1-1-2014

Teak: A Novel Computational And Gui Software Pipeline For Reconstructing Biological Networks, Detecting Activated Biological Subnetworks, And Querying Biological Networks.

Thair Judeh
Wayne State University,

TEAK: A NOVEL COMPUTATIONAL AND GUI SOFTWARE PIPELINE FOR RECONSTRUCTING BIOLOGICAL NETWORKS, DETECTING ACTIVATED BIOLOGICAL SUBNETWORKS, AND QUERYING BIOLOGICAL NETWORKS.

by

THAIR B. JUDEH

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

2014

MAJOR: COMPUTER SCIENCE

Approved by:

_____

Advisor Date

_____

_____

_____

DEDICATION

To my MOTHER and FATHER

ACKNOWLEDGEMENTS

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

Furthermore, I thank my advisor Dr. Dongxiao Zhu for all of the good that he has done for me. I extend an extra special thanks to Dr. Anuj Kumar, Dr. Thaer Jayyousi, and Dr. Robert G. Reynolds with whom we shared an excellent and fruitful collaboration as well as Dr. Xuewen Chen who was always ready to lend an ear when needed. I also thank Dr. Lipi Acharya with whom I have collaborated with on a variety of research projects. I also thank the Department of Computer Science at Wayne State University for the generous funding they have provided in supporting the research that Dr. Zhu and I undertook.

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

Dedication ii

Acknowledgements iii

List of Figures vii

List of Tables viii

Chapter 1: Background and Introduction 1

Chapter 2: A Survey of Network Reconstruction Algorithms 7

2.1 Bayesian Networks 9

2.2 Frequency Method 11

2.3 GSGS and GSSA 12

2.4 Linear Path Augmentation (LPA) 14

2.5 Conclusions 19

Chapter 3: A Survey of Network Partitioning Algorithms 21

3.1 Kernighan-Lin Algorithm 25

3.2 Girvan-Newman Algorithm 27

3.3 Newman's Eigenvector Method 30

3.4 Infomap 35

3.5 Clique Percolation Method 38

3.6 Conclusions 41

Chapter 4: Gene Set Cultural Algorithm (GSCA) 44

4.1 Method Overview 45

4.2 The Belief and Population Spaces 49

4.3 Heuristic Fitness Function Justification 53

4.4 Simulated Data Analysis 57

4.5 Real Data Analysis 60

4.6 Conclusions 61
Chapter 5: Topology Enrichment Analysis frameworK (TEAK) 64
5.1 Method Overview 68
5.2 Subpathway Extraction 69
5.3 Subpathway Ranking 74
5.4 Conclusions and Results 77
Chapter 6: Query Structure Enrichment Analysis (QSEA) 80
6.1 Method Overview 83
6.2 Edge and Vertex Betweenness 85
6.3 Feedback Arc Set 87
6.4 Shortest Paths and Transitive Closure 94
6.5 Query Matching and Output 94
6.6 Conclusions 98
Chapter 7: Conclusions and Future Work 99
Appendix A: List of Publications 102
Appendix B: Copyrights. 104
References. 136
Abstract 137
Autobiographical Statement 139

LIST OF FIGURES

Figure 1.1 Gene Expression Data and Gene Sets. 3

Figure 1.2 Overall Dissertation Framework 5

Figure 2.1 Network Reconstruction from Gene Sets Problem Overview . 8

Figure 2.2 DAG Transpose Problem 15

Figure 2.3 LPA Overview 17

Figure 3.1 Two Communities 22

Figure 3.2 Directed Versus Undirected Communities 24

Figure 3.3 Zachary's Karate Club 24

Figure 3.4 A Dendrogram from a Divisive Clustering Algorithm 28

Figure 3.5 The Girvan-Newman Partitioning of Zachary's Karate Club . 29

Figure 3.6 Limits of Directed Modularity 35

Figure 3.7 Infomap's Partitioning of Zachary's Karate Club 38

Figure 3.8 CPM's Partitioning of Zachary's Karate Club 40

Figure 3.9 CPM Directed Cliques 41

Figure 3.10 CPMd Illustration 42

Figure 4.1 Cultural Algorithm Framework 46

Figure 4.2 GSCA Algorithm Overview 47

Figure 4.3 Domain Knowledge Usage 51

Figure 4.4 *E. coli* Fitness results 55

Figure 4.5 *Insilico* Fitness results 56

Figure 4.6 GSCA/GSGS Comparison Plots 57

Figure 4.7 GSCA Plots 58

Figure 4.8 Prior Knowledge Plots 59

Figure 4.9 GSCA Hughes' Data Results 62

Figure 5.1 TEAK Result 65
Figure 5.2 TEAK Framework 67
Figure 5.3 Illustration of Algorithm 5.1 71
Figure 5.4 I-equivalences of Feed-forward Loops 74
Figure 6.1 QSEA Framework 84
Figure 6.2 Feedback Arc Set Removal Example 86
Figure 6.3 QSEA Performance- Cyclic Networks 90
Figure 6.4 QSEA Performance- Acyclic Networks Made Cyclic 91
Figure 6.5 QSEA Query Input 95
Figure 6.6 QSEA Query Result 96

LIST OF TABLES

Table 4.1 DREAM3 and DREAM4 Network Statistics 54

Table 5.2 KGML Edge Directions 68

Table 5.3 TEAK Linear Comparisons 78

Table 5.4 TEAK Nonlinear Comparisons 79

Chapter 1: Background and Introduction[1]

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 a lacking of data needed to fully capture the underlying biomolecular activities. Thus, three problems have recently received a considerable amount of attention: (1) inferring biological pathway topologies from gene expression data and gene sets, (2) decomposing different biological pathways into functional units, and (3) querying pathways in search of support for biological hypotheses.

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 that include microarrays and second-generation sequencing technologies. These technologies now enable a systematic study of biomolecular activities and provide a copious amount of genome-wide measurements. While these genome-wide measurements continue to be accumulated into numerous databases by research labs across the world, extracting 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 structural or functional similarity. This lead to the birth of *gene sets* as a new source of data leading to the development of novel algorithms

[1]The content in this chapter is largely derived from original author text and contributions found in [Judeh, 2011].

that infer biological pathway topologies 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 column corresponds to an experiment or mutant. Each entry of the matrix corresponds to the gene expression level for a given gene under a given experiment or mutant. 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 may have 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 the expression levels of the genes.

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 [Gunderson et al., 2004; Lockhart et al., 1996; Schena et al., 1995] or deep sequencing experiments [Shendure and Ji, 2008; Shendure et al., 2004]. Some representative microarray platforms include Agilent Microarray, Affymetrix GeneChip, and Illumina BeadArray.

Gene sets, on the other hand, are defined as a group of genes that share biological similarities. They are a rich source of data for reconstructing the topologies 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 a chromosome, and gene expression data (Figure 1.1). There are a variety of methods to rank gene sets including GSEA-P [Subramanian et al., 2007] and GSA [Efron and Tibshirani, 2007]. A major advantage of working with gene sets

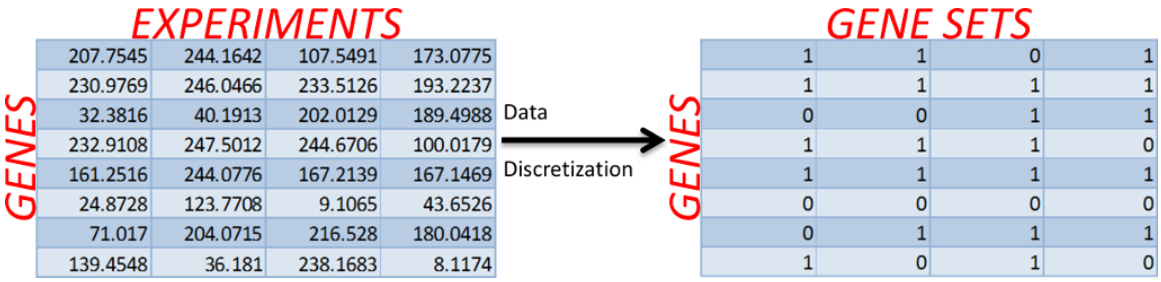


Figure 1.1: Relationship between gene expression data and gene sets. **Left:** A sample gene expression data matrix. **Right:** After data discretization, the gene expression data matrix now consists of 0's and 1's. Each column in the discretized matrix represents a gene set. For example, in the first column, the gene set consists of $\{1, 2, 4, 5, 8\}$.

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. This may be due in part that some gene sets may only capture correlation between the genes and not necessarily causality.

An important underlying assumption when trying to reconstruct a biological pathway topology 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 biological pathways using gene sets and/or gene expression data.. First, a biological pathway 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) [Alberts et al., 2002]. 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 wirings or different transduction events.

Undirected networks, on the other hand, are typically exemplified by Protein-Protein Interaction (PPI) networks [Vert, 2008]. 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 has been reconstructed, one needs to examine it at a finer level as it may be the case that only part of a biological pathway is involved in a process of interest. Thus, decomposing different biological pathways into subpathways may be needed. By retrieving the subpathways, one is able to predict gene functionality and relevant subpathways 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 profiling data and then extract biological insights about the subpathways most relevant to cancer.

Finally, given the existence of a vast amount of pathway databases including Reactome [Croft et al., 2011] and KEGG [Kanehisa and Goto, 2000; Kanehisa et al.,

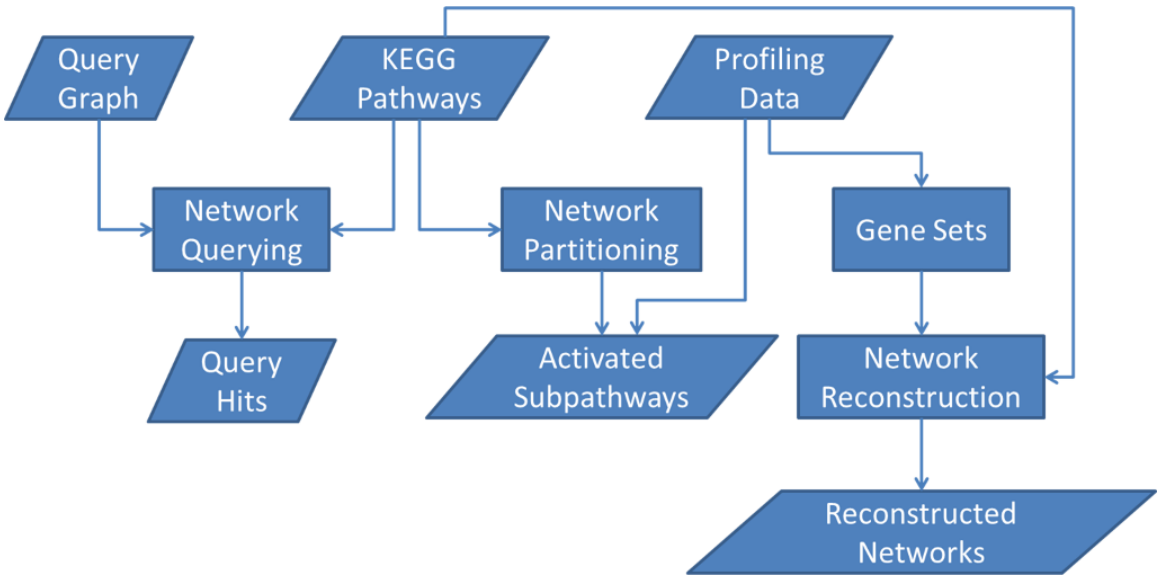


Figure 1.2: The overall framework for this dissertation. Using the KEGG pathways as an input source, three network problem domains were tackled. Starting from the right, molecular profiling data may be discretized into gene sets. Using the KEGG pathways as prior knowledge, condition specific networks may be reconstructed. In particular, the Gene Set Cultural Algorithm (GSCA) was developed to solve this problem. Furthermore, the KEGG pathways may be partitioned into smaller subpathways or subnetworks. In conjunction with molecular profiling data, activated subpathways may be returned as was done by the Topology Enrichment Analysis frameworK (TEAK). Finally, query graphs that represent biological hypotheses may be queried against the entire set of KEGG pathways to return a set of query hits as was done by the Query Structure Enrichment Analysis (QSEA) algorithm.

2012], biologists may design hypotheses in the form of query graphs. These query graphs may then be queried against the pathway databases to find support for a biological hypothesis among the pathways. Figure 1.2 succinctly summarizes the relationships amongst the various problem domains discussed in this dissertation.

To outline the remainder of this dissertation, the three problem domains will now be examined in more detail. Chapters 2 and 3 will briefly present a survey of relevant network reconstruction and network partitioning algorithms, respectively. Chapter 4 will present the Gene Set Cultural Algorithm (GSCA). Chapter 5 will present the Topology Enrichment Analysis frameworK (TEAK). Chapter 6 will present the

Query Structure Enrichment Analysis (QSEA) algorithm. Finally, Chapter 7 will conclude this dissertation with possible directions for future work. It should be noted that the work presented throughout this dissertation is largely based and derived from original author contributions in [Acharya et al., 2012a] (http://dx.doi.org/10.1002/9781118346990.ch1), [Judeh et al., 2010], [Judeh, 2011] (http://scholarworks.uno.edu/td/463/), [Judeh et al., 2012] (http://dx.doi.org/10.1145/2382936.2382997), [Judeh et al., 2013a] (http://dx.doi.org/10.1093/nar/gks1299), [Judeh et al., 2013b] (http://dx.doi.org/10.1145/2506583.2506650), and [Judeh et al., 2014].

Chapter 2: A Survey of Network Reconstruction Algorithms[2]

In the bioinformatics domain, network reconstruction algorithms may use gene expression data to reconstruct the underlying biological network. Previous approaches to reconstruct biological networks from gene expression data include Boolean or Probabilistic Boolean networks [Kaderali et al., 2009; Shmulevich et al., 2002], Bayesian networks [Friedman et al., 2000; Segal et al., 2003], mutual inference based methods [Margolin et al., 2006; Zoppoli et al., 2010], and ordinary differential equations [Bansal et al., 2006; di Bernardo et al., 2005]. While these methods may be useful, they may be unable to exploit signaling cascades illustrated in Figure 2.1. In Figure 2.1, the underlying signaling pathway may have different components activated in response to various biological conditions. Various components may be activated through linear signaling cascade mechanisms. In one paradigm, a cell membrane receptor is bounded by a growth factor. This in turn causes a signal to be transmitted to the nucleus, which results in a change in gene expression levels [Li, 2005]. In particular, the linear signaling cascades may be thought of as ordered sets of genes but are observed as unordered sets of genes. Approaches that are specifically designed for gene sets may then be of use.

Reconstructing networks from unordered gene sets or overlapping sets of occurrences has applications in different domains, including telecommunication networks [Rabbat et al., 2005, 2008]. In particular, in the bioinformatics domain, previous works for reconstructing signaling pathways from unordered gene sets include the Gene Set Gibbs Sampler (GSGS) algorithm [Acharya et al., 2012b] and the Gene Set Simulated Annealing algorithm (GSSA) [Acharya et al., 2012c]. These works focused primarily on reconstructing signaling pathway topologies and assume that each in-

[2]The content in this chapter is largely derived from original author text and contributions found in [Acharya et al., 2012a; Judeh et al., 2010; Judeh, 2011].

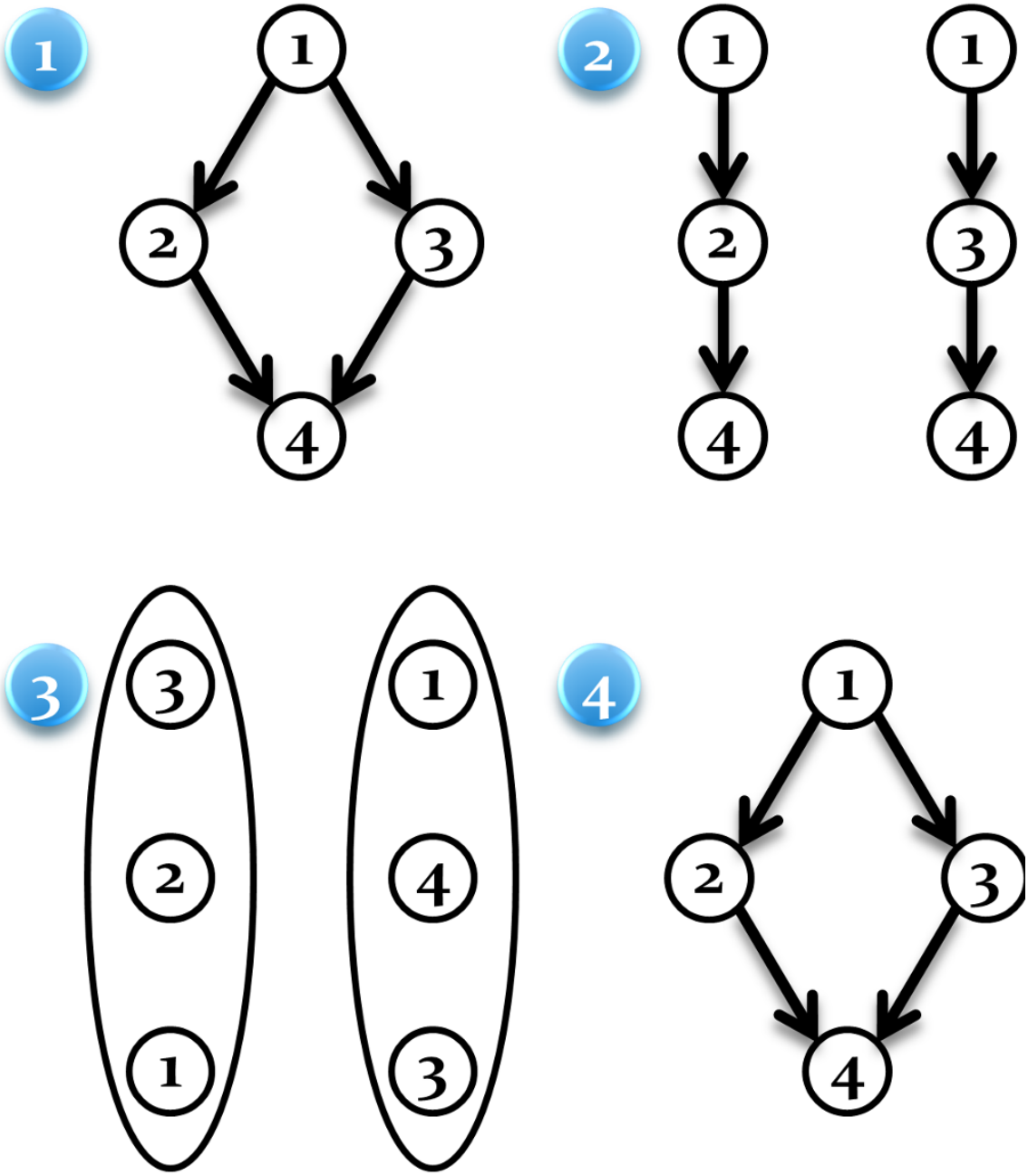


Figure 2.1: 1) The underlying signaling pathway to be reconstructed. 2) A signaling pathway may consist of several overlapping signaling transduction events that may be represented using ordered and linear chains of genes. Signal transduction events whose orders are known are denoted as ordered gene sets. 3) The indirect observed measurements are available as input as unordered gene sets. 4) Using the unordered gene sets in (3), the goal is to reconstruct the underlying network found in (1).

dividual gene set corresponds to a linear signalling cascade of events. Given enough overlapping gene sets, they sought to reconstruct the underlying signaling pathway topology that produced the gene sets.

The remainder of this chapter will now be outlined. First, Bayesian networks for reconstructing networks from gene expression data will be examined. Afterwards, the Frequency Method will be examined, which reconstructs networks from sets of co-occurrences (in their essence, co-occurrences may be though of as sets of nodes or vertices analogous to gene sets). This will then be followed by the discussion of three network reconstruction algorithms from gene sets, namely the Gene Set Gibbs Sampler (GSGS) algorithm, the Gene Set Simulated Annealing (GSSA) algorithm, and the Linear Path Augmentation (LPA) algorithm.

2.1 Bayesian Networks

A Bayesian network [Friedman et al., 2000; Needham et al., 2007] is a model that combines a graphical model and probabilistic relationships between the vertices. From a network structural view, a Bayesian network embodies the conditional dependencies and independencies 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 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, which may make an exhaustive search 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 the TEAK chapter, it may be the case that the structures of interest are already available. Thus, one may venture to say that *scoring* structures may be more important than searching in the context above. 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. For large enough N, one may use 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 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 popular in many fields, and many implementations, such as the Bayes Net Toolbox (BNT) [Murphy, 2001], exist that allow users to harness their power.

2.2 Frequency Method

The Frequency Method [Rabbat et al., 2005] is a method to reconstruct directed networks from sets of co-occurrences. It makes three important assumptions about the sets of co-occurrences. First, it assumes that tree structures in the paths correspond to the sets of co-occurrences. Another assumption is the availability of the source and destination nodes or vertices of each set of co-occurrences, which may not necessarily be known for biological systems. Finally, it is assumed that the directed edges used to form a tree in each set of co-occurrence are already available, but their order is unknown.

Using terminology similar to [Acharya et al., 2012a], 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 sets of co-occurrences, where $x_m(i) = 1$ indicates that m is involved with i^{th} set of co-occurrences. By letting s_i be the fixed beginning of the i^{th} set of co-occurrences and d_i its destination, the order of vertices

or nodes for the i^{th} set of co-occurrences 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 closest 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 a network and is computationally efficient. A major drawback is the stringent assumptions made by it such as knowing the source and destination genes of each set of co-occurrences. Furthermore, if there exist multiple paths between a pair of vertices, the Frequency Method may fail.

2.3 GSGS and GSSA

The GSGS algorithm [Acharya et al., 2012b] solved the problem illustrated in Figure 2.1 by using a Gibbs Sampler approach that inferred the order of gene sets based on partially observed networks. Given U gene sets, GSGS first fixes the order of $U - 1$ gene sets, which initially may be a random starting point for the first iteration of the algorithm. From these $U - 1$ ordered gene sets, the transition probability matrix Π and the initial probability vector π are calculated. For the u^{th} gene set that is not fixed, the likelihoods of all of the permutations or possible orderings of the gene set are calculated. The sum of the likelihoods are then normalized to 1, and the new

order for the gene set is then found by randomly selecting a permutation where the probability of selecting a permutation or order for a gene set is directly proportional to its likelihood. The process is repeated for each gene set, i.e., fixing the orders of $U - 1$ gene sets while permuting the order of the gene set under examination.

After each gene set's order is updated in the aforementioned manner, one iteration is completed for the GSGS algorithm. GSGS is than ran for a fixed amount of iterations, the burn-in stage, in hopes of reaching the underlying joint distribution. In the burn-in stage, samples that could be extracted from the iterations are discarded as it is assumed that the underlying joint distribution has not been reached. Once the burn-in stage is completed, it is assumed that the underlying joint distribution of the gene sets corresponds to the true signaling pathway topology. A specified number of samples are then drawn from the joint distribution, and each gene set is given an ordering corresponding to the most frequent ordering seen in the samples collected after the burn-in stage. The ordered gene sets are then combined to return a reconstructed network. Given the nature of the algorithm, it should be noted that the reconstructed network may possess cycles although the original underlying network may have lacked cycles. Furthermore, for gene sets of longer length (≥ 13), significant amounts of memory may be required.

GSSA [Acharya et al., 2012c], on the other hand, seeks to reach a point estimate network or topology that best fits the unordered gene sets as opposed to a distribution of likely candidate networks. In addition to assuming that the gene sets are linear, the GSSA algorithm also assumes that the end points for each gene set is fixed and known. This process may be facilitated by using known pathways such as KEGG [Kanehisa and Goto, 2000; Kanehisa et al., 2012] to place genes in different layers by using the BFS-level algorithm [Yu and Gerstein, 2006], for example, as prior knowledge. After randomly initializing the gene sets, a simulated annealing algorithm

is used to explore the neighborhoods of the current network topology at each iteration by minimizing the negative maximum log likelihood of the entire set of U gene sets by calculating the transition probability matrix Π and the initial probability vector π in a similar fashion to GSGS. To ensure a good balance between space exploration and space exploitation, the algorithm may move to a candidate network topology that fits the unordered gene sets more poorly than the current network topology. After a specified number of iterations, the algorithm terminates and returns the network with the lowest energy as the reconstructed network.

2.4 Linear Path Augmentation (LPA)[3]

LPA (Linear Path Augmentation) [Judeh et al., 2010] is a novel network reconstruction algorithm. The goal of LPA is to reconstruct an original biological network 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. As for GSGS and GSSA, Figure 2.1 provides an overview of the problem that LPA attempts to solve.

Before proceeding to the details of LPA, the details of Algorithm 2.1 used to simulate data will be examined. To be able to test a variety of algorithms, it is necessary to be able to generate some linear paths from the original network, which is accomplished by Algorithm 2.1. 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 or matrices. For

[3]The content in this section is largely derived from original author text and contributions found in [Judeh et al., 2010; Judeh, 2011].

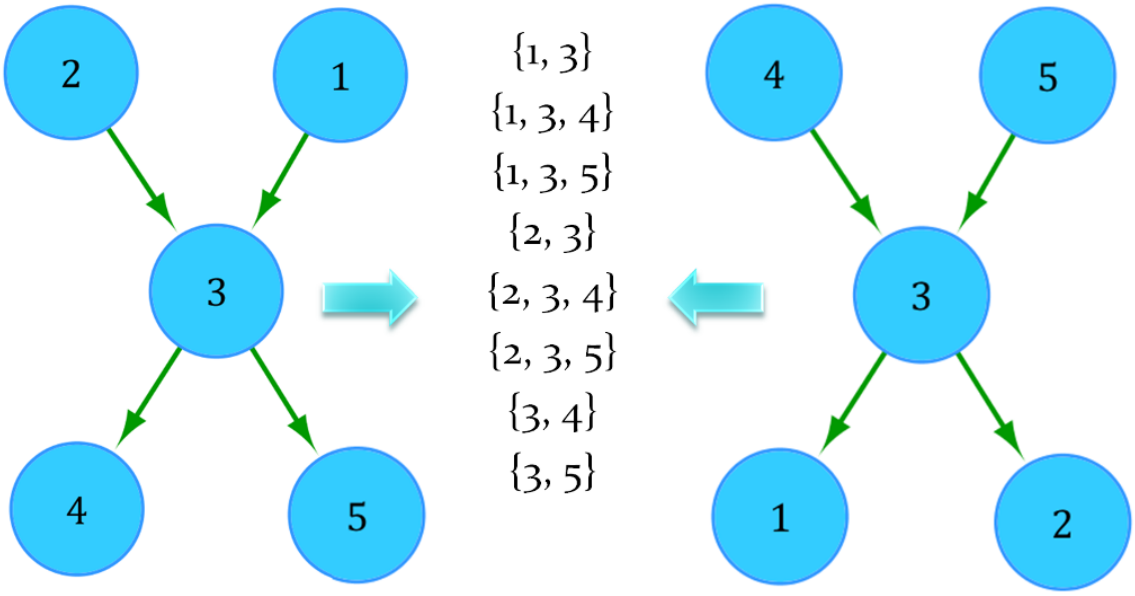


Figure 2.2: A network and its transpose. By running Algorithm 2.1 on both networks, the same set of gene sets is produced. In essence, this states that without any prior information a network and its transpose are both equal in terms of finding the final network. This phenomenon is also applicable to some types of Bayesian networks.

dense pathways or matrices, other algorithms that randomly sample simple paths may be needed.

One important note is that any network and its transpose can produce the same set of linear paths. As such, algorithms that perform network reconstruction from gene sets or sets of co-occurrences should account for this fact. At least for biological networks, though, this problem is somewhat mitigated as biologists should usually be able to easily determine the proper matrix. For example, biologists would not label a transcription factor as a leaf node. Thus, from an algorithmic perspective, some prior knowledge is necessary.

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

Algorithm 2.1: Graph to Unordered Gene Sets

Input: A directed graph G and a prune length L
Output: All unordered gene sets with lengths $\geq L$.
Remove all self-loops from G
Convert the graph G into the set of adjacency lists A
Set the visit vector V of size $|G|$ to *false*
for $i = 1, ..., |G|$ **do**
    if *vertex i has no children* **then**
        continue
    else
        Add vertex v_i to the Stack Q
        Set $V[v_i]$ to *true*
        while *S is not empty* **do**
            Let the vertex n be the top element of Q
            Remove every child c of n from the adjacency list $A[n]$ that has $V[c]$ as *true*
            if *A[n] is not empty* **then**
                Pop a child c of node n from $A[n]$
                Set $V[c]$ to *true*
                Add node c to Q
            else
                Append the contents of Q as a new information flow to the final output
                `/* Backtracking from vertex` n `*/`
                Reconstruct $A[n]$ from the graph G
                Pop the vertex n from Q
                Set $V[n]$ to *false*
            end
        end
    end
end
Prune all information flows of length $< L$ from the output.
Randomly permute both the orders of the information flows and the order of genes in each information flow.
Return all of the remaining unordered gene sets as the final output.

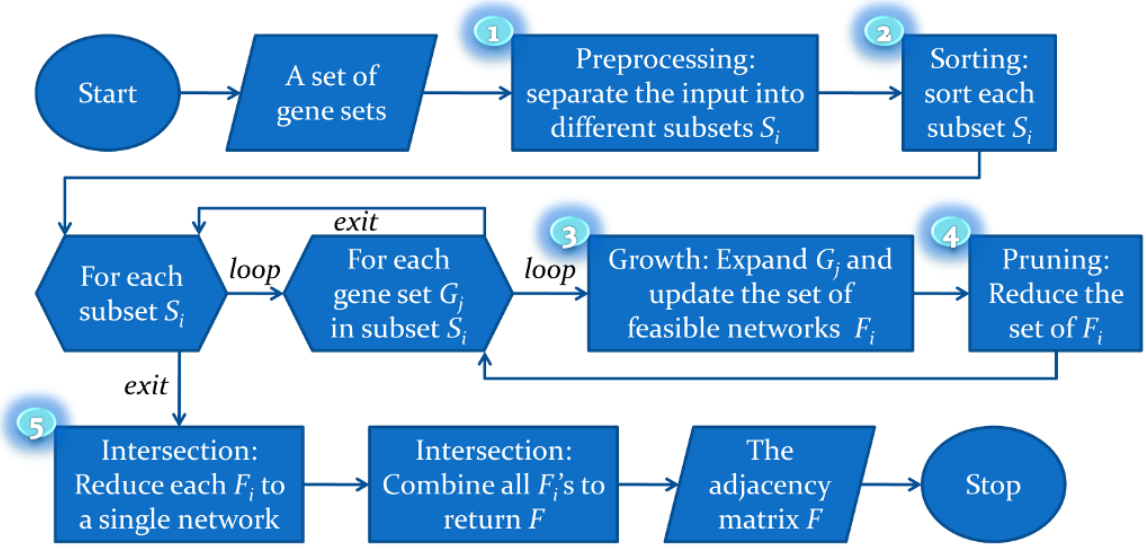


Figure 2.3: The LPA algorithm consists of five key stages. The first stage, *preprocessing*, separates the gene sets into components. The second stage, *sorting*, places the gene sets in order. The third stage, *growth*, *searches* for candidate networks. The fourth stage, *pruning*, *scores* the candidate 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.

similar to it. In addition to its novelty, it is quite modular consisting of *preprocessing*, *sorting*, *growth*, *pruning*, and *intersection* stages. This modularity allows for ease of updating stages individually. An overview of the LPA algorithm is presented in Figure 2.3.

