GOMIT: A Generic and Adaptive Annotation Algorithm Based on Gene Ontology Term Distributions

In-Yee Lee[1,2], Jan-Ming Ho[2], Ming-Syan Chen [1]
[1] *Department of Electrical Engineering, National Taiwan University, Taiwan*
[2] *Institute of Information Science, Academia Sinica, Taiwan*
E-mail: iylee@iis.sinica.edu.tw

Abstract

We address the issue of providing highly informative annotations using information revealed by the structured vocabularies of Gene Ontology (GO). For a target, a set of candidate terms used to infer the target's property is collected and forms a unique distribution on the GO directed acyclic graph (DAG). We propose a generic and adaptive algorithm—GOMIT, which bases on term distributions and GO hierarchical characteristics to assign correct annotations for a target. We establish a quantitative model with parameters that can be trained for optimal performance for different applications. We propose several criteria for evaluating GOMIT's performance, and conducted three experiments involving a) automated functional annotations, b) biological annotations of microarray data clusters and c) protein family GO assignments. In these experiments, we used our proposed criteria to compare GOMIT with other algorithms. Results not only reflect GOMIT's generality and adaptability, but also suggest that GOMIT is better or comparable to other works for assigning correct annotations.

1. Introduction

High-throughput genomic sequencing methods have been used to create large repositories of public domain data. To improve knowledge sharing efficiency, large databases are being populated with data that gain value only after annotation tasks are performed. Uncharacterized sequence annotation entails either human effort or automated methodology [6]. Both human and machine-generated sequence annotations are difficult to manage electronically. Many designers and researchers are therefore promoting the idea of using ontology-based annotations, since ontologies provide controlled or structural vocabularies that allow for standardized annotation mechanisms. The Gene Ontology Consortium [14] maintains a Gene Ontology (GO) consisting of sets of domain-specific vocabularies for describing molecular characteristics across several organisms. The three ontologies of the GO term hierarchy are molecular functions, biological processes, and cellular components. GO terms and their associated "is-a" and "part-of" relations form directed acyclic graphs (DAGs) in which a parent node describes functions exhibited by its child nodes. Terms that are lower in height (i.e., closer to the root) describe more general functions; the greater the height, the more specific the function.

The purpose of this research can be explained in terms of a well-known application. In an automated functional annotation of an uncharacterized sequence $\bar{s}$, one can use homologue identification (e.g., sequence similarity) algorithms to derive a set $H_s = \{\bar{h} \mid \bar{h} \text{ is a homologue of } \bar{s}\}$. Each homologue $\bar{h} \in H_s$ is associated with a set $Term(\bar{h})$ of GO terms that describe the functional properties of $\bar{h}$. Let $T_s = \bigcup\{Term(\bar{h}) \mid \bar{h} \in H_s\}$. Based on GO characteristics, not only the terms in T_s but also the terms inferred from T_s are meaningful for annotating $\bar{s}$. Therefore, the structured GO graph is ideal for identifying more informative annotations.

On the GO graph, T_s and its ascendant terms form a unique DAG distribution. The distribution can be analyzed to determine a set M_s of GO terms that best represent the properties of $\bar{s}$. We refer M_s as the *most informative terms*. This eliminates the need to present an entire set T_s to biologists; presenting an entire set increases the potential for confusion and makes it harder for biologists to determine which GO term within T_s is the most significant, especially when $|T_s|$ is large. Using T_s's distribution on the GO DAG, the GOMIT algorithm identifies a set M_s of terms from T_s

Proceedings of the 5th IEEE Symposium on Bioinformatics and Bioengineering (BIBE'05)
0-7695-2476-1/05 $20.00 © 2005 IEEE

that best describes $\overline{s}$'s functions. The application touches on an important issue: given a gene set *G* of interest, how does one go about determining the GO nodes that best represent *G* [7]? GOMIT solves this problem by considering the distribution of *G* on the GO DAG.

On the GO DAG, a parent node describes a property exhibited by all of its child nodes. Therefore, evaluating how well a candidate GO term describes a target (e.g., an unannotated sequence or a microarray cluster) requires two criteria: a) *coverage*, reflecting whether a term covers all aspects of a target's properties; and b) *precision*, reflecting a term's specificity [7]. We propose using a quantitative model to assign to each candidate term a confidence value that reflects the term's significance. To make GOMIT adaptive, we propose a training method to derive the set of parameters that allows optimal performance for a particular application. This approach also has the advantage of being adaptive and capable of being trained to suit different applications.

We make the following contributions in this paper. First, we propose a generic and adaptive algorithm GOMIT for extracting a set of most informative terms from a set of candidate GO terms. Second, we show how GOMIT can be trained to yield optimal results for a particular application. Third, we propose several criteria for evaluating GOMIT and for comparing it with other algorithms. Finally, we cite related works and discuss GOMIT's advantages and disadvantages by comparing its performance with other algorithms under three experiments.

2. Related Works

Many researchers have adopted statistical approaches to retrieve significant GO categories from large GO term sets. Examples include Zhang and Schmoyer's *GOTM* [19], Beissbarth and Speed's *GOstat* [3], Al-Shahrour et al.'s *FatiGO* [1], and Zeeberg et al.'s *GOMiner* [18]. When used to assist in the functional profiling of GO-based gene expressions, these mechanisms fulfill the important objective of identifying GO categories that are over- (or under-) represented in a gene set with respect to a reference group.

