CyFinder Tutorial

Raihanul Bari Tanvir * rtanv003@fiu.edu

Cesar Martin * cmart504@fiu.edu

Ananda Mohan Mondal amondal@fiu.edu

* Equal Contribution

TABLE OF CONTENTS

INTRODUCTION

Diseases such as cancers can be represented as biological networks. These networks can be further represented by a mathematical object called a Graph. The vertices or nodes of these graphs are the genes or proteins of the network. The Edges of the graph are the interactions between the proteins or genes and their numeric weight values can represent different attributes of the interactions.

Cytoscape is an open-source software platform that aids in the visualization and analysis of molecular Networks and biological Pathways, as well as gene expression. Cytoscape offers a basic set of Core Applications to perform these tasks but also provides an option to add user-made plugins in the form of Java Archive (JAR) Maven Install Files. These plugins can be similar to the core apps and can help network visualization, generation, and analysis. Our project was developed for Cytoscape 3.8.2 [1], [2].

CyFinder is a Cytoscape application that aids in the finding and visualization of subgraphs within a selected network. CyFinder accomplishes its tasks by wrapping a simple API of Graphs and Graph algorithms with Cytoscape tasks that convert Cytoscape Networks to the Graphs in the API and execute the algorithms. Since the standard graph file format when using Cytoscape assumes undirected edges, CyFinder's first functionality is to make the selected Network undirected by either making all the edges undirected or adding directed edges in reverse for all directed edges.

The Networks can then be analyzed to find subnetworks. Features include finding the Network's Connected Components, Maximal Cliques, Maximal Bicliques, Extremum Spanning Forests, and Community Structure with various algorithms. The purpose of these subnetworks is the identification of biomarkers in disease networks; the presence of these biomarkers is expected to signal that the disease or a similar disease is present. In addition to finding subnetworks from a single network, there are features to find common subnetworks in two networks, which helps further identify biomarkers across different disease networks. The found subnetworks can be saved and ordered in several ways, such as to a file, as a new Cytoscape Network, or as in-place annotations in the source Network.

To aid with Bipartite Network Visualization, CyFinder also includes a Bipartite Layout option that arranges the Networks Nodes into two columns that correspond to the Graph's two partite sets.

How To Install CyFinder

CyFinder can be installed from the Cytoscape application manager. The recommended version of Cytoscape is 3.8.2.

Steps:

1. Go to the application manager.

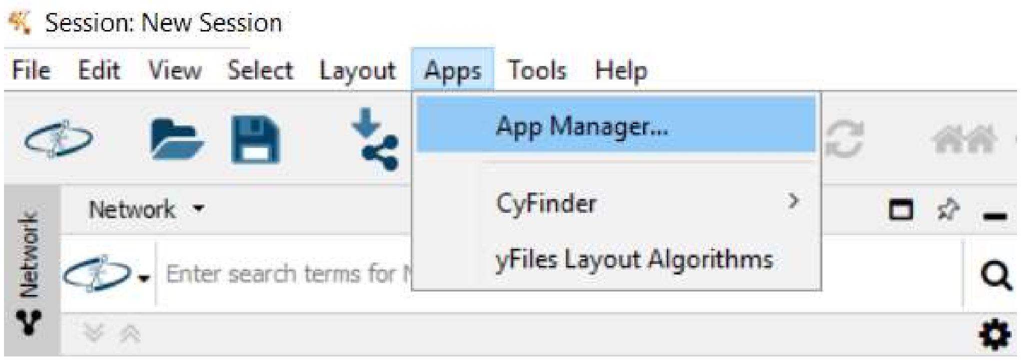


Figure 1: App Manager in the menu.

2. Search for CyFinder and click the install button.

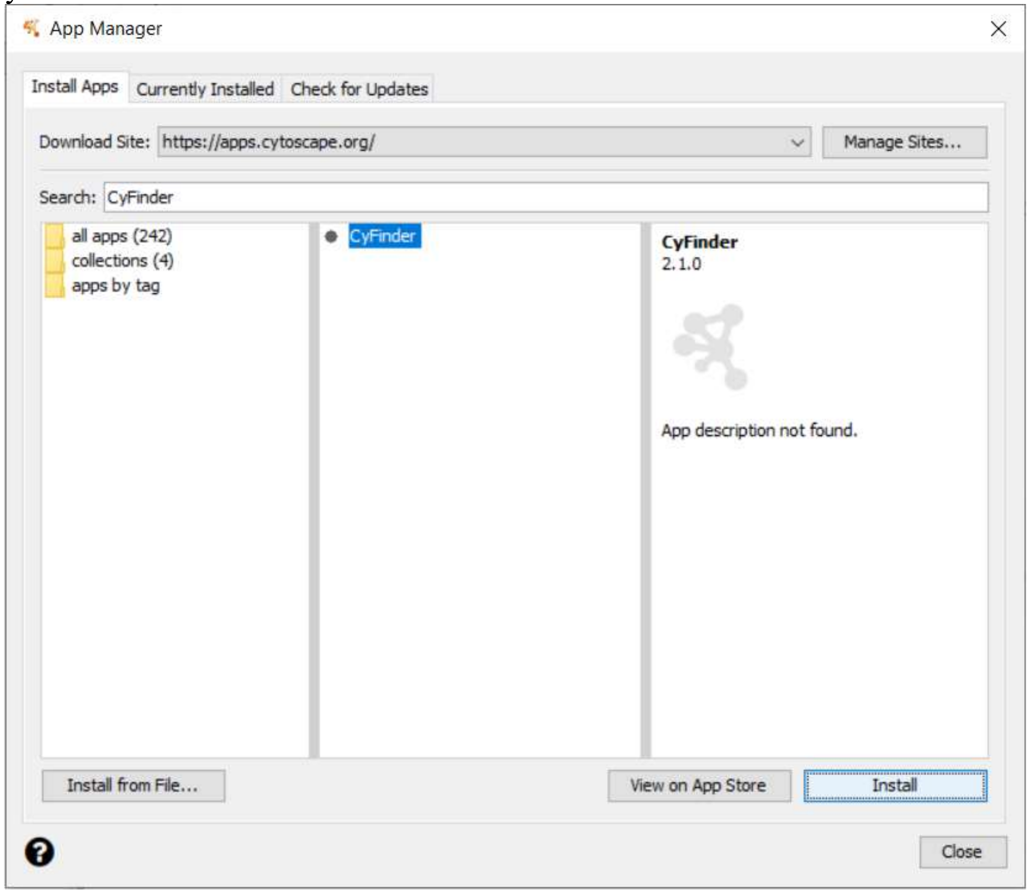


Figure 2: Installing CyFinder from the application manager.

To use CyFinder, first import a Network to Cytoscape. If algorithms that use weights will be used, make sure the network file has at least one type of numeric edge attribute to use as edge weight.

FEATURE 1 - MAKING NETWORK UNDIRECTED

When a Network is imported into Cytoscape, the edges are directed by default. Many of our algorithms work with undirected graphs by default so we need to convert the directed edges first. CyFinder offers two methods to accomplish this: the Additive Method and the Transformative method. The additive method adds a directed edge in the opposite direction for every directed edge originally in the graph. The transformative method converts the original directed edges into undirected. Both options are applied to all selected networks.

To use the additive method: **Apps -> CyFinder -> Make Undirected -> Additive Method**
To use the transformative method: **Apps -> CyFinder -> Make Undirected -> Transformative Method**

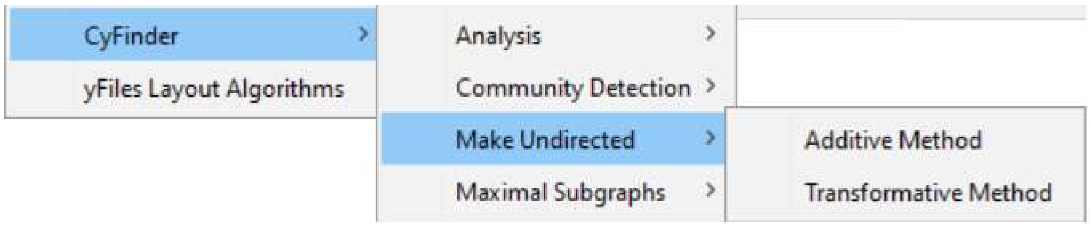
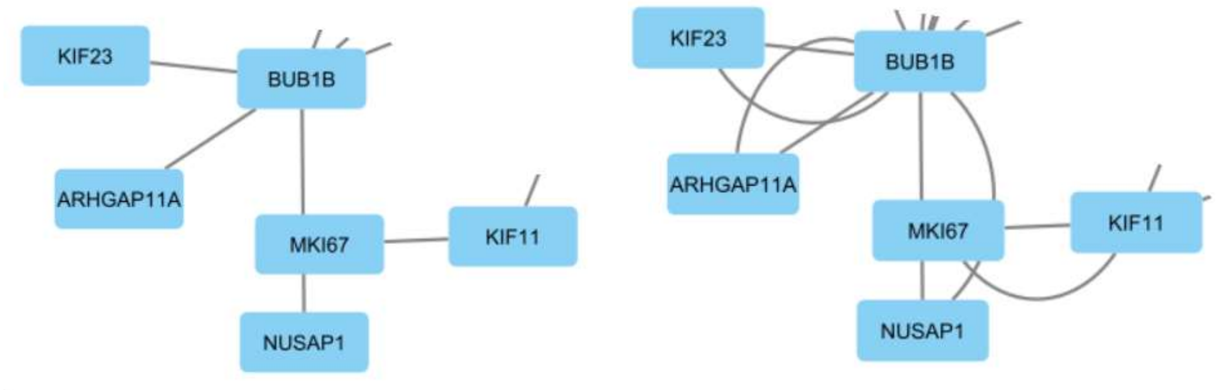


Figure 3: Make Undirected Options.



input output

Figure 4: Input and output of Additive Method.

There is no visual change with the transformative method, so it is not shown.

We will use the transformative method for all our demonstrations since it is more intuitive to work with undirected graphs.

Feature 2 - Subgraph Finder

The Subgraph Finder features allows us to find cliques and bipartite subgraphs. A clique is a graph in which all nodes are connected to all other nodes. A bipartite graph is a graph with two partite sets in which no nodes in the same partite set are connected.

