

SAR: AN ALGORITHM FOR A STRUCTURAL ANALYSIS OF PROTEIN INTERACTION NETWORKS

By
CHARLES ALLEN SCHULTZ II

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science in the
Department of Mathematics & Computer Science

Claflin University
Orangeburg, South Carolina
March 2016

Dr. Ananda Mondal
Thesis Advisor

Dr. Ramaier Sriram
Committee Member

Dr. Selvarajah Mohanarajah
Committee Member

Dr. Ananda Mondal
Academic Advisor

Received by: _____ Date: _____
Dr. Ramaier Sriram, *Department of Mathematics & Computer Science*

Received by: _____ Date: _____
Dr. Verlie Tisdale, *Dean, School of Natural Sciences & Mathematics*

Received by: _____ Date: _____
Dr. Angela Peters, *Vice Provost for Academic Programs*

ACKNOWLEDGEMENTS

I would like to first thank Claflin University for providing me with the opportunity to perform this research on campus over the past three years, and providing funding via the SCAMP and NSF HBCU grants.

Additionally, I would like to thank the Computer Science department for providing me with an innate understanding of my field through many, many hours of instruction which prepared me not only for this research, but for my future career as a Computer Scientist.

Finally, I would like to thank my mentor, Dr. Ananda Mondal, for providing me with the guidance needed to bring my research this far. Without him I wouldn't have been able to find my way in the myriad of mystery and discovery that is scientific inquiry.

THESIS STATEMENT

By properly deconstructing a source graph of protein interactions into meaningful subgraphs, applying Network Ontology Analysis, and collating these results one can obtain a better visual representation of the primary ontological components involved within a particular network.

ABSTRACT

COMPUTER SCIENCE

CHARLES ALLEN SCHULTZ II

B.S. CLAFLIN UNIVERSITY, 2016

SAR: AN ALGORITHM FOR A STRUCTURAL ANALYSIS OF PROTEIN INTERACTION NETWORKS

Thesis Advisor: Dr. Ananda Mondal

Thesis Dated: March, 2016

Motivation: Gene Term Enrichment Analysis is a commonly used way to analyze protein expression data gathered by techniques such as microarray assays. Common methods of Enrichment Analysis have analyzed the genes in list form. Improvements by other researchers have led to a network-based analysis (NOA). Still, the quantity and complexity of data produced by such an analysis is often difficult to interpret directly. This leads to difficulty communicating the significance of a network via graphical representations due to hard to interpret diagrams and figures produced by existing methods.

Results: As a solution to this, the researcher developed the SAR algorithm. SAR, which stands for *Segmentation, Analysis, and Reintegration*, is capable of reducing a network into an easily interpreted representation. This is accomplished via careful application of network-based Gene Term Enrichment to selected subgraphs of the original gene network under consideration.

Application: SAR is immediately applicable in condensing the raw output of gene ontology analysis to a form that highlights the most significant ontological components interacting within the source network.

KEYWORDS AND ABBREVIATIONS

Keywords: gene-term enrichment analysis, gene ontology, protein interaction networks, cytoscape, subgraphs, algorithm, SAR

Abbreviations:

BFS	Breadth-First Search (Algorithm)
BSG	Bipartite Subgraph
CSG	Clique Subgraph
DFS	Depth-First Search (Algorithm)
GO	Gene Ontology
NOA	Network Ontology Analysis (Algorithm)
PPI	Protein-Protein Interaction
SAR	Segmentation, Analysis, and Reintegration (Algorithm)
SNB	Subnetwork Biomarker

LIST OF TABLES

		<u>PAGE</u>
Table 1.	Subset of Breast Cancer Subnetwork Biomarker	5
Table 2.	Typical NOA Analysis Results	15
Table 3.	Typical BiNGO Analysis Results	15
Table 4.	Unordered Adjacency Matrix	22
Table 5.	Ordered Adjacency Matrix	22

LIST OF FIGURES

	<u>PAGE</u>
Figure 1.	Depiction of SAR Algorithm 7
Figure 2.	Bipartite Subgraph 9
Figure 3.	Biological Significance of Bipartite Graphs 9
Figure 4.	Clique Subgraph 9
Figure 5.	Biological Significance of Clique Graphs 9
Figure 6.	Breast Cancer Subnetwork Biomarker 12
Figure 7.	Bipartite Subgraph in Breast Cancer SNB 12
Figure 8.	Clique Subgraph in Breast Cancer SNB 12
Figure 9.	Typical NOA Analysis Results 14
Figure 10.	Typical BiNGO Analysis Results 14
Figure 11.	Breast Cancer non-SAR Enrichment Analysis 18
Figure 12.	Breast Cancer SAR Enrichment Analysis 18
Figure 13.	An Example Bipartite Graph 22
Figure 14.	Branching Process 24
Figure 15.	BFS – “Nearness” Example 26
Figure 16.	BFS – Process 26
Figure 17.	DFS – “Chains” Example 26
Figure 18.	DFS – Process 26
Figure 19.	Subgraph Finder General Process 31
Figure 20.	Constructive (Search-Based) Algorithm Process 31
Figure 21.	Edge Preservation Demonstration 34
Figure 22.	Make Undirected – Additive Demonstration 36
Figure 23.	Configuration Task Screenshot 38

TABLE OF CONTENTS

	<u>PAGE</u>
Acknowledgements.....	ii
Thesis Statement.....	iii
Abstract.....	iv
List of Keywords and Abbreviations	v
List of Tables.....	vi
List of Figures.....	vii
Table of Contents.....	viii
Introduction.....	1
Background.....	2
Materials.....	4
The SAR Algorithm.....	6
Segmentation.....	8
Analysis.....	13
Reintegration.....	17
Methods.....	20
Inefficient Segmentation Approaches.....	20
Efficient Segmentation Approaches.....	25
Discussion.....	28
Automating the Algorithm.....	29
Conclusion and Future Works.....	40
References.....	41

INTRODUCTION

This research began in the year 2014 with an inquiry into the existence of any correlations between ontological components associated with a Protein Protein Interaction network and its subgraphs. It was born out of a desire to analyze the aforementioned PPI networks in search of information that may assist in the comprehension of disease and its evolution.

Over time, this research changed slowly in scope and depth to become what it is today: a description of an algorithm that may be useful in the analysis of PPI networks and, perhaps, other biological networks as well. Parts of the algorithm, specifically those which were the object of the most investigation in the beginning, have been automated and developed into existing software applications for release to the public. Other parts still need further automation.

All-in-all, the algorithm as developed so far is outlined in the following document. The discussion begins with a little background on the realm of Bioinformatics to which the algorithm is applicable and follows this up with a description of the materials utilized within this research. Then the Segmentation component of the SAR algorithm is described in detail due to being the primary avenue of new research and development performed by the researcher. Finally, a brief overview of typical results are given and a discussion about the research as a whole is made, specifically with information as to the eventual full automation of the algorithm for mass data processing and statistical evaluation of its performance.

BACKGROUND

Bioinformatics is a multidisciplinary field that brings together the vast datasets composed in biological research with modern-day computational power brought about by the information age. One definition of bioinformatics is the application of computational tools and techniques to the management and analysis of biological data.^[1] Another source defines bioinformatics in a three pronged manner: “bioinformatics is conceptualising biology in terms of molecules ... and applying 'informatics techniques' ... to understand and organize the information associated with these molecules on a large scale.”^[2] Ultimately, it is a way for the biologists of today to harness the massive computing power made available by the technological revolution of the late twentieth century for better understanding of biological phenomena.

Bioinformatics has been applied to many problems faced by today's scientists. Of particular interest to this research is the area of Gene Term Enrichment Analysis. Gene Term Enrichment Analysis, also referred to as Gene Ontology Enrichment Analysis is the statistical process of determining which genes within a set are over- or under-represented; that is to say, it is a process that identifies which gene products occur more or less frequently than one would suspect them to in a random sampling.^[3] The Gene Ontology itself is composed of many terms representing these products. These are generally grouped into three domains: Cellular Components, Molecular Functions, and Biological Processes.^[4]

Over the years researchers have developed many ways for performing Gene Term Enrichment Analysis. Notably, various statistical methods such as Fisher's Exact Text and the Hypergeometric Test (which is a special case of Fisher's Exact Test) are utilized to determine if a particular gene product is over or under represented in a sample.^[5] In 2003, researchers further improved Gene Term Enrichment methodologies with the development of “false discovery rate controlling procedures” which reduce the number of false identifications within an Enrichment Analysis.^[6]

Several common tools for performing Enrichment Analysis include the AmiGO, GOrilla, BiNGO, and NOA toolsets. AmiGO is a web-based tool developed by the Gene Ontology Consortium for accessing their annotation information.^[7] GOrilla is another web-based tool that operates on a list of genes.^[8] BiNGO^[9] and NOA^[10] are both Cytoscape apps (plugins) that perform Enrichment Analysis in various ways. The former operates on a list of genes while the latter is useful in analyzing gene networks.

Cytoscape is another common bioinformatics tool which a few of the aforementioned Gene Term Enrichment Analysis tools are integrated with. Cytoscape is “an open source platform for visualizing complex networks and integrating these with any type of attribute data.”^[11] In short, it is a powerful environment for developing bioinformatics tools, performing bioinformatics research, and is a central piece to this thesis.

Ultimately, this research aims to couple Gene Term Enrichment Analysis with the analysis of specific PPI subnetworks referred to as Subnetwork Biomarkers. Utilizing Cytoscape, these SNBs shall be analyzed in a piecewise manner via the previously mentioned Cytoscape apps for Enrichment Analysis and the results collated to obtain better representations of the ontological components within.

MATERIALS

The materials utilized by this research can be divided into two categories: data and tools. The first category consists of a large set of Subnetwork Biomarkers provided by the research efforts of Timalsina *et al.* In total, there are 144 Subnetwork Biomarkers for various human pathways and diseases within the data set. Of note are the biomarkers for various cancers such as Breast Cancer, Lung Cancer, Liver Cancer, Leukemia and Lymphoma. For more information about the identification of these Subnetwork Biomarkers please see the paper provided by Timalsina *et al.*^[12]

Each SNB is actually a Protein-Protein Interaction network. Specifically, each SNB is an undirected network where each node represents a protein and each edge is an interaction between two proteins. Edges are annotated with a value ranging from 150 to 999. This edge weight determines the likelihood of two proteins interacting. Table 1 contains a subset of the Breast Cancer Subnetwork Biomarker.

