Summer 8-4-2011

SEA: a novel computational and GUI software pipeline for detecting activated biological sub-pathways

Thair Judeh
University of New Orleans, tjudeh@uno.edu

SEA: a novel computational and GUI software pipeline for detecting activated biological sub-pathways

A Thesis

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
in
Computer Science
Bioinformatics

by

Thair Judeh

B.S. Loyola University New Orleans, 2005

August, 2011

Acknowledgments

I thank God who gave me the perseverance to complete this thesis and to Whom I owe all good in this life.

Furthermore, I thank my major professor Dr. Dongxiao Zhu whom I hope to one day emulate in his dedication to his work and his advisees. Without a doubt I have greatly benefited from his guidance and expertise. I also thank the other committee members Dr. Adlai DePano and Dr. Christopher Summa for their invaluable advice and stimulating discussions. I also thank my colleague Lipi Acharya with whom I have collaborated with on many interesting research projects. I also thank the Research Institute for Children and Tulane University for the generous funding they have provided in supporting the research that Dr. Zhu and I undertook and the Department of Computer Science at UNO for providing me with an assistantship to support my graduate studies.

A special thanks is entitled to my family. I thank my mother who has always sought to instill into my siblings and I a sense of responsibility. I thank my father who sacrificed greatly to ensure the quality of the education that I received throughout my life. Finally, I thank my beloved wife Honida who has constantly pushed me to excel in my research and in life in general.

Table of Contents

List of Figures

Abbreviations

API Application Programming Interface

BIC Bayesian Information Criterion

BNT Bayes Net Toolbox

COSINE COndition-SpecIfic sub-Network

CPD Conditional Probability Distribution

CPM Clique Percolation Method

CPT Conditional Probability Table

DAG Directed Acyclic Graph

DFS Depth First Search

DNA DeoxyriboNucleic Acid

FTP File Transfer Protocol

GenMAPP Gene Map Annotator and Pathway Profiler

GSGS Gene Set Gibbs Sampler

GUI Graphical User Interface

KEGG Kyoto Encyclopedia of Genes and Genomes

KGML KEGG Markup Language

LPA Linear Path Augmentation

MLE Maximum Likelihood Estimator

mRNA messenger RNA

NCBI National Center for Biotechnology Information

PPI Protein-Protein Interaction

RNA RiboNucleic Acid

SEA Structure Enrichment Analysis

SOAP Simple Object Access Protocol

TPM Transitional Probability Matrix

WSDL Web Service Definition Language

XML Extensible Markup Language

Abstract

With the ever increasing amount of high-throughput molecular profile data, biologists need versatile tools to enable them to quickly and succinctly analyze their data. Furthermore, pathway databases have grown increasingly robust with the KEGG database at the forefront. Previous tools have color-coded the genes on different pathways using differential expression analysis. Unfortunately, they do not adequately capture the relationships of the genes amongst one another. Structure Enrichment Analysis (SEA) thus seeks to take biological analysis to the next level. SEA accomplishes this goal by highlighting for users the sub-pathways of a biological pathways that best correspond to their molecular profile data in an easy to use GUI interface.

Network Partitioning, Network Reconstruction, Structure Enrichment Analysis, Community Detection Algorithms, Biological Networks, KEGG

Chapter 1: Background and Introduction

The world of biological systems is a vast and complex system of regulation processes and biomolecular interactions. An underlying goal for biologists is to arrive at a theory that shines light on the complicated interaction patterns in living organisms. These interaction patterns result in various biological phenomena where recognition of these patterns can provide much needed insight into biomolecular activities. Capturing these biomolecular activities, however, is a daunting task due to the complexity of the systems at hand as well as lacking of data needed to fully capture the underlying biomolecular activities. Thus, two problems have recently received a considerable amount of attention: (1) inferring biological pathway structures from gene expression data and gene sets and (2) decomposing different biological pathway structures into functional units.

A revolution in the understanding of biomolecular interaction mechanisms has occurred in large part due to the rapid and significant advances in high-throughput technologies. Such technologies, such as microarrays and second-generation sequencing, now enable a systematic study of biomolecular activities due to the copious amount of genome-wide measurements. These genome-wide measurements continue to be accumulated into numerous databases by research labs across the world. Unfortunately, gaining biological insights from large-scale gene expression data is a daunting task due to the *curse of dimensionality*. To overcome this task, many computational and experimental models have been developed to group genes into various sets based on either a structural or functional similarity. This lead to the birth of *gene sets* as a new source of data leading to a burst in novel algorithms that infer biological pathway structures from gene sets. These two types of data, gene expression data and gene sets, will now be examined in more detail.

First, gene expression data is represented as a matrix of numerical values. Each row corresponds to a gene while each columns corresponds to an experiment. Each entry of the matrix corresponds to the gene expression level for a given gene under a given experiment. Gene expression profiling has thus allowed the simultaneous measurement of the expression levels of thousands of genes. A systematic study of biomolecular interaction mechanisms is now possible on a genomic scale. One typical example of gene expression data is microarray data. For microarray data one usually has a glass slide that is coated with oligonucleotides corresponding to specific gene coding regions. The slide is then labeled and hybridized with purified RNA . A laser is scanned on the washed microarray slide to obtain gene expression data.

Ways to obtain genome-wide measurements have also grown. There are a wide array of microarray platforms, and genome-wide measurements can be obtained via conventional hybridization based microarray [14, 20, 31] or deep sequencing experiments [32, 33]. Some representative microarray platforms include Agilent Microarray, Affymetrix GeneChip, and Illumina BeadArray.

Moving on to gene sets, gene sets are defined as a group of genes that share biological similarities. They are a rich source of data for reconstructing the structure of biological pathways as they tend to participate in the same biological process. Gene sets are derived from a variety of sources including PubMed text, ChIP-chip, co-localization along the a chromosome, and gene expression data. There are a variety of methods to rank gene sets with GSEA-P [34] being one of the most popular methods. A major advantage of working with gene sets is their capability to incorporate with ease higher-order interaction patterns. They are also more robust to noise than gene expression data and are capable of integrating data from a variety of sources. Given the ways a gene set may be derived, one must keep in mind the possibility that not all gene sets may represent network structures.

An important underlying assumption when trying to reconstruct a biological pathway structure using gene sets or gene expression data is that these sets of data were originally emitted from unobserved signaling pathways. There are various algorithms based on this assumption that attempt to reconstruct the structure of biological pathways using gene sets and/or gene expression data.. First, a biological pathway structure is a graph $G(V, E)$ where V is the set of vertices or nodes. E is the set of edges. In the case of biological pathways, a vertex $v \epsilon V$ may either be a gene or protein whereas an edge $e \epsilon E$ joining two such vertices represents the biological properties connecting them. The final underlying network may either be directed or undirected, and both types of networks occur naturally in biological systems.

For example, a signal transduction is a typical example of a directed network in biological systems. According to the Central Dogma of Molecular Biology, DNA encodes the genetic information of living organisms. DNA directs protein synthesis via the formation of messenger RNA (mRNA) [4]. A signal transduction is thus the primary means that decodes DNA into mRNA and then into protein synthesis. For a signal transduction to occur, cytokines or chemokines bind to the transmembrane proteins which in turn activates a sequential activation of signal molecules leading to a biological end-point. In this case a directed edge represents one event in a signal transduction activating another, and a signaling pathway is thus composed of a web of gene regulatory wiring or different transduction events.

Undirected networks, on the other hand, are typically exemplified by Protein-Protein Interaction (PPI) networks [35]. These networks have no self-loops, and all vertices consist of proteins. An edge exists between two proteins if they can physically interact.

Once a biological pathway structure has been reconstructed, one needs to examine it at a finer level as usually only part of a biological pathway structure is involved in a biological process of interest. Thus, decomposing different biological pathway structures into sub-pathways is a must. By retrieving the sub-pathways, one is able to accomplish two major

goals: predict gene functionality and relevant sub-pathways for different phenotypes. For example, if gene A is clustered with other genes responsible for apoptosis, one may infer that gene A also plays a role in apoptosis. This leads to predicting a new gene functionality for gene A that may have been previously unknown. As another example, one may possess cancer molecular profile data. By "enriching" the sub-pathways, one may extract new biological insights about the sub-pathways most relevant to cancer. Figure 1.1 succinctly summarizes the relationships amongst the various topics discussed in this introduction.

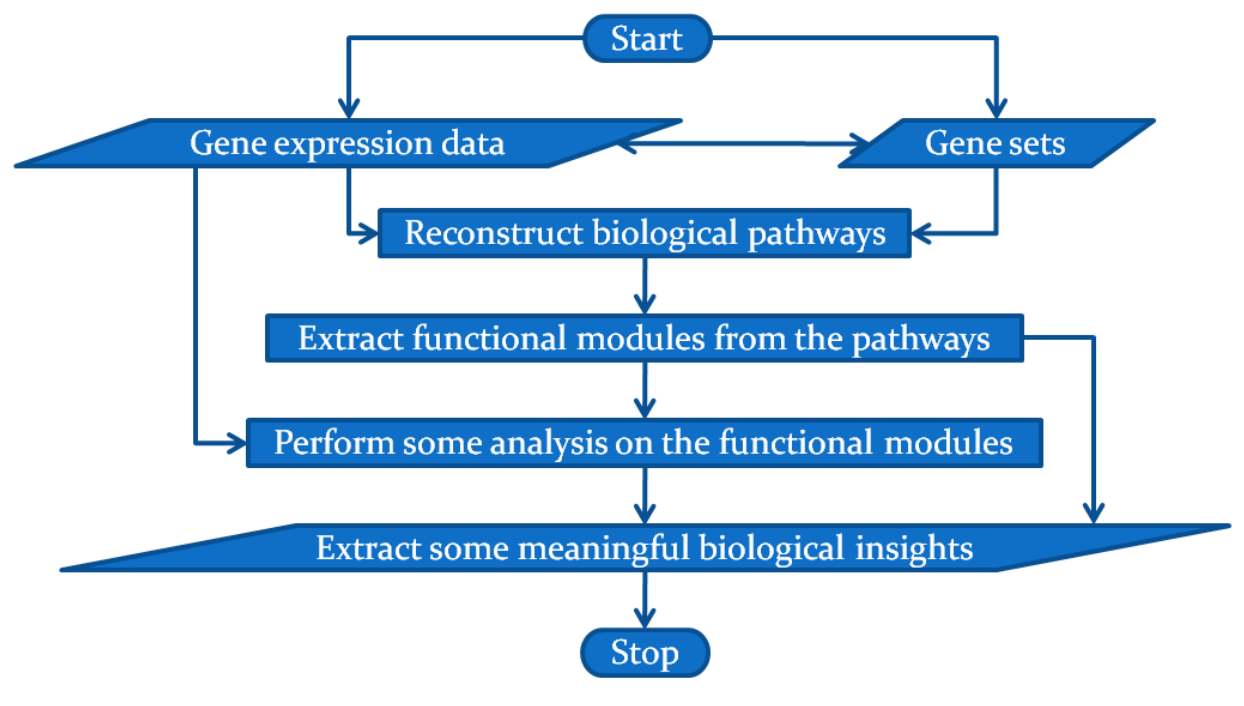


Figure 1.1: The big picture. Gene expression data and gene sets may be converted from one to another. Furthermore, given gene expression data or gene sets, one can reconstruct different biological pathway structures. Given that only a sub-pathway is usually activated for a particular biological process, decomposing a biological pathway structure into sub-pathways is a must. From these sub-pathways, one may extract useful biological insights. Otherwise, one may use molecular profile data in conjunction with sub-pathways to extract the most relevant sub-pathways for the data at hand.

To outline the remainder of this thesis, three areas will now be examined in more detail. Chapter 2 will examine three network reconstruction algorithms. The first approach

is Bayesian networks [24, 12], which is an approach based on gene expression data. The second approach is the Frequency Method [29], which is a gene set based approach. The final approach is Linear Path Augmentation (LPA) [15], which is an *original* contribution to the field. Chapter 3 will examine three network partitioning algorithms including the Kernighan-Lin algorithm [19], the Girvan-Newman algorithm [13, 26], and the Clique Percolation Method (CPM) [27, 28]. Finally, for Chapter 4 the focus will be on an *original* and novel software pipeline, SEA (Structure Enrichment Analysis), which closely resembles Figure 1.1.

Chapter 2: Network Reconstruction

Given gene expression data and gene sets, it is often the case that more biological insight needs to be extracted from them. One concise manner to extract data from gene expression data and gene sets is to reconstruct a biological pathway structure. Reconstructing a biological pathway structure is a key step as it is often the gateway for further analysis. For example, it may be a difficult task to accurately extract signal cascades if the underlying network is unknown. A biological pathway structure can also illustrate how various sub-pathways cross-talk within one another. Thus, there are a plethora of reasons to reconstruct a biological pathway structure.

There are a variety of methods to reconstruct biological pathways. Some methods, such as Bayesian networks, rely on gene expression data. Other methods, such as Frequency Method, rely on gene sets. Both of these methods will be examined later on in the chapter. In addition, an *original* and novel algorithm, Linear Path Augmentation (LPA), will be presented in detail later on in this chapter as well.

2.1 Bayesian Networks

A Bayesian network [24, 12] is a graphical model that ties with its vertices some probabilistic relationships. From a network structural view, a Bayesian network embodies the conditional dependencies and indepedencies of its various vertices. It also efficiently encodes the joint probability distribution of all the vertices in the graph. A Bayesian network is represented by a DAG (directed acyclic graph), which automatically rules out Bayesian networks from representing feed-back loops and other cyclic structures.