The first stage for LPA is to preprocess the sets of gene sets. The idea behind the preprocessing stage is to divide the gene sets into different connected components. This 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. This allows for a divide and conquer approach where the next steps are run k times, once for each set of gene sets.

The second stage for LPA involves assigning 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 a 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 reach a good solution.

The growth stage is the third stage for LPA and is 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 2.2). 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.

The pruning stage is LPA's fourth stage and is 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 Algorithm 2.1 on the network, its score consists of the intersection of its unordered linear paths with the gene sets.

The fifth and final stage for LPA is needed only when some candidate network solutions still remain. 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 sum of edges in the upper triangular matrix is larger. Naturally, this process may fail when the upper triangular and lower triangular matrices of the original network have an equal number of edges.

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 necessary 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.

Although the LPA algorithm has some interesting concepts, 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 in the future.

2.5 Conclusions

In this chapter, a variety of methods for network reconstruction were briefly mentioned. For network reconstruction from gene expression data, Bayesian networks were examined. For network reconstruction from gene sets, the Gene Set Gibbs Sam-

pler algorithm, the Gene Set Simulated Annealing algorithm, and the Linear Path Augmentation algorithm were examined. Given the robustness of gene sets in consolidating data across platforms as well as their robustness against noise, it is hoped that great promise lies within gene set based methods.

In particular, due to the complexity of reconstructing networks either from gene expression data or gene sets, the usage of prior knowledge may be necessary. As such, methods that can exploit prior knowledge may be useful, especially since prior knowledge is increasingly available in the form of pathway databases such as KEGG and Reactome.

Chapter 3: A Survey of Network Partitioning Algorithms[4]

It may be the case that a reconstructed network may be too broad of a representation for a specific biological process of interest. As such, it may be prudent to decompose or partition the network into smaller subnetworks, especially given that only a specific part of a biological pathway may be activated for a biological process. A finer level of detail may be needed when examining the structure of biological pathways. Thus, decomposing a biological pathway structure into subpathways is important as subpathways may provide valuable insight into various biological processes.

In order to better address and examine subpathways, it is necessary to examine a similar concept in social networks, namely communities. 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. It is hoped that via the study of communities the extraction of subpathways is more accessible.

For detecting communities, there are two major approaches, namely *graph clustering* and *community detection* algorithms [Newman, 2006]. The former type of algorithms has its origins in computer science and other related fields. The latter type of algorithms was originally used by sociologists. It now encompass algorithms in applied mathematics, physics, and biology.

For traditional graph clustering algorithms, a user must specify the number of clusters or partitions. A graph clustering algorithm will then 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

[4]The content in this chapter is largely derived from original author text and contributions found in [Acharya et al., 2012a; Judeh, 2011].

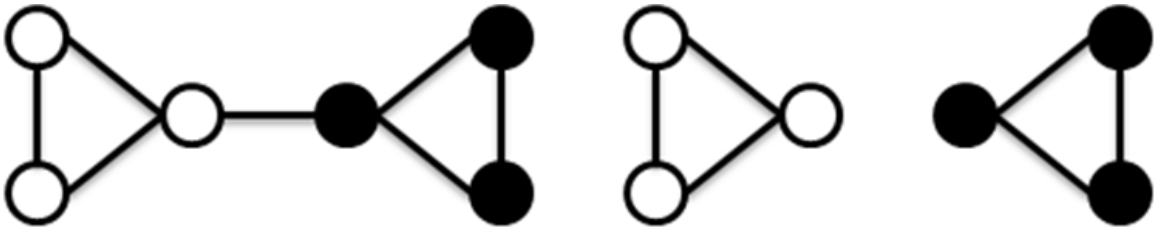


Figure 3.1: In the example illustrated above, the network displayed consists of two communities shaded white and black, respectively. Both communities exhibit high internal connections. Furthermore, there exists only a single edge connecting the two communities. As seen in the example above, the external connections between the two communities are few whereas the internal connections within the communities are plentiful.

include improving the paging properties of programs and placing the components of an electronic circuit onto printed circuit cards [Kernighan and Lin, 1970].

One may ask, "Why study *graph clustering* algorithms for biological pathways?" A major reason is that these algorithms may serve as an inspiration for community detection algorithms. For example, the Laplacian matrix used in graph clustering algorithms can be modified to perform eigenvector decomposition [Newman, 2006]. Furthermore, in one particular example, namely Newman's eigenvector method [Newman, 2006], the Kernighan-Lin algorithm [Kernighan and Lin, 1970] was an inspiration for a post-processing algorithm, namely Algorithm 3.1.

As far as community detection algorithms are concerned, the underlying assumption behind these algorithms is that a network or graph can "naturally" be divided into subpathways or communities. Thus, the subnetworks of a graph can be viewed as a topological property of the graph. This underlying shift in views is a major difference between community detection and graph clustering algorithms.

Before discussing some algorithms in detail, it may be helpful to first discuss the nature of these algorithms. Previously, a good number of algorithms in this field were designed for undirected networks and produce mutually exclusive partitions.

Algorithm 3.1: 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
  end
  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.*

Furthermore, extending an algorithm for undirected networks to directed networks may not necessarily be a trivial task [Fortunato, 2010]. In some cases, an algorithm designed for undirected networks was extended to directed networks by simply treating the directed networks as undirected networks. As seen in Figure 3.2, however, this approach may not be adequate.

It is helpful to have some baseline gold standard networks to compare the different algorithms. What constitutes a gold standard network is an area of research in and of itself. To illustrate the performance of different algorithms, Zachary's karate club [Zachary, 1977] has often been used as a gold standard network. This social network has its origins in the relationships among 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 [Kernighan and Lin, 1970] will be discussed to provide a flavor for graph clustering algorithms. This discussion will be followed by an examination of the Girvan-Newman Algorithm [Girvan and Newman, 2002; Newman and Girvan, 2004],

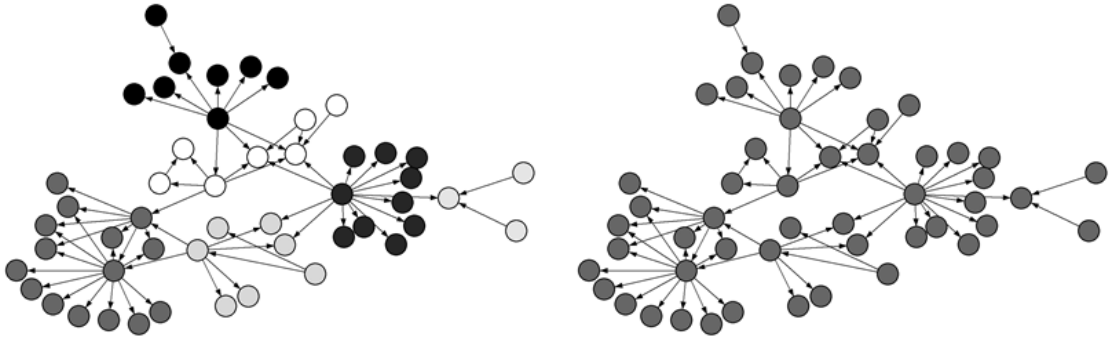


Figure 3.2: An *E. coli* network from the DREAM3 Network Challenges [Marbach et al., 2009]. (Left) Using the undirected version of InfoMap [Rosvall and Bergstrom, 2008], six communities were found when edge direction was ignored. (Right) When taking into account edge direction and using the directed version of InfoMap, no communities were found. In both cases, the appropriate version of InfoMap was run for 100,000 iterations.

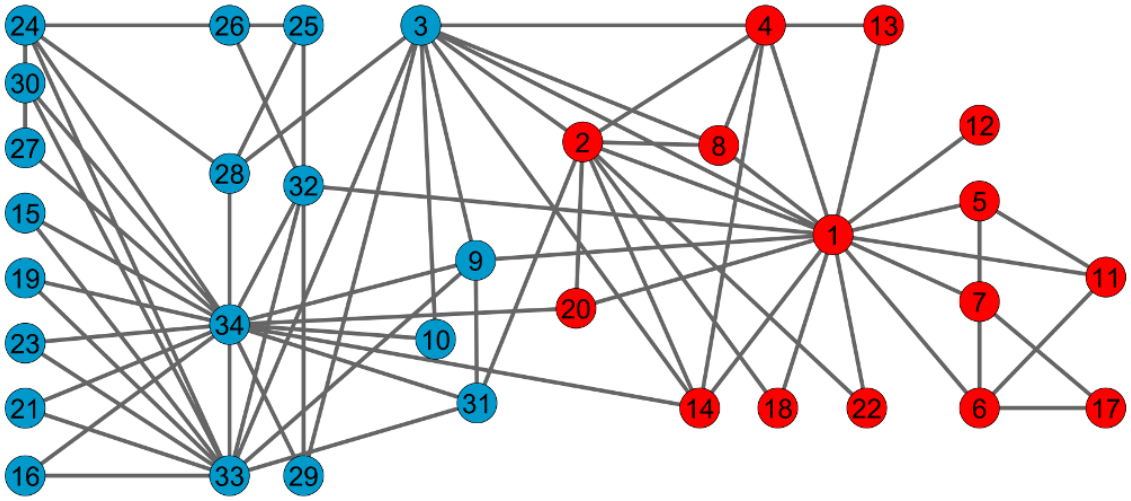


Figure 3.3: The True Partitioning of Zachary's Karate Club.

a popular community detection algorithm. Later on, Newman's eigenvector method [Newman, 2006] and Infomap [Rosvall and Bergstrom, 2008] will be examined. Finally, the Clique Percolation Method (CPM) [Palla et al., 2005] will be discussed.

3.1 Kernighan-Lin Algorithm

Developed in the 1970s, the Kernighan-Lin algorithm is a well-known graph clustering algorithm. Given its applicability, it may be used as a subroutine for other algorithms. It was initially developed in order to divide electronic circuits on boards. Since the connections between the various circuits were expensive, minimizing the number of connections between the various circuits was 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 thus to minimize the total weight of the edges connecting the k partitions.

Since the algorithm divides a network into two subnetworks, it may be applied in a recursive fashion if more clusters are needed. 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 $x \epsilon X$ is

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

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 3.2: 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.
    end
    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.
    end
until $\Gamma \leq 0$

The complexity of the Kernighan-Lin algorithm (Algorithm 3.2) is $O(|V|^2 log|V|)$. It is sensitive to the initial guesses for the subnetworks X and Y and may perform

poorly for a random initialization. As such, a different algorithm may be used to provide the initial guesses for the subnetworks, and the Kernighan-Lin algorithm is then run upon the initial subnetworks. From a biological standpoint, the Kernighan-Lin algorithm may not be very 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 subpathways which may not be biologically valid. Nevertheless, despite its origins as a graph clustering algorithm, the Kernighan-Lin algorithm did provide the inspiration for a post-processing community detection algorithm (Algorithm 3.1) developed by Newman [Newman, 2006].

3.2 Girvan-Newman Algorithm

The Girvan-Newman algorithm [Girvan and Newman, 2002] is one of the more 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 may be good at finding 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 [Newman and Girvan, 2004].

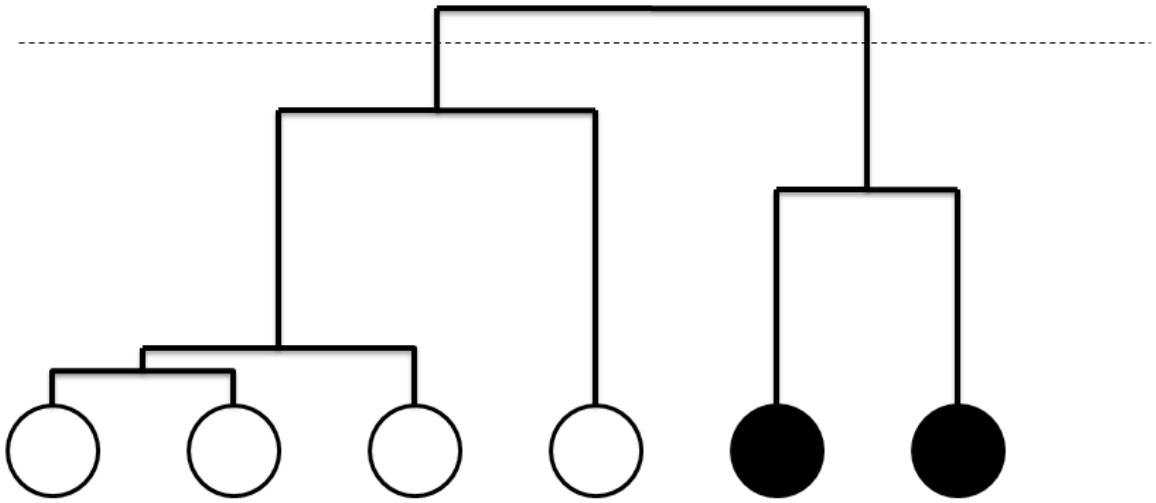


Figure 3.4: A dendrogram is produced as the output of a divisive clustering algorithm. To determine the final number of communities, the dendrogram needs to be cut. Where the dendrogram is cut is an area of research in and of itself. For this dendrogram, the given cut line divides the network into two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

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.

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 [Anthonisse, 1971; Freeman, 1977]. 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

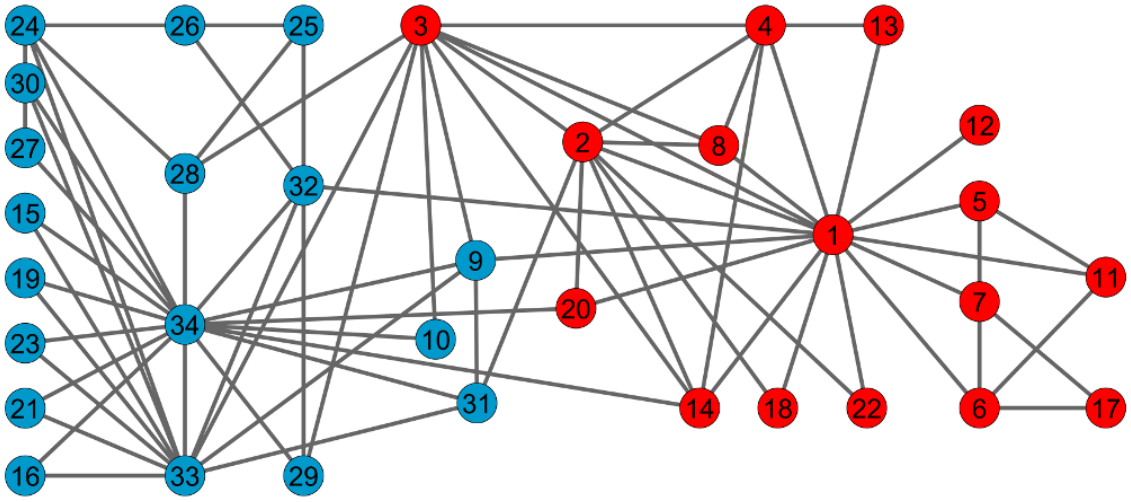


Figure 3.5: For Zachary's karate club, the Girvan-Newman algorithm [Girvan and Newman, 2002] mislabeled a single node, namely node 3.

shortest-path betweenness is that it provides the best combination of performance and accuracy [Newman and Girvan, 2004].

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 [Newman and Girvan, 2004] for details on calculating shortest-path betweenness scores for an $O(|V||E|)$ algorithm. Overall, the Girvan-Newman algorithm displayed in Algorithm 3.3 has complexity $O(|V||E|^2)$. A sample result of the Girvan-Newman algorithm on Zachary's karate club may be seen in Figure 3.5.

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 subpathways. It may be the case, though, that a researcher is only interested in the best partition amongst all available candidate partitions. Thus,

Algorithm 3.3: Girvan-Newman Algorithm

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

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 Q 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 [Fortunato, 2010], [Newman and Girvan, 2004], and [Newman, 2006].

3.3 Newman's Eigenvector Method

In the preceding section, Newman and Girvan [Newman and Girvan, 2004] introduced a new quality function called modularity in which a quality function assigns a score to a partitioning of a graph [Fortunato, 2010]. Whereas the Girvan-Newman algorithm used modularity to determine where to cut the dendrogram, there are many methods that optimize modularity directly including greedy techniques, simulated annealing, extremal optimization, and spectral optimization [Fortunato, 2010].

A major driving force behind modularity is that random graphs do not possess community structure [Fortunato, 2010]. Newman and Girvan proposed a model in which the original edges of the graph are randomly moved, but the overall expected degree of each node matches its degree in the original graph. In other words, modularity quantifies the difference of the number of edges falling within communities and the expected number of edges for an equivalent random network [Newman, 2006]. Modularity can be either negative or positive. High positive values of modularity indicate the presence of communities, and one can search for good divisions of a network by looking for partitions that have a high value for modularity. There are various modifications and formulas for modularity, but the focus for this section will be the modularity introduced by Newman [Newman, 2006].

For Newman's eigenvector method, Newman reformulates the problem by defining modularity in terms of the spectral attributes of the given graph. The eventual algorithm is very similar to a classical graph clustering algorithm called *Spectral Bisection* [Fortunato, 2010]. Suppose the graph G contains n vertices. Given a particular bipartition of the graph G, let $s_i = 1$ if vertex i belongs to the first community. If vertex i belongs to the second community, then $s_i = -1$. Let A_{ij} denote the elements of the adjacency matrix of G. Normally, A_{ij} is either 0 or 1, but it may vary for graphs where multiple edges are present. Placing edges at random in the network yields a number of expected edges $k_i k_j / 2m$ between two vertices i and j, where k_i and k_j are the degrees of their respective vertices. The number of undirected edges in the network is $m = \sum_{ij} A_{ij}/2$. The modularity Q is then defined as

$$Q = \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2m}) s_i s_j. \tag{3.5}$$

As evident from Equation 3.5, a single term in the summation of modularity equals zero if vertices i and j belong to different communities. The modularity Q can be written in condensed form as

$$Q = \frac{1}{4m} s^T \; B \; s, \tag{3.6}$$

where the column vector s has elements s_i. Here, B is a symmetric matrix called the *modularity matrix* with entries equal to

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{3.7}$$

The modularity matrix B has special properties akin to the graph Laplacian [Newman, 2006]. Each row and column sums to zero yielding an automatic eigenvector of $(1, 1, \ldots)$ with eigenvalue 0. Modularity can now be rewritten as

$$Q = \frac{1}{4m} \sum_{i=1}^{n} (u_i^T \cdot s)^2 \beta_i, \tag{3.8}$$

where u_i is a normalized eigenvector of B with eigenvalue β_i. Let u_M denote the eigenvector with the largest eigenvalue β_M. Modularity can thus be maximized by choosing the values of s, where $s_i \epsilon \{-1, 1\}$, that maximize the dot product $u_M^T \cdot s$. This occurs by setting s_i to 1 when the corresponding element $u_{M_i} \succ 0$ and -1 otherwise. Newman's eigenvector method is as follows:

Additional communities can be found by recursively applying Algorithm 3.4 to the discovered communities after a modification to Q [Newman, 2006]. Using the power method to find u_M, Newman's eigenvector method has complexity $\mathrm{O}\big(|V|^2 \log |V|\big)$, where $|V|$ is the number of vertices in the graph [Fortunato, 2010]. Newman's eigenvector method excels in its speed. Another useful property of Newman's eigenvector

Algorithm 3.4: Newman's Eigenvector Method

1: **Data**: An undirected network G.
2: **Result**: Two partitions of graph G such that the modularity Q is maximized.
3: Find the eigenvector u_M corresponding to the largest eigenvalue β_M of the modularity matrix B.
4: Let $s_i = 1$ if $u_{M_i} \succ 0$ and -1 otherwise.
5: Return two partitions X and Y. X consists of all nodes whose corresponding s_i equal to 1. Y consists of all nodes whose corresponding s_i equal to -1.

method involves the values of u_M. The value $|u_{M_i}|$ corresponds directly to the strength of node i's membership in its cluster. Newman's eigenvector method also possesses a built-in stopping criterion. For a given graph G, if there are no positive eigenvalues, then G is a community in and of itself. Its major drawback is the same as spectral bisection where the algorithm gives the best results for the initial bisection of the graph [Fortunato, 2010]. Another major drawback involves the use of modularity as a quality function.

Specifically, Fortunato [Fortunato, 2010] lists three major flaws for modularity. First, there are random graphs that may have partitions with high modularity, which undermines the very concept behind modularity. Second, modularity-based methods may suffer from a resolution limit. In other words, meaningful communities that are small with respect to the overall graph may be subsumed by larger communities. Finally, it has been shown that there exists an exponential number of partitions that have a high modularity, especially for networks possessing a strong hierarchical structure as most real networks do. Finding the global maximum may be computationally intractable.

Leicht and Newman [Leicht and Newman, 2008] later on modified Newman's eigenvector method to make it applicable towards directed networks. Leicht and

Newman first begin by modifying Equation 3.6 into

$$Q = \frac{1}{2m} s^T \ B \ s. \tag{3.9}$$

The modularity matrix B is tweaked to account for edge direction and is given by

$$B_{ij} = A_{ij} - \frac{k_i^{in} k_j^{out}}{m}, \tag{3.10}$$

where A_{ij} is 1 in the presence of an edge from node j to node i and 0 otherwise. The term k_j^{out} is the out-degree or the number of edges leaving node j, k_i^{in} is the in-degree or the number of edges entering node i and m is the total number of edges in the adjacency matrix of the graph G.

The modularity matrix B as presented in Equation 3.10 is asymmetrical, which may cause technical problems later on. To remedy this situation, the matrix B is replaced in Equation 3.9 with the sum of B and its transpose ensuring symmetry. Equation 3.9 now becomes

$$Q = \frac{1}{4m} s^T (B + B^T) s. \tag{3.11}$$

The algorithm to partition the graph G is essentially the same as Algorithm 3.4 except that the modularity matrix B defined in Equation 3.7 has been replaced with a symmetrical matrix $B + B^T$, where the latter B is defined in Equation 3.10. An advantage to this method is that essentially the underlying Newman's eigenvector method can be used unchanged except for some minor tweaks to account for edge direction. However, the given definition of modularity to incorporate edge direction is flawed. Kim et al. [Kim et al., 2010] illustrated the shortcoming of the new definition for modularity as seen in Figure 3.6.

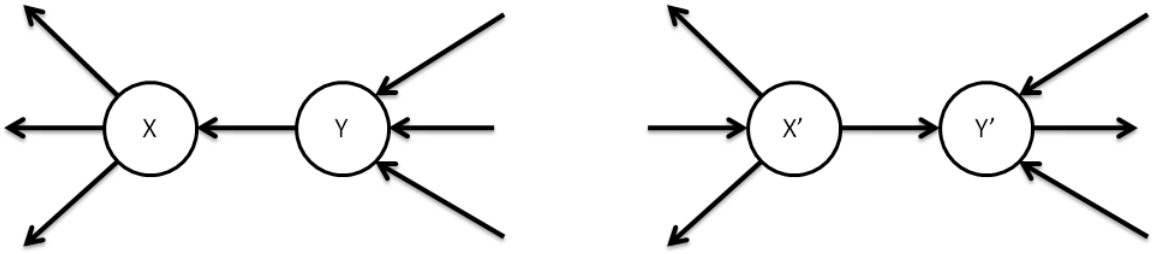


Figure 3.6: The two networks illustrate the problem with the directed version of modularity introduced by Leicht and Newman [Leicht and Newman, 2008]. The in-degrees and out-degrees for nodes X and X' are the same. The same scenario holds for Y and Y'. The result is that the directed version of modularity is unable to distinguish between the two given networks [Kim et al., 2010].

3.4 Infomap

The main idea behind Infomap [Rosvall and Bergstrom, 2008] is to identify the partitions of a graph using as minimal information as needed to provide a coarse-grain description of the graph. Infomap uses a random walk to model information flow. A community is defined as a set of nodes for which the random walker spends a considerable time traversing between them. If the communities are well-defined, a random walker does not traverse between different communities often. A two-level description for a partition M is used where unique names are given to the communities within M, but individual node names across different communities may be reused. It is akin to map design where states have unique names but cities across different states may have the same name. The names for the communities and nodes are generated using a Huffman code. A good partitioning of the network thus consists of finding an optimal coding for the network. The map equation simplifies the procedure by providing a theoretical limit of how concisely a network may be described given a partitioning of the network. Using the map equation, the actual codes for different partitions do not have to be derived in order to choose the optimal amongst them. The objective

becomes minimizing the MDL (minimum description length) of an infinite walk on the network. In other words, the MDL consists of the Shannon entropy of the random walk between communities and within communities [Fortunato, 2010]. The map equation is as follows:

$$L(M) = qH(Q) + \sum_{i=1}^{m} p^i H(P^i). \tag{3.12}$$

In the above equation, m is the number of communities. q is defined as

$$q = \sum_{i=1}^{m} q_i, \tag{3.13}$$

where each q_i is the probability per step that the random walker exits the i^{th} community. $H(Q)$ is the movement entropy between communities and is calculated as

$$H(Q) = \sum_{i=1}^{m} \frac{q_i}{\sum_{j=1}^{m} q_j} \log \frac{q_i}{\sum_{j=1}^{m} q_j}. \tag{3.14}$$

The weight of the entropy of movements within the i^{th} community, denoted by p^i, is defined as

$$p^i = q_i + \sum_{\alpha \epsilon i} p_\alpha. \tag{3.15}$$

Each p_α for node α in the i^{th} community is the ergodic node visit frequency, i.e., the average node visit frequencies for a random walk of infinite length. This is done using the power method. The entropy of movements within the i^{th} community is calculated as

$$H(P^i) = \frac{q_i}{q_i + \sum_{\beta \epsilon i} p_\beta} \log \frac{q_i}{q_i + \sum_{\beta \epsilon i} p_\beta} + \sum_{\alpha \epsilon i} \frac{p_\alpha}{q_i + \sum_{\beta \epsilon i} p_\beta} \log \frac{p_\alpha}{q_i + \sum_{\beta \epsilon i} p_\beta}. \tag{3.16}$$

Algorithm 3.5: Infomap

1: **Data**: An undirected network G.
2: **Result**: A partition M such that Equation 3.12 is minimized.
3: **repeat**
4: Assign each node into its own module.
5: Visit the modules in a random sequential order where at each module i, combine module i to the neighboring module, if it exists, that reduces Equation 3.12 the most.
6: **until** *until no move reduces Equation 3.12*

Algorithm 3.5 is the core of Infomap. There are two further subroutines that improve upon the results of the main algorithm listed in [Rosvall and Bergstrom, 2010]. The three routines run for a user-specified number of iterations. The result returned is the best partition found amongst all of the iterations. It is important to note that while modularity focuses on the pairwise relationships between nodes, Infomap focuses on the flow of information within a network [Fortunato, 2010]. This underlying difference may often cause modularity-based methods and Infomap to generate different partitions. The result of running the undirected version of Infomap on Zachary's karate club is displayed in Figure 3.7.

The extension of Infomap from the undirected case to the directed case is very straightforward. In the directed version of Infomap, a "teleportation probability" τ is introduced. With probability τ, the random walker jumps to a random node anywhere in the graph. This modification changes the undirected random walker into a directed "random surfer" akin to Google's PageRank algorithm. The default choice of 0.15 for τ is also akin to the damping factor $d = 0.85$ in Google's PageRank algorithm [Rosvall and Bergstrom, 2008]. While the map equation remains the same, the exit probabilities q_i where $q = \sum_{i=1}^{m} q_i$ and m equals the number of communities, must be updated to include the contribution of τ. The underlying algorithm remains the same.

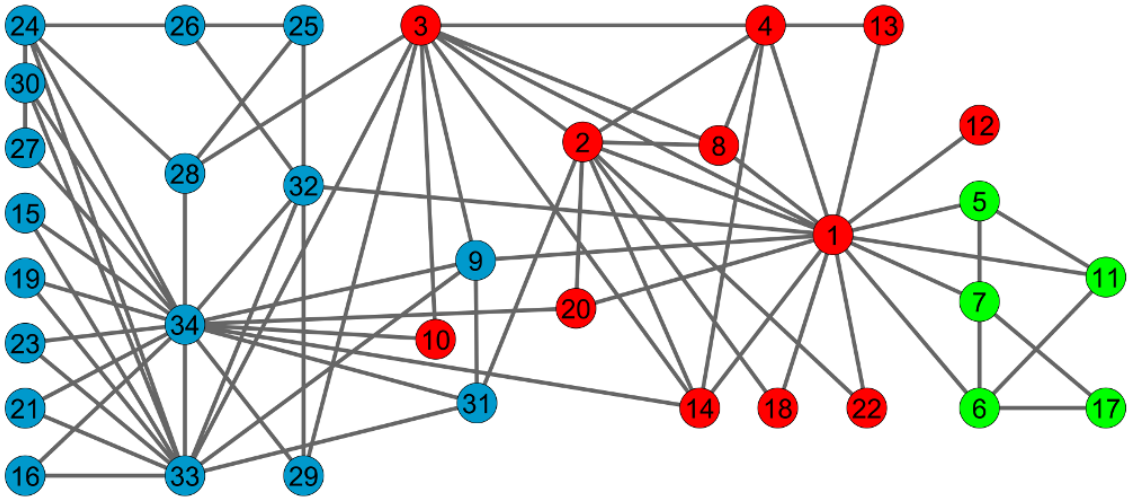


Figure 3.7: For Zachary's karate club, Infomap [Rosvall and Bergstrom, 2008] mislabeled two nodes, namely nodes 3 and 10. Furthermore, the community in red was subdivided into another community, colored in as green.

3.5 Clique Percolation Method

The Clique Percolation Method (CPM) [Palla et al., 2005] is a community detection algorithm that allows for overlapping subpathways. 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 important 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 [Bron and Kerbosch, 1973] to find all of the maximal cliques in a network. Letting the total number of cliques found be denoted as n, CPM also needs to build 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 3.6.

Algorithm 3.6: Clique Percolation Method

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

For biological pathways, one major advantage of CPM over other methods is that it allows for overlapping communities or subpathways. More importantly, Fortunato [Fortunato, 2010] has 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 nontrivial task. One potential solution to address 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 3.8 illustrates the application of CPM on Zachary's karate club.