The second category of materials utilized by the researcher are the tools used in the data processing. The aforementioned Cytoscape visualization suite and two plugins were the best choices for performing complex visual analyses on the data. Due to the evolution of the research, however, two separate versions of Cytoscape were used. Initially, version 2.8 of Cytoscape was used for visualizing the subgraphs found within an SNB, but this changed in early 2015 when the researcher developed a Cytoscape 3 app for identifying subgraphs within a network. Cytoscape 3 was chosen since it contained a better structured App environment for integrating custom code with the visualization suite. Additionally, Cytoscape 2 is in the process of slowly being phased out.

Additionally, the Network Ontology Analysis (NOA) plugin developed by Wang *et al.* was utilized to perform Gene Term Enrichment Analysis (manually) on the subgraphs identified by the researcher. Given the constant evolution of this research, however, NOA is no longer a viable option for analysis in Cytoscape 3 without an upgrade of its codebase. As such, the BiNGO tool is a better alternative for continued research until the NOA plugin for Cytoscape 2 can be upgraded to an app for Cytoscape 3.

TABLE 1. SUBSET OF BREAST CANCER SUBNETWORK BIOMARKER

Protein A	Protein B	PPI Score
ABCB1	ABCG2	469
ABCB1	ADAM23	193
ABCB1	AKT1	215
ABCB1	AR	215
ABCB1	BCL2	954
ABCB1	BIRC5	352
ABCB1	CDKN1A	215
ABCB1	CTSD	235
ABCB1	EGF	800
ABCB1	EGFR	284
ABCB1	ERBB2	623
ABCB1	ESR1	247
ABCB1	GLI1	227
ABCB1	GSTP1	823
ABCB1	IGF1	224
ABCB1	IL6	217
ABCB1	JUN	832
ABCB1	MGMT	643
ABCB1	NME1	274
ABCB1	NR3C1	216
ABCB1	PGR	402
ABCB1	PTEN	247
...	...	...

A subset of the Breast Cancer Subnetwork Biomarker. Each row of the table represents a Protein-Protein Interaction. Values in the PPI Score column may range from 150 (least interactive) to 999 (most interactive).

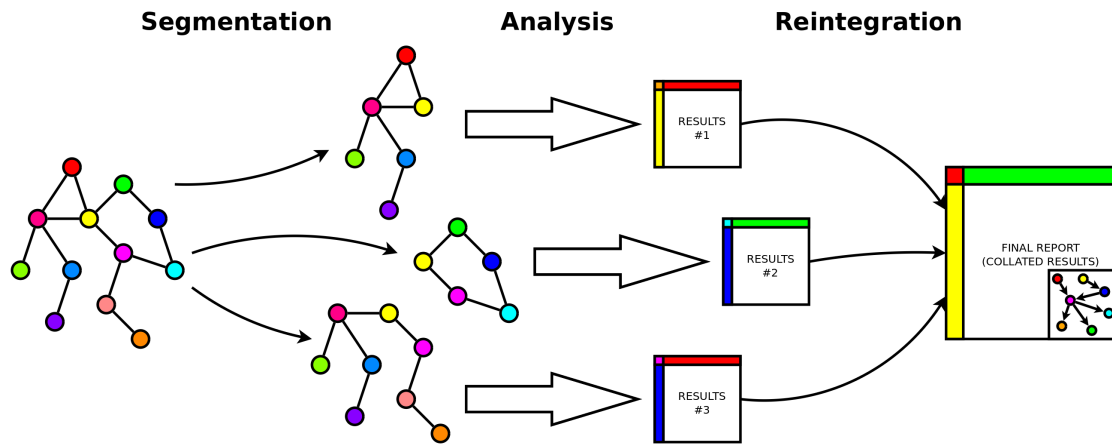
THE SAR ALGORITHM

The SAR Algorithm is an iterative algorithm composed of three separate stages from which it obtains its name: Segmentation, Analysis, and Reintegration. Each stage operates independently of the others and must be completed prior to continuing on to the following stage. Each of the stages are outlined in sequence below:

1. Segmentation – The source graph is broken into component subgraphs via an appropriate metric.
2. Analysis – Each component subgraph is independently analyzed via a Gene Term Enrichment algorithm or tool.
3. Reintegration – The results for each of the analyses performed on the subgraphs in the second stage are collated to form a single report and / or representation.

This process is outlined in Figure 1. The algorithm takes a single input, a biological network, and passes this to the first stage: Segmentation. Segmentation then breaks the original network down into component subgraphs based upon user configuration of the Subgraph Finder. Each of these subgraphs are then processed via the second stage: Analysis. It is during this stage that each subgraph is processed via a Gene Term Enrichment Algorithm (or tool) and the results saved. Finally, the results produced in the second stage are collated together in the Reintegration stage to create a final report or visual representation.

More information and details on each of the stages is presented in the following sub-sections.

FIGURE 1. DEPICTION OF SAR ALGORITHM

This is a depiction of the SAR Algorithm and its stages. A single graph is supplied as input (on the left), broken down into subgraphs, analyzed for over- or under-expressed gene products, and a report and / or visualization created from the results. It should be noted that all three stages can receive various levels of user configuration.

– *Segmentation* –

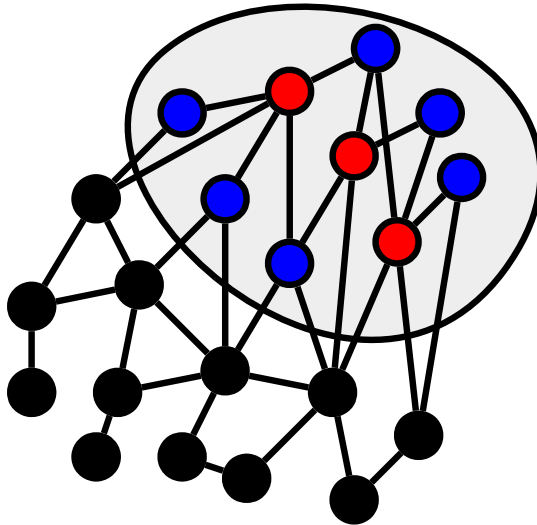
The Segmentation stage is the first stage of the SAR Algorithm process. It was also the stage in which the most research and work was performed by the researcher. This was primarily due to an interest in various forms of subgraphs (such as Bipartite Graphs and Cliques, described below) and also because there already exist prominent methodologies for performing Gene Term Enrichment Analysis.

The Segmentation stage prepares the data for use in the Analysis stage by breaking down a source graph into component subgraphs. This can be accomplished via any method that the user desires. As part of this research several methods for decomposing a graph into many subgraphs were developed. They can generally be grouped into two categories: inefficient and efficient methods. These methods for Segmentation are outlined much more in detail in later sections (beginning on pages 20 and 25). In total there are four methods: the Matrix Pattern Analyzer, the Branch Search, the Breadth-First Search, and the Depth-First Search.

Each of these methods have their own positives and negatives. Most often, the inefficient methods are the most optimal in that they find all possible subgraphs based on user defined conditions. The efficient methods trade optimality for much shorter runtime. This significantly lowers the time required to complete the Segmentation process by reducing the number of subgraphs found.

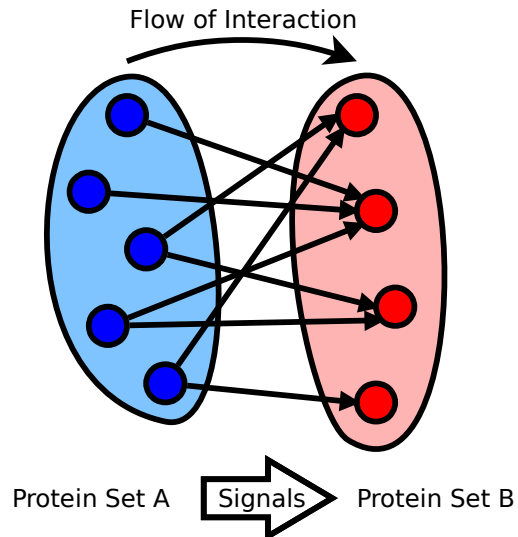
This research focused on two particular subgraph types: Bipartite Subgraphs (BSGs) and Clique Subgraphs (CSGs). Examples of each of these subgraph types can be seen in Figures 2 and 4. Each of the subgraph types were selected due to their unique structure and the meaning such structures could have within a biological context. Figures 3 and 5 frame the particular subgraph type in a biological context.

**FIGURE 2.
BIPARTITE SUBGRAPH**



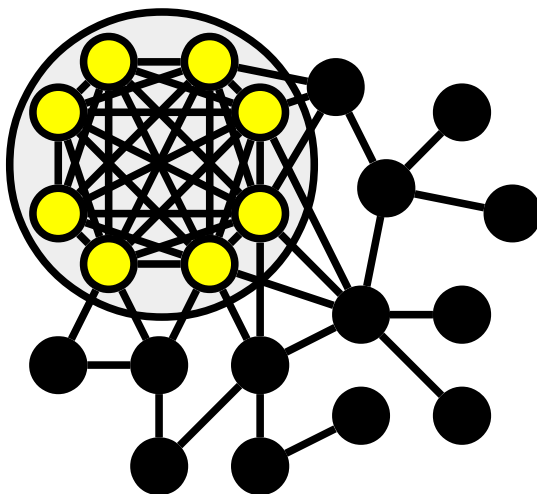
A BSG is highlighted by the gray region. Notice that nodes within a bipartite graph may not be fully connected.

**FIGURE 3.
BIOLOGICAL SIGNIFICANCE
OF BIPARTITE GRAPHS**



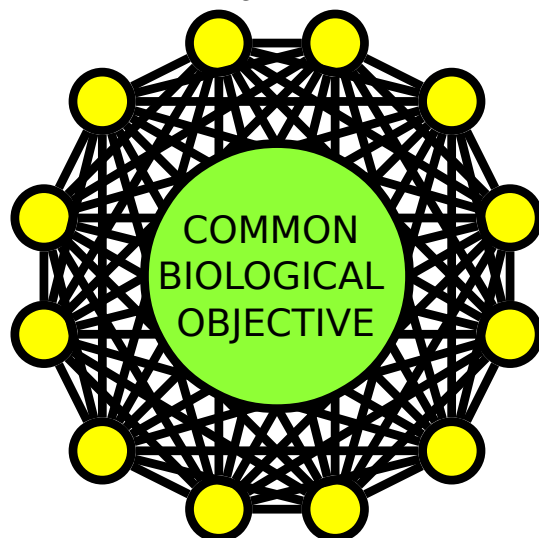
A bipartite graph models a flow of interaction within a biological network.

**FIGURE 4.
CLIQUE SUBGRAPH**



A CSG is highlighted by the gray region. Notice the intrinsic pattern formed by many overlapping edges.