All of them are statistical methods that view GO as a list of categories, without considering structural relationships among those categories [7]. While effective for visualization purposes, as statistical methods they fail to fully quantify GO hierarchies represented by graphs. In other words, a tree or DAG is primarily used for visualization and not for essential computation [10]. Their primary purpose is to visualize overrepresented terms with respect to a certain reference set rather than to generate automated GO assignments. On the other hand, GOMIT considers hierarchical GO properties when identifying significant terms from a set of candidate GO terms, and can be used with automated annotations [9], obtaining the biological meanings of microarray clusters, protein family GO assignments, and other similar applications. Lee et al. [10] created a system that uses a graph theory-based algorithm to identify the representative biological meanings of gene clusters. Taking advantage of the modified structure of a GO DAG, the algorithm extracts common biological attributes of genes within a cluster or a group of interest. Compared to their wok, GOMIT also uses GO hierarchical properties, but without the GO DAG reconstruction requirement.

All of the approaches described so far address assistance of microarray functional profiling instead of automated functional annotations. Similar to GOMIT, Khan et al. [8] and Schug et al. [11] interpret GO DAGs to assist automated functional annotations. Khan et al.'s *GOFigure*, a web annotation tool for predicting GO terms, accepts input sequences and uses BLAST [2] to identify homologues in GO annotated databases. In Section 5 we will describe our comparison of their work with GOMIT. Schug et al. [11] added a heuristic algorithm to generate rules for function-domain associations based on the intersections of GO functions assigned to gene products (GOAPs) that contain ProDom and/or CDD domains. Compared to their heuristic algorithm, GOMIT adopted a more general model to analyze term distributions. Joslyn et al.'s *Gene Ontology Categorizer* [7] considers GO in a manner that is similar to our previous efforts [9] and to Lee et al.'s [10] solution—that is, as a finite and partially ordered set, with a separate scoring function set that determines a node's weighted rank. The specificity and coverage of Joslyn et al.'s method depends on input parameters, with users responsible for supplying and fine-tuning parameters for optimal performance. In contrast, the fully automated GOMIT uses trainable parameters to create optimal balances between specificity and coverage for individual applications. Finally, GOMIT exhibits two significant advantages—generality and adaptability—over our previous work [9], Lee et al. [10], GoFigure [8], Schug et al. [11], and Gene Ontology Categorizer [7]. We offer evidence in support of this assertion in Section 4.

3. Description of the Algorithm

3.1. Algorithm Definition

We consider the GO hierarchy as a rooted tree; that is, $DAG_{GO} = \{GO, <_T\}$ where a) GO is the set of GO terms, b) $<_T$ is a partial ordering of GO, and c) for any term $\hat{t} \in GO$, $\{\hat{d} \in GO, \hat{d} <_T \hat{t}\}$ is well-ordered. To describe the GOMIT algorithm, we first define some terminology and notations. For two GO nodes (terms) $\hat{t}_x$ and $\hat{t}_y$, we say that $\hat{t}_x$ is a descendant of $\hat{t}_y$ (and $\hat{t}_y$ is an ancestor of $\hat{t}_x$) if $\hat{t}_y$ is on the unique path from root to $\hat{t}_x$. We say that $\hat{t}_x$ is a child of $\hat{t}_y$ (and $\hat{t}_y$ is a parent of $\hat{t}_x$) if $\hat{t}_y$ is an ancestor that immediately precedes $\hat{t}_x$. We denote by $\hat{t}_x \prec \hat{t}_y$ if $\hat{t}_x$ is a descendant of $\hat{t}_y$, and $\hat{t}_x \succ \hat{t}_y$ if $\hat{t}_x$ is an ancestor of $\hat{t}_y$. Finally, for a GO term $\hat{t}$ and a sequence $\bar{h}$, we say that $\hat{t}$ *annotates* $\bar{h}$ (with respect to the reference database) if $\hat{t}$ is associated to $\bar{h}$ in some existing reference database (e.g., SwissProt).

GOMIT is a generic and adaptive algorithm to extract the most informative GO terms from a set of GO annotations that possibly describe a *target*. For a target $\bar{x}$, let H_x denote the set of sequences from which $\bar{x}$'s GO terms should be inferred, M_x denote the set of terms selected by GOMIT as best describing $\bar{x}$ and O_x denote the set of $\bar{x}$'s original annotations. The GOMIT annotation process is illustrated in Figure 1. To annotate a target $\bar{x}$, we first derive from H_x a) a set T_x of GO terms possibly describing $\bar{x}$ (referred to as *initial candidate terms*), and b) the *initial confidence values* of terms in T_x (see Equation 1 below).

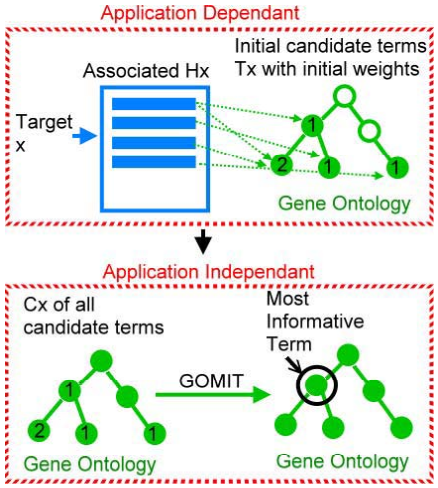

Figure 1. The GOMIT annotation process.

