GEA: a Toolkit for Gene Expression Analysis

Jessica M. Phan
U. British Columbia

Raymond Ng*
U. British Columbia

Man Saint Yuen
U. British Columbia

Steve Jones
British Columbia Genome Sequence Centre

Abstract

In recent years, relating gene expression to cancer development and treatment has received a lot of attention. Unfortunately, the availability of effective analysis tools lacks far behind the availability of data. In this paper, we present the Gene Expression Analyzer (GEA) for performing cluster analysis on gene expression data. In particular, the GEA is developed to support the reality that cluster analysis is typically a multi-step process. The underlying model of the GEA provides a set of algebraic operators for manipulating the data, as well as the intermediate results. Moreover, the GEA provides facilities to help the user to identify candidate genes for further clinical analysis. Last but not least, the GEA is optimized to handle the high dimensionality of gene expression data.

1 Introduction

Currently gene expression data are being produced at a phenomenal rate. The general objective is to try to gain a better understanding of the functions of cellular tissues. In particular, one specific goal is to relate gene expression to cancer diagnosis, prognosis and treatment. However, a key obstacle is that the availability of analysis tools, or lack thereof, impedes the use of the data, making it difficult for cancer researchers to perform analysis efficiently and effectively. The Gene Expression Analyzer (GEA) presented in this paper is designed and developed to provide better data mining and analysis support for gene expression data. It makes three key contributions:

1. Amongst all data mining paradigms that have been proposed and studied, clustering is the most widely adopted paradigm for analyzing genomic data. Examples include the studies conducted by Eisen et al [3], Alon et al [1], and Den-Dor et al [2]. However, all these studies regard cluster analysis as a one-step process. That is, there is the assumption that one needs to apply a clustering algorithm only once to the data in order to get the desired outcome. Unfortunately, real data mining and cluster analysis is rarely a one-step process; it involves repeated manipulation of the data, as well as the results of previous manipulations. The proposed GEA attempts to model this reality by providing a set of operations for cancer/biology researchers to conduct cluster analysis more effectively. Underlying the GEA is an algebraic framework that allows the output of one operation to become the input of another.

2. A key objective for performing cluster analysis on gene expression data is to identify candidate genes for further analysis, including clinical studies in a more traditional laboratory setting. But conventional clustering techniques, such as the k-means algorithm, CLARANS [7], CLICK [11], and OPTICS [8], do not provide too much help in identifying such genes. In contrast, the proposed GEA provides various facilities to accomplish this task. One example is the integrated support for finding fascicles, as proposed by Jagadish et al [5].

3. There are two general types of data essential to effective gene expression analysis. As discussed above, the first type is the actual gene expression data, e.g., produced by the microarray technique [10], or the SAGE technique [12]. The second type is auxiliary meta data, such as for mapping tags to genes

*Person handling correspondence. Postal: Department of Computer Science, 2366 Main Mall, Vancouver, BC V6T 1Z4, Canada. Email: rng@cs.ubc.ca. Phone: 604-822-2394.

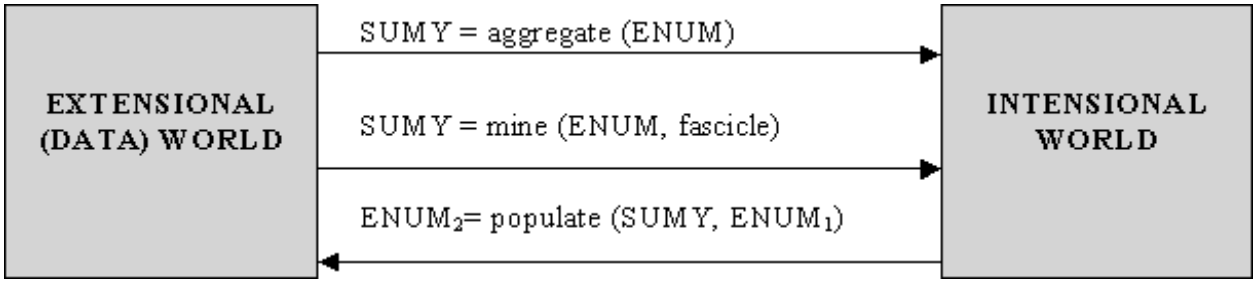

Figure 1: The GEA Model

Library Name	Tag_1	...	Tag_n
Library_1	$\text{expr-level}_{1,1}$	...	$\text{expr-level}_{1,n}$
...	...	...	...
Library_m	$\text{expr-level}_{m,1}$	...	$\text{expr-level}_{m,n}$