A Bayesian network consists of a pair (G, Θ) where G represents a DAG. The $|V| = n$ nodes of G are random variables $X_1, X_2, ..., X_n$ that may represent discrete or continuous

random variables. Θ denotes the set of parameters for each of the random variables and is needed to encode a random variable's CPD (conditional probability distribution) or CPT (conditional probability table) depending on whether it is discrete or continuous. More formally, one can define Θ as

$$\Theta_{x_i|pa(x_i)} = P(x_i|pa(x_i)) \tag{2.1}$$

$\forall\ x_i\ \epsilon\ X_i$ given the set of parents of x_i in G. Θ is often learned by assuming some underlying distribution and using gene expression data to derive Θ. Using the factorization definition, one can express the joint probability distribution as a product of the conditional probabilities

$$P(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} P(x_i|pa(x_i)). \tag{2.2}$$

Using Bayesian networks often consist of using a structure learning algorithm that consists of two major components: *searching* for "good" structures and then *scoring* them. It is necessary to employ a heuristic to search for structures since the search space is super-exponential rendering an exhaustive search to be implausible. For these types of problems, a *greedy* algorithm is a natural choice where one begins with either a full network or empty network. One then adds, deletes, or reverses an edge until a local maximum is reached. One may also employ simulated annealing to aid for the search of a global solution.

As will be seen in Chapter 4, it may be the case that the structures of interest are already available. Thus, one may venture to say that *scoring* structures may ultimately be more important than searching. Often times an approximation may be used such as the Bayesian Information Criterion (BIC) defined as $\ln p(D|\hat{\theta}_G, G) - \frac{d}{2} \ln N$ where D is the dataset, G is the structure, d is the number of parameters, and N is the size of the dataset. $\hat{\theta}_S$ is an estimate of the model parameters, and for large enough N, one may use the MLE .

Thus, a Bayesian network is a good probabilistic modeling approach to learn the structure of a biological pathway from gene expression data. They are also quite robust

against noisy data, which in turn prevents over-fitting of the data. Its main disadvantages lie in its computational complexity and its restriction to DAGs. Regardless, Bayesian networks are still quite popular in many fields, and many implementations, such as BNT [23], exist that allow users to harness their power.

2.2 Frequency Method

The Frequency Method [29] is a method to reconstruct directed networks from gene sets. It makes three important assumptions about the gene sets. First, it assumes that tree structures in the paths correspond to gene sets. Another assumption is the availability of the source and destination of each gene set, which may not necessarily be known for all biological systems. Finally, it is assumed that the directed edges used to form a tree in each gene set are already available, but their order is unknown.

Using terminology similar to [2], let S be the set of source nodes, D be the set of destination or target nodes, and E is the collection of all directed edges of the graph. Each member $m \epsilon S \cup D \cup E$ can be associated with a binary vector of length N, the number of gene sets, where $x_m(i) = 1$ indicates that m is involved with i^{th} gene set. By letting s_i be the fixed beginning of the i^{th} gene set and d_i its destination, the order of genes for the i^{th} gene set is found by satisfying

$$e^* = arg \max_{e \epsilon E} \lambda_i(e) \tag{2.3}$$

where $\lambda_i(e)$ is defined as

$$\lambda_i(e) = x_{s_i}^T x_e - x_{d_i}^T x_e \tag{2.4}$$

$\forall\ e\ \epsilon\ E$ with $x_e(i) = 1$. It should be noted that $\lambda_i(e)$ is used to determine whether e is closer to its source s_i than its destination d_i. The result of Equation 2.3 is that e^* is placed closet to s_i. Thus, the edges are placed in proximity to s_i based on their λ scores.

The Frequency Method leads to a unique solution in reconstructing the biological pathway structure and is computationally efficient. A major drawback is the stringent assumptions made by it such as knowing the source and destination genes of each gene set. Furthermore, if there exist multiple paths between a pair of genes, the Frequency Method may fail.

2.3 LPA

LPA (Linear Path Augmentation) [15] is an *original* and novel network reconstruction algorithm. The goal of LPA is to reconstruct an original biological pathway structure using gene sets as the input. The underlying hypothesis of LPA is that gene sets correspond to signal cascades and that the underlying network corresponds to a DAG (Directed Acyclic Graph). With these assumptions LPA has a robust pipeline to reconstruct biological pathways using gene sets as input. Figure 2.1 provides an overview of the problem that LPA attempts to solve.

Before proceeding to the details of LPA, it is prudent to describe how simulations were conducted. To be able to test LPA as well as other algorithms, it is necessary to be able to generate some linear paths from the original network. To accomplish this goal, the algorithm *All Linear Paths* was developed. It is important to note that for a fully connected DAG, there are $\sum_{j=1}^{n-1}\sum_{i=1}^{j-1}\binom{j}{i}$ linear paths where n is the number of vertices in the DAG. Thus, this algorithm is only feasible for very sparse pathways. Figure 2.2 presents a flow chart describing the *All Linear Paths* algorithm.

A very significant step that can easily be overlooked is permuting the order of the gene sets at the very end. It is natural for algorithms to handle gene sets one at a time. An issue that arises, though, occurs if some assumption or calculation is made using the remaining gene sets. One example is GSGS(Gene Set Gibbs Sampler) by [1]. In particular,

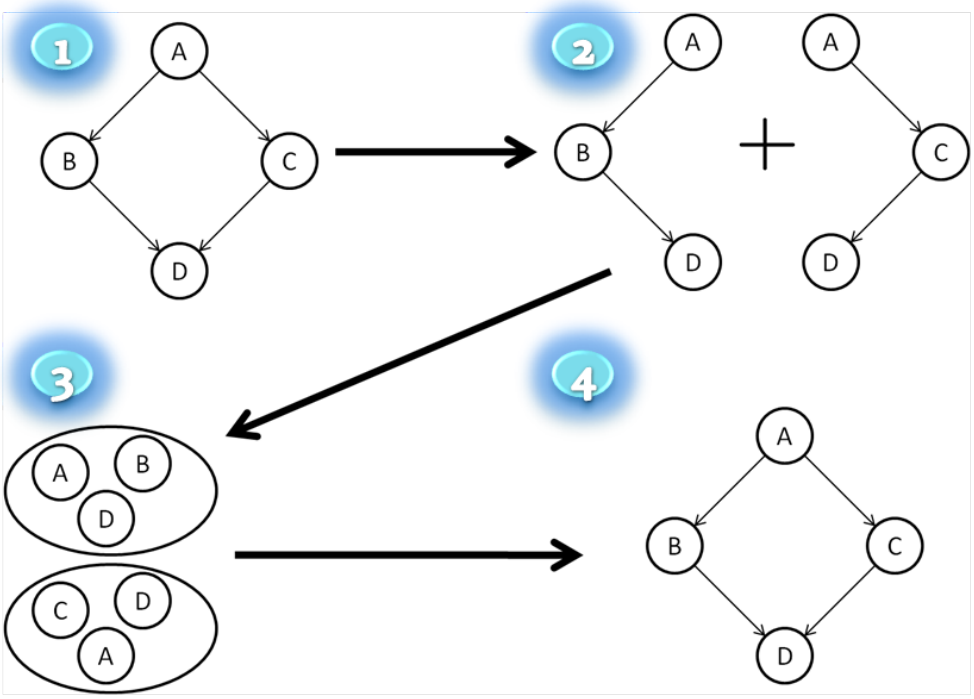


Figure 2.1: This sample network illustrates the problem that LPA attempts to solve. At step 1, one has the original, unobserved biological pathway. At step 2 the pathway consists of signal cascades. Unordered gene sets corresponding to the signal cascades are represented at step 3. Finally, using a network reconstruction algorithm, the original biological pathway is reconstructed from the gene sets in step 3. This original author contribution first appeared in [15].

the remaining gene sets in GSGS are used to calculate the TPM (Transitional Probability Matrix). It is hoped that with a good number of gene sets this effect is diminished as the weight of a single gene set in calculating the TPM is reduced. Similarly, for LPA the remaining gene sets play a significant role in the score function to be discussed later on in subsection 2.3.3. For both cases mentioned, the order of the gene sets may affect the final results with LPA being affected far more significantly than GSGS.

One important note is that any network and its transpose can produce the same set of linear paths. Any algorithm that does network reconstruction must always keep this fact in mind. At least for biological networks, though, this problem is somewhat mitigated as biologists should usually be able to easily tell the proper matrix. For example, biologists

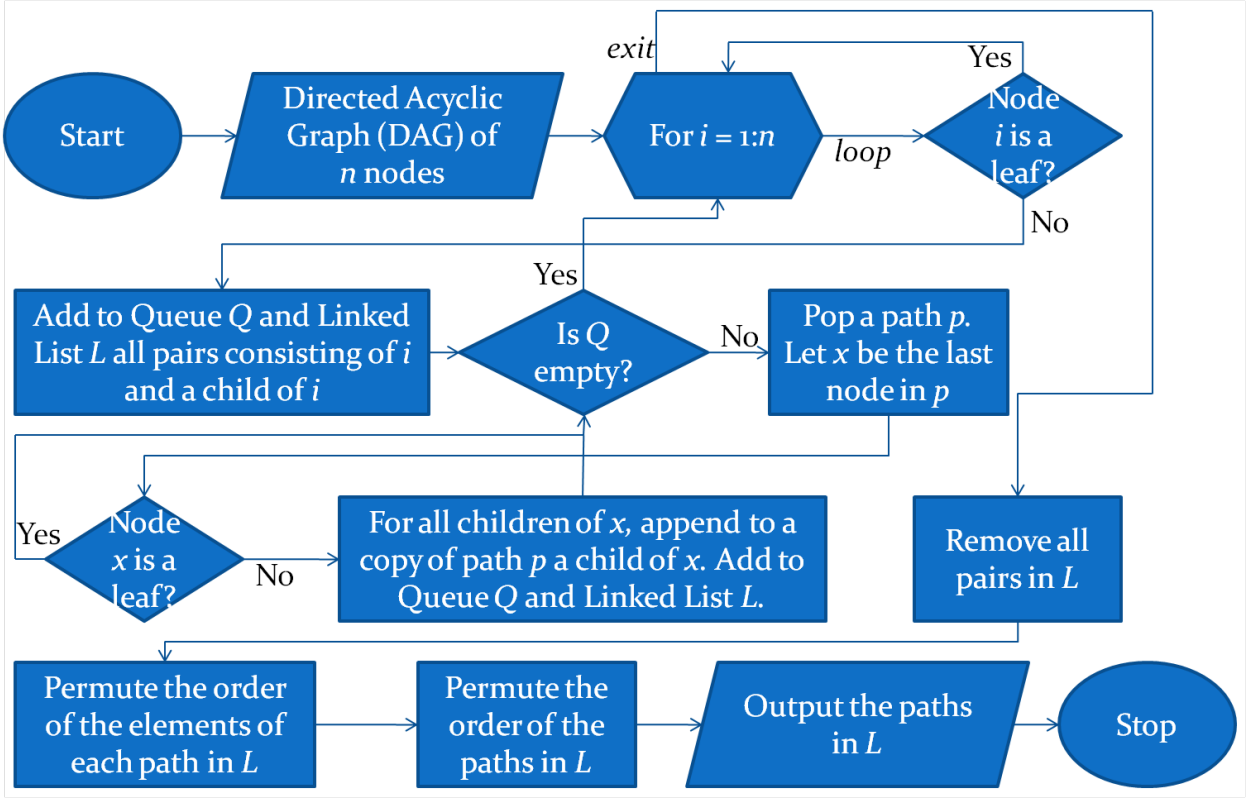


Figure 2.2: All Linear Paths

would not label a transcription factor as a leaf node. Thus, from an algorithmic perspective, some prior knowledge is a must.

The final step needed for simulation studies are some gold standard networks. The gold standard networks chosen are from the DREAM3 Network Challenges [22]. Furthermore, the chosen networks are all DAGs and small-scale as well. Table 2.1 lists a set of networks from the DREAM3 Network Challenges as well as some useful statistic per network. Results of the LPA algorithm are also displayed.

The LPA algorithm itself is a novel combination of a variety of techniques. Its name, Linear Path Augmentation, is based on augmenting matrices with linear paths. Based on the available knowledge, no other algorithm functions in a manner similar to it. In addition to its novelty, it is quite modular consisting of *preprocessing*, *sorting*, *growth*, *pruning*, and

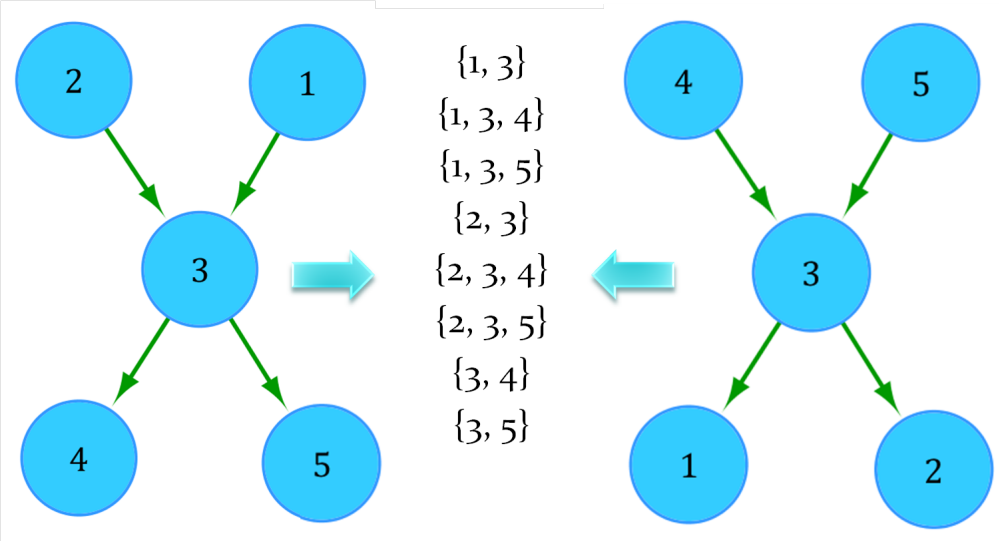


Figure 2.3: A network and its transpose. By running the All Linear Paths algorithm detailed in Figure 2.2 on both networks, the same set of gene sets is produced. In essence, this states that without any prior information a network and its transpose are both equal in terms of finding the final network.