One final note concerning the CPM algorithm is that it has a directed version called CPMd [Palla et al., 2007]. One key concept is extending the notion of k-cliques

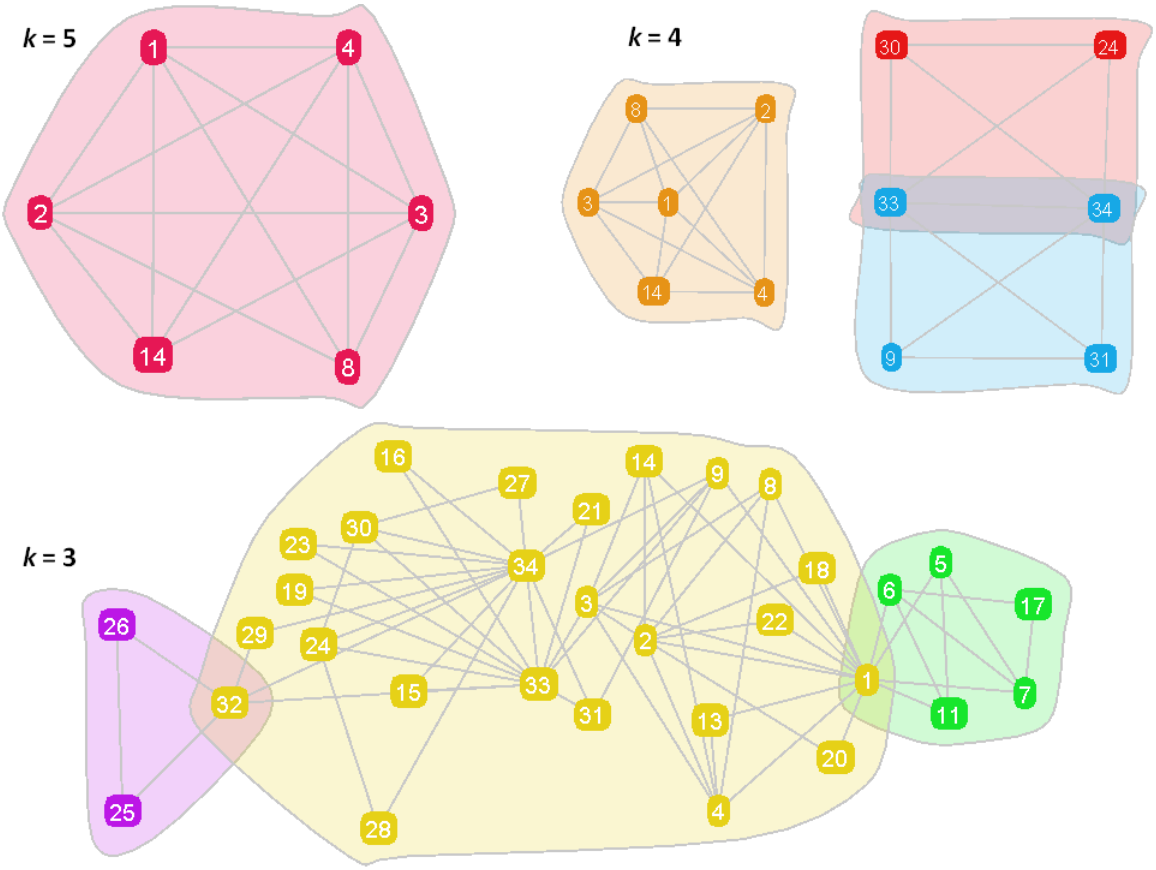


Figure 3.8: Using CFinder [Adamcsek et al., 2006] Zachary's karate club is divided into three types of communities based on their k value. At the top left for $k = 5$, only a single community is returned. At the top right for $k = 4$, three communities are returned with two communities sharing two vertices. At the bottom for $k = 3$, three communities are returned. As evident from the k values, lower values of k may lead to larger but less strongly connected communities whereas higher values of k may lead to smaller but more strongly connected communities. As evident from the figure, the partitions returned by CPM differ from the partitions seen in Figures 3.3, 3.5, and 3.7, which in turn may suggest that certain algorithms are better suited for certain types of networks over other algorithms.

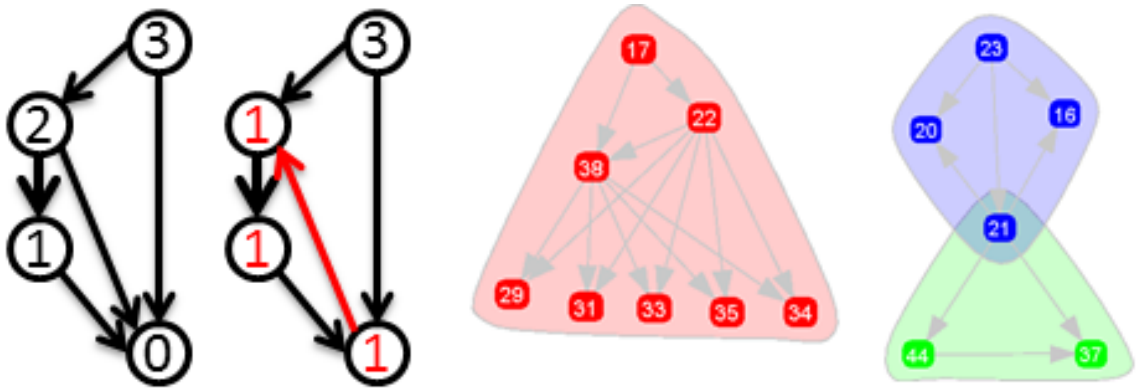


Figure 3.9: (Left) A directed acyclic graph and a directed 4-clique. The node labels refer to the outdegree of each node. (Middle-Left) While a 4-clique, it is not a directed 4-clique due to the presence of a cycle. Furthermore, it is necessary that each node has a unique outdegree in order for it to be a directed 4-clique. (Right) Using CFinder [Adamcsek et al., 2006] 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.

for undirected networks to *directed k-clique* for directed networks. 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 a directed acyclic graph, the edges of a directed k-clique always point from a node with a higher order to a node with a lower order. Equivalently, all nodes within the specified k-clique have different orders. The order of a node i within a k-clique is simply the sum of all edges leaving node i to the other nodes within the given k-clique. In [Palla et al., 2007] a directed 4-clique was shown as in Figure 3.9. Furthermore, an illustration of how CPMd works is reproduced in Figure 3.10.

3.6 Conclusions

In this chapter, a variety of algorithms for network partitioning were reviewed. The network partitioning algorithms were categorized as graph clustering algorithms and community detection algorithms. Graph clustering algorithms are applicable to Very

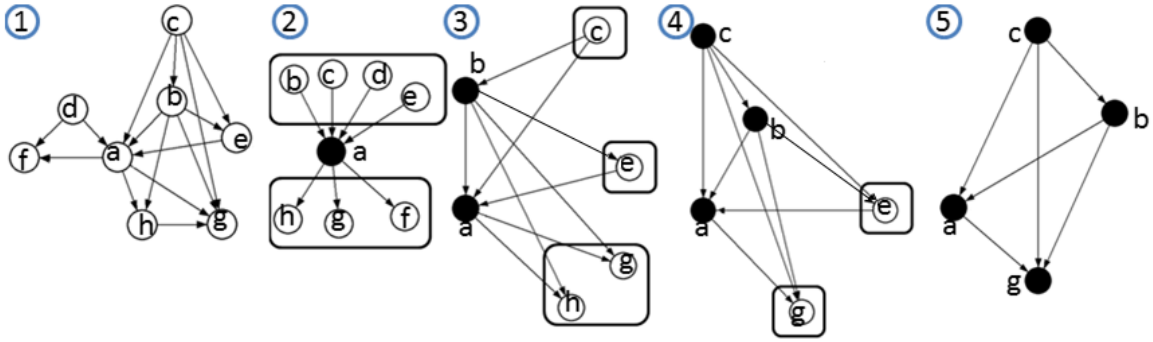


Figure 3.10: (1) The underlying network topology. (2) a is selected as the start node. The in-neighbors of a are placed in a container above a. The out-neighbors are placed in a container below a. (3) Select a new node from either container. In this case, b is selected. d and f are removed because they are not neighbors of b. e is placed in its own container as it is between a and b. (4) c is added. h is removed as it is not a neighbor of c. (5) g is added. Since e is not a neighbor of g, e is removed [Palla et al., 2007]. For graphs with cycles and a more detailed explanation of the algorithm, one may refer to [Palla et al., 2007].

Large-Scale Integration (VLSI), distributing jobs on a parallel machine, and other applications found in computer science. Community detection algorithms, on the other hand, are more applicable to biological and social networks.

For graph clustering algorithms, the Kernighan-Lin algorithm was examined. It has complexity $\mathrm{O}\big(|V|^2 \log |V|\big)$. Although the Kernighan-Lin algorithm may not be directly applicable to biological networks, its "descendant", Algorithm 3.1, is directly applicable as a post-processing step [Newman, 2006].

The first community algorithm presented was the Girvan-Newman algorithm with complexity $\mathrm{O}\big(|V||E|^2\big)$. The essence of the Girvan-Newman algorithm is that edges between communities have high edge-betweenness scores. By focusing on edge-betweenness, the Girvan-Newman algorithm focuses on the flow of the network as opposed to the immediate connection between nodes. Its major drawback is the lack of a proper criterion to determine the cut line of a dendrogram. Modularity was used to remedy the situation, but as seen later on in the chapter, modularity itself has

its own drawbacks. An interesting solution may be replacing modularity as a quality function with the map equation introduced by Rosvall.

Next, Newman's eigenvector method was examined. This method is quite interesting as by defining modularity via Equation 3.6, the modularity matrix B defined in Equation 3.7 takes the position of the graph Laplacian in the Spectral Bisection algorithm. Newman's eigenvector method is considered to be quite fast with complexity $\mathrm{O}\big(|V|^2 \log |V|\big)$. The method focuses on the degrees and connections of immediate nodes as opposed to the flow of information in a given graph. A more useful aspect is that the value of $|u_{M_i}|$ for a node i corresponds directly to its participation strength in its community. The major drawback of Newman's eigenvector method is the same as Spectral Bisection method in which its core strength lies in finding the initial bipartition of a graph. There are also drawbacks involved with the choice of modularity as the quality function. Finally, extending Equation 3.6 to its directed counterpart Equation 3.9 may not incorporate edge direction in an efficient manner.

Infomap was then examined, which utilizes information theory to compress good partitions and describe them using the least amount of bits possible. While modularity concentrates on the pairwise relationships between nodes, Infomap focuses on the flow of information within a network similar to the original Girvan-Newman algorithm.

Finally, the Clique Percolation method was examined. The Clique Percolation method is suitable for partitioning biological networks as it allows for overlap between different communities. It has some drawbacks as it may not place all nodes in a community, especially leaf nodes. The complexity of the Clique Percolation method cannot be expressed in closed form. Moreover, for the case of directed networks, the definition for directed k-clique may seem a bit arbitrary.

Chapter 4: Gene Set Cultural Algorithm (GSCA)[5]

Recently, a wave of publications has emerged incorporating pathway topologies into the analysis of molecular profiling data sets and their derivatives including Paradigm [Vaske et al., 2010], SubpathwayMiner [Li et al., 2009], and TEAK [Judeh et al., 2013a]. These approaches and others often use existing pathway database resources such as Reactome [Croft et al., 2011] and KEGG [Kanehisa and Goto, 2000; Kanehisa et al., 2012], which are manually curated from the literature. Given the abundance of gene expression data sets and their derived gene sets, novel algorithms that reliably infer biological pathways topologies may be of use. Furthermore, reconstructing a biological network may be an important piece for further analysis such as network partitioning and network querying algorithms.

Gene Set Enrichment Analysis [Subramanian et al., 2005] and Gene Set Analysis [Efron and Tibshirani, 2007] are some of the many approaches currently available that focus on the analysis of gene sets, which may be obtained via databases such as the Molecular Signatures Database [Subramanian et al., 2005] or by discretizing time series data [Li et al., 2010] and steady state data. Gene sets are more interpretable as they correspond to lists of biological processes [Klema et al., 2011] and may be thought of as derived sample features that succinctly summarize the original gene expression data [Holec et al., 2009]. Furthermore, by using gene sets, data sets from multiple platforms may be integrated [Holec et al., 2009]. These previous approaches, however, may focus only on gene sets individually in relation to gene expression data sets and may not necessarily focus on the interactions that various gene sets may have with one another. In particular, for a set of highly overlapping gene sets, sufficient information may be present that allows for the reconstruction of the underlying

[5]The content in this chapter is largely derived from original author text and contributions found in [Judeh et al., 2013b, 2014].

biological network that may have emitted the gene sets. By reconstructing the original network, the information stored across the gene sets may be aggregated into one central representation, and further analysis may then be conducted.

Prior knowledge may also be exploited and used to reduce the search space and to improve the reconstructed networks. In the work of Liu and Zhao [Liu and Zhao, 2004], gene expression data was utilized to better delineate the pathway components of the *S. cerevisiae* MAPK signaling pathway found in protein-protein interaction data. In the work of Hashemikhabir et al [Hashemikhabir et al., 2012], the problem of reconstructing a signaling pathway was framed as finding the minimum number of operations to modify a reference pathway that bests corresponded to the input RNAi data. For the Gene Set Cultural Algorithm (GSCA) , prior knowledge via the KEGG pathways may be used to hierarchically order the genes using a topological sort ordering.

4.1 Method Overview

At the heart of Gene Set Cultural Algorithm (GSCA) is Reynolds' cultural algorithm framework [Reynolds, 1979, 1994]. The cultural algorithm framework is an evolutionary computational model consisting of three major components: the population space, the belief space, and the communication protocol that allows the population space to influence the belief space and vice-versa as illustrated in Figure 4.1. Furthermore, newer versions of the cultural algorithm framework may exploit a total of five sources of knowledge [Engelbrecht, 2007; Reynolds and Gawasmeh, 2012]. The first knowledge source is situational knowledge, which is responsible for keeping track of the most fit solutions found at each generation. Normative knowledge is then used to provide guidelines and standards for individual behaviors. Domain knowledge is

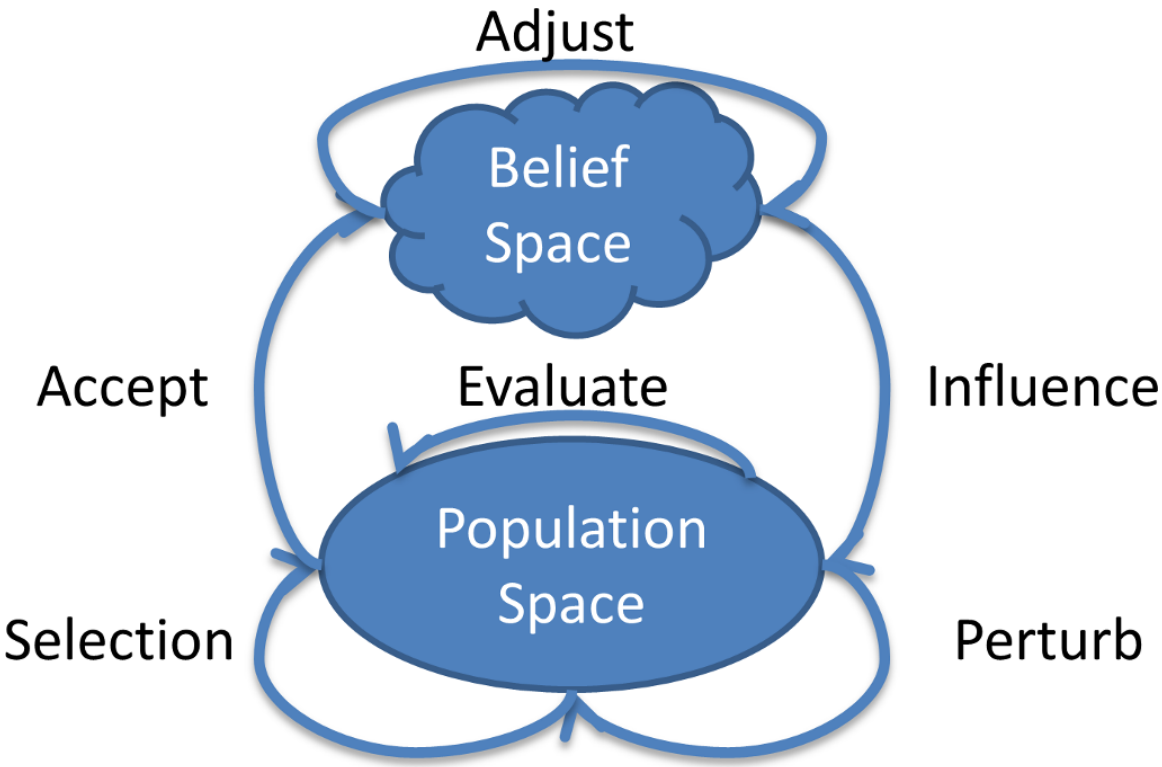


Figure 4.1: Reynolds' cultural algorithm framework. Elements in the belief space are used to influence the next generation in the population space. Elements in the population are then perturbed, and their fitness is evaluated. Some elements in the population are then selected to influence the belief space, which in turn may then be adjusted. The process then repeats itself until conditions for terminating the algorithm are met. Adapted from [Engelbrecht, 2007].

similar to situational knowledge except that it is not updated at the end of each generation. As such, prior knowledge may serve as domain knowledge. History knowledge maintains information about changes within the search space and may be modeled via the use of a tabu list [Glover et al., 1993]. Finally, topographical knowledge represents the population space as a multi-dimensional grid. Topographical knowledge can thus be used to guide a search towards unexplored areas. GSCA is able to use situational knowledge, domain knowledge, and history knowledge.

The overall framework of GSCA is presented in Figure 4.2 and Algorithm 4.1. In addition to using the cultural algorithm framework, GSCA uses topological sort

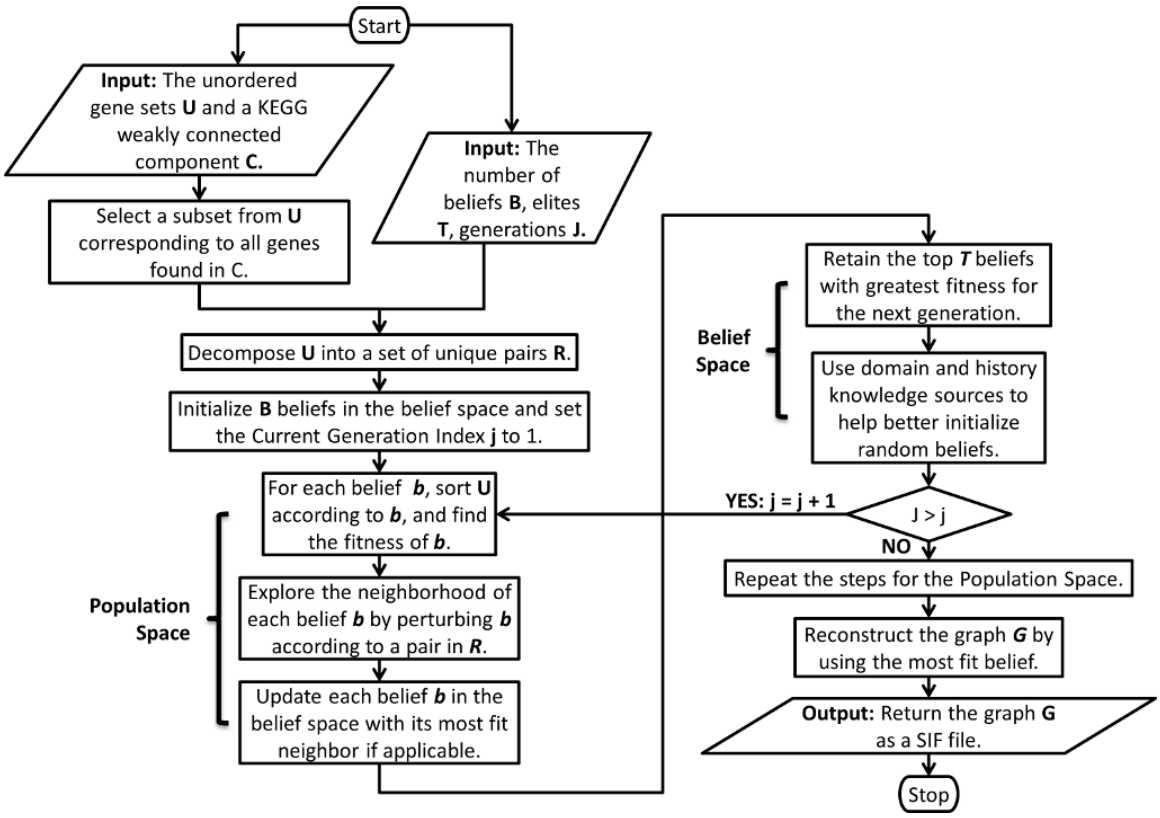


Figure 4.2: An overview of the Gene Set Cultural Algorithm (GSCA).

orderings to reconstruct a network from unordered linear gene sets. It also uses the KEGG pathways as prior knowledge to reconstruct the latent networks. It should be noted that GSCA makes an additional assumption that the unordered gene sets originated from a directed acyclic graph. While this assumption may lead to loss of representative power (for example, feedback loops cannot be represented by a directed acyclic graph), it is not overly restrictive.

Briefly, a topological sort ordering is a partial linear ordering of a network's vertices or nodes such that all directed edges go from left to right [Cormen et al., 2009]. Searching over topological sort orderings has been successfully applied to Bayesian networks [Teyssier and Koller, 2012] and is applicable for reconstructing networks from gene sets if the original network was a directed acyclic graph. Once the true

Algorithm 4.1: GSCA

Input: The unordered gene sets U, the number of search agents/ beliefs B, the number of elites T, and the number of generations J.
Output: The directed acyclic graph G of the most fit belief.
Randomly initialize B beliefs of length N in the belief space.
Set the exploration status E of all beliefs to false.
Decompose the unordered gene sets U into a set of unique pairs R.
for $j = 1, ..., J$ **do**
    for $i = 1, ..., B$ **do**
        if E_i *is true* **then**
            Continue
        end
        Let the set S be the empty set.
        Sort **U** according to a belief B_i.
        Add B_i to the set S.
        Find the fitness of B_i.
        Set the top belief B_T as B_i.
        for $k = 1, ..., R$ **do**
            Swap a pair of nodes in B_i specified by R_k to generate a new belief B_{ik}.
            if *fitness*$(B_{ik}) >$ *fitness*(B_T) **then**
                $B_T = B_{ik}$.
                Empty S.
                Add B_{ik} to S.
            else if *fitness*$(B_{ik}) =$ *fitness*(B_T) **then**
                Add B_{ik} to S.
            end
        end
        With uniform probability, randomly select B_T from S to replace B_i.
        if $B_T = B_i$ **then**
            Set E_i to *true*.
        else
            $B_i = B_T$
        end
    end
    Select the top T beliefs with best fitness values for the next generation.
    Randomly generate $B - T$ new beliefs to be added to the belief space.
    Set the exploration status E of the new beliefs as false.
end
Repeat the steps of the Population Space.
Reconstruct the output graph G from the most fit belief.

topological sort ordering is known, reconstructing the network becomes straightforward since a topological sort ordering contains the ordering information that has been previously lost. Thus, by employing an additional assumption, the problem of reconstructing a network from unordered gene sets may now be casted as finding a topological sort ordering of the original network.

The major parameters for GSCA are the number of generations or iterations J, the number of independent search agents/ beliefs B, and the number of elite beliefs to retain T. Both J and B play a role in the algorithm's complexity whereas T helps to determine the number of random topological sort orderings to be introduced into the population each generation. It should also be noted that T plays the role of the size of the situational knowledge preserved at the end of each generation in GSCA where a smaller value of T will lead to greater exploration as $B - T$ new topological sort orderings are introduced. However, a smaller value of T may also lead to lack of exploitation of fit topological sort orderings. A balance between exploration and exploitation is sought by fixing T to be $B/2$.

4.2 The Belief and Population Spaces

GSCA proceeds by dividing the unordered gene sets U into a set of unique pairs R. R is bounded by $O(N(N-1)/2)$ where N is the number of unique nodes or genes in U. Via the use of R, one is able to define a neighborhood for a topological sort ordering by randomly swapping a pair of nodes in a topological sort ordering. For example, if the unordered gene sets are $\{(1,2,3,4),(2,3,4,5)\}$, then R consists of the pairs $\{(1,2),(1,3),(1,4),(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)\}$. If the topological sort ordering is $(1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5)$ and the pair from R is $(1,2)$, the topological sort ordering is then changed to $(2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5)$. Furthermore, by limiting the

pair swaps to R, one may avoid swapping a pair of genes that are not present together within at least a single gene set. Using the example above, $(1, 5)$ will be considered an invalid swap since 1 and 5 are not present together in at least one gene set.

GSCA then proceeds to initialize the belief space. The belief space is represented by B topological sort orderings, which are then transferred into B search agents whose neighborhoods are explored. In the absence of prior knowledge, the belief space is randomly initialized to B different topological sort orderings. If prior knowledge were available from pathway databases such as KEGG, any cycles or strongly connected components are first removed from the pathway. In particular, the heuristic from Query Structure Enrichment Analysis (QSEA) [Judeh et al., 2012] is used. After removing any applicable cycles, the transitive closure of the prior knowledge is calculated and stored. A topological sort ordering based on prior knowledge is then constructed by iteratively selecting one of the roots of the prior knowledge's transitive closure with uniform probability. Upon selecting a root, the root and all of its edges are removed. The root is then added to the end of a topological sort ordering. The process of repeatedly selecting a root with uniform probability and removing all applicable edges is repeated until all edges are removed. By using this procedure, a topological sort ordering that obeys prior knowledge is extracted and retrieved. A simple example illustrating this procedure is illustrated in Figure 4.3.

At this point, GSCA enters its population space. In the population space, each search agent/ belief B_i or topological sort ordering has its neighborhood explored by applying a unique pair from R one at a time and swapping the corresponding nodes in B_i. If a pair swap from R leads to neighboring belief that contradicts with prior knowledge, the neighboring belief is discard. To achieve this goal, the transitive closure of the prior knowledge matrix is calculated. For a neighbor of a B_i, it is first reversed. Then all weak orders in the reversed belief are checked against the transitive

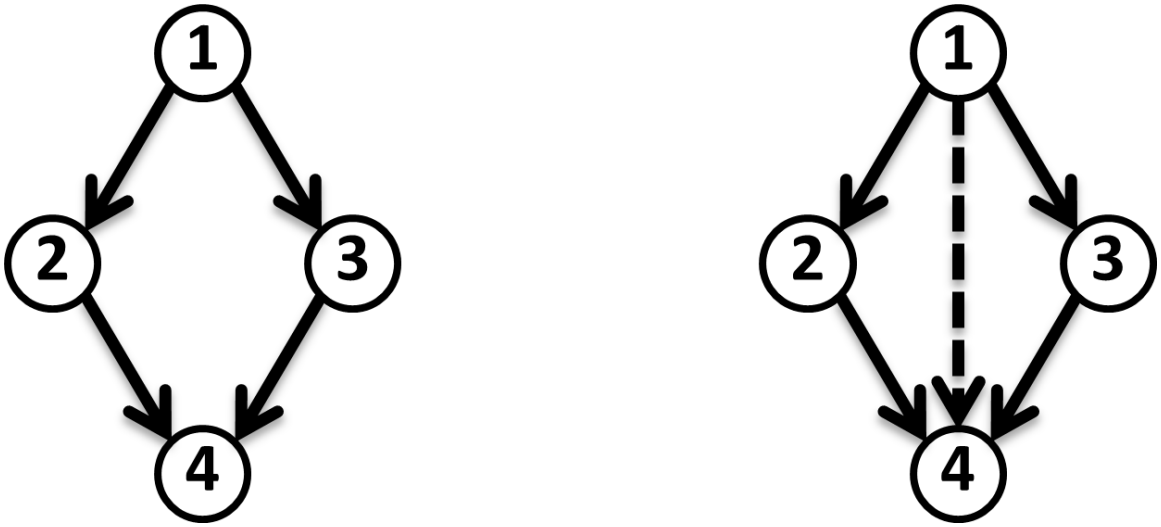


Figure 4.3: An example graph (left) and its transitive closure (right). In the beginning, the only root in the transitive closure is 1. 1 is added to the start of the topological sort ordering. After removing 1 and all of its edges, the vertices 2 and 3 are roots. With uniform probability, one of them is selected to be added to the topological sort ordering. Suppose 3 was added to yield the partially constructed topological sort ordering ($1 \rightarrow 3$). Then, 3 and all of its edges are removed. At the next step, 2 is the root, so it is added to yield ($1 \rightarrow 3 \rightarrow 2$). After removing 2 and all of its edges, only 4 remains. After adding 4, the topological sort ordering is ($1 \rightarrow 3 \rightarrow 2 \rightarrow 4$). The algorithm terminates as no vertices remain in the graph. It should be noted that using the aforementioned procedure, another valid topological sort ordering, ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4$), can also be generated.

closure of the prior knowledge matrix. If any weak order from the reversed belief is found to exist in the transitive closure of the prior knowledge, it is determined that random belief goes against prior knowledge and is thus discarded.

For each belief B_i and its applicable neighbors, the fitness is calculated by sorting the unordered gene sets U according to each topological sort order. A transition matrix

$$M = [c_{xy}]_{N \times N} \tag{4.1}$$

is first reconstructed from the ordered gene sets where c_{xy} is the count of node x appearing directly before node y across all ordered gene sets. The matrix M is very

similar to the transition probability matrix Π used by the GSGS and GSSA algorithms except that its rows are not normalized to sums of 1. The rationale behind this action is to preserve magnitude information found in the counts, which is otherwise lost if M is transformed into a transition probability matrix.

After reconstructing M, a heuristic based on the Chu-Liu [Chu and Liu, 1965] and Edmonds' algorithms [Edmonds, 1967] is used. Briefly, the Chu-Liu and Edmonds' algorithms allow one to find the maximum weighted arborescence of a directed graph. An arborescence is a graph where for a root vertex x and its descendant y, there is exactly one path from x to y. As such, an arborescence may take the form of a directed rooted tree where all edges point away from a root x. Based on the implementation used by GSCA, it also possible to generate a forest of directed trees. It should be noted that the concept of arborescences for directed graphs is analogous to the concept of spanning trees for undirected graphs. Since a reconstructed M corresponds to a directed acyclic graph, there is no need to check for cycles. The fitness score FS is calculated as

$$FS = \frac{\Sigma_{n=1}^{N}(max(M_{n\cdot}) + max(M_{\cdot n}))}{|M_E|} \tag{4.2}$$

where $M_{n\cdot}$ corresponds to the n^{th} row of M, $M_{\cdot n}$ corresponds to the n^{th} column of M, and $|M_E|$ corresponds to the number of edges or nonzero elements in M. As such, Equation 4.2 can be interpreted as calculating the sum of the arborescences of M and its transpose while dividing by the number of edges in M to favor more sparse networks.

The searching in the population space thus influences the belief space where a belief B_i or its neighbor with highest fitness score FS is promoted to the belief space B to influence the next generation. At this stage, both history and domain

knowledge, if available, may be used to guide the choice of random topological sort orderings. For the history knowledge component, a tabu list is used to keep track of all beliefs or topological sort orderings last seen within a window of fixed size. The use of the tabu list thus helps to avoid visiting recently explored beliefs and in turn yields a more efficient search. Domain knowledge may be available through the use of the KEGG pathways, for example. Thus, using both history knowledge in the form of a tabu list and domain knowledge in the form of prior knowledge may better guide the search for the underlying network. The belief space B is then exited after introducing $B - T$ random topological sort orderings to avoid being trapped in local peaks.