Figure 2: The Structure of an ENUM Table

(UNIGENE), for mapping genes to proteins (SWISSPROT), for classifying proteins into functional families (PFAM), or even for identifying scientific publications studying a particular gene or protein (PUBMED). From a database standpoint, the second type of data is more traditional in that the main operations on the data are search, retrieval and archival. The first type of data is somewhat different, in that the main operation on the data is analysis. Thus, the key challenge here is how to design and develop a system that can support these diverse requirements, preferrably on top of a relational DBMS. As summarized above, the GEA allows for mining and analysis on the one hand. It supports search and retrieval operations on the auxiliary databases on the other hand.

2 The GEA Model and Structures

Figure 1 shows the model underlying the GEA, within which gene expression data and intermediate clusters/fascicles can take on dual identity. In [6], the 3W model was proposed by Johnson et al. to capture the fact that data mining is often a multi-step process. The 3W stands for the three worlds for data mining in general [6]. For the kind of analysis to be supported here, the framework underlying the proposed GEA is a specialization of the general 3W model in at least three key aspects. First, there need only be two worlds, not three. Second, the structures are tailor-made for the GEA, namely the SUMY and GAP structures to be introduced shortly. Finally, there are additional operators, such as *diff()*, that are included. The following sections elaborate on these aspects.

While in the extensional world, a cluster is represented by an explicit enumeration of all the libraries contained in that cluster. Alternatively, while in the intensional world, a cluster is represented by a definition (i.e., a set of conditions) that are satisfied by all the libraries contained in the cluster. This is the key to capturing the multi-step nature of a cluster analysis process, which often involves manipulations of previous intermediate results/clusters. Below we first introduce the structures in the two worlds. In the next section, we will show how these structures can be manipulated. And in section 5, we will give comprehensive examples of these manipulations.

A cluster in the extensional world is represented as a relation. In particular, we call such a relation an ENUM table (as in "enumeration") with its structure shown in Figure 2. The columns represent the (compact) tags, and the rows correspond to the libraries. For instance, if cluster/fascicle A consists of the 1st, 2nd, 3rd and 4th libraries, then there are four rows in the corresponding ENUM table, with the columns representing the compact tags of the fascicle. To continue, the ENUM table for another cluster B may have a different number and a different set of columns (because the sets of compact tags are different), as well as a different number of rows (even though the same library may be included in multiple clusters).

Because an ENUM table is nothing more than an instance of an ordinary relation, the extensional world also supports any other ordinary relation. More specifically, the original SAGE data set can be stored in an

Tag Name	Range	Average	Standard Dev.
Tag_1	$[\min_1, \max_1]$	avg-level_1	dev-level_1
...	...	...	...
Tag_n	$[\min_n, \max_n]$	avg-level_n	dev-level_n

(a) A SUMY Table

Tag Name	Gap
Tag_1	gap-level_1
...	...
Tag_n	gap-level_n

(b) A GAP Table

Figure 3: Structures in the Intensional World

ENUM table as shown in Figure 2, as a "degenerate" cluster. And there may be auxiliary data associated with each library, such as the tissue type, cancerous or normal, the number of tags, etc. These pieces of auxiliary data are stored in ordinary relations; see section 6 for a more comprehensive discussion.

Whereas in the extensional world a cluster is represented by an explicit enumeration of all the libraries it contains, in the intensional world, a cluster is represented by its definition - the set of compact tags and their ranges. This leads to the SUMY table (as in "summary") shown in Figure 3(a). The rows correspond to the compact tags of the cluster. The columns give the range, the mean and the standard deviation of the mRNA expression levels for that tag in the cluster. Additional columns, such as those representing other aggregate values, may also be included.

In the intensional world, apart from the SUMY table, there is another structure called the GAP table. As will be shown in the next section, a Gap table is used to summarize the "difference" between two SUMY tables. This is particularly useful for distinguishing candidate genes that may have different levels of expression in different clusters.

3 Algebraic Operations of the GEA

So far, we have introduced the basic structures within the GEA. In this section, we describe how these structures can be manipulated, with an emphasis on algebraic operations beyond the standard ones.