GOMIT then takes T_x as input and expands it into a set C_x of *derived candidate terms* (see Definition 1 below) with the additional terms having initial confidence values of zero. Finally, GOMIT derives M_x by selecting the most informative terms from C_x. In our experiments, we evaluate M_x's quality by comparing it with O_x.

Methods to derive T_x vary depending on the target's type. One example of a target is an uncharacterized sequence $\bar{s}$. To derive T_s, we first use a homologue identification algorithm together with reference databases DB to derive a set of $\bar{s}$'s homologues H_s=$\{\bar{h} \mid \bar{h} \in$ DB and $\bar{h}$ is a homologue of $\bar{s}\}$. Since $\forall \bar{h} \in H_s, \{\hat{t} \mid \hat{t}$ annotates $\bar{h}\} \neq \varnothing$, we can derive $T_s = \bigcup_{\forall \bar{h} \in H_s} \{\hat{t} \mid \hat{t}$ annotates $\bar{h}\}$.

Consider the following scenario (illustrated in Figure 2-a): assume that a) the target is an uncharacterized sequence $\bar{s}$, b) using homologue discovery, we identified two homologues $\bar{s}_1$ and $\bar{s}_2$, c) in the reference database used for homologue discovery, GO terms $\hat{t}_1$ and $\hat{t}_2$ annotates $\bar{s}_1$ and $\bar{s}_2$ respectively, and d) on the GO DAG, $\hat{t}_1$ and $\hat{t}_2$ have an identical parent $\hat{t}_3$. Because of GO hierarchy's characteristics, a parent term will always include the properties of all child terms. In this scenario, the parent $\hat{t}_3$ includes the properties of both $\hat{t}_1$ and $\hat{t}_2$. Therefore, even though term $\hat{t}_3$ is not associated with any matching homologue, it serves as a strongly informative description of $\bar{s}$. From this observation, we define Definition 1 as:

Definition 1: For a target $\bar{x}$, the set C_x of candidate terms includes all T_x terms and their ancestor terms.

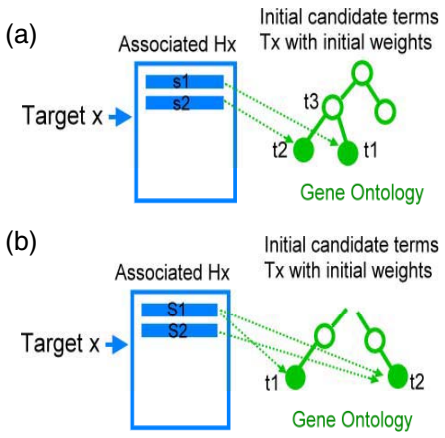

Figure 2. (a) A simple annotation scenario. **(b)** A more complicated annotation scenario.

To consider a more complex scenario (illustrated in Figure 2-b), assume for the uncharacterized sequence $\bar{s}$, a) $\bar{s}_1$ is a homologue of $\bar{s}$ with a similarity (e-value) of e^{-30} and $\bar{s}_2$ is a homologue with a similarity of e^{-5}, and b) in the reference database used for homologue discovery, the GO terms $\hat{t}_1$ and $\hat{t}_2$ annotate $\bar{s}_1$ and $\hat{t}_2$ annotates $\bar{s}_2$. In one respect, since $\hat{t}_1$ annotates $\bar{s}_1$, $\hat{t}_2$ annotates $\bar{s}_2$, and $\bar{s}_1$ is more similar to $\bar{x}$ than $\bar{s}_2$ (e^{-30} compared to e^{-5}), $\hat{t}_1$ should be a stronger candidate than $\hat{t}_2$. However, in another respect, since $\hat{t}_2$ annotates both $\bar{s}_1$ and $\bar{s}_2$ while $\hat{t}_1$ only annotates $\bar{s}_1$, $\hat{t}_2$ should be a stronger candidate. These observations suggest that each term should be assigned a *confidence value* that is determined by a) the similarity and b) the number of the sequence(s) in H_x that it annotates. We therefore define Definition 2 as:

Definition 2: For a target $\bar{x}$, each candidate term $\hat{t}_c \in C_x$ is given a confidence value $CV(\hat{t}_c)$, which is initially affected by the summation of the similarity (to $\bar{x}$) of each sequence in H_x that is annotated using $\hat{t}_c$.

For a given H_x and the derived T_x, let $C_x = T_x \cup \{\hat{t} \mid \hat{t}_i \in T_x, \hat{t} \succ \hat{t}_i\}$, then initially,

$$\begin{cases} CV_i(\hat{t}) = \sum_{\forall \bar{h} \in H_x \text{ annotated by } \hat{t}} similarity(\bar{h}, \bar{x}), \text{ where } \hat{t} \in T_x \\ CV_i(\hat{t}) = 0, \text{ where } \hat{t} \in C_x - T_x \end{cases} \quad ..(1)$$

On the GO graph, since the relation between a parent term and a child term is either a "is-a" or a "part-of" relation, a parent node describes properties exhibited by all its child nodes. Therefore, if a low-score term $\hat{t}_p \in C_x$ ($CV(\hat{t}_p)$ is low) has a set of high-score child nodes $CH = \{\hat{t}_c \mid \hat{t}_c \prec \hat{t}_p, CV(\hat{t}_c) \text{ is high}\}$, then although it has a low initial score, $\hat{t}_p$ is considered more informative than all *CH* terms and should be given a higher confidence value (score). Based on this observation, we define Definition 3 as:

Definition 3: The confidence values of child candidates propagate to and accumulate at their parent candidates.