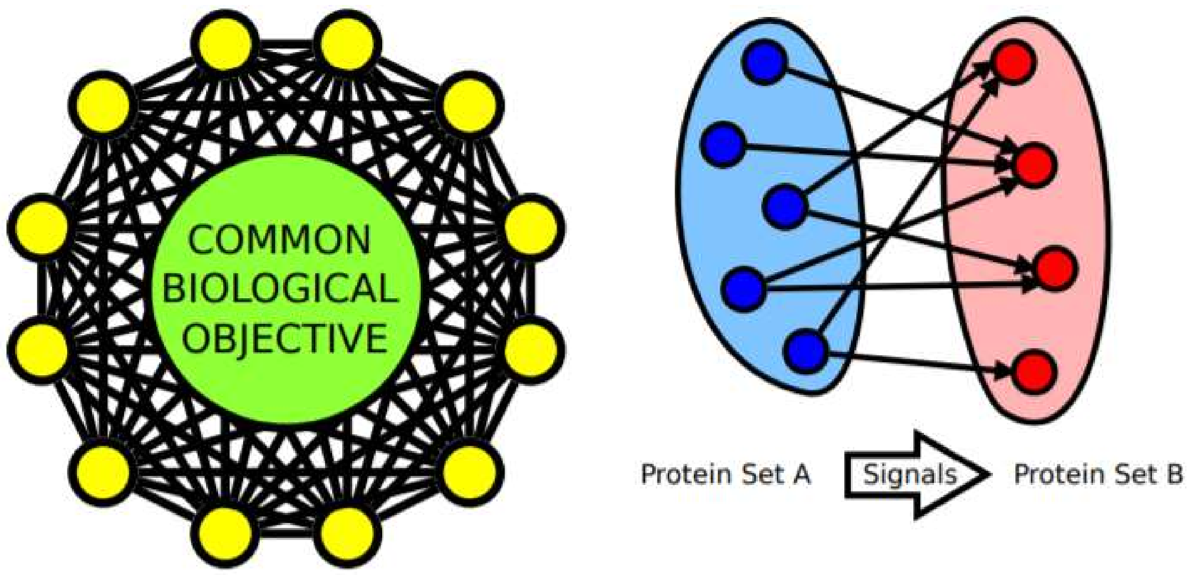
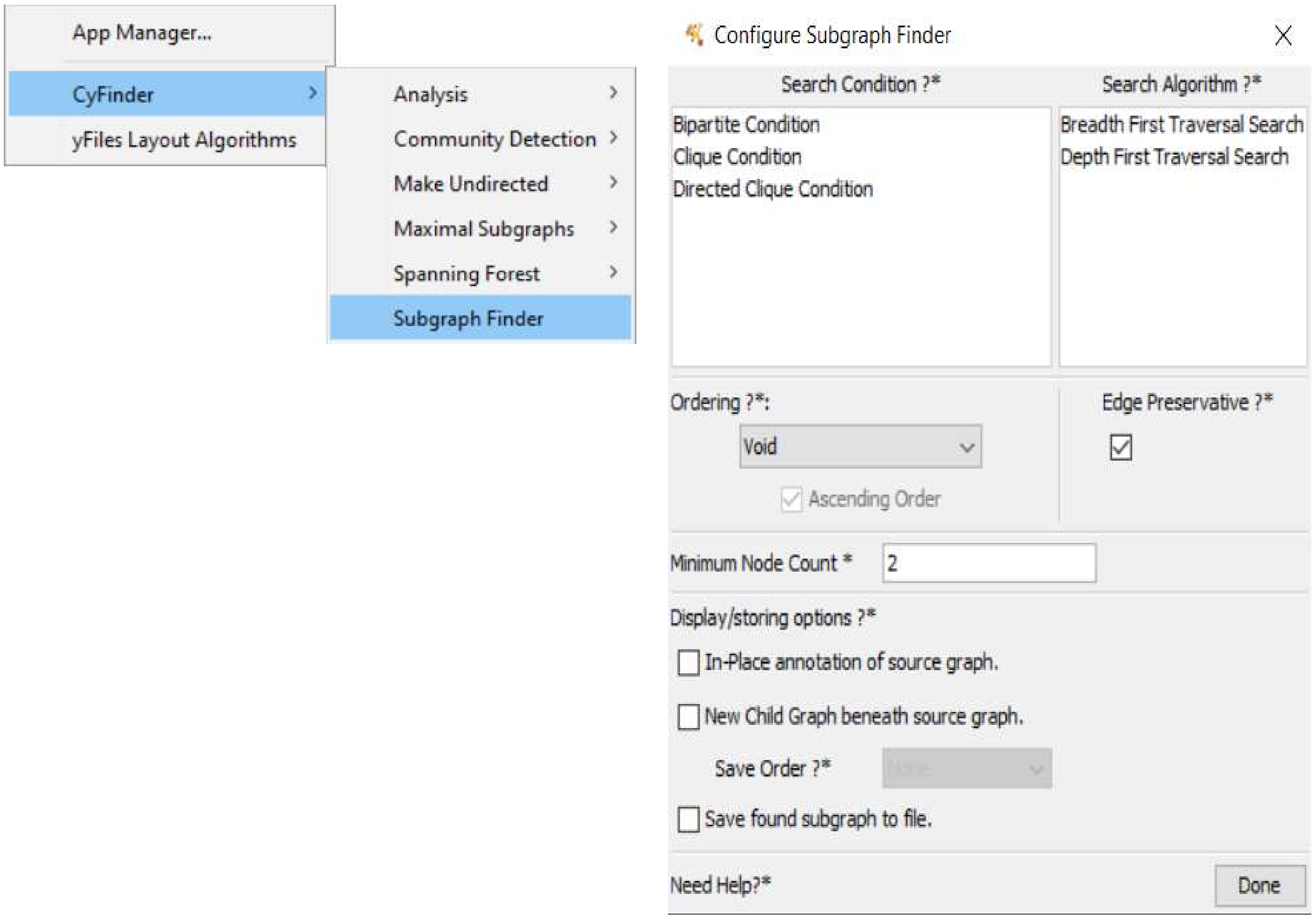


Figure 5: Examples of a clique (left) and bipartite graph (right). A clique (left) graph models several proteins working together to accomplish a common objective. A bipartite graph (right) models a flow of interaction within a biological network [8].

Subgraph Finder finds these graphs using either a Breadth First Search or a Depth First Search. While these algorithms are not guaranteed to find all subgraphs, they are computationally inexpensive so the results can be obtained quickly [8].

To use Subgraph Finder: **Apps -> CyFinder -> Subgraph Finder**. Upon selecting the feature, a Configuration Dialog will appear.



Search Condition ?*
Bipartite Condition
Clique Condition
Directed Clique Condition

Search Algorithm ?*
Breadth First Traversal Search
Depth First Traversal Search

Ordering ?*: Void
[✓] Ascending Order

Edge Preservative ?* [✓]

Minimum Node Count * 2

Display/storing options ?*
[ ] In-Place annotation of source graph.
[ ] New Child Graph beneath source graph.
Save Order ?* None
[ ] Save found subgraph to file.

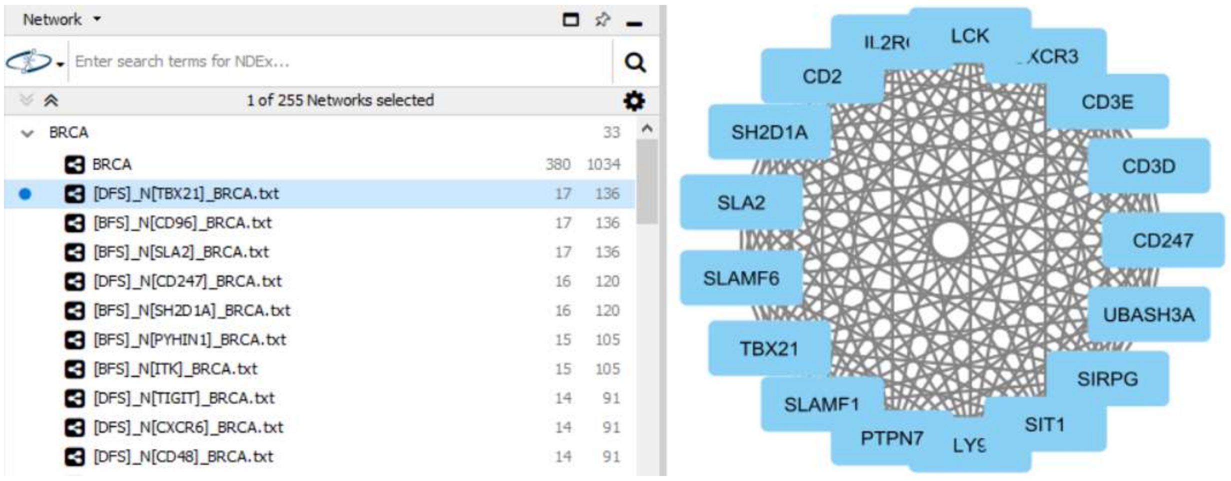
Need Help?*

Figure 6: Subgraph Finder location (left). The Subgraph Finder Configuration Dialog (right).

The Search Conditions indicate what to look for: Bipartite graphs, Cliques, and/or Directed Cliques. The Search Algorithms indicate which algorithm(s) will be executed. The Ordering indicates what attribute to use to order the subgraphs. The Edge Preservative conditions will tell the algorithms to respect the underlying structure of the original network [8]. The Minimum Node Count filters out subgraphs that have a lower node count than specified.

The Save Options tell the algorithm how to save the subgraphs:

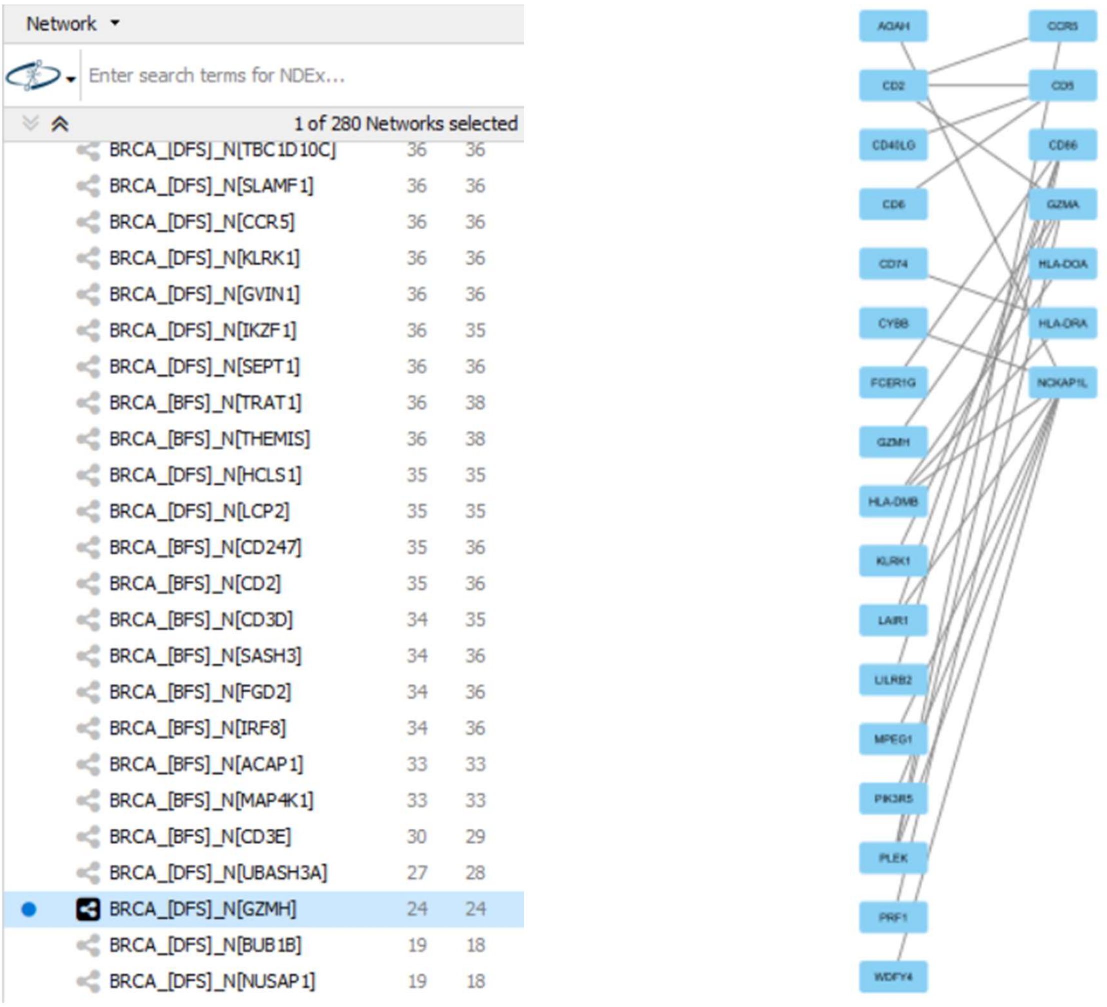
1. In-Place Annotations: boolean attributes in the Node/Edge tables indicating node/edge presence.
2. New Child Graphs, ordered by ascending/descending node count or average weight.
3. Save to File: save to a file in the file system.



1 of 255 Networks selected

BRCA		33
BRCA	380	1034
● [DFS]_N[TBX21]_BRCA.txt	17	136
[BFS]_N[CD96]_BRCA.txt	17	136
[BFS]_N[SLA2]_BRCA.txt	17	136
[DFS]_N[CD247]_BRCA.txt	16	120
[BFS]_N[SH2D1A]_BRCA.txt	16	120
[BFS]_N[PYHIN1]_BRCA.txt	15	105
[BFS]_N[ITK]_BRCA.txt	15	105
[DFS]_N[TIGIT]_BRCA.txt	14	91
[DFS]_N[CXCR6]_BRCA.txt	14	91
[DFS]_N[CD48]_BRCA.txt	14	91

Figure 7: Subgraph Finder Cliques with both BFS and DFS on BRCA. 32 Cliques with at least 10 Nodes. Executed in less than 1 minute.