3.1 Moving between the Worlds

Let us first consider the primary operation of data mining. In the general case, the *mine()* operator basically takes a data set from the extensional world and produces an intermediate result represented in the intensional world. Specifically for this paper, the mining operation is cluster/fascicle production, with the input being the SAGE data set (or any ENUM table) and the output being a cluster represented in a SUMY table. This is shown in Figure 1. Note that in the general case, the mining operation can be something other than fascicle production. But for simplicity, we focus only on fascicles here.

Recall that a SUMY table only stores the definition of the cluster. To obtain an explicit enumeration of all the libraries satisfying the definition, the *populate()* operator takes a data set (or any ENUM table) and a SUMY table, and finds all rows in the input data set satisfying the conditions laid out in the SUMY table. In other words, the populate operator converts a cluster from its intensional/SUMY form to its extensional/ENUM form, with respect to a given data set.

Finally, to complete the discussion of the operators shown in Figure 1, the *aggregate()* operator can be regarded as an inverse operation to the populate operator. It converts a cluster from the extensional/ENUM form to the intensional/SUMY form.

Recall that there are two key objectives of the GEA: (i) to support the multi-step nature of cluster analysis; and (ii) to help to identify candidate genes for further clinical analysis. So far, we have seen how the GEA achieves the first objective. That is, it uses algebraic operators to manipulate the data and the clusters. Output from one operator can be input for another operator. With respect to the second objective, so far fascicle production is the only avenue, because typically the number of compact tags in a fascicle is smaller than the original number of tags. Having said that, however, the number of compact tags can still

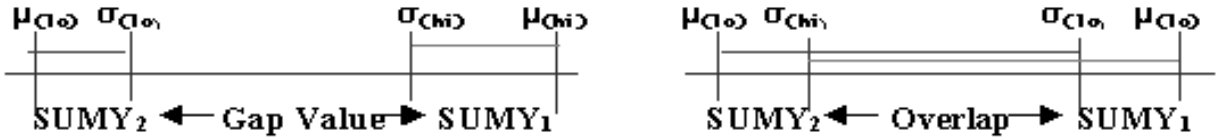

Figure 4: Gap Level Definition: Non-overlap (left) Overlap (right)

Tag Name	Range	Avg.	SD.
Tag_1	[5,5]	5	0
Tag_2	[0,7]	3	1
Tag_3	[10,120]	70	15
Tag_4	[0,20]	10	4

(a) Table $SUMY_1$

Tag Name	Range	Avg.	SD.
Tag_1	[0,14]	7	1
Tag_3	[10,130]	60	25
Tag_4	[0,12]	3	1
Tag_5	[0,50]	20	15

(b) Table $SUMY_2$

Tag Name	Gap
Tag_1	-1
Tag_3	null
Tag_4	+2

(c) Table GAP

Figure 5: An Example: GAP = diff($SUMY_1$, $SUMY_2$)

be too large (e.g., tens of thousands) for further clinical analysis. This motivates some of the operations presented next for manipulating the SUMY and GAP tables in the intensional world.

3.2 From SUMY Tables to GAP Tables

One key reason for identifying candidate genes is to capture the differences between libraries in different tissue types, and/or differences between libraries in the same tissue type but in different categories (e.g., cancerous or normal). This comparative analysis provides the opportunities to discover the potentially interesting information between the libraries. Within the GEA, the key operation in this regard is the *diff()* operator that takes two SUMY tables to produce one GAP table, i.e., GAP = diff(SUMY1, SUMY2). For instance, the first SUMY table may correspond to a cluster containing only cancerous breast tissues, whereas the second SUMY table corresponds to normal breast tissues.

Many definitions of a gap value/level are acceptable. In the sequel, we use the following definition that appears to be meaningful in practice. For each tag common in both SUMY tables, the gap level is defined as: gap level = $(\mu_{(hi)} - \sigma_{(hi)}) - (\mu_{(lo)} + \sigma_{(lo)})$, where $\mu_{(hi)}$ and $\sigma_{(hi)}$ represent the average and the standard deviation of the tag in the SUMY table having a higher average, and $\mu_{(lo)}$ and $\sigma_{(lo)}$ are the counterparts for the SUMY table having the lower average. Figure 4 shows the two cases depending on whether there is overlap. In the first case, for the given tag, there is no overlap, and the gap level is positive if the first SUMY table is having a higher average, negative otherwise. In the second case, there is overlap, and the gap level is defined to be null.

