

# **GeneTrek: An Information Retrieval System for Gene Clustering**

## **Study and Implementation of Interface and Study of Chameleon Clustering Algorithm**

**Saurav Sahay**

### **1. Motivation**

The vast amount of gathered genomic data from Microarray and other high throughput experiments makes it extremely difficult for the researcher to interpret the data and form conclusions about the functions of the discovered genes. We can make use of extensive biomedical literature databases like Medline to find the functional similarity between genes using various information retrieval and clustering techniques.

### **2. Introduction**

GeneTrek is a text mining system that takes gene names as input and finds out functional similarity between these gene names using various text processing algorithms and finally clusters these genes according to functional similarity. The text processing algorithms prepare the documents using the 'bag of words' model and process them using stemming algorithm, irrelevant word elimination and calculating the term frequency – inverse document frequency (TFIDF) for the keywords related to the genes in order to cluster them into related groups. This project involved a study and deployment of this system using a web interface. The interface takes gene names as input into the system and gives functionally related keywords and clusters the gene names according to the relationships. This required getting access to the Medline repository locally, converting the Medline XML database to relational format for querying; and interfacing the database over Apache server with other text processing programs to have a web application of GeneTrek system. Another efficient clustering algorithm called Chameleon was studied for its possible integration with this system.

### **3. Building the Medline Database**

MEDLINE is a large online and continuously growing biomedical bibliographic database that contains over 12 million citations from over 4,600 journals through PubMed web interface. It is a rich source of biomedical literature that is appropriate for research on text mining and information extraction in biomedical domains. Medline content is available to the licensee as XML text files (around 55 GB of uncompressed XML files). These huge files do not render themselves well for processing and searching and need proper indexing and searching utilities that can be provided by relational systems. Relational database schema can be designed from the DTD provided by NLM that defines the XML structure of these files. There are some tools available that parse these XML files and load the data to specific relational systems. A java package called BioTextEngine recently developed by a joint team at University of California at Berkeley and Stanford University was used for GeneTrek. This package parses the XML database to DB2 format using a java SAX (Simple API to XML) parser and communicates with

the database using JDBC. GeneTrek has been currently implemented using PostgreSQL DBMS since it is open-source, freely available and modifiable database system. The biotextengine package needed some modifications to parse the Medline XML files to PostgreSQL form. Currently, a part of Medline has been converted to relational format for testing. It took about 22 hours to parse 600 MB of XML data to PostgreSQL format on a Pentium M (1.6 GHz, 512 MB RAM) processor computer. The process took this much of time as SAX parser receives data through stream and recognizes document elements in an event-driven manner and loads them into the database.

#### 4. Setting up the Server and database connectivity

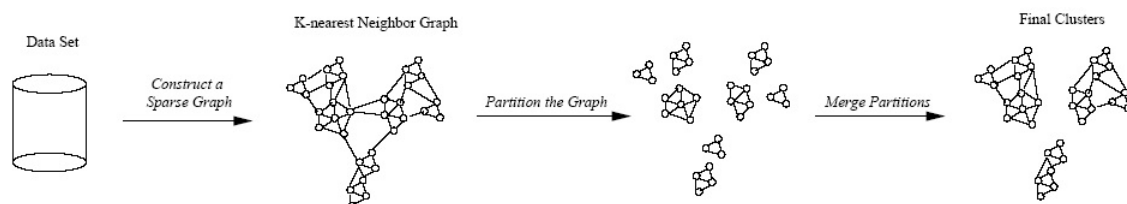
In order to implement the system over the web, Apache server was setup to allow CGI scripts and communicate with the database. These CGI scripts had to interface the database, the server and the GeneTrek perl scripts to connect all the components together. Perl DBI module was used to connect and query the Medline\_citation table in the database. Relevant abstracts were retrieved from the database for text processing and clustering.

#### 5. CHAMELEON Algorithm

Clustering is an unsupervised learning and discovery process that organizes a set of objects into meaningful groups. These groups can be of any shape and size that capture the most natural form of associated data. CHAMELEON is a clustering algorithm that explores graph partitioning and dynamic modeling in hierarchical clustering. It applies to all types of data as long as a similarity function can be constructed. Two clusters are merged in the clustering process only if the inter-connectivity and closeness between two clusters are high relative to the internal inter-connectivity and closeness of items within the clusters. This process facilitates the discovery of natural and homogeneous clusters.

CHAMELEON uses a graph partitioning algorithm to cluster the sparse graph data objects into a large number of relatively small subclusters. It then uses an agglomerative hierarchical clustering algorithm to find the genuine clusters by repeatedly combining these clusters using the connectivity and closeness measures.