Network	#Nodes	#Edges	Network Diameter	#Paths	#Paths – #Pairs	# of nodes in longest path	Sensitivity	Specificity	PPV	Time
E. coli 1	50	62	4	187	125	5	91.43%	100%	100%	3.08s
E. coli 2	100	125	4	266	141	5	67.19%	100%	100%	168.26s
E. coli 3	100	119	3	233	114	5	69.81%	100%	100%	65.77s

Table 2.1: Statistics concerning E. coli networks from the DREAM3 Network Challenges. Also displayed are the results of the LPA algorithm where Sensitivity $= \frac{TP}{TP+FN}$. Specificity $= \frac{TN}{TN+FP}$. Positive Predictive Value (PPV) $= \frac{TP}{TP+FP}$. TP equals true positives, FP equals false positives, TN equals true negatives, and FN equals false negatives.

intersection stages. This modularity allows for ease of updating stages individually. Figure 2.4 presents a high-level flow chart of the LPA algorithm.

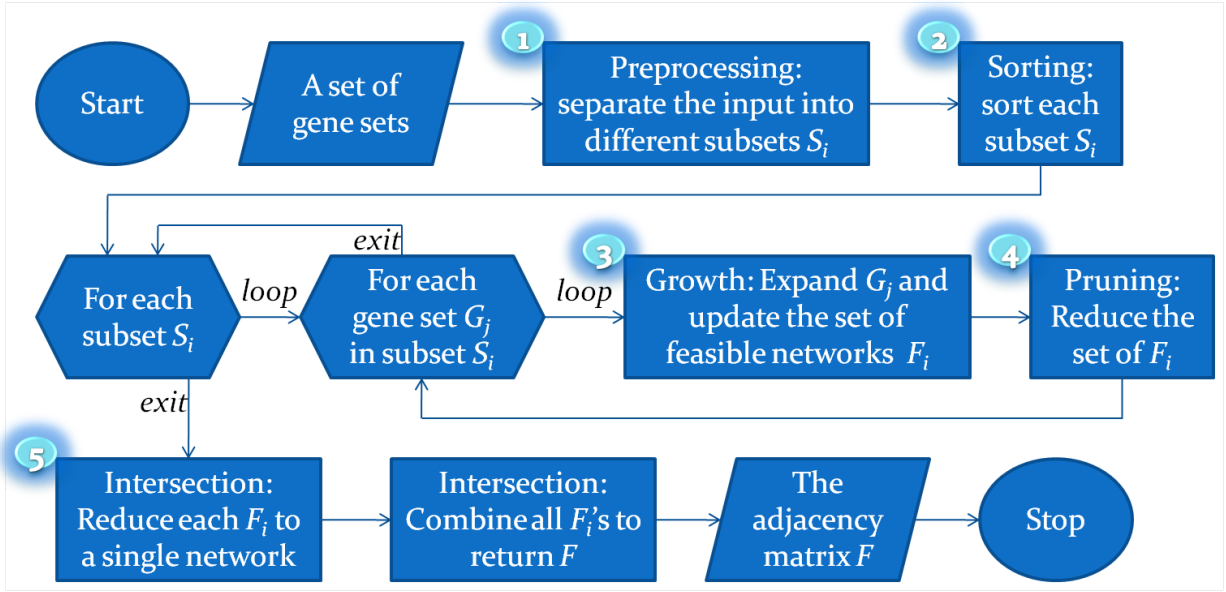


Figure 2.4: The LPA algorithm consists of five key stages. The first stage, *preprocessing*, separates the gene sets into components. The second stage, *sorting*, places the gene sets in order. The third stage, *growth*, *searches* for candidate networks. The fourth stage, *pruning*, *scores* the candidate solutions and removes candidate solutions with low score. The final and fifth stage, *intersection*, is needed in the absence of prior data to reconcile any candidate solutions still left.

2.3.1 Preprocessing

The idea behind the preprocessing stage is to divide the gene sets into "components." The process is relatively straightforward. If two gene sets A and B share at least one node, they are placed in the same component. If gene set C shares at least one node with either gene set A or B, it is also placed in the same component. If the original network is a single connected component, than all gene sets will fall into one component. Similarly, if the original network had k disconnected components, then there will be k sets of gene sets. For all scenarios listed, it is assumed that no gene sets are missing so the number of sets of gene sets in practice may vary. This allows for a divide and conquer approach where the next steps are run k times, once for each set of gene sets.

2.3.2 Sorting

This stage assigns an order for a set of gene sets. The LPA algorithm is very sensitive to the order of the gene sets. The order of the gene sets can actually determine whether the algorithm converges to the correct solution and may have a direct affect on its computational complexity. The current approach places the longest gene sets first. While this increases the computational complexity of the algorithm, it makes it more likely to converge to the correct solution.

2.3.3 Growth

The growth stage is very akin to the "searching" stage of a structure learning algorithm. For the first iteration, assuming no prior knowledge has been provided, $\frac{length(G_1)!}{2}$ networks are constructed where G_1 is the first gene set. Each network corresponds to one linear path from the $\frac{length(G_1)!}{2}$ possible permutations. The quantity is divided by two as the reverse of the permutations are automatically discarded (Figure 2.3). These networks are stored in a set of candidate networks F_i^1. After the pruning stage, one now begins with the pruned $F_i^{1'}$. Each network in $F_i^{1'}$ is expanded using $\frac{length(G_2)!}{2}$ permutations for G_2. However, to reduce the search space, the topological sort order of each network is taken into account. Thus, only permutations that do not violate its topological sort order are added. For example, if a pathway P consists of the linear path $1 \rightarrow 2 \rightarrow 3$ and the new gene set is $\{2, 3, 4\}$, $3 \rightarrow 2 \rightarrow 4$ will not be added as it violates the topological sort order. $\{2 \rightarrow 3 \rightarrow 4, 2 \rightarrow 4 \rightarrow 3, ...\}$, on the other hand, are valid permutations, and P will split into new networks accordingly. The new augmented networks are then added to F_i^2 while the networks in $F_i^{1'}$ are discarded. The process repeats itself until all gene sets are used and is illustrated in Figure 2.5.

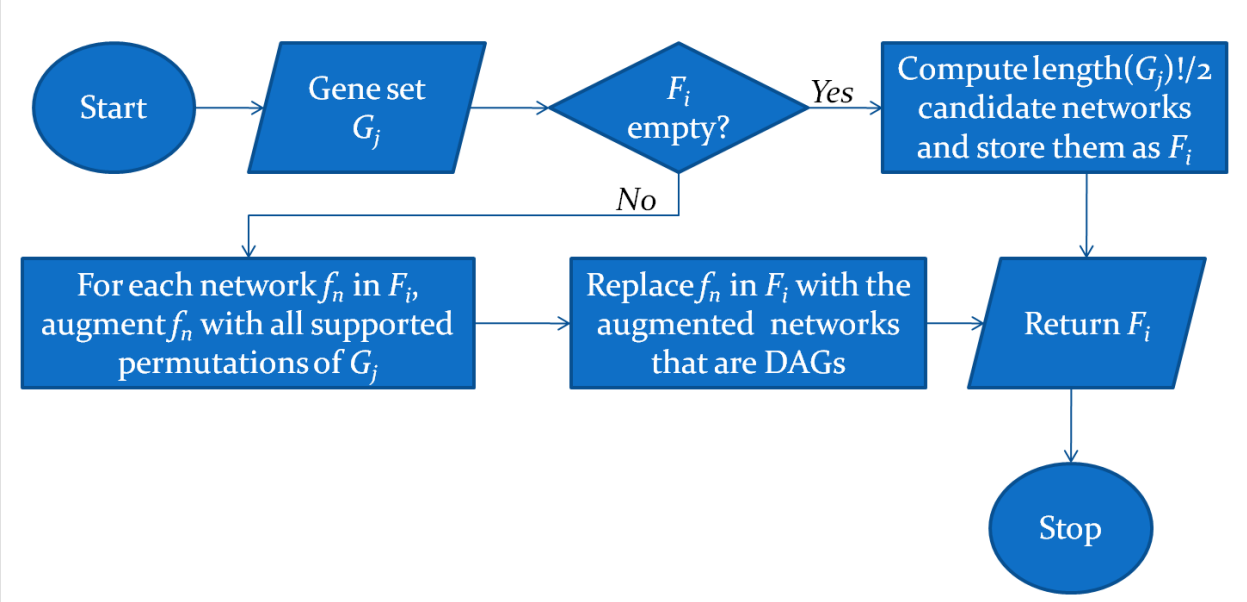


Figure 2.5: Growth Stage.

2.3.4 Pruning

The pruning stage is very akin to the "scoring" stage of a structure learning algorithm. This stage attempts to reduce even further the set of candidate solutions. An important part of this stage is that it uses all gene sets to compute a score for each network. In its essence, this score measures how many gene sets that the underlying network can support. In other words, if one were to run the All Linear Paths algorithm on the network, its score consists of the intersection of its unordered linear paths with the gene sets. Figure 2.6 provides further details on the pruning stage.

2.3.5 Intersection

The final stage is needed only when there still remain some candidate network solutions. Thus, the final network returned is the intersection of all remaining candidate network solutions. In the absence of prior knowledge, one must choose between a network and its transpose. An ad hoc solution at the moment is to choose the network whose upper

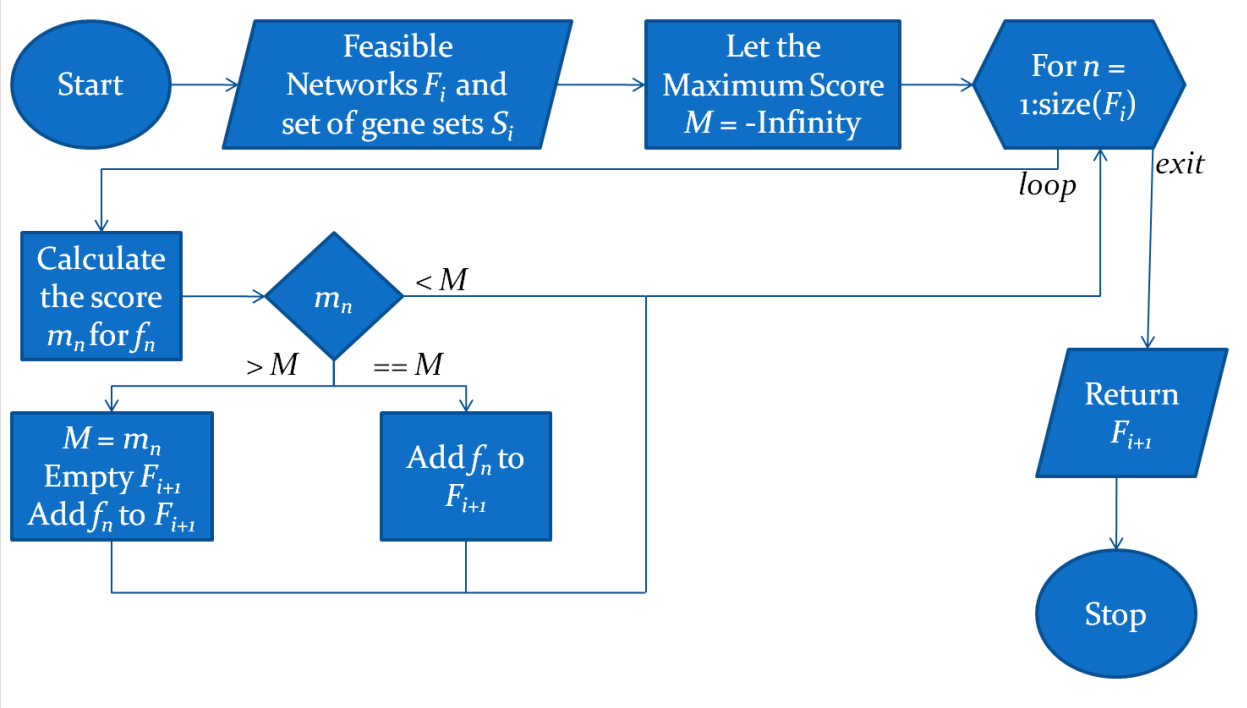


Figure 2.6: Pruning Stage.

triangular matrix is heavier. Naturally, this process may fail when the upper triangular and lower triangular matrices have an equal number of edges. Figure 2.7 provides an example of the intersection stage.

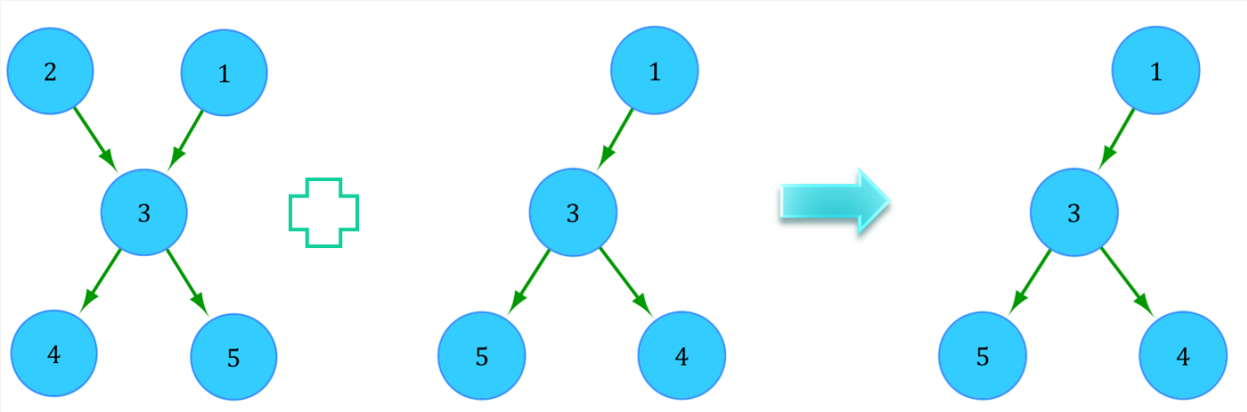


Figure 2.7: Intersection Stage.

