02-512 Final Project

# Determining Gene Regulatory Networks in Human Astrocytes in Response to Influenza H5N1

Alex Muralles, Ruhani Mumick
Carnegie Mellon University

**Abstract:**

H5N1 is an influenza virus that infects humans and other animals. The mortality rate for humans infected with the disease is estimated to be about 60%, meaning that 60% of patients who contract the disease die. Astrocyte, a neural subtype important in neural homeostasis are affected by the virus, causing encephalopathy and encephalitis, diseases that impact the function and structure of the brain. Understanding the genetic networks that control astrocytes' response to infection can provide guidance on how to control the onset of encephalopathy and encephalitis. (Lin 2015).

Microarray gene expression data of 41091 genes were analyzed at 0h, 6h, 12h, and 24h after H5N1 infection. These genes span a wide variety of functions but include immune response genes and regulatory genes (GDS6010).

A graph minimum k-cut model is used to determine the clusters of genes that are involved in its regulatory network. The gene ontology of these genes in the clusters will then be found and used to determine the biological significance of our results. Since the k-cut model is an NP hard problem, we have used two heuristics for solving a hard problem - a brute force approach and a greedy algorithm. One of our goals was to compare the results from both approaches. Due to runtime issues, we were able to run our greedy algorithm on a graph size of 100 genes, yielding 20 clusters. These clusters include ones with genes with similar gene ontologies, indicating that our results make biological sense. Our brute force model is very slow, allowing us to only include 8 nodes. From these results, we conclude that we would need a more efficient model than our greedy algorithm to run minimum cut model on the graph size that microarray data can actually produce but that a minimum k-cut model does reveal biologically relevant clusters of genes.

## Introduction

The number of cases of H5N1 influenza virus infection has increased since 2007. However, vaccines have been stockpiled in order to prevent them from spreading, an initiative that has prevented the disease from spreading uncontrollably. In semi-rare cases, neural cells become infected with the virus and encephalopathy or encephalitis can occur. These brain diseases are commonly temporary but can be permanent.

The gene regulatory networks involved in the response of astrocytes to H5N1 is generally unknown. We seek to investigate, through biological modeling, the clusters of genes that work together to create a response. Specifically, genes that were screened involve those involved in osmotic stress, forming biofilms, and liposaccharide modification. These are pathways that are already thought to have a role in bacteria's response to antibiotics, either through robust or hypothetical hypotheses. These pathways are currently unclear because

methods of understanding biologically relevant genes in the past have been through knockout experiments where candidate genes are made to lose their function one-by-one. These methods are inefficient and take exorbitant amounts of lab time and money. They also fail to answer questions quickly about entire networks. Studying one gene may prove that it is relevant but its overall role within the entire regulatory network will remain unknown. To analyze large numbers of genes at once, computational methods are becoming necessary. Microarray data on its own can be used to compare a control set with an experimental one but direct comparison again leads to understanding the significance of a particular gene rather than a network.

A computational network of all genes solves this problem by considering the genes as a graph from the beginning rather than as individual players. A minimum k-cut method is used in this instance. This will determine subsets of genes that interact amongst themselves as a regulatory network. We will use two methods of solving the problem, a brute force algorithm and a greedy algorithm. Generally, brute force is much too inefficient to use reliably when considering large data sets. However, it is algorithmically simple and returns the optimal solution in theory. A greedy algorithm is more efficient but is not required to return the optimal solution. It may output a local minimum instead of a global one. Gene ontology provides a resource to determine what the functions of genes are. We used this database to compare with our regulatory networks to determine if they make biological sense.

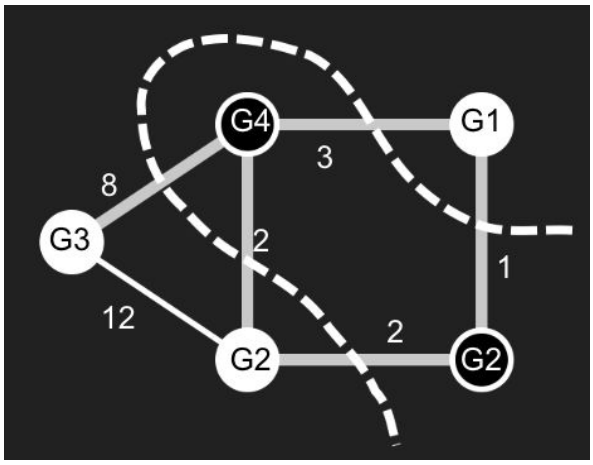



Figure 1. Model graph indicating a cut in the graph

This paper will introduce our graph problem methodology. It will also provide an analysis of the positives and negatives of using a brute force algorithm in relation to a greedy algorithm based on the results obtained.