GSCA concludes after $J-1$ generations or iterations have been reached. Since GSCA begins with the belief space, the steps for the population space are undertaken one more time. After entering the population space for the last time, the output graph G may be reconstructed using a number of ways. For the purposes of this chapter, the most fit belief B_i or topological sort ordering is used to order the unordered gene sets U. After ordering U, one can simply trace the linear paths in U to add edges to reconstruct the output graph G.

4.3 Heuristic Fitness Function Justification

The choice of Equation 4.2 is now justified. To test the performance of Equation 4.2, four *E. coli* networks and five *Insilico* networks were extracted from GeneNetWeaver [Schaffter et al., 2011] corresponding to gold standard networks from DREAM3 and DREAM4 [Marbach et al., 2009, 2010; Prill et al., 2010]. Furthermore, it should be noted that the heuristic for QSEA [Judeh et al., 2012] was used to preprocess and remove feedback arc sets for the *Insilico* networks. After exhaustively generating all

Table 4.1: DREAM3 and DREAM4 Network Statistics

Network	$\|V\|$[a]	$\|E\|$[b]	Diameter[c]	Max[d]	$\|U\|$[e]	% Used[f]
E. coli 1	27	33	4	5	125	100%
E. coli 2	30	35	3	4	34	100%
E. coli 3	48	53	4	5	141	100%
E. coli 4	42	47	3	5	114	100%
Insilico[g]1	82	107	5	7	150	41.10%
Insilico 2	93	178	6	7	150	28.90%
Insilico 3	98	173	10	17	150	3.56%
Insilico 4	97	176	9	14	150	3.02%
Insilico 5	93	171	9	11	150	5.33%

[a] the number of vertices in the network
[b] the number of edges in the network
[c] the network diameter
[d] the length of the longest gene set in the network
[e] the number of gene sets available for the network
[f] the percentage of the gene sets used for the network
[g] Feedback arcs sets were removed for all *Insilico* networks.

simple paths of the DREAM3 and DREAM4 gold standard networks, all gene sets of length 2 were removed. The networks were then reconstructed from the pruned gene sets. All gene sets for the *E. coli* networks were used whereas 150 gene sets for the *Insilico* networks were randomly sampled. Some summary statistics of the reconstructed networks are displayed in Table 4.1.

In Figures 4.4 and 4.5, $1,000,000$ random topological sort orderings were generated (with replacement), and the gene sets were ordered according to a random topological sort ordering and scored. The two score functions used include GSCA's Equation 4.2 as well as the log of the maximum likelihood function used by both GSSA and GSGS. After sorting the gene sets according to a given topological sort ordering, both the fitness score used by GSCA and the maximum likelihood score were calculated for the underlying network topologies. The scores for GSCA were scaled to $(0, 1]$ by dividing by the maximum score for each network for each plot. The

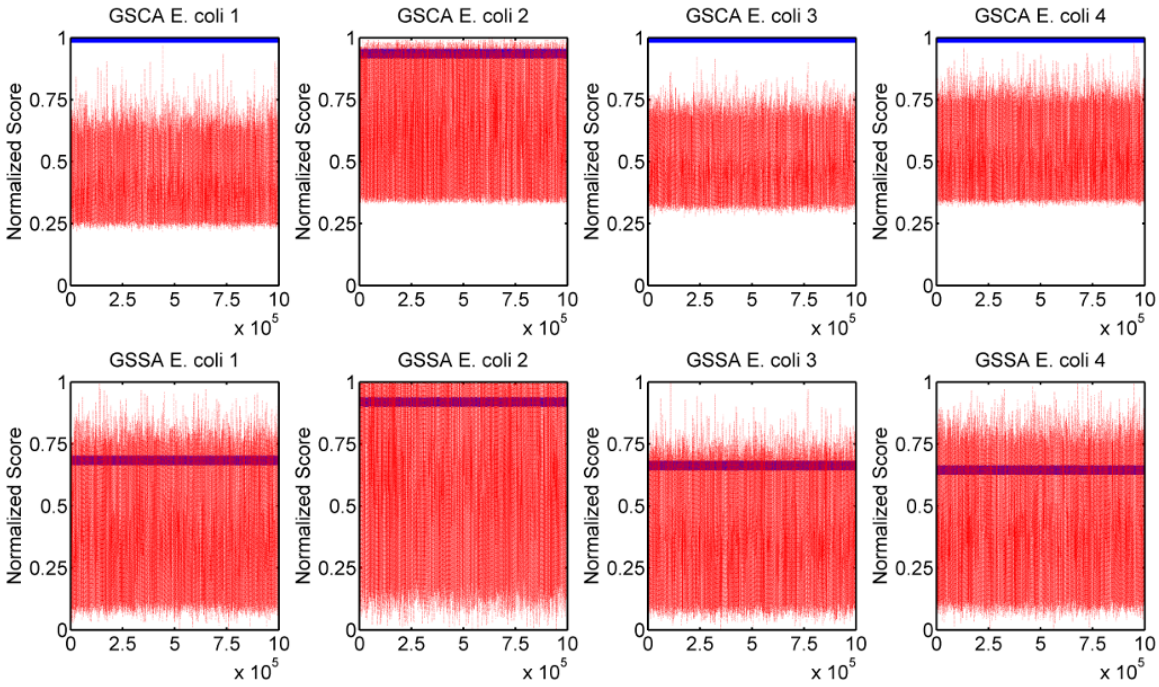


Figure 4.4: GSCA's Equation 4.2 versus the log of the maximum likelihood function used by both GSSA and GSGS for four *E. coli* networks from the DREAM3 and DREAM4 initiatives [Marbach et al., 2009, 2010; Prill et al., 2010]. $1,000,000$ random topological sort orderings were generated for all networks. Equation 4.2 performs similarly to the log of the maximum likelihood across all networks.

log of the maximum likelihood scores were scaled to $[0, 1]$ by shifting the scores by the maximal likelihood score to the right and by then dividing by the maximum score of the shifted scores for each network for each plot. The fitness of random topological sort orderings are represented as red dots whereas the fitness of the true topological sort ordering is represented by a solid blue line.

For Equation 4.2, only the *E. coli* 2 network had 0.2959% of random topological sort orderings dominating the true topological sort ordering whereas for all other networks, none of the scores of the random topological sort orderings dominated the scores of the true topological sort orderings. For the maximum likelihood function, on the other hand, the number of random topological sort orderings dominating the score of the true topological sort orderings were 0.7631%, 1.8777%, 0.3938%, 2.1189%,

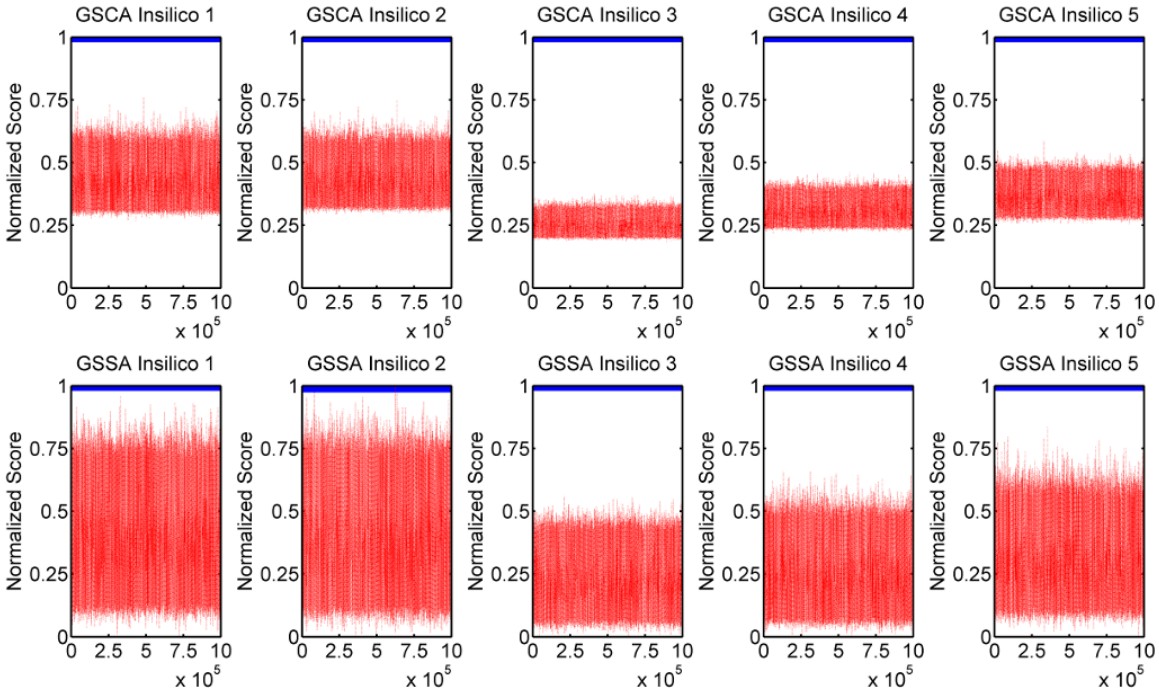


Figure 4.5: GSCA's Equation 4.2 versus the log of the maximum likelihood function used by both GSSA and GSGS for five *Insilico* networks from the DREAM3 and DREAM4 initiatives [Marbach et al., 2009, 2010; Prill et al., 2010]. $1,000,000$ random topological sort orderings were generated for all networks. Equation 4.2 performs similarly to the log of the maximum likelihood across all networks.

and 0.0001% for the *E. coli* 1, *E. coli* 2, *E. coli* 3, *E. coli* 4, and *Insilico* 2 networks, respectively. As such, it may be inferred that when ample or sparse gene sets are available, Equation 4.2 performs similarly to the maximum likelihood function.

Furthermore, one may note that GSCA's score has a lower bound of 2*(the number of nodes in the reconstructed network - 1) / (the number of transitions in the gene sets). In the case of the *E. coli* 1 network, the network size is 27, and the number of transitions in the gene sets is 349. As such, GSCA's *E. coli* 1 score is bounded below by 2*27/349 = 0.1490. One is subtracted from the total number of nodes since there must be at least one root and one leaf for the reconstructed network to be a DAG. In reality, the actual minimum score may be higher as the above assumes that

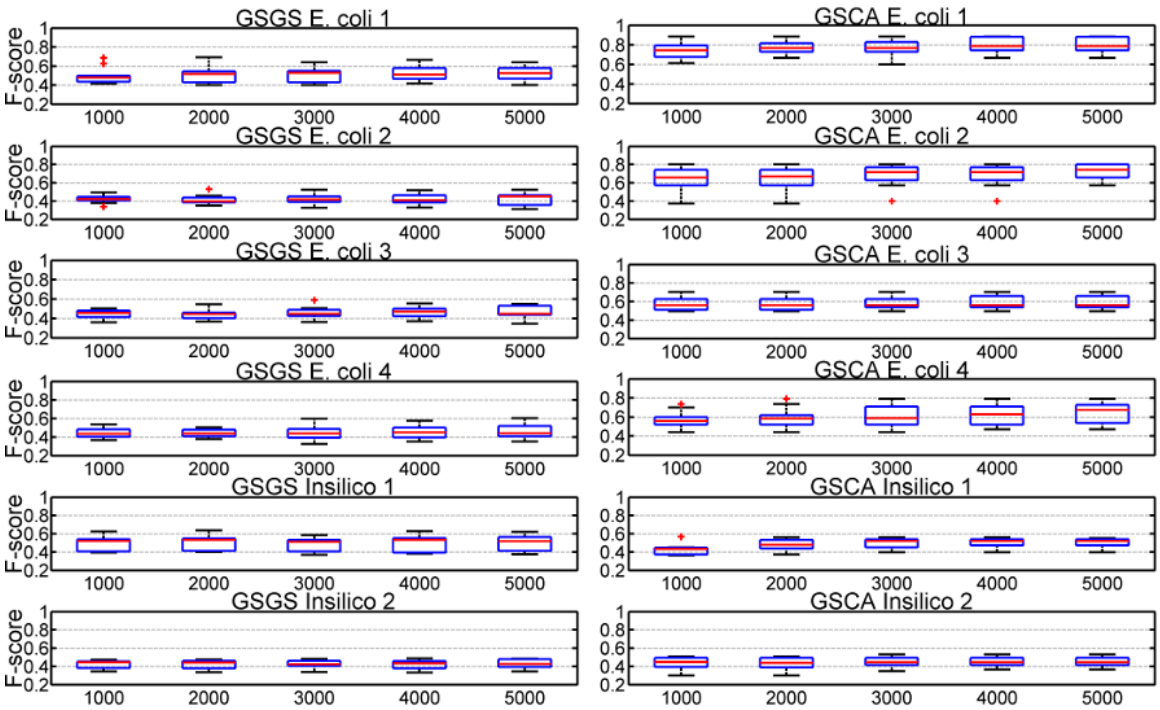


Figure 4.6: A comparison of the performance of GSGS and GSCA. On the y-axis, the $F\text{-}score = 2 * Sensitivity * PPV/(Sensitivity + PPV)$ is measured. On the x-axis, snapshots of the performance of the GSGS and GSCA algorithm at varying number of iterations or generations is presented. Overall, GSCA outperforms (the *E. coli* networks) or performs similarly to the GSGS algorithm.

all edges in the reconstructed network have weight one and that each transition in the gene sets translates into a unique edge.

4.4 Simulated Data Analysis

In Figure 4.6, the performances of GSGS and GSCA were compared. GSSA was not used since knowing the end terminals of gene sets in conjunction with the DAG assumption made by GSCA may be sufficient to reconstruct the underlying network in the presence of ample gene sets. The primary parameters for GSGS are the number of iterations for the burn-in stage and the number of samples to collect after the burn-in stage is completed. Briefly, the burn-in stage is part of the Gene Set Gibbs Sampler

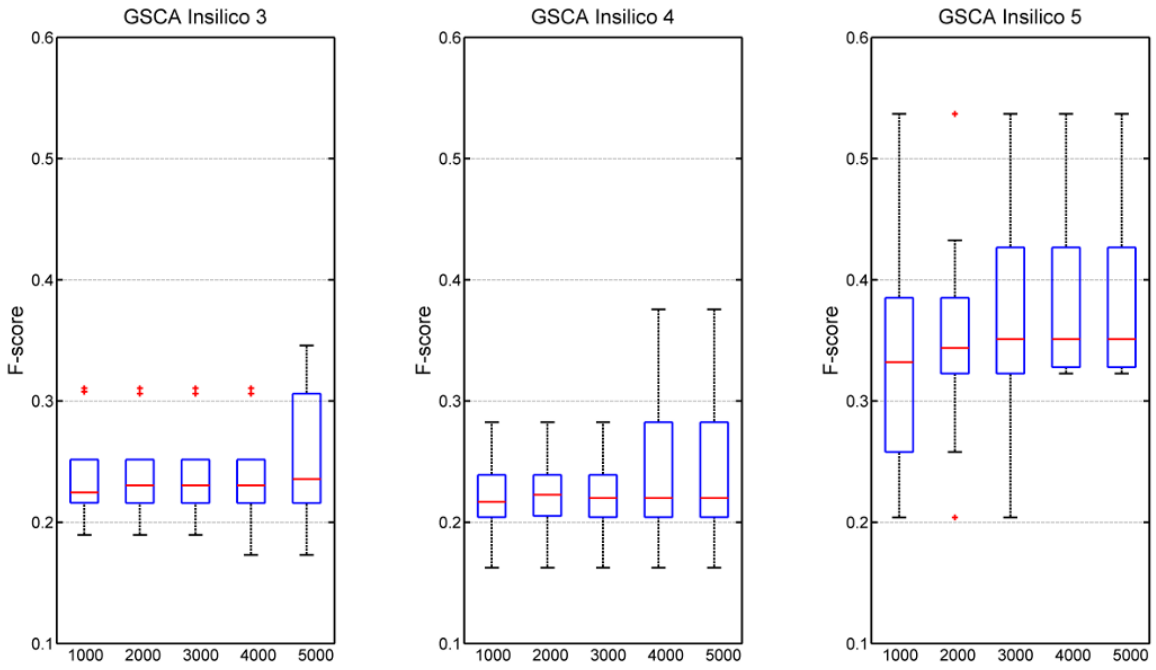


Figure 4.7: The performance of the GSCA algorithm for three *Insilico* networks. GSGS was unable to run due to insufficient memory on a workstation with 4 GB of RAM.

algorithm that discards the results of the initial iterations as the joint distribution of gene sets moves to what is hoped to be the true distribution. As for GSCA, the relevant parameters are the number of generations J, the number of beliefs/search agents B, and the number of elite beliefs to retain after each generation T.

For each network in Figure 4.6, GSGS and GSCA were run 10 times each on randomly ordered gene sets of sizes described in Table 4.1. The parameters for the GSGS algorithm were $5,000$ iterations each for both the burn-in state and for sample collection for a total of $10,000$ iterations for each run. For the $5,000$ iterations of sample collection, networks were reconstructed after the collection of $1,000$, $2,000$, $3,000$, $4,000$, and $5,000$ samples. For GSCA, it was run for a total of $5,000$ generations or iterations for each run. The number of search agents/ beliefs B was set to 10, and the number of elite solutions T preserved after each generation was set to 5.

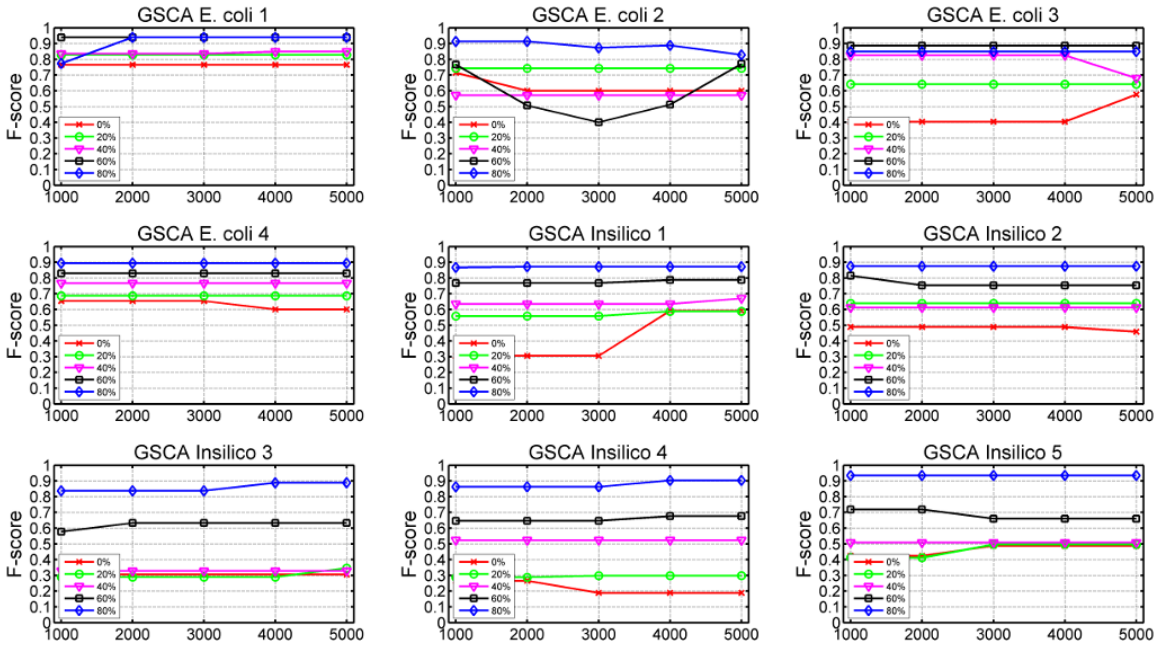


Figure 4.8: The use of prior knowledge for the nine DREAM3 and DREAM4 networks. On the y-axis, the $F\text{-}score = 2 * Sensitivity * PPV/(Sensitivity + PPV)$ is measured. On the x-axis, snapshots of the performance of the GSCA algorithm at varying number of generations is presented. The lines in red represent no prior knowledge. The lines in green represent 20% prior knowledge. The lines in pink represent 40% prior knowledge. The lines in black represent 60% prior knowledge, and the lines in blue represent 80% prior knowledge. As can be seen overall, prior knowledge leads to an overall better performance for GSCA.

The size of the tabu list was set to $100,000$ beliefs. Similar to GSGS, $5,000$ generations were run, and the $F\text{-}score = 2 * Sensitivity * PPV/(Sensitivity + PPV)$ for the most fit belief was calculated after $1,000$, $2,000$, $3,000$, $4,000$, and $5,000$ generations. *Sensitivity* is calculated as the number of true positives, i.e., the number of predicted edges agreeing with true edges, divided by the total number of true edges. *PPV* or the *Positive Predictive Value* is the number of true positives divided by the total number of predicted edges. In particular, additional iterations for sample collection do not lead to vastly improved results for the GSGS algorithm as illustrated in Figure 4.6. For the GSCA plots, a "learning curve" may be observed for the E. coli 1, 2, and 4 networks. As seen in Figure 4.6, GSCA outperforms GSGS across all four *E.*

coli networks and performs similarly for two *Insilico* networks. In addition, for three *Insilico* networks, results were presented only for GSCA in Figure 4.7 as the memory requirements for GSGS were cost prohibitive for a workstation with 4 GB of RAM.

Finally, the use of prior knowledge for the DREAM3 and DREAM4 networks may be seen in Figure 4.8. Prior knowledge was obtained by randomly sampling the specified percentage of edges from the networks presented in Table 4.1. On the y-axis, the $F\text{-}score = 2 * Sensitivity * PPV/(Sensitivity + PPV)$ is measured. On the x-axis, snapshots of the performance of the GSCA algorithm at varying number of generations is presented. The lines in red represent no prior knowledge. The lines in green represent 20% prior knowledge. The lines in purple represent 40% prior knowledge. The lines in black represent 60% prior knowledge, and the lines in blue represent 80% prior knowledge. As can be seen overall, prior knowledge leads to an overall better performance for GSCA.

4.5 Real Data Analysis

For real data analysis, the well-studied compendium of 5,350 genes and 300 expression profiles corresponding to diverse mutations and chemical treatments in the budding yeast *S. cerevisiae* [Hughes et al., 2000] was used. Using the MTBA toolbox [Jayesh Kumar Gupta], the Cheng and Church algorithm [Cheng and Church, 2000] was used on the non-log scaled fold change data matrix to produce three biclusters. In particular, the bicluster consisting of 4,826 genes and 274 samples was selected for further analysis. Prior knowledge corresponding to the largest weakly connected component of the KEGG Cell Cycle pathway was used. Genes present in the weakly connected component were discretized as 1 if the absolute value of their $\log_{10}$ fold change ratios were greater than or equal to $\log_{10}(2)$ and 0 otherwise. After converting

the discretized data into gene sets, 23 gene sets with lengths ranging from 2 to 6 were extracted and in conjunction with the prior knowledge present in the KEGG Cell Cycle weakly connected component, GSCA was run for 50,000 iterations. As seen in Figure 4.9, GSCA preserves most of the weak order information found in the prior knowledge.

4.6 Conclusions

In this chapter, the Gene Set Cultural Algorithm (GSCA) for reconstructing networks from unordered sets of genes was presented. The primary focus of GSCA is to search the space of topological sort orderings that may represent the underlying network from which the gene sets may have originated. A simulation study of the performance of the heuristic used as the fitness function algorithm for nine DREAM3 and DREAM4 networks was presented. Simulation studies for the performance of GSCA across nine simulated sets of gene sets for the aforementioned networks from the DREAM initiatives was also presented. Finally, a case study involving the use of 300 gene expression profiles was presented. The network reconstructed using GSCA preserved most of the weak order information found in the KEGG network utilized as prior knowledge.

The approach presented here is useful since it robustly incorporates and exploits prior knowledge. Furthermore, each search agent/belief acts independently of one another in the search space allowing for a rather straightforward extension to threaded programming. The results produced by GSCA may also be thought of a set of weak orders. From this angle, the output of GSCA may then be used by other algorithms, such as the Bayesian based K2 algorithm, that rely upon a robust starting point to produce good results. As such, future hybrid algorithms may examine

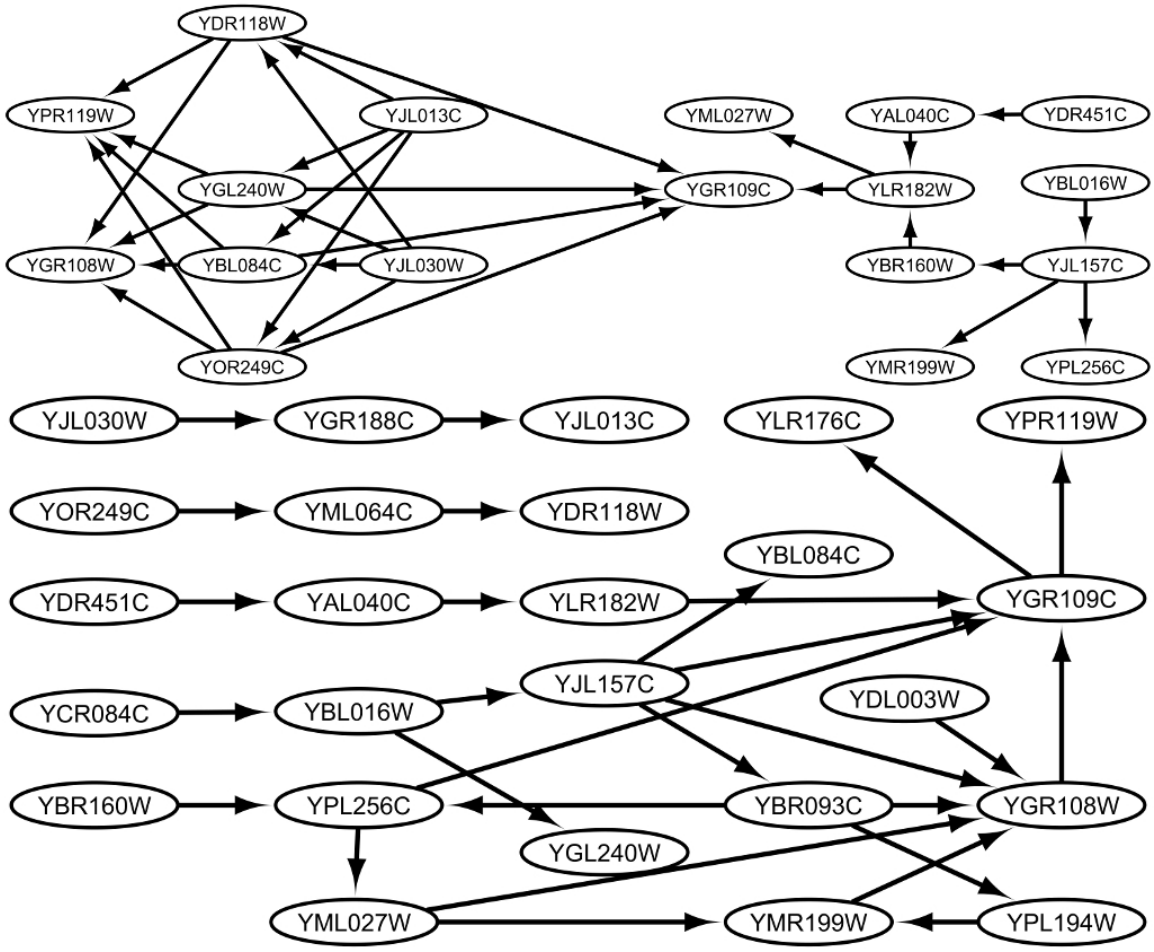


Figure 4.9: **Top:** From KEGG the following *S. cerevisiae* Cell Cycle Pathway was used. **Bottom:** The network reconstructed by GSCA using the prior knowledge and the 23 out of the 300 *S. cerevisiae* samples from Hughes et al [Hughes et al., 2000] comprising of 25 genes. GSCA preserves 17 weak order pairs extracted from the prior knowledge in its reconstructed network. Namely, it preserves the following: YDR451C to YAL040C, YAL040C to YGR109C, YBL016W to YGR109C, YBR160W to YGR109C, YDR451C to YGR109C, YJL157C to YGR109C, YLR182W to YGR109C, YBL016W to YJL157C, YAL040C to YLR182W, YDR451C to YLR182W, YBL016W to YML027W, YBR160W to YML027W, YJL157C to YML027W, YBL016W to YMR199W, YJL157C to YMR199W, YBL016W to YPL256C, and YJL157C to YPL256C.

the data both from the aspects of gene sets (column view of the data) as well as the individual genes (row view of the data). Furthermore, future work may consist of examining in detail the relationships between the number of generations J, the number of beliefs B, and the number of elite beliefs T in hopes of finding an automated method of tuning the parameters based on the data set being used.

Chapter 5: Topology Enrichment Analysis frameworK (TEAK)[6]

From an application perspective, network partitioning can be used to get a finer scale of detail when combining gene expression data and existing pathways. Given the exponential growth of high-dimensional gene expression data, biologists need versatile tools at their disposal to quickly extract important biological insights from their data since the pace of data accumulation far exceeds the pace of analysis. Simultaneously, many resources are now available describing the pathways of many different biological processes including KEGG [Kanehisa and Goto, 2000; Kanehisa et al., 2012], Reactome [Croft et al., 2011; Matthews et al., 2009], and Biocarta (www.biocarta.com). The increasing availability of high-throughput gene expression data and high-fidelity pathways has led to an evolution in bioinformatics analysis from the analysis of single genes to gene sets and now to subpathways.

Before presenting the main approach for this chapter, it is prudent to review some of the existing methods currently present in the literature. A classical approach for analyzing high-dimensional gene expression data is to use an over representation approach (ORA). Many methods exist [Khatri and Draghici, 2005] including Pathway Processor [Grosu et al., 2002], PathMAPA [Pan et al., 2003], PathwayMiner [Pandey et al., 2004], ArrayXPath [Chung et al., 2004], GenMAPP [Dahlquist et al., 2002], and LVPP (Low Variance Pathway Predicator) [Zhu, 2009]. In an ORA approach, one typically analyzes the number of differentially expressed genes within a pathway gene set against the number of genes expected to be found by chance. While these previous approaches are useful, they may fail to take into account the inherent regulatory relationships found in biological pathways among the different genes. Biological pathways are effectively reduced to sets of gene sets using an ORA approach. In other

[6]The content in this chapter is largely derived from original author text and contributions found in [Judeh et al., 2013a].