A post-processing step is the combination of the separate components, if any, produced by the algorithm. At this stage, the presence of prior knowledge is a must as a

network and its transpose are equally likely in the absence of prior knowledge. After this step is finished, the final network is ready for presentation to the user.

LPA has some novel contributions. At this stage, though, it needs a better sorting, growth, and pruning stages for it be computationally feasible. Given its modular nature, though, it is hoped that finding improvements for these stages will be an achievable task.

Chapter 3: Network Partitioning

It is often the case that a reconstructed network is too broad of a representation for a process of interest. Furthermore, there are now readily available high fidelity biological networks with the Kyoto Encyclopedia of Genes and Genomes (KEGG) [16, 18, 17] being at the forefront of the databases. Since not all of a biological pathway structure is activated at once, a finer level of detail is needed when examining the structure of biological pathways. As such, decomposing a biological pathway structure into sub-pathways is of utmost importance as they may provide valuable insight into various biological processes.

It is vital to first define what a sub-pathway is. For biological pathways the concept sub-pathway is very similar to the concept of communities in social networks. A *community* is a subgraph of a given graph such that (1) the connections within the community from node to node are strong and (2) the external connections between other communities are few and weak. Figure 3.1 provides an illustration of the concept of communities.

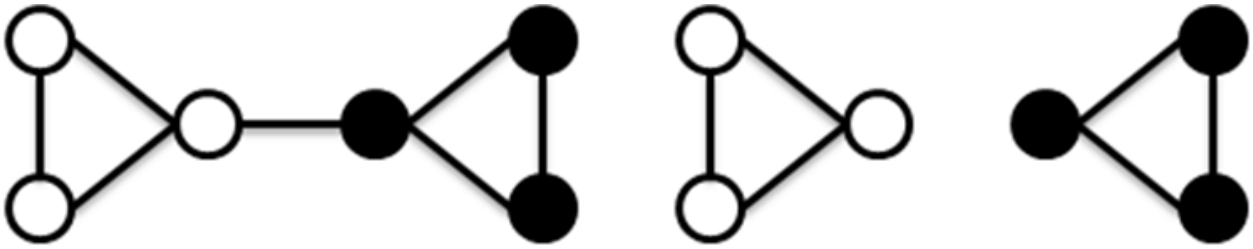


Figure 3.1: The network displayed consists of two communities shaded white and black, respectively. Both communities exhibit high internal connections. Furthermore, the connections between the two communities consists only of a single edge. This original author contribution is set to also appear in [2].

There are two approaches for finding the sub-pathways of a biological pathway structure or graph, namely *graph clustering* and *community detection* algorithms [25]. The former type of algorithms have their origin in computer science and other related fields. The latter

type of algorithms were originally used by sociologists. They now encompass algorithms in applied mathematics, physics, and biology.

For graph clustering algorithms, a user must specify the number of clusters or partitions. A graph clustering algorithm will always return the specified number of partitions regardless of whether the underlying graph is partitionable. These algorithms were designed with specific applications in mind. Some applications include improving the paging properties of programs and placing the components of an electronic circuit onto printed circuit cards [19].

One may ask, "Why study *graph clustering* algorithms for biological pathways?" This is indeed a pertinent question. The major reason is that these algorithms often serve as an inspiration for community detection algorithms. For example, the Laplacian matrix whose use is popular in graph clustering algorithms can be modified to perform eigenvector decomposition [25]. Another example can also be found in Newman's eigenvector method [25]. In this paper Newman used the Kernighan-Lin algorithm [19] as inspiration for a post-processing algorithm, namely Algorithm 2.

Concerning community detection algorithms, the underlying assumption behind these algorithms is that a network or graph can "naturally" be divided into sub-pathways or communities. Thus, the sub-pathways of a graph can be viewed as a topological property of the graph. This design philosophy is a major difference between community detection and graph clustering algorithms.

Before discussing some algorithms in detail, it is prudent to discuss the nature of these algorithms. Most algorithms in this field work for undirected networks and produce mutually exclusive partitions. It is often far from trivial to extend the undirected version of an algorithm to work for directed networks [10]. It is often the case that an algorithm that works only for undirected graphs is simply applied to directed graphs by ignoring the edge direction in the directed graphs. As seen in Figure 3.2, this approach is far from adequate.

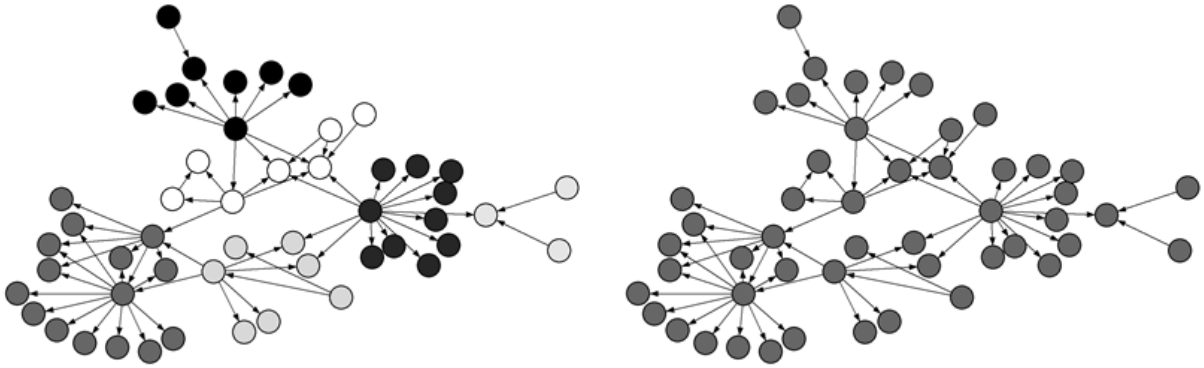
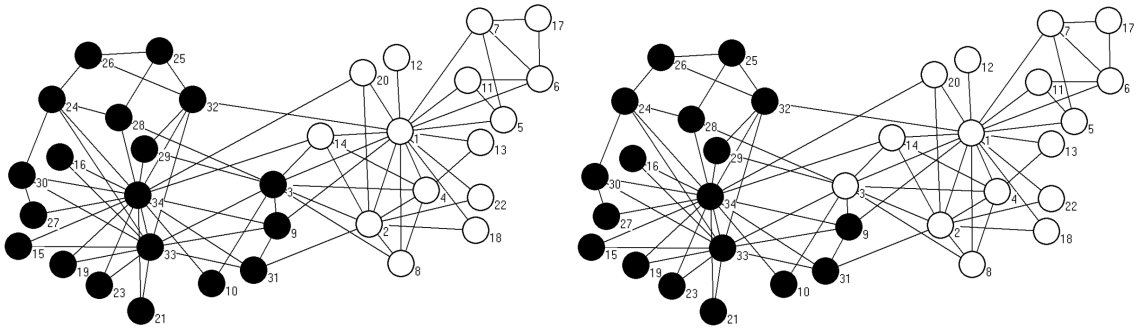


Figure 3.2: An E. coli network from the DREAM3 Network Challenges [22]. (Left) The six communities of the network ignoring edge direction. (Right) Taking edge direction into account, no communities could be found. In both cases, the appropriate version of InfoMap [30] was run for 100,000 iterations. This original author contribution is set to also appear in [2].

As with the network reconstruction algorithms outlined earlier, it is very helpful to have some gold standard networks to compare different algorithms. What constitutes a gold standard network is an area of research itself. For illustration purposes Zachary's karate club [36] has often been used as a "gold standard" network. This social network has in its origin the relationships amongst 34 karate club members. A disagreement arose between the club's administrator and the instructor with the latter splintering off to form a new club as seen in Figure 3.3.

The remainder of this chapter will now be outlined. First, the Kernighan-Lin algorithm [19] will be discussed to provide a flavor for graph clustering algorithms. This discussion will be followed by an examination of the Girvan-Newman Algorithm [13, 26], a very popular community detection algorithm. Finally, the Clique Percolation Method (CPM) [27] will be discussed. Compared to the previous two algorithms, CPM has a version that works with directed networks and also produces nonexclusive sub-pathways.



(a) The true partitioning (b) Girvan-Newman partition

Figure 3.3: (Left) The true partitioning of Zachary's karate club. (Right) The partitioning as returned by the Girvan-Newman algorithm [13] which mislabels a single node. This original author contribution is set to also appear in [2].

3.4 Kernighan-Lin Algorithm

Developed in the 1970s, the Kernighan-Lin algorithm is a well-known graph clustering algorithm. Given its applicability it is often used as a subroutine for other algorithms. It was initially developed in order to divide electronic circuits on boards. The connections between the various circuits were quite expensive. Minimizing the number of connections between the various circuits is a key goal. Formally, the Kernighan-Lin algorithm is a heuristic method that sought to solve the following combinatorics problem: provided a weighted graph G, divide the vertices in V into k partitions such that no partition is larger than a user-specified m. The objective function is that to minimize the total weight of the edges connecting the k partitions.

The algorithm itself seeks to divide a network into two subnetworks. If more clusters are needed, the algorithm may be applied in a recursive fashion. To begin one has an undirected graph G of size $|V| = n_1 + n_2$ where n_1, n_2 correspond to the size of the subnetworks X,Y, respectively. Without loss of generality, assume that $n_1 \leq n_2$. Let c_{ij} be the cost from vertex i to vertex j. All c_{ii} equal zero, and the adjacency matrix representing G is

symmetrical. Thus, the goal of the Kernighan-Lin algorithm is to minimize the cost C of the edges connecting the subnetworks X and Y, where for $y \epsilon Y$ and $x \epsilon X$

$$C = \sum_{X \times Y} c_{xy}. \tag{3.1}$$

For each node $\alpha \epsilon A$ where A may be either X or Y, let

$$D_\alpha = \sum_{\beta \epsilon \bar{A}} c_{\alpha\beta} - \sum_{\alpha' \epsilon A} c_{\alpha\alpha'} \tag{3.2}$$

where the first sum represents the intracluster costs between a vertex α and all other vertices in the opposite cluster. The second sum represents the intercluster costs between vertex α and all other vertices in its own cluster. Another important quantity to note is the *gain* g for swapping two nodes between their respective clusters. Let

$$g = D_x + D_y - 2c_{xy}. \tag{3.3}$$

Algorithm 1: Kernighan-Lin Algorithm

Data: An undirected network G and initial guesses for X and Y
Result: The subnetworks X and Y such that Equation 3.1 is minimized.
repeat
  Calculate D values $\forall\ x \epsilon X,\ y \epsilon Y$
  Let $Y' = Y$, $X' = X$.
  for $i = 1 : n_1$ **do**
    Select $y \epsilon Y'$ and $x \epsilon X'$ that maximizes g_i.
    Let $y'_i = y$ and $x'_i = x$.
    Remove the selected x and y from their respective clusters X' and Y'.
    Recalculate the D values for the remaining elements.
  Select j to maximize $\Gamma = \sum_{i=1}^{j} g_i$.
  if $\Gamma \succ 0$ **then**
    Swap the 1 to j x'_i's and y'_i's between X and Y.
until $\Gamma \leq 0$

The complexity of the Kernighan-Lin algorithm is $O(|V|^2 log|V|)$. It is very sensitive to the initial guesses for the subnetworks X and Y and may perform quite poorly for a random initialization. It is often the case that a different algorithm provides the initial guesses for the subnetworks, and the Kernighan-Lin algorithm improves upon those guesses. From a biological standpoint, the Kernighan-Lin algorithm may not be quite applicable as initial guesses for X and Y may be hard to obtain, especially if prior knowledge is lacking. Furthermore, the Kernighan-Lin algorithm imposes a minimum number of sub-pathways which may not be biologically valid. Regardless, the Kernighan-Lin algorithm did provide the inspiration for a post-processing community detection algorithm developed by Newman [25].

Algorithm 2: Post-processing Community Optimization

Data: An undirected network G and initial guesses for X and Y
Result: The subnetworks X and Y such that some quality function F is maximized.
repeat
  for $i = 1 : |V|$ **do**
    Move a vertex v from either X to Y or vice-versa that maximizes F.
    Remove vertex v from any further consideration.
    Store the resulting partition of G as P_i
  Select P_i that maximizes F.
  Let $X = X_i$ and $Y = Y_i$ obtained from P_i.
until *no further improvement in F can be obtained.*

3.5 Girvan-Newman Algorithm

The Girvan-Newman algorithm [13] is an extremely popular divisive clustering algorithm. Divisive clustering algorithms are machine-learning algorithms that provide users with partitions of varying sizes. They are also a type of hierarchical clustering algorithms of which a second type is agglomerative clustering. A brief description of the two types of hierarchical clustering algorithms now follows.

First, agglomerative clustering focuses on building clusters from the bottom up. One begins an agglomerative clustering algorithm with each vertex or node in its own cluster. Based on a specified distance metric, the two most similar clusters or partitions are combined into a single cluster. This process is recursively repeated until all nodes belong to a single cluster. While these algorithms are strong at find the core of different communities, they are weak in finding the outer layers. They have also been shown to produce inconsistent results for networks whose partitions are known [26].

On the other hand, divisive clustering algorithms use a top-down approach. Initially, all nodes belong to a single partition and are recursively divided until each node belongs to its own partition. These type of algorithms produce a dendrogram as can be seen in Figure 3.4.