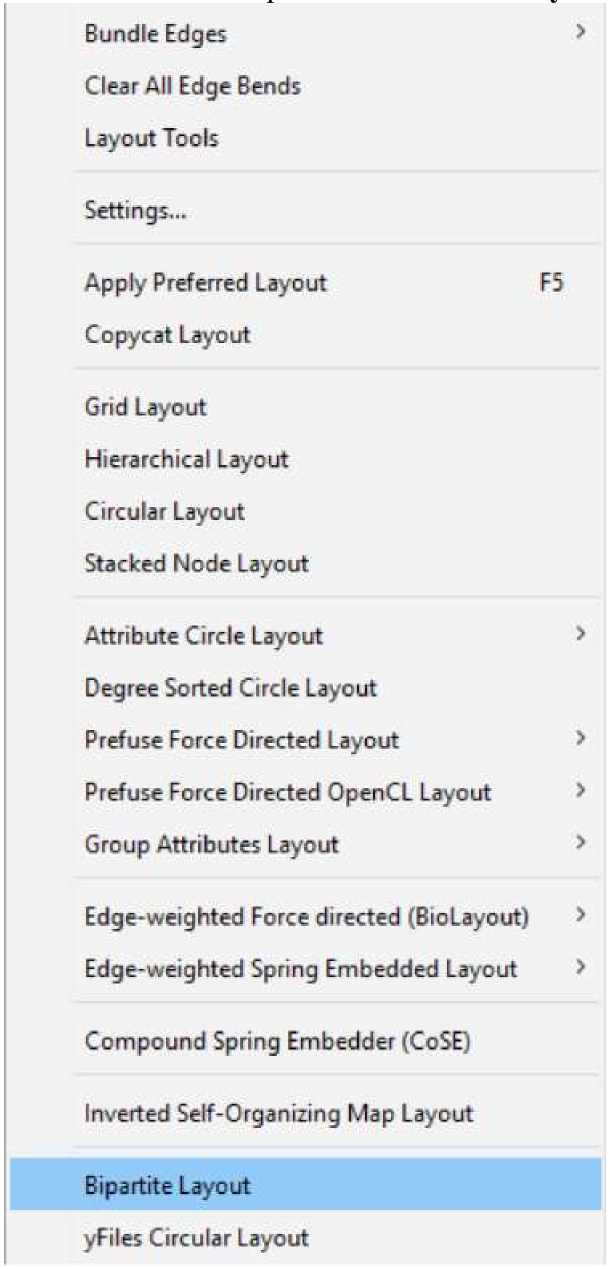
1 of 280 Networks selected

BRCA_[DFS]_N[TBC1D10C]	36	36
BRCA_[DFS]_N[SLAMF1]	36	36
BRCA_[DFS]_N[CCR5]	36	36
BRCA_[DFS]_N[KLRK1]	36	36
BRCA_[DFS]_N[GVIN1]	36	36
BRCA_[DFS]_N[IKZF1]	36	35
BRCA_[DFS]_N[SEPT1]	36	36
BRCA_[BFS]_N[TRAT1]	36	38
BRCA_[BFS]_N[THEMIS]	36	38
BRCA_[DFS]_N[HCLS1]	35	35
BRCA_[DFS]_N[LCP2]	35	35
BRCA_[BFS]_N[CD247]	35	36
BRCA_[BFS]_N[CD2]	35	36
BRCA_[BFS]_N[CD3D]	34	35
BRCA_[BFS]_N[SASH3]	34	36
BRCA_[BFS]_N[FGD2]	34	36
BRCA_[BFS]_N[IRF8]	34	36
BRCA_[BFS]_N[ACAP1]	33	33
BRCA_[BFS]_N[MAP4K1]	33	33
BRCA_[BFS]_N[CD3E]	30	29
BRCA_[DFS]_N[UBASH3A]	27	28
● BRCA_[DFS]_N[GZMH]	24	24
BRCA_[DFS]_N[BUB1B]	19	18
BRCA_[DFS]_N[NUSAP1]	19	18

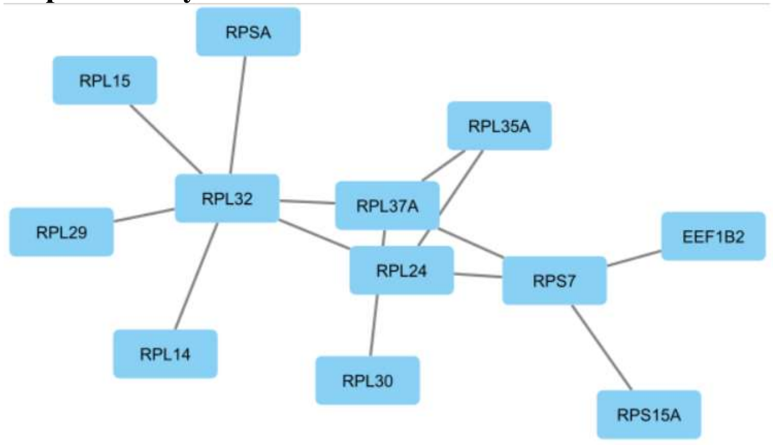
Figure 8: Subgraph Finder Bipartite graphs with both BFS and DFS on BRCA. 270 Bipartite graphs with at least 10 Nodes. Executed in less than 1 minute.

FEATURE 3 - BIPARTITE LAYOUT

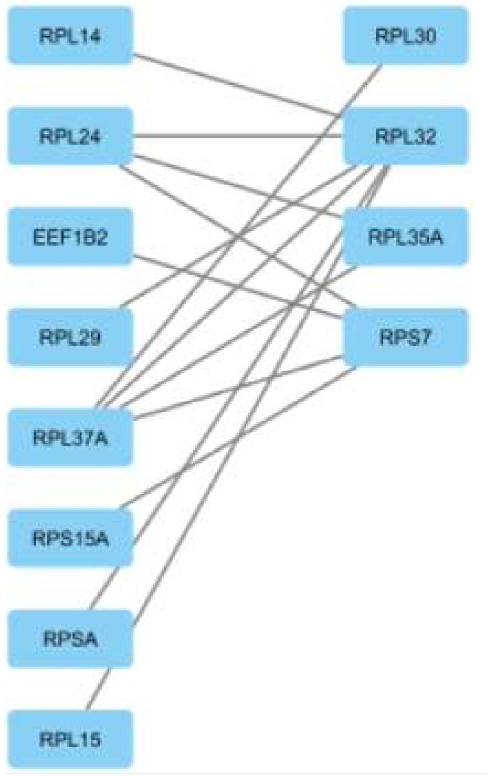
When dealing with bipartite graphs that are results of an algorithm, Cytoscape usually displays them with a random layout. We want to visualize bipartite graphs as two columns each representing a partite set. To accomplish this, we use CyFinder's Bipartite Layout feature. Bipartite Layout is applied to all the selected network views. The option is located in: **Layout -> Bipartite Layout**.


Bundle Edges
Clear All Edge Bends
Layout Tools
Settings...
Apply Preferred Layout F5
Copycat Layout
Grid Layout
Hierarchical Layout
Circular Layout
Stacked Node Layout
Attribute Circle Layout
Degree Sorted Circle Layout
Prefuse Force Directed Layout
Prefuse Force Directed OpenCL Layout
Group Attributes Layout
Edge-weighted Force directed (BioLayout)
Edge-weighted Spring Embedded Layout
Compound Spring Embedder (CoSE)
Inverted Self-Organizing Map Layout
Bipartite Layout
yFiles Circular Layout

a



b



c

Figure 9: (a) Bipartite Layout Location in the Layout Menu. (b) Network before Bipartite Layout. (c) Network after Bipartite Layout. Executed in less than 1 minute.

Feature 4 - Connected Components

The Cancer Networks that we import to Cytoscape may or may not be connected when in an undirected state. We want to separate these disconnected fragments into Connected Components: the set of connected sections of a network. We accomplish this with a Depth-First search that relocates to a different component when the search in a component ends [9].

To find Connected Components: **Apps->CyFinder->Maximal Subgraphs->Connected Components**

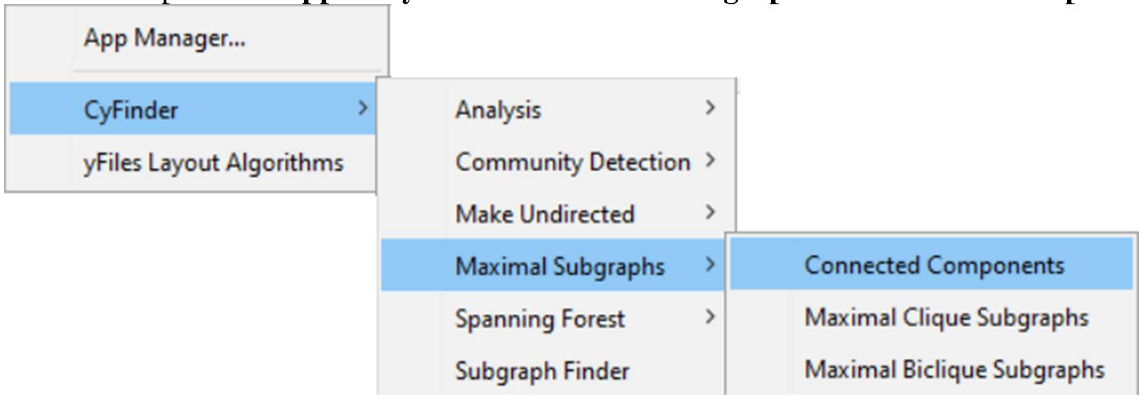
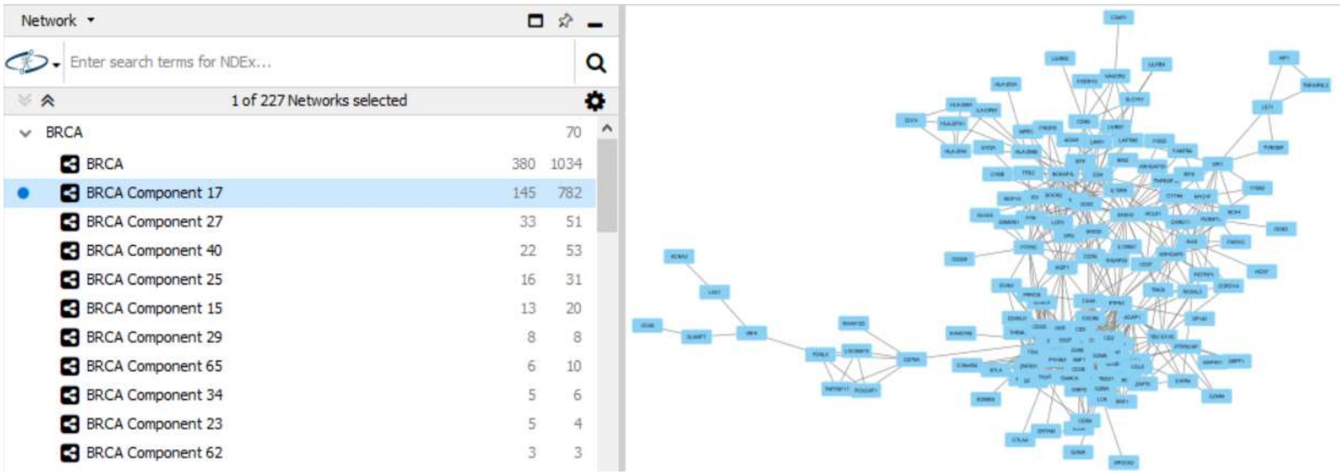


Figure 10: Connected Components in Maximal Subgraphs Options.



BRCA		70
BRCA	380	1034
BRCA Component 17	145	782
BRCA Component 27	33	51
BRCA Component 40	22	53
BRCA Component 25	16	31
BRCA Component 15	13	20
BRCA Component 29	8	8
BRCA Component 65	6	10
BRCA Component 34	5	6
BRCA Component 23	5	4
BRCA Component 62	3	3

Figure 11: Connected Components of BRCA. 69 Components. Executed in less than 1 minute.