**FIGURE 5.
BIOLOGICAL SIGNIFICANCE
OF CLIQUE GRAPHS**



A clique graph models several proteins working together to accomplish a common biological objective.

Bipartite Subgraphs were selected as a Segmentation subgraph type as they are defined as a relationship between two disjoint sets. In mathematical terms that relationship is defined as follows:

Let U and V be two sets of nodes and E a set of edges:

$$U \cap V = \emptyset$$

$G = \{U \cup V, E\}$ is the graph represented by the union of U and V

$\forall u_1, u_2 \in U$, the edge $(u_1, u_2) \notin E$

$\forall v_1, v_2 \in V$, the edge $(v_1, v_2) \notin E$

This relationship dictates that no two nodes (proteins) may have any edge (interaction) between them if they are from the same disjoint set. This can, without loss of generality, be used to model a flow of interaction from one set of proteins to another. (See Figure 3.) Therefore, by identifying BSGs within a larger network the flow of interaction within the network as a whole can be captured and used to produce more meaningful ontological analysis.

Likewise Clique Subgraphs were selected as a plausible Segmentation subgraph type due to their mathematical definition:

Let N be a set of nodes and E a set of edges:

$G = \{N, E\}$ is the graph composed of the aforementioned sets

$\forall n_1, n_2 \in N$, $\exists$ an edge $(n_1, n_2) \in E$

This relationship dictates that for every two nodes in a graph there **must** be an interaction between them. As such, a CSG can be used to model a highly interactive cluster of proteins within a larger biological network. These proteins might be working together to accomplish some specific biological process or function. The ontological analysis of these particular interactive clusters would no doubt be of biological significance.

Additionally, a limiting value was also selected to be used during the Segmentation process. Since the networks under consideration are scored Protein-Protein Interaction networks, each interaction is assigned an integer value indicating its likelihood. This value was used to give preference to certain interactions over others when performing any of the “efficient searches”. Specifically, interactions with higher likelihood are given

first preference during the segmentation process. More information on this is outlined in the Methods section beginning on page 25.

Finally, Figures 6, 7, and 8 give examples of the Segmentation process in action. Figure 6 contains the Breast Cancer SNB in a simple grid layout. In total, there are 84 proteins and 1900 interactions. Recall that there are 144 total SNBs and each of them have similar numbers of nodes and edges. This demonstrates the size of the data set and the need for proper automation to efficiently process it.

Figure 7 shows a Bipartite Subgraph extracted from the original Breast Cancer SNB. The BSGs disjoint sets are colored blue and red respectively. All proteins not involved in the BSG are colored green.

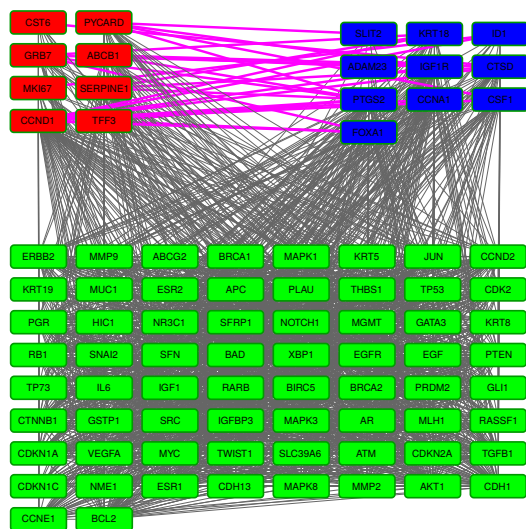
Figure 8 shows a Clique Subgraph extracted from the original Breast Cancer SNB. It is a highly interactive cluster of 21 proteins. Each of the proteins involved in the CSG are colored yellow. All proteins not involved in the CSG are colored green.

FIGURE 6. BREAST CANCER SUBNETWORK BIOMARKER



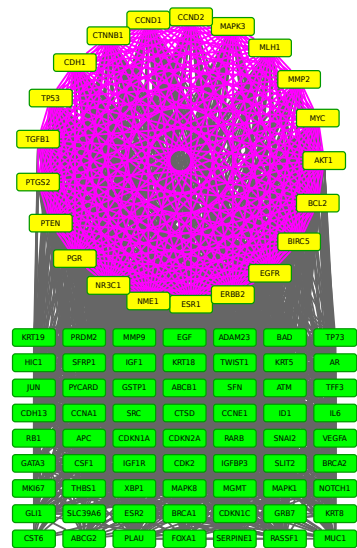
The Breast Cancer Subnetwork Biomarker as a whole. Note how dense the graph is.

FIGURE 7.
BIPARTITE SUBGRAPH IN
BREAST CANCER SNB



A BSG extracted from the Breast Cancer SNB. The disjoint sets are highlighted in blue and red.

FIGURE 8.
CLIQUE SUBGRAPH IN
BREAST CANCER SNB



A CSG extracted from the Breast Cancer SNB. It consists of 21 nodes and 210 edges.

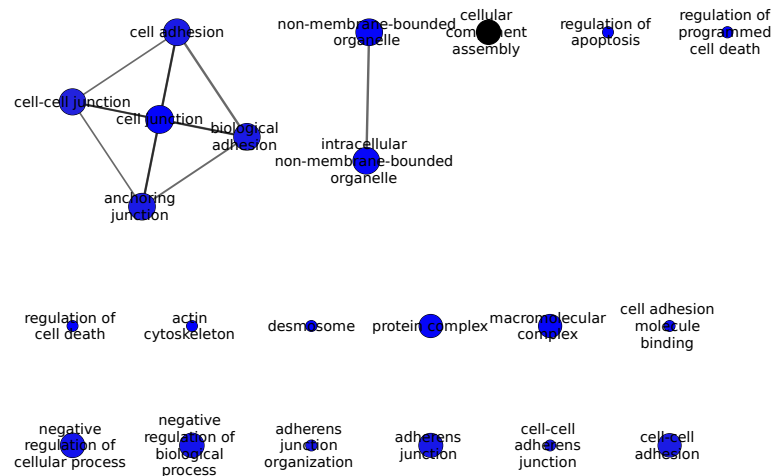
– Analysis –

The primary objective of the Analysis stage is to perform Gene Ontology Enrichment Analysis on each of the individual subgraphs and save these results for use in the Reintegration stage. As such, any of the existing Enrichment Analysis tools are perfect candidates for the Analysis stage. This research has mostly focused on the use of NOA, or Network Ontology Analysis, as developed by Zhang *et al.*^[10] This selection was made because NOA focuses on the network – its nodes and interactions – during the analysis process, and this focus aligns with the goal of this research: to analyze the subgraphs of an SNB for biological significance.

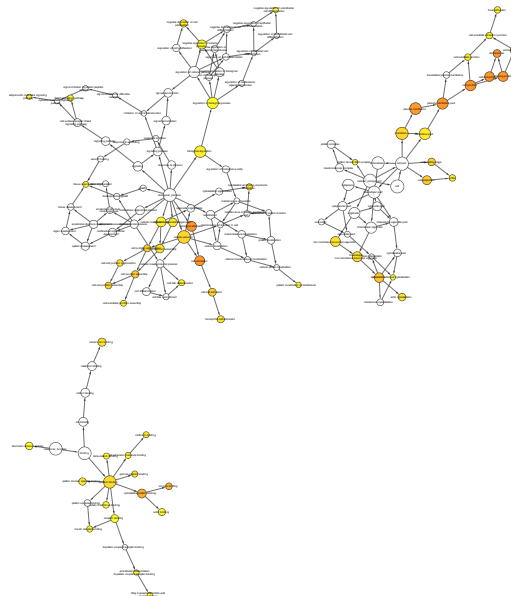
But NOA is not without its downsides. It appears to be out of date and unmaintained. This is due to the fact that it has not yet been updated to Cytoscape 3 despite the latest version of Cytoscape 3 being version 3.3.0 (as of this writing). Utilizing the NOA plugin in a Cytoscape 3 environment would require updating its codebase, and would amount to a large work effort for a small gain. As such, alternatives for the Analysis process have been sought and found in the form of BiNGO.^[9]

BiNGO is a Cytoscape 3 App that performs Gene Term Enrichment Analysis on a list of genes. The focus on Gene List Analysis as opposed to Network Analysis is a minor drawback to be considered, but it is still an effective tool for performing Ontology Enrichment. The fact that BiNGO is readily available for Cytoscape 3 makes it a very attractive alternative for the Analysis stage of the SAR Algorithm.

Before continuing with an explanation of the Reintegration phase it would be appropriate to give some typical results of the Analysis stage. As such, Figures 9 and 10 contain the results of one particular subgraph obtained from the Breast Cancer SNB for both NOA and BiNGO analyses. Additionally, each analysis process produces a table of information of which a small subset is given in Tables 2 and 3.

FIGURE 9. TYPICAL NOA ANALYSIS RESULTS

An overview network of the results in Table 2. Each node represents an Ontological Term while edges represent overlapping genes (from the analyzed network) involved in both GO terms.

FIGURE 10. TYPICAL BINGO ANALYSIS RESULTS

An overview network of the results in Table 3. Each node represents an Ontological component with nodes that are colored in being overrepresented. The more orange a node is the more overrepresented its GO term is.

TABLE 2. TYPICAL NOA ANALYSIS RESULTS

GO-ID	Type	p-val	Test	Reference	Description	Associated Edges
16337	BP	0.008	6/40	21/435	cell-cell adhesion	VEZT (pp)...
22610	BP	0.008	15/40	91/435	biological adhesion	CDSN (pp) DSC2, ...
7155	BP	0.008	15/40	91/435	cell adhesion	CDSN (pp) DSC2, ...
34332	BP	0.011	3/40	6/435	adherens junction	MLLT4 (pp)...
48519	BP	0.014	9/40	45/435	Neg. Reg. of BP	NOTCH2 (pp)...
48523	BP	0.014	9/40	45/435	Neg. Reg. of CP	NOTCH2 (pp)...
10941	BP	0.039	4/40	15/435	regulation of cell death	NOTCH2 (pp)...
43067	BP	0.039	4/40	15/435	regulation of...	NOTCH2 (pp)...
42981	BP	0.039	4/40	15/435	regulation of apoptosis	NOTCH2 (pp)...
22607	BP	0.049	9/40	55/435	cellular component...	MLLT4 (pp)...
5911	CC	0.005	14/40	78/435	cell-cell junction	PERP (pp)...
...	...	...	...	...	...	...