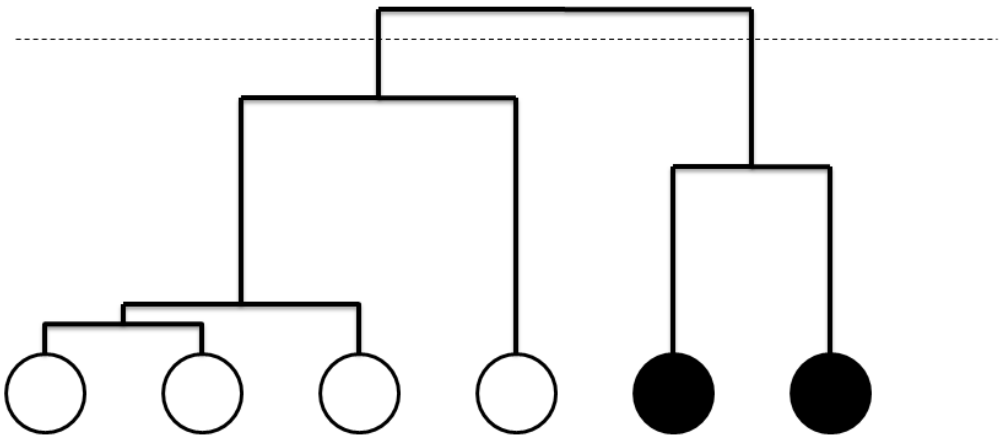


Figure 3.4: A dendrogram is produced as the output of a divisive clustering algorithm. To determine the final number of communities, the dendrogram needs to be cut. Where the dendrogram is cut is an area of research in of itself. For this dendrogram, the given cut line divides the network into two communities shaded white and black, respectively. This original author contribution is set to also appear in [2].

For the Girvan-Newman algorithm, it follows the spirit of divisive clustering algorithms. Compared to previous approaches, the Girvan-Newman algorithm focuses on the "information flow" of the network as opposed to its structure. As such, it focuses on highly

significant edges that serve as "bridges" between different communities. These edges tend to have a high value of "edge betweenness", which is an extension of vertex betweenness [11]. The authors introduced three types of edge betweenness: random-walk betweenness, current-flow betweenness, and shortest-path betweenness. In practice, shortest-path betweenness is most used and will be the focus for this section. The major reasons for using shortest-path betweenness is that it provides the best combination of performance and accuracy [26].

To calculate the shortest-path betweenness scores for all of the edges, one must first calculate all shortest paths between all pairs of vertices. For any given edge e, its betweenness score measures how many shortest paths possess it as an edge. One may refer to [26] for details on calculating shortest-path betweenness scores for an $O(|V||E|)$ algorithm. Overall, the Girvan-Newman algorithm displayed in Algorithm 3 has complexity $O(|V||E|^2)$. A sample result of the Girvan-Newman algorithm on Zachary's karate club may be seen in Figure 3.3.

Algorithm 3: Girvan-Newman Algorithm

Data: An unweighted and undirected network G
Result: A dendrogram representing the hierarchy of the different communities. The place where the dendrogram is cut determines the output communities.
Compute the shortest-path betweenness score $\forall$ edges $e \in E$.
for $i = 1 : |E|$ **do**
    Remove the edge $e \in E$ that possesses the largest shortest-path betweenness score from E.
    For all edges affected by the removal of e, recalculate their shortest-path betweenness scores.

The Girvan-Newman algorithm returns a varying number of communities depending on where the dendrogram is cut. Thus, one can have a myriad of resolutions to view the resulting communities by cutting the dendrogram at various locations. For the structure of biological pathways this allows a researcher to view a variety of hypothesized sub-pathways. It is often the case, though, that a researcher is only interest in the best partition amongst

all available candidate partitions. Thus, determining where to cut the dendrogram is a significant issue and subject to more research. Newman and Girvan attempted to address this limitation by introducing the concept of *modularity*. If a graph G divides into k communities, the modularity A is defined as

$$Q = \sum_i e_{ii} - ||e^2|| \tag{3.4}$$

where e is a $k \times k$ symmetric matrix where an entry e_{ij} measures the fraction of all edges that link community i and community j. For more details on modularity, one may refer to [2, 10, 26, 25].

3.6 Clique Percolation Method

The Clique Percolation Method (CPM) [27] is a community detection algorithm that allows for overlapping sub-pathways. This is an important feature, especially for biological pathways where a node in a biological pathway may participate in different biological processes. The building blocks of CPM are k-cliques. A k-clique is a maximal subgraph of size k such that any two nodes in the k-clique possess an edge between them. Another critical concept is *adjacent k-cliques*. Two k-cliques are said to be *adjacent* if and only if they share $k-1$ nodes. Thus, a *k-clique community* is the union of all adjacent k-cliques.

Concerning the algorithm itself, one key step is to find all of the maximal cliques within a given network. While the authors introduced a methodology to find maximal cliques, one may simply use the well-known Bron-Kerbosch algorithm [6] to find all of the maximal cliques in a network. Letting the total number of cliques found be denoted as n, another crucial concept for CPM is building an $n \times n$ clique-clique overlap matrix M. In this matrix M, each M_{ij} denotes the number of nodes shared between clique i and clique j. For details on CPM, one may refer to Algorithm 4.

Algorithm 4: Clique Percolation Method

Data: An unweighted and undirected network G and the size k of the k-clique communities to find.
Result: A set of k-clique communities.
For the graph G, find all of its maximal cliques.
Build an $n \times n$ clique-clique overlap matrix M.
Set all entries on the main diagonal of M less than k to zero.
Set all off-diagonal entries of M less than $k-1$ to zero.
Return the k-clique communities consisting of the connected cliques whose entries remain in M.

Probably one of the most major attractions for CPM in terms of biological pathways is its ability to find overlapping communities or sub-pathways. More importantly, Fortunato [10] stated that CPM has the ability to distinguish between graphs with community structure and random graphs. However, a major drawback for CPM is that not all of the nodes on the periphery of the network may participate in a module making it somewhat similar to agglomerative clustering algorithms. Furthermore, choosing a good value for k a priori is a daunting task. A potential solution to this problem is to extract all possible k-clique communities and then use a quality function like modularity to determine the best partition. CPM also has issues from a complexity perspective as its complexity cannot be expressed in closed form. At the very minimum, its complexity is in NP-complete since it involves finding maximal cliques, which is known to be NP-complete. Figure 3.5 illustrates the application of CPM on Zachary's karate club.

One final note of interest is that the CPM algorithm has a directed version noted as CPMd [28]. The key to this algorithm is to extend the concept of k-clique to *directed k-clique*. In its simplest form, a directed k-clique is simply a graph that has a subset of edges that produce a k-clique and a directed acyclic graph. For more details on finding directed k-cliques, one may refer to [28]. Figure 3.6 provides an illustration of a directed 4-clique.

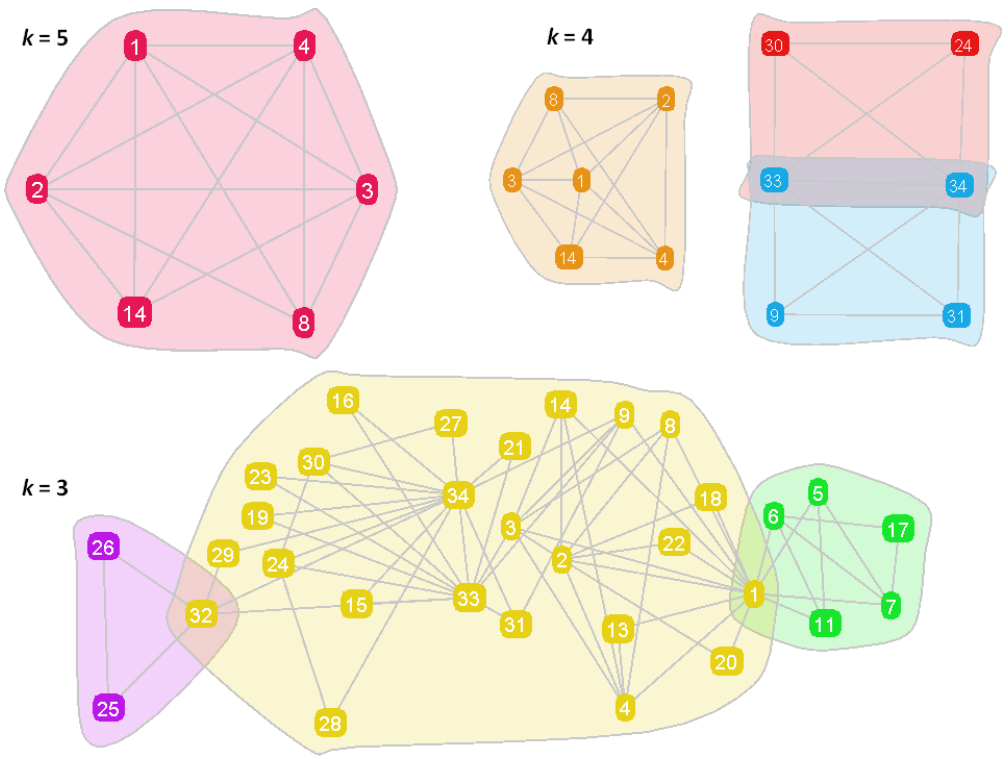


Figure 3.5: Using CFinder [3] Zachary's karate club is divided into three types of communities based on their k value. The partitions returned by CPM vary quite differently with the partitions seen in Figure 3.3. This original author contribution is set to also appear in [2].

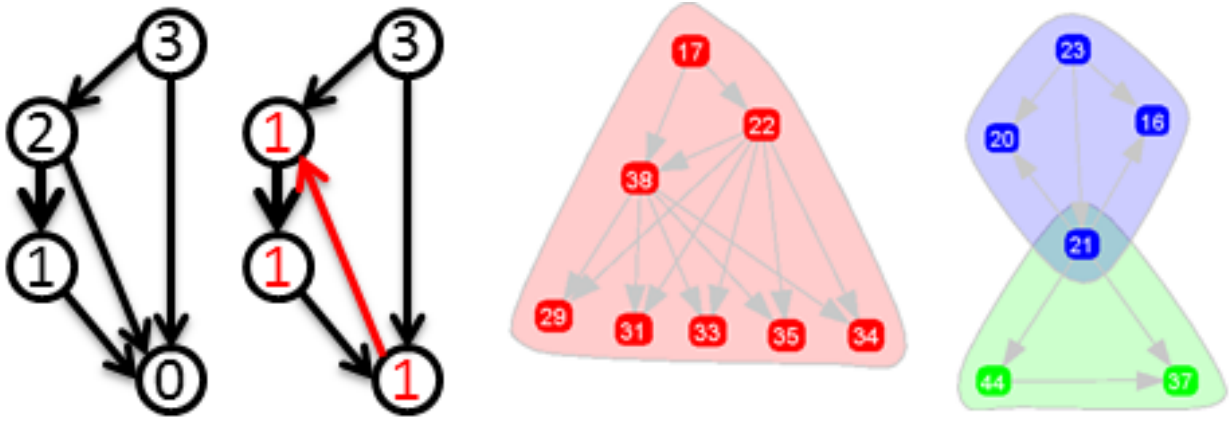


Figure 3.6: (Left) A directed acyclic graph and a directed 4-clique. The node labels refer to the outdegree of each node. (Middle-Left) While a 4-clique, it is not a directed 4-clique due to the presence of a cycle. Furthermore, it is necessary that each node has a unique outdegree in order for it to be a directed 4-clique. (Right) Using CFinder [3] the different 3-communities of the E. coli network in Figure 3.2 are found. Many nodes were left out of the final partitioning, which may prove problematic for analyzing the structure of some biological pathways. This original author contribution is set to also appear in [2].

Chapter 4: SEA

Structure Enrichment Analysis (SEA) is a standalone GUI software tool implemented in Matlab. It consists of a robust and modular software pipeline that allows for greater control of its core functionality. This robustness and modularity also makes incorporating components of SEA into other applications easy as well. SEA seeks to be a standard tool in the repertoire of tools available to biologists since there is an ever increasing amount of high-throughput data that needs analysis. Figure 4.1 presents an overview of the software pipeline.

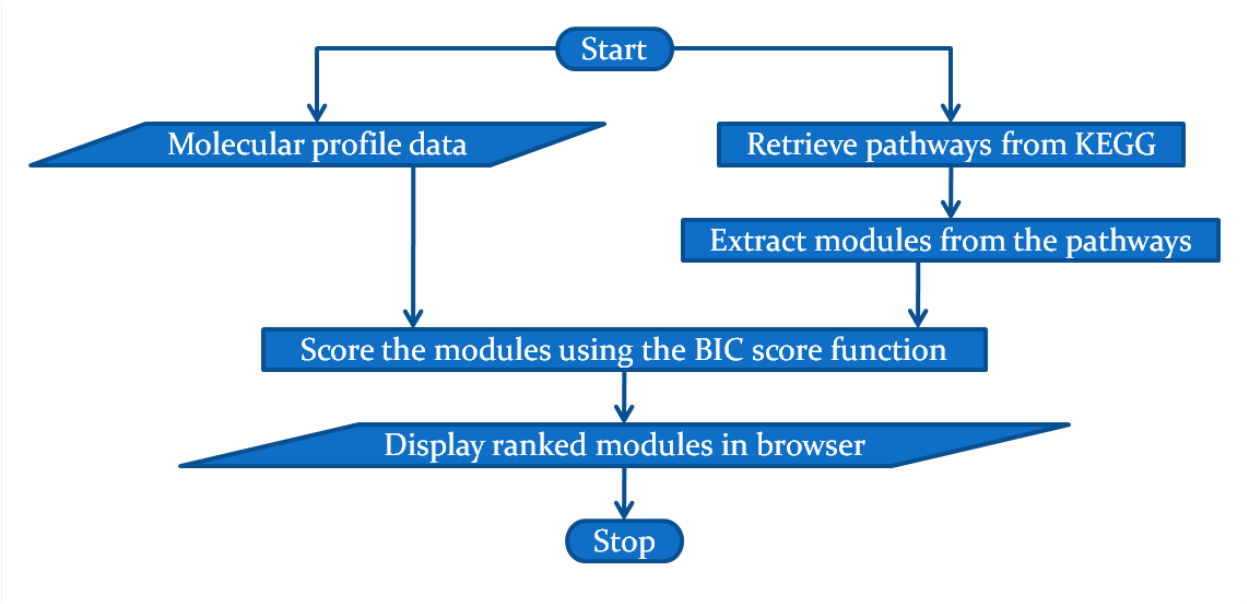


Figure 4.1: Overview of SEA: SEA first uses the KEGG (Kyoto Encyclopedia of Genes and Genomes) API [16, 18, 17] to extract adjacency matrices for the various pathways found in the KEGG pathway database. SEA then extracts signal cascades and nonlinear regulatory modules from each pathway. In conjunction with molecular profile data provided by users, SEA then uses the Bayesian Information Criterion (BIC) score function found in BNT [23] to score each module. SEA then displays the sub-pathways in a ranked list, and by clicking upon a result, the desired sub-pathway is displayed in the default web browser.