However, the definition does not consider the information revealed by the distance between two nodes. If confidence values propagate indefinitely towards the root, the root node of C_x would always possess the highest score. To solve this, we first define the distance between two nodes:

Definition 4: We define the distance between two nodes $dist(\hat{t}_x, \hat{t}_y)$ on DAG_{GO} as the minimum number of edges between $\hat{t}_x$ and $\hat{t}_y$.

We note that Definition 4 assumes a uniform proximity function for the entire GO DAG—i.e., the mapping between two nodes' distance and similarity is the same regardless of their GO location. We will discuss this in more detail in Section 3.2.

We then study the effect of distance on nodes using Figures 3-a and 3-b, both in which $\hat{t}_x$ and $\hat{t}_y$ have a common ancestor $\hat{t}_a$. If we only consider Definition 3, we will assign to both $\hat{t}_a$ identical confidence values without taking into account their distance from $\hat{t}_x$ and $\hat{t}_y$. However, the $\hat{t}_a$ in Figure 3-a should be more informative than that in Figure 3-b because the $\hat{t}_a$ in Figure 3-b is biologically weaker in precision. This intuition leads to Definition 5.

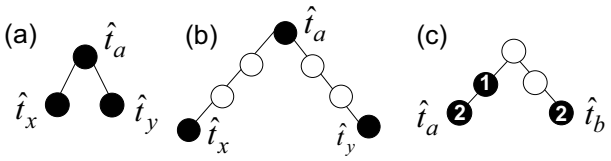

Figure 3. Examples of CS distributions.

Definition 5: The weight of a term decays exponentially for each propagation to a lower level.

For a node $\hat{t}_d$, Let $CV_p(\hat{t}_d, \hat{t}_a)$ denote the value it propagates to an ancestor node $\hat{t}_a$, then

$$CV_p(\hat{t}_d, \hat{t}_a) = CV(\hat{t}_d) * \alpha^{\tau}, \quad 0 < \alpha < 1, \text{and } \tau = dist(\hat{t}_a, \hat{t}_d) \quad \text{.....(2)}$$

From (1) and (2), we derive:

$$CV(\hat{t}_a) = \left(\sum_{\forall \hat{t}_d \in C_x, \hat{t}_d \prec \hat{t}_a} CV_p(\hat{t}_d, \hat{t}_a) \right) + CV_i(\hat{t}_a) \quad \text{.....(3)}$$

α values range between 0 and 1. The closer that α is to 1, the more the function favors coverage. In other words, α determines the amount of a node's contribution to its ancestor nodes. Essentially, Definitions 3 to 5 suggest that a node is considered a strong candidate if it has descendants that are strong candidates. A more in-depth observation is that strong candidates often form a cluster, and Definitions 3 to 5 try to identify clusters within a C_x distribution. However, this intuitive model neglects an important fact—nodes having strong candidate parents should also be considered as strong candidates. Figure 3-c

illustrates a C_x distribution; numbers within a node indicate the initial confidence value of that node. In the figure, although nodes $\hat{t}_a$ and $\hat{t}_b$ have the same initial confidence value, $\hat{t}_a$ should be a stronger candidate than $\hat{t}_b$, because an obvious cluster exists around $\hat{t}_a$. To enhance our model, we propose Definition 6.

Definition 6: The confidence values of parent candidates propagate to their child candidates.

Based on Definition 6, for a node $\hat{t}_a$, let $CV_{p2}(\hat{t}_a, \hat{t}_d)$ denote the value it propagates to a descendent node $\hat{t}_d$, then:

$$CV_{p2}(\hat{t}_a, \hat{t}_d) = CV_i(\hat{t}_a) * \beta^{\tau}, 0 < \beta \text{ and } \tau = dist(\hat{t}_a, \hat{t}_d) \quad ...(4)$$

β determines the weight of a node's contributed to descendent nodes. Finally, combining (1), (2) and (4), we derive Equation 5:

$$CV(\hat{t}) = \left(\sum_{\forall \hat{t}_d \in C_x, \hat{t}_d \prec \hat{t}} CV_p(\hat{t}, \hat{t}_d) \right) + \left(\sum_{\forall \hat{t}_a \in C_x, \hat{t}_a \succ \hat{t}} CV_{p2}(\hat{t}, \hat{t}_a) \right) + CV_i(\hat{t}) \quad .....(5)$$

In (5), the $\lambda = \alpha / \beta$ ratio determines whether precision or coverage is favored; a greater λ results in the model favoring coverage over precision, while a lower λ results in an emphasis on precision. Since proper α and β values vary for different applications, we propose (in Section 4) an algorithm that uses application-specific training data to determine the optimal α and β combination. This allows our model to be adaptive to different applications. Finally, for a target $\bar{x}$, the time complexity of GOMIT is $O(|C_x|)$.

3.2. Proximity Function

Since no literature yet exists on a biologically reasonable proximity function for the GO DAG, we have assumed a uniform function for this work. On the surface, this assumption may seem biologically inaccurate since the same distance may reflect different similarities on different GO DAG locations. However, a deeper observation justifies our assumption.

Accordingly, we can design a proximity function based on the number of GO terms that share a path to the root. The distance between two nodes is normalized by considering the longest path from root that contains the two nodes. We implemented this seemingly more reasonable proximity function and conducted experiments to compare its performance with the original uniform proximity function. The uniform function yielded much superior results, and we give the reasons below.