FEATURE 5 - MAXIMAL CLIQUES

When we search for Cliques with the Subgraph Finder, we are not guaranteed to find all the cliques. In this use case, we want to directly find the Maximal Cliques of the Network. For this purpose, we use the Bron-Kersboch Algorithm. The algorithm enumerates all Maximal Cliques: cliques that cannot be expanded and remain cliques [10]. The algorithm is implemented in its pivoting variant, which improves efficiency by reducing the number of recursive calls from the search tree [11].

To find Maximal Cliques: **Apps->CyFinder->Maximal Subgraphs-> Maximal Clique Subgraphs**

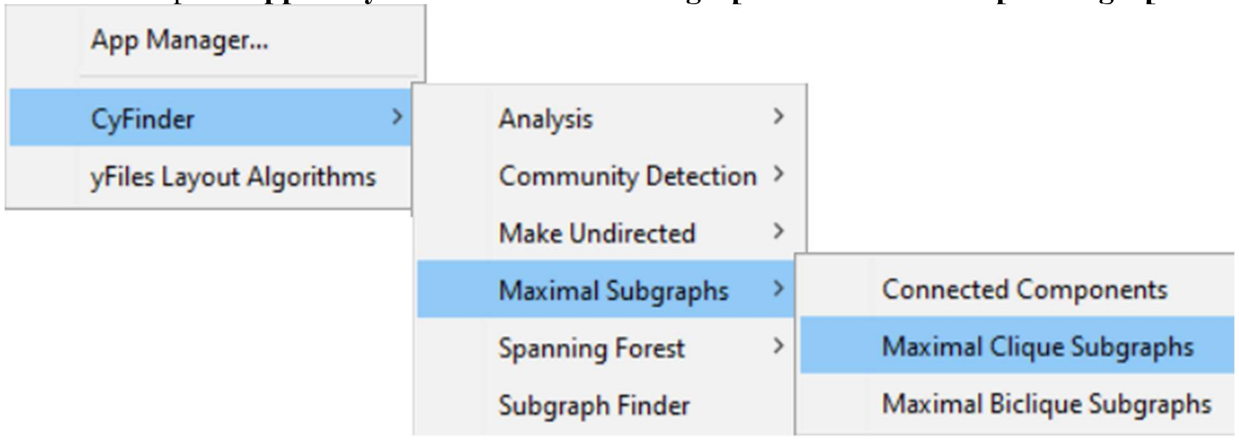
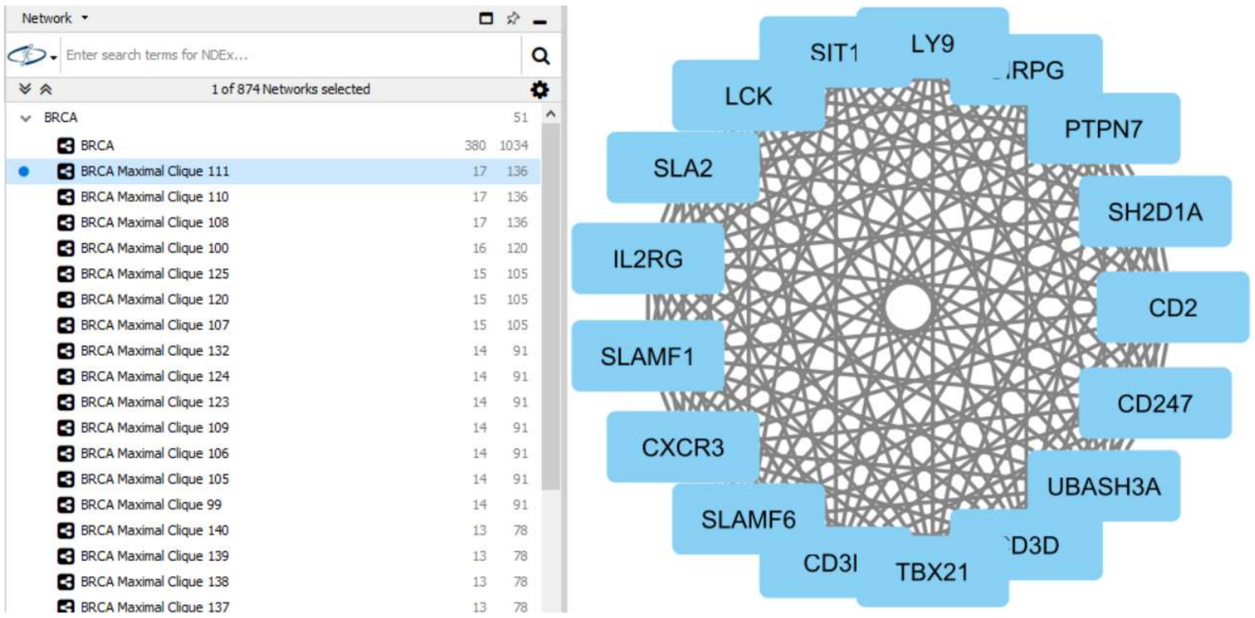


Figure 12: Maximal Clique Subgraphs in Maximal Subgraphs Options.



```
1 of 874 Networks selected
BRCA                          51
  BRCA                        380  1034
● BRCA Maximal Clique 111     17   136
  BRCA Maximal Clique 110     17   136
  BRCA Maximal Clique 108     17   136
  BRCA Maximal Clique 100     16   120
  BRCA Maximal Clique 125     15   105
  BRCA Maximal Clique 120     15   105
  BRCA Maximal Clique 107     15   105
  BRCA Maximal Clique 132     14   91
  BRCA Maximal Clique 124     14   91
  BRCA Maximal Clique 123     14   91
  BRCA Maximal Clique 109     14   91
  BRCA Maximal Clique 106     14   91
  BRCA Maximal Clique 105     14   91
  BRCA Maximal Clique 99      14   91
  BRCA Maximal Clique 140     13   78
  BRCA Maximal Clique 139     13   78
  BRCA Maximal Clique 138     13   78
  BRCA Maximal Clique 137     13   78
```

Figure 13: Maximal Cliques of BRCA with at least 10 Nodes. 50 Cliques. Executed in less than 1 minute.

FEATURE 6 - MAXIMAL BICLIQUES

We want to directly find the Maximal Bicliques or Maximal Complete Bipartite subgraphs of a Bipartite Network. We also accomplish this with the Bron-Kersboch algorithm. Since the input network is bipartite, we add edges between all nodes in the same partite set, run Bron-Kersboch on it, and then remove the added edges from the cliques from the output.

To find Maximal Bicliques: **Apps->CyFinder->Maximal Subgraphs-> Maximal Biclique Subgraphs**

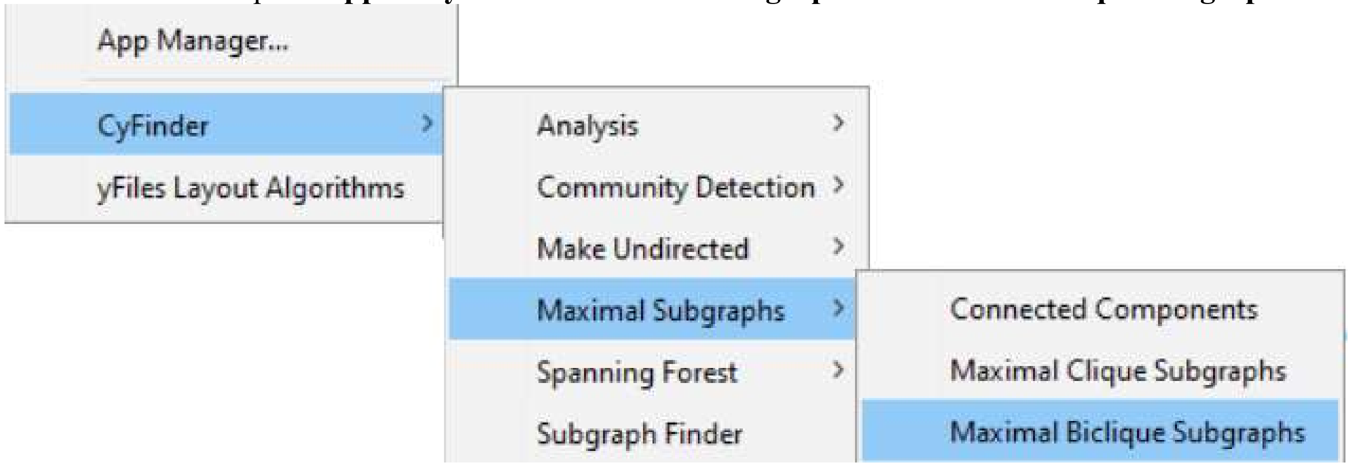
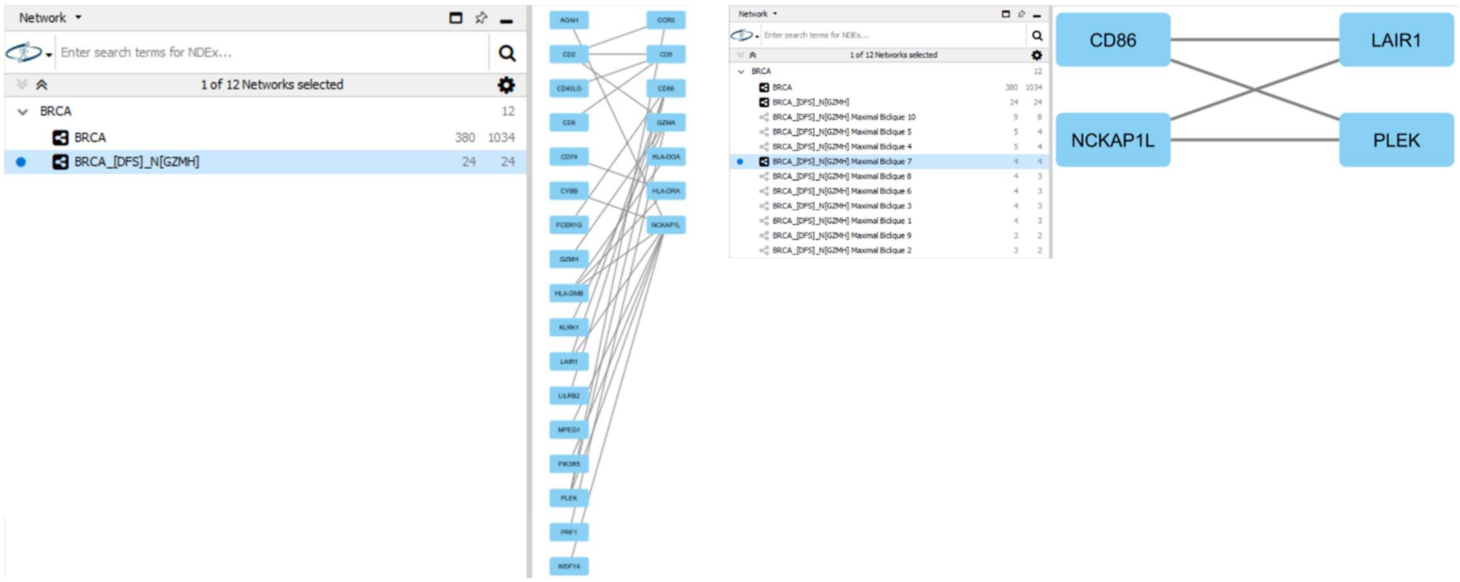


Figure 14: Maximal Biclique Subgraphs in Maximal Subgraphs Options.