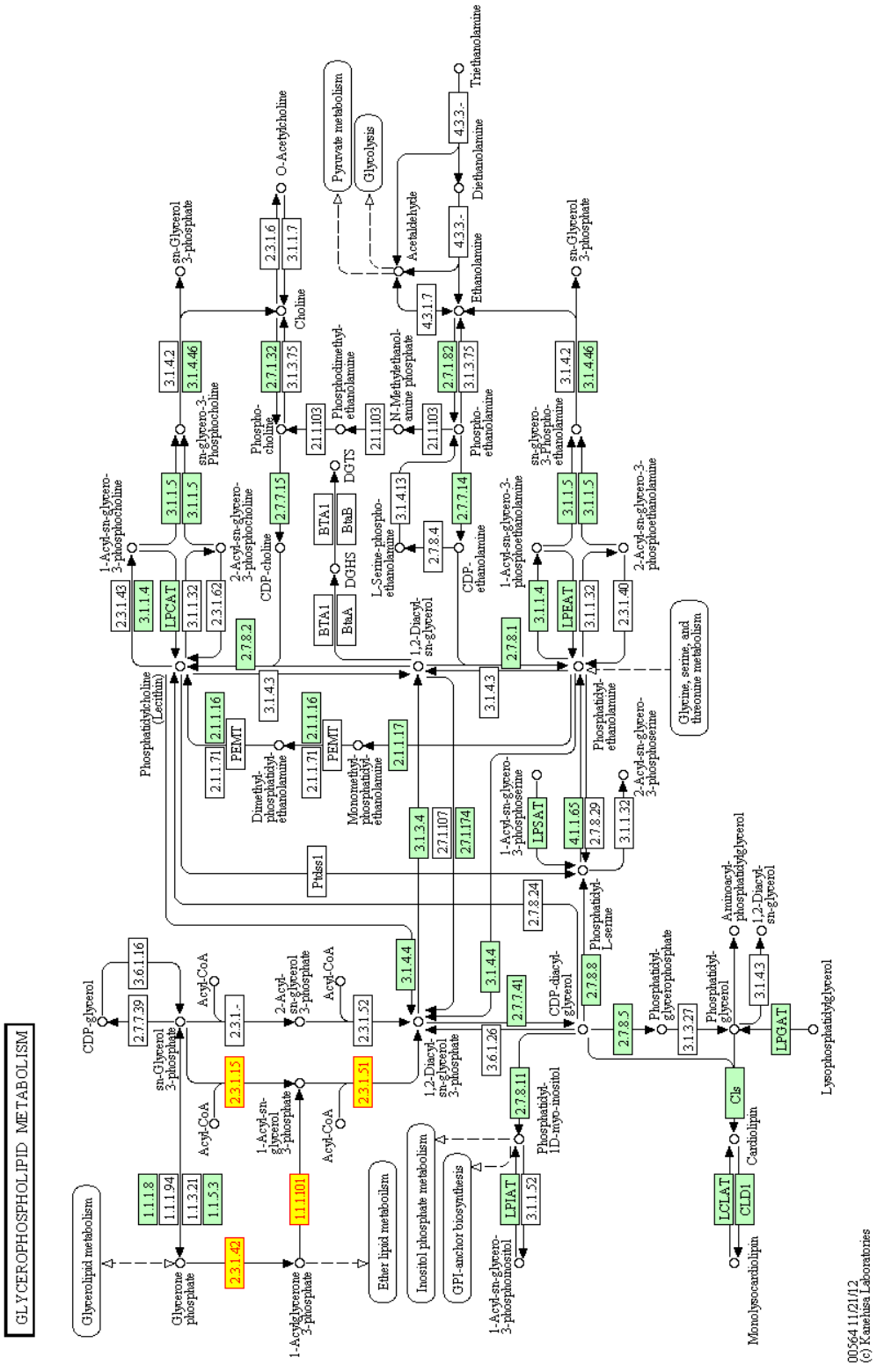


Figure 5.1: TEAK result.

words, a rich source of information, namely pathway topology, remains untapped and unused.

More recent tools including SPIA [Tarca et al., 2009] and Paradigm [Vaske et al., 2010] use pathway topologies. Whole pathways, however, may be too broad to represent some biological processes that may instead be succinctly represented by subpathways. One approach, SubpathwayMiner [Li et al., 2009], extracts k-clique subpathways, i.e., the distance between any two nodes in a subpathway is not larger than k. Another approach [Chen et al., 2011] focuses on extracting linear subpathways using a depth-first search (DFS) algorithm. While the focus on subpathways is laudable, the approaches mentioned may be limited since a hypergeometric test is used for SubpathwayMiner and a Euclidean-based measure is used for the latter approach. Such approaches may fail to fully capture the underlying topological information present in biological pathways as permuting the structure of the subpathways using either approach will yield the same results. Currently, frameworks that combine both approaches have not been extensively studied.

The contribution of this chapter, the Topology Enrichment Analysis frameworK (TEAK), seeks to detect activated subpathways underlying biological processes. TEAK uses an in-house developed graph traversal algorithm to extract all root to leaf linear subpathways of a given pathway while it uses a tailor-made Clique Percolation Method [Palla et al., 2005, 2007] for nonlinear subpathways. For subpathways activated under a specific context or condition, e.g., a single data matrix corresponding to time series data or a set of samples corresponding to relevant mutants, TEAK deploys the Bayesian Information Criterion [Schwarz, 1978] implemented in the Bayes Net Toolbox (BNT) [Murphy, 2001] to fully capture the topological information and regulatory relationships inherent in both linear and nonlinear subpathways. For differential subpathways between case and control conditions, TEAK uses the Kullback-

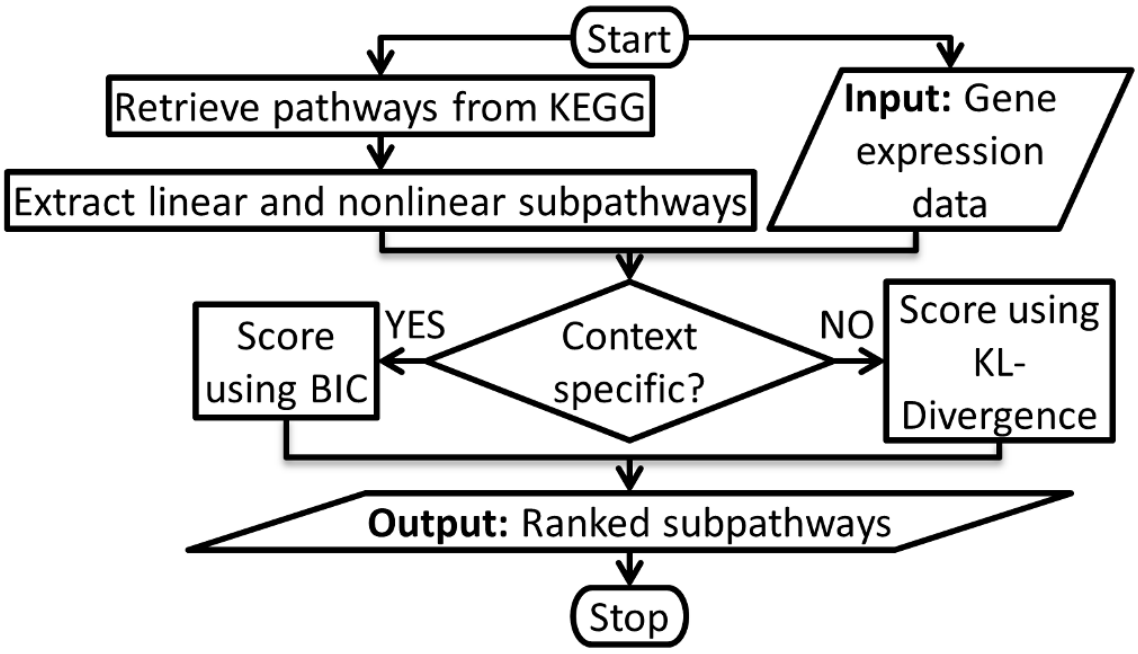


Figure 5.2: TEAK Overview. TEAK requires gene expression data using one of the many label systems supported by KEGG including Entrez, NCBI-GI, and ORF (for *S. cerevisiae*) as input. By default TEAK includes a set of subpathways for *H. sapiens*, *M. musculus*, and *S. cerevisiae*. For other organisms, the extraction of subpathways from the KEGG pathways needs to be done once or as needed. For context specific data, TEAK uses the Bayesian Information Criterion to score the Gaussian Bayesian network (GBN) fitted for each subpathway. For case-control data, TEAK first fits two GBNs, a case network and a control network, for each subpathway. Via TEAK's GUI, a user can then display a subpathway highlighted in its parent pathway as seen in Figure 5.1.

Leibler divergence of two Gaussian Bayesian networks (GBNs), i.e., a case subpathway and a control subpathway, transformed into their multivariate Gaussian forms to score each subpathway. TEAK thus provides an innovative view of the data from a fresh angle allowing users to visualize a subpathway within its respective parent pathway as illustrated in Figure 5.1.

Edge Subtype(s)	ECrel	PPrel	GErel	Directed
compound	*	*		NO
hidden compound	*			NO
1) activation and 2) inhibition		*		YES
1) expression and 2) repression			*	YES
indirect effect		*	*	YES
state change		*		YES
1) binding/association and 2) dissociation		*		NO
missing interaction		*	*	N/A
phosphorylation		*		YES
dephosphorylation		*		YES
glycosylation		*		YES
ubiquitination		*		YES
methylation		*		YES

Table 5.2: A reproduction of the edge subtypes table found in the KGML documentation at `http://www.kegg.jp/kegg/xml/docs/` [Kanehisa and Goto, 2000; Kanehisa et al., 2012]. From the KGML documentation, ECrel is defined as an "enzyme-enzyme relation, indicating two enzymes catalyzing successive reaction steps." PPrel is defined as a "protein-protein interaction, such as binding and modification." Finally, GErel is defined as a "gene expression interaction, indicating relation of transcription factor and target gene product." For each edge subtype, TEAK uses the "Directed" column to determine whether or not to treat the edge as directed or undirected.

5.1 Method Overview

Figure 5.2 outlines TEAK. Using the KEGG API [Kanehisa and Goto, 2000; Kanehisa et al., 2012], TEAK first fetches all metabolic and nonmetabolic pathways for the organism under study. TEAK extracts a subset of the KEGG pathways consisting of gene products and/or complexes of gene products as nodes. For edges TEAK extracts all KEGG enzyme-enzyme relations, protein-protein interactions, and gene expression interactions to create a set of unweighted adjacency matrices to represent the KEGG pathways (one may refer to Table 5.2 for more details). Post publication, support has been extended to compound networks. This process, including the extraction of linear and nonlinear subpathways, occurs only once per organism or as needed, and

its results are pre-computed and included by default for *H. sapiens*, *M. musculus*, *S. cerevisiae*, and many other model organisms.

5.2 Subpathway Extraction

Subpathways play a major role in biological processes since only part of a pathway may be activated at a specific time given an underlying condition. Often times a biological condition may be controlled by a specific subpathway, but the subpathway's contribution may be obscured by its parent pathway. As such, subpathway extraction is an important component of TEAK. TEAK extracts two types of subpathways: linear and nonlinear. Linear subpathways are represented by root to leaf linear paths of a pathway whereas nonlinear subpathways are represented by a union of adjacent and overlapping feed-forward loops. The underlying reasoning and procedures for extracting subpathways of these types will now be examined.

Algorithm 5.1, based on the Algorithm 2.1, extracts root to leaf linear paths or subpathways from the KEGG nonmetabolic pathways. An example is presented in Figure 5.3. In this example, the following steps are taken: (1) At the beginning, the visit status of all nodes are false, and the stack is empty. (2) 1 is visited since it is a root and is added to the stack. (3) From the unvisited children of 1, 2 is selected and is removed from 1's adjacency list. 2's visit status is set to true and is added to the stack. (4) 3 is visited from 2, and its visit status is set to true. 3 is removed from 2's adjacency list and is added to the stack. Since 3 is a leaf node, output the stack contents, i.e., $1 \rightarrow 2 \rightarrow 3$, as a root to leaf linear path. (5) 3 has no unvisited children. Backtrack to 2, pop 3 from the stack, and set its visit status to false. (6) 2 has no unvisited children. Backtrack to 1, reconstruct 2's adjacency list, set 2's visit status to false, and pop 2 from the stack. (7) Visit 3 from 1, and remove 3 from

Algorithm 5.1: Linear Subpathways

```
1: Input: An unweighted, directed graph G
2: Output: All root to leaf linear subpathways
3: Remove all self-loops from G
4: Convert the graph G into the set of adjacency lists A
5: Set the visit vector V of size |G| to false
6: Find the roots R and leaves L of graph G
7: for j = 1,...,|R| do
8:     Add root r_j to the Stack S
9:     Set V[r_j] to true
10:    while S is not empty do
11:        Let the node n be the top element of S
12:        Remove every child c of n from the adjacency list A[n] that has V[c]
           as true
13:        if A[n] is not empty then
14:            Pop a child c of node n from A[n]
15:            Set V[c] to true
16:            Add node c to S
17:        else
18:            if n is a member of L then
19:                Append the contents of S as a new subpathway to the final
                   output
20:            else
21:                Reconstruct A[n] using the graph G
22:            end
23:            Pop a node from S
24:            Set V[n] to false
25:        end
26:    end
27: end
28: Return all of the subpathways extracted as the final output
```

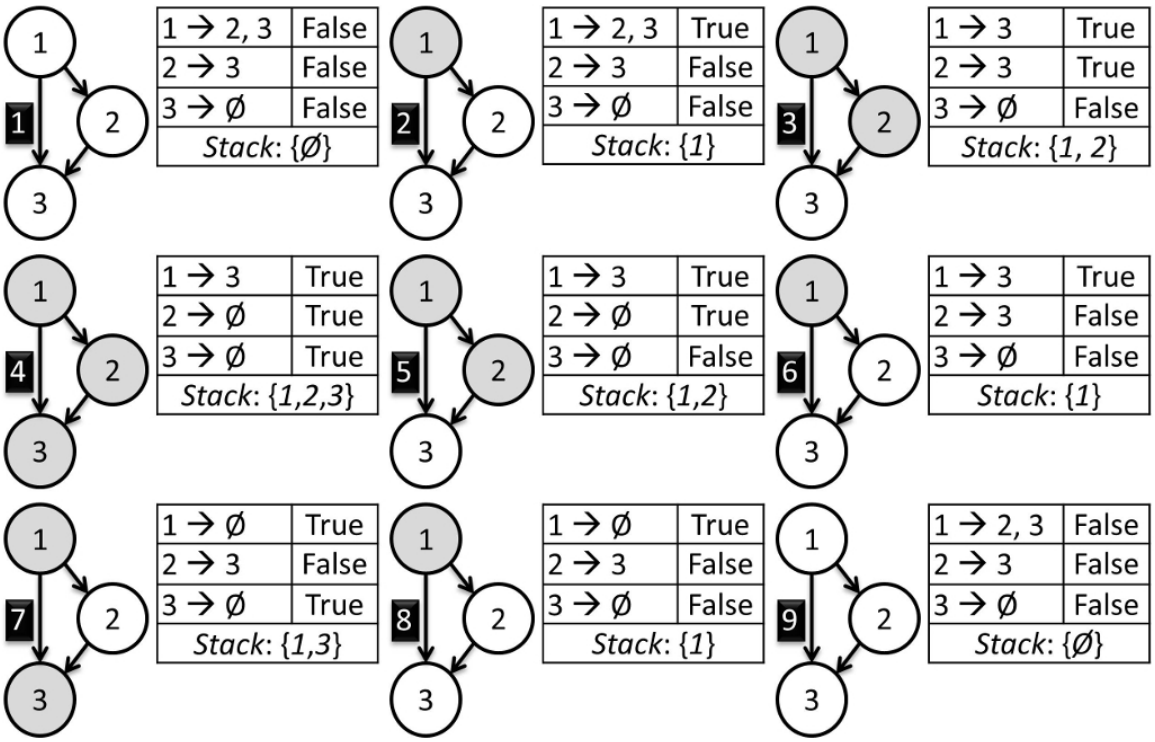


Figure 5.3: Illustration of Algorithm 5.1 using a feed-forward loop.

1's adjacency list. Set 3's visit status to true, and add 3 to the stack. Since 3 is a leaf node, output the stack contents, i.e., $1 \rightarrow 3$, as a root to leaf linear path. (8) 3 has no unvisited children. Backtrack to 1, pop 3 from the stack, and set 3's visit status to false. (9) 1 has no unvisited children. It is popped from the stack, has its adjacency list reconstructed, and has its visit status set to false. Since there are no other root nodes to visit, the algorithm terminates with the root to leaf linear paths $\{1 \rightarrow 2 \rightarrow 3, 1 \rightarrow 3\}$.

Before applying Algorithm 5.1, TEAK first converts a nonmetabolic KEGG pathway consisting of many different genes or gene products and their links into an unweighted, directed graph G where each node n in G represents either a single gene or a gene complex. Standard concepts are used to represent the *in degree*, denoted as k_{in}, as the number of links to a node and *out degree*, denoted as k_{out}, as the number

of links leaving a node. A node r is a root if and only if $r_{k_{in}} = 0$ and $r_{k_{out}} > 0$ whereas a node l is a leaf if and only if $l_{k_{in}} > 0$ and $l_{k_{out}} = 0$.

Root to leaf subpathways are important since they represent one of the possible routes taken from the beginning of a pathway to its end. Furthermore, in terms of signaling pathways, it is hypothesized that root to leaf subpathways effectively model signal transduction events. In one type of signal transduction paradigms, a growth factor binds to a cell membrane receptor that then propagates a signal via intracellular signaling pathways to reach the nucleus and cause a change in gene expression [Li, 2005]. Signal transduction pathways regulate cell proliferation, survival, motility, and differentiation [Li, 2005] and play vital roles in cancer [Bianco et al., 2006], mammalian associative conditioning [Selcher et al., 2002], and cellular response to stress [Shinozaki and Yamaguchi-Shinozaki, 1997]. For the KEGG signaling pathways, roots may be growth factors. For example, the epidermal growth factor (EGF) is a root of the MAPK Signaling Pathway.

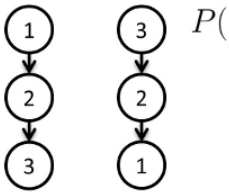
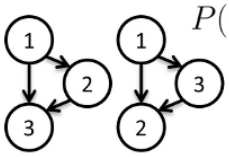
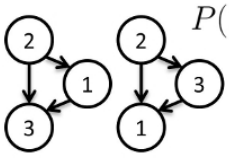
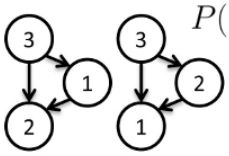
Since root to leaf linear subpathways may not effectively model the underlying biological condition under study for all of the KEGG pathways extracted, a different type of subpathway is needed. In this case, the Clique Percolation Method (CPM) [Palla et al., 2005, 2007] was tweaked. TEAK implements CPM with one notable change in which feed-forward loops, which are directed cliques of size three, are found instead of extracting all of the maximal cliques of a pathway. The focus on feed-forward loops to the detriment of larger clique sizes is justified since the feed-forward loop is one of the most common motifs in biological networks [Alon, 2007]. Nevertheless, Algorithm 3.6 shares many of the advantages found in the original CPM: 1) Genes may participate in multiple subpathways whereas most other approaches extract mutually exclusive subpathways. Biologically, the former approach may be more relevant as a gene may regulate multiple biological processes. 2) There exists

no gene or link whose removal would disjoin a subpathway, i.e., no single cut-node or cut-link exists in a subpathway.

Algorithm 5.2: Feed-Forward Subpathways

1: **Input**: An unweighted, directed graph G
2: **Output**: All feed-forward subpathways
3: Remove all self-loops from G
`/* CPM uses maximal cliques.` `*/`
4: **Find all of the feed-forward loops of G.**
5: Build the clique-clique overlap matrix M [Palla et al., 2005].
6: Set all off-diagonal entries of matrix M less than two to zero.
7: Return the connected components remaining in M as the feed-forward subpathways.

The algorithms for subpathway extraction and the Gaussian Bayesian networks used by TEAK are only applicable to directed networks, i.e., the KEGG nonmetabolic pathways. For the undirected networks extracted from the KEGG metabolic pathways, TEAK takes a slightly different approach. In order to extract directed linear subpathways from an undirected pathway, TEAK first extracts the longest shortest paths that are not contained within other shortest paths. For directionality TEAK then selects one of two directed linear paths that most closely resemble root to leaf linear subpathways. They may be obtained by fixing one terminal end of the shortest path as a root and the other terminal end of the shortest path as a leaf as illustrated in Figure 5.4. Since the pair of directed linear subpathways are I-equivalent, i.e., the pair of directed linear subpathways can be represented by the same set of conditional independence assertions [Koller and Friedman, 2009], either directed linear subpathway can serve as a "root to leaf" subpathway corresponding to the original shortest path. Then, in order to extract directed nonlinear subpathways from an undirected pathway, TEAK first extracts cliques of size 3. It then selects one of six I-equivalent feed-forward loops corresponding to the clique as illustrated in Figure 5.4.

$$
\begin{aligned}
P(1,2,3) &= P(1)P(2|1)P(3|2) \\
&= P(1)\frac{P(1,2)}{P(1)}\frac{P(2,3)}{P(2)} \\
&= \frac{P(1,2)P(2,3)}{P(2)}
\end{aligned}
\qquad
\begin{aligned}
P(1,2,3) &= P(3)P(2|3)P(1|2) \\
&= P(3)\frac{P(2,3)}{P(3)}\frac{P(1,2)}{P(2)} \\
&= \frac{P(1,2)P(2,3)}{P(2)}
\end{aligned}
$$

$$
\begin{aligned}
P(1,2,3) &= P(1)P(2|1)P(3|1,2) \\
&= P(1)\frac{P(1,2)}{P(1)}\frac{P(1,2,3)}{P(1,2)} \\
&= P(1,2,3)
\end{aligned}
\qquad
\begin{aligned}
P(1,2,3) &= P(1)P(3|1)P(2|1,3) \\
&= P(1)\frac{P(1,3)}{P(1)}\frac{P(1,2,3)}{P(1,3)} \\
&= P(1,2,3)
\end{aligned}
$$

$$
\begin{aligned}
P(1,2,3) &= P(2)P(1|2)P(3|1,2) \\
&= P(2)\frac{P(1,2)}{P(2)}\frac{P(1,2,3)}{P(1,2)} \\
&= P(1,2,3)
\end{aligned}
\qquad
\begin{aligned}
P(1,2,3) &= P(2)P(3|2)P(1|2,3) \\
&= P(2)\frac{P(2,3)}{P(2)}\frac{P(1,2,3)}{P(2,3)} \\
&= P(1,2,3)
\end{aligned}
$$

$$
\begin{aligned}
P(1,2,3) &= P(3)P(1|3)P(2|1,3) \\
&= P(3)\frac{P(1,3)}{P(3)}\frac{P(1,2,3)}{P(1,3)} \\
&= P(1,2,3)
\end{aligned}
\qquad
\begin{aligned}
P(1,2,3) &= P(3)P(2|3)P(1|2,3) \\
&= P(3)\frac{P(2,3)}{P(3)}\frac{P(1,2,3)}{P(2,3)} \\
&= P(1,2,3)
\end{aligned}
$$

Figure 5.4: **Top:** For the undirected linear graph $1 \leftrightarrow 2 \leftrightarrow 3$, there are two possible directed graphs starting at one of the two ends, namely $1 \rightarrow 2 \rightarrow 3$ and $3 \rightarrow 2 \rightarrow 1$. Both linear graphs are I-equivalent, and the result holds in general for linear graphs of arbitrary sizes. **Bottom:** The six possible feed-forward loops for a clique of size 3. All six feed-forward loops are I-equivalent. As such, it does not matter which orientation of the six directed feed-forward loop orientations are chosen for the undirected clique of size 3.

Overall, the TEAK framework may be extended beyond the approaches listed in Algorithms 5.1 and 5.2 for subpathway extraction. Other methods, such as the Girvan-Newman algorithm [Girvan and Newman, 2002] and Infomap [Rosvall and Bergstrom, 2008], may also be utilized with some modifications. Thus, TEAK is sufficiently flexible to address the needs of a variety of users.

5.3 Subpathway Ranking

To rank the linear and nonlinear subpathways, TEAK first uses the Bayes Net Toolbox (BNT) [Murphy, 2001] to fit a context specific Gaussian Bayesian network for each

subpathway. Briefly, a Gaussian Bayesian network is a Bayesian network in which all of its nodes are linear Gaussians. In other words, for a continuous node Y with continuous parents $X_1, ..., X_k$, the Conditional Probability Distribution of Y is

$$p(Y|x_1, ..., x_k) = N(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k; \sigma^2) \tag{5.1}$$

where $\beta_0, ..., \beta_k$ are the regression coefficients and σ^2 is the variance [Koller and Friedman, 2009]. It should be noted that Bayesian networks are only applicable to DAGs (Directed Acyclic Graphs). However, since all of the subpathways extracted by TEAK are DAGs, this limitation is not applicable. The choice of Bayesian networks is justified since they have already been used to discover networks from gene expression data [Friedman et al., 2000, 1999].

In general, using a Bayesian network consists of "search" and "score" parts, i.e., searching for a good candidate network and then scoring it. These parts may be implemented independently of one another, which is the case with TEAK. The candidate networks are in effect the linear and nonlinear subpathways. For scoring TEAK takes two approaches depending on whether context specific data or case-control data is used.

For context specific data, TEAK fits a single Gaussian Bayesian network for each subpathway and uses the Bayesian Information Criterion (BIC) [Schwarz, 1978] implemented in BNT for scoring each Gaussian Bayesian network. Briefly, BNT implements BIC as

$$Score_{BIC} = \log P(D|\hat{\theta}) - .5d \log N \tag{5.2}$$

where D corresponds to the gene expression data, $\hat{\theta}$ corresponds to the maximum likelihood estimate of the parameters used to represent the linear Gaussian node, d

is the number of parameters, and N is the number of samples in the gene expression data. Since BIC is decomposable, i.e., each node's score is calculated individually and then summed to return the final score, each subpathway is normalized by its number of nodes so that the scores are comparable.

Given that most researchers nowadays have case-control data, TEAK also supports case-control data, i.e., two data matrices, by fitting two Gaussian Bayesian networks, one for each data matrix. It then transforms each context specific Gaussian Bayesian network into its equivalent multivariate Gaussian form (please refer to the appendix of [Shachter and Kenley, 1989] for details and [Gmez et al., 2010; Gmez-Villegas et al., 2007] for examples). TEAK calculates the Kullback-Leibler (KL) divergence of the case multivariate Gaussian, q, from the control multivariate Gaussian, p, as

$$\begin{aligned} KL(q||p) &= \frac{1}{2}\log\frac{|\Sigma_p|}{|\Sigma_q|} + \frac{1}{2}Tr(\Sigma_p^{-1}\Sigma_q) \\ &+ \frac{1}{2}(\mu_q - \mu_p)^T\Sigma_p^{-1}(\mu_q - \mu_p) - k/2 \end{aligned} \tag{5.3}$$

where μ is the mean vector, Σ is the covariance matrix, $|\Sigma|$ is the determinant of the covariance matrix, Tr is the trace of a matrix, and k is the number of nodes (please refer to the appendix of [Roberts and Penny, 2002] for some additional details).

After scoring is completed, the ranked subpathways are displayed in TEAK's GUI. In this case, the top-ranked nonlinear subpathway, illustrated in Figure 5.1, was biologically validated in the results section. In particular, TEAK fetches the subpathway highlighted in yellow within its respective parent pathway to be displayed in a user's web browser using the KEGG API. In this case, the highlighted *Glycerophospholipid metabolism* subpathway is the top ranked nonlinear subpathway.

Furthermore, the Kumar lab has biologically validated that the *SLC1* gene (part of EC:2.3.1.51) is needed for yeast filamentous growth under nitrogen stress.

5.4 Conclusions and Results

TEAK (the BIC score function was used) was also compared against SPIA [Tarca et al., 2009] and SubpathwayMiner[Li et al., 2009] on the differentially expressed gene sets found by the Kumar Lab and presented in the main paper [Judeh et al., 2013a]. Tables 5.3 and 5.4 present TEAK's top 20 linear and nonlinear subpathway results, respectively. In the top ranked nonlinear Glycerophospholipid metabolism subpathway identified by TEAK alone, the *SLC1* gene was validated by the Kumar lab as necessary for filamentous growth under SLAD growth conditions.

TEAK, freely available at `http://code.google.com/p/teak` for Windows and MAC, is an innovative approach to detect activated subpathways. First, TEAK uses an in-house graph traversal algorithm to extract all root to leaf linear subpathways of a given pathway. Its major contributions include fully accounting for the topological information of subpathways and its ability to provide an interactive view of the data in the KEGG pathways. Furthermore, TEAK's GUI allows easy accessibility for a diverse set of users, and it implements an efficient algorithm to extract root-to-leaf linear subpathways where breadth-first and depth-first search algorithms may fail. Compared to previous approaches, TEAK also does not use differential gene expression analysis to determine modules of interest and is thus not sensitive to threshold values. Finally, by integrating the computational TEAK with experimental approaches, previously uncharacterized subpathways were discovered and experimentally validated involving the yeast stress response to nitrogen starvation.

KEGG ID	Name	SPIA	SubpathwayMiner
0970	Aminoacyl-tRNA biosynthesis	NA	No
0071	Fatty acid metabolism	NA	No
3013	RNA transport	Yes	NA
0290	Valine, leucine and isoleucine biosynthesis	NA	Yes
0480	Glutathione metabolism	NA	No
0480	Glutathione metabolism	NA	No
0480	Glutathione metabolism	NA	No
0290	Valine, leucine and isoleucine biosynthesis	NA	Yes
0600	Sphingolipid metabolism	NA	No
0240	Pyrimidine metabolism	NA	Yes
0480	Glutathione metabolism	NA	No
0450	Selenocompound metabolism	NA	No
0250	Alanine, aspartate and glutamate metabolism	NA	No
4111	Cell cycle - yeast	NA	No
0250	Alanine, aspartate and glutamate metabolism	NA	No
0564	Glycerophospholipid metabolism	NA	No
0513	Various types of N-glycan biosynthesis	NA	No
0230	Purine metabolism	NA	No
0620	Pyruvate metabolism	NA	No
0290	Valine, leucine and isoleucine biosynthesis	NA	Yes

Table 5.3: The top 20 linear subpathway results for TEAK. In the SPIA [Tarca et al., 2009] and SubpathwayMiner [Li et al., 2009] columns, "No" means that the pathway did not appear in a method's top 20 results, "Yes" means that the pathway did appear in a method's top 20 results, and "NA" for not applicable indicates that the method does not examine a given pathway. It needs to be noted that most of the *S. cerevisiae* KEGG pathways are metabolic, which SPIA does not support.

KEGG ID	Name	SPIA	SubpathwayMiner
00564	Glycerophospholipid metabolism	NA	No
00500	Starch and sucrose metabolism	NA	Yes
00480	Glutathione metabolism	NA	No
00900	Terpenoid backbone biosynthesis	NA	No
00270	Cysteine and methionine metabolism	NA	No
00230	Purine metabolism	NA	No
00270	Cysteine and methionine metabolism	NA	No
00330	Arginine and proline metabolism	NA	No
00270	Cysteine and methionine metabolism	NA	No
00600	Sphingolipid metabolism	NA	No
00900	Terpenoid backbone biosynthesis	NA	No
00250	Alanine, aspartate and glutamate metabolism	NA	No
00030	Pentose phosphate pathway	NA	No
00500	Starch and sucrose metabolism	NA	Yes
00230	Purine metabolism	NA	No
00230	Purine metabolism	NA	No
00030	Pentose phosphate pathway	NA	No
00600	Sphingolipid metabolism	NA	No
00051	Fructose and mannose metabolism	NA	No
04011	MAPK signaling pathway - yeast	Yes	NA

Table 5.4: The top 20 nonlinear subpathway results for TEAK. In the SPIA [Tarca et al., 2009] and SubpathwayMiner [Li et al., 2009] columns, "No" means that the pathway did not appear in a method's top 20 results, "Yes" means that the pathway did appear in a method's top 20 results, and "NA" for not applicable indicates that the method does not examine a given pathway. It needs to be noted that most of the *S. cerevisiae* KEGG pathways are metabolic, which SPIA does not support.