First, our proposed non-uniform distribution implies that certain (shorter) GO branches can be divided into more detailed GO terms. This is true biologically, but in practice, biologists were limited to GO terms available at hand when annotating sequences. Therefore, although distances computed by the non-uniform function do reflect better the actual biological similarity between GO terms, they do not reflect the real biological distances between the functional/biological/cellular properties of sequences. This introduces an inherent imprecision in the reference database—that is, a target's candidate term distribution no longer characterizes its function as precisely.

Second, candidate GO terms in any GOMIT application will form small, dense groups on the GO DAG. The goal of the GOMIT algorithm is to correctly handle weight propagation within these small groups in order to identify the most informative terms. A uniform distribution serves this purpose better than a non-uniform distribution.

We therefore suggest that adopting a uniform proximity function is justifiable.

4. Experiment Design

To evaluate GOMIT, we developed a system consisting of three main modules: GO-VIS, GO-MIT, and GO-GC. GO-GC contains several plugin programs for different experimental purposes. GO-MIT implements GOMIT algorithm. GO-VIS uses a tree structure similar to the public tool AmiGO [14]. We demonstrate GOMIT's generality by using it in three different applications—determining functional annotations, obtaining biological meanings of microarray data clusters, and assigning GO terms to protein families.

4.1. Evaluation Criteria

In our experiments, each target $\bar{x}$ is originally annotated with a set O_x of GO terms, and GO-MIT selects a set of terms having the largest confidence values to form a most informative term set M_x. The number of the selected terms depends on the application and user requirements. X is a set of targets for a certain application. For a target $\bar{x}$, we propose two criteria to compare M_x and O_x and to reflect GOMIT's accuracy.

Criteria 1 (Precision): We define criterion ρ as the percentage of GOMIT's annotations that matches the real ones.

$$\rho = \frac{\left|\left\{(\hat{t},\bar{x}) \mid \bar{x} \in X \text{ and } \hat{t} \in M_x \text{ and } \hat{t}_o \in O_x \text{ and } \text{dist}(\hat{t},\hat{t}_o) \le 1\right\}\right|}{\left|\left\{(\hat{t},\bar{x}) \mid \bar{x} \in X \text{ and } \hat{t} \in M_x\right\}\right|}$$

For a target, ρ evaluates GOMIT's ability to predict exactly one function that is within one distance from the target's correct function.

Criteria 2 (Recall): We defined σ as the percentage of O_x matched by M_x. It reflects the percentage of the real annotations that were matched by GOMIT. Formally,

$$\sigma = \frac{\left|\left\{(\hat{t}_o,\bar{x}) \mid \bar{x} \in X \text{ and } \hat{t}_o \in O_x \text{ and } \hat{t} \in M_x \text{ and } \text{dist}(\hat{t}_o,\hat{t}) \le 1\right\}\right|}{\left|\left\{(\hat{t}_o,\bar{x}) \mid \bar{x} \in X \text{ and } \hat{t}_o \in O_x\right\}\right|}.$$

4.2. Automated Functional Annotation

For our first experiment, automated functional annotations, we developed a GO-GC plugin that implements a homologue-discovering algorithm. The plugin uses BLAST [2] to search for similarities among organism sequences collected from SwissProt [17]. In conjunction with our previous developed EST annotation system [4], two parameters are associated with the homologue discovering algorithm: a) ψ, the length of the matched segment between a target sequence and a sequence in the reference database (SwissProt database), and b) φ, the percentage of matching residues among all the residues in the segment. We adopted ψ =50 and φ =50 for our experiments. We also gathered GO annotations (each one associated with an evidence code) for SwissProt from the Gene Ontology website.

We used the 6,688 sequences of the *Saccharomyces* Genome Database (SGD) [16] [13] as our original test set. GO annotations associated with the SGD sequences were also taken from the Gene Ontology website. In SGD, 6,440 sequences are assigned to NON-IEA (Inferred from Electronic Annotation) codes. From the 6,440 SGD sequences, we excluded the 2,395 ones having functional annotations with ND (non-defined) evidence and used the remaining 4,045 sequences. We then refined the 4,045 sequences into a subset *R*, which contained only those entries having at least one homologue—that is, a similar SwissProt sequence. Each of the 3,283 sequences in *R* was associated with at least one GO term. For each target, which is a sequence $\bar{s} \in R$, GO-GC processed $\bar{s}$ to generate a candidate term set C_s. Each term is initially weighted by a normalized score transferred from the e-value gathered from the alignment results [8]. After processing C_s, GO-MIT selected two terms with the largest confidence values to form a most informative term set M_s (i.e., $|M_s|$=2). To derive optimal α and β values for a particular application, we propose the following training method. We choose a set of training data whose correct annotations are known. We then annotate all training targets and calculate ρ for all combinations of α and β, where $0 \le \alpha \le 1$, $0 \le \beta \le 1.5$, and we increment by 0.1. In this experiment using SGD, we randomly chose 540 sequences (almost 1/6 of *R*) from *R* as our training set, and used ρ for evaluation. GOMIT's performance in terms of ρ is shown in Figure 4, where the optimal ρ occurs when α =0.6 and β =1.0. Therefore, we adopted this combination for this experiment.