BRCA		12
BRCA	380	1034
BRCA_[DFS]_N[GZMH]	24	24

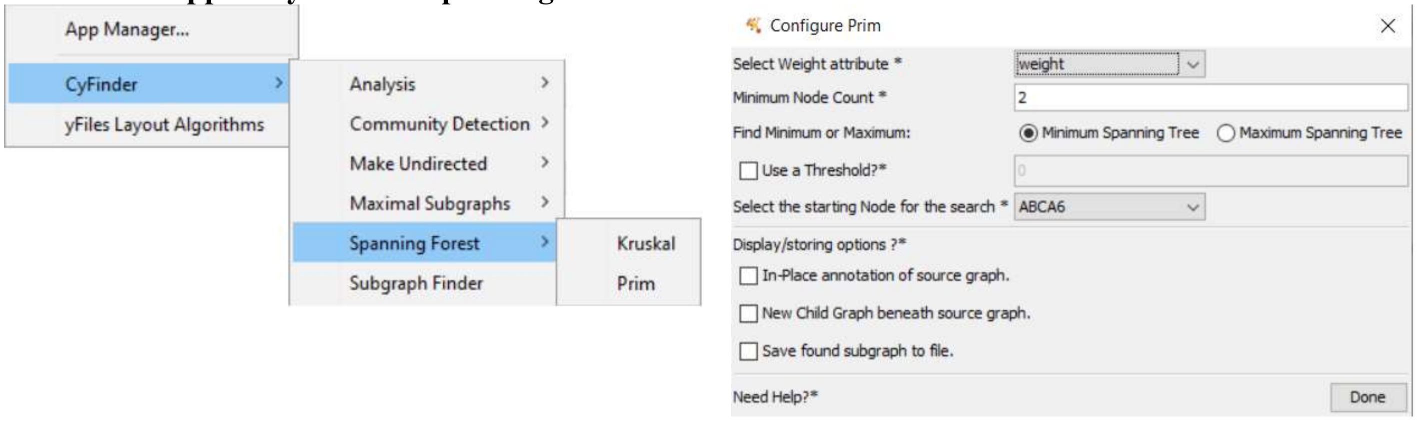
BRCA		12
BRCA	380	1034
BRCA_[DFS]_N[GZMH]	24	24
BRCA_[DFS]_N[GZMH] Maximal Biclique 10	9	8
BRCA_[DFS]_N[GZMH] Maximal Biclique 5	5	4
BRCA_[DFS]_N[GZMH] Maximal Biclique 4	5	4
BRCA_[DFS]_N[GZMH] Maximal Biclique 7	4	4
BRCA_[DFS]_N[GZMH] Maximal Biclique 8	4	3
BRCA_[DFS]_N[GZMH] Maximal Biclique 6	4	3
BRCA_[DFS]_N[GZMH] Maximal Biclique 3	4	3
BRCA_[DFS]_N[GZMH] Maximal Biclique 1	4	3
BRCA_[DFS]_N[GZMH] Maximal Biclique 9	3	2
BRCA_[DFS]_N[GZMH] Maximal Biclique 2	3	2

Figure 15: Bipartite Subgraph of BRCA: 24 Nodes, 24 Edges (left). Maximal Bicliques of the Subgraph: 10 Bicliques (right). Executed in less than 1 minute.

FEATURE 7 - MAXIMUM SPANNING FOREST

We want to find the Maximum Spanning Trees of the Cancer Networks. The reason we want Maximum instead of Minimum is because weight attributes can represent the strength of the interaction between proteins [8]. We accomplish this using two Extremum Spanning Tree algorithms: Kruskal, and Prim. Kruskal simply selects the edges whose weights have priority, in this case high weight. Prim constructs the Tree by doing a search that prioritizes edges with high weights [9]. The implementations of the algorithms in CyFinder accept disconnected graphs, which results in Extremum Spanning Forests rather than simple Extremum Spanning Trees.

To use Kruskal: **Apps->CyFinder->Spanning Tree->Kruskal**
To use Prim: **Apps->CyFinder->Spanning Tree->Prim**



Configure Prim

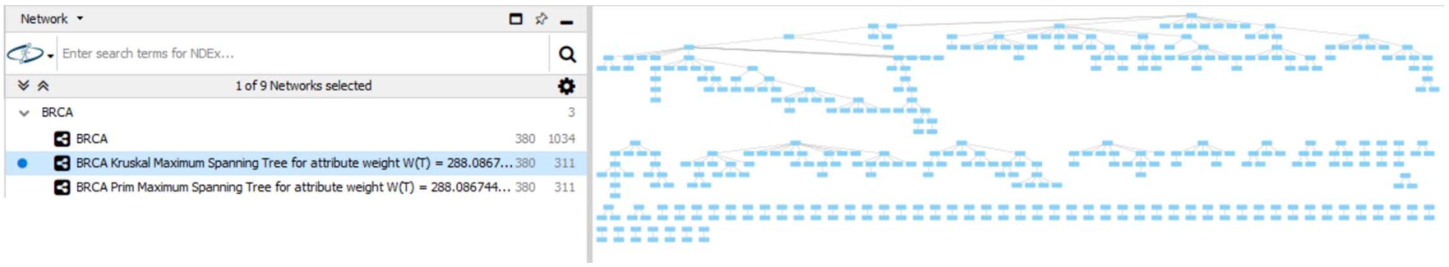
Select Weight attribute *	weight
Minimum Node Count *	2
Find Minimum or Maximum:	[✓ Minimum Spanning Tree] Maximum Spanning Tree
☐ Use a Threshold?*	0
Select the starting Node for the search *	ABCA6

Display/storing options ?*
☐ In-Place annotation of source graph.
☐ New Child Graph beneath source graph.
☐ Save found subgraph to file.

Need Help?* Done

Figure 16: Spanning Forest Options (left). Prim Configuration Dialog (Right).

Prim has a Configuration Dialog with new options. It gives the ability to choose a weight attribute to use in the algorithm, the option to find Minimum or Maximum Spanning Tree, and the starting node of the Prim search. A threshold for the weights can also be chosen. When finding a Minimum Spanning Tree edges with weight higher than the threshold are filtered out. When finding Maximum Spanning Tree edges with weight lower than the threshold are filtered out. The configuration dialog for Kruskal is the same except it does not have the starting node selection.



1 of 9 Networks selected

BRCA		3
BRCA	380	1034
BRCA Kruskal Maximum Spanning Tree for attribute weight W(T) = 288.0867...	380	311
BRCA Prim Maximum Spanning Tree for attribute weight W(T) = 288.086744...	380	311

Figure 17: Maximum Spanning Forests of BRCA with Kruskal and Prim. Forest has 311 edges, and a total weight of 288.087. Executed in less than 1 minute.

There is a Cytoscape plugin that contains a Tree Layout to better observe Graphs as Trees or Forests [12].

FEATURE 8 - COMMUNITY DETECTION

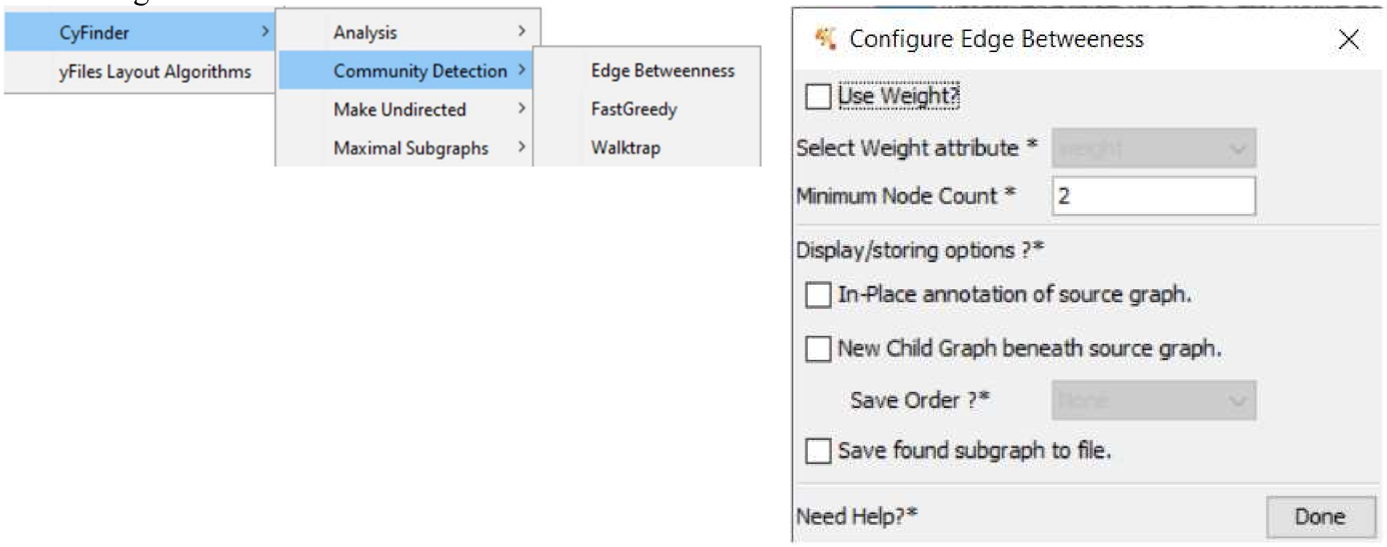
Community Detection allow us to find non-overlapping collections of genes that are strongly co-expressed among themselves and weakly co-expressed with genes in other communities. In Community Detection we find clusters of related Nodes in a Network [13].

CyFinder currently offers three Community Detection Algorithms: Fastgreedy [14], Edge Betweenness [15], and Walktrap [16].

The Community Detection algorithms vary in method; therefore their results and performance differ. Fastgreedy is the fastest while Edge Betweenness is the slowest.

The Communities outputted by the algorithms belong to the input Network's Community structure with the best Modularity. Modularity is a numeric property that gives the quality of the potential Community structure of a Network [14], [17].

The implementations of the Community Detection algorithms are based on those in the igraph software package [18]. The algorithms have a no weight variant in which all edges are given an equal value of 1 and a weighted variant in which a weight attribute is chosen among the Edge attributes of the Networks in Cytoscape. We will use the weighted variant on the cancer.



Configure Edge Betweeness

☐ Use Weight?

Select Weight attribute * weight

Minimum Node Count * 2

Display/storing options ?*

☐ In-Place annotation of source graph.

☐ New Child Graph beneath source graph.

Save Order ?* None

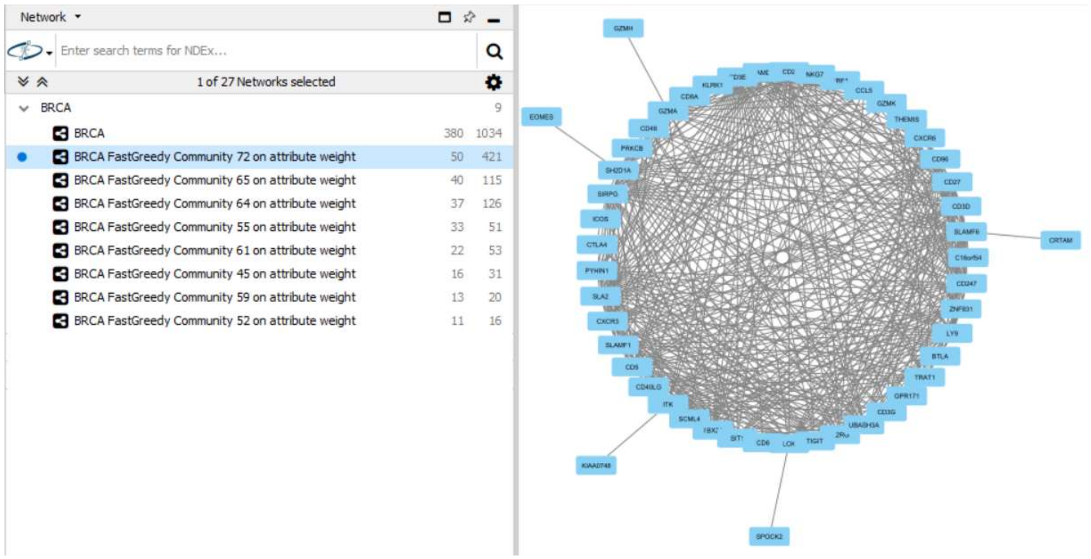
☐ Save found subgraph to file.

Figure 18: Community Detection Options (left). Community Detection Algorithms Config Dialog (right); Note option to use weighted variant and weight attribute selection.

Feature 8.1 - Fastgreedy algorithm

The FastGreedy algorithm [14] is a greedy optimization algorithm on the Modularity as the chosen community structure quality measure. It begins by creating single-node communities with the Network genes; then, at each step, two communities are merged so that the Modularity is maximized. The weighted variant adjusts the candidate modularity changes to account for the edge weights. As the algorithm runs, the Modularity increases until it peaks; then the communities at that peak are outputted.