Chapter 6: Query Structure Enrichment Analysis (QSEA)[7]

As biological pathway databases continually increase in size and availability, efficient tools and techniques to query these databases are needed to mine useful biological information. A plethora of existing techniques already allow for exact or approximate query matching. Despite initial success, powerful techniques used for XML and RDF query matching have yet to be sufficiently exploited for use in query matching in the bioinformatics domain.

Many resources are now available that describe the pathways of different biological processes including KEGG [Kanehisa and Goto, 2000; Kanehisa et al., 2012], Biocarta (`http://www.biocarta.com`), and Reactome [Croft et al., 2011; Matthews et al., 2009]. These databases and others contain a wealth of biological information. Since the number of overall databases and the number of pathways within a database are continuously increasing, extracting meaningful biological insights may be too tedious to do manually. There is a need for various frameworks to at least partially automate the process, and they can be divided into three major categories [Sharan and Ideker, 2006].

First, network alignment is used to compare two or more networks of the same type from different species. Some of its major goals include identifying functional or conserved protein modules and network evolution analysis. Network alignment can also be used to predict novel interactions that may exist in one species but are absent in another.

Another category, network integration, focuses on combining different networks from the same species. These networks may be transcription regulatory networks, protein-protein interaction networks, signaling pathways, and metabolic net-

[7]The content in this chapter is largely derived from original author text and contributions found in [Judeh et al., 2012].

works. Some major goals include the identification of conserved modules across several networks, the relationships between different data types, and the prediction of interactions.

Finally, network querying is used to find a subnetwork module or query across a network or database of networks. Some of its major goals include knowledge transfer across species and the identification of conserved or repeated instances of the query across a network or database of networks. In particular, network querying holds great promise to extract useful biological insights from the pathway databases on a large scale and is the focus of this chapter.

Currently, there are a variety of different frameworks and tools that perform network querying varying from techniques that only produce exact matches to techniques that produce approximate matches. QPath [Shlomi et al., 2006], for example, takes as input a linear query and outputs a linear subpathway. It allows for results that do not contain all of the query proteins and also allows for the introduction of non-query proteins as well. QNet [Dost et al., 2008] later on extended QPath by supporting tree queries on subnetworks of bounded tree width. Both QPath and QNet rely on the color coding scheme introduced by Alon et al. [Alon et al., 1995] to identify subnetworks with a simple topology in an underlying network. Another framework, SAGA [Tian et al., 2007], also performs an approximate matching of the query network to the target network. It calculates a similarity distance between the two and takes into account the structural similarity, the number of vertex mismatches, and the number of gap vertices. For more details on current network querying techniques, one may refer to [Fionda and Palopoli, 2011].

While the current techniques have proven useful and beneficial, there exist other useful techniques in the XML/RDF querying domain that have yet to be fully exploited to query biological pathways. In the XML/RDF querying domain, a great

multitude of XML/RDF documents need to be queried both efficiently and accurately. Many techniques in this domain inherently focus on hierarchical matching from which the bioinformatics domain may greatly benefit from.

Starting with XML querying, historically the focus was on querying twigs, a tree-like "network" where edges are either a direct parent-child relationship or an ancestor-descendant relationship, i.e., reachability in the latter case. One popular algorithm, TwigStack [Bruno et al., 2002], has two major stages. First, it computes partial solutions for query root-to-leaf paths and compactly represents them using a chain of linked stacks. In the second stage, TwigStack merges and joins the partial results to compute twig query results. TwigStack, in conjunction with modified B-trees, can match query twig patterns in sub-linear time.

Vanilla XML documents, however, are trees and do not allow for a robust range of networks. Thus, for RDF documents, a more general and robust representation would be a DAG (Directed Acyclic Graph). For example, TwigStackD [Chen et al., 2005] is an extension of TwigStack that at its essence uses the transitive closure to process twig queries. Unlike other approaches, though, TwigStackD does not precompute the transitive closure or path indices for graphs. Instead, it represents a DAG using a combination of interval encodings on the aforesaid DAG's spanning tree. It also uses a customized predecessor index to determine the reachability of vertices based on the remaining edges not present in the spanning tree. Using this alternative representation for DAGs, the transitive closure is derived losslessly. TwigStackD can efficiently query twigs against DAGs with quadratic complexity in the average size of query variable bindings and a linear space cost for the data.

Given the powerful techniques available in the XML/RDF domain, it is worthwhile to explore their use to query biological pathways. QSEA specifically used the transitive closures of both query and queried graphs to focus more on the shared hier-

archies between queries and pathways. The use of transitive closures allow us to focus on DAGs and go beyond linear paths and trees as in QPath and QNet, respectively. QSEA also allows for approximate solutions by allowing any number of unmapped query vertices and absent ancestor-descendant edges from the initial query graph. Finally, a robust heuristic to solve the feedback arc set problem was developed.

6.1 Method Overview

Figure 6.1 presents the QSEA framework. QSEA may be divided into a preprocessing stage and a query processing stage. For the preprocessing stage, QSEA 1) uses the KEGG API to extract all pathways P for the selected organism. 2) For each pathway P_i, its edge and vertex betweenness are calculated. 3) For any pathway with cycles, Algorithm 6.1 is used to heuristically remove a feedback arc set. For these pathways, their edge and vertex betweenness are recalculated. 4) All shortest paths and the transitive closure for each pathway are then calculated. 5) For each pathway the shortest paths with largest mean edge and vertex betweenness are retained with precedence given to edge betweenness. These steps occur only once or when an organism needs to be updated.

For processing queries, QSEA does the following: 1) a user inputs a query graph stored in a space delimited Simple Interaction Format (SIF) file that can also be used by Cytoscape [Shannon et al., 2003]. 2) The user query graph is then mapped to each pathway P_i. Both the number of unmapped query vertices and the number of missing query hierarchical edges, i.e., ancestor-descendant relationships, are recorded for use in sorting later on. 3) A set of results is constructed by using combinations of all shortest paths between any two reachable query genes that are also reachable in the target pathway. 4) Results are sorted by first maximizing the number of query

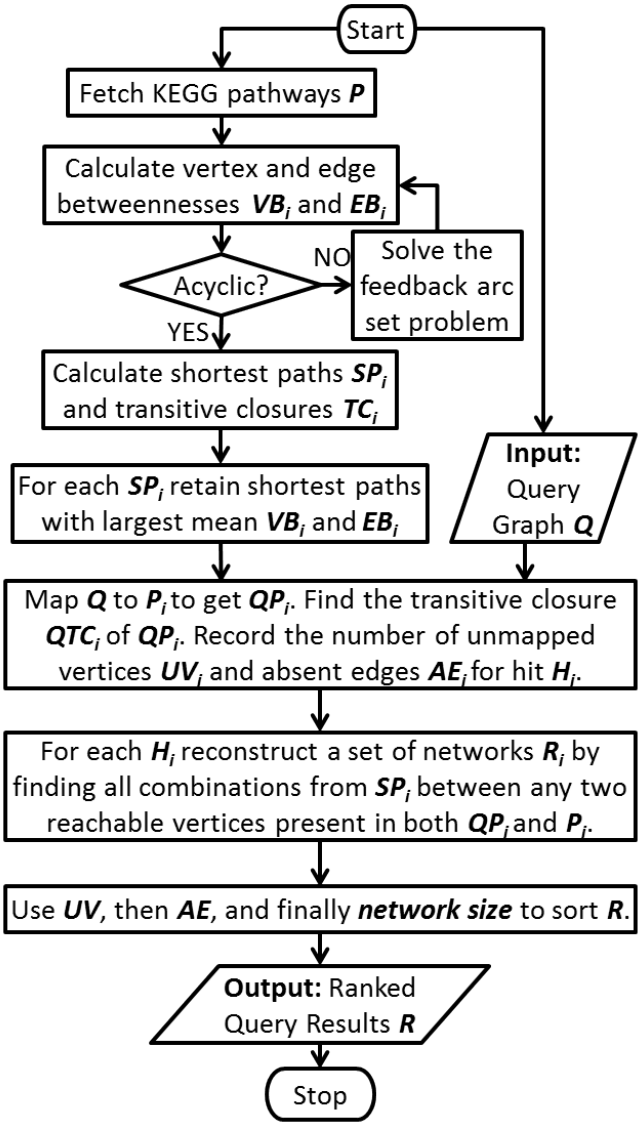


Figure 6.1: The QSEA framework

vertices found, then minimizing the number of missing hierarchical edges, and finally minimizing the size of the result, i.e., the number of "gap" vertices introduced. 5) The results are displayed in the QSEA GUI. Upon clicking a result, QSEA will use the default web browser to fetch the KEGG pathway and highlight the query result as illustrated in Figure 6.6.

6.2 Edge and Vertex Betweenness

QSEA treats each KEGG pathway as an unweighted, directed graph $G(V, E)$ with vertex set V and edge set E where the KEGG proteins are vertices and the relationships among the proteins form the edges. For each KEGG pathway, its edge and vertex betweenness are calculated. Originally introduced by Anthonisse [Anthonisse, 1971] and Freeman [Freeman, 1977], vertex betweenness is a measure of centrality that quantifies the number of shortest paths that pass through a given vertex. Furthermore, if there are n shortest paths between any two vertices, then a vertex that lies on one of their shortest path receives a contribution of $\frac{1}{n}$. In essence, vertices with high betweenness scores are quite important as a good number of a graph's shortest paths pass through them. Their loss may significantly impact if not make impossible the flow of information between various vertices.

Edge betweenness is an extension to vertex betweenness except that it applies to edges instead of vertices. Similar to vertices with a high vertex betweenness, an edge with a high edge betweenness has many shortest paths passing through it. In fact, Newman and Girvan [Newman and Girvan, 2004] used edge betweenness to divide a pathway into different communities or modules, and they provided an efficient $O(VE)$ algorithm as well. It should be noted that Brandes [Brandes, 2001] has also presented an $O(VE)$ algorithm for vertex betweenness that can be extended

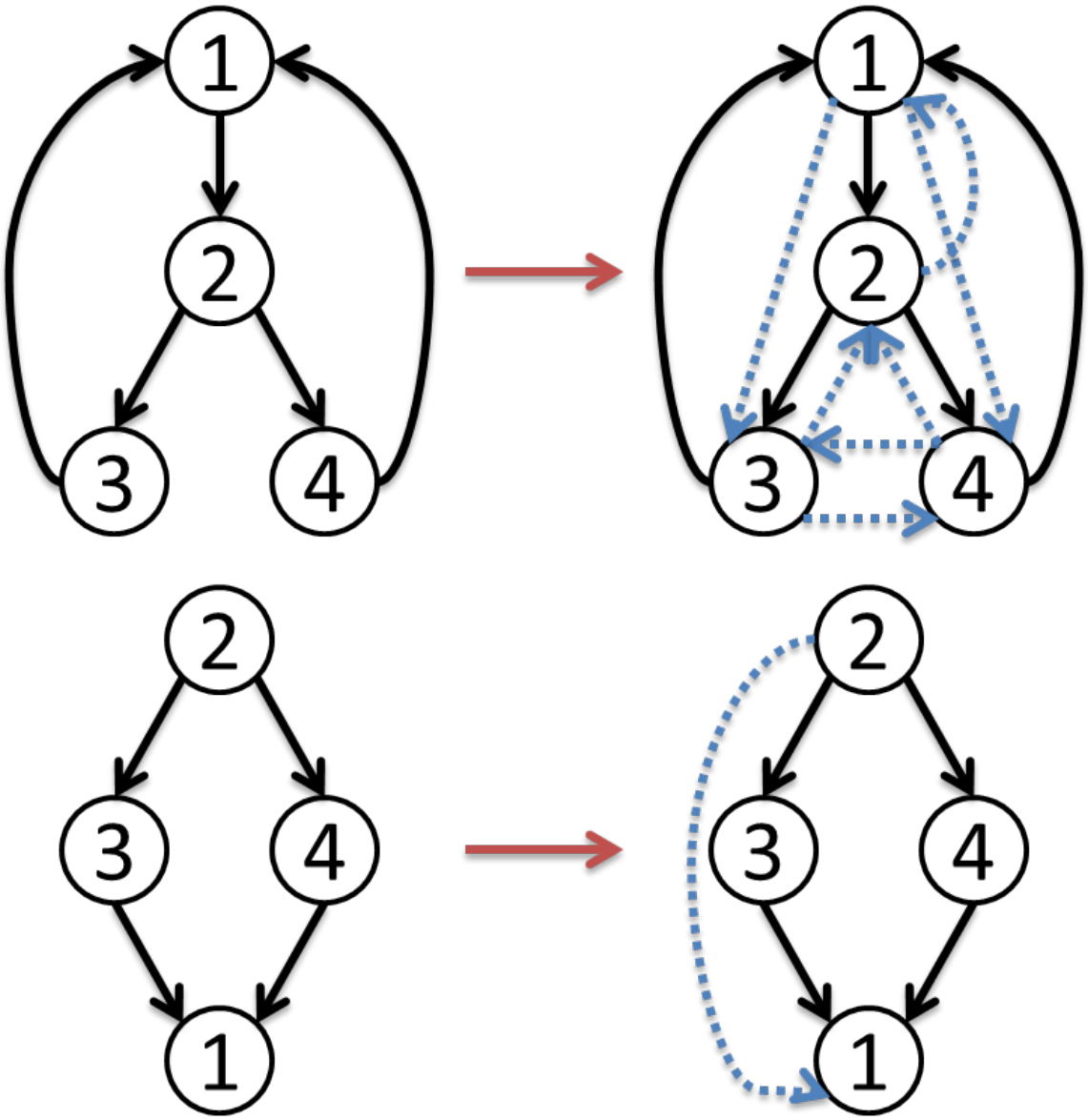


Figure 6.2: Top: An example of a strongly connected component. Its transitive closure on the right is a fully connected graph. In this case, all DAG queries would result in a query hit. Bottom: Removing the edge $1 \rightarrow 2$ produces a diamond subgraph, which happens to be a common biological network motif [Alon, 2007]. Its transitive closure on the right possess a hierarchy of vertices unlike the original subgraph.

Algorithm 6.1: A heuristic to remove feedback arc sets

1: **Input**: An unweighted, directed graph $G(V, E)$ with vertex set V and edge set E
2: **Output**: The acyclic graph $G(V, E - E_{FAS})$ where E_{FAS} is a feedback arc set
3: Remove all self-loops from G
4: Find all non-trivial strongly connected components S using Tarjan's algorithm [Tarjan, 1972]
5: Find edge and vertex betweenness EB and VB
6: Extract from G the adjacency lists A
`/* Prioritizes exploring edges with high betweenness scores */`
7: Sort A first according to EB and then VB in descending order
8: **for** *i = 1,...,|S|* **do**
9: Run **SCC-DFS(**S_i**)**
10: **end**
11: Return G which is now acyclic. The set of edge sets $E_{S_i,j,FAS}$ removed from the strongly connected components form E_{FAS}
`/* Strongly Connected Component Depth-First Search */`
12: **SCC-DFS**
13: Sort the vertices V_{S_i} of S_i in descending order of vertex betweenness
14: Extract from A the set of sorted adjacency lists A_i that represents S_i
15: **for** *j = 1,...,|S_i|* **do**
16: Begin a depth-first search at vertex $V_{S_i,j}$ using A_i to prioritize the order of vertices to visit. Remove all of the back edges found in the depth-first search to obtain $E_{S_i,j,FAS}$.
17: **end**
18: Remove from E the set of edges $E_{S_i,j,FAS}$ that minimizes first the number of edges removed and then the edge betweenness of the edges removed

to edge betweenness, and it is this implementation that was used. In QSEA's case, edge and vertex betweenness are used to both guide a depth-first search (DFS) and select shortest paths that are most significant.

6.3 Feedback Arc Set

For its hierarchical querying, QSEA relies on the transitive closure. Briefly, for any directed graph $G(V, E)$ with vertex set V and edge set E, its transitive closure TC

Algorithm 6.2: Minimum feedback arc set removal

1: **Input**: An unweighted, directed graph $G(V, E)$ with vertex set V and edge set E
2: **Output**: The acyclic graph $G(V, E - E_{MFAS})$ where E_{MFAS} is the minimum feedback arc set
3: Remove all self-loops from G
4: Find all non-trivial strongly connected components S using Tarjan's algorithm [Tarjan, 1972]
5: **for** *i = 1,...,|S|* **do**
6: Extract the subgraph G_i that represents S_i
7: Set the number of edges to remove R to 0
8: **while** G_i *is cyclic* **do**
9: Increment R by 1
10: Find all combinations C of the edges E_{S_i} choosing R edges at a time
11: **for** *j = 1,...,|C|* **do**
12: Reverse the set of edges C_j in G_i
13: **if** G_i *is acyclic* **then**
14: Break from the inner for loop
15: **end**
16: Undo the edge set reversal
17: **end**
18: **end**
19: Remove the last edge set C_j from E where C_j is a minimum feedback arc set for G_i
20: **end**
21: Return G which is now acyclic. Collectively, all Cs previously removed form E_{MFAS}

is a concise representation of the reachability of the vertices in V. More specifically, TC's vertex set V_{TC} equals V while its edge set E_{TC} is a superset of E. Furthermore, an edge (i, j) in E_{TC} either denotes the presence of an edge (i, j) in E or the presence of a series of edges in E that can be traversed to reach vertex j starting from vertex i . The transitive closure may be calculated using the Floyd-Warshall algorithm in $O(|V|^3)$ since the existence of a shortest path from vertex i to vertex j means that j is reachable from i. The transitive closure may also be computed in $O(|V|^{2.376})$.

For more details on efficiently calculating the transitive closure, one may refer to [Bang-Jensen and Gutin, 2009].

Furthermore, it should be noted that for DAGs, the transitive closure can represent a hierarchy of vertices. It is this property that QSEA exploits to process a query graph Q against a queried graph P since an exact query hit occurs if and only if E_{TC_Q} is a subset of E_{TC_P}. In other words, one can also find a topological sort ordering that is common to the transitive closures of the query graph Q and queried graph P. Briefly, a topological sort ordering is a non-unique linear ordering of a DAG's vertices such that for any edge (i, j), i will always appear before j in the linear ordering. As a result, since QSEA relies on the transitive closure for its hierarchical querying, it is of particular importance that both Q and P are DAGs. Otherwise, it is not possible to exploit a hierarchy of vertices in the presence of cycles as illustrated in Figure 6.2. In Figure 6.2, the top subgraph is a strongly connected component from the KEGG *H. sapiens MAPK Signaling Pathway*. Its transitive closure is a fully connected graph. As a result, any query graph without self-loops will satisfy the subset condition and will result in a query hit, which does not allow for any discriminative biological insights. The diamond subgraph on the bottom of Figure 6.2, on the other hand, is a DAG and possesses an meaningful hierarchy for querying. In this case, only a subset of queries will result in a query hit.

To ensure that both the query graphs and queried graphs are DAGs, QSEA uses two different approaches. For query graphs, the solution is to simply restrict all queries to DAGs. For the queried KEGG pathways, however, such an option may remove meaningful pathways. As such, QSEA uses a robust heuristic to remove cycles from the pathways while preserving the overall directional flow of a pathway.

First, it is prudent to present some background on the underlying problem. For a directed graph $G(V, E)$, $E_{FAS} \subset E$ is a feedback arc set if the removal of E_{FAS}

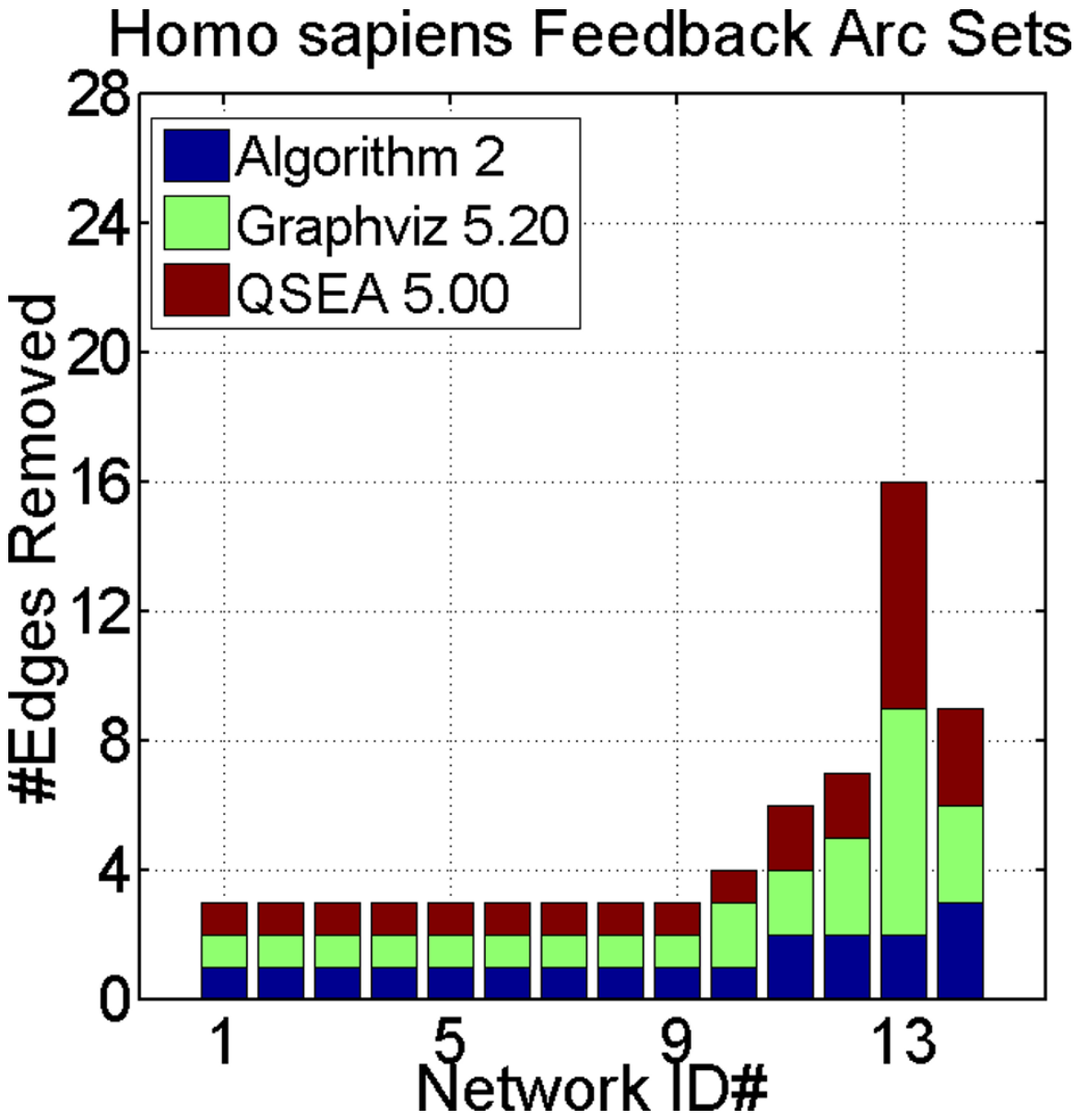


Figure 6.3: The number of edges removed from 14 cyclic *H. sapiens* KEGG pathways. QSEA equals or outperforms Graphviz 100% of the time.

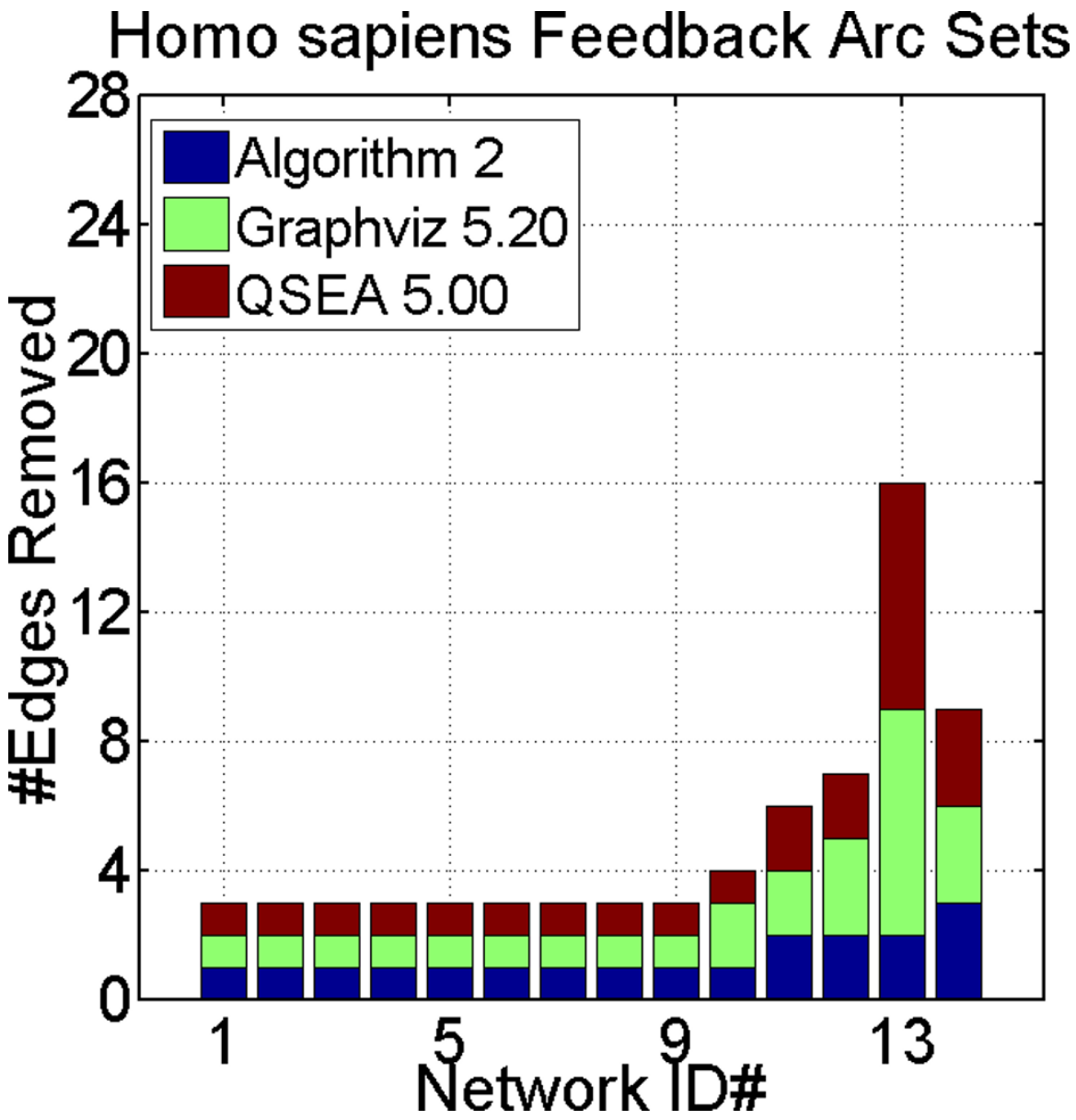


Figure 6.4: The number of edges removed from 121 acyclic *H. sapiens* KEGG pathways made cyclic by adding up to four edges. QSEA equals or outperforms Graphviz 98% of the time.

makes G acyclic [Bang-Jensen and Gutin, 2009]. In particular, a minimum feedback arc set $E_{MFAS} \subset E$ is the minimum number of edges whose reversal makes G acyclic [Bang-Jensen and Gutin, 2009]. For arbitrary graphs, the problem is known to be NP-hard, and the best known approximation has ratio $O(\log |V| \log \log |V|)$ [Even et al., 1998].

Heuristically solving the feedback arc set is also a well known subproblem in drawing directed graphs with minimal edge crossings. For example, Graphviz [Ellson et al., 2002] uses a depth-first search (DFS) to heuristically eliminate some edges to break cycles in order to rank the vertices. Briefly, DFS divides edges into tree edges and non-tree edges consisting of forward edges, back edges, and cross edges [Cormen et al., 2009]. Edges whose vertices are visited for the first time form the tree edges of a depth-first forest. Forward edges are non-tree edges that directly connect a vertex i with a descendant vertex j in a depth-first tree whereas back edges are the opposite, i.e., they directly connect a descendant j with an ancestor i. Finally, any other non-tree edge is classified as a cross edge.

It should be straightforward to observe that all back edges found by DFS form a feedback arc set. As such, Graphviz takes one non-trivial strongly component, i.e., a subgraph in which any two vertices are connected either directly or indirectly through a number of intermediate vertices, and counts the number of times an edge in the strongly connected component forms a back edge via a depth-first search. The edge with maximal count is removed, and the process is repeated until no strongly connected components remain [Gansner et al., 1993].

QSEA takes a similar approach to Graphviz with some major differences as described in Algorithm 6.1. The most major difference is the use of vertex and edge betweenness as presented in Section 6.2 to guide DFS when it chooses vertices to visit and edges to explore. By using betweenness to guide DFS, QSEA greedily focuses on

edges and vertices with high betweenness. This allows the forest of trees generated by DFS to be extracted deterministically. More importantly, though, it is hoped that the edges in cycles that become back edges have a low betweenness score, which in turn may be of less importance for the overall pathway.

To compare the performance of QSEA and Graphviz, Algorithm 6.2, outlined in [Ispolatov and Maslov, 2008], is used as a reference point as it is able to determine a minimum feedback arc set for each strongly connected component. Algorithm 6.2 is able to find E_{MFAS} via a naïve approach that checks all possible combinations of edges for a given number of edges R. Given a non-trivial strongly component S with edge set E_S and $R = |E_{S,MFAS}|$, Algorithm 6.2 has to check $\sum_{e=1}^{R} \binom{|E_S|}{e}$ edge combinations in G in order to find the minimum feedback arc set for S. This process may be unfeasible for relatively small values of E_S.

In Figures 6.3 and 6.4, the performances of QSEA (Algorithm 6.1), a naïve approach (Algorithm 6.2), and Graphviz were compared. It should be noted that the focus was solely on the size of the feedback arc sets removed and does not take into account the actual edges removed by either algorithm. As illustrated in Figures 6.3 and 6.4, QSEA performs no worse than Graphviz and outperforms it in some cases.

One may also observe from Figures 6.3 and 6.4 that both heuristics perform quite well in general and are able to remove a minimum feedback arc set for some of the pathways selected. When they differ, though, QSEA removes less edges overall compared to Graphviz as indicated by their respective Euclidean distances from Algorithm 6.2. Briefly, the Euclidean distance was calculated as

$$\sqrt{\sum_{i=1}^{n}(E_{MFAS_i} - E_{FAS_i})^2}$$

where E_{MFAS} corresponds to the vector of minimum feedback arc set sizes removed by Algorithm 6.2 and E_{FAS} corresponds to the vector of feedback arc set sizes removed by either QSEA or Graphviz.