4.7 Related Work

Before discussing the details of SEA, it is prudent to discuss some of the other methods currently available. These include GenMAPP [9], the work by [7], and COSINE [21]. These methods provide biologists with useful concepts and tools for analysis of their data. However, each method mentioned has their own shortcoming that needs to be addressed. Concerning the current trend in the research, the overall trend seems to be combining molecular profile data and biological pathways in a meaningful manner.

4.7.1 GenMAPP

GenMAPP (Gene Map Annotator and Pathway Profiler) is a popular tool that takes as input two sets of gene expression data from the user. It then color-codes different genes based on fold changes. The genes are mapped across different pathways. Figure 4.2 provides a sample output of GenMAPP.

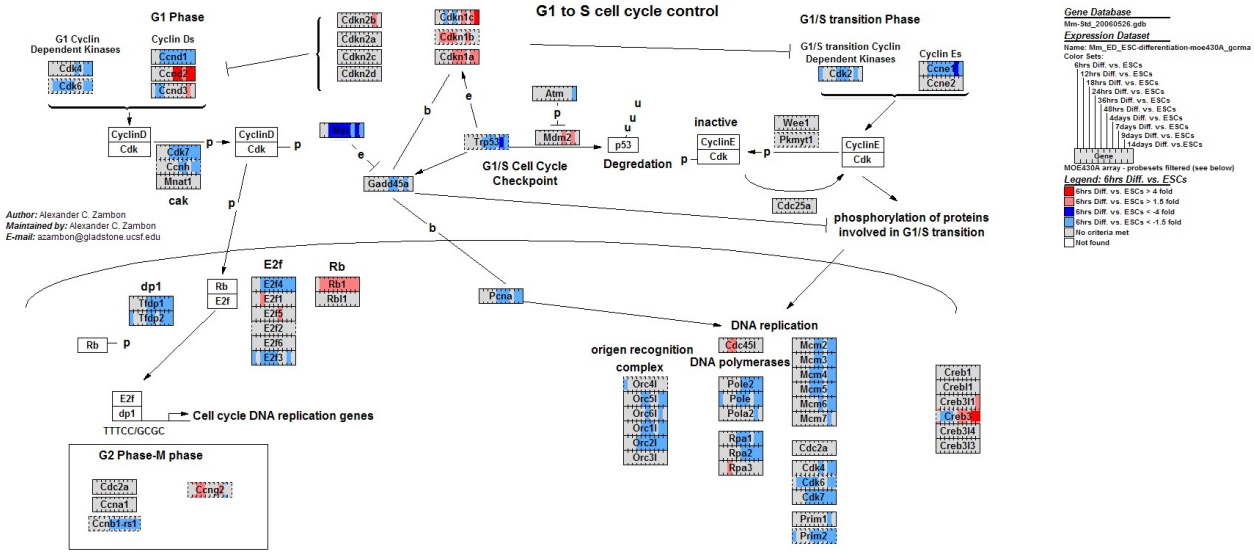


Figure 4.2: An illustration of GenMAPP [9]

While GenMAPP has proven to be quite a useful and popular tool for biologists, the use of fold changes to determine *activated* genes cannot properly detect some biological processes such as stress response and transcriptional programs. These processes are typically distributed across subnetworks where changes may be quite subtle at the level of individual genes. Also, as seen in Figure 4.2, GenMAPP makes no use of the pathway topology. There is no structure that connects genes on a given pathway.

4.7.2 The Work of Chen *Et Al.*

Chen *et al.* used the pathways in the KEGG (Kyoto Encyclopedia of Genes and Genomes) pathway database as their starting point for their work. For these pathways they constructed DFS-trees and extracted root to leaf linear paths from these trees. They then proceed to score each linear path using Euclidean distance to determine the significance of a linear path from the two sets of gene expression data.

Chen *et al.* provided quite an interesting concept that tries to fully account for the topology of a given module. Their approach does have limitations, though. First, their score function makes use of Euclidean distance. By using Euclidean distance, they do not fully capture pathway topology as one can permute the order of the genes within the linear path and still obtain the same score. They also do not account for nonlinear sub-pathways. Finally, they do not provide a software package to allow users to make use of their work.

4.7.3 COSINE

Another approach is COSINE (COndition-SpecIfic sub-Network). COSINE is novel since it focuses both on differential expression of genes and the differential correlation of gene pairs. It uses a genetic algorithm to find the best subnetwork for a given background pathway or network. It is currently available as an R package.

While COSINE is quite an interesting work, its weakness lies in using gene pairs to build its subnetwork. Given the score function used in COSINE, this network is naturally undirected. While undirected networks may suffice for PPI networks, they lack the important directionality information needed by signaling pathways. Furthermore, COSINE does not consider gene relationships beyond a pair.

4.8 Goals and Original Contributions

The goals of SEA are manyfold based on the underlying assumption that pathways are activated either via signaling cascades (linear sub-pathways) or nonlinear regulatory modules (nonlinear sub-pathways). First, SEA focuses on network structures. The topology of these structures are fully accounted for by making use of the BIC (Bayesian Information Criterion) score function found in BNT [23]. This allows SEA to go beyond the placement of genes on a pathway or the order of genes within a gene set. SEA also provides users with a GUI (Graphical User Interface) to allow for ease of use and the visualization of significant sub-pathways. SEA will also seek an answer to the important biological question: "From between signal cascades and nonlinear nonlinear regulatory modules, which one is more significant?"

Given the knowledge available at the moment, the original contributions of SEA are manyfold. First, SEA focuses both on linear and nonlinear sub-pathways of a given pathway. Furthermore, SEA fully accounts for the topology of sub-pathways via the use of the BIC score function as opposed to statistical tests. Finally, SEA does not need multiple classes of data to map molecular profile data onto different pathways. SEA only needs to make use of steady-state data or time series data.

4.9 Pathway Extraction

Figure 4.1 succinctly surmises the software pipeline that makes SEA. The first step within the pipeline is fetching the pathways from the KEGG pathway database. There are two approaches from which one can obtain pathway data. One approach is by parsing KGML (KEGG Markup Language) files. Essentially an XML file, a KGML file consists of *entries*, *relations*, and *reactions*. Entries correspond to the nodes or vertices of the pathway. Relations and reactions are two sets of edges between the entries corresponding to a network of proteins and a network of chemical compounds, respectively. One can then parse the KGML files to extract their corresponding pathways.

The major downside to using KGML files for pathway extraction is obtaining these files in an automatic fashion. Previously, one could use the KEGG FTP server to automatically download all of the relevant pathways for any given organism. However, from July 1, 2011 onwards, the KEGG FTP server is no longer freely available for academic users. Instead, users can now download the respective KGML file from the pathway's webpage. However, how one can obtain a list of pathways remains to be seen.

Before describing the KEGG API, it is prudent to discuss *entries* and *relations* as this same information can be extracted using the KEGG API. Given that SEA works with molecular profile data, only protein networks comprised from relations are of interest so *reactions* are not extracted. The important components of an *entry* in a KGML file are its ID, gene name, and components if any. An entry's ID most closely resembles an ID of a node in an adjacency matrix. Gene name describes what gene the entry corresponds to using the KEGG naming system. It may very well be the case that multiple entries have the same gene name as can be seen in Figure 4.3. Components are used to define a compound gene where compound genes are defined in two manners. First, a gene name may consist of multiple genes. The latter method uses components that are composed of multiple IDs.

One must then find the entries with the corresponding IDs to extract the gene names within those IDs. Multiple temporary map structures are used to achieve this goal.

Concerning *relations*, the set of *relations* are very similar in concept to a list of edges where each relation essentially consists of an entry ID pointing towards another entry ID. Another important characteristic is the type of edge. As of now, edges corresponding to *maplink* are pruned as one of their nodes correspond to a whole pathway, which cannot be represented effectively using molecular profile data.

Given the above, for pathway extraction and pathway visualization, the KEGG API has been chosen. The KEGG API provides a SOAP/WSDL interface available in Perl, Ruby, Python, Java, and Matlab. The KEGG API provides a robust set of methods for a variety of functions of which a subset is used. These include fetching the relations of a pathway and the entries of a pathway. To extract the pathways, the major methods used are as follows: list_pathways, get_elements_by_pathway, and get_element_relations_by_pathway.

The first method, list_pathways, is necessary to obtain the list of pathways for a given organism. The KEGG pathway database is constantly updated. As such, it is necessary to use this method to list all of the pathways for a given organism. The latter two methods are needed to extract the nodes and edges of the pathways, respectively. Both sets are pruned to eventually extract the final adjacency matrices corresponding to their respective pathways. It is important to note that KEGG pathways may posses an element of *redundancy* as illustrated in Figure 4.3. At this stage a faithful representation of KEGG pathways is kept. Only self-cycles and edges mapping a gene to another pathway are removed at this stage. Any pathway that does not possess at least one linear path of length one is removed from further consideration. Concerning duplicate genes that may occur, Section 4.11 details in more detail how sub-pathways with duplicate gene elements are handled.

Concerning the pathway that is extracted, a one to one relationship is kept between both the KEGG element ID and the adjacency matrix ID representing the pathway. Often-

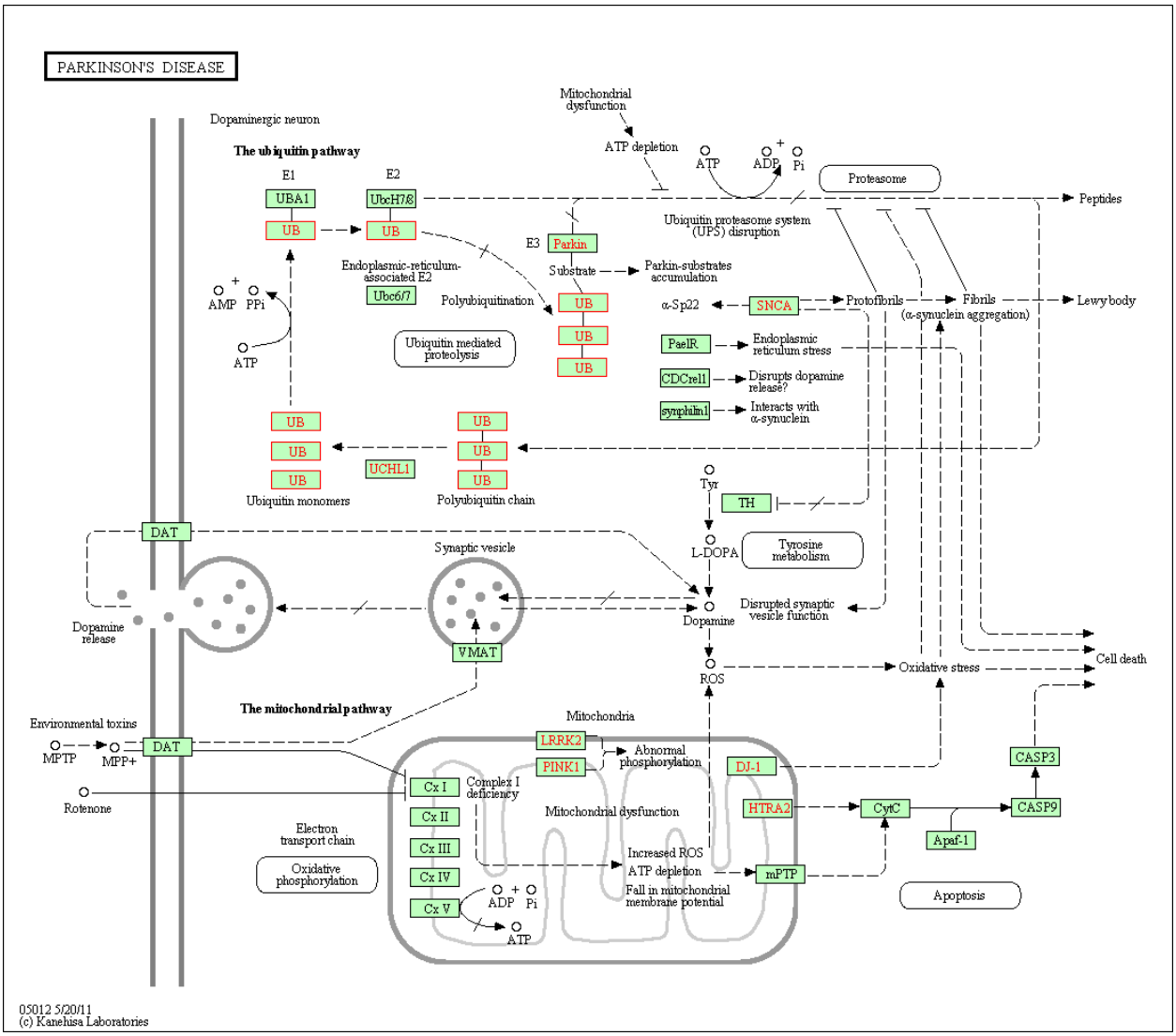


Figure 4.3: A sample pathway illustrating the duplication found in KEGG. KEGG essentially tries to represent biological processes as opposed to constructing an adjacency matrix. In this pathway Ubiquitin B (UB), highlighted with a red border, appears multiple times. Thus, a programmer must choose whether to consolidate the different entries or represent the networks as they are. For SEA the latter approach was chosen.

times this may create a very sparse matrix as not all matrix IDs have corresponding KEGG element IDs, but this allows SEA to represent as faithfully as possible the original pathway.