To use FastGreedy: **Apps -> CyFinder -> Community Detection -> FastGreedy**.



1 of 27 Networks selected

BRCA	9	
BRCA	380	1034
BRCA FastGreedy Community 72 on attribute weight	50	421
BRCA FastGreedy Community 65 on attribute weight	40	115
BRCA FastGreedy Community 64 on attribute weight	37	126
BRCA FastGreedy Community 55 on attribute weight	33	51
BRCA FastGreedy Community 61 on attribute weight	22	53
BRCA FastGreedy Community 45 on attribute weight	16	31
BRCA FastGreedy Community 59 on attribute weight	13	20
BRCA FastGreedy Community 52 on attribute weight	11	16

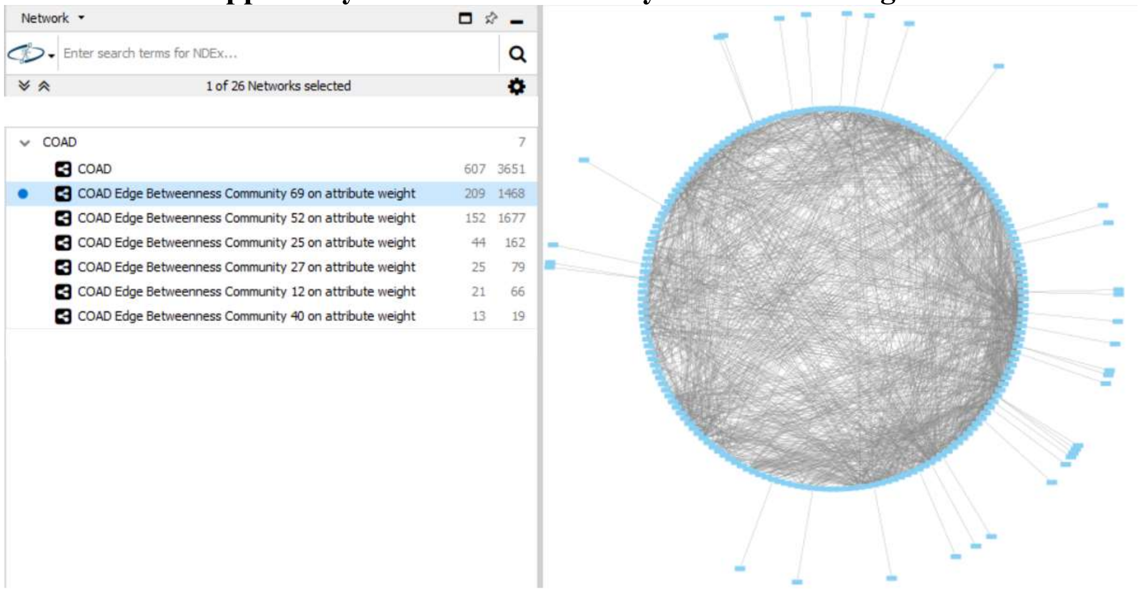
Figure 19: FastGreedy weighted algorithm on BRCA. 8 Communities with at least 10 Nodes. Executed in less than 1 minute.

The FastGreedy algorithm has the best performance of the implemented algorithms [14].

Feature 8.2 - Edge Betweenness Algorithm

The Edge Betweenness Algorithm, also called Girvan-Newman algorithm [15] relies on the notion of Edge Betweenness. If we find all the shortest paths between all pairs of Nodes in the Network, then we can define the Edge Betweenness of an edge as the number of these shortest paths that included it [15]. The weighted variant adjusts the Edge Betweenness values to account for the Edge weights. The algorithm starts with one Community with all Nodes, removes the edge with the highest Edge Betweenness and outputs the resulting connected components as the iteration's Communities then recomputes the values. The Communities with the best modularity are outputted.

To use Edge Betweenness: **Apps -> CyFinder -> Community Detection -> Edge Betweenness**.



COAD		7
COAD	607	3651
● COAD Edge Betweenness Community 69 on attribute weight	209	1468
COAD Edge Betweenness Community 52 on attribute weight	152	1677
COAD Edge Betweenness Community 25 on attribute weight	44	162
COAD Edge Betweenness Community 27 on attribute weight	25	79
COAD Edge Betweenness Community 12 on attribute weight	21	66
COAD Edge Betweenness Community 40 on attribute weight	13	19

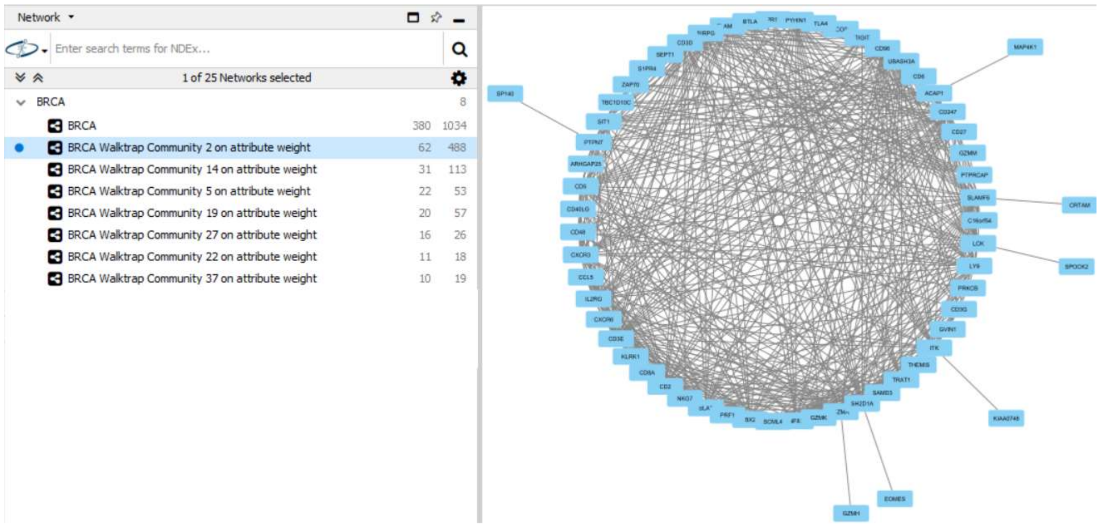
Figure 20: Edge Betweenness weighted algorithm on COAD. 6 Communities with at least 10 Nodes. Executed in less than 3 minutes.

The Edge Betweenness algorithm is the most computationally expensive of the algorithms.

Feature 8.3 - Walktrap Algorithm

The Walktrap Algorithm is based on the idea that random walks starting from a Community will tend to stay "trapped" in that same Community [16]. Instead of simulating random walks, we base the implementation on the probabilities of reaching destination Nodes from a source Node in a random walk of given length. The random walk probabilities will enable us to define a distance between communities [16]. The algorithm starts with 1-Node Communities and on each iteration merges the connected Communities that minimize the mean of the squared distances between each node and its community; this greedy approach is called the Ward method [16], [19]. The length of the walk is proportional to the running time of the algorithm and is inputted in an additional field in the Community Detection Algorithm Config Dialog. We will use the default walk length of 4.

To use Walktrap: **Apps -> CyFinder -> Community Detection -> Walktrap**.



1 of 25 Networks selected

BRCA		8
BRCA	380	1034
BRCA Walktrap Community 2 on attribute weight	62	488
BRCA Walktrap Community 14 on attribute weight	31	113
BRCA Walktrap Community 5 on attribute weight	22	53
BRCA Walktrap Community 19 on attribute weight	20	57
BRCA Walktrap Community 27 on attribute weight	16	26
BRCA Walktrap Community 22 on attribute weight	11	18
BRCA Walktrap Community 37 on attribute weight	10	19

Figure 21: Walktrap weighted algorithm on BRCA with walk length 4. 7 Communities with at least 10 Nodes. Executed in less than 1 minute.

FEATURE 9 – ANALYSIS

When we obtain potential biomarkers in the form of cliques or communities across different cancers, we might want to compare the biomarkers of the different disease Networks. For this purpose, CyFinder has an Analysis section. The Analysis features at this point are few and very simple: Intersecting two Networks and Intersecting all Networks in two collections. The intersection of two Graphs is defined as the Graph that has the common Nodes and Edges of the two input Graphs.

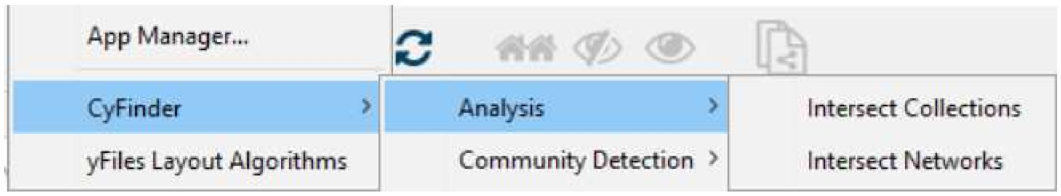
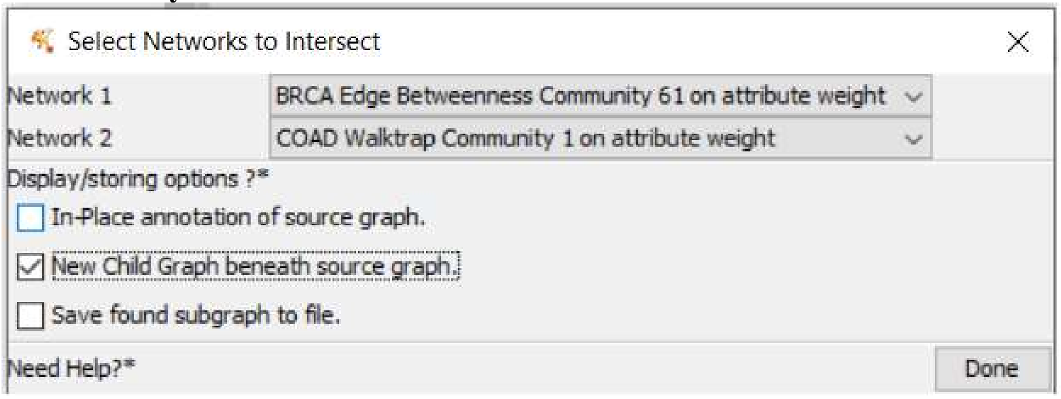


Figure 22: Analysis Options.

Feature 9.1 – Intersecting Networks

In this feature, two different Networks are selected, and their intersection is outputted. To use it:
Apps -> CyFinder -> Analysis -> Intersect Networks.



Select Networks to Intersect

Network 1: BRCA Edge Betweenness Community 61 on attribute weight
Network 2: COAD Walktrap Community 1 on attribute weight

Display/storing options ?*

- [ ] In-Place annotation of source graph.
- [✓] New Child Graph beneath source graph.
- [ ] Save found subgraph to file.

Need Help?*

Figure 23: Intersect Networks Config Dialog.