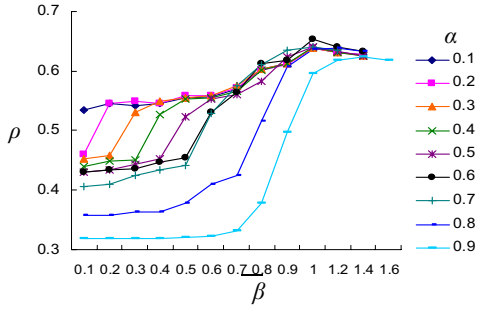

Figure 4. GOMIT's performance in terms of ρ for different combination of α and β values. The x axis represents β values, the y axis ρ, and each curve represents a particular α value.

We conduct several experiments on *R* to evaluate GOMIT's performance and to compare it with other algorithms. The first experiment E_{GOMIT} uses GOMIT with the trained α =0.6 and β =1.0. The second experiment E_{LCA} uses a least common ancestor (LCA) algorithm. The third experiment E_{HOM} directly analyzes the annotation associated with the homologues. It is similar to a statistical approach that chooses terms with the maximum initial confidence value as the most informative term. In E_{LCA}, the least common ancestor of a candidate set is considered the most informative term. E_{HOM} considers the term with the highest initial weight (See Equation 1 in Section 3) as the most informative term. Table I compares the three algorithm's performances. E_{GOMIT} has the best performance both in ρ (0.67) and δ (0.76), while E_{LCA} exhibited the worst performance. We observe that

one cause of this difference is attributed to noises within candidate distributions. E_{GOMIT}'s advantage reflects its noise elimination capability. E_{HOM} also demonstrated a low ρ (0.49), which shows that a statistical approach is weak for this application. This reflects the performance limitation when candidate distributions are neglected. Khan et al. [8] evaluated their GoFigure using only ρ, and claimed a ρ of 0.65. If GO-GC adopts ψ =70 and φ =50, GOMIT demonstrates an ρ of 0.69.

Table I. Experimental results.

	E_{GOMIT}	E_{LCA}	E_{HOM}	GoFigure
ρ	0.67	0.32	0.49	0.65
σ	0.70	0.38	0.46	N/P

N/P: not provided by Khan et al. [8].

4.3. Obtaining the Biological Meanings of Microarray Data Clusters

We designed a second set of experiments (Experiment 2) to test GOMIT's annotation ability with microarray data clusters. We used Eisen et al.'s [5] test set. They successfully applied clustering analysis to gene expression data for the budding yeast *S.cerevisiae*; all of the genes in their set have annotations in SGD [16]. We adopted the top 10 clusters, which were originally annotated by Eisen et al. In Table II, all of Eisen's keywords that are not found in the current GO are marked as "NON." For a cluster $\overline{G}$, we denoted Eisen et al.'s original annotations as E_G. For each $\overline{g} \in H_G$, GO-GC automatically gathered an associated GO term set $T_g = \{\hat{t} \mid \hat{t} \text{ annotates } \overline{g}\}$ from the Biological Process ontology (a GO branch). By analyzing $T_G = \bigcup_{\forall g \in H_G} T_g$, GOMIT derives M_G, which determines the most informative biological meaning(s) of $\overline{G}$. GOMIT used α =0.5, β =1.0 and selected two terms with the largest confidence values to derives M_G which determines the most informative biological meaning(s) of $\overline{G}$ ($|M_G|$=2). To assess GOMIT, we compare M_G with both E_G and annotations produced from Lee et al.'s [10] AverPd method (denoted by V_G). The M_G, E_G, and V_G annotations are shown in Table II.

We analyzed the difference between M_G, E_G, and V_G. When $\hat{t}_e$ was marked as NON, we compared M_G to V_G. As shown in Table II, when adopting $|M_G| = 2$, GOMIT was capable of predicting all the clusters correctly. For those $\hat{t}_v$ marked as NON (meaning no such term exists in the current GO version) in E_G, GOMIT's annotations was fully consistent with those of AverPd's (clusters C, F, G, H, I, K). For those $\hat{t}_e$ not marked as NON, GOMIT predicted correct (clusters E, K) or more specific annotations (B, J). For those marked as NON in both E_G and V_G, GOMIT predicted a most similar term in the current Gene Ontology (cluster D). For those clusters whose AverPd annotations differ from Eisen's, GOMIT provided annotations (cluster B, K) that were closer to Eisen's annotations. Even when adopting $|M_G| = 1$, only cluster D's annotations (see Table II) matched neither E_G nor V_G. This however does not imply cluster D's annotations were insignificant because both E_G, and V_G's annotations were marked as NON, meaning that the annotations are not available in the current Gene Ontology. These results suggest that GOMIT is capable of obtaining informative biological meanings for microarray data clusters.

Table II. The M_G, E_G, and V_G annotations for Eisen et al.'s data set.