A subset of the NOA output table. This analysis was performed on a BSG of the Adheren's Junction SNB. The p-values have been truncated to fit inside the columns. Additionally, the Associated Edges column is greatly truncated.

TABLE 3. TYPICAL BiNGO ANALYSIS RESULTS

GO-ID	p-val	corr p-val	x	n	X	N	Description	Genes
70161	6E-22	4.6E-19	14	150	30	17789	anchoring junction	TLN1 TLN2 ...
30054	2E-17	8.5E-15	16	521	30	17789	cell junction	TLN1 TLN2 ...
5911	1E-16	2E-14	12	191	30	17789	cell-cell junction	DSG4 TLN1 ...
30057	3E-13	6E-11	6	19	30	17789	desmosome	DSG4 DSG2 ...
44459	1E-11	1E-9	19	1998	30	17789	plasma membrane part	TLN1 TLN2 ...
7155	3E-11	3E-9	13	710	30	17789	cell adhesion	TLN2 ...
22610	3E-11	3E-9	13	711	30	17789	biological adhesion	TLN2 ...
5912	4E-11	3E-9	8	134	30	17789	adherens junction	PVRL4 ...
5886	1E-8	8E-7	21	3730	30	17789	plasma membrane	TLN1 TLN2 ...
8092	9E-8	7E-6	9	507	30	17789	cytoskeletal protein...	TLN1 NME1 ...
17166	3E-7	2E-5	3	9	30	17789	vinculin binding	TLN1 ...
...	...	...	...	...	...	...	...	...

A subset of the BiNGO output table. This analysis was performed on a BSG of the Adheren's Junction SNB. The p-values have been truncated to fit inside the columns. Additionally, the Genes column is greatly truncated.

The primary output of the analyses is of the tabular form given in Tables 2 and 3. Both tables have similar information held within. Perhaps the most important data points are the “GO-ID” and “p-val” columns. The “GO-ID” is the Gene Ontology ID and is used to identify the particular Ontological Component from the Gene Ontology database. Recall that there are three types of GO components: Biological Processes, Cellular Components, and Molecular Functions. The p-value is a number used to indicate how statistically expressed the particular component is. Generally, GO components are considered overexpressed if their p-values are less than a given threshold (usually 0.05).

Differences in the tabular format become apparent when observing the remaining columns. The NOA method operates on a network. As such, the Test and Reference values are related to the number of nodes or edges involved in the expression of the term. The BiNGO analysis is specifically in regards to the number of nodes (as it operates on a gene list). For example, the NOA column “Test” is used in an identical manner to the BiNGO columns “x” and “X” where “x” is the the number of objects related to the particular GO term and “X” is the total number of objects. To elaborate, 14 out of 30 genes are related to GO-ID 70161 (first row of Table 3). In the NOA analysis 6 out of 40 edges are related to GO-ID 16337 (first row of Table 2).

The last columns in both tables are also immediately useful. In Table 2, the “Associated Edges” column gives a listing of all the edges related to a particular GO term. They are generally of the form ``<node> (<interaction type>) <node>`` where `<interaction type>` defines the kind of interaction occurring. Here “pp” indicates a protein-protein interaction. The “Genes” column of the BiNGO analysis lists all the proteins directly involved with the particular GO term. Each gene is separated by a vertical bar. It should be noted that the genes of Table 3 and the Nodes of Table 2 come directly from the nodes within the analyzed BSG.

In addition to these tabular outputs, NOA and BiNGO also create visualizations of the enriched terms. These visualizations can be seen in Figures 9 and 10 and are unique to each analysis methodology.

For more information on each of these analysis methods, see the NOA manual^[13] and the BiNGO manual^[14] for a detailed explanation of the various outputs.

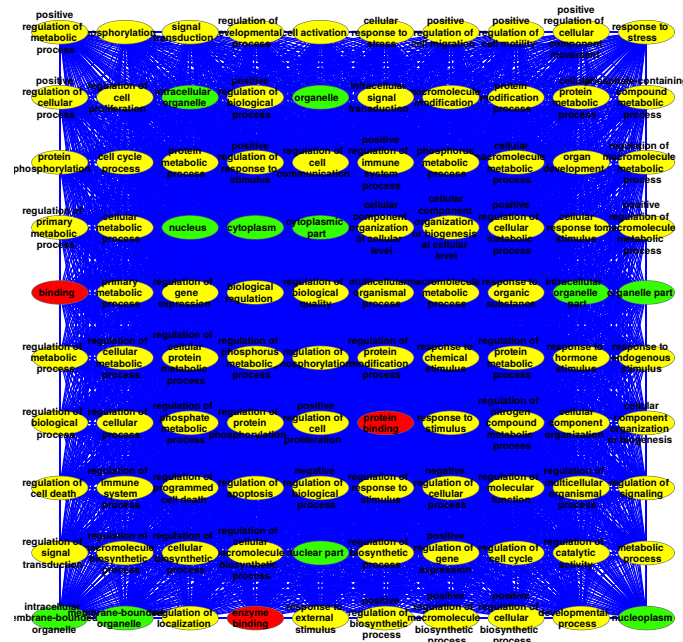
– *Reintegration* –

The Reintegration process is perhaps the simplest of the three stages. In most cases, Reintegration can be easily achieved by performing a Union of all the visualizations produced by the particular Analysis method. This tends to result in a sparser graph of ontological components that is more readily understood by the researcher. It is believed this sparseness is due to the culling of less significant ontological components.

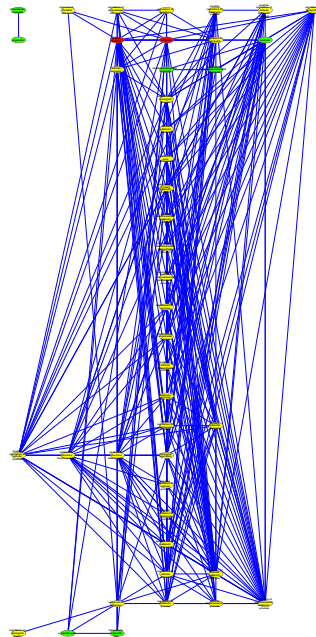
Reintegration, however, is greatly dependent upon the types of results produced by the Analysis. While this research did not significantly utilize the tabular output produced by NOA or BiNGO, other forms of Reintegration may be designed and developed to do so. More discussion on this is held on page 28.

Before moving on to a description of the Segmentation methods developed during this research, a small example and comparison of non-SAR and SAR analysis is given. Figures 11 and 12 contain example outputs for non-SAR based Gene Term Enrichment and SAR based Gene Term Enrichment analysis. In both cases, the NOA method was utilized to perform the analyses. Specifically, SAR was performed (for Figure 12) as follows:

- Segmentation: BSGs were identified using the Breadth First Traversal Search for identifying Subgraphs with a descending Edge Weight Ordering (the PPI Score) and with emphasis on preserving the source graph's structure (Edge Preservation).
- Analysis: Each found BSG was analyzed independently using the NOA plugin for Cytoscape 2.8.
- Reintegration: The produced Overview networks for each independent analysis were joined together with a Cytoscape graph Union and the unlinked ontological components (those without any edges to other components) removed from the overview graph.

FIGURE 11. BREAST CANCER NON-SAR ENRICHMENT ANALYSIS

NOA Gene Term Enrichment Analysis performed on the Breast Cancer SNB. The end result contains some 100 overexpressed GO components with many, many common proteins. The graph is very dense and hard to understand.

FIGURE 12. BREAST CANCER SAR ENRICHMENT ANALYSIS

The (visualized) result of an application of the SAR Algorithm to the Breast Cancer SNB. The end result is much more sparse with half the number of (identified) overexpressed Ontological Components.

Firstly, it should be noted that the colors of the nodes within the two preceding graphs indicate the type of the ontological component: Red nodes are Molecular Functions, Green nodes are Cellular Components, and Yellow nodes are Biological Processes.

When comparing the non-SAR and SAR overview networks it is obvious that the former is particularly congested. This is not necessarily a bad thing as the high number of edges indicates a tightly bound network in which many of the ontological components share genes (proteins) within the source graph. However, the high number of edges makes it difficult to determine which of the edges are interacting with greater frequency; information which can be obtained from the utilization of the PPI scores that accompany the SNB.

The SAR overview network is much more clear. Since each subgraph was segmented with consideration given to the interaction scores of the network and analyzed independently the produced overview graph should be of biological significance. Specifically, it is hypothesized that the overview graph contains the “backbone” ontological components responsible for driving the network and performing the function that the SNB identifies (in this case, the progression of breast cancer).

It should be noted that the particular layout of the SAR overview network was accomplished via the Hierarchal Layout feature provided within Cytoscape 2.8. For more information on hierarchal layouts see Sugiyama *et al.*^[15]

METHODS

INEFFICIENT SEGMENTATION APPROACHES

In total, four methods of segmentation were developed during this research. They are divided into two categories, inefficient and efficient approaches. The methods deemed “inefficient” are declared so due to their extremely high algorithmic runtime complexity. This complexity stems from the fact that all of the inefficient approaches can find all optimal subgraphs. That is to say, an optimal approach is capable of finding unique subgraphs of maximum size. Because very large networks may have many, many subgraphs of a particular type locating them all becomes a combinatorics problem.

Two inefficient segmentation approaches were developed: the Matrix Pattern Analyzer and the Branch Search. Both are described in this section.

– Matrix Pattern Analyzer –

The Matrix Pattern Analyzer was the first Segmentation methodology developed. It is also the only non-search based method developed. (Search based methods are described beginning with the subsection relating to the Branch Search.) The Matrix Pattern Analyzer takes advantage of a common property shared by all graphs of a special form: its Adjacency Matrix often contains a unique pattern.

It should be noted that the Matrix Pattern Analyzer was developed back when the research focused explicitly on Bipartite Subgraphs. As such, it is deprecated in use for the time being until it can be made more general. It is described below in reference to identifying BSGs specifically.

For those not familiar with the structure, an adjacency matrix is an 'n by n' matrix associated with a graph containing 'n' number of nodes. Specifically, each column and row in an adjacency matrix represents a particular node and each cell in the matrix an edge between said nodes. Mathematically speaking, the value at row 'i' of column 'j' of an adjacency matrix in some way denotes the absence or presence of an edge between nodes 'i' and 'j'.