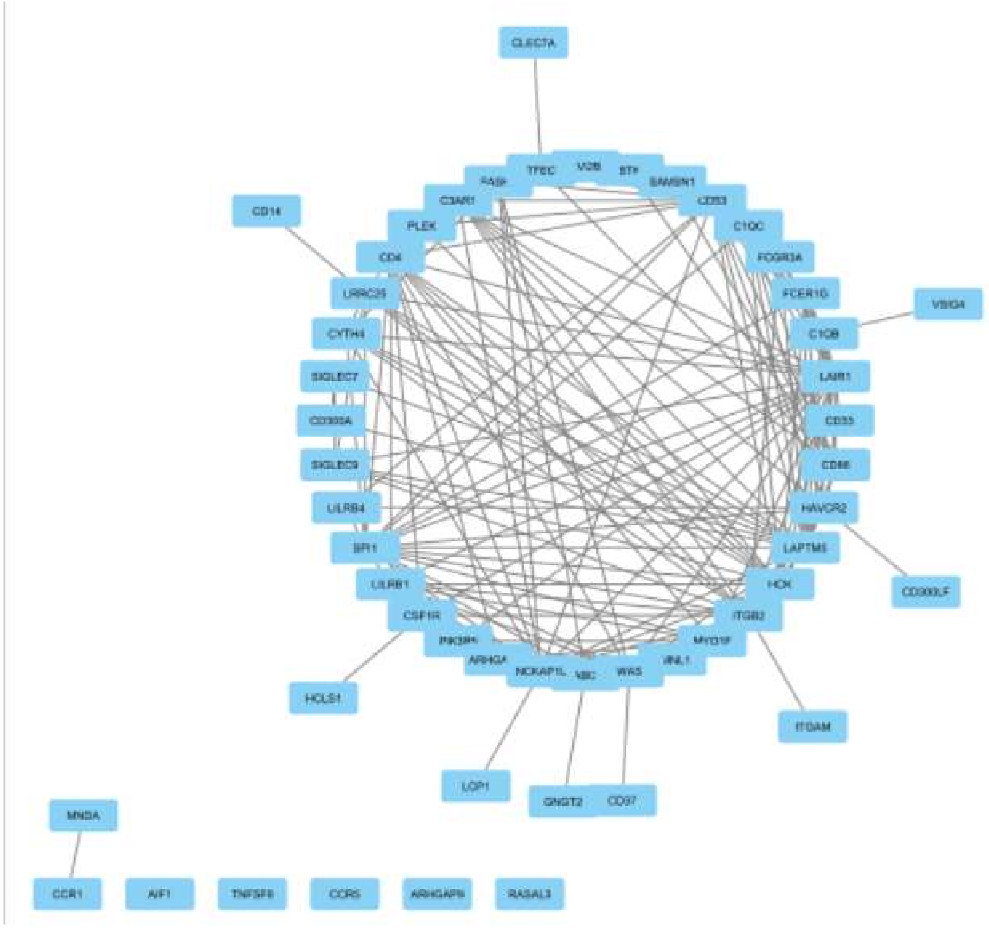
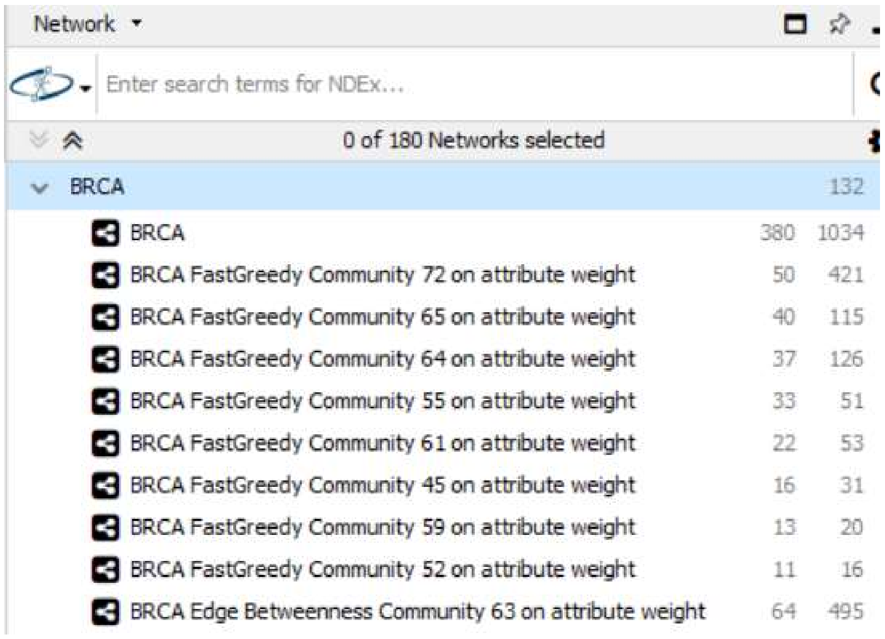


Figure 24: COAD Edge Betweenness Community 20 - GBM Edge Betweenness Community 51 Intersection. 52 Nodes, 163 Edges. Executed in less than 1 minute.

Feature 9.2 – Intersecting Collections

We usually deal with dozens or thousands of Networks in a single Cytoscape Session. For that reason, intersecting individual networks can become very time consuming. To overcome that, we can intersect all the Networks within a Collection with all the Networks in another Collection.



Network ▾

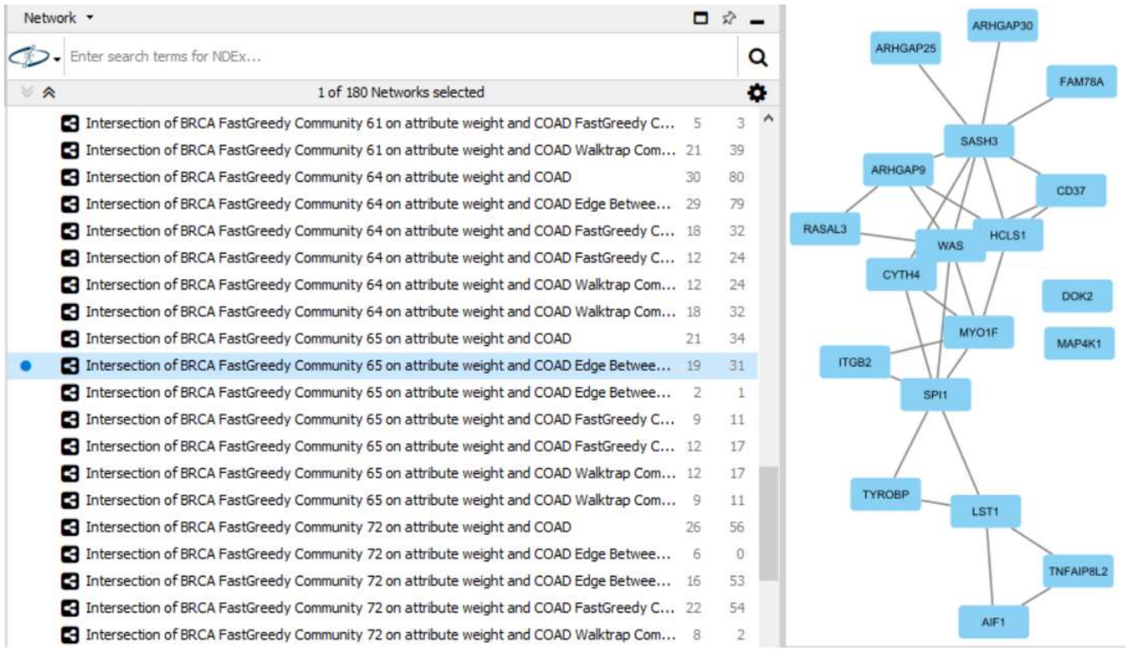
Enter search terms for NDEx...

0 of 180 Networks selected

BRCA		132
BRCA	380	1034
BRCA FastGreedy Community 72 on attribute weight	50	421
BRCA FastGreedy Community 65 on attribute weight	40	115
BRCA FastGreedy Community 64 on attribute weight	37	126
BRCA FastGreedy Community 55 on attribute weight	33	51
BRCA FastGreedy Community 61 on attribute weight	22	53
BRCA FastGreedy Community 45 on attribute weight	16	31
BRCA FastGreedy Community 59 on attribute weight	13	20
BRCA FastGreedy Community 52 on attribute weight	11	16
BRCA Edge Betweenness Community 63 on attribute weight	64	495

Figure 25: A selected Collection of the BRCA Cancer. Number of Networks to the right.

To Intersect Collections: **Apps -> CyFinder -> Analysis -> Intersect Collections.**
The Configuration Dialog is the same, but Collections are chosen instead of Networks and there is a minimum Node count selector.



Network ▾

Enter search terms for NDEx...

1 of 180 Networks selected

Intersection of BRCA FastGreedy Community 61 on attribute weight and COAD FastGreedy C...	5	3
Intersection of BRCA FastGreedy Community 61 on attribute weight and COAD Walktrap Com...	21	39
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD	30	80
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD Edge Betwee...	29	79
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD FastGreedy C...	18	32
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD FastGreedy C...	12	24
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD Walktrap Com...	12	24
Intersection of BRCA FastGreedy Community 64 on attribute weight and COAD Walktrap Com...	18	32
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD	21	34
● Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD Edge Betwee...	19	31
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD Edge Betwee...	2	1
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD FastGreedy C...	9	11
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD FastGreedy C...	12	17
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD Walktrap Com...	12	17
Intersection of BRCA FastGreedy Community 65 on attribute weight and COAD Walktrap Com...	9	11
Intersection of BRCA FastGreedy Community 72 on attribute weight and COAD	26	56
Intersection of BRCA FastGreedy Community 72 on attribute weight and COAD Edge Betwee...	6	0
Intersection of BRCA FastGreedy Community 72 on attribute weight and COAD Edge Betwee...	16	53
Intersection of BRCA FastGreedy Community 72 on attribute weight and COAD FastGreedy C...	22	54
Intersection of BRCA FastGreedy Community 72 on attribute weight and COAD Walktrap Com...	8	2

Figure 26: Intersection of BRCA and COAD Collections. 108 Networks. Executed in less than 1 minute.

GLOSSARY

- BFTS/BFS – Breadth First Traversal Search
- Biclique – A bipartite graph in which each Node is connected to all the Nodes in the other partite set.
- Bipartite graph – A graph in which nodes are separated into two partite sets. Nodes in the same partite set are not connected.
- Clique – A graph in which each Node is connected to all other Nodes.
- Cluster – See Community
- Clustering – See Community Structure
- Community – Grouping of related Nodes in a Graph.
- Community Structure – Collection of non-overlapping Communities.
- Connected Components – Maximally Connected subgraph of a Graph.
- CSG – Clique Subgraph
- DFTS/DFS – Depth First Traversal Search
- Forest – A collection of disconnected Trees.
- GUI – Graphical User Interface
- Maximal – Cannot be expanded further.
- Maximum – Element with the highest size according to some attribute.
- Modularity – Quantity that gives the quality of a Community Structure in relation to its original Graph.
- Partite Set – Group of Nodes in a bipartite Graph. No Nodes in the group are connected.
- PPI – Protein-Protein Interaction
- SNB – Subnetwork Biomarker
- Spanning Forest – A forest obtained from the Spanning Trees of the Connected Components of a disconnected Graph.
- Spanning Tree – A Tree obtained from a Graph such that it contains all the Nodes in the Graph
- Tree – A connected Graph with no cycles.
- UI – User Interface

APPENDIXES

Appendix A – Save Options Explained

There is a consistent UI interface for storing results for each implemented functionality.

