocabularies as GO and KEGG. Interesting biological information is often revealed when one observes known annotations of a group of similar genes. Gene groups may be further clustered, based on annotations in the group. If a well annotated and similarly expressed gene group includes some unknown sequences, there may be strong statistical evidence that suggests annotation for the unknown sequence. Known cellular components and molecular functions of a gene group may aid a researcher in investigating biological processes or improving existing annotations.

Methods are needed for determining common Gene Ontology annotations and biological pathway references to a group of genes. Such methods need to be fast in order to enable large-scale simultaneous analysis of numerous gene groups. Statistical means and other measures are required for evaluating the significance and importance of annotations resulting a group of genes.

Chapter 2

GOSt - Gene Ontology Statistics

Previous chapter contained a brief introduction into hierarchical vocabularies such as GO and KEGG, and corresponding gene annotations. We provided some biological background of the matter and described an example of microarray experiments, where a researcher would profit from fast methods for finding common GO and KEGG terms for a group of genes.

This chapter constructs a mathematical model of vocabularies and annotations, defines queries of genes and results of terms, and provides means for assessing the importance of results.

2.1 Gene Ontology representation

Gene Ontology consists of controlled hierarchical vocabularies for three independent domains of Biological Process, Molecular Function, and Cellular Component. We have added the KEGG Pathway domain as the 4^{th} vocabulary. Vocabularies are represented as Directed Acyclic Graphs, where every vertex corresponds to a term of the vocabulary, and the edges between terms describe hierarchical relations between general and specific terms (Figure 1.1).

It would be correct to treat every domain to be an independent vocabulary and represent it in a different graph. In current work, we use a simple model and consider the four independent ontologies of BP, MF, CC and PW as one vocabulary stored in a single DAG. We store the domain of every vertex, and apply the condition that there may be no edges connecting vertices of different domains.

Let us define a hierarchical vocabulary as a triplet $\mathcal{V}$. The three elements in $\mathcal{V}$ are sets. T is a set of vocabulary terms. Ξ is a set of hierarchical relation types that may hold between terms.

$$\mathcal{V} = (T, \Xi, R)$$

First two relation types in Ξ are specific to Gene Ontology. First relation type is_a defines child as a more specific version of the parent, while second type $part\_of$ declares child to be a component of the parent. Third type is a generic $child\_of$ child-to-parent type, introduced to bind terms in the newly-added KEGG Pathways domain, as relation types are originally not specified in KEGG.

$$\Xi = \{is\_a, part\_of, child\_of\}$$

The third element R contains term relations that define the actual hierarchical DAG structure of the vocabulary. Every element r in R may formally defined as the following triplet.

$$r = (t, t', \xi)$$

Informally, R contains a number of *facts* r, each stating that a given term t is a child of another term t', where both of the terms are in the term set T, and there is a specific parent-child relationship ξ between the terms, for example $t\ is\_a\ t'$. We should also note that the relationship is unique for a given parent-child pair, meaning that there can be only one fact $r = (t, t', \xi)$ for any given pair of parent term and child term.

Vocabulary holds exactly one top-level term t_0 for every domain of MF, BP, CC and PW. These terms are called *roots*. Roots do not have parents; the collection R does not contain facts where any of t_0 is a child. All other terms are either directly or indirectly related to roots, and as stated above, terms are only related to their domain-specific root. Connecting edges between terms of different domains are not allowed.

We may combine facts in the set R to define some more useful sets. The DAG structure allows to define hierarchical relations, so that a given term t is a child of multiple terms. Let T_t be the set of *parents* for term t. This means that we need to observe all relation facts r containing t as a child, and collect all parents t' in these facts. Note that for a root term t_0, the set of parents is empty, $T_{t_0} = \emptyset$.

$$T_t = \{t' \in T | \exists \xi \in \Xi : (t, t', R) \in R\}$$

The set of *children* for a given term t may be defined in a similar way.

In order to understand and apply True Path Rule, we also define the set of *ancestors* of term t as the set T'_t. This set is defined iteratively; we need to collect parents T_t of a term t, and then grandparents of term t, parents of these grandparents, and so on, until we reach root term t_0 that by definition does not have any parents.

$$T'_t = T_t \cup \bigcup_{t' \in T_t} T'_{t'}$$

For a root term t_0, the set of ancestors is also empty, $T'_{t_0} = \emptyset$. The set of *descendants* may be defined analogously as an iterative union of children, children of children, etc, of term t.

As any term may have multiple parents, every level of hierarchy may involve multiple paths to root. All such paths need to be verified when collecting the set of ancestors. The DAG definition states that the structure may not involve loops. This means that no term t may have another related term t', that is simultaneously its ancestor and its descendant. The term t must not have itself among its ancestors or descendants.

2.2 Annotations and True Path Rule

Informally, an annotation is the link between a gene g and a vocabulary term t. Annotations of genes to GO are maintained independently of GO. Furthermore, annotations of different organisms are independent of each other, and incorporate different subsets of vocabulary terms.

A collection of annotations can be written as the quadruple $\mathcal{A}$. Every element in the quadruple is a set. The set G includes all genes of the genome of a given organism. The set $T_G \subseteq T$ consists of such vocabulary terms, that are directly annotated in the given organism. *Evidence codes* in $\mathcal{E}$ are relation types, presenting different types of knowledge that support annotating a gene g to a term t. Role of evidence codes is similar to previously defined relation types in Ξ.

$$\mathcal{A} = (G, T_G, \mathcal{E}, A)$$

The fourth element in the quadruple $\mathcal{A}$ is A, that in its essence is the same as R defined in the previous section. The set A contains a collection of facts, that

declare a given gene g to be a representative of a GO/KEGG term t. The fact may more formally be written down as a triplet a, where $\epsilon \in \mathcal{E}$ denotes type of evidence used to declare the fact a.

$$a = (g, t, \epsilon)$$

Unlike hierarchical relations ξ between vocabulary terms, a gene-term annotation pair may be supported by more than one evidence code. In other words, there may be several facts a with different evidence codes $\epsilon_1, \epsilon_2, \ldots$ for gene-term pair.

True Path Rule of GO states that for every term t, all paths towards the root term t_0 must remain true. Any gene directly annotated to a term must be explicitly annotated to all ancestors of t. Let us extend the collection $\mathcal{A}$ to $\mathcal{A}'$, so that it complies with True Path Rule.

$$\mathcal{A} = (G, T_G, \mathcal{E}, A) \longrightarrow \mathcal{A}' = (G, T'_G, \mathcal{E}, A')$$

Firstly, we need to update the collection of facts A into A' and make it compliant with True Path Rule. For every fact $a = (t, g, \epsilon) \in A$ binding a gene g and a term t, we need to observe all the ancestors of term t via all paths to root, and add gene g to these ancestors with the same evidence code ϵ. This is another reason why a single pair (g, t) may propagate a number of different evidence codes.

$$A' = A \cup \{(t', g, \epsilon) \in T_G \times G \times \mathcal{E} | \exists t \in T_G : (t, g, \epsilon) \in A \wedge t' \in T'_t\}$$

Secondly, we need to update the set of terms T_G into T'_G. It is important to know that genes are normally directly annotated to the most specific and exact terms low in hierarchy. Other annotations to the parents and ancestors are to be added explicitly. The set of terms T_G may lack some terms on paths to root that are not directly referenced in annotations. These terms need to be added as well; the new set of terms T'_G includes all directly annotated terms of the genome, plus all ancestors of these terms.

$$T'_G = T_G \cup \bigcup_{t \in T_G} T'_t$$

2.3 Annotation sets and querying

It is useful to compose some new sets based on the collection of facts A' defined in previous section. We are interested in genes of the genome G annotated

to a particular term t. We denote this set G_t, the *annotation set* of term t. We compose this set by observing all facts $a \in A'$ that contain term t, and collecting all genes in these facts. The composition of G_t has to be done only after True Path Rule from the above section has been applied.

$$G_t = \{g \in G | \exists \epsilon \in \mathcal{E} : (t, g, \epsilon) \in A'\}$$

We compose annotation sets for every directly or indirectly annotated GO and KEGG term, and collect these sets into the collection $\mathcal{G}$. The collection $\mathcal{G}$ can be added as the fifth element in the annotation dataset of genome G.

$$\mathcal{A}' = (G, T'_G, \mathcal{E}, A') \longrightarrow \mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$$

Now we have the two essential components for ontology mining - the vocabularies dataset $\mathcal{V}$ and the extended annotation dataset $\mathcal{A}'$.

$$\mathcal{V} = (T, \Xi, R)$$
$$\mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$$

One of the most common goals for Gene Ontology data analysis would involve determining annotations for a *query* of genes $G_q = g_1, g_2, \ldots, g_l$. A query is a user-defined set of genes, and user wants to find GO/KEGG annotations to genes in the query.

We say that a term *matches* the query, if G_q involves genes that are annotated to term t. This can be evaluated by the simple intersection $G_q \cap G_t$. Result set of terms T_q for gene query G_q simply includes all terms that have at least one gene present in the user query.

$$T_q = t | t \in T'_G \wedge G_q \cap G_t \neq \emptyset$$

2.4 Ranking results

Due to True Path rule of GO, annotation sets of terms are greatly overlapping. Sets in top hierarchy contain most of the genome's genes. For a single gene-term annotation, all parent terms in paths towards root are populated with the same

gene. Therefore we get a large number of matching GO terms to any input query of genes.

In order to find relevant terms to reveal biological knowledge, we first need to evaluate the importance of each occurring result. We may then reorder results by importance, and claim top of result list to contain more relevant information, while matches in remaining list are probably random coincidences or referring to very general terms in the top hierarchy.

The key questions for ranking results are the following:

- How large is input query of genes G_q?
- How large is the set of genes G_t, annotated to matching GO term t?
- How many genes of the input query G_q are common to GO term t, *i.e.* how large is intersection $G_q \cap G_t$?
- How many genes are there all together in the genome, how large is G?

There are various measures for estimating importance of results through above questions. Next subsections describe a few of these.

2.4.1 Precision and recall

The decision whether a term t matches a query G_q is made upon the intersection of two sets $G_q \cap G_t$. The most intuitive method to rank the resulting intersection is comparing it with its basic set components G_q and G_t. Set ratios principally assume that the goal of user query G_q is to retrieve the best possible match to relevant results in an annotation set G_t.

Precision $\mathbf{pre}(G_q, G_t) \in [0, 1]$ measures the fraction of relevant results in all retrieved results, *i.e.* the fraction of common genes $G_q \cap G_t$ within user query G_q.

Precision is also called *positive prediction value* $\mathbf{ppv}(G_q, G_t)$, measuring the amount of *true positives* $G_q^{t^+}$ and *false positives* $G_q^{t^-}$ within retrieved results. True positives are genes common to query G_q and term annotation set G_t. False positives are these genes retrieved by query G_q, that were not annotated to term t.

$$\mathbf{pre}(G_q, G_t) = \frac{|G_q \cap G_t|}{|G_q|} = \mathbf{ppv}(G_q, G_t) = \frac{|G_q^{t^+}|}{|G_q^{t^+} \cup G_q^{t^-}|} = \frac{|G_q \cap G_t|}{|(G_q \cap G_t) \cup (G_q \backslash G_t)|}$$

Recall $\mathbf{rec}(G_q, G_t) \in [0, 1]$ measures the proportion of retrieved relevant results in all relevant results. In our case, recall shows the fraction common genes $G_q \cap G_t$ within the annotation set G_t.

Recall is also called *sensitivity* $\mathbf{sen}(G_q, G_t)$, measuring the amount of true positives and false negatives $G_t^{q^-}$ of available relevant results. False negatives are those genes of term t that were not retrieved by user query G_q.

$$\mathbf{rec}(G_q, G_t) = \frac{|G_q \cap G_t|}{|G_t|} = \mathbf{sen}(G_q, G_t) = \frac{|G_q^{t^+}|}{|G_q^{t^+} \cup G_t^{q^-}|} = \frac{|G_q \cap G_t|}{|(G_q \cap G_t) \cup (G_t \backslash G_q)|}$$

F-measure $\mathbf{F}(G_q, G_t) \in (0, 1]$ is a combined measure of precision and recall.

$$\mathbf{F_n}(G_q, G_t) = \frac{(1 + n^2) \cdot \mathbf{pre}(G_q, G_t) \cdot \mathbf{rec}(G_q, G_t)}{(n^2 \cdot \mathbf{pre}(G_q, G_t)) + \mathbf{rec}(G_q, G_t)}$$

In most common case, weighted harmonic mean $\mathbf{F_1}(G_q, G_t)$ balances weight and precision equally. Other often used measures are $\mathbf{F_{0.5}}(G_q, G_t)$ and $\mathbf{F_1}(G_q, G_t)$ that weigh either precision or recall twofold.

$$\mathbf{F_1}(G_q, G_t) = \frac{2 \cdot \mathbf{pre}(G_q, G_t) \cdot \mathbf{rec}(G_q, G_t)}{\mathbf{pre}(G_q, G_t) + \mathbf{rec}(G_q, G_t)}$$

Overlap $\mathbf{ovr}(G_q, G_t)$ is a simple set ratio, combined of the above measures of precision and recall. Overlap estimates the raw similarity between the query G_q and the genes of term t, by measuring the fraction of true positives within all related genes. Overlap disregards the different notions of false positive and false negative results, and treats these commonly as mismatches. This loss of information is misleading in cases where sizes of components sets differ greatly.

$$\mathbf{ovr}(G_q, G_t) = \frac{|G_q \cap G_t|}{|G_q \cup G_t|} = \frac{|G_q \cap G_t|}{|G_q| + |G_q| - |G_q \cap G_t|}$$

Information expressed by set ratios is often not sufficient to convey the importance of a match. Let us consider three trivial examples. Firstly, let us view the matches of a one-gene query $G_q = \{g_1\}$. As there are numerous terms with only one annotated gene (Figure 1.4), chances are good that a term t is found, so that $\mathbf{pre}(G_q, G_t) = \mathbf{rec}(G_q, G_t) = \mathbf{F_1}(G_q, G_t) = 1$. Secondly, let us consider

root terms t_0.. For any incoming query G_q, recall $\mathbf{rec}(G_q, G_{t_0}) \simeq 1$ is very good. Thirdly, fairly large queries G_q will often match a term with small annotation set, resulting in very good precision values $\mathbf{pre}(G_q, G_t) \simeq 1$.