CHAMELEON algorithm has been derived based on the observation of the weakness of two popular hierarchical clustering algorithms, CURE and ROCK. CURE and related schemes ignore information about the aggregate inter-connectivity of objects in two different clusters, they measure similarity between two clusters based on the similarity of the closest pair of the representative points belonging to different clusters. ROCK and related schemes ignore information about the closeness of two clusters while emphasizing their inter-connectivity, they only consider the aggregate inter-connectivity across the pairs of clusters and ignores the value of the stronger edges across clusters.



## Overall framework of CHAMELEON Algorithm [1]

CHAMELEON uses k-nearest neighbor graph approach to represent its objects. This graph captures the concept of neighborhood dynamically and results in more natural clusters. The neighborhood is defined narrowly in a dense region, whereas it is defined more widely in a sparse region.

CHAMELEON determines the similarity between each pair of clusters  $C_i$  and  $C_j$  according to their relative inter-connectivity  $RI(C_i; C_j)$ , and their relative closeness  $RC(C_i; C_j)$ . The relative inter-connectivity  $RI(C_i; C_j)$  between two clusters  $C_i$  and  $C_j$  is defined as the absolute inter-connectivity between  $C_i$  and  $C_j$ , normalized with respect to the internal inter-connectivity of the two clusters  $C_i$  and  $C_j$ .

$$RI(C_i; C_j) = \frac{|EC_{\{(C_i; C_j)\}}|}{\frac{|EC_{C_i}| + |EC_{C_j}|}{2}}$$

The Edge-Cut,  $EC_{\{(C_i; C_j)\}}$  (from Graph Theory) is defined to be the sum of the weight of the edges that connect the vertices in  $C_i$  to vertices in  $C_j$ . The Min-Cut bisector,  $EC_{C_i}$  is the weighted sum of edges that partition the graph into two roughly equal parts.

The relative closeness  $RC(C_i; C_j)$  between a pair of clusters  $C_i$  and  $C_j$  is the absolute closeness between  $C_i$  and  $C_j$ , normalized with respect to the internal closeness of the two clusters  $C_i$  and  $C_j$ .

$$RC(C_i; C_j) = \frac{S_{EC_{\{(C_i; C_j)\}}}}{\frac{|C_i| S_{EC(C_i)} + |C_j| S_{EC(C_j)}}{|C_i| + |C_j|}}$$

$S_{EC(C_i)}$  and  $S_{EC(C_j)}$  are the average weights of the edges that belong in the min-cut bisector of clusters  $C_i$  and  $C_j$ , respectively, and  $S_{EC_{\{(C_i; C_j)\}}}$  is the average weight of the edges that connect vertices in  $C_i$  to vertices in  $C_j$ . CHAMELEON's hierarchical clustering algorithm selects to merge the pair of clusters for which both  $RI(C_i, C_j)$  and  $RC(C_i, C_j)$  are high.

## **6. CHAMELEON Implementation**

The algorithm can be divided into three different parts. The first part comprises building the k-Nearest Neighbor (k-NN) Graph from the similarity matrix. The second part consists of partitioning the graph to find initial sub-clusters, and the third part relates to merging the partitions using RI and RC values.

### **6.1. Input**

The input to the first algorithm was a 26 Gene x Gene Matrix file that contained affinity values [2] (dot products of keyword relevance using z-score values) between the Genes.

This matrix was read by the first C program to compute the k-NN Graph from the file. The k value was passed as a parameter to the program.

## **6.2. Converting the matrix into k-NN Graph**

After obtaining the matrix, the individual genes were treated as vertices and the affinity values as the edges between the vertices. The algorithm computed the k highest affinity values from the individual genes to store as another graph consisting of k nearest vertices from each 26 set of genes.

## **6.3. Finding initial sub-clusters**

CHAMELEON uses a graph partitioning algorithm to find the initial sub-clusters to partition the k nearest neighbor graph of the data set into a large number of partitions such that the edge-cut is minimized. The original algorithm used the hMetis (Hypergraph partitioning package) library to partition the graph. hMetis is a software package for partitioning large hypergraphs, especially those arising in circuit design.[3] These algorithms are fine-tuned to work on very large graphs.

The 26 Gene set input was not suitable for hypergraph partitioning using hMetis. Instead, another software package library called Metis[4] that partitions irregular graphs efficiently using the same edge-cut minimization procedure was used to partition the k-NN Gene graph. The input to the Metis library was a formatted graph file containing the k-NN edges and their weights. Since the program computes the multilevel k way partitioning of the graph, k value was provided as input to the program. This k denotes the initial number of subclusters that are computed. It should be a value less than the expected number of natural clusters which can be later discovered by the merging part of the algorithm.