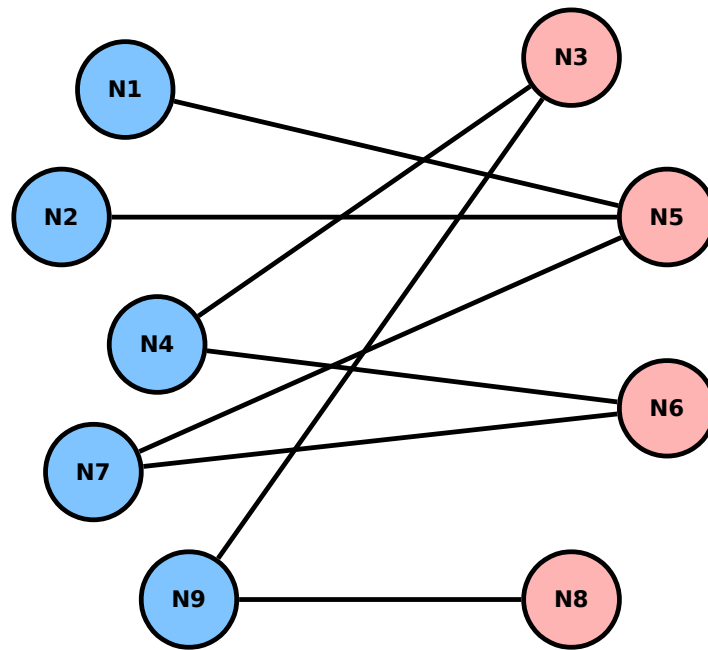
For Bipartite Graphs, the adjacency matrix of such graphs has a very identifying pattern. A small Bipartite Graph is given in Figure 13 and its adjacency matrix in Table 4. The disjoint sets are colored red and blue respectively and the rows of the Adjacency Matrix are colored to match. (The columns could also be colored similarly, but the distinction between rows and columns is not needed unless one is working with directed graphs. As such, only the column header is colored for clarity.)

Notice how when the rows are unordered there is little pattern or sense within the matrix. Now observe Table 5. The rows have been ordered so that the entirety of one disjoint set is listed before the other. A very clear pattern emerges in which two large, square regions of zeros form.

The Matrix Pattern Analyzer exploits this phenomenon. Specifically, it operates in a highly combinatoric manner to search every arrangement of rows in order to locate these particular patterns. Once the pattern is found, the particular disjoint sets (and their edges) are saved as a new subgraph.

This brute force method, however, is extremely costly for large graphs. As the number of nodes within the graph increases, the time it takes to complete the algorithm increases dramatically. Specifically, the complexity class of the Matrix Pattern Analyzer is $\Omega(n!)$. That is, the Matrix Pattern Analyzer has a lower bound of 'n' factorial where 'n' is the number of nodes in the graph. The actual complexity is greater depending upon the specifics of how a particular arrangement of the Adjacency Matrix is determined to contain a BSG. It was estimated, however, that the Matrix Pattern Analyzer has a complexity of $\Theta(n! \cdot (n-1)^2 \cdot n^2)$ in its original implementation.

Due to its incredibly large time complexity, the Matrix Pattern Analyzer is only applicable to small graphs, usually of 10 nodes or less. This limitation quickly ruled it out as an effective method for Segmentation despite the algorithm being an optimal approach. It is simply too inefficient to use on graphs in which the average node count is approximately 80.

FIGURE 13. AN EXAMPLE BIPARTITE GRAPH

A simple Bipartite Graph. Each node is assigned a unique number and colored according to which disjoint set it belongs to.

TABLE 4.
UNORDERED ADJACENCY MATRIX

N	1	2	3	4	5	6	7	8	9
1	0	0	0	0	1	0	0	0	0
2	0	0	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	0	1
4	0	0	1	0	0	1	0	0	0
5	1	1	0	0	0	0	1	0	0
6	0	0	0	1	0	0	1	0	0
7	0	0	0	0	1	1	0	0	0
8	0	0	0	0	0	0	0	0	1
9	0	0	1	0	0	0	0	1	0

The unordered Adjacency Matrix for the above Bipartite Graph.

TABLE 5.
ORDERED ADJACENCY MATRIX

N	1	2	4	7	9	3	5	6	8
1	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	1	0	1	0
7	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	1	0	0	1
3	0	0	1	0	1	0	0	0	0
5	1	1	0	1	0	0	0	0	0
6	0	0	1	1	0	0	0	0	0
8	0	0	0	0	1	0	0	0	0

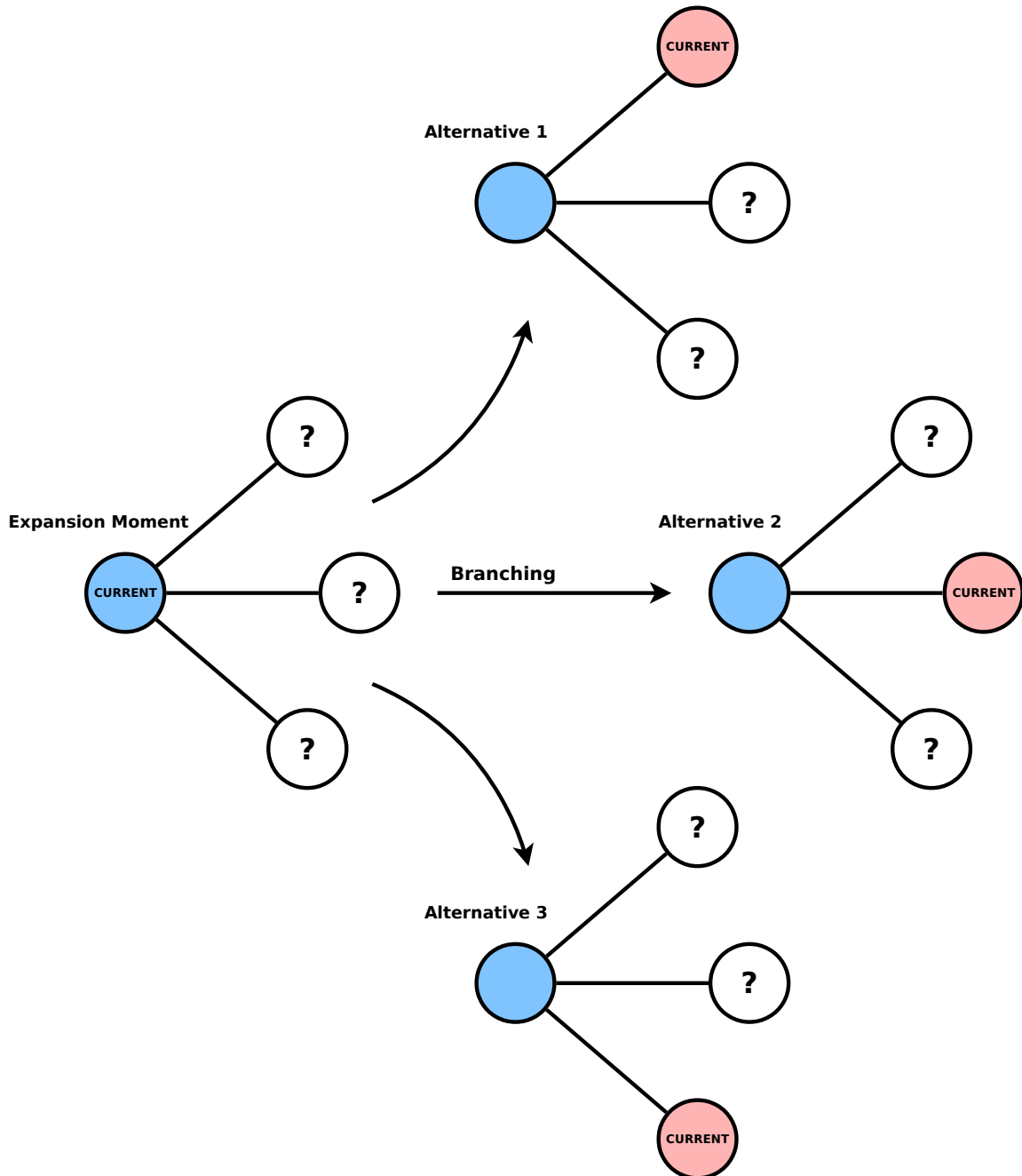
The ordered Adjacency Matrix for the above Bipartite Graph. Notice the squares of zeros in the upper-left and lower-right.

– Branch Search –

The Branch Search was actually the final segmentation method developed by the researcher. It was an attempt to regain the optimal results but with a runtime that is less than factorial in complexity. It should be noted, however, that the Branch Search still contains a runtime that is non-polynomial in nature.

Another interesting factor of the Branch Search is that it's really not an algorithm all to itself. That is to say, its primary feature – Branching – is only an addition to an existing algorithm: the Uniform Cost Search as defined in *Artificial Intelligence: A Modern Approach*.^[16] As such, the Branching feature is applicable to any search-based algorithm. Search-based algorithms operate in a simple manner: they iterate over a source graph and select nodes and edges for addition to a found subgraph. As such, this process is only applicable to finding certain kinds of subgraph types. Luckily, the subgraph types focused on in this research are easily found utilizing search-based methods. (However, Directed Cliques, those cliques which contain directed edges, are not easily found using the search-based methodologies developed in this research. This research, however, utilized undirected networks and this was not a problem.) The exact implementation of search-based methods is discussed in more detail starting on page 29.

Ultimately, the Branch Search is a Uniform Cost Search algorithm with the addition of “Branching”. When a search is “Branching” it doesn't perform a single search. Instead it performs all of them sequentially. Recall that search-based algorithms iterate over a network and identify nodes and edges for expansion. This is usually done by following the edges from node to node within the network. When a Branching search, such as the Branch Search, decides to expand the source graph from a particular node, it tries all the possible expansions independently of each other. This process is illustrated in Figure 14. Notice how for each node the search needs to conduct “b” independent searches where “b” is the branching constant or the average number of edges extending from each node. Using this information the time complexity of Branch Search can be determined to be $O(b^n)$. While not great, this is an improvement over the factorial complexity of the Matrix Pattern Analyzer.

FIGURE 14. BRANCHING PROCESS

The Branching Process applied to a search for Bipartite Subgraphs. Nodes are colored according to which disjoint set they belong to. Notice how during the Expansion Moment there are three possibilities to choose from. A search-based method which is “Branching” will try all three alternatives.

METHODS

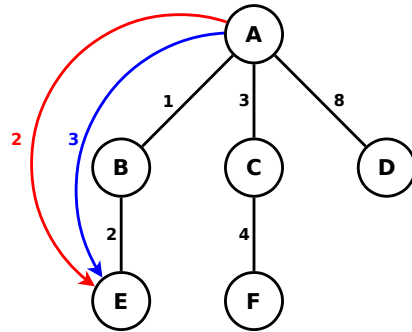
EFFICIENT SEGMENTATION APPROACHES

Two efficient segmentation methods were developed by the researcher: the Breadth-First Search and the Depth-First Search. Both, as their names convey, are search-based approaches and are commonly referred to as Traversal Searches. They operate in a manner similar to the aforementioned Uniform Cost Search in that they iterate over a graph identifying nodes and edges for expansion. How they differ from Uniform Cost Search is the manner in which they iterate. Both Traversal Searches operate in $O(n^2)$ time where “n” is the number of nodes within the source network.