**Methods**

To use as analysis, microarray data containing gene expression of genes suspected to be involved in astrocyte response to H5N1 was used from the Gene Expression Omnibus (GEO). This experiment included about 41,000 genes. Due to runtime issues, we picked a random sample of 100 genes to run the greedy algorithm on and a random sample of of 7

genes to run the brute force algorithm on. Results of gene networks were compared to Gene Ontology on the online Gene Ontology tool (amigo.geneontology.org).

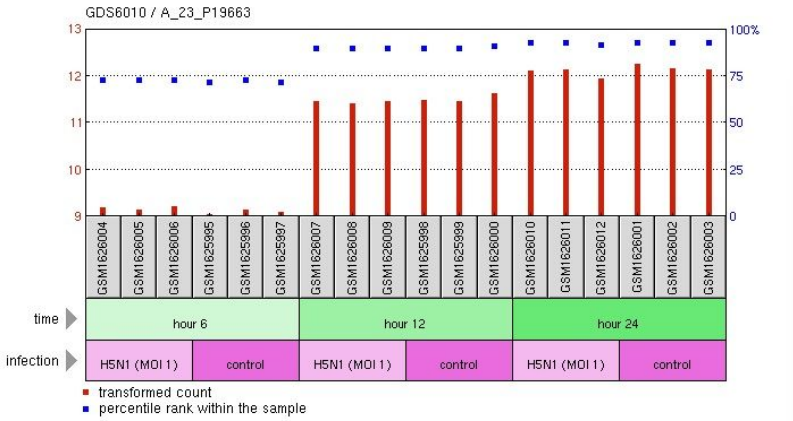



*Figure 2. Sample Datapoint from the GDS6010 dataset, with timepoints after 6,12,and 24 hours*

In order to determine subsets of genes that may represent a gene regulatory network, a minimum k-cut problem is posed. In this graph optimization problem, we represent genes as nodes and edges as relationships between the nodes in a weighted undirected graph. We start with all genes connected to all other genes.The following section describes the derivation of the weight function.

I. Weight Function

Gene expression data from the microarray analysis provided normalized data. For each replicate of a gene **r**, we calculated a difference d in gene expression **g** between control and experimental conditions at each time point **t**.

$$d_{gene\ x,r_1} = \left(g_{control} - g_{experimental}\right)^2$$

The d values for all r's are then averaged to get one value for a gene.

$$average\ d_{gene\ x} = \frac{\left(d_{r_1} + d_{r_2} + d_{r_3} + \cdots d_{r_n}\right)}{n}$$

where n is the number of replicates in the experiment. The average change in gene expression over time is then compared to the average gene expression of every other gene over time. A Pearson correlation coefficient.

$$Pearson\ Coefficient = \frac{COV(avg(d_{gene\ i}), avg(d_{gene\ j})}{\sqrt{VAR(d_{gene\ i})VAR(d_{gene\ j})}}$$

This Pearson correlation coefficient is the weight of the edge between gene i and gene j. We then remove all edges with a weight below 0.6.

Once this graph is created, then the goal of the minimum k-cut will be to partition the graph into k sections, denoted “clusters”, such that the weight on the edges it needs to cut is minimized.

This goal will be implemented through two methods - brute force and a greedy algorithm. In the brute force approach, we go through every combination of edges that it is possible to cut and that partitions the graph into the desired k sections. The algorithm then saves the weight of the cut and continually replaces the optimal solution as it goes.