Throughout the pathway extraction process, essential information is stored in a data structure. One vital piece is a map that maps matrix IDs, essentially entry IDs,

to their equivalent gene names. This map is essential especially in Section 4.11 where a module is mapped from local matrix IDs to their gene names and finally a global matrix integer. These local matrix IDs are kept throughout the lifetime of a module since *get_html_of_colored_pathway_by_elements* from the KEGG API needs the entry IDs to display the results as seen in Subsection 4.14.6.

4.10 Retrieving NCBI Gene IDs

Once the pathways are extracted, a list of all genes present in all of the pathways are extracted. It is important to remember that some of these nodes in the KEGG pathways are actually a combination of different genes. The decomposition of these compound genes into individual genes is handled automatically. Thus, SEA now has a large list of genes. It is also important to note that these genes have labels specific to KEGG only. For example, *hsa:7314* is the KEGG gene for UB where *hsa* corresponds to *Homo Sapiens* and 7314 is the KEGG gene label.

To get the NCBI Gene IDs, the *bconv* method from the KEGG API is used. Using the list of genes as input, this returns a large string consisting of the KEGG gene label and its equivalent amongst many other databases. Since the NCBI Gene ID system is most complete (in fact, for the case of *hsa*, there is a one to one mapping from KEGG to NCBI Gene ID), the NCBI Gene IDs are extracted with their corresponding KEGG gene labels. Two very important maps are now built. The first map, called $GeneToGlobalID$, maps a KEGG node, including compound genes, to a global integer. The key for this map is the KEGG gene label or a set of KEGG gene labels for compound genes. $GeneToGlobalID$ is needed to map a gene to a row of the data matrix. The second map, called $NCBItoKEGG$, maps NCBI Gene IDs to KEGG gene labels, which is needed later on for Section 4.12.

4.11 Decomposing the Pathways

With the list of pathways complete as well as a map that maps KEGG genes to a global integer, SEA proceeds to extract both linear and nonlinear components. For each pathway, both linear and nonlinear sub-pathways are extracted as seen in subsections 4.11.1 and 4.11.2. Using the map $GeneToGlobalID$ in conjunction with the local maps per pathway mentioned in Section 4.9, sub-pathways now have an equivalent, global representation. Thus, the problem illustrated earlier in Figure 4.3 is now easily solved by checking for any duplicate global IDs in the new representation for the module. If any duplicates are found, the module is simply discarded.

4.11.1 Signal Cascades

Extracting signal cascsades is not necessarily a trivial task. The *natural* way in doing so would be to simply extract all root to leaf linear paths of the original pathway. However, such an approach will produce for some pathways a computationally intractable number of signal cascades to analyze. Thus, a *sample* of the total signal cascades is needed.

To obtain a sample of the signal cascades, the "vanilla" DFS algorithm found in [8] has been modified. The major modifications involve modifying the order for which DFS visits nodes within a pathway. The order first places roots at the forefront and all other nodes afterwards. Each sublist is ranked by the outdegree of each node such that nodes with a high outdegree are prioritized. Finally, only tree edges are kept in the DFS-tree Dt where forward edges, back edges, and cross edges are discarded. Once the tree is constructed, all root to leaf paths of Dt are extracted. This produces a sample of linear paths that is a subset of the full set of linear paths. It is important to note, though, that a root to leaf linear path of a DFS-tree may not necessarily correspond to a root to leaf linear path of the original pathway as indicated in Figure 4.4.

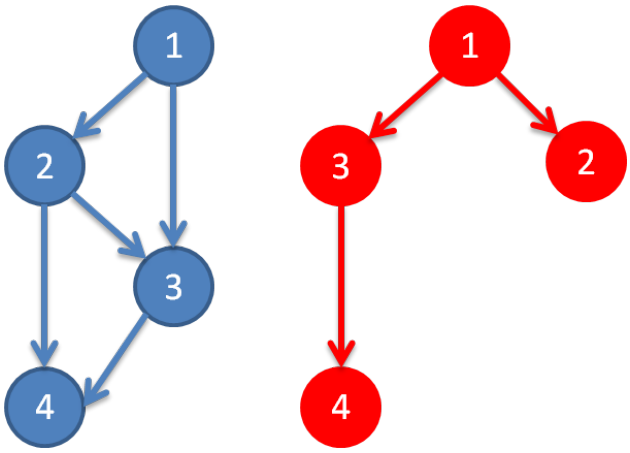


Figure 4.4: A network (blue) and a sample DFS-tree (red). While $\{1, 2\}$ is a root to leaf linear path of the DFS-tree, it is not the case for the original network.

4.11.2 Nonlinear Regulatory Modules

Nonlinear sub-pathways are extracted using a modified version of the CPM algorithm [27] as detailed previously in Section 3.6. Essentially, instead of finding all cliques, only feed-forward loops are found, which are directed cliques of size three. All other details of the algorithm remain the same. Furthermore, the choice of feed-forward loops is well-justified given their biological significance [5].

It is important to note that the procedures outlined from Sections 4.9 to 4.11 occur only once or whenever the user updates a chosen organism. This reduces the computational complexity of the SEA software pipeline as there is no need to compute a list of sub-pathways every time the user runs the program. It is sufficient to use a precomputed list of sub-pathways.

4.12 User Input

For input SEA takes molecular profile data in the form of a tab-delimited text file. SEA takes this tab-delimited text file and extracts a data matrix D where the map $GeneToGlobalID$

maps a KEGG gene to a row in D. Concerning the input file, each line must consist of an NCBI Gene ID and the corresponding molecular profile data. SEA can also handle multiple occurrences of an NCBI Gene ID within a file. It keeps the row with the highest average to include in its data matrix. It can also handle multiple NCBI Gene IDs per row as well and updates the corresponding row of D accordingly.

To map the NCBI Gene IDs properly, the $NCBItoKEGG$ map constructed in Section 4.10 is used. Once all single genes are mapped, compound genes consisting of multiple single genes are also mapped. For a row in D corresponding to a compound gene, its mapped value consists of the average of all rows for its element genes. Once manipulation of the data file is complete, the user is then informed of the number of sub-pathways that their data supports as it may be the case their data set does not have all of the genes found in the extracted KEGG pathways.

4.13 Scoring the Sub-pathways

Given the precomputed list of sub-pathways as well as molecular profile data loaded by the user, the user may proceed to scoring and ranking the precomputed list of sub-pathways. To score their sub-pathways, the BIC score function found in BNT (Bayes Net Toolbox) is used. The underlying assumptions made are that the data originated from a Gaussian distribution. Furthermore, another underlying assumption is that the module is a DAG (directed acyclic graph). Given the manner in which the sub-pathways were extracted in Section 4.11, all of the sub-pathways extracted are by their nature DAGs. After the user scores the desired sub-pathways, they are displayed in ranked order as seen in Figure 4.6.

4.14 The Graphical User Interface (GUI)

The final component is the GUI that provides the user with access to the various functionality of SEA. There are a variety of useful features in the GUI that provide the user with a variety of options. The first feature of note is the user-friendly "quick start guide" that appears on program execution. It can also be accessed via the menu by selecting *Help* → *Quick Start Guide*. Figure 4.5 shows the guide visible to users of SEA.

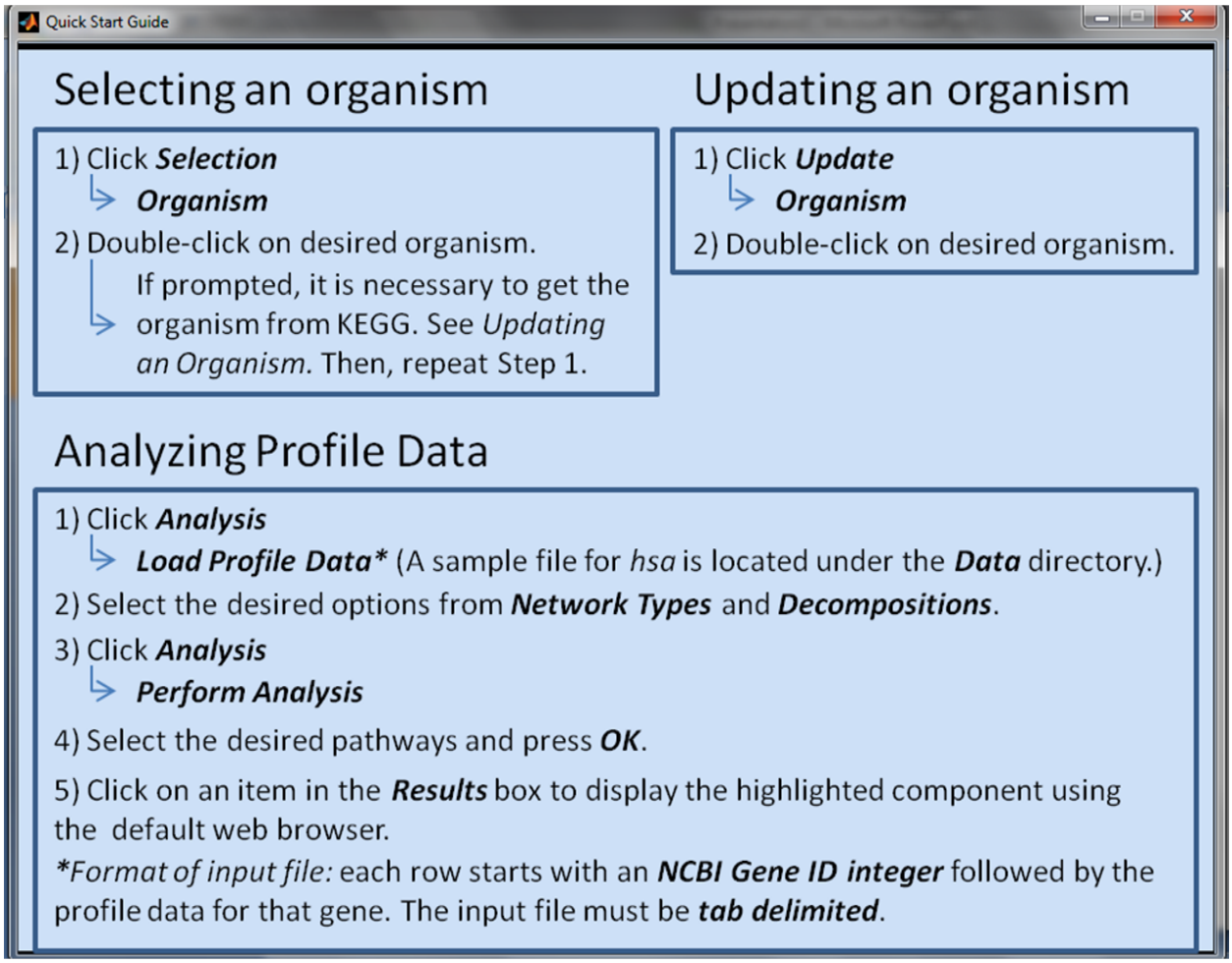


Figure 4.5: Quick Start Guide for SEA

Beyond the "quick start guide," there are a variety of features available that allow the user a wide range of flexibility. The overall interface can be seen in Figure 4.6. These

features include *updating the list of organisms*, *selecting or updating an organism*, *loading profile data*, *selecting a subset of sub-pathways*, *ranking the sub-pathways*, *viewing results*, and *saving and loading previous results*. These features will be examined in further detail below.

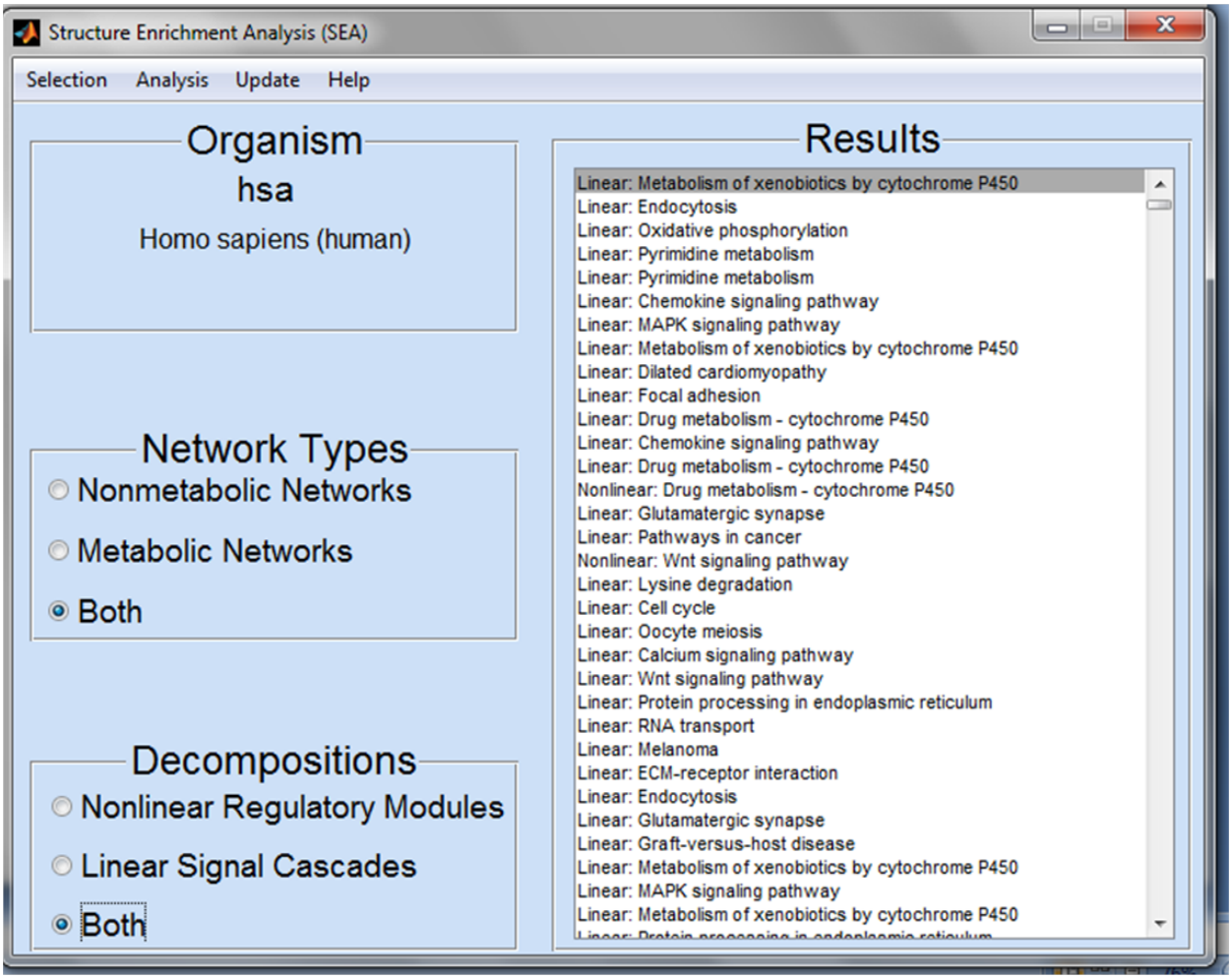


Figure 4.6: SEA Interface

4.14.1 Updating the List of Organisms

The KEGG pathway database is constantly updated. Usually the updates deal with adding and modifying pathways, but there are times when new organisms are added. Thus, it is

necessary to update the list of organisms from time to time. The user can do so by selecting *Update* → *List of Organisms* from the menu. For this feature the method of interest from the KEGG API is *list_organisms*.