Breadth-First Search iterates by exploring all nodes “nearest” to the start node first. Nodes may be defined to be “near” the start node in many ways. If the edges are not weighted, then two nodes are defined as near if they take fewer steps (that is, edges) to reach each other. This is illustrated in Figure 15 (red). Nodes may also be defined as “near” dependent upon the weights of the edges. Lesser or greater weights may influence how close two nodes are considered to be. This is also illustrated in Figure 15 (blue). The Breadth-First Search process itself is shown in Figure 16. A start node is selected and explored from in order to identify a subgraph. In this case, a BSG is being identified. A gray node does not belong to the subgraph found. The number on the node indicates the order in which it was added to the subgraph.

Depth-First Searches operate in a different manner. Whereas the Breadth-First Search explores the “nearest” node first, the Depth-First Search explores a “chain” of nodes. Example chains for unweighted (red) and weighted (blue) searches are given in Figure 17. The Depth-First Search process is shown in Figure 18. A start node is selected and explored from just like in the Breadth-First Search. This is actually a characteristic of all Traversal Searches. This figure also identifies a BSG, operating on the **same** graph as the example in Figure 16. This demonstrates how different methods operating on the same graph may identify different subgraphs.

FIGURE 15.
BFS – “NEARNESS” EXAMPLE

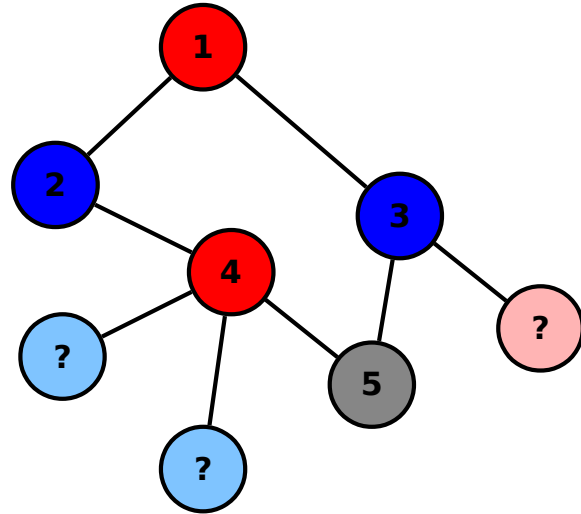


Unweighted:
 $A \rightarrow B = A \rightarrow C = A \rightarrow D = 1$
 $A \rightarrow E = A \rightarrow F = 2$

Weighted:
 $A \rightarrow B = 1$
 $A \rightarrow E = A \rightarrow C = 3$
 $A \rightarrow F = 7$
 $A \rightarrow D = 8$

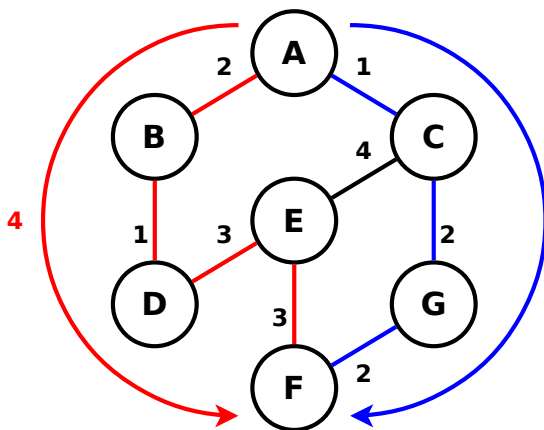
Demonstration of how nodes may be considered “near”. Distances for all nodes are given relative to node A.

FIGURE 16.
BFS – PROCESS



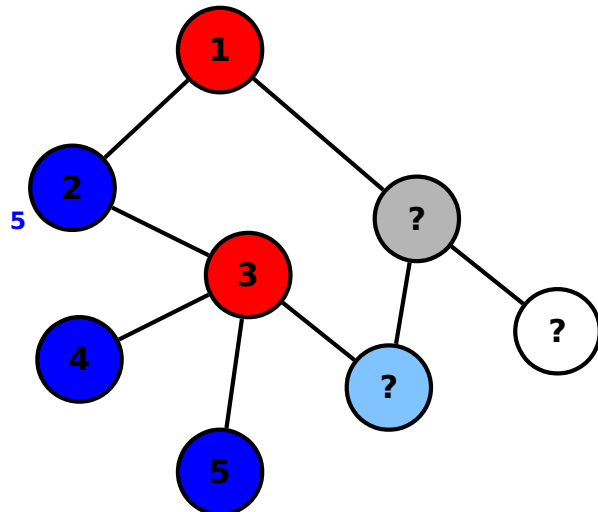
The BFS Process in action. Darkly colored nodes are already explored, lightly colored nodes still need to be explored.

FIGURE 17.
DFS – “CHAINS” EXAMPLE



This figure demonstrates how different exploration chains may occur when a DFS is unweighted or weighted.

FIGURE 18.
DFS – PROCESS



The DFS Process in action. Like above, darkly colored nodes are explored, and lightly colored nodes will be explored.

The Traversal Searches developed by the researcher, however, are not without modifications. While their designs are based on the definition of Breadth-First and Depth-First Traversal Searches given in *An Introduction to The Design and Analysis of Algorithms*,^[17] they do not operate exactly like the originals. Notably, the Breadth-First Search can be coerced to behave like the Uniform Cost Search by configuring it to operate on an ordered list of edge weights. Likewise, part of this characteristic can imbued into the Depth-First Search as well.

Further, there are other characteristics built into the algorithms by the researcher that affect their original performance. Notably, the concept of Edge Preservation and the previously mentioned limiting value. The discussion on Edge Preservation is belayed until the section beginning on page 29 in which the Automation of the Algorithm is discussed. For now, know that if a method is configured to be “Edge Preserving” then this influences how many network components (nodes and edges) are added at once and whether an expansion is considered valid.

The limiting value, however, has already been demonstrated: the weight of an edge may influence the order in which nodes are selected for expansion. Recall that for every PPI within one of the Subnetwork Biomarkers there exists a score which rates the likelihood of an interaction. The Traversal Searches developed by the researcher can put emphasis on this score and utilize it when making a decision on how to search for subgraphs. Specifically, since higher scores are preferred, the two Traversal Searches can be configured to give preference to nodes that are linked to the source graph with higher edge weights (that is, higher PPI scores are explored first).

To conclude the discussion of methods it should be noted that the inefficient methods have been considered deprecated and were not focused on heavily during the testing and automation phases of this research. In the future, however, they may be returned to for continued analysis, but the researcher has chosen to focus on the more efficient Traversal Searches for the time being.

DISCUSSION

Overall, the first half of this research as presented so far was dedicated to identifying particular subgraph types within a larger graph. This led to the development of several methodologies for segmenting the graph and ultimately the hypothesis that the overview network produced by the Reintegration method contained GO components that are of particular interest to the researcher.

Primarily, the focus has been on visualizing the interactions as opposed to any particular analysis of the overexpressed gene products. This led to the previously mentioned utilization of the overview networks produced by NOA and BiNGO and has left another avenue for research into Reintegration methodologies that utilize the tabular output as well.

Once the SAR algorithm was proposed, a way to perform analysis more efficiently was required. Various avenues of progression were considered. In the end, the creation of an effective software suite to perform Segmentation was decided upon. This led to the development of the Subgraph Finder utility and the CyFinder Cytoscape 3 App. Both of these software components are thoroughly discussed in the next section, and their development consumed the majority of the effort applied to the project.

DISCUSSION

AUTOMATING THE ALGORITHM

Manually processing the numerous amounts of data available to the researcher would be an infeasible task. As such, the researcher elected to develop a software tool to apply the SAR algorithm to a selected network. Given that Analysis is easily achieved using existing tools, software components were only needed to perform the Segmentation and Reintegration stages. As of this writing, the Segmentation component is at a stage of development acceptable for release.

Ultimately, two software components were developed: the Subgraph Finder Utility, and the CyFinder Cytoscape App. The Subgraph Finder Utility is a stand-alone program developed to perform Segmentation as defined in the Methods section of this research. The CyFinder App bundles the Subgraph Finder Utility and makes it available to users of Cytoscape 3. Both software systems are described below.

– The Subgraph Finder Utility –

The Subgraph Finder Utility is a simple but extensible software tool used for identifying subgraphs of various kinds within a larger graph. It is capable of logging its processes as well as processing numerous graphs in a batch mode. The Utility, as a stand-alone program, is only operable from the terminal prompt.

Notably, the Subgraph Finder Utility is divided into two important components: Graph objects and Algorithm objects. The Graph objects represent networks in memory via a list of nodes and edges. This allows for standard operations such as retrieving an adjacency list to be completed easily. Additionally, there are two types of graph objects. Normal graph objects, and Conditioned Graphs. Conditioned Graphs restrict the expansion of a graph based upon its attached Conditions.

Condition objects represent a point of extensibility for the Utility. They are directly utilized by the software when determining if a subgraph is one which it is looking for. As such, to search for any subgraph type one merely needs to select appropriate Condition

objects for use or to develop new Condition classes. This, however, is not as simple as it sounds due to the operation of the second component: the Algorithm objects.

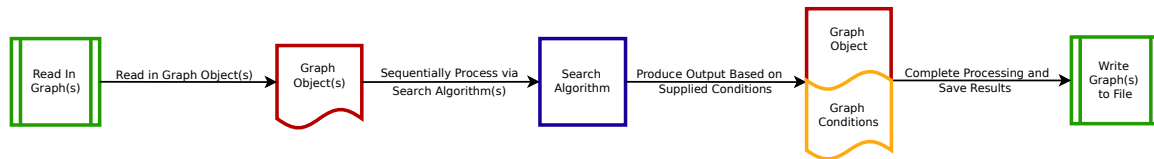
Algorithm objects are used for finding subgraphs. Specifically, they implement the various Segmentation methods discussed earlier. The base Algorithm is abstract and defines an interface for inheritance. Its inputs consist of a single Graph object to process, and its output is a list of Graph objects identified. This allows for easy extension of the Utility by other developers. Because each Algorithm object may find graphs in a different manner, it is not guaranteed that all subgraphs found will be maximal in size nor that it will find all possible subgraphs. In other words, not all Algorithms are optimal.