Let us consider the situation of the two SUMY tables shown in Figure 5. First, the resultant table GAP consists of rows corresponding to Tag_1, Tag_3 and Tag_4, because these are the common tags between the two SUMY tables. For Tag_1, the gap level is (7-1) - (5+0) = 1. But the sign of the gap level is negative because table $SUMY_1$ has a lower average. For Tag_3, there is an overlap and the gap level is null. Finally, for Tag_4, the gap level is (10 - 4) - (3+1) = 2, with a positive sign.

3.3 Other Operations in the Intensional World

Towards identifying candidate tags, a GAP table may still have too many rows. To further reduce the number of tags, the selection operator becomes handy. It takes as input a GAP table and produces a GAP table satisfying the specified selection conditions, as usually done in relational algebra. Just as easy, the selection operator can be applied to a SUMY table to produce another SUMY table. In this case, because a SUMY table contains range values (e.g., [min,max]), range selection conditions such as overlap are allowed.

Tag Name	Gap
Tag_1	-11
Tag_2	2
Tag_3	null
Tag_4	5

(a) Table GAP_1

Tag Name	Gap
Tag_1	-8
Tag_3	9
Tag_4	10
Tag_5	11

(b) Table GAP_2

Tag Name	Gap
Tag_2	2

(c) Table GAP_3

Tag Name	Gap_1	Gap_2
Tag_1	-11	-8
Tag_3	null	9
Tag_4	5	10

(d) Table GAP_4

Figure 6: An Example: GAP_3 = minus(GAP_1, GAP_2) and GAP_4 = intersect(GAP_1, GAP_2)

Range arithmetic is supported in the GEA; see [9] for details.

There are set operations allowed in the intensional world of the GEA as well. These operations apply to either a pair of GAP tables or a pair of SUMY tables. The intent is to manipulate at the level of tags. For example, the set minus operator extracts the tags appearing in the first table, but missing in the second table. Table GAP_3 in Figure 6 shows a simple example.

The set intersection operator extracts the common tags from the two tables and their corresponding values. Table GAP_4 in Figure 6 provides a simple example. Note that in this particular case, the GAP table has two gap columns. In general, as an extension to the basic situation shown in Figure 3(b), a GAP table must have one column on tag names, and at least one column on gap levels. Similarly, in general, a SUMY table can have more aggregate columns than the ones shown in Figure 3(a), so long as it has those columns shown in the figure. Finally, the union operator is defined similar to intersection. And there is the standard projection operator to remove unwanted columns in a GAP or SUMY table.

To complete the presentation of the algebraic framework underlying the GEA, we mention briefly the operations allowed in the extensional world. Because the extensional world is relational, the relational algebra, extended with standard aggregation operations such as sum, average, etc. and sorting, is sufficient. From an implementation point of view, this is significant because there is little effort required for the implementation of the extensional world.

4 Case Studies for the GEA

To demonstrate the kinds of results the GEA can deliver, we present in the following one case study we conducted using the SAGE data set. Other interesting case studies, involving different types of cancer analysis, can be found in [9].

Let SAGE be the relation containing the data. In this study, we compared normal brain tissues to cancerous brain tissues. Specifically, these were the steps taken. First, We used a simple relational selection to collect all libraries involving brain tissues in the extensional world, i.e., Ebrain = σ tissueType = 'brain' (SAGE). Then, We applied the fascicle algorithm to produce a cluster containing only cancerous tissues, i.e., $SUMY_1$ = mine(Ebrain, fasiscle). Correspondingly, we applied the populate operation to create the appropriate ENUM table, i.e., $ENUM_1$ = populate($SUMY_1$, Ebrain). Next, to set up the control groups for the comparative analysis, we created the following two ENUM tables. $ENUM_2$ contained all cancerous brain libraries that were not in the cluster, i.e., $ENUM_2$ = (σ tissueStatus = 'cancerous' (Ebrain)) - $ENUM_1$. Similarly, $ENUM_3$ contained all the normal brain tissues, i.e., $ENUM_3$ = (σ tissueStatus = 'normal' (Ebrain)). Similarly, to create the corresponding SUMY tables, we applied the aggregation operation, i.e., $SUMY_2$ = aggr($ENUM_2$) and $SUMY_3$ = aggr($ENUM_3$). Next, to contrast the various groups, we created the GAP table, i.e., GAP_1 = diff($SUMY_1$, $SUMY_3$). Finally, we removed all the tags in GAP_1 with overlapping ranges, sorted the remaining ones and plotted the results. Two examples are shown in Figure 7 and Figure 8.