*Greedy Algorithm Pseudocode*

1. *Start with the graph G and edges E, representing the graph in question, and $k_{desired}$ which is the desired k clusters.*
2. *TotalCost ← 0*
3. *Sort E by weight in ascending order.*
4. *Determine the current k, $k_{graph}$*
5. *While $k_{graph} < k_{desired}$:*
    a. *Select the lowest weight edge*
    b. *Eliminate the selected edge from the graph*
    c. *Add the cost of the cut to the total cut*
    d. *Recalculate $k_{graph}$*
6. *Determine the clusters C:*
    a. *While the graph has unclustered genes*
        i. *Start with one randomly selected node*
        ii. *Find everything connected to that node (and its neighboring nodes to exhaustion), this is a “cluster*
        iii. *Remove the “cluster” from the original graph*
7. *Return C, the modified graph (in clusters) and the cost of the minimum cut*

In the greedy algorithm, all edge weights are gone through iteratively in order of increasing weight and removed from the graph. After each edge removal, the graph is evaluated as to how many individual clusters there are. If there are less than k, then the next edge is removed. This is continued until the there are the desired number of clusters. This function alone runs in polynomial time, but the asymptotic complexity of determining the clusters and

building the graph are still to be considered in the entire program run itself, and need further optimization.

In the brute force approach to solving the problem, every possible set of cuts that could be made are considered, and their relative costs are calculated before the cut corresponding to the lowest cost cut is selected and returned. This algorithm does require the consideration of every possible set of cuts, and therefore runs in $2^n$ time where n is the size of the input edge set. It is not efficient whatsoever, but was used for comparison purposes.

*Brute Force Algorithm Pseudocode*

1. *Begin with the graph = (G,E), $k_{desired}$, and an optional stopping point threshold, t*
2. *All Possible Cuts ← the entire possible set of cuts. This is P(E), the powerset of our edge set*
3. *Minimum cost ← Infinity (to initialize)*
4. *Minimum cut ← a variable to hold the minimum cut itself*
5. *For every possible set of cuts S in all possible cuts:*
    a. *For every edge e in this particular set of cuts:*
        i. *Perform the cut e on the graph G*
        ii. *Cost = $\sum_{e}^{S} weight_e$*
    b. *If the cut results in k = $k_{desired}$:*
        i. *If Cost < minimum cost:*
            1. *Minimum cost ← cost*
            2. *Minimum cut ← cut S*
    c. *If the Minimum cost is below a threshold, should one exist, we are finished, return the cut graph and its clusters and cost.*
    d. *Repair the graph of all cuts before continuing to the next possible cut*
6. *Apply the resultant minimum cut to the graph*
7. *Determine the clusters from the minimum cut:*
    a. *While the graph has unclustered genes*
        i. *Start with one randomly selected node*
        ii. *Find everything connected to that node (and its neighboring nodes to exhaustion), this is a "cluster*
        iii. *Remove the "cluster" from the original graph*
8. *Return the Clusters, the Cut Graph, and the Minimum Cost*

**Results**

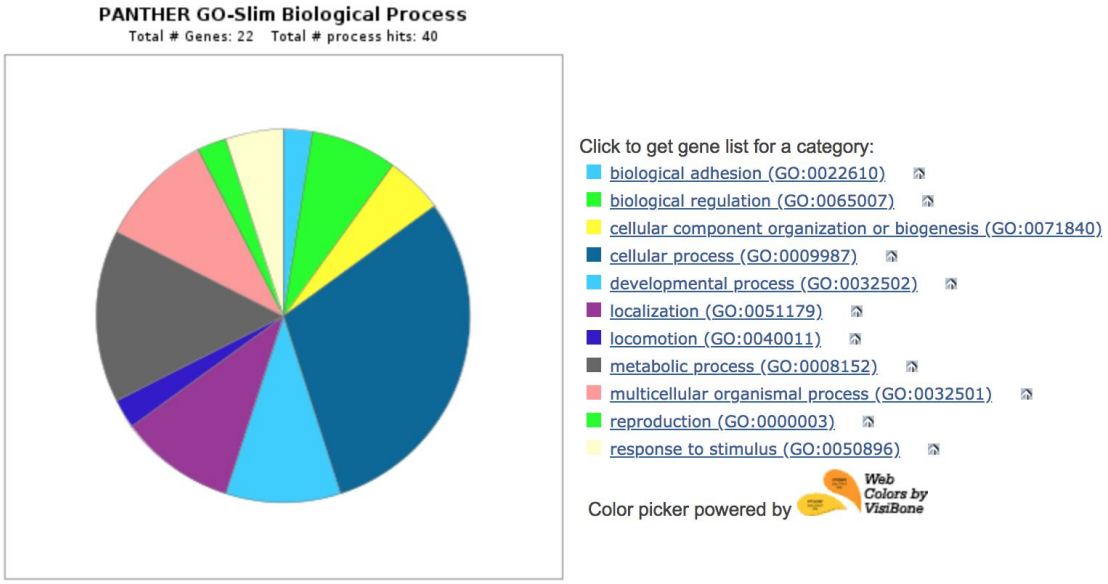
Table 1. Cluster results from using the greedy algorithm on 100 genes with 20 cuts