4.14.2 Selecting or Updating an Organism

Selecting or updating an organism is a straightforward process. The procedure for each can be obtained from Figure 4.5. For updating an organism, essentially the procedures detailed in Sections 4.9 to 4.11 occur. As for selecting an organism, it loads into memory a precomputed list of sub-pathways.

4.14.3 Loading Profile Data

To load some molecular profile data, the user selects *Analysis* → *Load Profile Data*. Essentially, this feature makes use of the procedure listed in Section 4.12. As stated before, a message box informs the user of the sub-pathways that are supported by his data set.

4.14.4 Selecting a Subset of Sub-pathways

It may very well be the case that a user is not interested in examining all of the sub-pathways for all of the KEGG pathways. As seen in Figure 4.6, there are a variety of radio buttons that allow the user maximum flexibility over the types of sub-pathways they wish to analyze. One group of radio buttons allows the user to specify the type of pathways they wish to study whether they are metabolic, nonmetabolic, or both. The other group of radio buttons allows the user to specify the type of sub-pathways they wish to examine whether they are linear, nonlinear, or both. Upon selecting *Analysis* → *Perform Analysis*, the user is presented with a customized list of pathways corresponding to the options selected.

4.14.5 Ranking the Sub-pathways

In order to rank the sub-pathways, the user can select from the menu *Analysis* → *Perform Analysis*. This essentially calls the procedure found in Section 4.13, which calculates the BIC score for each module. The sub-pathways are then sorted in descending order based on their BIC scores and are then displayed in the Results box as seen in Figure 4.6.

4.14.6 Viewing Results

After ranking the sub-pathways, the user is presented with a list of sub-pathways in ranked order as seen in Figure 4.6. Clicking on an item in the Results box displays a pathway with the module highlighted as seen in Figure 4.7. Users can further click upon an item in their web browser to view detailed information for their selected entry. The essential method from the KEGG API being used is *get_html_of_colored_pathway_by_elements*.

4.14.7 Saving and Loading Results

This feature allows users to save or load results. The former is accomplished by selecting from the menu *Analysis* → *Save Current Results* while the latter is accomplished by selecting from the menu *Analysis* → *Load Previous Results*. These features will allow users to share their data with one another as well as performing joint analysis.

4.15 Conclusions

In this thesis two major pieces of original work were presented. The first work, LPA (Linear Path Augmentation) was presented in Section 2.3. LPA is a novel algorithm that seeks to reconstruct networks using gene sets only. A variety of novel techniques were presented, but its current limitation is its computational complexity. The second work, SEA, presents

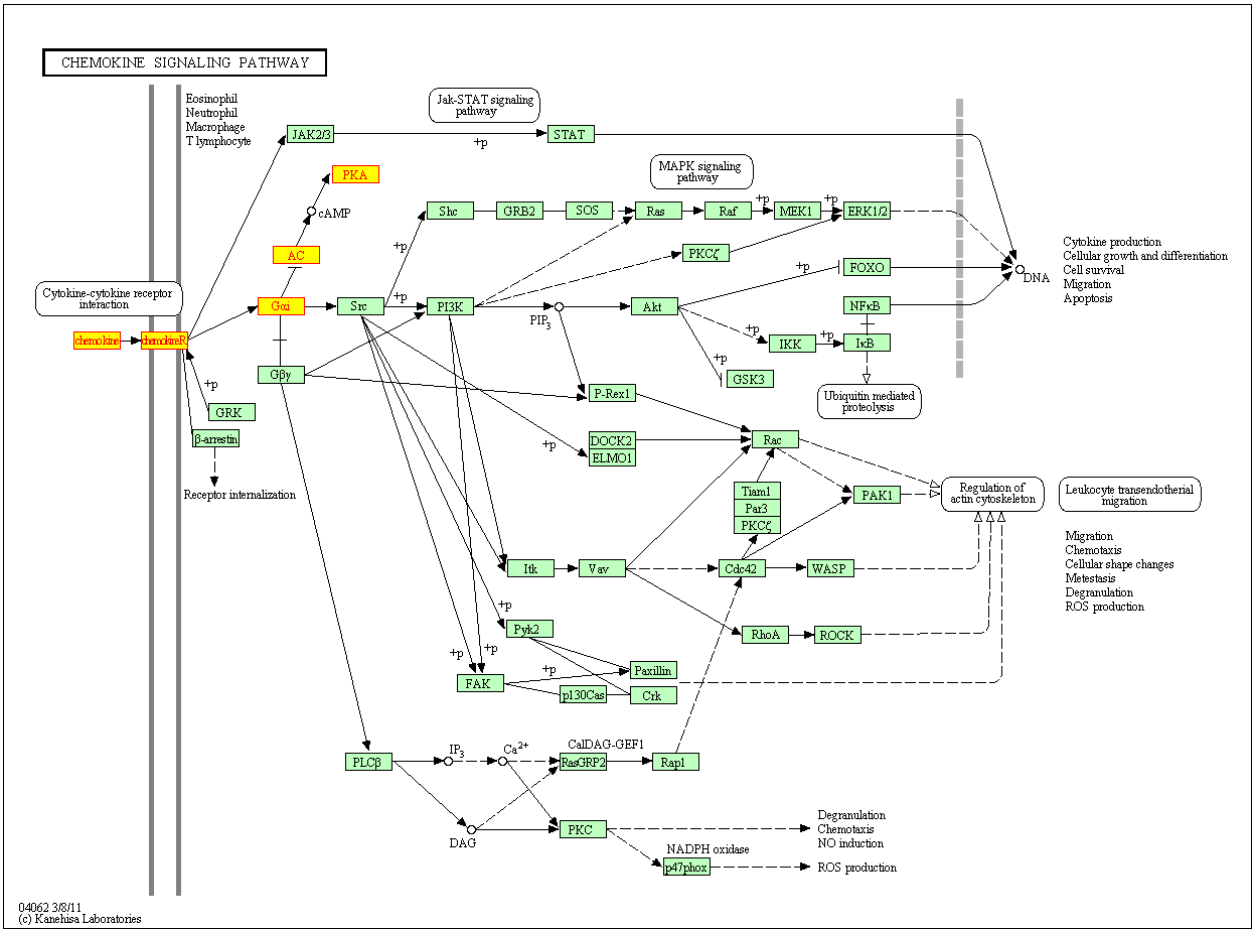


Figure 4.7: Results displayed in the default web browser using the KEGG API.

a novel pipeline to highlight significant sub-pathways on pathways. It is essentially ready for deployment save for some final validation studies. It is hoped that SEA will become a standard tool used by biologists worldwide.

Finally, for future work there are a variety of directions that can be taken. Both LPA and SEA can be further improved upon. Given the modular nature of both pipelines, further improvement and refinement is not very difficult. For LPA the major improvement needs to be in the *Growth* stage that is the current bottleneck. New techniques can also be incorporated to allow LPA to handle gene sets that do not originate from a DAG. For SEA there are a variety of improvements that can be pursued. Support for additional pathway databases can be added. The ideal, though, would be to use an algorithm such as LPA to

infer the pathways. One can then use these context-specific pathways in addition to the KEGG pathways as the starting point. Further research can also be conducted in scoring the sub-pathways. In short, given the robustness and versatility of the SEA pipeline, there is no shortage of areas for future research and improvements.

References

[1] L. Acharya, T. Judeh, Z. Duan, M. Rabbat, and D. Zhu. GSGS: A Computational Framework to Reconstruct Signaling Pathways from Gene Sets. *ArXiv e-prints*, January 2011.

[2] L. Acharya, T. Judeh, and D. Zhu. A survey of computational approaches to biological network reconstruction and partition. In M. Dehmer, editor, *Machine Learning Approach for Network Analysis: Novel Graph Classes and Classification Techniques*. Wiley Publishing, To appear 2011.

[3] B. Adamcsek, G. Palla, I. J. Farkas, I. Derenyi, and Vicsek T. Cfinder: Locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[4] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Molecular Biology of the Cell, fourth edition*. Garland Science, 2002.

[5] Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6):450–461, 2007.

[6] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577, September 1973.

[7] Xiujie Chen, Jiankai Xu, Bangqing Huang, Jin Li, Xin Wu, Ling Ma, Xiaodong Jia, Xiusen Bian, Fujian Tan, Lei Liu, Sheng Chen, and Xia Li. A sub-pathway-based approach for identifying drug response principal network. *Bioinformatics*, 27(5):649–654, 2011.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[9] Kam D. Dahlquist, Nathan Salomonis, Karen Vranizan, Steven C. Lawlor, and Bruce R. Conklin. Genmapp, a new tool for viewing and analyzing microarray data on biological pathways. *Nature Genetics*, 31(1):19–20, 2002.

[10] S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, February 2010.

[11] Linton C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, March 1977.

[12] Nir Friedman, Michal Linial, and Iftach Nachman. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.

[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[14] Kevin L. Gunderson, Semyon Kruglyak, Michael S. Graige, Francisco Garcia, Bahram G. Kermani, Chanfeng Zhao, Diping Che, Todd Dickinson, Eliza Wickham, Jim Bierle, Dennis Doucet, Monika Milewski, Robert Yang, Chris Siegmund, Juergen Haas, Lixin Zhou, Arnold Oliphant, Jian-Bing Fan, Steven Barnard, and Mark S. Chee. Decoding Randomly Ordered DNA Arrays. *Genome Research*, 14(5):870–877, May 2004.

[15] Thair Judeh, Lipi Acharya, and Dongxiao Zhu. Gene network inference via linear path augmentation. In *BIOT 2010*, 2010.

[16] Minoru Kanehisa and Susumu Goto. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.

[17] Minoru Kanehisa, Susumu Goto, Miho Furumichi, Mao Tanabe, and Mika Hirakawa. Kegg for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38(suppl 1):D355–D360, 2010.

[18] Minoru Kanehisa, Susumu Goto, Masahiro Hattori, Kiyoko F. Aoki-Kinoshita, Masumi Itoh, Shuichi Kawashima, Toshiaki Katayama, Michihiro Araki, and Mika Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Research*, 34(suppl 1):D354–D357, 2006.

[19] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(1):291–307, 1970.

[20] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, and E. L. Brown. Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nature Biotechnology*, 15:1359–1367, 1997.

[21] Haisu Ma, Eric E. Schadt, Lee M. Kaplan, and Hongyu Zhao. Cosine: Condition-specific sub-network identification using a global optimization method. *Bioinformatics*, 27(9):1290–1298, 2011.

[22] Schaffter T. Mattiussi C. Marbach, D. and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229 – 239, 2009.

[23] Kevin P. Murphy. The bayes net toolbox for matlab. *Computing Science and Statistics*, 33:2001, 2001.

[24] Chris J Needham, James R Bradford, Andrew J Bulpitt, and David R Westhead. A primer on learning in bayesian networks for computational biology. *PLoS Comput Biol*, 3(8):e129, 08 2007.

[25] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, Feb 2004.

[27] Gergely Palla, Imre Derenyi, Illes Farkas, and Tamas Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[28] Gergely Palla, Ills J Farkas, Pter Pollner, Imre Dernyi, and Tams Vicsek. Directed network modules. *New Journal of Physics*, 9(6):186, 2007.

[29] M G Rabbat, J R Treichler, S L Wood, and M G Larimore. Understanding the topology of a telephone network via internally-sensed network tomography. In *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 977–980, 2005.

[30] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[31] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science (New York, N.Y.)*, 270(5235):467–470, October 1995.

[32] J. Shendure, R. D. Mitra, C. Varma, and G. M. Church. Advanced sequencing technologies: methods and goals. *Nature Reviews Genetics*, 5(5):335–344, 2004.

[33] Jay Shendure and Hanlee Ji. Next-generation dna sequencing. *Nat Biotechnol*, 26(10):1135–1145, October 2008.

[34] Aravind Subramanian, Heidi Kuehn, Joshua Gould, Pablo Tamayo, and Jill P. Mesirov. Gsea-p: a desktop application for gene set enrichment analysis. *Bioinformatics*, 23(23):3251–3253, 2007.

[35] J.-P. Vert. Reconstruction of biological networks by supervised machine learning approaches. *ArXiv e-prints*, June 2008.

[36] W W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

Vita

The author was born in New Orleans, Louisiana. He received a double major in Mathematics and Computer Science from Loyola University New Orleans in 2005. He joined the University of New Orleans in the Fall of 2009 to pursue graduate studies in Computer Science. He became a member of Dr. Dongxiao Zhu's group in Spring 2010.