On the other hand, if the query has a few hundred genes, it is almost impossible to get very good precision and recall. It is clear, that if the user actually presents a large and rare gene query, even results with moderately low precision and recall values can reveal relevant biological information. Statistical methods should be investigated to overcome the limitations of set ratios.

2.4.2 Statistical significance

Statistical significance describes the notion of events occurring by greater or smaller chance. An occurred result is considered *significant*, if it is unlikely to have occurred by random chance. Chance is measured with a probability. The smaller is the probability of random occurrence, the higher is the significance of the event, and we have more *statistical evidence* to assume that the event occurred on purpose.

Significance measure is used in *hypothesis testing*. A *null hypothesis* H_0 is set up to be rejected, in order to support an *alternative hypothesis* H_1. Null hypothesis is presumed true until statistical evidence in the form of a statistical hypothesis test indicates otherwise.

In case of determining significance of term t matching a user query G_q, the hypotheses are presented as follows.

$$\begin{cases} H_0 & \text{Term } t \text{ has matched query } G_q \text{ by chance} \\ H_1 & \text{Term } t \text{ has not matched } G_q \text{ by chance, and the event may contain} \\ & \text{interesting biological information} \end{cases}$$

In order to nullify H_0, we need to perform a statistical test and examine the probability of the match $G_q \cap G_t$ as $\mathbf{p}(G_q, t)$. The probability value $\mathbf{p}(G_q, t)$ (often referred to as *p-value*) is defined as the probability of observing current event, summed with the probabilities of any more extremal events unfavourable to H_0. In our case, we need to observe the current intersection $G_q \cap G_t$, and all possible larger intersections for given G_q and G_t.

If the probability of an event to occur by random chance is sufficiently small, we may accept H_1. The border between significant and insignificant results is defined as significance level. *Significance level* α of a statistical test is maximum probability of *Type I error, i.e.* the probability of accidentally rejecting a true null hypothesis H_0 and introducing false positives. Commonly used significance levels are $\alpha = 0.01$, $\alpha = 0.05$, $\alpha = 0.10$. In our case, significance level means the chance of proposing a randomly appearing match to be statistically significant.

$$\mathbf{p}(G_q, t) \begin{cases} > \alpha, & \text{Retain } H_0\text{, result appeared by chance} \\ \leqslant \alpha, & \text{Accept } H_1\text{, result is significant} \end{cases}$$

A stronger significance level α may result in more frequent *Type II errors*. Type II error means that the true *alternative hypothesis* H_1 is rejected in a statistical test and a false negative result is introduced. This means that actually significant matches are considered random and therefore ignored.

2.4.3 Hypergeometric probability. Fisher's exact test

One commonly used statistical significance measure is *hypergeometric probability* $\mathbf{p_h}(k = x)$, well described by the following classical urn problem.

Let us pick by chance and without replacement n balls from an urn of N black and white balls. What is the probability of getting exactly $k = x$ white balls and $n - k$ black balls, if the urn is known to contain K white and $N - K$ black balls?

Hypergeometric probability is calculated as follows.

$$\mathbf{p_h}(k = x) = \mathbf{p_h}(n, k; N, K) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

In the case of evaluating common elements of a gene query G_q and an annotation set G_t, the value N is interpreted as the total number of genes in the genome, and K as the number of genes annotated to term t. The value n is user query length, and k counts genes in intersection $G_q \cap G_t$. Above formula may be rewritten using the following parameters.

$$\mathbf{p_h}(|G_q \cap G_t| = x) = \mathbf{p_h}(|G_q|, |G_q \cap G_t|; |G|, |G_t|)$$

Fisher's exact test of statistical significance is used to examine the importance of association between two variables in a 2×2 *contingency table*, where the data is divided into two independent categories in two different ways. The contingency table for our gene annotation analysis is shown in Table 2.1. Fisher showed that the probability for obtaining the set of values in contingency tables in given by the previously defined hypergeometric distribution.

Fisher's test is also used in several publicly available Gene Ontology tools, such as FatiGO [ASDUD04], GoMiner [ZFW+03], BinGO [MHK05] and FuncAssociate [BKB+03]. Some of these tools apply a slightly different version of the contingency table by viewing only direct associations or combining terms of a specific hierarchy level [OZC04]. A comparative analysis of a few GO tools is available in a recent publication [MP05].

	$g \in G_t$	$g \notin G_t$	$\bigcup_g$
$g \in G_q$	$G_q \cap G_t$	$G_q \backslash G_t$	G_q
$g \notin G_q$	$G_t \backslash G_q$	$G \backslash (G_q \cup G_t)$	$G \backslash G_q$
$\bigcup_g$	G_t	$G \backslash G_t$	G

Table 2.1: Contingency table for association between query G_q and annotation set G_t of term t in genome G

Hypergeometric probability formula and equivalent contingency table for Fisher's test evaluate particular arrangement of data, and give exact probability of matching G_q genes to a term t. Exact arrangement $G_q \cap G_t$ is not sufficient according to the definition of p-value. In order to assess significance of match, we need to consider the probability of exact arrangement, as well as all possible more extreme arrangements.

This is known as *one-tailed Fisher's test* and expressed by *cumulative hypergeometric probability* $\mathbf{p_{ch}}(k \geqslant x)$. A graphical example of hypergeometric distribution can be seen in Figure 2.1. Following formula presents cumulative hypergeometric probability for picking x *or more* white balls from the classical urn.

$$\mathbf{p_{ch}}(k \geqslant x) = \mathbf{p_{ch}}(n, k; N, K) = \sum_{k=x}^{n} \mathbf{p_h}(n, k; N, K) = \sum_{k=x}^{n} \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

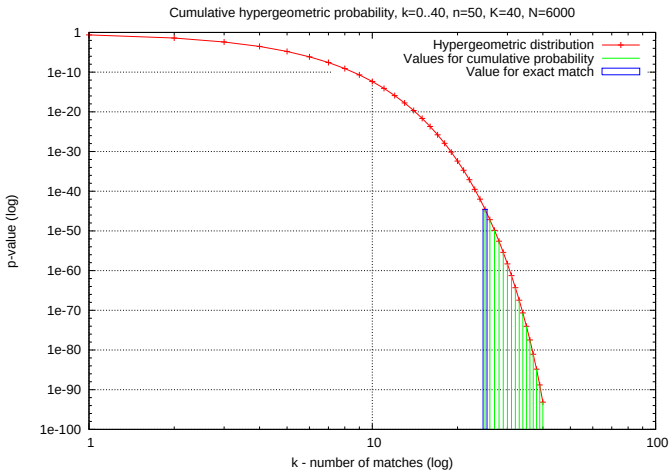

Figure 2.1: Hypergeometric distribution for $N = 6000$, $K = 40$, $n = 50$, $k \geqslant 25$

For assessing the significance $\mathbf{p}(G_q, t)$ of an intersection $G_q \cap G_t$, we need to consider hypergeometric probability of current intersection, as well as any more extreme arrangements. It is obvious that the intersection $G_q \cap G_t$ cannot be larger than one of its initial set components G_q and G_t.

$$\mathbf{p}(G_q, t) = \mathbf{p_{ch}}(|G_q \cap G_t| \geqslant x) = \mathbf{p_{ch}}(|G_q|, |G_q \cap G_t|; |G|, |G_t|) = \\ = \sum_{x=|G_q \cap G_t|}^{\mathbf{min}(|G_q|,|G_t|)} = \mathbf{p_h}(|G_q|, x; |G|, |G_t|)$$

2.4.4 Multiple testing

Multiple testing refers to increased amount of Type I errors when independent or dependent statistical tests are performed repeatedly [GDS03]. The idea is well described by an example of coin tossing. If a coin shows tail 95 times out of 100 tries, we may suggest that the coin is not fair. However, if this happens within a series of a million tosses, the significance of $95/100$ is not very extraordinary.

Let our experiment involve statistical tests with a set of terms T_q that match a gene query G_q. Let there be n such terms $t \in T_q$. For every term t we perform

a statistical test $p = \mathbf{p}(G_q, t)$, and we compare resulting p-value against a significance level to filter out random matches. In order to keep the experiment-wide significance level at predefined α, we may need to consider a stronger significance level α_i for every individual test of the experiment.

Bonferroni correction [Bon36] is a simple and well-known *Family Wise Error Rate* p-value correction for multiple testing. Family Wise Error Rate measures the probability of at least one Type I error within experiment. Bonferroni correction only takes into account only number of performed independent or dependent tests n in given experiment, and defines individual significance level α_B as follows. Every p-value above *Bonferroni corrected significance level* α_B is discarded as insignificant.

$$\alpha_B = \frac{\alpha}{n} = \frac{\alpha}{|T_q|}$$

Two different approaches for calculating α_B are mentioned in literature. A more common approach above considers the number of tests equal to the number of matching terms t to a specific query G_q [GDS03]. *Osier et al*, on the other hand, suggest that n should be considered equal to the number of all annotated terms T_G of the genome G [OZC04]. The first case would involve correction for a few hundred tests, while the second case observes tests with several thousand terms.

$$\alpha_B = \frac{\alpha}{n} = \frac{\alpha}{|T_G|}$$

Bonferroni correction is considered rather conservative in the sense that it increases the rate of Type II errors and discards some truly significant results. As any term t may have many parents T_t and genes are automatically annotated to all ancestors T'_t, the correction is very strong because of numerous results, and few if any of resulting terms remain significant [ZFW$^+$03]. Correction for all terms proposed by [OZC04] is even more conservative. Matches with smaller annotation sets never appear significant, as even a 100% overlap of G_q and G_t will not result in a sufficiently high p-value. Unless indicated otherwise, we will use the first approach $\alpha_B = \frac{\alpha}{|T_q|}$ throughout the work.

A more liberal multiple testing correction for independent tests is *False Discovery Rate* (FDR), that measures expected proportion of Type I errors within results [GDS03]. Benjamini-Hochberg FDR approach takes into account the observed p-values in the experiment [BH95]. The latter is often considered more ap-

plicable for Gene Ontology analysis than Bonferroni correction [OZC04, MHK05].

Let α be defined significance level for an experiment of n statistical tests. FDR method sorts n p-values from tests in increasing order, and picks the largest p-value, p_j, that is smaller than its proportional significance level. Proportional significance level for a p_i is calculated as the experiment-wide level α, multiplied by the fraction $\frac{i}{n}$ that presents the position of current p_i in the increasing list of n p-values. Every p-value above *Benjamini-Hochberg corrected significance level* $\alpha_{BH} = p_j$ is discarded as insignificant.

$$\alpha_{BH} = \mathbf{max}(\{p_j \in \{p_1, p_2, \ldots, p_n\} | p_j \leqslant \frac{j \cdot \alpha}{n}\})$$

Above FDR method is argued improper for analysis of gene queries, as GO terms are hierarchically related, gene annotation sets are highly intersecting because of True Path Rule, and therefore, statistical tests $\mathbf{p}(G_q, t)$ with many matching terms $t \in T_t$ should not be considered entirely independent of each other [Slo02]. It is not yet clear whether Gene Ontology hierarchy complies with variants of FDR designed for unindependent testing [MHK05].

2.5 Simulation of significance thresholds

Above sections described a number of different approaches for ranking terms that match a user query G_q, and filtering out matches that have possibly occurred by random chance. These well-known and rather general statistical methods are widely used in Gene Ontology tools. However, there is no consensus whether means such as Fisher's one-tailed test with multiple testing corrections actually comply with the partly dependent and hierarchically related annotation sets of Gene Ontology and KEGG pathways.

2.5.1 Experimental approach

This subsection describes a simulation experiment that observes possible p-values resulting in Fisher's one-tailed tests with randomised data and investigates possible multiple testing correction values. A brief outline of our experiment is found in the paragraphs below, while a more detailed and technical description of the simulation is depicted in Algorithm 1.

The goal of our experiment is to propose a method for distinguishing truly significant terms from random matches. The method is in accordance with multiple testing principles and takes into account the hierarchical and partly dependent gene annotation sets of GO and KEGG pathways.

For a given genome G and its GO and KEGG annotations, we propose an *experimental significance threshold* β_l for some user query length $l = |G_q|$. Thresholds β are based on best p-values that normally result randomised queries. Out of the best p-values of multiple simulations, we pick such a threshold p-value that would correspond to the significance level α, so that most simulations would result in a larger p-value. We compare thresholds with two widely used multiple testing corrections α_B and α_{BH}.