The general process by which the Subgraph Finder Utility operates is diagrammed in Figure 19. The process is an iterative one in which there are three main operations. The first and last operations are the input and output operations respectively. The second object in the diagram is the source graph, and the fourth object is the list of graphs found by the Utility.

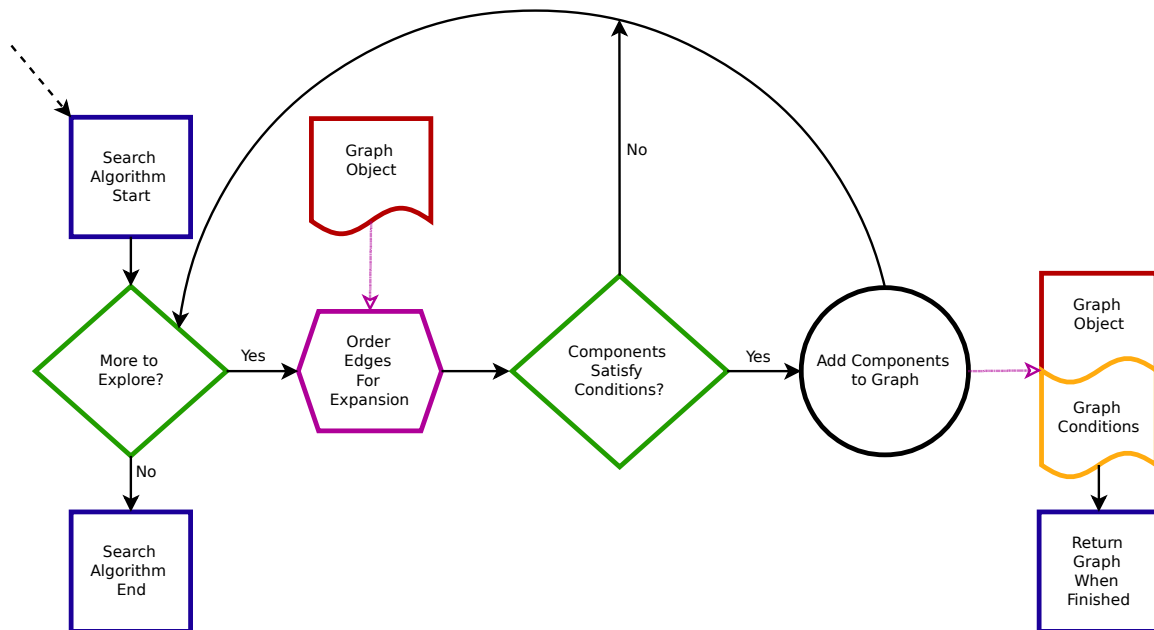
The third component of the diagram is the most interesting one. It represents the algorithm selected by the user for finding the subgraphs. As such, its operations and processes constitute a description and diagram of their own. As of the moment only a single type of Algorithm has been developed: the aforementioned constructive or “search-based” algorithms.

Constructive algorithms build up the subgraph from nothing as they progress. They start with an empty Graph object (with Conditions applied) and construct a graph by inserting a few nodes and edges at a time into the Conditioned Graph. The number of edges and nodes added as the algorithm progresses is dependent upon the algorithm itself. Recall that there are two constructive algorithms that have been developed: Breadth-First and Depth-First traversal searches.

The process of a constructive algorithm can be seen in Figure 20.

FIGURE 19. SUBGRAPH FINDER GENERAL PROCESS

The Subgraph Finder General Process. It has two input and output stages (green). Graphs are outlined in red while the blue square in the center represents the algorithm selected for identifying subgraphs.

FIGURE 20. CONSTRUCTIVE (SEARCH-BASED) ALGORITHM PROCESS

The Constructive Algorithm Process. It is moderately complex containing many branch points and a loop.

Constructive algorithms are far more complex than the general process as a whole. They contain a loop dependent upon the state of the search. As long as there remain unexplored nodes in the source graph the search will continue. This can be seen in the “More to explore?” decision diamond in the figure. If the algorithm decides to proceed, then it orders the nodes adjacent to the current node for expansion. If the user declines to configure an ordering the static ordering is used. One possible ordering, as previously discussed, is an edge weight based ordering in which target nodes are ordered for expansion based upon the largest (or smallest) edge weight leading to the node. The next step of the process is to determine if the network components selected for addition to the subgraph comply with the conditions attached to the subgraph object. This is a relatively complex task based upon the complexity of the Conditions attached. Additionally, the nature of constructive search adds additional complexity. Primarily, the user needs to decide how to handle the back-edges in the DFS algorithm and the cross-edges in the BFS algorithm.

The solution developed by the researcher is given the moniker “Edge Preservation”. If an algorithm is configured to be Edge Preservative then the algorithm ensures the following expression is true for any source graph, $G=\{V,E\}$, its subgraph, $G'=\{V',E'\}$, and the edge (u,v) :

$$\forall u,v \in V', (u,v) \in E \rightarrow (u,v) \in E'$$

The above states that any subgraph produced from a source graph by an algorithm that is Edge Preservative honors all the underlying edges within the source graph; that is to say, any subgraph produced this way will be an accurate reflection of all the interactions between its nodes in the source graph. This is an important distinction to make compared to the alternative where this property doesn't necessarily hold true. Ultimately, this property governs how a search-based expansion may occur.

If this property is not held true, then there are many ways in which an expansion may occur. The algorithms developed take the following approach. First, the explored node and edge linking it back into the graph are added provided the Conditions on the

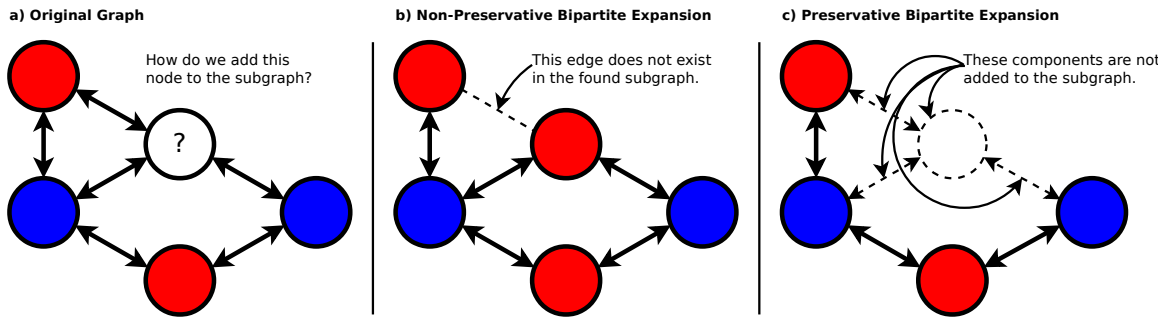
Conditioned object hold true after the expansion. Then, when exploring later from the newly added node any edges linking back into the subgraph are considered for inclusion in the resultant graph.

If the Edge Preservative property is strictly enforced for a particular run of the algorithm the expansion operates quite differently. Specifically, the adjacent node being selected for addition to the subgraph and all edges linking it back to the subgraph as it is defined so far are simultaneously added at once. This ensures the Edge Preservation property is true for any subgraph produced.

An example of the differences caused by the Edge Preservation property can be seen in Figure 21. The graph in section (a) represents a possible expansion that could occur. Nodes are being added to the subgraph with the leftmost blue node as the one from which it is currently exploring. The algorithm then needs to decide how to add the white node the graph while preserving the bipartite condition on the subgraph.

If the algorithm is not configured to ensure the Edge Preservation property is maintained, then it will expand and add the white node to the graph as part of the red grouping. It will then disregard the edge connecting the newly added red node to an existing red node. This is demonstrated in section (b). In some cases this may be the desired functionality of the expansion algorithm. The alternative, however, can be seen in section (c).

If an algorithm preserves edges then it will attempt to add the node for expansion and all the edges linking back into the subgraph at the same time. It is unable to do so because attempting to add those components would violate one of the conditions on the subgraph (the bipartite condition), and it rejects all the network components under consideration and moves one.

FIGURE 21. EDGE PRESERVATION DEMONSTRATION

A demonstration of how the Edge Preservation property can affect a subgraph's identification. Section (a) contains the portion of the graph under consideration. Section (b) contains a non-Preservative expansion. Section (c) contains a Preservative expansion.

– *The CyFinder App* –

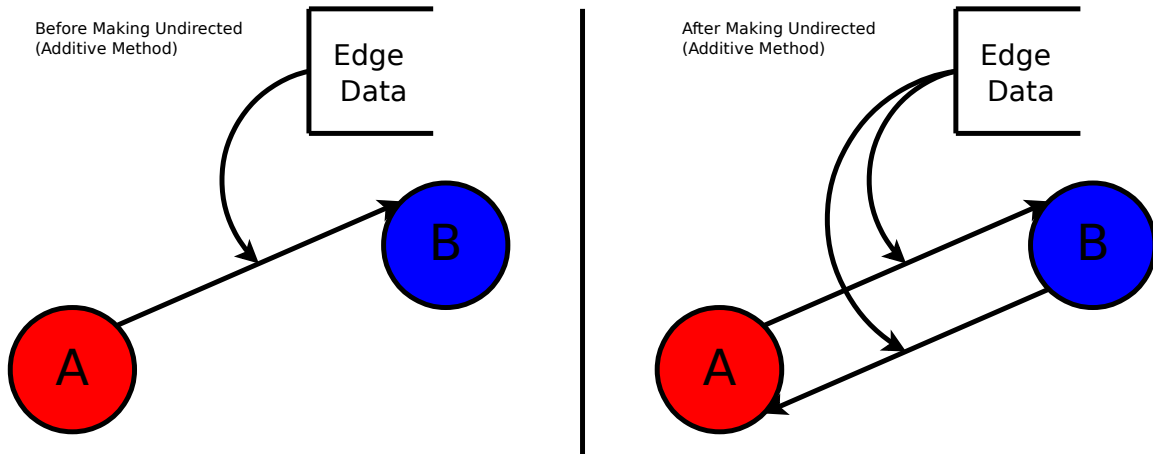
The CyFinder App is an adaptation of the previously described Subgraph Finder Utility for the Cytoscape 3.0 software environment. It wraps the independent Java code in another layer to provide its functionality to users of Cytoscape. This allows users to search through complex network data such as protein interaction networks for subgraphs of specific type and form within a familiar research environment.