6.4 Shortest Paths and Transitive Closure

After the feedback arc sets are removed and vertex and edge betweenness are recalculated, all shortest paths between any two vertices i and j are calculated. Since there may be multiple shortest paths between any two vertices, only the shortest paths with largest mean edge and vertex betweenness are kept while prioritizing edge betweenness. One may hypothesize that these shortest paths play a more significant role compared to other shortest paths as they capture a larger snapshot of the pathway or graph at a global level. Regardless, there may still be multiple shortest paths between any two vertices i and j. As such, QSEA will display all combinations of shortest paths in the results when needed. The transitive closures for each pathway are also calculated and stored.

6.5 Query Matching and Output

Once preprocessing is complete, QSEA is ready to take as input user query graphs that follow a space-delimited Simple Interaction Format (SIF). Using the KEGG API, QSEA is able to support a variety of labeling schemes including KEGG, NCBI GeneID (Entrez), NCBI GI, and UniProt. In the SIF file, users can construct two types of edges similar to TWIGs presented in the introduction. The first edge is a direct parent-child edge in which vertex i connects directly to vertex j. The second edge is an ancestor-descendant edge in which there may be from 0 to any number of arbitrary vertices between i and j. Finally, it should be noted that a gene may

YAL041W - YNL271C YCL027W
YLR332W - YNL271C
YNL053W - YCL027W
YER118C - YLR362W

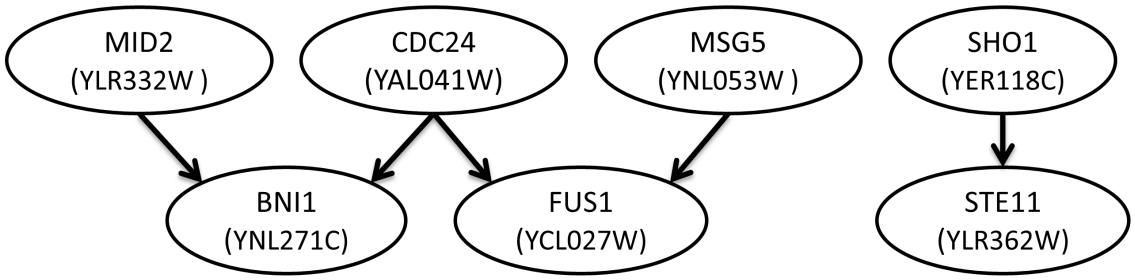


Figure 6.5: Top: An example query file. A minus sign represents an ancestor-descendant edge whereas an equal sign represents a direct parent-child edge. In this case, there are only ancestor-descendant edges in the query. Middle: QSEA transforms the query into its pathway specific form on the right for the KEGG *S. cerevisiae MAPK signaling pathway*, which has two instances of SHO1 and three instances of STE11. For ease of cross-referencing with the KEGG results in Figure 6.6, gene names are used.

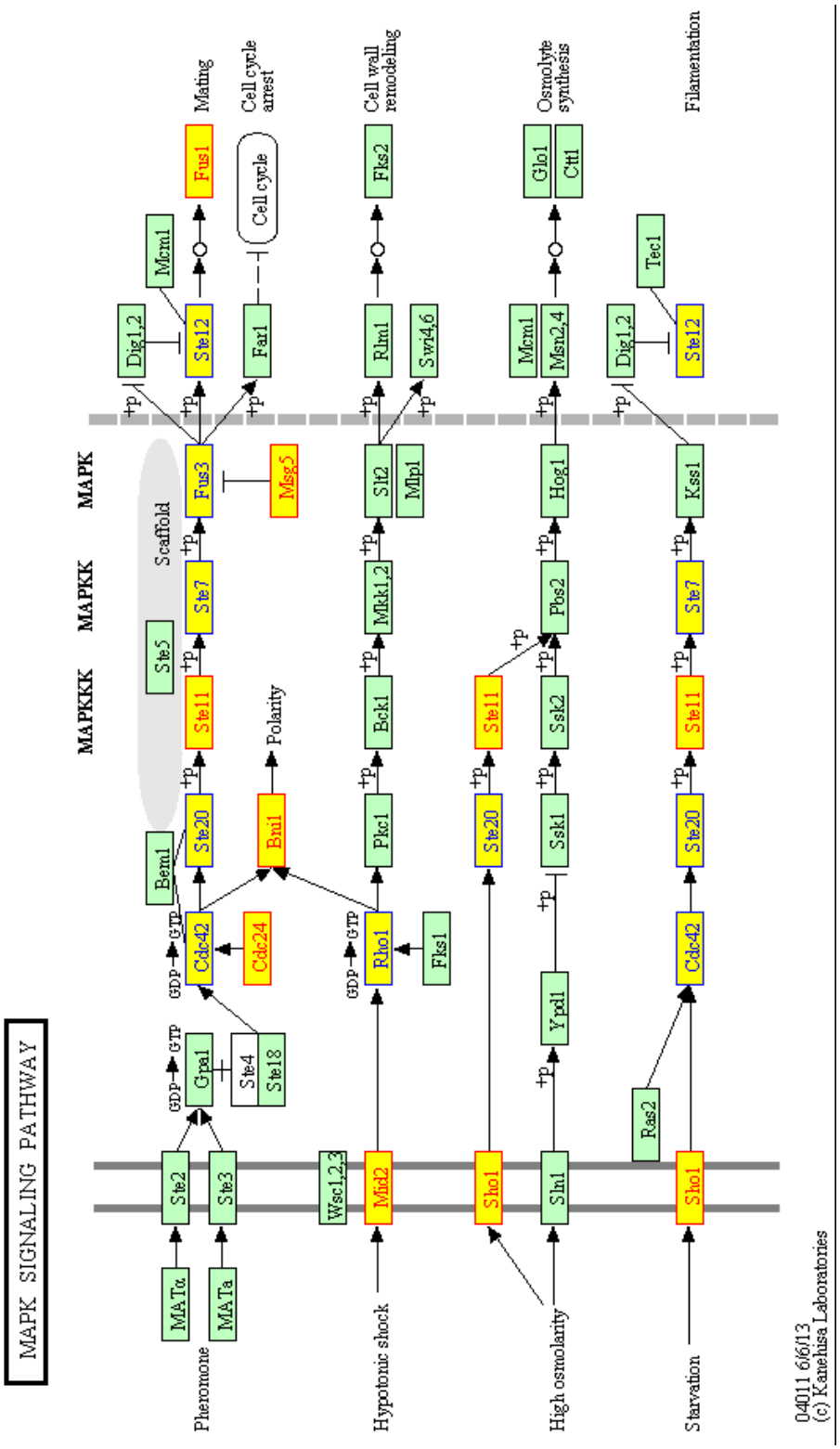


Figure 6.6: The result of the query in Figure 6.5.

map to multiple locations in a KEGG pathway, which QSEA is able to handle as illustrated in Figures 6.5 and 6.6.

For query matching QSEA is able to support both exact and approximate query graph solutions. For this purpose, after QSEA maps a query Q onto a pathway P to get QP_i, QSEA makes note of the number of unmapped query vertices UV_i for each pathway P_i. QSEA then also calculates the transitive closure for QP_i to get QTC_i. First, for direct edges mentioned previously, QSEA requires an exact match. Otherwise, for ancestor-descendant edges, QSEA allows for any number of mismatches. Thus, QSEA also takes into account the number of absent ancestor-descendant edges AE. Once this process is complete, QSEA constructs a set of graphs as results from all possible combinations of shortest paths between two vertices i and j that are reachable both in P_i and QP_i.

After processing all of the pathways and extracting their resulting graphs, QSEA ranks them according to three criteria. First, it maximizes the presence of query vertices. Then, it minimizes the number of absent ancestor-descendant edges. Finally, it minimizes the size of the resulting graphs. Using these criteria, QSEA prioritizes query results that possess all of the query vertices and edges with minimal size.

An example of the query matching and output is presented in Figures 6.5 and 6.6. First, a SIF file is constructed consisting of two direct parent-child edges and three ancestor-descendant edges. The SIF file is then processed into a graph where each vertex uniquely represents a gene. After mapping the query graph to its MAPK signaling pathway version, a query hit is found. The highlighted query result is fetched from KEGG and displayed in a user's default web browser. QSEA is able to display simultaneously multiple instances of a gene all at once. All vertices highlighted in yellow are part of the query result. Depending on the foreground color,

red denotes a query vertex whereas blue denotes a "gap" vertex introduced to connect the query vertices. It should be noted that due to the transition of the KEGG API from a SOAP version to a REST version, individual nodes are no longer addressable. As a result, STE7 and STE12 are visible as genes are now addressed as opposed to individual nodes as in the SOAP version of QSEA.

6.6 Conclusions

In this chapter, the Query Structure Enrichment Analysis (QSEA) was presented. QSEA's contributions are two-fold. First, QSEA made use of XML/ RDF techniques into biological pathway querying. Specifically, the use of the transitive closure allows for focus on matching hierarchies between queries and their target pathways. Second, QSEA introduced a robust heuristic to solve the feedback arc set problem which has a promising performance. Our contributions are implemented in an easy-to-use GUI software. Binaries for QSEA are freely available at `http://code.google.com/p/teak/` for Windows and MAC.

Chapter 7: Conclusions and Future Work

Given the abundance of molecular profiling data sets and pathway resources, computational approaches that utilize and exploit both resources are of interest. From one angle, network reconstruction from discretized molecular profiling data sets into gene sets allows the exploitation of molecular profiling data sets from different studies and different platforms. Community detection algorithms can be employed to partition pathways into subpathways for a finer grain of analysis. Finally, hypothesis can be queried against a database of pathways to discover shared modules across different pathways. To this end, three computational approaches were presented to exploit the relationships among these various types of data as presented in Figure 1.2.

First, the Gene Set Cultural Algorithm (GSCA) was presented that reconstructed networks from discretized gene expression data sets. By using the KEGG pathways as input, GSCA exploits prior knowledge to reduce the search space for the reconstructed network. Furthermore, unlike previous approaches, GSCA explores the search space of candidate networks in parallel allowing for a more robust capability to escape local extrema or false peaks.

Second, the Topology Enrichment Analysis frameworK (TEAK) uses both linearn and nonlinear algorithms to extract all root to leaf linear subpathways or simple paths of a network. By incorporating gene expression data sets, these subpathways can then be transformed into Bayesian networks and ranked accordingly. The results were then returned to the user to be view in the user's default web browser.

Finally, the Query Structure Enrichment Analysis (QSEA) approach allows for the "fuzzy" or hierarchical querying of a query against the KEGG networks. By employing a novel heuristic to remove feedback arc sets, the transitive closure was employed for a robust matching. Furthermore, edge and vertex betweenness were

utilized to highlight the paths within the network that may be the most important due to their position in the network.

The work presented allows for many possible future avenues of research:

- **Parallelization of Network Reconstruction Algorithms** The current iteration of GSCA easily lends itself to parallelization. Each search agent independently explores the search space and reconvenes together in the belief space to determine the most fit beliefs to retain. Since the bottleneck for the parallelization process involves the belief space, the belief space can be extended into a distributed algorithm where nodes communicate with one another the fitness of their beliefs versus the fitness of the beliefs of their neighbors. After exchanging of belief fitness values is complete, a node can then unilaterally determine whether or not to continue exploring its current solution or to jump to a random point in the search space.

- **Dynamic Generation of Subpathways** Currently, TEAK computes *a priori* the subpathways to be examined. Since the subpathways examined by TEAK are a subset of all possible simple paths that can be generated, the number of subpathways for more complex networks may be computationally infeasible. As such, novel algorithms that focus on calculating the scores of the subpathways dynamically are needed to reduce the potential computational overhead.

- **Incorporating Molecular Profiling Data into Queries** QSEA currently focuses only on the structure of the query versus the structures of the targets. The method may become more robust and useful if gene expression data sets were incorporated as an additional piece of information via the use of gene expression data sets to return query hits most relevant to the gene expression data set under examination.

The culmination of this work is a software suite called TEAK, which is freely available at `htpp://code.google.com/p/teak`.

Appendix A: LIST OF PUBLICATIONS

Journal Publications

1. T. Judeh, C. Johnson, A. Kumar, and D. Zhu, "Teak: topology enrichment analysis framework for detecting activated biological subpathways," *Nucleic Acids Res*, vol. 41, no. 3, pp. 1425–37, 2013.

2. L. Acharya, T. Judeh, Z. Duan, M. Rabbat, and D. Zhu, "Gsgs: a computational approach to reconstruct signaling pathway structures from gene sets," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 9, no. 2, pp. 438–50, 2012.

3. L. R. Acharya, T. Judeh, G. Wang, and D. Zhu, "Optimal structural inference of signaling pathways from unordered and overlapping gene sets," *Bioinformatics*, vol. 28, no. 4, pp. 546–56, 2012.

Conference Publications

1. T. Judeh, T. Jayyousi, L. Acharya, R. G. Reynolds, and D. Zhu, "Gsca: Reconstructing biological pathway topologies using a cultural algorithms approach," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, Conference Proceedings.

2. T. Judeh, T. Jayyousi, L. Acharya, R. G. Reynolds, and D. Zhu, "Gene set cultural algorithm: A cultural algorithm approach to reconstruct networks from gene sets." ACM, 2013, Conference Paper, pp. 641–648.

3. T. Judeh, T. Nguyen, and D. Zhu, "Qsea for fuzzy subgraph querying of kegg pathways." ACM, 2012, Conference Paper, pp. 474–481.

Book Chapters

1. L. Acharya, T. Judeh, and D. Zhu, *A Survey of Computational Approaches to Reconstruct and Partition Biological Networks*. John Wiley & Sons, Inc., 2012, pp. 1–43.

Master's Thesis

1. T. Judeh, "Sea: a novel computational and gui software pipeline for detecting activated biological sub-pathways," Thesis, The University of New Orleans, 2011.

Appendix B: Copyrights

Various copyright/licensing agreements allowing the use of previously published material is presented in this appendix.

JOHN WILEY AND SONS LICENSE TERMS AND CONDITIONS

Oct 18, 2013

This is a License Agreement between Thair Judeh ("You") and John Wiley and Sons ("John Wiley and Sons") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by John Wiley and Sons, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	3252190108782
License date	Oct 18, 2013
Licensed content publisher	John Wiley and Sons
Licensed content publication	Wiley oBooks
Licensed content title	A Survey of Computational Approaches to Reconstruct and Partition Biological Networks
Book title	Statistical and Machine Learning Approaches for Network Analysis
Licensed copyright line	Copyright © 2012 John Wiley %26 Sons, Inc.
Licensed content author	Lipi Acharya,Thair Judeh,Dongxiao Zhu
Licensed content date	Jun 21, 2012
Start page	1
End page	43
Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley chapter
Format	Print and electronic
Portion	Full chapter
Will you be translating?	No
Total	**0.00 USD**
Terms and Conditions	

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a "Wiley Company") or a society for whom a Wiley Company has exclusive publishing rights in relation to a particular journal (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

1. The materials you have requested permission to reproduce (the "Materials") are protected by copyright.

2.You are hereby granted a personal, non-exclusive, non-sublicensable, non-transferable, worldwide, limited license to reproduce the Materials for the purpose specified in the licensing process. This license is for a one-time use only with a maximum distribution equal to the number that you identified in the licensing process. Any form of republication granted by this license must be completed within two years of the date of the grant of this license (although copies prepared before may be distributed thereafter). The Materials shall not be used in any other manner or for any other purpose. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Material. Any third party material is expressly excluded from this permission.

Obtained from http://scholarworks.uno.edu/faq.html

General FAQ

What is ScholarWorks@UNO?
ScholarWorks@UNO is a digital collection of the research, scholarship, and creative work of the University of New Orleans. Using a seamless opennaccess platform, the university's unique academic and artistic achievements are collected, preserved, and disseminated to a global audience of students and scholars.

Who can contribute content?
Any college, department, center, or other unit of the University can participate as a community in ScholarWorks. Each community assigns a coordinator who can work with the Library to develop the unit's content in the repository.

Faculty members also have the option to maintain their own SelectedWorks pages, which can provide complete lists of publications along with other information about teaching and research interests, and link into repository content. We can set up your SelectedWorks site for you, or you can set it up yourself. Please contact the librarian for your department or email the ScholarWorks@UNO administrator at scholarworks@uno.edu.

What kind of content can be included?
ScholarWorks will support a variety of types of content, in a range of formats, including articles (previously published or pre/post prints), conference papers or presentations, technical papers or reports, images, software, data sets, and more.

Who has access to the content in ScholarWorks@UNO?
ScholarWorks@UNO is intended to be an *open access* repository. By *open access*, we mean that anyone with an internet connection will be able to access and download the files and documents it houses.

Why should I contribute research/scholarship/creative work to ScholarWorks@UNO?
ScholarWorks@UNO provides increased visibility and impact for your work, allowing barrier-free discovery by tools such as Google Scholar. The repository also provides long-term, stable preservation of your content, and gives you more control over access to your own work. By depositing your work in ScholarWorks@UNO you help to demonstrate the scientific, social, and economic value of UNO to the local community and to scholars and researchers worldwide.

How do I get my articles or other publications into ScholarWorks@UNO?
To submit your work, contact the librarian for your department or email the library's ScholarWorks administrator at scholarworks@uno.edu. We will work with you to check copyright restrictions, assign metadata to your work to enhance discoverability, and make it available. By depositing your work in your own SelectedWorks page, it can also be made available to be displayed in ScholarWorks.

What about copyright issues?
Authors retain the copyright for all content posted in the repository. The author agreement specifies a nonexclusive right to use, meaning the author is free to reuse the content elsewhere. If you have signed copyright over to a journal or other publisher, other steps may have to be taken to obtain permission to deposit a copy in the repository. See the SHERPA RoMEO website for a list of publisher copyright and self-archiving policies. The Library can work with you to verify copyright status and seek permission for previously published work.

Further questions? Please contact the Scholarworks administrator at scholarworks@uno.edu or 280-6547.

Author FAQ

- I don't have electronic versions of old working papers that I'd like to include in the repository. Is it okay to scan the printed page to a PDF file?
- When I copy and paste abstracts into the Submit form, some text is missing, quotes look odd, or strange characters appear in the abstract. What's going on?

- How do I include accents and special characters in the abstracts and titles?
- How do I revise a submission?
- How can I submit a multi-part file, such as multiple chapters for a book?
- Can I post related files (sound clips, data sets, etc.) alongside the published article?
- Can I post a reprint from a journal?
- A working paper in our repository site has been published in a slightly revised form in a journal. What should I do?

I don't have electronic versions of old working papers that I'd like to include in the repository. Is it okay to scan the printed page to a PDF file?

Yes--scanning printed pages is a great way to create PDF files for inclusion in the repository. There are two ways to scan a page: using OCR (Optical Character Recognition) or scanning the page as an image. Making OCR scans requires careful proofreading and loses the original formatting of the documents. Image scans cannot be searched. The best solution takes advantage of both of these methods. Many software applications allow for the OCR capture of image scans. When documents are scanned this way, users see the image scan but search the full-text of the document. This is the preferred method for scanning documents for the repository.

{ top }

When I copy and paste abstracts into the Submit form, some text is missing, quotes look odd, or strange characters appear in the abstract. What's going on?

When copying abstracts from a word processing file or a PDF file, and pasting the text into the submission form, you are taking text from an environment that may support fonts and special characters (like symbols or "smart quotes"). Because the abstract is intended to be presented on the web, the format of the abstract needs to be reduced to plain text with no fonts or special characters. We recommend the following changes to keep your titles and abstracts legible on the web:

- Change "smart" single and double quotes to straight quotes.
- Change an ellipsis to three periods (...)
- Change em- and en-dashes to hypens.

If you would like to use **bold** and *italic* in your abstracts, you may do so using the corresponding HTML codes. If submitting an abstract in HTML format, please be sure to select the corresponding option on the submission form.

The following HTML tags are recognized by the system and may be used to format an abstract (use lowercase tags):

<p> - paragraph	
<**p**>This is the first paragraph.</**p**> <**p**>This is the second paragraph.</**p**>	This is the first paragraph. This is the second paragraph.
** - line break**	
<**p**>This is a line of text with a linebreak here. <**br** /> This is text after</**p**>	This is a line of text with a linebreak here. This is text after
** - strong/bold**	
<**strong**>bold text</**strong**>	**bold text**
** - italics/emphasis**	
<**em**>italicized text</**em**>	*italicized text*
<sub> - subscript	
Text with <**sub**>subscript</**sub**>	Text with $_{\text{subscript}}$
<sup> - superscript	
Text with <**sup**>superscript</**sup**>	Text with $^{\text{superscript}}$

{ top }

How do I include accents and special characters in the abstracts and titles?

The repository software supports the ISO 8859-1 character set (this includes the numbers 0-9, upper- and lower-case letters A-Z, and standard English punctuation). Although you may take advantage of the complete character set, we recommend you consider not using special characters as these may inhibit user searches, both on the web and on the site.

{ top }

How do I revise a submission?

To revise a submission:

1. From the **My Account** page click **Submission Management**.
2. In the list of pending submissions, click the title of the article you want to change. (If you are revising a published submission, click on the **Published Submissions** link in the top left and select the title of the article from the resulting list.)
3. Click **Revise Submission** from the list in the top left.
4. Enter your changes in the Revise Submission form, and click **Submit** at the bottom of the page to submit your changes. (You only need to modify the portion of the form that corresponds to the changes you wish to make.)
5. If you are revising a pending submission, you may continue with the publication steps if appropriate. If you are revising a publised submission, be sure to click the option to **Update** the site to incorporate your changes to the web pages.

{ top }

How can I submit a multi-part file, such as multiple chapters for a book?

Combine all the sections together as one Microsoft Word file or PDF file and submit that.

To make one PDF file from multiple files, open the first PDF file, then choose **Document>Insert Pages** from Acrobat's menus to insert the second file (indicate it should go after the last page of the first file), and repeat for all documents. The result will be one compound PDF file which may then be submitted.

If you feel that the one large PDF file might be too large for some people to download, we suggest that you submit the consolidated file as the full text of the article, and then upload the separate chapters or sections of the document as **Associated Files**. These files will appear on the web page alongside the complete document. For more information about uploading associated files, see below.

{ top }

Can I post related files (sound clips, data sets, etc.) alongside the published article?

Yes. The bepress system refers to these supplementary items as **Associated Files**. You will be prompted to submit Associated Files when you upload your submissions. The name of the files you upload will appear on the web site along with your short description of it. Viewers must have the necessary software to open your files; that is not provided by the bepress system.

Please be sure that there are no permissions issues related to use of the associated material. Sometimes, especially with images, you must write a letter seeking permission to use the material before it can be posted.

Also note that where possible, items such as images, charts and tables that are referenced in the document (or otherwise an integral part of the document) should be included directly in the article itself and not posted just as associated files.

{ top }

Can I post a reprint from a journal?

It depends on what the journal allows, which is usually specified in their agreement with the author. If it would not violate copyright to post the reprint on your repository site, you're welcome to do so. Permissions for many publishers can be found at SHERPA RoMEO.

{ top }

A working paper in our repository site has been published in a slightly revised form in a journal. What should I do?

Many journals do not have any restrictions on working papers that preceded an article, especially if substantial revisions were made. The faculty member should check his/her author agreement with the journal to confirm that there is no problem with leaving the working paper on the site. The repository would constitute noncommercial use.

It is a good idea to include the citation to the published article on the cover page for the repository working paper. To add the citation:

1. From your My Account page, click **Submission Management**.
2. Choose the option at the top of the screen to view the **Posted Submissions**.
3. Locate the paper in the list at the bottom of the screen, and click the title.
4. Click **Revise Submission**, scroll to the bottom of the revision form to the **Comments** section, and enter your comment there. Click the button at the bottom of the page to submit the revision.
5. Click the **Update** link to update the article so that the new comment is visible to readers.
6. If you need to remove the full text from the site, click the **Remove Submission** link in the sidebar, and click the confirmation button to remove the submission and notify the author.

{ top }

UNO SelectedWorks Faculty Pages FAQ

What is SelectedWorks?
SelectedWorks is a component of ScholarWorks@UNO that allows UNO faculty to establish a personal author page to highlight research and scholarly efforts. Elements of the page can include personal introductory paragraph, list of publications (full-text can be uploaded directly to your site or linked to in ScholarWorks or publisher site), curriculum vitae, contact information, areas of expertise, courses taught, honors/awards received, and links to other websites. **The individual faculty member has direct control over the page and can update or revise content as frequently as they wish.**

What's the difference between SelectedWorks and ScholarWorks?
ScholarWorks is UNO's institutional repository, an online archive of the research and scholarly output of the University. SelectedWorks are pages focusing on individual faculty members' research interests and accomplishments. These pages are linked to from within ScholarWorks, but reside on a separate platform.

What types of documents can I post on my site?
The default document types for SelectedWorks are Articles, Books, Contributions to Books, Unpublished Papers, Popular Press, Presentations and Other. However, you will be able re-label any of these documents. For example, if you prefer to call Unpublished Papers "Working Papers" you can easily change this for your site. You can also organize your documents by subject, if you prefer.

Am I allowed to post my article if a publisher holds the copyright?
This will depend on the post-print policy of the publisher. Some publishers allow you to post any version of a paper on your personal site; others only allow certain versions, or do not allow any posting after publication. Contact the librarian for your department or the ScholarWorks Coordinator (scholarworks@uno.edu) for more information.

Can I use my site as a bibliography for all my work even if I cannot post all my papers due to copyright concerns?
Yes. SelectedWorks allows you to upload metadata and link directly to your paper on the publisher's site. Furthermore, in the advanced settings you can easily customize your citations so that they appear exactly how you want them to.

How do I create my SelectedWorks page?

- Go to http://works.bepress.com and click on "Start Your Selected Works Site."
- Create a free account and sign in. (You'll have to reply to the link in a confirmation email to activate your account).

How do I assign someone else to post my papers on my site for me?
Make sure you are logged into your SelectedWorks site. Click **My Editors** (found in the top navigational bar). Add the name and email address of the person that you would like to add and click **Save Changes**. Please note, your editors will be able to make any and all changes to your site (including uploading content and adding/revising personal information).

Questions?
Contact the librarian for your department or the ScholarWorks Coordinator (scholarworks@uno.edu) for more information. We are happy to help you with any aspect of the process.

Publication Rights Policies

What is our policy?

For the majority of journals[1] published by Oxford University Press, we have a policy of acquiring a sole and exclusive licence for all published content, rather than asking authors to transfer ownership of their copyright, which has been common practice in the past. We believe this policy more carefully balances the interests of our authors with our need to maintain the viability and reputation of the journals through which our authors are accorded status, recognition and widespread distribution. In developing this policy we have been guided by the following principles:

- As a university press and not-for-profit academic publisher, we rely heavily on the good relationships we have with our authors. Having a licensing policy which enables an author to be identified as the owner of the copyright in an article is one of the key ways of demonstrating how highly we value these relationships.
- An exclusive licence enables the centralised and efficient management of permissions and licencing, ensuring the widest dissemination of the content through intermediaries;
- Exclusive rights also enable OUP to take measures on behalf of our authors against infringement, inappropriate use of an article, libel or plagiarism;
- At the same time, by maintaining exclusive rights, in all media for all published content, we can monitor and uphold the integrity of an article once refereed and accepted for publication to be maintained;

Where to get a copy of the Licence to Publish

OUP cannot publish your article until a completed licence form has been received. You should receive a form as soon as your article is accepted for publication.

Footnotes to this section

1. A small number of OUP Journals still have a policy of requesting a full Assignment of Copyright. If unclear about the policy of the Journal concerned, please contact the Editorial office to clarify.

Government employees

- If you are or were a UK Crown servant and the article has been written in that capacity, we have an arrangement with HMSO to enable us to publish it while acknowledging that it is Crown Copyright. Please inform the Editorial office or Oxford University Press at the time of acceptance or as soon as possible that the article is Crown Copyright, so that we can ensure the appropriate acknowledgement and copyright line are used, as required by our arrangement with HMSO.
- If you are a US Government employee and the article has been written in that capacity, we acknowledge that the Licence to Publish applies only to the extent allowable by US law.

Re-use of third party content as part of your Oxford Journals article

- As part of your article, you may wish to reuse material sourced from third parties such as other publishers, authors, museums, art galleries etc. To assist with this process, we have a Permission Request form and accompanying Guidelines that specifies the rights required in order for third party material to be published as part of your Article. For a copy of this form, please email.
- Responsibility for clearing these third party permissions must be borne by the Author, and this process completed as soon as possible - preferably before acceptance of the manuscript, but if not possible, before the Article reaches the Production stage of the process.

Rights retained by ALL Oxford Journal Authors

- The right, after publication by Oxford Journals, to use all or part of the Article and abstract, for their own personal use, including their own classroom teaching purposes;
- The right, after publication by Oxford Journals, to use all or part of the Article and abstract, in the preparation of derivative works, extension of the article into book-length or in other works, provided that a full acknowledgement is made to the original publication in the journal;
- The right to include the article in full or in part in a thesis or dissertation, provided that this not published commercially;

For the uses specified here, please note that there is no need for you to apply for written permission from Oxford University Press in advance. Please go ahead with the use ensuring that a full acknowledgment is made to the original source of the material including the journal name, volume, issue, page numbers, year of publication, title of article and to Oxford University Press and/or the learned society.

The only exception to this is for the re-use of material for commercial purposes, as defined in the information available via the above url. Permission for this kind of re-use is required and can be obtained by using Rightslink:

With Copyright Clearance Center's Rightslink ® service it's faster and easier than ever before to secure permission from OUP titles to be republished in a coursepack, book, CD-ROM/DVD, brochure or pamphlet, journal or magazine, newsletter, newspaper, make a photocopy, or translate.

- Simply visit: www.oxfordjournals.org and locate your desired content.
- Click on (Order Permissions) within the table of contents and/ or at the bottom article's abstract to open the following page:
- Select the way you would like to reuse the content
- Create an account or login to your existing account
- Accept the terms and conditions and permission is granted

For questions about using the Rightslink service, please contact Customer Support via phone 877/622-5543 (toll free) or 978/777-9929, or email Rightslink customer care.

Preprint use of Oxford Journals content

- For the majority of Oxford Journals, prior to acceptance for publication, authors retain the right to make a pre-print [*A preprint is defined here as un-refereed author version of the article*] version of the article available on your own personal website and/or that of your employer and/or in free public servers of preprints and/or articles in your subject area, provided that where possible.
    - You acknowledge that the article has been accepted for publication in [Journal Title] ©: [year] [owner as specified on the article] Published by Oxford University Press [on behalf of xxxxxx]. All rights reserved.
    - Once the article has been published, we do not require that preprint versions are removed from where they are available. However, we do ask that these are not updated or replaced with the finally published version. Once an article is published, a link could be provided to the final authoritative version on the Oxford Journals Web site. Where possible, the preprint notice should be amended to:
    - This is an electronic version of an article published in [include the complete citation information for the final version of the Article as published in the print edition of the Journal.]
- Once an article is accepted for publication, an author may not make a pre-print available as above or replace an existing pre-print with the final published version. **NB**There are some Oxford Journals such as the Journal of the National Cancer Institute, which do not permit any kind of preprint use. For clarification of the preprint policy for any journal please contact the Rights and New Business Development Department.