Our experiment involves the baker's yeast genome and its GO and KEGG annotations from March 26^{th}, 2006, having alltogether $|G| = 6471$ genes with known annotations to $|T_G| = 4145$ terms. We perform the experiment for query lengths $l \in \{1, \ldots, 1000\}$. We repeat the experiment with mouse genome annotations, having $|G| = 17024$ genes and $|T_G| = 6597$ terms.

First we fix a user query length l and the experiment-wide significance level $\alpha = 0.05$, and randomly generate $r = 1000$ queries G_q of length l, using the genes in the genome. For every synthetic query G_q, we observe p-values $\mathbf{p}(G_q, t)$ of all matching terms $t \in T_q$. We store the lowest p-value p of every query, as well as Bonferroni corrected significance level α_B and Benjamini-Hochberg corrected significance level α_{BH} for this query. It is important to note that all these probabilities correspond to true H_0 hypothesis, as any terms t matching the queries must have occurred by random chance.

The simulation results in a thousand best p-values $\{p\}$, corrected significance levels $\{\alpha_B\}$ and $\{\alpha_{BH}\}$ for the given query length l. Out of these sets, we pick such values p', α'_B and α'_{BH}, that correspond to the original experiment-wide significance level α. This means that given an $\alpha = 0.05$ and $r = 1000$, 50^{th} gene in the ordered list would correspond to the 5% margin.. We call this value $\beta_l = p'$, the *experimental significance threshold* for query length l.

Our experimental significance threshold β fulfils the requirements of multiple testing, as experiment-wide significance level α is tested only once against best possible matches of all simulated user queries, and therefore, correction is not needed.

Algorithm 1 Simulation of significance threshold β and levels α'_B, α'_{BH}

Require: $l > 0$ {*Length of synthetic query*}
Require: $0 < \alpha < 1$ {*Experiment-wide significance level*}
Require: $r > 0$ {*Number of simulations*}
Ensure: β {*Significance threshold for given length* l}
Ensure: α'_B {*Bonferroni corrected significance level for length* l}
Ensure: α'_{BH} {*Benjamini-Hochberg corrected significance level for length* l}
  $P := [\,]$ {*Declare empty arrays for storing values*}
  $A_B := [\,]$
  $A_{BH} := [\,]$
  for all $x \in \{1, 2, \ldots, r\}$ **do**
    $G_q := \mathbf{rand}(l, G)$ {*Create random query of length* l *of genes in* G}
    $P_q := [\,]$
    for all $t \in T_q$ **do**
      $\mathbf{p}(G_q, t) \rightarrow P_q$ {*add p-value to array*}
    end for
    $n := |T_q|$
    $\alpha/n \rightarrow A_B$ {*Bonferroni correction*}
    $\mathbf{min}(\{p_j \in P_q\}) \rightarrow P$ {*Best p-value*}
    $\mathbf{sort_{asc}}(P_q)$
    $\mathbf{max}(\{p_j \in P_q | p_j \leqslant j \cdot \alpha/n\}) \rightarrow A_{BH}$ {*Benjamini-Hochberg correction*}
  end for
  $i := \mathbf{round}(r \cdot \alpha)$ {*Index corresponding to* α *level*}
  $\mathbf{sort_{asc}}(P)$ {*Sort values in ascending order*}
  $\mathbf{sort_{asc}}(A_B)$
  $\mathbf{sort_{asc}}(A_{BH})$
  return $(\beta, \alpha_B, \alpha_{BH}) = (P[i], A_B[i], A_{BH}[i])$

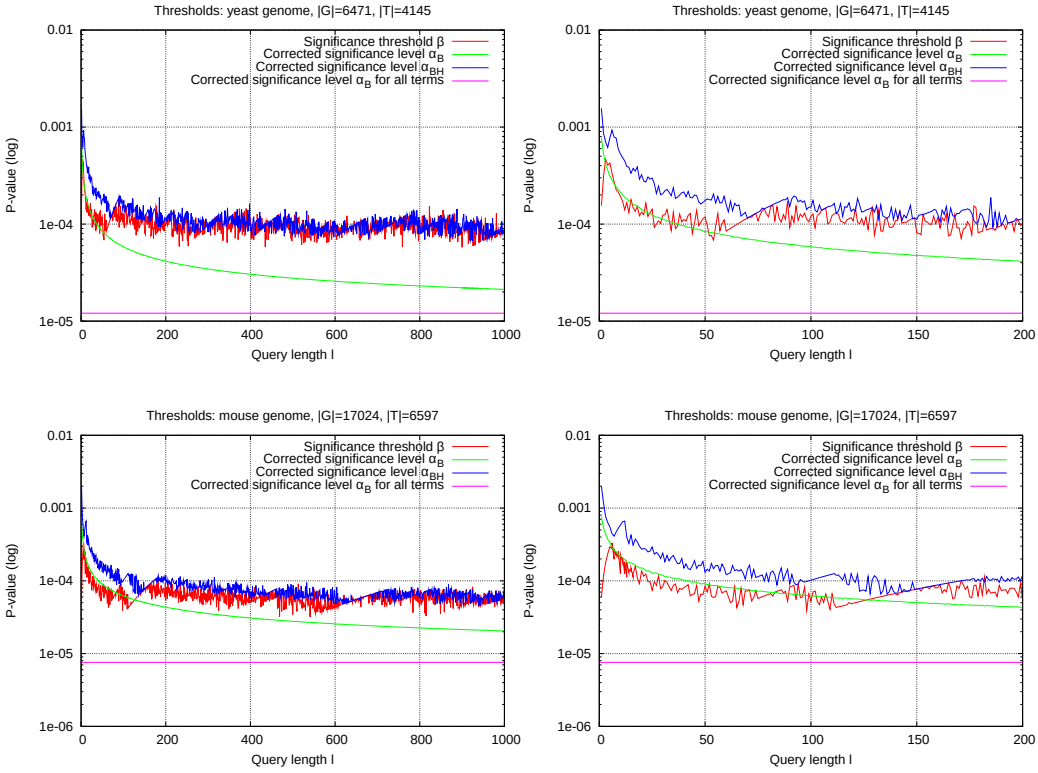

Figure 2.2: Corrected significance levels α_B, α_{BH} and thresholds β for baker's yeast (top) and mouse (bottom), simulated with $r = 1000$ synthetic queries at experiment-wide significance level $\alpha = 0.05$.

Results of our experiment are shown in Figure 2.2. Bonferroni corrected significance levels α_B reflect the number of matching terms that normally correspond to queries of no statistical significance. The smooth shape of α_B over different query lengths l suggests that there is a rather constant number of terms corresponding to an average gene. As query length increases, more genes are likely to match same terms. Bonferroni corrected level α_B depends only on number of matching terms, and therefore, corrected value seems to converge to a constant value as query length increases.

Benjamini-Hochberg corrected levels α_{BH} and significance thresholds β de-

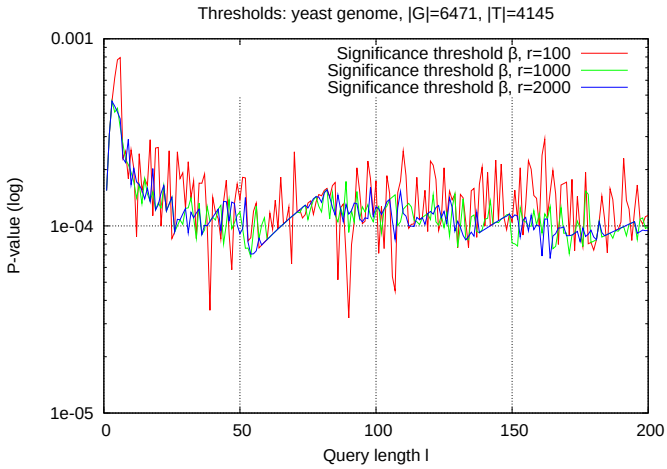

Figure 2.3: Baker's yeast significance thresholds β simulated with $r = 100$, $r = 1000$, $r = 2000$ synthetic queries at experiment-wide significance level $\alpha = 0.05$.

pend on actual p-values in a given query, and convey better the hierarchical and dependent intersections within annotation sets. The frequent hops of probabilities are caused by random sampling, as well as discrete nature of hypergeometric probability function, which decreases sharply when additional genes are added to intersection $G_g \cap G_t$.

We also performed a small experiment to observe the quality of experimental thresholds β with different simulation parameters r. Results in Figure 2.3 are rather expected. If less simulations are performed, β value becomes noisier, while more excessive randomisation will result in a clearer threshold. However, the threshold value will preserve its discreet and jagged nature.

It should be noted that for smaller query lengths l, experimental significance threshold β is very similar to Bonferroni corrected level, $\beta \simeq \alpha_B$. As the query length increases, Bonferroni correction becomes far too conservative and experimental significance threshold seems to approach Benjamini-Hochberg corrected level, $\beta \rightarrow \alpha_{BH}$.

The drawback of our experiment is the fact that it is computationally quite expensive. The values β for the yeast genome with $r = 1000$ queries of lengths $l = (1, \ldots, 1000)$ required more than 5 hours of work in a cluster of 20 computers.

Experimental approach for obtaining significance thresholds is clearly unsuitable for practical use and further methods should be investigated for calculating thresholds.

2.5.2 Analytical approach

This subsection constructs a probabilistic method for estimating significance thresholds β obtained through extensive simulations of above experiment.

Let us first fix a certain user query length l and a genome G. The collection of annotation sets $\mathcal{G}$ contains all gene annotation sets G_t of terms $t \in T_G$. This time we do not need the actual collection of sets, but the structure of collection. Let $(i_1, i_2, \ldots, i_m)$ include all possible set sizes of collection $\mathcal{G}$, and $(j_1, j_2, \ldots, j_m)$ include respective counts. In other words, for a given k, there are j_k sets of size i_k in collection $\mathcal{G}$.

Let S_i be an annotation set of fixed size i from the collection $\mathcal{G}$, let j denote the number of such sets in the collection $\mathcal{G}$. Let us observe all possible p-values $\mathbf{p}(G_q, S_i)$ that may be obtained from Fisher's one-tailed test of cumulative hypergeometric probability, when comparing the set S_i with any query G_q of fixed length l.

We need to consider every possible size of intersection $G_q \cap S_i$, and calculate corresponding p-value. It is clear that the intersection cannot exceed the size of its component sets G_q and S_i. We gather p-values into the set P.

$$P = \{p_1, p_2, \ldots, p_y | y = \mathbf{min}(|G_q|, i) \wedge \forall p_x \in P : p_x = \mathbf{p_{ch}}(|G_q|, x, |G|, i)\}$$

In order to decrease computational costs, we reduce probability space P to the range $(10^{-6}, 10^{-3}]$, only accounting for such "good" p-values that may or may not appear by random chance. This range is based on empirical observations of significance thresholds from previously described experiment. We call this range the *threshold range*.

$$P = P \backslash \{p \in P | p \leqslant 10^{-6} \vee p > 10^{-3}\}$$

We map p-values $p \in P$ for annotation set size i into an array of thousand elements $\mathcal{P}_i = (p_{1,i}, p_{2,i}, \ldots, p_{1000,i})$. Let γ denote an index in the array $\mathcal{P}_i$, and $p_{\gamma,i}$ denote corresponding p-value. We map p-values to the array proportionally,

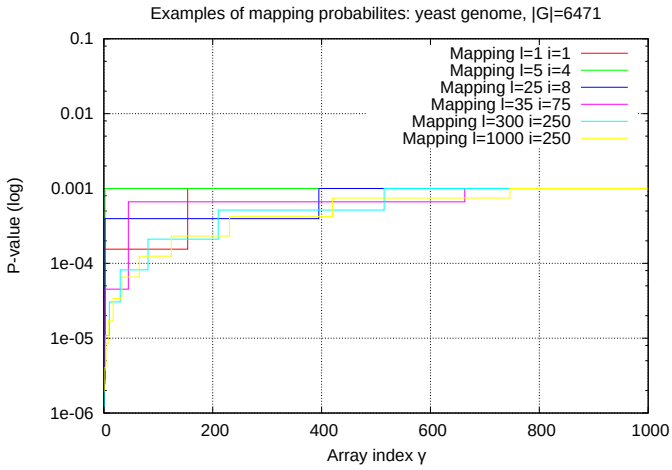

Figure 2.4: Examples of mapping p-values in the range $(10^{-6}, 10^{-3}]$ for different query lengths l and set sizes i, to an array of 1000 elements.

so that a few first elements in the array present the lowest p-value in the threshold range, following larger portion of elements present the second-lowest p-value, etc.

Every index γ of the array $\mathcal{P}_i$ presents a $\frac{1}{1000}$ fraction of the threshold range $(10^{-6}, 10^{-3}]$, and every corresponding p-value $p_{\gamma,i}$ shows the smaller closest p-value, that can be obtained from the test $\mathbf{p}(G_q, S_i)$ with some intersection $G_q \cap S_i$.

For a single set size i, the p-value $\mathbf{p}(G_q, S_i)$ stays on a constant level until a change in intersection $G_q \cap S_i$ causes a discrete jump. When first intersection appears with p-value greater than the range $(10^{-6}, 10^{-3}]$, all following elements from the jump to the last element will be stored as the upper limit 10^{-3}. Some examples of arrays can be seen in Figure 2.4.