Figure 7 shows an example of a gene identified by contrasting libraries in the fascicle and the normal brain libraries. The expression levels of Ribosomal Protein L12 of various libraries are shown, with asterisks corresponding to the cancerous libraries in the cluster, and squares corresponding to the normal libraries. The bar chart shows that the average expression level in the cancerous libraries in the cluster is around 275, compared with the average expression level of around 100 in the normal libraries. The values for the

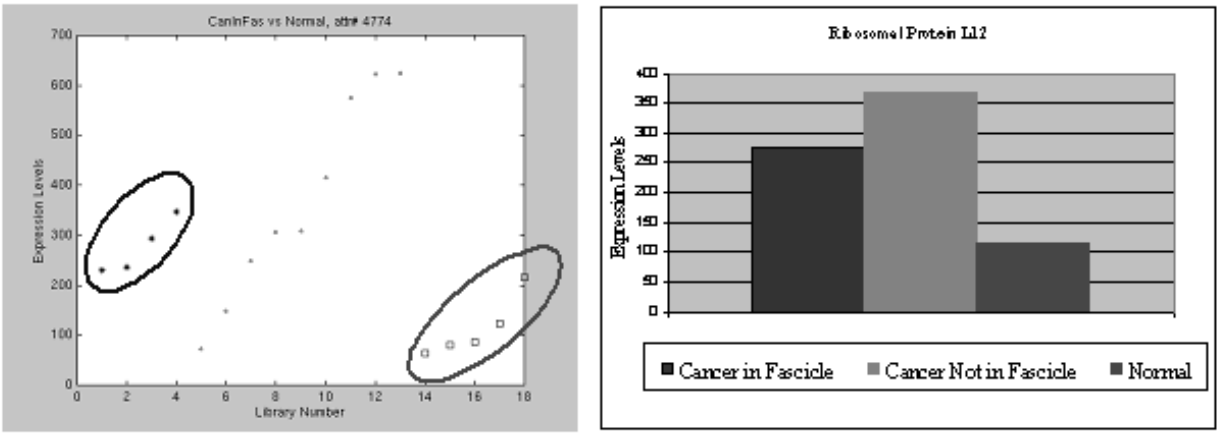

Figure 7: Fascicle vs Normal: Ribosomal Protein L12

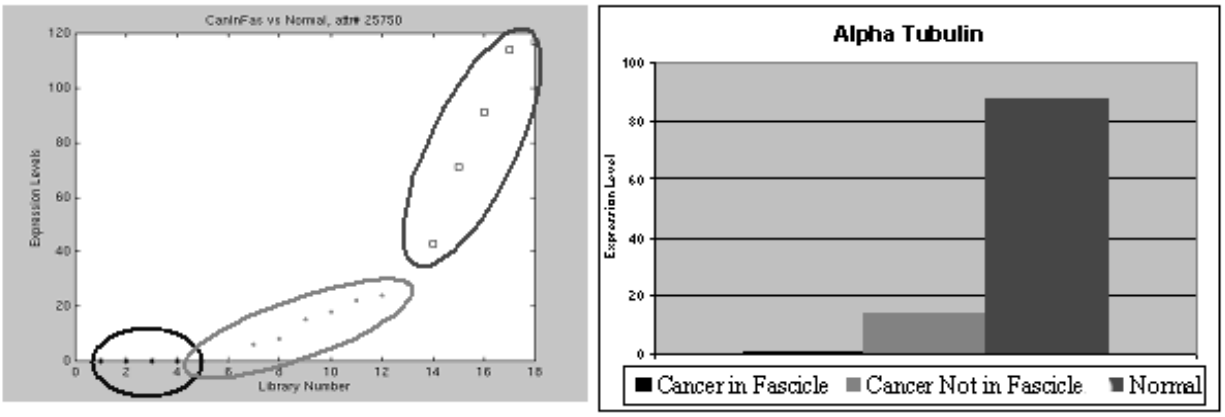

Figure 8: Fascicle vs Normal: Alpha Tubulin

cancerous libraries not in the cluster are also shown for reference.

While Figure 7 corresponds to a positive gap situation, Figure 8 corresponds to a negative gap scenario. The expression levels of Alpha Tubulin are much lower in the cancerous libraries (e.g., close to 0 for those in the cluster) than in the normal libraries (i.e., the average being close to 90).