Postprint use of Oxford Journals content:

[*A postprint is defined here as being the final draft author manuscript as accepted for publication, following peer review, BUT before it has undergone the copyediting and proof correction process*].

We have detailed policies on the use of postprints for all of our journals. To view these for individual journals please refer to the author self archiving policies on journal homepages. If you require further information please contact the Rights and New Business Development Department.

Other uses by authors should be authorized by Oxford Journals through the Rights and New Business Development Department.

Additional Rights retained by the Author when publishing in an Oxford Open participating journal

Please note that these rights only apply to content published in an Oxford Journal on an Open Access basis in exchange for payment of an author charge. For more details about how Oxford Open works please click here.

The right to reproduce, disseminate or display articles published under this model for educational purposes, provided that:

- the original authorship is properly and fully attributed;
- the Journal and OUP are attributed as the original place of publication with the correct citation details given;
- if an article is subsequently reproduced or disseminated not in its entirety but only in part or as a derivative work this must be clearly indicated
- the right to deposit the postprint and/or URL or PDF of the finally published version of the article into an institutional or centrally organized repository, immediately upon publication

Commercial Use of Open Access version

For all articles published under a Creative Commons Attribution Licence (CC BY 3.0) or an Open Government Licence permission is not required to make any kind of commercial use of the material.

For all articles published under a Creative Commons Non-Commercial Attribution Licence (CC BY NC 3.0) or a Non-Commercial Government Licence permission is required for all commercial reuse. In order to request permission please contact the Rights and New Business Development department: you want to use and a brief description of the intended use.

Commercial re-use guidelines for open access content

Definition of commercial use: any re-use of material from the Open Access part of an Oxford Journal for the commercial gain of the user and/or their employing institution. In particular,

- re-use by a non-author/third party/other publisher of parts of or all of an article or articles in another publication (journal or book) to be sold for commercial purposes. Permission to reproduce selected figures will generally be granted free of charge, although OUP reserves the right to levy a fee for the use of these and/or the full text of an article/articles
- the proactive supply of multiple print or electronic copies of items taken from the Journal to third parties on a systematic basis for marketing purposes. Permission for this kind of reuse should be obtained from the publisher, who retains the right to levy an appropriate fee
- re-use by an author of parts of or all of an article in other publications from commercial organizations. Permission for this kind of reuse should be obtained from the publisher. We would consider this to be commercial reuse but would not normally charge a permission fee if the author is involved.

NB: Please note that any income generated from permissions granted for this kind of use will be returned directly to the journal itself in order to help minimise the costs of making content from it available on an Open Access basis.

Permissions

- All requests to reuse the article, in whole or in part, in another publication will be handled by Oxford Journals. Unless otherwise stated, any permission fees will be retained by the Journal concerned. Where possible, any requests to reproduce substantial parts of the article (including in other Oxford University Press publications) will be subject to your approval (which is deemed to be given if we have not heard from you within 4 weeks of the permission being granted).

ACM Information for Authors

ACM Author Rights

ACM exists to support the needs of the computing community. For over sixty years ACM has developed publications and publication policies to maximize the visibility, impact, and reach of the research it publishes to a global community of researchers, educators, students, and practitioners. ACM has achieved its high impact, high quality, widely-read portfolio of publications with:

- Affordably priced publications
- Liberal Author rights policies
- Wide-spread, perpetual access to ACM publications via a leading-edge technology platform
- Sustainability of the good work of ACM that benefits the profession

Choose

Authors have the option to choose the level of rights management they prefer. ACM offers three different options for authors to manage the publication rights to their work.

- Authors who want ACM to manage the rights and permissions associated with their work, which includes defending against improper use by third parties, can use ACM's traditional copyright transfer agreement.
- Authors who prefer to retain copyright of their work can sign an exclusive licensing agreement, which gives ACM the right but not the obligation to defend the work against improper use by third parties.
- Authors who wish to retain all rights to their work can choose ACM's author-pays option, which allows for perpetual open access through the ACM Digital Library. Authors choosing the author-pays option can give ACM non-exclusive permission to publish, sign ACM's exclusive licensing agreement or sign ACM's traditional copyright transfer agreement.

Post

Authors can post the accepted, peer-reviewed version prepared by the author—known as the "pre-print"—to the following sites, with a DOI pointer to the Definitive Version in the ACM Digital Library.

- On Author's own Home Page *and*
- On Author's Institutional Repository *and*
- In any repository legally mandated by the agency funding the research on which the work is based.
- Prior to submission to ACM for peer-review, authors may post their original work in any informal, non-peer-reviewed aggregation or collection.

Distribute

Authors can post an *Author-Izer* link enabling free downloads of the Definitive Version of the work permanently maintained in the ACM Digital Library

- On the Author's own Home Page *or*
- In the Author's Institutional Repository.

Reuse

Authors can reuse any portion of their own work in a new work of *their own* (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

- Contributing complete papers to any edited collection of reprints for which the author is *not* the editor, requires permission and usually a republication fee.

Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected).

- Commercially produced course-packs that are *sold* to students require permission and possibly a fee.

Create

ACM's copyright and publishing license include the right to make Derivative Works or new versions. For example, translations are "Derivative Works." By copyright or license, ACM may have its publications translated. However, ACM Authors continue to hold perpetual rights to revise their own works without seeking permission from ACM.

- If the revision is minor, i.e., less than 25% of new substantive material, then the work should still have ACM's publishing notice, DOI pointer to the Definitive Version, and be labeled a "Minor Revision of"
- If the revision is major, i.e., 25% or more of new substantive material, then ACM considers this a new work in which the author retains full copyright ownership (despite ACM's copyright or license in the original published article) and the author need only cite the work from which this new one is derived.

Minor Revisions and Updates to works already published in the ACM Digital Library are welcomed with the approval of the appropriate Editor-in-Chief or Program Chair.

Retain

Authors retain all *perpetual rights* laid out in the ACM Author Rights and Publishing Policy, including, but not limited to:

- Sole ownership and control of third-party permissions to use for artistic images intended for exploitation in other contexts
- All patent and moral rights
- Ownership and control of third-party permissions to use of software published by ACM

Obtained from
https://www.ieee.org/publications_standards/publications/rights/permissions_faq.pdf

Frequently Asked Questions Regarding IEEE Permissions

- When is permission to reuse IEEE required?
- From whom do I need permission?
- What if I do not see the "Request Permission" link on either the Table of Contents or the Abstract Page in Xplore?
- Does IEEE require individuals working on a thesis or dissertation to obtain formal permission for reuse?
- If I want to republish an article in another language do I still need to obtain a license from IEEE?
- How do I obtain permission to use photographs or illustrations?
- Do I need to obtain permission to use IEEE material posted on its website?
- Does IEEE require certain rights when requesting permission to use material in an IEEE work?
- What is Rightslink®?
- Is IEEE an STM signatory publisher?

- **When is permission to reuse IEEE required?**

As a general rule, IEEE requires permission be sought to reproduce any substantial part of its intellectual property, including any text, illustrations, charts, tables, photographs, or other material from previously published sources used. IEEE also requires that all references or sources used be credited, whether or not permission is required. For further guidance, please contact pubs-permissions@ieee.org.

- **From whom do I need permission?**

Permission must be sought from IEEE to reuse its intellectual property. In most cases this will mean locating the material you wish to reuse in IEEE Xplore, where you will find a "request permission" link either on the Table of Contents or on the Article Abstract Page.

- **What if I do not see the "Request Permission" link on either the Table of Contents or the Abstract Page in Xplore?**

If you do not see a permission link on the Abstract Page, we recommend you review the front cover and/or the copyright page in the document itself (often, these pages are freely available for viewing in Xplore) in order to determine copyright owner. If you are unsure, please contact pubs-permissions@ieee.org.

- **Does IEEE require individuals working on a thesis or dissertation to obtain formal permission for reuse?**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

Textual Material

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtai the senior author's approval.

Full-Text Article

If you are using the entire IEEE copyright owned article, the following IEEE copyright/ credit notice shoulc be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

- **If I want to republish an article in another language do I still need to obtain a license from IEEE?**

If you are republishing IEEE intellectual property, we do require you obtain a license that includes any translations. The required translation disclaimer and other translation guidelines are available in the IEEE Terms and Conditions contained in the license provided by the Copyright Clearance Center (RightsLink service).

- **How do I obtain permission to use photographs or illustrations?**

IEEE does not always own reproduction rights to photographs or illustrations; rather, such rights may have been retained by the photographer or illustrator. If the source from which the material is borrowed does not indicate who owns reproduction rights, users of these photographs or illustrations are required to locate the rightsholder, directly.

- **Do I need to obtain permission to use IEEE material posted on its website?**

Yes. As a general rule, most material found on the internet is protected by copyright law even if a notice is not displayed. IEEE does require that you inquire about such permission before using any material found on all IEEE copyright owned websites.

- **Does IEEE require certain rights when requesting permission to use material in an IEEE work?**

IEEE does allow permission to reuse small portions of text in another IEEE copyright owned document only (e.g., the equivalent of several paragraphs only) and figures. Our only requirement is that you 1) provide full credit information pertaining to the original IEEE publications (e.g., author name, paper title, publication title, month and year of original publication). Requests for permission to reuse larger portions should be sent to pubs-permissions@ieee.org .

- **What is Rightslink®?**

Rightslink® is the Copyright Clearance Center's automated permissions granting service, which is used by IEEE along with many other STM publishers such as Springer, Elsevier, and Taylor & Francis. Through this permission service, customers can request permission for IEEE Periodical and Conference content from the point of access; (normally found on the abstract page of the individual article, in IEEE Xplore).

- **Is IEEE an STM signatory publisher?**

No, IEEE is not a signatory to the STM (International Association of Scientific, Technical & Medical Publishers) Permissions Guidelines, last updated February 2012.

April 2013, nb

REFERENCES

L. Acharya, T. Judeh, Z. Duan, M. Rabbat, and D. Zhu, "Gsgs: a computational approach to reconstruct signaling pathway structures from gene sets," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 9, no. 2, pp. 438–50, 2012.

L. R. Acharya, T. Judeh, G. Wang, and D. Zhu, "Optimal structural inference of signaling pathways from unordered and overlapping gene sets," *Bioinformatics*, vol. 28, no. 4, pp. 546–56, 2012.

L. Acharya, T. Judeh, and D. Zhu, *A Survey of Computational Approaches to Reconstruct and Partition Biological Networks*. John Wiley & Sons, Inc., 2012, pp. 1–43.

B. Adamcsek, G. Palla, I. J. Farkas, I. Derenyi, and T. Vicsek, "Cfinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021–3, 2006.

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular Biology of the Cell*, 4th ed. New York: Garland Science, 2002.

N. Alon, R. Yuster, and U. Zwick, "Color-coding," *Journal of the ACM*, vol. 42, no. 4, pp. 844–856, 1995.

U. Alon, "Network motifs: theory and experimental approaches," *Nat Rev Genet*, vol. 8, no. 6, pp. 450–61, 2007.

J. M. Anthonisse, "The rush in a directed graph," *Stichting Mathematisch Centrum. Mathematische Besliskunde*, no. BN 9/71, pp. 1–10, 1971.

J. Bang-Jensen and G. Z. Gutin, *Digraphs: theory, algorithms and applications*. Springer, 2009.

M. Bansal, G. Della Gatta, and D. di Bernardo, "Inference of gene regulatory networks and compound mode of action from time course gene expression profiles," *Bioinformatics*, vol. 22, no. 7, pp. 815–22, 2006.

R. Bianco, D. Melisi, F. Ciardiello, and G. Tortora, "Key cancer cell signal transduction pathways as therapeutic targets," *Eur J Cancer*, vol. 42, no. 3, pp. 290–4, 2006.

U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

N. Bruno, N. Koudas, and D. Srivastava, "Holistic twig joins: optimal xml pattern matching," in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, 2002, Conference Proceedings, pp. 310–321.

L. Chen, A. Gupta, and M. E. Kurul, "Stack-based algorithms for pattern matching on dags," in *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, Conference Proceedings, pp. 493–504.

X. Chen, J. Xu, B. Huang, J. Li, X. Wu, L. Ma, X. Jia, X. Bian, F. Tan, L. Liu, S. Chen, and X. Li, "A sub-pathway-based approach for identifying drug response principal network," *Bioinformatics*, vol. 27, no. 5, pp. 649–54, 2011.

Y. Cheng and G. M. Church, "Biclustering of expression data," pp. 93–103, 2000.

Y.-J. Chu and T.-H. Liu, "On the shortest arborescence of a directed graph," *Science Sinica*, vol. 14, no. 1396-1400, p. 270, 1965.

H. J. Chung, M. Kim, C. H. Park, J. Kim, and J. H. Kim, "Arrayxpath: mapping and visualizing microarray gene-expression data with integrated biological pathway resources using scalable vector graphics," *Nucleic Acids Res*, vol. 32, no. Web Server issue, pp. W460–4, 2004.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 2009.

D. Croft, G. O'Kelly, G. Wu, R. Haw, M. Gillespie, L. Matthews, M. Caudy, P. Garapati, G. Gopinath, B. Jassal, S. Jupe, I. Kalatskaya, S. Mahajan, B. May, N. Ndegwa, E. Schmidt, V. Shamovsky, C. Yung, E. Birney, H. Hermjakob, P. D'Eustachio, and L. Stein, "Reactome: a database of reactions, pathways and biological processes," *Nucleic Acids Res*, vol. 39, no. Database issue, pp. D691–7, 2011.

K. D. Dahlquist, N. Salomonis, K. Vranizan, S. C. Lawlor, and B. R. Conklin, "Genmapp, a new tool for viewing and analyzing microarray data on biological pathways," *Nat Genet*, vol. 31, no. 1, pp. 19–20, 2002.

D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, E. L. Eastwood, A. P. Wojtovich, S. J. Elliott, S. E. Schaus, and J. J. Collins, "Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks," *Nature biotechnology*, vol. 23, no. 3, pp. 377–383, 2005.

B. Dost, T. Shlomi, N. Gupta, E. Ruppin, V. Bafna, and R. Sharan, "Qnet: a tool for querying protein interaction networks," *J Comput Biol*, vol. 15, no. 7, pp. 913–25, 2008.

J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, vol. 71B, no. 4, p. 233, 1967.

B. Efron and R. Tibshirani, "On testing the significance of sets of genes," *The Annals of Applied Statistics*, pp. 107–129, 2007.

J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphvizopen source graph drawing tools," in *Graph Drawing*. Springer, 2002, Conference Proceedings, pp. 483–484.

A. P. Engelbrecht, *Computational intelligence: an introduction*. Wiley. com, 2007.

G. Even, J. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.

V. Fionda and L. Palopoli, "Biological network querying techniques: analysis and comparison," *J Comput Biol*, vol. 18, no. 4, pp. 595–625, 2011.

S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using bayesian networks to analyze expression data," *J Comput Biol*, vol. 7, no. 3-4, pp. 601–20, 2000.

N. Friedman, I. Nachman, and D. Per, "Learning bayesian network structure from massive datasets: the sparse candidate algorithm," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, Conference Proceedings, pp. 206–215.

E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo, "A technique for drawing directed graphs," *IEEE Transactions on Software Engineering*, vol. 19, no. 3, pp. 214–230, 1993.

M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proc Natl Acad Sci U S A*, vol. 99, no. 12, pp. 7821–6, 2002.

F. Glover, E. Taillard, and D. de Werra, "A user's guide to tabu search," *Annals of Operations Research*, vol. 41, no. 1-4, pp. 3–28, 1993.

M. A. Gmez, Villegas, P. M. , H. Navarro, Susi, and R. , "Dealing with uncertainty in gaussian bayesian networks from a regression perspective," *on Probabilistic Graphical Models*, p. 145, 2010.

M. A. Gmez-Villegas, P. Man, and R. Susi, "Sensitivity analysis in gaussian bayesian networks using a divergence measure," *Communications in Statistics - Theory and Methods*, vol. 36, no. 3, pp. 523–539, 2007.

P. Grosu, J. P. Townsend, D. L. Hartl, and D. Cavalieri, "Pathway processor: a tool for integrating whole-genome expression results into metabolic networks," *Genome Res*, vol. 12, no. 7, pp. 1121–6, 2002.

K. L. Gunderson, S. Kruglyak, M. S. Graige, F. Garcia, B. G. Kermani, C. Zhao, D. Che, T. Dickinson, E. Wickham, J. Bierle, D. Doucet, M. Milewski, R. Yang, C. Siegmund, J. Haas, L. Zhou, A. Oliphant, J. B. Fan, S. Barnard, and M. S. Chee, "Decoding randomly ordered dna arrays," *Genome Res*, vol. 14, no. 5, pp. 870–7, 2004.

S. Hashemikhabir, E. S. Ayaz, Y. Kavurucu, T. Can, and T. Kahveci, "Large-scale signaling network reconstruction," *IEEE/ACM Trans Comput Biol Bioinform*, vol. 9, no. 6, pp. 1696–708, 2012.

M. Holec, F. elezn, J. Klma, and J. Tolar, *Integrating multiple-platform expression data through gene set features.* Springer, 2009, pp. 5–17.

T. R. Hughes, M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, C. D. Armour, H. A. Bennett, E. Coffey, H. Dai, Y. D. He, M. J. Kidd, A. M. King, M. R. Meyer, D. Slade, P. Y. Lum, S. B. Stepaniants, D. D. Shoemaker, D. Gachotte, K. Chakraburtty, J. Simon, M. Bard, and S. H. Friend, "Functional discovery via a compendium of expression profiles," *Cell*, vol. 102, no. 1, pp. 109–26, 2000.

I. Ispolatov and S. Maslov, "Detection of the dominant direction of information flow and feedback links in densely interconnected regulatory networks," *BMC Bioinformatics*, vol. 9, no. 1, p. 424, 2008.

N. K. V. Jayesh Kumar Gupta, Sumanik Singh, "Matlab Toolbox for Biclustering Analysis (MTBA)," http://iitk.ac.in/iil/mtba/.

T. Judeh, L. Acharya, and D. Zhu, "Gene network inference via linear path augmentation," in *BIOT 2010*, 2010, Conference Proceedings.

T. Judeh, T. Nguyen, and D. Zhu, "Qsea for fuzzy subgraph querying of kegg pathways." ACM, 2012, Conference Paper, pp. 474–481.

T. Judeh, T. Jayyousi, L. Acharya, R. G. Reynolds, and D. Zhu, "Gene set cultural algorithm: A cultural algorithm approach to reconstruct networks from gene sets." ACM, 2013, Conference Paper, pp. 641–648.

T. Judeh, C. Johnson, A. Kumar, and D. Zhu, "Teak: topology enrichment analysis framework for detecting activated biological subpathways," *Nucleic Acids Res*, vol. 41, no. 3, pp. 1425–37, 2013.

T. Judeh, T. Jayyousi, L. Acharya, R. G. Reynolds, and D. Zhu, "Gsca: Reconstructing biological pathway topologies using a cultural algorithms approach," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014, Conference Proceedings.

T. Judeh, "Sea: a novel computational and gui software pipeline for detecting activated biological sub-pathways," Thesis, The University of New Orleans, 2011.

L. Kaderali, E. Dazert, U. Zeuge, M. Frese, and R. Bartenschlager, "Reconstructing signaling pathways from rnai data using probabilistic boolean threshold networks," *Bioinformatics*, vol. 25, no. 17, pp. 2229–35, 2009.

M. Kanehisa and S. Goto, "Kegg: kyoto encyclopedia of genes and genomes," *Nucleic Acids Res*, vol. 28, no. 1, pp. 27–30, 2000.

M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe, "Kegg for integration and interpretation of large-scale molecular data sets," *Nucleic Acids Res*, vol. 40, no. Database issue, pp. D109–14, 2012.

B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, 1970.

P. Khatri and S. Draghici, "Ontological analysis of gene expression data: current tools, limitations, and open problems," *Bioinformatics*, vol. 21, no. 18, pp. 3587–95, 2005.

Y. Kim, S. W. Son, and H. Jeong, "Finding communities in directed networks," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 81, no. 1 Pt 2, p. 016103, 2010.

J. Klema, M. Holec, F. Zelezny, and J. Tolar, *Comparative evaluation of set-level techniques in microarray classification*. Springer, 2011, pp. 274–285.

D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

E. A. Leicht and M. E. Newman, "Community structure in directed networks," *Phys Rev Lett*, vol. 100, no. 11, p. 118703, 2008.

C. Li, X. Li, Y. Miao, Q. Wang, W. Jiang, C. Xu, J. Li, J. Han, F. Zhang, B. Gong, and L. Xu, "Subpathwayminer: a software package for flexible identification of pathways," *Nucleic Acids Res*, vol. 37, no. 19, p. e131, 2009.

S. Li, "Mechanisms of cellular signal transduction," *Int J Biol Sci*, vol. 1, no. 4, p. 152, 2005.

Y. Li, L. Liu, X. Bai, H. Cai, W. Ji, D. Guo, and Y. Zhu, "Comparative study of discretization methods of microarray data for inferring transcriptional regulatory networks," *BMC Bioinformatics*, vol. 11, no. 1, p. 520, 2010.

Y. Liu and H. Zhao, "A computational approach for ordering signal transduction pathway components from genomics and proteomics data," *BMC Bioinformatics*, vol. 5, no. 1, p. 158, 2004.

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," *Nat Biotechnol*, vol. 14, no. 13, pp. 1675–80, 1996.

D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano, "Generating realistic in silico gene networks for performance assessment of reverse engineering methods," *J Comput Biol*, vol. 16, no. 2, pp. 229–39, 2009.

D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky, "Revealing strengths and weaknesses of methods for gene network inference," *Proc Natl Acad Sci U S A*, vol. 107, no. 14, pp. 6286–91, 2010.

A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera, and A. Califano, "Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7 Suppl 1, no. Suppl 1, p. S7, 2006.

L. Matthews, G. Gopinath, M. Gillespie, M. Caudy, D. Croft, B. de Bono, P. Garapati, J. Hemish, H. Hermjakob, B. Jassal, A. Kanapin, S. Lewis, S. Mahajan, B. May, E. Schmidt, I. Vastrik, G. Wu, E. Birney, L. Stein, and P. D'Eustachio, "Reactome knowledgebase of human biological pathways and processes," *Nucleic Acids Res*, vol. 37, no. Database issue, pp. D619–22, 2009.

K. Murphy, "The bayes net toolbox for matlab," *Computing science and statistics*, vol. 33, no. 2, pp. 1024–1034, 2001.

C. J. Needham, J. R. Bradford, A. J. Bulpitt, and D. R. Westhead, "A primer on learning in bayesian networks for computational biology," *PLoS Comput Biol*, vol. 3, no. 8, p. e129, 2007.

M. E. Newman, "Modularity and community structure in networks," *Proc Natl Acad Sci U S A*, vol. 103, no. 23, pp. 8577–82, 2006.

M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 69, no. 2 Pt 2, p. 026113, 2004.

G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–8, 2005.

G. Palla, I. J. Farkas, P. Pollner, I. Dernyi, and T. Vicsek, "Directed network modules," *New Journal of Physics*, vol. 9, no. 6, pp. 186–186, 2007.

D. Pan, N. Sun, K. H. Cheung, Z. Guan, L. Ma, M. Holford, X. Deng, and H. Zhao, "Pathmapa: a tool for displaying gene expression and performing statistical tests on metabolic pathways at multiple levels for arabidopsis," *BMC Bioinformatics*, vol. 4, p. 56, 2003.

R. Pandey, R. K. Guru, and D. W. Mount, "Pathway miner: extracting gene association networks from molecular pathways for predicting the biological significance of gene expression microarray data," *Bioinformatics*, vol. 20, no. 13, pp. 2156–8, 2004.

R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky, "Towards a rigorous assessment of systems biology models: the dream3 challenges," *PLoS One*, vol. 5, no. 2, p. e9202, 2010.

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 *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE*

International Conference on, vol. 3. IEEE, 2005, Conference Proceedings, pp. iii/977–iii/980 Vol. 3.

M. G. Rabbat, M. A. T. Figueiredo, and R. D. Nowak, "Network inference from co-occurrences," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4053–4068, 2008.

R. G. Reynolds and Y. A. Gawasmeh, "Evolving heterogeneous social fabrics for the solution of real valued optimization problems using cultural algorithms," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, Conference Proceedings, pp. 1–8.

R. G. Reynolds, *An adaptive computer model of the evolution of agriculture for hunter-gatherers in the valley of Oaxaca, Mexico*, 1979.

——, "An introduction to cultural algorithms," in *Proceedings of the third annual conference on evolutionary programming*. World Scientific, 1994, Conference Proceedings, pp. 131–139.

S. J. Roberts and W. D. Penny, "Variational bayes for generalized autoregressive models," *IEEE Transactions on Signal Processing*, vol. 50, no. 9, pp. 2245–2257, 2002.

M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc Natl Acad Sci U S A*, vol. 105, no. 4, pp. 1118–23, 2008.

——, "Mapping change in large networks," *PLoS One*, vol. 5, no. 1, p. e8694, 2010.

T. Schaffter, D. Marbach, and D. Floreano, "Genenetweaver: in silico benchmark generation and performance profiling of network inference methods," *Bioinformatics*, vol. 27, no. 16, pp. 2263–70, 2011.

M. Schena, D. Shalon, R. W. Davis, and P. O. Brown, "Quantitative monitoring of gene expression patterns with a complementary dna microarray," *Science*, vol. 270, no. 5235, pp. 467–70, 1995.

G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller, and N. Friedman, "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data," *Nat Genet*, vol. 34, no. 2, pp. 166–76, 2003.

J. C. Selcher, E. J. Weeber, A. W. Varga, J. D. Sweatt, and M. Swank, "Book review: Protein kinase signal transduction cascades in mammalian associative conditioning," *The Neuroscientist*, vol. 8, no. 2, pp. 122–131, 2002.

R. D. Shachter and C. R. Kenley, "Gaussian influence diagrams," *Management Science*, vol. 35, no. 5, pp. 527–550, 1989.

P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome Res*, vol. 13, no. 11, pp. 2498–504, 2003.

R. Sharan and T. Ideker, "Modeling cellular machinery through biological network comparison," *Nat Biotechnol*, vol. 24, no. 4, pp. 427–33, 2006.

J. Shendure and H. Ji, "Next-generation dna sequencing," *Nat Biotechnol*, vol. 26, no. 10, pp. 1135–45, 2008.

J. Shendure, R. D. Mitra, C. Varma, and G. M. Church, "Advanced sequencing technologies: methods and goals," *Nat Rev Genet*, vol. 5, no. 5, pp. 335–44, 2004.

K. Shinozaki and K. Yamaguchi-Shinozaki, "Gene expression and signal transduction in water-stress response," *Plant Physiol*, vol. 115, no. 2, pp. 327–334, 1997.

T. Shlomi, D. Segal, E. Ruppin, and R. Sharan, "Qpath: a method for querying pathways in a protein-protein interaction network," *BMC Bioinformatics*, vol. 7, no. 1, p. 199, 2006.

I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–74, 2002.

A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov, "Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles," *Proc Natl Acad Sci U S A*, vol. 102, no. 43, pp. 15 545–50, 2005.

A. Subramanian, H. Kuehn, J. Gould, P. Tamayo, and J. P. Mesirov, "Gsea-p: a desktop application for gene set enrichment analysis," *Bioinformatics*, vol. 23, no. 23, pp. 3251–3, 2007.

A. L. Tarca, S. Draghici, P. Khatri, S. S. Hassan, P. Mittal, J. S. Kim, C. J. Kim, J. P. Kusanovic, and R. Romero, "A novel signaling pathway impact analysis," *Bioinformatics*, vol. 25, no. 1, pp. 75–82, 2009.

R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

M. Teyssier and D. Koller, "Ordering-based search: A simple and effective algorithm for learning bayesian networks," *arXiv preprint arXiv:1207.1429*, 2012.

Y. Tian, R. C. McEachin, C. Santos, D. J. States, and J. M. Patel, "Saga: a subgraph matching tool for biological graphs," *Bioinformatics*, vol. 23, no. 2, pp. 232–9, 2007.

C. J. Vaske, S. C. Benz, J. Z. Sanborn, D. Earl, C. Szeto, J. Zhu, D. Haussler, and J. M. Stuart, "Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using paradigm," *Bioinformatics*, vol. 26, no. 12, pp. i237–45, 2010.

J.-P. Vert, "Reconstruction of biological networks by supervised machine learning approaches," *arXiv preprint arXiv:0806.0215*, 2008.

H. Yu and M. Gerstein, "Genomic analysis of the hierarchical structure of regulatory networks," *Proc Natl Acad Sci U S A*, vol. 103, no. 40, pp. 14 724–31, 2006.

W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452–473, 1977.

D. Zhu, "Semi-supervised gene shaving method for predicting low variation biological pathways from genome-wide data," *BMC Bioinformatics*, vol. 10 Suppl 1, no. Suppl 1, p. S54, 2009.

P. Zoppoli, S. Morganella, and M. Ceccarelli, "Timedelay-aracne: Reverse engineering of gene networks from time-course data by an information theoretic approach," *BMC Bioinformatics*, vol. 11, no. 1, p. 154, 2010.

ABSTRACT

TEAK: A NOVEL COMPUTATIONAL AND GUI SOFTWARE PIPELINE FOR RECONSTRUCTING BIOLOGICAL NETWORKS, DETECTING ACTIVATED BIOLOGICAL SUBNETWORKS, AND QUERYING BIOLOGICAL NETWORKS.

by

Thair Judeh

August 2014

Advisor: Dr. Dongxiao Zhu

Major: Computer Science

Degree: Doctor of Philosophy

As high-throughput gene expression data becomes cheaper and cheaper, researchers are faced with a deluge of data from which biological insights need to be extracted and mined since the rate of data accumulation far exceeds the rate of data analysis. There is a need for computational frameworks to bridge the gap and assist researchers in their tasks. The Topology Enrichment Analysis frameworK (TEAK) is an open source GUI and software pipeline that seeks to be one of many tools that fills in this gap and consists of three major modules. The first module, the Gene Set Cultural Algorithm, de novo infers biological networks from gene sets using the KEGG pathways as prior knowledge. The second and third modules query against the KEGG pathways using molecular profiling data and query graphs, respectively. In particular, the second module, also called TEAK, is a network partitioning module that partitions the KEGG pathways into both linear and nonlinear subpathways. In conjunction with molecular profiling data, the subpathways are ranked and displayed to the user within the TEAK GUI. Using a public microarray yeast data set, previously unreported fitness defects for dpl1Δ and lag1Δ mutants under conditions of

nitrogen limitation were found using TEAK. Finally, the third module, the Query Structure Enrichment Analysis framework, is a network query module that allows researchers to query their biological hypotheses in the form of Directed Acyclic Graphs against the KEGG pathways.

AUTOBIOGRAPHICAL STATEMENT

Thair Judeh 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. After receiving a Master's degree in Computer Science, he later on transferred with his advisor Dr. Dongxiao Zhu to Wayne State University to continue his Ph.D. studies. His research interests include reconstructing, decomposing, and querying biological networks.