Let us fix a certain array index γ and the corresponding p-value $p_{\gamma,i}$. We calculate the probability of opposite event, that is, the chance that we do not get such an intersection with the query G_q and the set S_i, that results with a p-value in the threshold range $(10^{-6}, 10^{-3}]$.

$$p'_{\gamma,i} = 1 - p_{\gamma,i}$$

The probability of two independent events a and b occurring at the same time

is the product of probabilities of these events.

$$\mathbf{p}(a \wedge b) = \mathbf{p}(a) \cdot \mathbf{p}(b)$$

The events of multiple annotation sets S_i matching a user query G_q may not be considered entirely independent, as hierarchical relations between terms create dependencies. However, we attempt to construct probabilities using the above formula.

There are j sets S_i with size i among the annotation sets in $\mathcal{G}$. We gather these together into a collection $\mathcal{S}_i$. We use the above product and calculate the probability $p'_{\gamma,\mathcal{S}_i}$, that none of the sets in collection $\mathcal{S}_i$ will intersect the query with a probability in the range $(10^{-6}, 10^{-3}]$. In other words, we need the probability that all sets of fixed size i simultaneously give a intersection with a p-value either above 10^{-3} or below 10^{-6}.

$$p'_{\gamma,\mathcal{S}_i} = (1 - p_{\gamma,i})^{(1)} \cdot (1 - p_{\gamma,i})^{(2)} \cdot \dots \cdot (1 - p_{\gamma,i})^{(j)} = \prod_{k=1}^{j} p'_{\gamma,i}$$

We may use the same idea again, and calculate the probability $p'_{\gamma,\mathcal{G}}$ that none of the sets in the collection $\mathcal{G}$ will result with p-value $(10^{-6}, 10^{-3}]$ in intersecting the query, or in other words, all of these sets will at the same time give a match with p-value out of the threshold range. The collection of annotation sets $\mathcal{G}$ has sets with m different sizes $(i_1, i_2, \dots, i_m)$. The *overall probability* of no matches is therefore a product of probabilities for all different set lengths.

$$p'_{\gamma,\mathcal{G}} = p'_{\gamma,\mathcal{S}_{i_1}} \cdot p'_{\gamma,\mathcal{S}_{i_2}} \cdot \dots \cdot p'_{\gamma,\mathcal{S}_{i_m}} = \prod_{i \in \{i_1, i_2, \dots, i_m\}} p'_{\gamma,\mathcal{S}_i}$$

The value $1 - p'_{\gamma,\mathcal{G}}$ is essentially the same as experiment-wide significance level α. It shows overall probability that a user query G_q of length l gets such a good intersection, so that the resulting p-value is in the range $(10^{-6}, 10^{-3}]$.

$$1 - p'_{\gamma,\mathcal{G}} = \alpha$$

Now we return to the idea of mapping p-values of annotation sets S_i to arrays $\mathcal{P}_i$, where every index γ presents a thousandth of the threshold range $(10^{-6}, 10^{-3}]$, and the $p_{\gamma,i} \in \mathcal{P}_i$ shows the smaller closest p-value that can be achieved with an

intersection $G_q \cap S_i$. P-values corresponding to consecutive indices γ are either constant or change discreetly as the intersection $G_q \cap S_i$ changes. However, if we view many arrays of different set sizes i simultaneously for the overall probability, increasing probabilities corresponding to consecutive indices become less discreet.

When we are given a predefined experiment-wide significance level $\alpha = 0.05$, we may use a technique of binary search to locate an index γ, that defines such p-values $p_{\gamma,i}$ in different arrays $\mathcal{P}_i$ for all different set sizes $i \in \{i_1, i_2, \ldots, i_m\}$, so that the overall probability $p'_{\gamma,\mathcal{G}}$ is closest to $1 - \alpha$.

Binary search is a divide-and-conquer type of algorithm, that works on sorted lists. The algorithm iteratively narrows down search space by comparing the search value with the middle element of the list, and deciding whether search should be continued in the first or the second half of the list.

In our case, search space would involve all possible γ values, and the γ giving the overall probability $p'_{\gamma,\mathcal{G}}$ closest to $1 - \alpha$ is retrieved as search result. We call this value γ_l, the *analytical significance threshold* for query length l at significance level α.

Figure 2.5 compares analytical threshold γ with experimental threshold β gained from synthetic queries. Values for γ show a similar pattern to experimental values β, however, analytical threshold function is more conservative. This can be explained with the fact that probability of simultaneous events $\mathbf{p}(a \wedge b) = \mathbf{p}(a) \cdot \mathbf{p}(b)$ is true if the events a and b are independent. Annotation sets and intersections with queries cannot be considered independent events, and therefore experimental results show higher probabilities. We define a correction x to overall probability $p'_{\gamma,\mathcal{G}}$. This correction influences binary search algorithm steps and returns a generally less conservative γ value.

$$p'_{\gamma,\mathcal{G}} = \frac{p'_{\gamma,\mathcal{G}}}{1 - x}$$

We experimented manually with different correction values and found out that $x = 0.03$ is a perfect correction for smaller query of up to 200 genes. (Figure 2.5). Later γ_x becomes more liberal than experimental threshold β. However, this increase is minute and should not influence general quality of results.

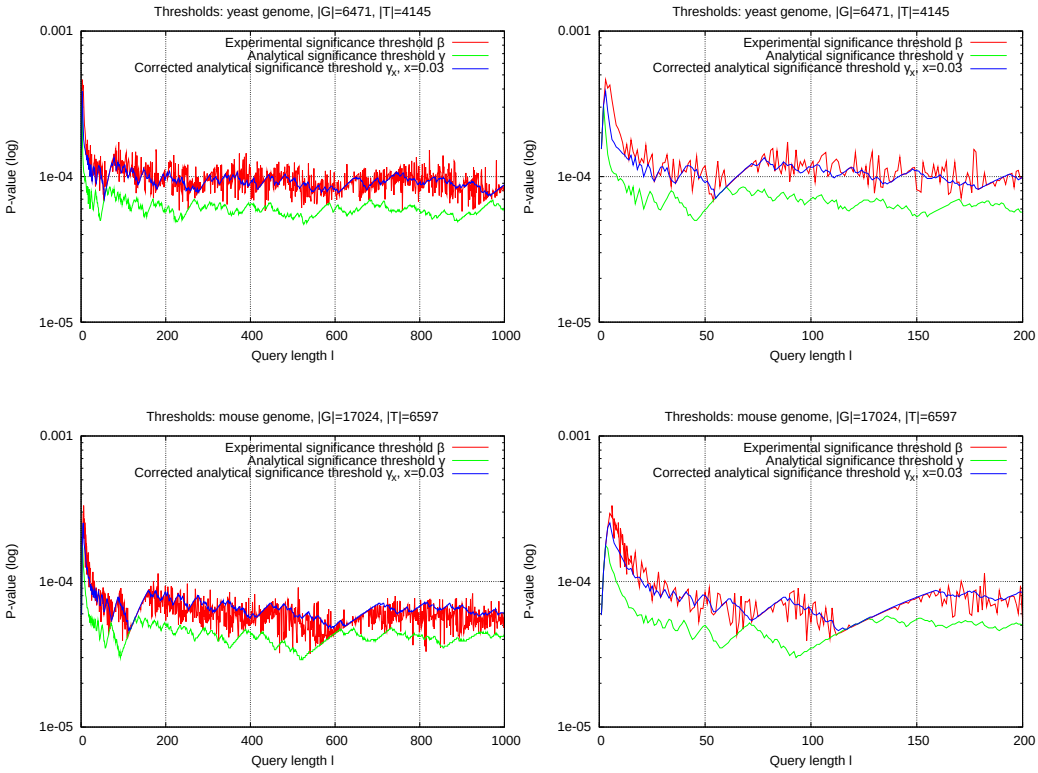

Figure 2.5: Experimental β and analytical γ significance thresholds for baker's yeast (top) and mouse (bottom). Thresholds β are simulated with $r = 1000$ synthetic queries. Experiment-wide significance level is $\alpha = 0.05$ for all thresholds. Analytical threshold γ_x uses correction $x = 0.03$.

Chapter 3

Mining GO with GOSt

Previous chapter defined a way of determining terms t that match a user query G_q. We concentrated on methods for estimating statistical significance of results, and described thresholds that discard presumably irrelevant and random matches.

This chapter investigates querying and mining Gene Ontology. We start off with a simple algorithm for determining all matches for a set of genes, define novel ordered queries and means for effective analysis of such, and conclude the chapter with a method for mining significant subgraphs of terms matching an ordered or unordered user query.

3.1 Simple queries. Method $\mathrm{GOSTMINER_1}$

One of the most common goals for Gene Ontology data analysis would involve determining annotations for a *query* of genes G_q. We say that a term *matches* the query, if G_q involves some genes that are annotated to term t, in other other words, the intersection $G_q \cap G_t$ is not an empty set.

The simplest algorithm $\mathbf{GOSTMINER_1}$ for determining all matches from GO/KEGG terms basically involves intersecting the user query G_q with all annotation sets G_t in the collection $\mathcal{G}$, and returning all such terms that have at least one annotated gene present in the query. The algorithm is shown in Algorithm 2.

Algorithm 2 $\mathbf{GOSTMINER_1}$: Determine matching terms to a user query G_q.

Require: $G_q = \{g_1, g_2, \ldots, g_l\}$ {*Query of genes of a given genome*}
Require: $\mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$ {*Annotations of the given genome*}
Ensure: $\mathcal{R} = \{(p, t, |G_q|, |G_t|, |G_q \cap G_t|)\}$ {*An array of results*}
  $\mathcal{R} := [\,]$ {*Array for storing results*}
  for all $G_t \in \mathcal{G}$ **do**
    if $|G_q \cap G_t| \neq \emptyset$ **then**
      $p := \mathbf{p_{ch}}(|G_q|, |G_q \cap G_t|; |G|, |G_t|)$
      $(p, t, |G_q|, |G_t|, |G_q \cap G_t|) \rightarrow \mathcal{R}$ {*Add matching term info to results*}
    end if
  end for
  return $\mathcal{R}$

3.2 Ordered queries. Method $\mathbf{GOSTMINER_2}$

So far we have treated the query as a plain unstructured set of genes. Another relevant approach involves the analysis of queries, where genes are sorted according to some measure. This novel feature is not common in popular Gene Ontology tools.

For example, we may view expression activity of hundreds of genes in a microarray knock-out experiment and sort genes according to their expression activity. Then most active genes on top of the sorted list would probably describe organism's counterreaction to the knockout. We would like to analyse different fractions of top-ranking genes, observe p-values, and for every matching term, determine length of list where the term reaches peak of significance.

Let $\tau(g)$ denote a measure of gene g. We may sort a gene query G_q according to the values of τ and create internal structure to the query. Then we may define an descending *ordered query* G_s from the original query G_q.

$$G_s = \{g_1, g_2, \ldots, g_l | \forall g_i, g_j \in G_s : i < j \Rightarrow \tau(g_i) \leqslant \tau(g_j)\}$$

An ascending ordered query is defined in a similar way.

Query G_s may be seen as a cluster of genes, where first element g_1 is the central element. A few elements in the beginning of the query are closely related to the first element, and as the rank i of gene increases, the greater becomes distance

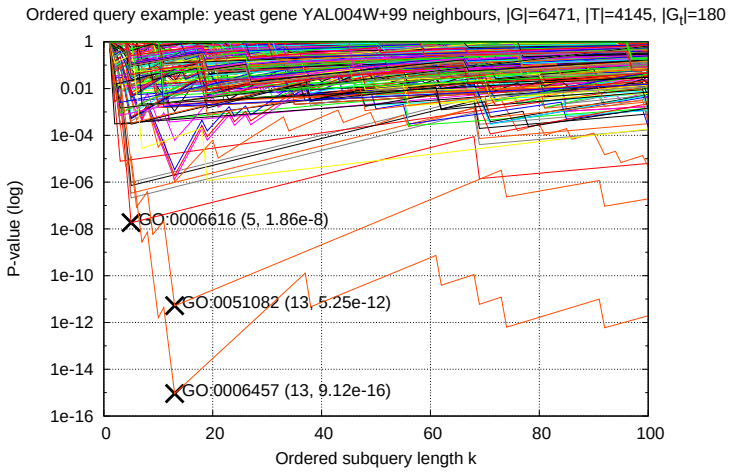

Figure 3.1: Yeast genes **YAL004W**+99 neighbours in an ordered query G_s, matching 180 GO/KEGG terms. Best peak significance levels for three terms are displayed. Term `GO:0051082` denotes *unfolded protein binding* (BP), and `GO:0006457` (MF) protein folding. The term `GO:0006616` is a specific type of *protein targeting* (BP). The latter two terms are hierarchically related through a common ancestor. The representatives of these terms are tightly clustered around the gene **YAL004W**.

between g_i and g_1. In other words, tightness of the cluster decreases, as we view more genes in the ordered list.

Let $G_{s,k} = \{g_1, g_2, \ldots, g_k\} \subseteq G_s$ denote a portion of k top-ranking genes from the head of the sorted list in query G_s. We call this portion $G_{s,k}$ the *ordered subquery* of G_s.

The main idea for ordered queries is determining such a portion of top-ranking genes from the beginning of the query, that would give the best probability, called *peak significance*, for a given GO/KEGG term t. Then we can disregard the rest of the query as insignificant in the context of current term t.

Figure 3.1 demonstrates an example of finding matching terms to an ordered query G_s. The query consists of yeast gene **YAL004W** and 99 its closest neighbours in an experiment of gene expression. The query matches alltogether 180 terms from GO and KEGG pathways. Three best peak significances for GO terms are labelled as examples.