C	N	Kd (E_G)	AverPd (V_G)	GOMIT (M_G)
B	11	$\hat{t}_{e1}$:cytoskeleton	$\hat{t}_v$:microtubule nucleation	$\hat{t}_{m1}$:cell growth and/or maintenance
		$\hat{t}_{e2}$:cell cycle		$\hat{t}_{m2}$:mitotic cell cycle
C	27	$\hat{t}_{e1}$:protein degradation (NON)	$\hat{t}_v$:ubiquitin-dependent protein catabolism	$\hat{t}_{m1}$:ubiquitin-dependent protein catabolism
				$\hat{t}_{m2}$:modification-dependent protein catabolism
D	14	$\hat{t}_{e1}$:mRNA splicing (NON)	$\hat{t}_v$:mRNA Splicing (NON)	$\hat{t}_{m1}$:aerobic respiration
				$\hat{t}_{m2}$:nuclear mRNA splicing, via spliceosome
E	17	$\hat{t}_{e1}$:glycolysis	$\hat{t}_v$:glycolysis	$\hat{t}_{m1}$:glycolysis
				$\hat{t}_{m2}$:main pathways of carbohydrate metabolism
F	22	$\hat{t}_{e1}$:protein synthesis (NON)	$\hat{t}_v$:protein biosynthesis	$\hat{t}_{m1}$:protein biosynthesis
				$\hat{t}_{m2}$:protein metabolism
G	15	$\hat{t}_{e1}$:ATP synthesis (NON)	$\hat{t}_v$:ATP Synthesis Coupled transport	$\hat{t}_{m1}$:ATP synthesis coupled proton transport
				$\hat{t}_{m2}$:oxidative phosphorylation
H	8	$\hat{t}_{e1}$:chromatin structure (NON)	$\hat{t}_v$:chromatin assembly/ disassembly	$\hat{t}_{m1}$:chromatin assembly / disassembly
				$\hat{t}_{m2}$:establishment and/or maintenance of chromatin architecture
I	138	$\hat{t}_{e1}$:protein synthesis (NON)	$\hat{t}_v$:protein biosynthesis	$\hat{t}_{m1}$:protein biosynthesis
				$\hat{t}_{m2}$:protein metabolism
J	5	$\hat{t}_{e1}$:DNA replication	$\hat{t}_v$:DNA replication initiation	$\hat{t}_{m1}$:DNA-dependent DNA replication
				$\hat{t}_{m2}$:pre-replicative complex formation and maintenance

K	15	$\hat{t}_{e1}$:TCA cycle (NON) $\hat{t}_{e2}$:oxidative phospho-rylation	$\hat{t}_v$:aerobic respiration	$\hat{t}_{m1}$:aerobic respiration $\hat{t}_{m2}$:tricarboxylic acid cycle	

*C—Cluster ID, N—Number of members, Kd—Keyword by Eisen et al., NON—No such term name in current GO.

4.4. Protein Family GO Assignments

We describe our third set of experiments for testing GOMIT performance in assigning GO terms to protein families. In an experiment involving their Gene Ontology Categorizer, Joslyn et al. [7] used data from the InterPro project, which catalogs the assignments of protein families, domains, and functional sites to GO terms [15]. We used InterPro data to evaluate GOMIT's automated GO assignment ability and to make a comparison between GOMIT and the Gene Ontology Categorizer. Here we use O_I to denote the original annotations for an InterPro family I, and P_I to denote the set of proteins belonging to I. For each protein $\bar{i} \in P_I$, GO-GC automatically produces an associated GO term set $T_i = \{\hat{t} \mid \hat{t} \text{ annotates } \bar{i}\}$ using data obtained from the Gene Ontology website. GOMIT then analyzes $T_I = \bigcup_{\forall i \in P_I} T_i$ to derive a set M_I of the most informative terms.

For our experiment, we used a newer (Dec. 8th of 2004) version of InterPro data compared to that used by Joslyn et al. We found that the average number of proteins in an InterPro family was 431. For each GO branch, we adopted $|M_I|$=3. To derive an optimal combination of α and β , we randomly chose 500 InterPro families to create a training set and used σ to evaluate GOMIT's performance using different α and β combinations. We observed few differences—in fact, the best performance occurred when α =0 and β =0, which reduced GOMIT to a statistical method relying on initial confidence values to determine the most informative terms (see (1) in Section 3). GOMIT achieved accuracies above 99% under this configuration, suggesting that a statistical algorithm is adequate for annotating the InterPro data set. Our results are shown in Table III.

Table III. Experimental results.

Ontology	InterPro #	σ
Molecular_function	5,810	>97.9%
Biological_process	5,269	>99.4%
Cellular_component	3,337	>99.9%

*InterPro #—the number of InterPro families with a GO assignment associated with a certain GO branch.

Joslyn et al. [7] listed the top 25 GO nodes with a $|M_I|$ of 25 for three ontologies and achieved a σ of 99.1%. We used a smaller $|M_I|$ (3 for each GO branch) yet achieved comparable performance. Even at $|M_I|$ = 2, GOMIT yielded an average σ above 97% for GO assignments of branch cellular components and biological processes, and above 92% for branch molecular functions.

5. Conclusion and Future Work

We used this paper to describe the GOMIT algorithm for extracting informative GO terms by analyzing term distributions on a GO DAG. GOMIT makes use of a quantitative model with parameters that adjust the algorithm's balance between coverage and precision. These adjustable parameters can be trained to yield optimal results for a specific application.

We designed three experiments to demonstrate GOMIT's characteristics as a generic and adaptive algorithm and to compare its performance with those of other algorithms. Evaluations were made according to two criteria: ρ and σ. In Experiment 1 (automated annotation of uncharacterized sequences), we tested GOMIT against three other algorithms—homologue annotation (E_{HOM}), least common ancestor (E_{LCA}), and GOFigure [8]—and found GOMIT to outperform the other algorithms in terms of σ and ρ. In Experiment 2 (obtaining the biological meanings of microarray clusters), we observed comparable results between GOMIT and Lee et al.'s AverPd [10]. In Experiment 3 (protein family GO assignments), we showed that when reduced to a statistical algorithm (α=β=0), GOMIT exhibited comparable results to Joslyn et al.'s Gene Ontology Categorizer [7]. From our experiments, we conclude the following conditions suitable for GOMIT application:

1) no one strong candidate exists and therefore statistical methods will be inaccurate,
2) numerous candidates exists, including noises, and
3) candidates are associated with partial GO terms.