| Cluster | Genes |
|---|---|
| 1 | *A_32_P93459, ATP5L, CBX2, A_24_P23454, HLA-DPB2, RFPL3, BRI3BP* |
| 2 | *ISM2, BG618521, AY227436* |
| 3 | *LYPD5, ZNF410,COPB1,A_24_P925292, WDR26, ZIC5, BU928689, PRKAG2, UBR1, BM978504, A_24_P759955, CLMP, HLA-DOA, EML3, TMEM50A, ZNF606, HLA-C, AHSA1, RPTOR* |
| 4 | *BX115813, DAAM2, KIF25-AS1, ERICH4, GREM1, MAGEE2, DIAPH3, BCOR, LOC100233156, PPP4R2, PCDHGA3, TPH2, NEAT1, PDIA4, FMN2, DR49, BQ183759, RBFOX2, ZDHHC19, A_24_P920447, B3GNT7, THOC1, SEMA3Cl, OTOP1, MYH4, A_23_P394562, BC070091, SLC29A4, APOL3* |
| 5 | *FCN2, RGPD4-AS1, LOC100127886, A_32_P209924, ZNF329* |
| 6 | *MRPL57* |
| 7 | *LINC01249, CDC42EP4, PTPN22, GTF3C2, HSD3B1, GCDH, SLC5A3, A_24_P712882* |
| 8 | *C5orf22* |
| 9 | *PRKCQ-AS1, A_32_P170436* |
| 10 | *AF085920* |
| 11 | *SLC5A8, FBLN1, A_24_P153324* |
| 12 | *MTURN, A_32_P66072, ZNF304, A_32_P22288, EYA1 LOC389906, PGM5-AS1, PRRG2, LINC00963, LINC00521* |
| 13 | *KCNE5* |
| 14 | *NEAT1. ZDHHC19. A_24_P920447* |
| 15 | *LYNX1, SORL1, BDKRB1* |
| 16 | *OR5P2, HELLS* |

| | |
|---|---|
| 17 | *CBFB* |
| 18 | *A_23_P143047* |
| 19 | *FRYL* |
| 20 | *SCYL2* |

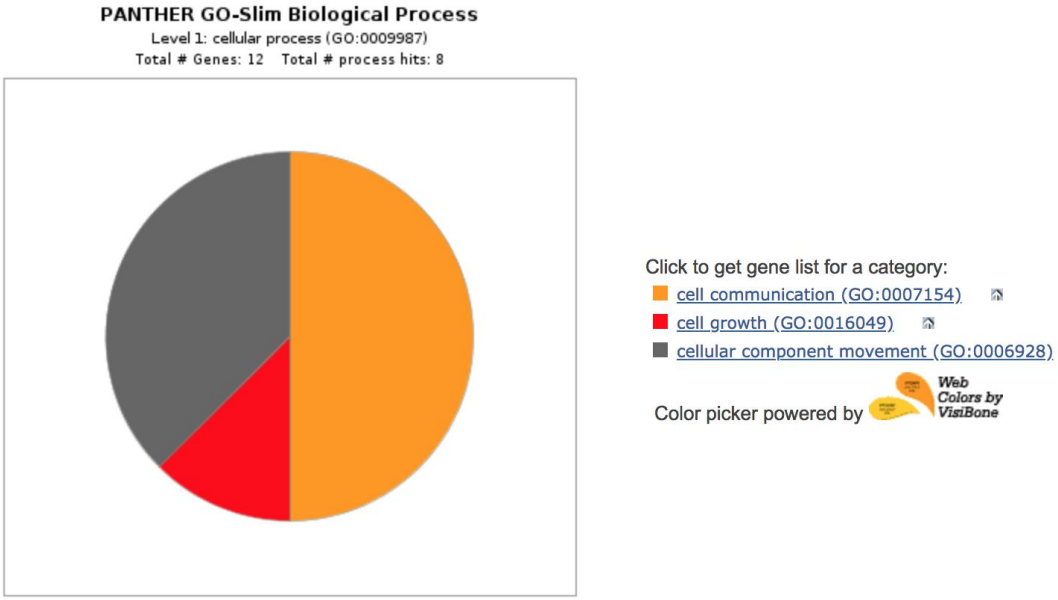
The following chart shows the gene ontology for cluster 4, found using pantherdb.org.