A simple exhaustive method $\mathbf{GOSTMINER_2}$ for analysing ordered sets is

Algorithm 3 $\mathbf{GOSTMINER_2}$: Determine matching terms to an ordered user query G_s. Exhaustive method.

Require: $G_s = \{g_1, g_2, \ldots, g_l\}$ {*Query of genes of a given genome*}
Require: $\mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$ {*Annotations of the given genome*}
Ensure: $\mathcal{R} = \{(p, t, |G_q|, |G_t|, |G_q \cap G_t|)\}$ {*An array of results*}
  $\mathcal{R} := [\,]$ {*Array for storing results*}
  for all $k \in \{1, \ldots, l\}$ **do**
    {*Perform search for every subquery of* G_s}
    $G_{s,k} := [\,]$
    for all $i \in \{1, \ldots, k\}$ **do**
      {*Insert first* k *elements into query*}
      $g_i \rightarrow G_{s,k}$
    end for
    $\mathbf{GOSTMINER_1}(G_{s,k}, \mathcal{A}') \rightarrow \mathcal{R}$ {*Add results from Algorithm 2 to array*}
  end for
  return $\mathcal{R}$

shown in Algorithm 3. We may easily split the ordered query G_s into a collection of l subqueries for lengths $\{1, \ldots, l\}$, and analyse each of these separately with the existing $\mathbf{GOSTMINER_1}$ method (Algorithm 2). Then additional sorting and filters may be applied for detecting best p-values.

Algorithm $\mathbf{GOSTMINER_2}$ is ineffective in the sense that it does not take advantage of information gained from intersections with previously calculated subqueries. For a given term t and ordered user query G_s with length l, the intersection $G_{s,k} \cap G_t$ is calculated for every length $(1, \ldots, l)$, and a p-value is calculated every time since first gene is found in intersection. Further approaches should be investigated to minimise the amount of needed intersection and p-value operations.

3.3 Approximation of probability function

In this section, we observe hypergeometric p-value function, that evaluates the significance of an ordered query G_s and its subqueries $G_{s,k}$ matching term t

through the intersections $G_{s,k} \cap G_t$. We investigate means to minimise intersection and p-value calculation operations for improving the method proposed in the previous section.

As we can see on the example plot in Figure 3.1, hypergeometric probability function for the intersection $G_{s,k} \cap G_t$ is discreet, as the subquery length $k \in \{1, \ldots, l\}$ increases. Every additional gene in query may either *match* (g^+) or *miss* (not match) (g^-) the term t. Additional $g^+ \in G_{S,k}$ will cause a sharp decrease p-value, while new $g^- \in G_{S,k}$ will increase p-value of occurring intersection. Such a discrete function may well be approximated by calculating a selection of extremal values.

Let us define the parameter *smoothness* σ for p-value approximation function. Smoothness $\sigma = 0$ means, that p-values are exhaustively calculated for all possible query lengths using the algorithm $\mathbf{GOSTMINER_2}$. When smoothness parameter is above zero, we attempt to skip some query lengths for every term t, and approximate the p-values through lesser intersection points. The greater is smoothness, the faster is approximation, and the less exact values are calculated. Let us observe different smoothness parameters through the following example.

$$G_s = (g_1^-, g_2^+, g_3^+, g_4^+, g_5^+, g_6^-, g_7^-, g_8^+, g_9^-)$$
$$G_s \cap G_t = (g_2^+, g_3^+, g_4^+, g_5^+, g_8^+)$$

Let us first discuss strongest smoothness $\sigma = 3$. When we are interested only in the best p-values for all intersections with possible subqueries $G_{s,k}$, we need to consider only the subqueries where last element is a match g^+. In above example, this would involve queries $G_{s,2}, G_{s,3}, G_{s,4}, G_{s,5}, G_{s,8}$. Moreover, p-value for consecutive matching genes $\{\ldots, g_{i-1}^+, g_i^+, g_{i+1}^+, \ldots\}$ is decreasing in rather constant nature. We may skip those matching genes that have both previous and next genes present in intersection, and view only beginning and end of p-value decrease. This leaves us with queries $G_{s,2}, G_{s,5}, G_{s,8}$ in above example.

Strongest smoothness $\sigma = 3$ is insufficient, when we are interested in the actual shape of p-values over different subqueries. By observing only local minimums of p-values, the resulting probability curve is lossy as large p-values around local maximums appear smaller than they really are.

With smoothness parameter $\sigma = 2$, we also view some genes g^- that miss the term t and push p-value to a local maximum. P-value for consecutive missing

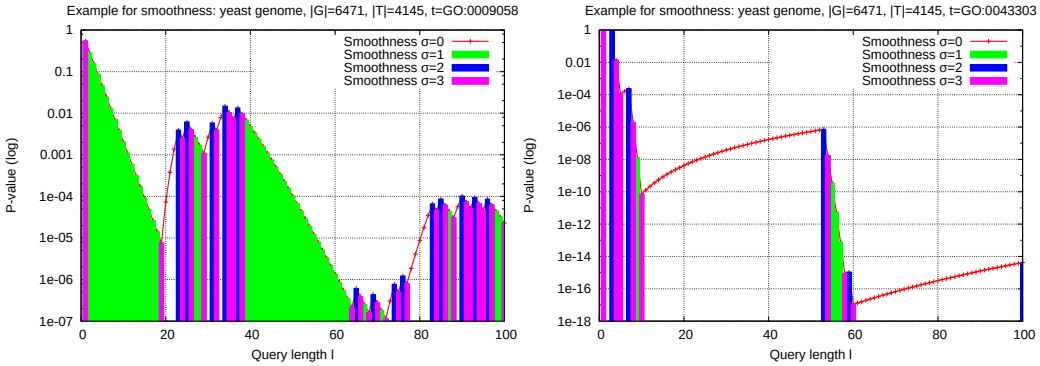

Figure 3.2: Examples of smoothness for an ordered query of 100 yeast genes. Figure on the left shows the probabilities for the general biological process *biosythesis*(`GO:0009058`). Figure on the right displays probabilities for a specific process *mast cell degranulation* (`GO:0043303`).

genes $\{\ldots, g_i^-, g_{i+1}^-, \ldots\}$ is rather constant as well. Therefore, we may only view local maximum of the worst miss, that is, the intersection where last gene is a miss. Local maximum p-value is reached directly before an additional match g^+ appears in the intersection. If first or last element of ordered query G_s is a miss, the respective subquery $G_{s,1}$ or $G_{s,l}$ presents a local maximum and should be included as well. In above example, we would view intersections with subqueries $G_{s,1}, G_{s,2}, G_{s,5}, G_{s,7}, G_{s,8}, G_{s,9}$. Approximation with smoothness parameter $\sigma = 1$ would also include intermediate matching genes g_3^+, g_4^+.

Some examples of analysing intersections using different smoothness parameters are available in Figure 3.2.

3.4 Ordered queries. Method GOSTMINER$_3$

This section describes an effective algorithm **GOSTMINER$_3$** that approximates p-values for analysing matches with term t across an ordered query. The method is available as Algorithm 4, while a less formal discussion follows below.

Let us fix a certain term t and an ordered user query $G_s = \{g_1, g_2, \ldots, g_l\}$. P-

values with above smoothness parameters may be effectively approximated from genes in the intersection $G_s \cap G_t = (g_1^+, g_2^+, \ldots, g_m^+)$. It is sufficient to calculate the intersection only once. Let us first define the function $\mathbf{pos}$, that shows the position of a gene in a query. The position is in the range $[1, l]$.

$$\mathbf{pos}(g_c, G_s) = c \Leftrightarrow \text{Gene } g_c \text{ is in ordered query } G_s \text{ at position } c.$$

We preserve internal sorted structure of genes $g \in G_s$ in the intersection $G_s \cap G_t$. The order of genes in the intersection is the original order in the query; any gene g_c^+ that appears before g_d^+ in the intersection $G_s \cap G_t$, must also appear before g_d^+ in the original query G_s.

For every intersection element g^+, we store its positions in an associative array I of *intersection points*. Every key k of array I is the position of a g^+ in intersection $G_s \cap G_t$, and every value j is the position of g^+ in the original ordered query G_s.

$$I = (1 \mapsto j_1, 2 \mapsto j_2, \ldots, m \mapsto j_m) = \\ = \{i \mapsto j_i | i = \mathbf{pos}(g_i^+, G_s \cap G_t) \wedge j_i = \mathbf{pos}(g_i^+, G_s)\}$$

For smoothness $\sigma = 3$, we need to consider only subqueries $G_{s,k}$, where last gene in intersection $G_{s,k} \cap G_t$ is a match g_k^+. We need to view intersection points $(i \mapsto j_i)$ of the array I. Every array value j_i shows the number of genes in the subquery, *i.e.* $j_i = |G_{s,k}| = k$, and every corresponding key i shows the number of genes in the subquery that have a match in term t, *i.e.* $i = |G_{s,k} \cap G_i|$. The values i and j_i are used in cumulative hypergeometric probability calculation.

$$\mathbf{p}(G_{s,k}, t) = \mathbf{p_{hc}}(j_i, i; |G|, |G_t|)$$

For smoothness $\sigma < 3$, we also need consider some subqueries $G_{s,u}$ that end with a miss g_u^- and cause a local maximum in p-values. Local probability maximum occurs directly before another match $g_k^+ \in G_s$ causes a p-value decrease. In order to detect such local maximums, we view all intersection points $(i \mapsto j_i)$ as above. In locations where a miss g_u^- is directly followed by a match $g_{u+1}^+ = g_k^+$, we also calculate cumulative hypergeometric probability for previous query length $|G_{s,u}| = j_i - 1$, and intersection size $|G_{s,u} \cap G_t| = i - 1$.

$$\mathbf{p}(G_{s,u}, t) = \mathbf{p_{hc}}(j_i - 1, i - 1; |G|, |G_t|)$$

Algorithm 4 $\mathbf{GOSTMINER_3}$: Determine matching terms to an ordered user query G_s. Approximate method.

Require: $G_s = \{g_1, g_2, \ldots, g_l\}$ {*Query of genes of a given genome*}
Require: $\mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$ {*Annotations of the given genome*}
Ensure: $\mathcal{R} = \{(p, t, |G_q|, |G_t|, |G_q \cap G_t|)\}$ {*An array of results*}
  $\mathcal{R} := [\,]$ {*Array for storing results*}
  for all $G_t \in \mathcal{G}$ **do**
    $(g_1, g_2, \ldots, g_m) := G_s \cap G_t$ {*Preserve order of genes in intersection*}
    $I := [\,]$ {*Declare empty array for gene positions*}
    for all $g_i \in G_s \cap G_t$ **do**
      $I[\mathbf{pos}(g_i, G_s \cap G_t)] := \mathbf{pos}(g_i, G_s)$ {*Store positions of intersecting genes*}
    end for
    if $I[1] \neq 1$ **then**
      $(1.0, t, 1, |G_t|, 0) \rightarrow \mathcal{R}$ {*Add max* $p = 1.0$*, missing 1st gene* $(g_1^-, \ldots)$}
    end if
    for all $i \in \mathbf{keys}(I)$ **do**
      {*Check every matching element*}
      if $\sigma > 1 \wedge 1 < i < m \wedge I[i+1] = I[i] + 1 \wedge I[i-1] = I[i] - 1$ **then**
        goto (next) {*Skip if consecutive matches* $(\ldots, g_{i-1}^+, g_i^+, g_{i+1}^+, \ldots)$}
      end if
      if $\sigma < 3 \wedge i > 1 \wedge I[i-1] \neq I[i] - 1$ **then**
        $p := \mathbf{p_{ch}}(I[i] - 1, i - 1; |G|, |G_t|)$
        $(p, t, I[i] - 1, |G_t|, i - 1) \rightarrow \mathcal{R}$
        {*Add max* p*, previous miss* $(\ldots, g_{i-1}^-, g_i^+, \ldots)$}
      end if
      $p := \mathbf{p_{ch}}(i, I[i]; |G|, |G_t|)$
      $(p, t, I[i], |G_t|, i) \rightarrow \mathcal{R}$ {*Add min p, match* $(\ldots, g_i^+, \ldots)$}
      label (next)
    end for
    if $\sigma < 3 \wedge I[m] < l$ **then**
      $p := \mathbf{p_{ch}}(l, m; |G|, |G_t|)$
      $(p, t, l, |G_t|, m) \rightarrow \mathcal{R}$ {*Add max* p*, miss last gene,* $(\ldots, g_l^-)$}
    end if
  end for
  return $\mathcal{R}$

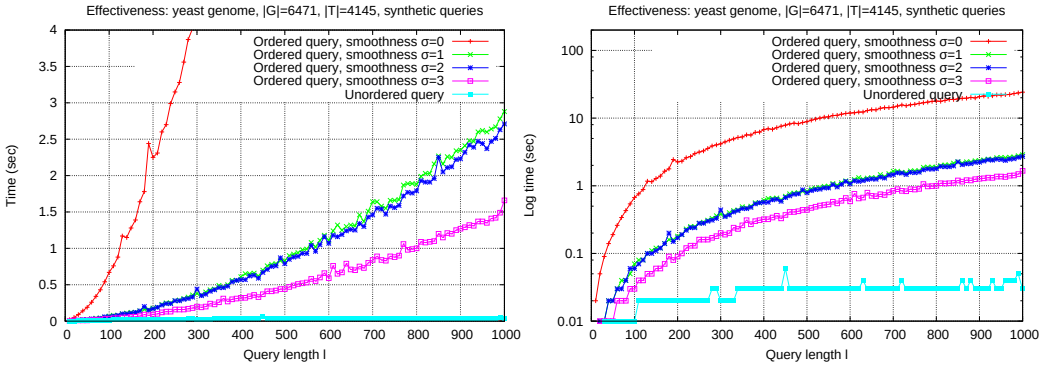

Figure 3.3: Effectiveness comparison of p-value approximations with different smoothness parameters, using synthetic queries of baker's yeast genome. Tests were performed on `bioinf.ebc.ee`, $2 \times 2.8GHz$ Intel Xeon, $3.8Gb$ RAM.