Compared to statistical analysis methods described in [1], [3], [12], [18], and [19], GOMIT takes full advantage of GO's hierarchical properties to determine a small set of the most informative terms among a group of weighted GO terms, rather than simply isolating overrepresented terms with respect to a reference gene set. Furthermore, no modification of the GO DAG is required. Finally, GOMIT eliminates unrelated noise; for algorithms based on lowest common ancestors [10], noises force ancestor selection toward lower levels, thus reducing a selection's biological significance.

GOMIT has two unique advantages over algorithms that also consider GO's hierarchical features (e.g., Lee et al. [10], GoFigure [8], Gene Ontology Categorizer [7], and Schug et al. [11]): generality and adaptability. According to our results, GOMIT can be applied to automated functional annotations, biological annotations of microarray clusters, and protein families GO assignments. To the best of our knowledge, GOMIT is the first generic algorithm that can be applied to such a wide range of applications.

In terms of adaptability, GoFigure's [8] cutoff threshold was obtained by a manual inspection of 500 SGD sequences, Schug et al. [11] incorporated heuristics rules derived from empirical research, while Gene Ontology Categorizer [7] relies on user-tuned parameters for optimal performance. In contrast, GOMIT's parameters can be trained to fit the requirements of different applications.

For future work, we have developed CLUGO which adopts another strategy to first identify a set of significant clusters within a candidate distribution. After clusters are identified, it assigns a representative term for each cluster. Therefore, the $|M_x|$ is variable for each target $\overline{x}$.

Acknowledgements

The work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC93-2752-E-002-006-PAE and NSC93-3112-B-001-018-Y.

References

[1] Al-Shahrour F., Diaz-Uriarte R., Dopazo J., "FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes," *Bioinformatics*, Mar 1;20(4), pp. 578-80, 2004.

[2] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J. "Basic local alignment search tool," *J Mol Biol*, 215(3), pp. 403–10, 1990.

[3] Beissbarth T., Speed T.P., "GOstat: find statistically overrepresented Gene Ontologies within a group of genes." *Bioinformatics*, 20(9), pp. 1464-5, 2004.

[4] BIO101 website. http://bio101.iis.sinica.edu.tw/.

[5] Eisen M.B., Spellman P.T., Brown P.O., Botstein D., "Cluster analysis and display of genome-wide expression patterns," *Proc. National Academy of Sciences*, Dec 8;95(25), pp. 14863-8, 1998.

[6] Hennig S., Groth D., Lehrach H., "Automated Gene Ontology annotation for anonymous sequence data," *Nucleic Acids Research* 31(13), pp. 3712-3715, 2003.

[7] Joslyn C.A., Mniszewski S.M., Fulmer A., Heaton G., "The Gene Ontology Categorizer," *Bioinformatics*, v. 20:s1, pp. 169-177, 2004.

[8] Khan S., Situ G., Decker K., Schmidt C.J., "GoFigure: automated Gene Ontology annotation," *Bioinformatics*, Dec 12;19(18), pp. 2484-5, 2003.

[9] Lee I.Y. , Ho J.M., Lin W.C., "An algorithm for generating representative functional annotations based on Gene Ontology," *Proc. DEXA Workshops*, pp. 10-15, 2003.

[10] Lee S.G., Hur J.U., Kim Y.S., "A graph-theoretic modeling on GO space for biological interpretation of gene clusters," *Bioinformatics*, 20(3), pp. 381-8, 2004.

[11] Schug, J., Diskin, S., Mazzarelli, J., Brunk, B.P., Stoeckert CJ Jr., "Predicting Gene Ontology Functions from ProDom and CDD Protein Domains," *Genome Res* 12, pp. 648-655, 2002.

[12] Smid M., Dorssers L.C., "GO-Mapper: functional analysis of gene expression data using the expression level as a score to evaluate Gene Ontology terms," *Bioinformatics*, 2004.

[13] Stanford Genomic Resources. http://genome-www.stanford.edu.

[14] The Gene Ontology Consortium, "Creating the gene ontology resource: design and implementation," *Genome Res* 11, pp. 1425-1433.

[15] The InterPro website. http://www.ebi.ac.uk/interpro/.

[16] The *Saccharomyces* Genome Database. http://www.yeastgenome.org/.

[17] The UniProt/Swiss-Prot Protein Knowledgebase. http://www.ebi.ac.uk/swissprot/.

[18] Zeeberg B.R., Feng W., Wang G., Wang M.D., Fojo A.T., Sunshine M., Narasimhan S., Kane D.W., Reinhold W.C., Lababidi S., Bussey K.J., Riss J., Barrett J.C., Weinstein J.N., "GoMiner: a resource for biological interpretation of genomic and proteomic data," *Genome Biol.* 4(4), pp. R28, 2003.

[19] Zhang B., Schmoyer D., Kirov S., Snoddy J., "GOTree Machine (GOTM): a web-based platform for interpreting sets of interesting genes using Gene Ontology hierarchies," *BMC Bioinformatics*, 5(1), pp. 16, 2004.