**Figure 3. Cluster 4 Gene Ontology**




This pie chart shows 11 gene ontologies for 22 genes. Though there seem to be many ontologies represented, looking more closely at the ontologies of genes having to do with cellular process, there are fewer gene functions represented. A pie chart of this data is shown below in figure 4. Due to the small data set size, there is a relatively high false discovery rate. Ideally, we would have run this

**Figure 4. Cluster 4 Gene Ontology within Cellular Process**




A runtime analysis of the greedy algorithm and brute force algorithm was completed for varying numbers of nodes. Results are shown below in Figure 5.

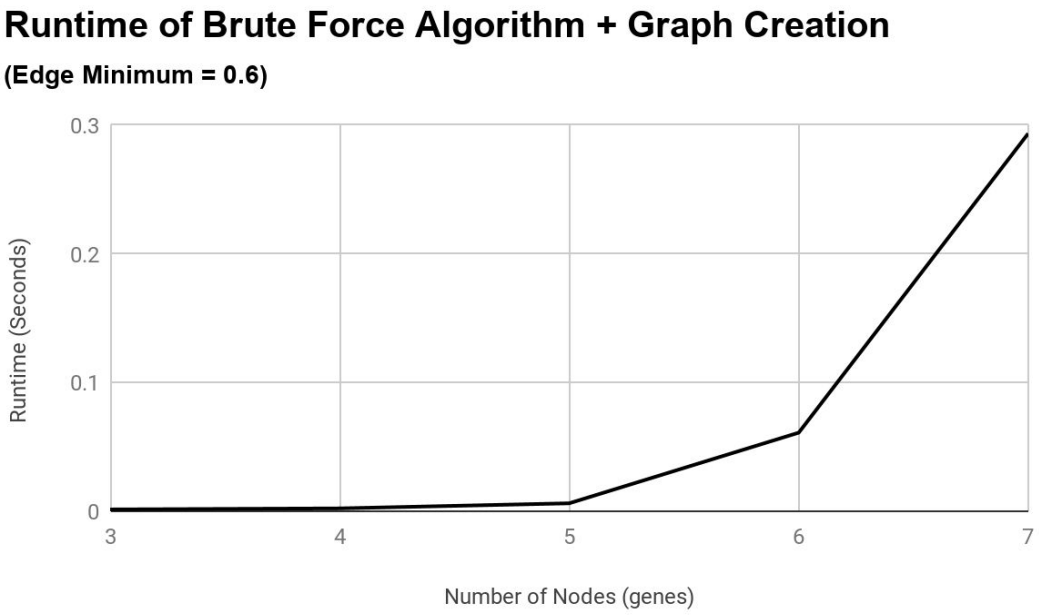



*Figure 5. Runtime of Brute Force Algorithm and Graph Creation, Runtime(n=8) >> 1800 Seconds*