3.5 Significant subgraphs. Method $\mathbf{GOSTMINER_4}$

Previously we described algorithms for finding matching terms to a user-defined query of genes, and constructed methods for evaluating the significance of every resulting term. We defined thresholds and multiple testing corrections for recognising significant results.

So far we have observed both significant and insignificant terms of a user-defined query, and treated all results to be independent of each other. We have disregarded parent-child relations between the ontology terms that define Directed Acyclic Graph structure of vocabularies. However, discarded information may prove useful in interpreting resulting set of terms, and moreover, True Path Rule suggests that significant terms may be hierarchically related.

This section constructs a method for filtering results, grouping terms through underlying graph structure and determining significant subgraphs. Let $G_s = \{g_1, g_2, \ldots, g_l\}$ be a fixed user-defined ordered query genes, $G_{s,k}$ a subquery of G_s, and $T_s = \{t_1, t_2, \ldots, t_n\}$ be the set of GO/KEGG terms that match any subquery $G_{s,k}$. The following method naturally applies to the more general case of unordered queries G_q. A condensed version of the method is available as $\mathbf{GOSTMINER_4}$ (Algorithm 5).

Let $\omega(G_{s,k}, t)$ be significance filter function that recognises significant matching terms $t^+ \in T_s$ for a subquery $G_{s,k}$, and filters out others $t^- \in T_s$ that show too large a probability value. The probability $\mathbf{p}(G_{s,k}, t)$ is cumulative hypergeometric p-value from Fisher's one-tailed test. Here we apply the previously constructed analytical significance threshold γ, but other means as multiple testing corrections or user-defined thresholds may also be applied. The trivial case ω_1 compares p-values against 1.0 and passes all terms as significant.

$$\omega(G_{s,k}, t) \begin{cases} \mathbf{p}(G_{s,k}, t) \leqslant \gamma_k, & \textbf{true}, \text{ Term } t \text{ is significant} \\ \mathbf{p}(G_{s,k}, t) > \gamma_k, & \textbf{false}, \text{ Term } t \text{ is insignificant} \end{cases}$$

Let us create the set T_s^+, consisting of all terms $t^+ \in T_s$ that match and appear significant with the ordered query G_s. A term t is included in the set of results T_s^+, if there exists at least one subquery $G_{s,k}$ of any length $k \in \{1, \ldots, l\}$, so that the intersection $|G_{s,k} \cap G_t|$ ranks significant and passes the filter ω.

$$T_s^+ = \{t^+ | t^+ \in T_s \wedge \exists k \in \{1, \ldots, l\} : \omega(G_{s,k}, t^+) = \textbf{true}\}$$

We need to take advantage of GO/KEGG vocabularies, defined in the beginning of the chapter as the triplet $\mathcal{V} = (T, \Xi, R)$. We are interested in the collection of facts R. Every fact $r = (t, t', \xi) \in R$ states that a term t is a child of another term t', and the special relationship between t and t' is ξ. The set R essentially defines the DAG structure of GO.

Let us construct $R_s^+ \subseteq R$, a collection of facts that describes the set of terms T_s^+. For every matching term $t^+ \in T_s^+$, we observe corresponding facts $r^+ = (t^+, t', \xi)$ from the collection R, and construct the set of parents $t' \in T_t$ for the term t. If some parent $t' \in T_t$ also ranks significant according to the filter ω and is present in the result set $t' \in T_s^+$, we add the corresponding fact r^+ to the new collection R_s^+.

$$R_s^+ = \{r^+ = (t^+, t'^+, \xi) \in T_s^+ \times T_s^+ \times \Xi | r^+ \in R\}$$

If some term t^+ is present in the significant set T_s^+, but none of its parents $t' \in T_t$ passes as significant and included in the result set T_s^+, the term t^+ is considered a root term in the collection of facts R_s^+ and corresponding graph.

Algorithm 5 $\mathbf{GOSTMINER_4}$: Determine collection facts R_s^+ with significant terms to an ordered user query G_s.

Require: $G_s = \{g_1, g_2, \ldots, g_l\}$ {*Query of genes of a given genome*}
Require: $\mathcal{A}' = (G, T'_G, \mathcal{E}, A', \mathcal{G})$ {*Annotations of the given genome*}
Require: $\mathcal{V} = (T, \Xi, R)$ {*GO/KEGG vocabularies*}
Ensure: $\mathcal{R} = \{(p, t, |G_q|, |G_t|, |G_q \cap G_t|)\}$ {*Array of results*}
Ensure: $R_s^+ = \{(t, t', \xi)\}$ {*Array of facts with significant matching terms*}
  $\mathcal{R} := \mathbf{GOSTMINER_3}(G_s, \mathcal{A}, \sigma = 2)$ {*Get results from Algorithm 4*}
  $T_s^+ := [\,]$ {*Declare empty arrays*}
  $R_s^+ := [\,]$
  for all $(p, t, k, |G_t|, |G_{s,k} \cap G_t|) \in \mathcal{R}$ **do**
    if $\omega(G_{s,k}, t) = \mathbf{true}$ **then**
      $T_s^+[t] := \mathbf{true}$ {*Collect significant terms*}
    end if
  end for
  for all $(t \in \mathbf{keys}(T_s^+) : T_s^+[t] = \mathbf{true})$ **do**
    $T_t := \mathbf{parents}(t, R)$ {*Get all parents of* t *from facts in* R}
    for all $t' \in T_t$ **do**
      $r \Leftarrow (t, t', \xi) \in R$
      if $t' \in T_s^+$ **then**
        $r \rightarrow R_s^+$ {*Collect facts with significant terms*}
      end if
    end for
  end for
  return $R_s^+, \mathcal{R}$

The set of relations R_s^+ defines subgraphs that are part of the general hierarchical ontology structure. Every vertex of these subgraphs, a term t^+, has appeared as a significant match to some subquery of the ordered query G_q. Significant terms connected with edges, that is, bound with direct parent-child relations $r^+ = (t^+, t'^+, \xi)$ and united into related term families. True Path Rule populates intersecting genes to the topmost ancestor, however, significance filter ω cuts off parents that are too general, and child terms that are insufficiently represented in the query. There may be several significant and unconnected subgraphs of the same domain, as the common ancestor of the families does not pass significance filter.

In case of trivial filter ω_1, all paths towards root are considered significant and a query normally results in three subgraphs for the domains of Molecular Function, Biological Process and Cellular component. The 4^{th} graph for KEGG Pathways appears if any of the genes in query is annotated to KEGG.

Chapter 4

The tool GOSt: usage and features

Previous chapters constructed a mathematical model of Gene Ontology and corresponding gene annotations. We described methods for determining GO terms that match a group of genes, and provided means for assessing the significance of results.

The practical aim of our work is a usable tool that applies the above methods for mining Gene Ontology with gene groups from large-scale expression analysis experiments. This chapter gives a brief overview of the mining tool GOSt (Gene Ontology Statistics), and describes its features, such as graphical user interface, browsing of subgraphs, and expression analysis pipeline.

4.1 General usage

GOSt is made up of a set of modules written in Perl, and the mining algorithm family **GOSTMINER** is implemented as a C/C++ class. The former technology was chosen for its ease of manipulating various data structures, while the latter programming language has an advantage in computing performance. At the time, GOSt runs only on Linux platform.

The most basic form of mining Gene Ontology with GOSt is performed via command line interface. A simplest user query requires a genome id and a space-separated query of genes G_q of the chosen organism, and results in a list of terms $t \in T_q$, that match the query G_q. The query may also be loaded from a file. Query lines may contain comments, everything after symbol # will be disregarded by

```
1:          2:             3:         4:   5: 6:  7:    8:     9: 10:
GO:0006096  --?M-MM---M-   3.99e-08  21   12  4  0.333 0.190  BP glycolysis
KEGG:00620  pp?pp-------   2.69e-07  33   12  4  0.333 0.121  PW Pyruvate metabolism
GO:0031980  AA?---------  ~3.61e-02 165   12  2  0.167 0.012  CC mitochondrial lumen
GO:0008150  aA?MMMMOOOMO  ~1.00e+00 6471  12 11  0.917 0.002  BP biological_process
```

Figure 4.1: GOSt output example with a query of 12 yeast genes; the 3^{rd} gene is unknown. Columns from left to right: (1) id of term t; (2) mapping intersection $G_q \cap G_t$ onto query G_q, letters represent evidence codes, dashes - show no match; (3) p-value, tilde ~ shows insignificant values; (4) size of term set G_t; (5) length of query $l = |G_q|$; (6) size of intersection $G_q \cap G_t$; (7) precision; (8) recall; (9) Domain of t; (10) name of t.

the program. An alternative query format requires an identifier t of GO term or KEGG pathway. Then all the genes annotated to t will be included in the query, creating a possibility to compare different ontology terms and detect significant overlaps. Figure 4.1 displays a few lines of example output.

Resulting terms are ordered by p-value. The experiment-wide significance level is fixed at $\alpha = 0.05$. The user may define a thresholds filter ω for distinguishing significant results. Currently available options are shown in the list below. Multiple thresholds may be active at the same time and the strongest will be chosen for filtering terms. Is is also possible to show only matches that rank significant according to current threshold scheme in function ω.

- Analytical significance threshold γ (used by default);
- Bonferroni multiple testing corrected significance level α_B, $n = |T_q|$;
- Benjamini-Hochberg False Discovery Rate corrected level α_{BH};
- User-defined threshold (default value 1.0).

The query may be defined as an ordered set of genes G_s, where genes are sorted in descending order of importance. The output then shows peak significance p-value for every term, and the subquery length $k = |G_{s,k}|$ where the peak occurred. The term is considered significant, if peak p-value passes significance filter. Analytical significance threshold considers the stored value for the given subquery length k, while multiple testing corrected levels α_B and α_{BH} evaluate all probabilities that emerged in ordered query analysis.

User may choose to sort resulting terms according to hierarchical structure instead of p-value. Significant subgraphs procedure described in the previous chapter is then activated, and subgraphs are evaluated with current significance filter ω. Terms are then grouped as subgraphs and printed out in depth-first order, beginning from the topmost term in the hierarchy.

Command line interface of GOSt is intended for large-scale semiautomatic analysis. Compact mode of GOSt skips additional information such as mapping intersection to query and GO names, and is therefore considerably faster and suitable for scripting. There are also options for exporting results as Perl storable data structures and image files.

4.2 Graphical user interface. Evidence codes

Graphical user interface of GOSt is designed for interactive data analysis, and implemented as a dynamical web environment with PHP: Hypertext Preprocessor [PHP]. Most features of command line GOSt may be combined; an additional set of functionality enables enhanced interactive mining.

Web interface of GOSt uses PNG (Portable Network Graphics) image format for results output. An example output is shown in Figure 4.2. Extensive visualisation in graphical GOSt is enabled by the Simple Web Object Graphics language SWOG [Han05].

Gene Ontology Consortium web page [GO] suggests a partial loose ordering of evidence codes to assess the quality of results annotated with specific codes. We have followed the suggestion and created a colour legend that reflects evidence code quality by ranking better evidence codes with warmer colours, and lower-quality codes with colder colours. Such a legend gives a bird's eye view, and allows the user to decide upon the common evidence behind her results. The legend is available in Table 4.1.

KEGG Pathways are not part of the ordering, and brought out as black to be clearly distinguished from GO terms. It should be noted that evidence code `ND` refers to the fact that the gene in question is annotated to GO, but nothing is known about it. The code `NA`, on the other hand, is part of GOSt interface rather than GO evidence codes, and denotes that the gene in question is not known in GO/KEGG annotations at all.