5 Implementation and Efficiency Issues

The GEA is implemented in Java. The system details are all encapsulated inside the Java classes, each of this class responsible for a component of the system. One of the reasons for using Java is the simple integration and portability. The commercial database tool DB2 is used for providing the basic relational database support. The GEA has a tight coupling with the underlying DBMS to ensure efficient data access, storage and retrieval. For a discussion on some of the complications encountered during the development of the GEA, see [9]. In the following, we discuss some of the key optimization issues.

The GEA is intended to support interactive analysis. Thus, in the following, we discuss the complexity of the various operations. First, let us consider the three inter-world operations shown in Figure 1. The complexity of the mine() operator clearly depends on the exact nature of the mining operation. Specifically, for clustering, there are algorithms of varying complexities, from linear to polynomial on the number of libraries. In the case of fascicles, the complexity is linear with respect to the number of libraries and the number of compact tags [5].

Next, for the creation of a SUMY table, the time taken to apply the aggregate() operator depends on

the exact nature of the aggregation. If the aggregation on each tag simply amounts to finding the range, the average and the standard deviation, then one pass over the libraries is sufficient. But if the aggregation is more complex (e.g., finding the median), the complexity can be higher (e.g., $O(n \log n)$). As for the creation of a GAP table, the time taken is linear on the number of tags.

Finally, for the populate() operator, its performance turns out to be non-trivial. Recall that for a given SUMY table and an ordinary relation, the populate() operator finds all tuples in the relation that satisfy *all* the tag ranges contained in the SUMY table. On first sight, the operation is nothing more than a conjunction of a number, say p, of range conditions. However, the problem here is that p is very large by the normal standard of relational queries. For the case studies shown in the previous section, a SUMY table easily contains $p = 25,000$ or $p = 30,000$ tags. Thus, the populate operation becomes extremely high-dimensional, amounting to a conjunction of 25,000 or 30,000 range conditions.

Clearly, we cannot index each and every one of them. Thus, the first question is which ones to pick. One natural strategy is to pick the most selective tags. Because a SUMY table is produced by a mining operation, there is little that is known a priori as to which tags and what ranges will be included in the SUMY table. Our heuristic is to pick the tags with the highest entropy, i.e., highest variation. More specifically, we seek to build m indices for the tags with the top-m highest entropy.

The second, and more difficult, question is what an appropriate value of m is. To solve for m, we conduct the following analysis. Let the total number of tags be n (e.g., 60,000), the number of tags in a SUMY table be p (e.g., 25,000), and the number of indices built be m. Assume that there is a uniform distribution as to which tags are included in the SUMY table. Then the probability that a tag in the SUMY table is not indexed is:

$$Prob(a\ non{-}indexed\ tag\ selected) \quad = \quad (1 - \frac{m}{n})$$

Among the p tags included in the SUMY table, the probability that only non-indexed tags are picked is

$$Prob(none\ of\ the\ m\ indices\ hit) \quad = \quad (1 - \frac{m}{n})^p$$

To generalize, let w be the number of indices hit. That is, among the p tags included in the SUMY table, there are w tags with associated indices. Thus, the above equation corresponds to $Prob(w = 0)$, and can be generalized as follows:

$$Prob(exactly\ w\ indices\ hit) \quad = \quad C_w^p * (\frac{m}{n})^w * (1 - \frac{m}{n})^{p-w}$$

We can then fix a probability threshold say 0.999 such that we solve for the *smallest* m that guarantees at least a 99.9% chance of hitting at least w indices:

$$Prob(at\ least\ w\ indices\ hit) \quad = \quad [1 - \Sigma_{i=0}^{w-1} Prob(exactly\ i\ indices\ hit)] \quad \geq \quad 0.999$$

Based on $n = 60,000$ total tags, and $p = 25,000$ tags in a SUMY table, the solutions of m based on the above equation are:

At Least w Indices Hit	Number of Indices Required (m)
1	17
2	23
3	27
4	32
5	36

Finally, based on the SAGE data set used in the case studies, the following table shows the percentage of time saved for the populate() operator if w indices are hit.

w Indices Hit	0	1	2	3	4	5
Time Saved	0%	45%	76%	78%	85%	85%

The two tables combined tell an interesting story. To save around 50% of time executing the populate() operator, there is a 99.9% chance that if the top-17 highest entropy tags are selected for index creation, at least 1 index will be hit (out of the 25,000 tags included in the SUMY table). Similarly, if a 85% time saving is desired, then 32 indices are needed to guarantee that there is a 99.9% chance to have at least 4 indices hit.