Under the OSGI model now employed by Cytoscape for managing Apps, each App is composed of tasks that are executed by Cytoscape when it receives instruction to do so from a user action or the execution of another tasks. The CyFinder App currently has two directly accessible tasks: “Make Undirected” and “Subgraph Finder”. A third task, the “Configuration” Task, is only accessible programmatically. All three tasks require that the user currently have a network selected when executed.

The “Make Undirected” Task is a simple task that coerces the selected network into an undirected state. This is useful given the lack of support by Cytoscape for importing undirected networks. Making a network undirected, however, can be accomplished in many different ways. As such, two variants for making a network undirected have been developed: the Additive Method and the Transformative Method.

The Additive Method works by leaving the network, technically, in a directed state, but adding additional edges to the network so that for every edge from node A to node B there exists an edge from node B to node A. The algorithm merely copies the attributes of the original edge to the new edge and reverses the source-target relationship. This is demonstrated in Figure 22.

The Transformative Method, however, alters existing edges by replacing them with exact duplicates except they have the internal Cytoscape flag for directed-ness of edges set to false. Cytoscape does not offer a way to alter an edge's directed-ness value so the algorithm copies the contents of the original edge entirely and recreates it but in such a manner that the “isDirected()” method on the edge returns false. This allows Apps (including the CyFinder App) to respond to undirected edges in a meaningful manner.

FIGURE 22. MAKE UNDIRECTED – ADDITIVE DEMONSTRATION

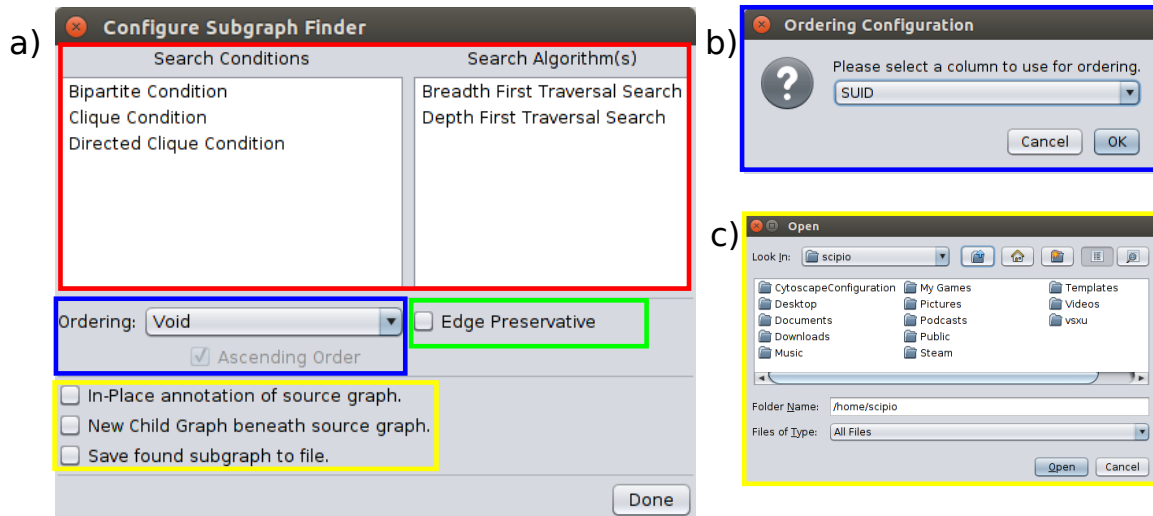
The process by which the Additive Method for making edges undirected functions. Here, the original, directed edge is copied and the source and target of the original edge swap places. This results in two edges existing for each original edge.

The “Configuration” Task was the most complex Task to develop for the CyFinder App due to being the only one for which a large amount of new code would be generated. Specifically, a UI window was developed for collecting search information from the user. Figure 23 contains a screenshot of the “Configuration” Task UI.

The screenshot is divided into three sections. The main window in section (a) is where the majority of the configuration occurs. Here the user can select which Conditions to search for and what Algorithms to use (red box), the ordering of the edges should they be ordered (blue box), how back-edges are handled (green box), and what methods to use for saving the found results (yellow box). Note that there are two Clique Conditions. This is due to the operation of search algorithms and the manner in which they expand to identify a subgraph. At the moment the Directed Clique Condition allows for cliques to be found in directed networks.

Section (b) contains a configuration dialog for selecting which Cytoscape Column to use for ordering the edges when performing a search. The dialog only displays columns that are numeric in nature.

Lastly, section (c) contains a dialog that allows the user to select where to save found subgraphs. This dialog is only displayed if the user selects the third save option presented in section (a). The found subgraphs are saved utilizing the Subgraph Finder Utility itself and not Cytoscape. If the user wishes to save found subgraphs using Cytoscape then they should select one of the two other saving methodologies and export the graph objects from within Cytoscape.

FIGURE 23. CONFIGURATIN TASK SCREENSHOT

A screenshot of the various Configuration Task UI Dialogs. It is divided into one major component and two minor dialogs (left and right respectively).

The “Subgraph Finder” Task was simple to implement and operates in three steps: Graph Conversion, Graph Processing, and Graph Saving. The first step converts the Cytoscape network to a Graph object as defined by the Subgraph Finder Utility. This usually consists of recreating the node and edge relationships and attaching any ordering data to the edges (edge weight for example).

The Graph Processing portion is the simplest of the three in terms of reused code. It utilizes the Subgraph Finder Utility directly to process the Graph based on the Configuration Bundle as defined by the user. It directly calls the Algorithm object's “process” method to operate over the graph and produce subgraphs which are then used in the next and final step.

The saving of graphs is dependent entirely upon which operations the user selected for saving graphs. The first option, “In-Place”, saves the subgraphs as Cytoscape Columns on the original source graph. For each found subgraph, a new column is made on the Node and Edge tables of the network. If the node or edge is included in the subgraph it is marked true, else it is marked false.

If the “New Child” option is selected then the App will create a new Cytoscape network beneath the original for each found subgraph. This is, perhaps, the most useful option for saving subgraphs as it loads them into the current Cytoscape session as their own network objects. This then allows them to be analyzed using Cytoscape or any Cytoscape App the user may have installed.

Finally, there's the “To File” option. This option saves the found subgraphs to a directory of the user's choice. The subgraphs are saved using the Subgraph Finder Utility, and are tab-delimited. Additionally, they only contain attribute information used for ordering the edges in memory. If a more robust copy of the subgraph is desired by the user they should, as previously mentioned, use a different saving methodology and export the graph from within Cytoscape.

CONCLUSION AND FUTURE WORKS

In conclusion, the SAR algorithm was designed and hypothesized to be useful in the analysis of various biological networks. Additionally, the Segmentation aspect of the SAR algorithm was thoroughly developed, and a software tool for performing segmentation was designed, developed, and embedded within the Cytoscape App environment for ease of use.

Future works remaining to be completed include further modification and optimization of the Subgraph Finder Utility and the CyFinder App. Additionally, the SAR Algorithm itself needs to be automated thoroughly so large amounts of data may be processed in short periods of time. Finally, the results produced by processing the 144 Subnetwork Biomarkers that consist of this research's dataset need to be biologically validated and a decision as to their usefulness and significance made.

REFERENCES

1. Tisdall, James D. *Beginning Perl for Bioinformatics*. Beijing: O'Reilly Media, 2011. Print.
2. Luscombe, N. M., D. Greenbaum, and M. Gerstein. "What Is Bioinformatics? A Proposed Definition and Overview of the Field." *Methods Archive* 40.4 (2001): 346-58. Web. 29 Feb. 2016.
3. "GO Enrichment Analysis." *Gene Ontology Consortium*. The Gene Ontology. Web. 1 Mar. 2016. <<http://geneontology.org/page/go-enrichment-analysis>>.
4. "Ontology Documentation." *Gene Ontology Consortium*. The Gene Ontology. Web. 1 Mar. 2016. <<http://geneontology.org/page/ontology-documentation>>.
5. Rivals, I., L. Personnaz, L. Taing, and M.-C. Potier. "Enrichment or Depletion of a GO Category within a Class of Genes: Which Test?" *Bioinformatics* 23.4 (2006): 401-07. Web.
6. Reiner, A., D. Yekutieli, and Y. Benjamini. "Identifying Differentially Expressed Genes Using False Discovery Rate Controlling Procedures." *Bioinformatics* 19.3 (2003): 368-75. Web.
7. Carbon, S., A. Ireland, C. J. Mungall, S. Shu, B. Marshall, and S. Lewis. "AmiGO: Online Access to Ontology and Annotation Data." *Bioinformatics* 25.2 (2008): 288-89. Web.
8. Eden, Eran, Roy Navon, Israel Steinfeld, Doron Lipson, and Zohar Yakhini. "GORilla: A Tool for Discovery and Visualization of Enriched GO Terms in Ranked Gene Lists." *BMC Bioinformatics* 10.1 (2009): 48. Web.
9. Maere, S., K. Heymans, and M. Kuiper. "BiNGO: A Cytoscape Plugin to Assess Overrepresentation of Gene Ontology Categories in Biological Networks." *Bioinformatics* 21.16 (2005): 3448-449. Web.
10. Zhang, C., J. Wang, K. Hanspers, D. Xu, L. Chen, and A. R. Pico. "NOA: A Cytoscape Plugin for Network Ontology Analysis." *Bioinformatics* 29.16 (2013): 2066-067. Web.
11. Shannon, P. "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks." *Genome Research* 13.11 (2003): 2498-504. Web.
12. Timalisina, P., Charles, K., and Mondal, A. M., "STRING PPI Score to Characterize Protein Subnetwork Biomarkers for Human Diseases and Pathways," *Proceedings, 14th IEEE International Conference on Bioinformatics and Bioengineering, BIBE-2014*, p. 251-156 (2014).
13. "NOA User Manual." *NOA User Manual*. National Resource for Network Biology. Web. 11 Mar. 2016. <<http://nrnb.org/tools/noa/>>.
14. "BiNGO." *BiNGO*. Web. 11 Mar. 2016. <<http://www.psb.ugent.be/cbd/papers/BiNGO/Home.html>>.
15. Sugiyama, Kozo, Shojiro Tagawa, and Mitsuhiro Toda. "Methods for Visual Understanding of Hierarchical System Structures." *IEEE Transactions on Systems, Man, and Cybernetics IEEE Trans. Syst., Man, Cybern.* 11.2 (1981): 109-25. Web.
16. Russell, Stuart J., Peter Norvig, and Ernest Davis. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2010. Print.
17. Levitin, Anany. *Introduction to the Design & Analysis of Algorithms*. Boston: Pearson, 2012. Print.