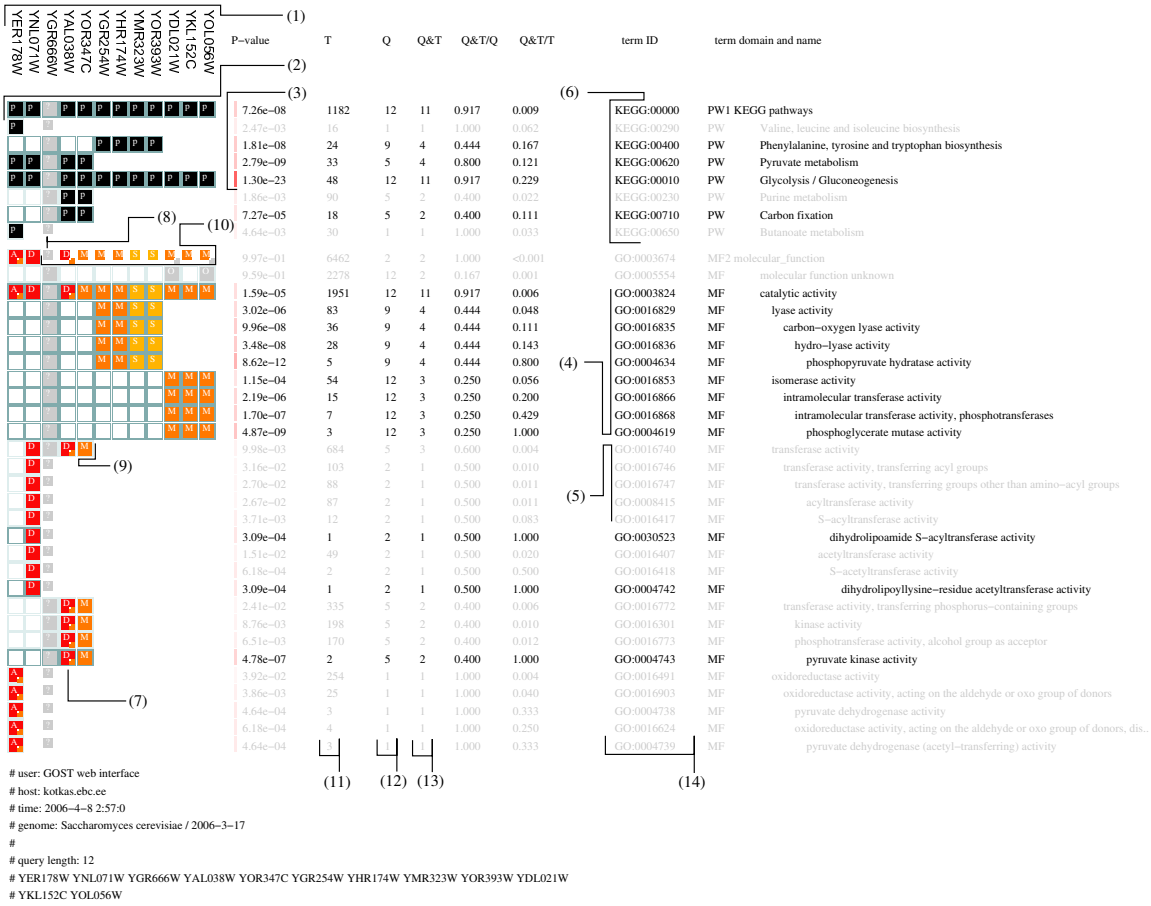

Figure 4.2: Example of GOSt graphical output for an ordered query G_s of 12 yeast genes. Numerical columns are mostly ordered as shown in Figure 4.1. (1) Genes present in query; (2) mapping intersection $G_s \cap G_t$ onto query G_s, letters and different colours represent evidence codes, white boxes with gray edges show no match; (3) p-value heatmap for visual aid in locating very good values; (4) significant results drawn in black; (5) insignificant results drawn in gray; (6) results sorted by hierarchical structure into subgraphs; (7) multiple evidence codes; (8) unknown gene in query; (9) matching gene at significance peak; (10) matching genes after significance peak; (11) hyperlink to query with genes in G_t; (12) hyperlink to peak significance query with genes in $G_{s,k}$; (13) hyperlink to peak significance query with genes in $G_{s,k} \cap G_t$; (14) hyperlink to external GO/KEGG data sources.

	C	IC	Inferred by curator
	A	TAS	Traceable author statement
	D	IDA	Inferred from direct assay
	M	IMP	Inferred from mutant phenotype
	G	IGI	Inferred from genetic interaction
	P	IPI	Inferred from physical interaction
	S	ISS	Inferred from sequence or structural similarity
	X	IEP	Inferred from expression pattern
	a	NAS	Non-traceable author statement
	R	RCA	Inferred from reviewed computational analysis
	E	IEA	Inferred from electronic annotation
	o	NR/ND	Not recorded, No biological data available
	p	KEGG	Mapped to KEGG pathway
	?	NA	Not annotated

Table 4.1: Colour-coded evidence code legend used in GOSt.

4.3 Visualisation of matching terms

GOSt graphical web interface has the possibility to view subgraphs of GO terms that are represented in given user query. Terms with parent-child relations and long descriptive names are drawn out as graphs (Figure 4.3). Graph drawing is based on the Graphviz software [GV] and included in SWOG interface [Han05].

Resulting terms are grouped as connected graphs, meaning that all terms in a group share one or more common ancestors and are connected with paths. If significance filter is active, all the terms need to be above threshold as well. The user may also choose a term t in output to see the subgraph formed by the (significant) descendants of t.

The user may choose to view all matching terms, or only those that ranked significant according to the threshold filter ω. In the first case, the query will nor-

mally result in a large subgraph for every domain of Biological Process, Molecular Function and Cellular Component (and sometimes KEGG Pathway), as of the True Path Rule, all paths toward root are always true for every term. In the second case, resulting subgraphs may consist of several closely related and unconnected fragments with only a few terms, as common parents for these fragments have been ruled out by significance filter.

4.4 Ordered queries analysis

Fast analysis of ordered queries is one of the most important properties of GOSt. The program's graphical user interface offers some additional features for visualising the peak significance and performing new interactive queries based on previous results.

For every term matching an ordered query, the peak significance length an p-value is displayed. Genes appearing after the peak are visualised in a different manner, as shown of Figure 4.2, points $(9) - (10)$. We believe that investigating unannotated genes around significance peak may aid in estimating putative functional annotations of such genes.

The user may choose to plot out p-value approximations over ordered query length for single terms, groups of terms based on connected DAG fragments, or subgraphs formed by descendants of a single term. If a significance threshold filter is active, only p-values below threshold are shown. Figure 4.4 shows p-value approximation for Molecular Function terms resulting the above example query.

4.5 Expression data analysis pipeline

GOSt includes a set of loosely bound modules that form a pipeline for high-throughput or interactive analysis of large sets of gene groups, such as results from expression data analysis. We have developed means for sorting, filtering and combining results from different methods, enabled parallel calculations on computing clusters and grids.

First stage in the pipeline would involve query generation. *Gost Sorted List*

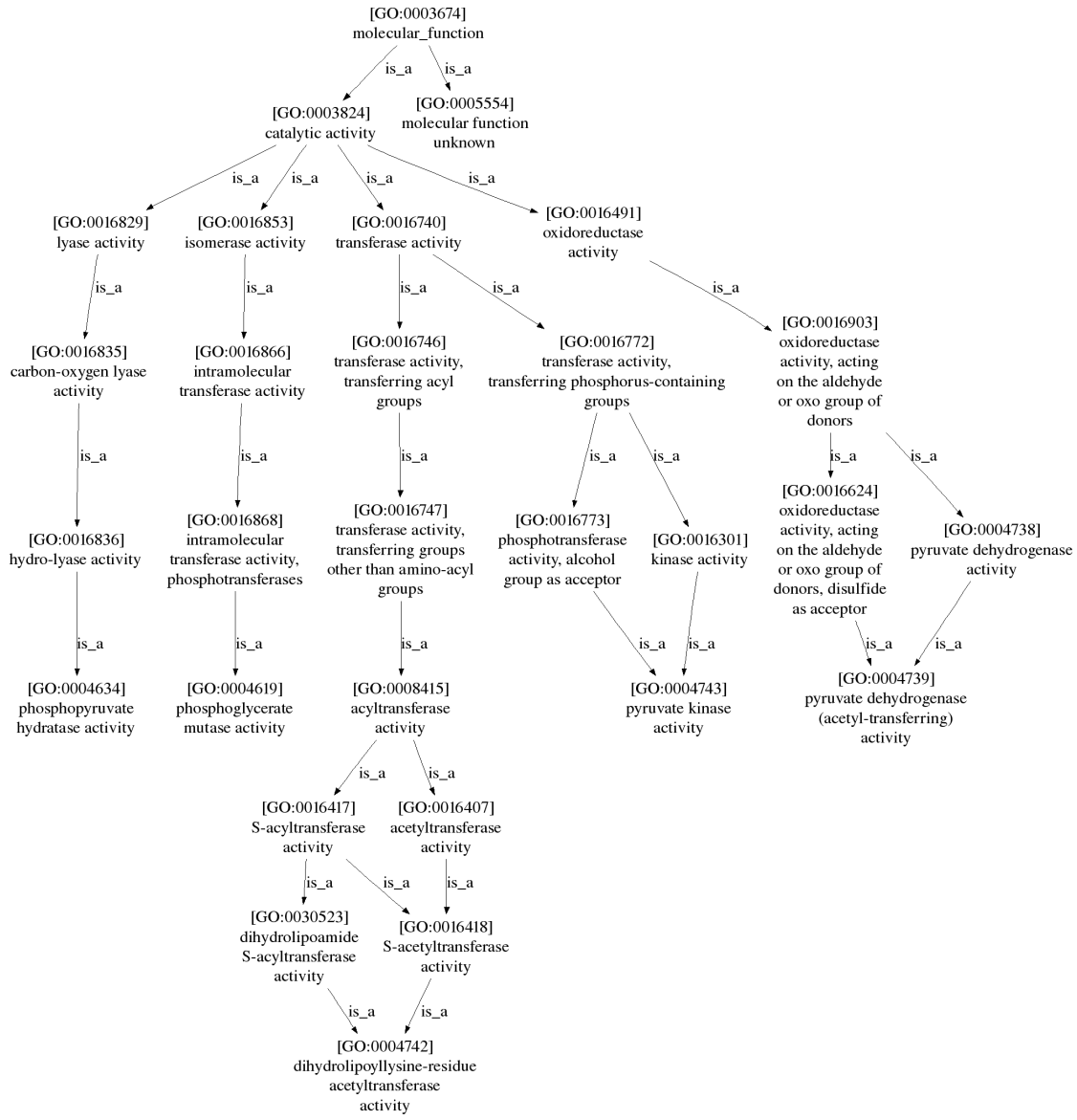

Figure 4.3: Example of GOSt graphical output for an ordered query G_s of 12 yeast genes. The graph represents terms of the Molecular Process domain that matched the query in Figure 4.2. Significance filter was not applied, and therefore, all matching terms are shown.

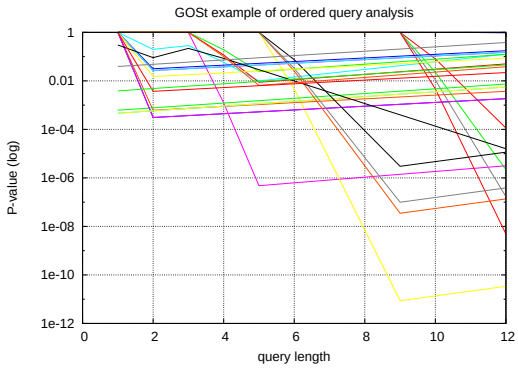

Figure 4.4: Example of GOSt graphical output for an ordered query G_s of 12 yeast genes. The plot shows approximation for p-values of Molecular Process terms over the ordered query in Figure 4.2. Significance filter is not applied, and therefore, all matching terms and p-values are shown.

Browser is a tool for finding similar genes to a given gene g_1. The program requires an expression data matrix in textual format, and returns a user-defined number of genes most similar to gene g_1. Multiple datasets of different organisms may be added to Sorted List Browser. The user needs to specify a distance measure for determining the similarity of genes. Several distance measures are available, for example *Euclidean distance* and *correlation distance*. Sorting is based on a wrapper for the program *Distances From*, implemented as part of fast clustering algorithms in [Kul04]. The user may also interactively search for genes in related local datasets, using keywords, names, partial matches and regular expressions. Resulting gene identifiers may be then directed to GOSt for Gene Ontology analysis. *GOSt Expression Plotter* may be activated for viewing the expression profile plot of the genes in the query. Another output leads to URLMAP, a tool designed to serve as a knowledge hub for linking external datasources [Vil02].

A possible expression data analysis involves creating a number of sorted lists, every list beginning with an *a priori* interesting gene. We retrieve all significant matches from GOSt for these ordered queries. We then filter the results by stating some interesting thresholds for precision, recall, or p-value. We observe expression profiles that occur within lists, and hierarchically cluster the lists according to expression profiles. Lists may also be grouped according to significant terms,

or represented subgraphs of significant terms. Finally we group and count significantly represented subgraphs to find Gene Ontology terms that characterise best the expression data set.

Most of the time-consuming calculations described above are already experimentally implemented in parallel environments, such as local computer clusters, and larger GRID-based computing farms such as BalticGrid [BG]. The task of finding similar genes or significant terms is rather simple in its most basic form. Every gene or sorted list may be considered a separate process, that does not require interaction with other processes. In this case, the key for parallelisation is the distribution of input parameters, that is, genes, to different processes, and the aggregation of results from remote locations. In the case of local clusters, this can be easily achieved with Message Passing Interface (MPI) technology [MPI]. Similar methods and interfaces are available for GRID-based computing.

Summary

Large-scale genetic studies such as microarray experiments result in numerous groups of genes that have shown biological evidence to be similar in some sense. It is often useful to know what is commonly known about the genes that have shown similar expression.

This Master's Thesis studies Gene Ontology (GO) hierarchical vocabularies of molecular function, biological process, and cellular component. GO terms are widely used for annotating genes and gene products of various genomes. We study the Directed Acyclic Graph structure of GO, and propose a model for integrating biological pathway data from the KEGG database. The integration scheme is applicable to other kinds of knowledge; future developments involve integration of different pathway databases as well as protein-protein interaction data.