6 Ongoing Work: Integrated Genomic Analysis

In this section, we describe ongoing work, specifically on how the GEA can be linked to other databases for integrated genomic analysis. Recently, the BC Genome Sequence Centre has started a project to build a repository integrating databases of various types. These databases include: (a) UNIGENE, which can be used to map a tag to a gene; (b) SWISSPROT, which can be used to relate genes to proteins; (c) PFAM, which classifies proteins into functional families; (d) KEGG, which gives information on genetic and biochemical pathway; (e) GENBANK, which contains information on nucleotides; and (f) PUBMED, which is a literature and patent database.

In the following, we show a few examples as to how these databases can be linked with the GEA to provide even more information to the user. The key point is that because the GEA model is consistent with the relational model, all the linkages between the GEA and these other databases can happen in a traditional querying environment, mainly through join queries. For example, to understand the biological meaning of a tag, the first step is to use the UNIGENE database. To look up the gene corresponding to a tag, the following relational algebraic expression suffices: $GeneRel = \pi_{Unigene.gene}(\sigma_{TagRel.tag=Unigene.tag}(TagRel \bowtie Unigene))$, where $TagRel$ is a relation with a column on tags (e.g., a SUMY or GAP table), and $Unigene$ is the relation containing a mapping from tags to genes. To continue, the user can map the genes to protein sequences. For example, $ProtRel = \pi_{Swissprot.sequence}(\sigma_{GeneRel.gene=Swissprot.gene}(GeneRel \bowtie Swissprot))$ retrieves the associated proteins.

These examples can go on and on, but the final example is a less conventional one. For a particular gene or protein sequence, many studies may have been published. It is important for the user to be linked to these publications. To this end, the PUBMED database can be used. The result from querying this database using the earlier relations $GeneRel$ or $ProtRel$ consists of a list of publications, linking the user to full-text journal articles at the appropriate web sites.

References

[1] U. Alon, N. Barkai, D. A. Notterman, K. Grish et al. Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. In *Proc. of National Academy of Sciences USA*, pp. 6745–6750, 1999.

[2] Amir Den-Dor, Ron Shamir, Zohar Yakhini. Clustering Gene Expression Patterns. In *Proc. The Third International Conference on Computational Molecular Biology*, pp. 281–297, 1999.

[3] Michael Eisen, Paul Spellman, Patrick Brown, and David Botstein. Cluster Analysis and Display of Genome-wide Expression Patterns. In *Proc. of National Academy of Sciences USA*, pp. 14863–14868, 1998.

[4] R. Heller, M. Schena et al. Discovery and Analysis of Inflammatory Disease-related Genes using cDNA Microarrays. In *Proc. of National Academy of Sciences USA*, pp. 2150–2155, 1997.

[5] H.V. Jagadish, Jason Madar, and Raymond T. Ng. Semantic Compression and Pattern Extraction with Fascicles. In *Proc. VLDB*, pp. 186–198, September 1999.

[6] Theodore Johnson, Laks V. S. Laksmanan, and Raymond T. Ng. The 3W Model and Algebra for Unified Data Ming. In *Proc. VLDB*, pp. 21–32, September 2000.

[7] Raymond T. Ng and Jiawei Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. VLDB*, pp. 144-155, September 1994.

[8] Raymond Ng, Jorg Sander, and Monica Sleumer. Hierarchical Cluster Analysis of SAGE data for Cancer Profiling. In *Proc. of BIOKDD Workshop on Data Mining in Bioinformatics*, pp 65–72, August 2001.

[9] Jessica M. Phan. GEA: a Toolkit for Gene Expression Analysis. Master Thesis, Department of Computer Science, University of British Columbia, May 2002.

[10] M. Schena, D. Shalon, R. Davis, and P. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray. In *Science*, 270, pp. 467–470, 1995.

[11] Ron Shamir, and Roded Sharan. CLICK: A Clustering Algorithm for Gene Expression Analysis, 1999.

[12] V. E. Velculescu, L. Zhang, B. Vogelstein, K. Kinzler. Serial Analysis of Gene Expression. In *Science*, pp. 484–487, November 1995.