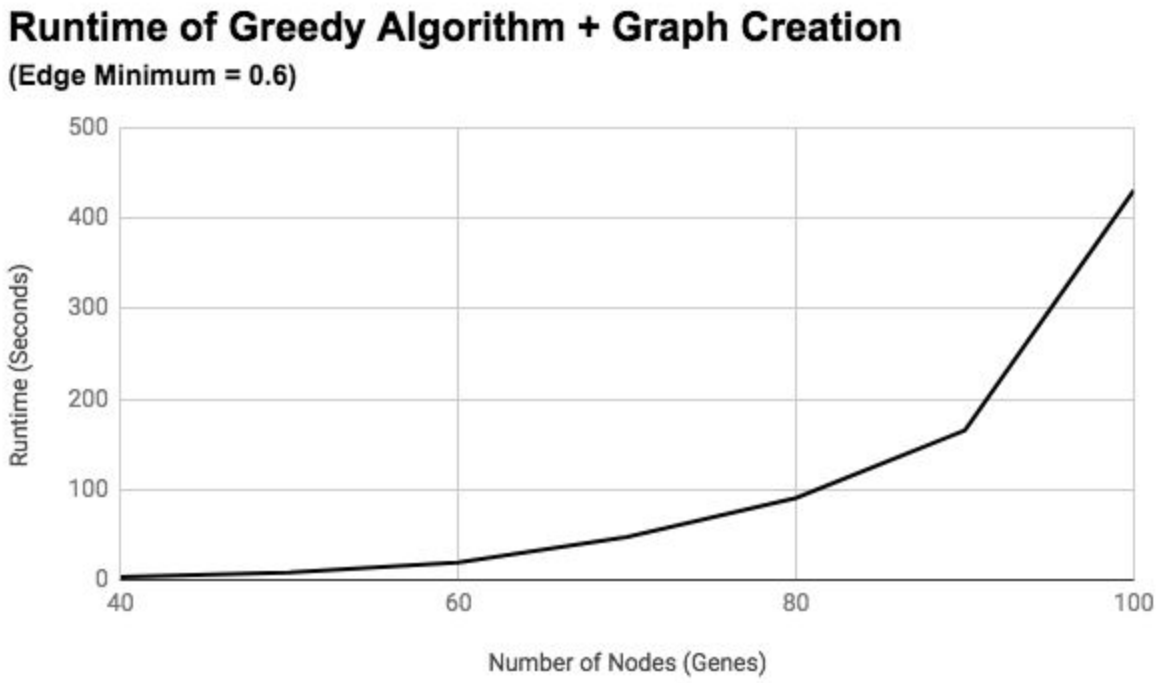



*Figure 6. Runtime of Greedy Algorithm with Graph formation, Runtime(n=150) >> 1800 Seconds*

**Discussion**

In this study, we sought to determine groups of genes that signify regulatory networks in human astrocyte response to the Influenza H5N1. In order to accomplish this, gene expression data from control cells and cells exposed to H5N1 at various time points were used to create a graph model. The graph was used to perform a minimum k-cut, which separates the graph into gene regulatory networks. Our results show that using a minimum k-cut model can result in clustering of biologically related genes. However, they also show that runtime is a large issue, particularly for the brute force algorithm.

There were two approaches taken to solve the minimum k-cut problem, which each had different benefits. The greedy algorithm is relatively faster but is algorithmically more complex. The brute force approach has an extremely slow runtime, making it essentially intractable for large data sets. However, it is algorithmically simple.

Due to time restraints, we were unable to perform our algorithm on data sizes larger than 100 genes. The greedy algorithm could not be run on gene sets larger than 7, making it extremely  However, as a future direction, it would be interesting to run the program for a longer time simply to find more robust clusters of gene regulatory networks.

In our approach, we leave the k value up to the user. However, it would also be interesting to optimize k biologically such that the gene ontology within clusters makes sense. However, this optimization must be balanced to as not to preclude the discovery of novel gene relationships.

Overall, we present an approach to help determine gene regulatory networks in any biological system. This method can be used on any biological system and is not restricted to the problem of astrocyte infection with H5N1 used here.

**References**

1. Lin, Xian, et al. “Insights into Human Astrocyte Response to H5N1 Infection by Microarray Analysis.” Viruses, vol. 7, no. 5, 2015, pp. 2618–2640., doi:10.3390/v7052618.
2. “Figure 2f from: Irimia R, Gottschling M (2016) Taxonomic Revision of Rochefortia Sw. (Ehretiaceae, Boraginales). Biodiversity Data Journal 4: e7720. Https://Doi.org/10.3897/BDJ.4.e7720.” doi:10.3897/bdj.4.e7720.figure2f.
3. Gene Ontology Consortium. “AmiGO 2: Welcome.” AmiGO 2: Term Details for "Aspartic-Type Endopeptidase Activity" (GO:0004190), amigo.geneontology.org/.