## **6.4. Finding Relative Interconnectivity and Relative Closeness**

In order to find the RI and RC values between different clusters, the Metis program was used to get the edge-cut of different clusters efficiently. The min-cut bisector of a cluster was calculated by adding all the weights of the edges of the individual cluster. This was done under the assumption that the vertices of the cluster were all tightly connected to each other and all of them were needed to be cut to partition the graph into two roughly equal parts.

## **6.5. Hierarchical Merging of the clusters**

Clusters are merged hierarchically if the RI and RC values are above some user specified threshold values. These threshold values control the variability of the clusters and as such have no absolute values. One can find good threshold by many experiments for a particular task. For our small Gene set, the RI threshold was chosen to be 0.1 and RC threshold was chosen to be 0.3.

## **6.6. Results**

CHAMELEON algorithm correctly identified 23 of 26 Genes in right clusters on the second pass of the merging phase based on particular threshold values. On choosing a different set of thresholds, it merged two clusters incorrectly. This difference could be highlighted since we knew which set of Genes were functionally related.

The initial sub-clusters:

(the numerical values indicate Gene numbers)

Cluster 1:	1	2	3	4	9		
Cluster 2:	10	11	12	13	14	15	16
Cluster 3:	19	20	21				
Cluster 4:	5	22	23	26			
Cluster 5:	6	17	18				
Cluster 6:	7	8	24	25			

The non-zero RI and RC values for the above sub-clusters:

(all other RI and RC values between clusters were 0)

RI(1,2) = 0.0093	RC(1,2) = 0.1474
RI(1,6) = 0.0141	RC(1,6) = 0.1683
RI(2,3) = 0.1234	RC(2,3) = 0.36
RI(4,6) = 0.1649	RC(4,6) = 0.825
RI(5,6) = 0.1913	RC(5,6) = 0.49

Based on the RI threshold of 0.1 and RC threshold of 0.3, clusters 2 and 3; 4 and 6; and 5 and 6; were merged and identified 2 of 4 clusters correctly and merged the remaining 2 clusters into one cluster.

## 7. Discussion

The outcome of this short experiment shows promising results for applying CHAMELEON algorithm for keyword based clustering of large number of genes. The k-NN graph captures the data to be modeled very nicely. The well-studied cut-based graph algorithms are suited for graph partitioning. Dynamic modeling is conceptually a sound method for any type of clustering. It inherently works to optimize the underlying clustering principle of minimizing the inter cluster similarity and maximizing the intra-cluster similarity. The problem with this algorithm is to automatically find good threshold values for the data. The Metis graph partitioning package has been written in C and could not be currently integrated with the GeneTrek system that is completely based on Perl and CGI scripts.

GeneTrek currently uses Bond Energy Algorithm (BEA) for clustering that is basically a database optimization algorithm that rearranges similar data together. It has been found out to be one of the best clustering algorithms presently for this task and does better than k-means, SOM and other popular algorithms. One problem with BEA similar to Chameleon is to decide the number of clusters in which to partition the data. This is again a hard problem and currently can be done heuristically.

## 8. Conclusion and Future Work

The initial GeneTrek interface can be further improved to get more significant and user friendly results. One thing that needs to be implemented is a graphical display of clustered results. We need to specify the number of clusters to partition the data now and the user does not know this value beforehand. This value needs to be computed automatically using various tools like AutoClass (Bayesian Classification). This again has a complex integration issue. Now that we have the entire Medline repository to our access, we can provide more search options and provide customized pre-computed clustered results to the users with a neat graphical display of results. The GeneTrek system can also be further improved to find alias names of the genes using gene ontologies and then do an exhaustive search on the database. There is scope for using Natural Language Processing (for ex., exclusion of sentences with negation) and include phrases as gene names to do effective searches. Furthermore, it can be combined with other classification tools like BLAST and KEGG to give an overall picture to the biologist to infer meaning from the data.

## 9. References

- [1] *Text Mining Biomedical Literature for Discovering Gene-to-Gene Relationships: A comparative study of algorithms.*  
Ying Liu, Alex Pivoshenko, Jorge Civera, Venu Dasigi, Ashwin Ram, Brian Ciliax, Ray Dingledine and Shamkant B. Navathe
- [2] *Tools for loading MEDLINE into a local relational database*  
Diane E Oliver, Gaurav Bhalotia, Ariel S Schwartz, Russ B Altman and Marti A Hearst
- [3] *Chameleon: Hierarchical clustering using dynamic modeling*  
George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar
- [4] *Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*  
George Karypis and Vipin Kumar
- [5] *PostgreSQL Documentation*
- [6] *Perl DBI::Pg Module Documentation*