The goal of GO and KEGG data analysis involves determining common annotations to a user-defined group of genes. Due to hierarchical nature of GO, any query will result in a number of matching terms, and statistical significance of results needs to be observed for filtering out insignificant matches. We investigate statistical methods for determining significance of results, and standard multiple testing corrections used in current Gene Ontology tools. Standard methods are argued to be unsuitable for Gene Ontology analysis for partly dependent terms. We investigate the behaviour of multiple testing corrections in simulation experiments, and based on observed p-values, propose a new method for distinguishing significant results.

Our work introduces a novel concept of analysing ordered gene queries. The idea may prove useful in mining microarray data. For example, one may be interested in a particular gene and a list of its closest similar neighbours, and determine the portion of genes from the head of the list, where a certain GO term or KEGG

pathway reaches peak of significance. We propose a fast algorithm that observes intersection points, and estimates probabilities across a given ordered query and GO term.

Using significance thresholds and ideas for ordered sets, we propose a method for mining significant groups of related GO terms. For a given ordered user query, we determine all terms that have matched some portion of the list with high statistical significance. These terms are grouped together as subgraphs of the general GO structure. The significance filter effectively cuts off parent terms that are too general to be interesting, and child terms that are too specific to represent the query.

The practical result of this Master Thesis is GOSt, a Gene Ontology mining tool. GOSt may be used as a command line tool for large-scale computational pipeline for microarray data mining. The graphical web interface of GOSt enables highly interactive analysis of gene sets. We have put great effort in visualisation to make complex results understandable and interpretable to users.

Geeniontoloogiate kaevandamise programm GOSt

Magistritöö (40 AP)

Jüri Reimand

Kokkuvõte

Tehnoloogilised arengud on muutmas molekulaarbioloogia traditsioonilisi meetodeid. Töö keskmes ei pruugi olla vaid üksik geen. Tänu tõhusale sekveneerimisele on saadaval aina enamate organismide genoomid. Mikrokiipidega teostatakse mahukaid katseid ning mõõdetakse üheaegselt paljude geenide ekspressiooni. Tulemuseks on arvukad grupid geenidest, mis on katses sarnaselt avaldunud. Edasiseks uurimiseks on kasulik vaadelda sarnaste geenide teadaolevaid omadusi.

Geeniontoloogiad (GO) on esitatavad graafidena ning koosnevad valdkonna hierarhiliselt struktureeritud sõnastikest. GO terminitega saab kirjeldada geeni ja geeniprodukti funktsiooni, protsessi ning asukohta rakus. GO on kasutusel paljude organismide geenide annoteerimiseks. Käesoleva magistritöö esimene peatükk annab ülevaate töö bioloogilisest taustast, uurib GO struktuuri ning pakub välja skeemi KEGG bioloogiliste radade andmebaasi integreerimiseks GO andmekogusse. Skeemi saab tulevikus rakendada alternatiivsete andmekogude, näiteks valk-valk interaktsioonide lisamiseks.

Geeniontoloogiate ja bioloogiliste radade analüüsis on oluliseks ülesandeks kindlale geenihulgale teadaolevate annotatsioonide leidmine. Ontoloogiate hierarhilise struktuuri tõttu vastab igale kasutaja geenipäringule suur hulk erineva

spetsiifilisusega termineid. Seetõttu tuleb hinnata iga tulemuse statistilist olulisust ning jätta välja terminid, mis ei ole päringus piisavalt esindatud.

Käesoleva töö teises peatükis uuritakse võimalikke meetodeid GO tulemuste olulisuse hindamiseks. Samuti vaadeldakse standardseid mitmese testimise parandusi, mis on kasutusel levinud GO töövahendites oluliste tulemuste filtreerimisel. Mitmete autorite arvates ei ole standardsed parandused GO analüüsiks sobivad, kuna terminite vahel kehtivad hierarhilised seosed ning tegemist ei ole statistiliselt sõltumatu testimisega. Magistritöös on teostatud simulatsioon, milles vaadeldakse sünteetiliste päringute statistilist olulisust ning mitmese testimise parandusi. Simulatsiooni tulemusena pakutakse välja uudsed läved oluliste tulemuste eristamiseks, mis võtavad arvesse ontoloogiate struktuuri ning on vastavuses mitmese testimise põhimõtetega.

Magistritöö kolmandas peatükis vaadeldakse GO kaevandamiseks sobivaid kiireid algoritme ning pakutakse välja uudne idee järjestatud geenipäringute analüüsiks. Näiteks võib uurijale huvi pakkuda mõni konkreetne geen ning hulk sellele sarnaselt avaldunud geene. Geenide hulga võib sorteerida avaldumise sarnasuse järgi ning uurida, kui palju esimese geeni lähimaid naabreid on oluliselt esindatud mõne GO terminina või KEGG bioloogilisel rajal. Järjestatud loetelude analüüsiks on töös toodud kiire algoritm, mis vaatleb GO termini ja geenide loetelu lõikepunkte ning hindab olulisuse muutumist erinevatel loetelu pikkustel.

Eelnevas kirjeldatud olulisuse lävesid ning järjestatud loetelude analüüsi on töös kombineeritud oluliste terminite gruppide leidmiseks. Meetodis kombineeritakse hierarhiliste sidemetega gruppidesse need terminid, mis on statistiliselt olulised mõnel loetelu pikkusel. Tulemusena saadakse hulk GO alamgraafe, milles on esindatud antud loetelu kontekstis olulised terminid. Olulisusläve abil jäetakse graafidest välja hierarhiliselt liiga üldised ja liiga spetsiifilised terminid.

Magistritöö praktiliseks väljundiks on geeniontoloogiate kaevandamise programm GOSt, millele on pühendatud töö neljas peatükk. Peatükis on toodud töövahendi võimalused, juhised kasutuseks, näited väljundist ja lühiülevaade mikrokiibi andmeanalüüsi töökavast. Mitmetahuliste tulemuste uurimisel on abiks programmi graafiline kasutajaliides. Visualiseerimisel on töövahendis GOSt tähtis roll ning seda võib pidada programmi üheks suuremaks tugevuseks.

Acknowledgements

I am most grateful to my supervisor Dr. Jaak Vilo, for bringing me to bioinformatics, for numerous educating conversations and seminars, motivating ideas, and guidance over the past two years, that have aided in the development and outcome of this project.

Jaak's research group BIIT has formed a perfect interdisciplinary environment for studying Bioinformatics, and I am thankful to all current and former members for providing interesting seminars and discussions, pointing to literature, and aiding in technical questions.

Hedi Peterson has been my guide to the biological side of bioinformatics, and an excellent companion during the long nights of study and research. I am especially grateful to Meelis Kull for aid on the algorithmic and statistical side, as well as proofreading several drafts of the paper. Jaanus Hansen's SWOG project of data visualisation has enhanced the look and feel of GOSt. Priit Adler, Raivo Kolde, Prof. Per Kraulis, Asko Tiidumaa, and Konstantin Tretjakov have provided useful background knowledge, and as first users of GOSt, given valuable tips for improving usability of the tool. Sten Ilmjärv, Priit Palta, and Pavlos Pavlidis have been great labmates, and during the time, and shared lots of interesting biological details with me.

Last, but not least, I would like to thank my girlfriend Mari-Liis, my family, and my friends of the external world. The closest people to me have been very supportive during this exhaustive period.

This Master's Thesis has been partially supported by ETF grants (ETF-5722 and ETF-5724), as well as ATD project (LSHG-CT-2003-503329), and FunGenES project (LSHG-CT-2003-503494).

Bibliography

[ABB+00] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[ABB+01] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Creating the Gene Ontology Resource: Design and Implementation. *Genome Research*, 11/8:1425–1433, 2001.

[ABB+04] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research, Database issue*, 32:D258–D261, 2004.

[AFF] Affymetrix.
www.affymetrix.com.

[ASDUD04] Fátima Al-Shahrour, Ramón Díaz-Uriarte, and Joaquín Dopazo. FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. *Bioinformatics*, 20(4):578-580, 2004.

[BCD+04] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich1, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L. Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The Pfam protein families database. *Nucleic Acids Research, Database issue*, 32:D138–D141, 2004.

[BG] BalticGrid Project.
www.balticgrid.org.

[BH95] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society B*, 57(1):289-300, 1995.

[BKB+03] Gabriel F. Berriz, Oliver D. King, Barbara Bryant, Chris Sander, and Frederick P. Roth. Characterizing gene sets with FuncAssociate. *Bioinformatics*, 19(18):2502-2504, 2003.

[Bon36] C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

[BPSS01] Alvis Brazma, Helen Parkinson, Thomas Schlitt, and Mohammadreza Shojatalab. A quick introduction to elements of biology - cells, molecules, genes, functional genomics, microarrays.
www.ebi.ac.uk/microarray/biology_intro.html, 2001. Draft.

[FLY] A Database of the Drosophila Genome.
flybase.bio.indiana.edu.

[GDS03] Yongchao Ge, Sandrine Dudoit, and Terence P. Speed. Resampling-based multiple testing for microarray data analysis. Technical Report 633, Dept. of Statistics, UC Berkeley, 2003.

[GDSH+03] Jr Glynn Dennis, Brad T. Sherman, Douglas A. Hosack, Jun Yang, Wei Gao, H. Clifford Lane, and Richard A Lempicki. DAVID: Database for Annotation, Visualization, and Integrated Discovery. *Genome Biology*, 4 (9), 2003.

[GN87] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Kaufmann, Los Altos, CA, 1987.

[GO] The Gene Ontology Consortsium. `www.geneontology.org`.

[Gru93] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[GV] Graphviz - Graph Visualization Software. `www.graphviz.org`.

[Han05] Jaanus Hansen. Graphics language SWOG. Bachelor Thesis (In Estonian), 2005. University of Tartu, Estonia.

[HSA+02] Ada Hamosh, Alan F. Scott, Joanna Amberger, Carol Bocchini, David Valle, and Victor A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research, Database issue*, 30:52–55, 2002.

[Kar01] Peter D. Karp. Pathway Databases: A Case Study in Computational Symbolic Theories. *Science*, 293:2040 – 2044, 2001.

[KEG] Kyoto Encyclopedia of Genes and Genomes. `www.genome.jp/kegg`.

[KGH+06] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F. Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. From genomics to chemical genomics: new developments in KEGG. *Nucleic Acids Research, Database Issue*, 34:D354 – D357, 2006.

[Kul04] Meelis Kull. Fast Clustering in Metric Spaces. Master's thesis, University of Tartu, Estonia, 2004.

[MGI] Mouse Genome Informatics. `www.informatics.jax.org`.

[MHK05] Steven Maere, Karel Heymans, and Martin Kuiper. BiNGO: a Cytoscape plugin to assess overrepresentation of Gene Ontology categories in Biological Networks. *Bioinformatics*, 21(16):3448–3448, 2005.

[MMP04] Marco Masseroli, Dario Martucci, and Francesco Pinciroli. GFINDer: Genome Function INtegrated Discoverer through dynamic annotation, statistical analysis, and mining. *Nucleic Acids Research, Web Server Issue*, 32:W293–W300, 2004.

[MP05] Marco Masseroli and Francesco Pinciroli. Using Gene Ontology and genomic controlled vocabularies to analyze high-throughput gene lists: Three tool comparison. *Computers in Biology and Medicine*, 2005.

[MPI] MPI - Message Passing Interface. `www-unix.mcs.anl.gov/mpi`.

[NHG] National Human Genome Research Institute. `www.genome.gov`.

[OGS+99] Hiroyuki Ogata, Susumu Goto, Kazushige Sato, Wataru Fujibuchi, Hidemasa Bono, and Minoru Kanehisa. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 27/1:29 – 34, 1999.

[OZC04] Michael V. Osier, Hongyu Zhao, and Kei-Hoi Cheung. Handling multiple testing while interpreting microarrays with the Gene Ontology Database. *Bioinformatics*, 5(124), 2004.

[PHP] PHP: Hypertext Preprocessor. `www.php.net`.

[Sch04] Carl F. Schaefer. Pathway Databases. *Annals of the New York Academy of Sciences*, 1020:77 – 91, 2004.

[SGD] Saccharomyces Genome Database. `www.yeastgenome.org`.

[Slo02] Donna K. Slonim. From patterns to pathways: gene expression data analysis comes of age. *Nature Genetics*, 32:502-508, 2002.

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

[ZFW+03] Barry R Zeeberg, Weimin Feng, Geoffrey Wang, May D Wang, Anthony T Fojo, Margot Sunshine, Sudarshan Narasimhan, David W Kane, William C Reinhold, Samir Lababidi, Kimberly J Bussey, Joseph Riss, J Carl Barrett, and John N Weinstein. GoMiner: a resource for biological interpretation of genomic and proteomic data. *Genome Biology*, 4(R:28), 2003.

[ZMC+04] Wen Zhang, Quaid D Morris, Richard Chang, Ofer Shai, Malina A Bakowski, Nicholas Mitsakakis, Naveed Mohammad, Mark D Robinson, Ralph Zirngibl, Eszter Somogyi, Nancy Laurin, Eftekhar Eftekharpour, Eric Sat, Jörg Grigull, Qun Pan, Wen-Tao Peng, Nevan Krogan, Jack Greenblatt, Michael Fehlings, Derek van der Kooy, Jane Aubin, Benoit G Bruneau, Janet Rossant, Benjamin J Blencowe, Brendan J Frey, and Timothy R Hughes. The functional landscape of mouse gene expression. *Journal of Biology*, 3(5)(21), 2004.