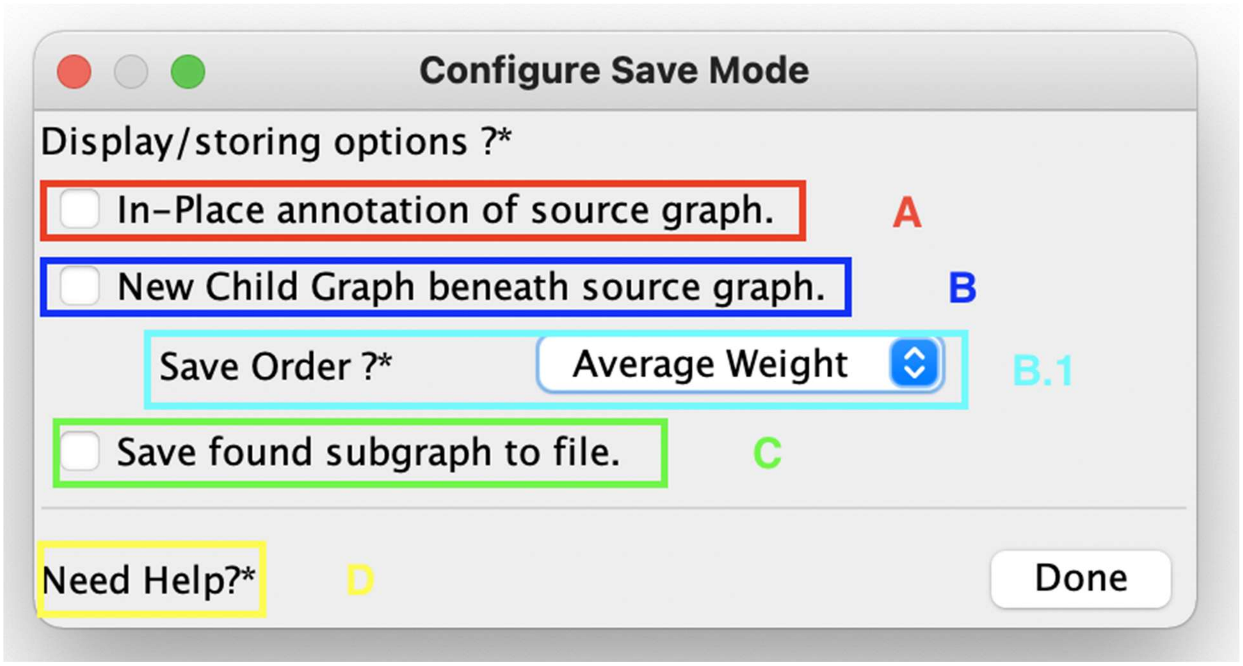
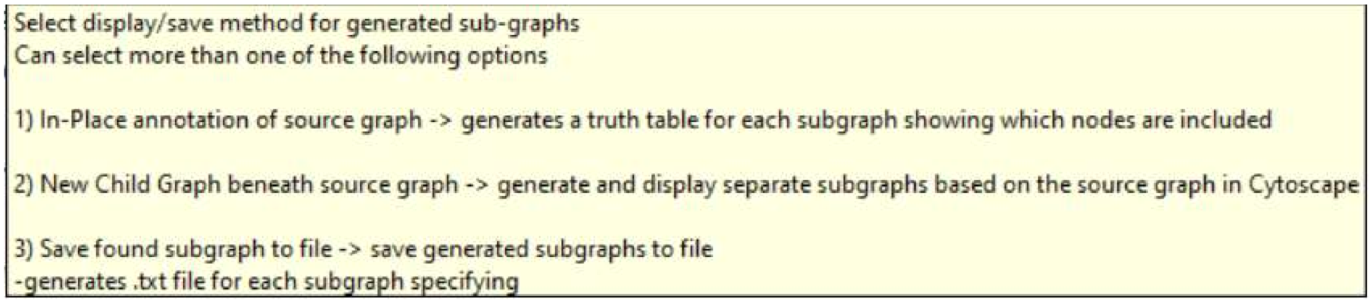

Configure Save Mode
Display/storing options ?*
In-Place annotation of source graph.
A
New Child Graph beneath source graph.
B
Save Order ?*
Average Weight
B.1
Save found subgraph to file.
C
Need Help?*
D
Done

Figure 27: Breakdown of result saving options.



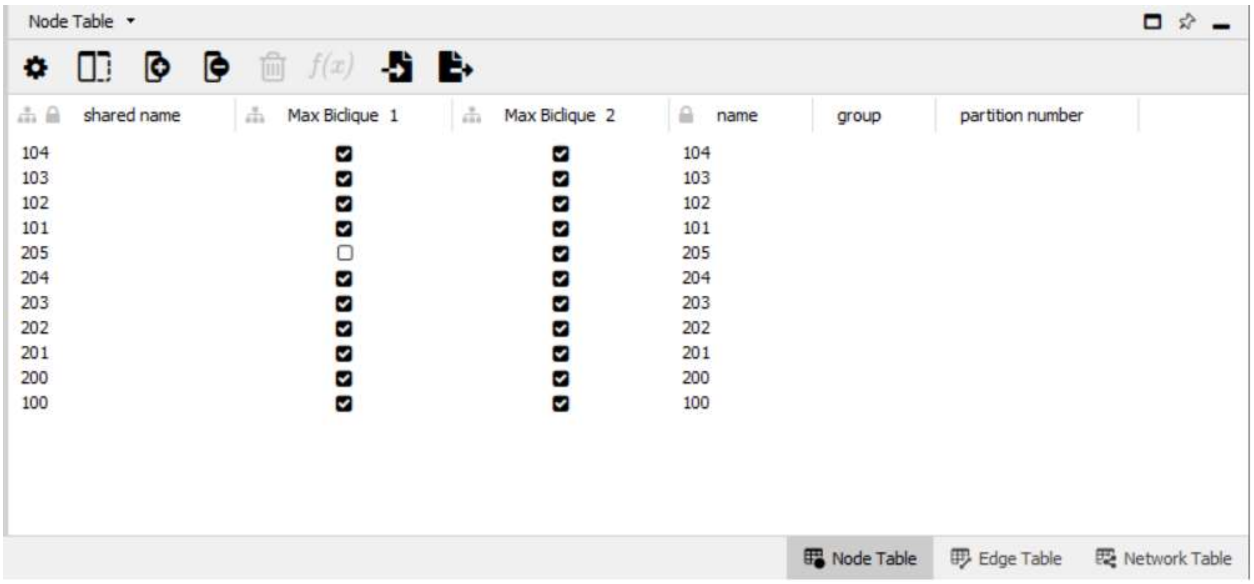
Select display/save method for generated sub-graphs
Can select more than one of the following options

1) In-Place annotation of source graph -> generates a truth table for each subgraph showing which nodes are included

2) New Child Graph beneath source graph -> generate and display separate subgraphs based on the source graph in Cytoscape

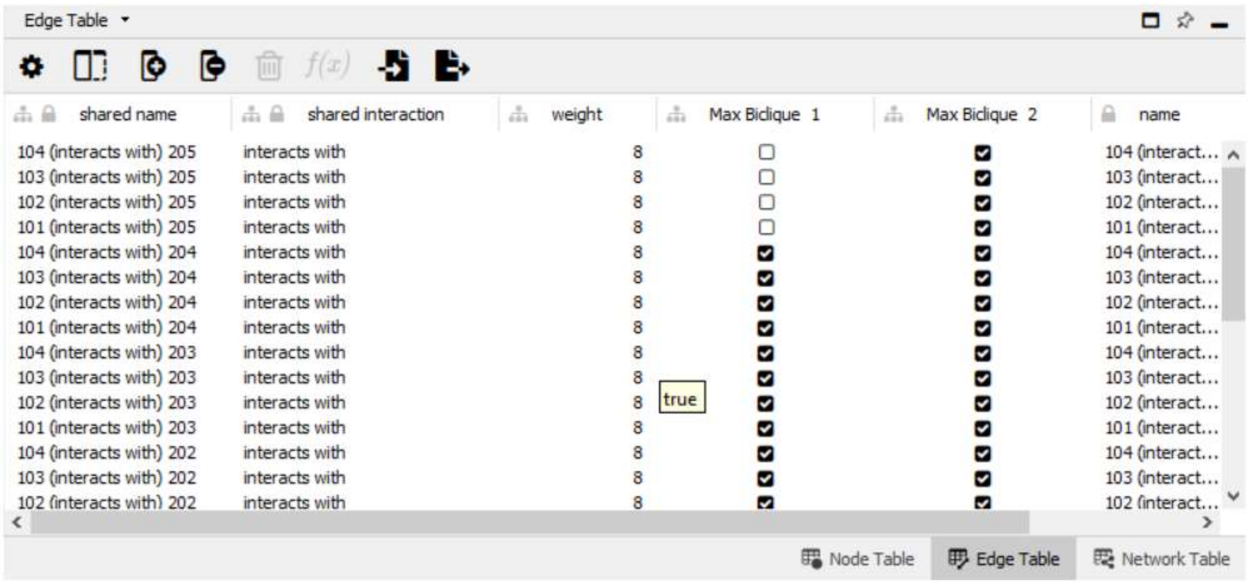
3) Save found subgraph to file -> save generated subgraphs to file
-generates .txt file for each subgraph specifying

Figure 28: Tooltip of storing result options.



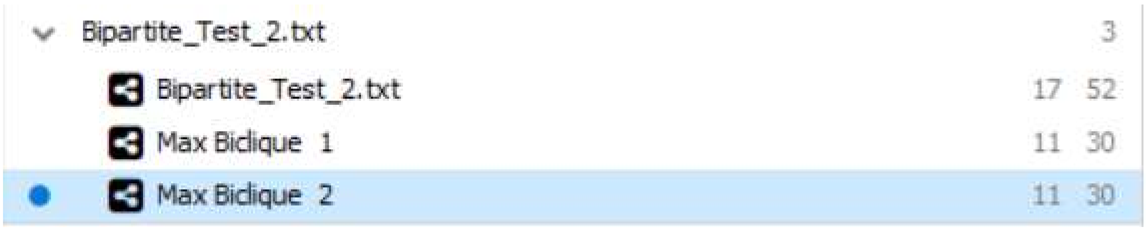
shared name	Max Biclique 1	Max Biclique 2	name	group	partition number
104	☑	☑	104		
103	☑	☑	103		
102	☑	☑	102		
101	☑	☑	101		
205	☐	☑	205		
204	☑	☑	204		
203	☑	☑	203		
202	☑	☑	202		
201	☑	☑	201		
200	☑	☑	200		
100	☑	☑	100		

Figure 29: Node Table in In-Place Annotation.



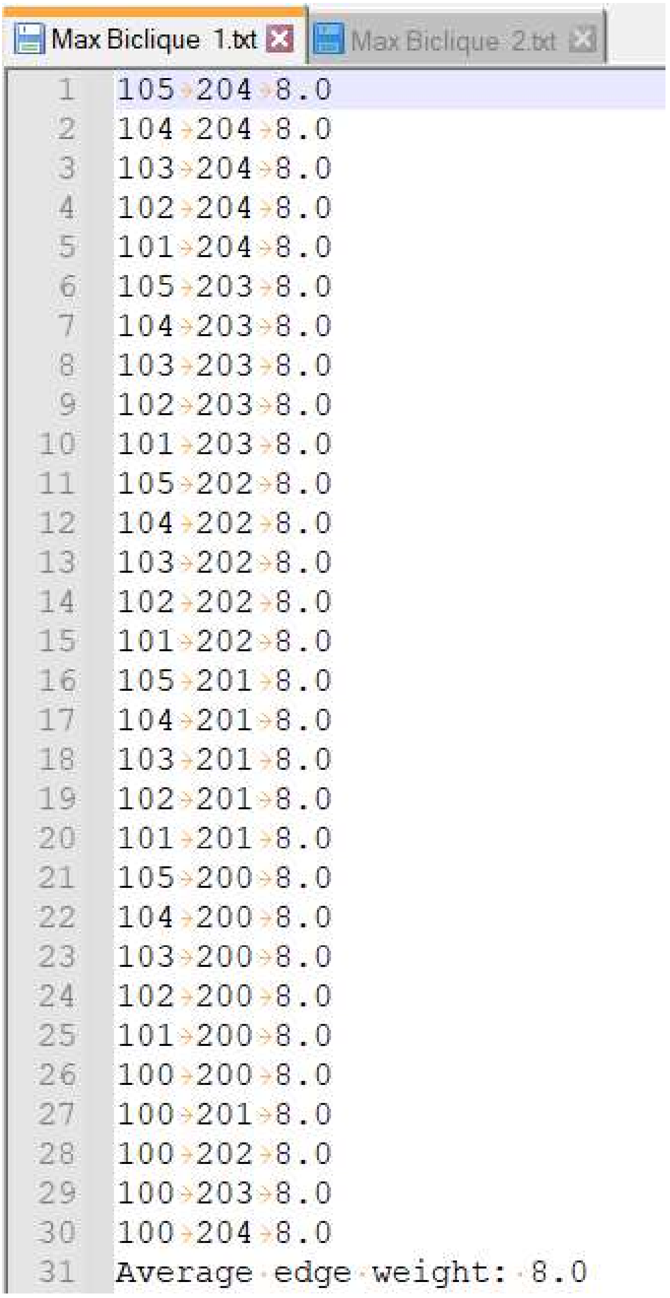
shared name	shared interaction	weight	Max Biclique 1	Max Biclique 2	name
104 (interacts with) 205	interacts with	8	☐	☑	104 (interact…
103 (interacts with) 205	interacts with	8	☐	☑	103 (interact…
102 (interacts with) 205	interacts with	8	☐	☑	102 (interact…
101 (interacts with) 205	interacts with	8	☐	☑	101 (interact…
104 (interacts with) 204	interacts with	8	☑	☑	104 (interact…
103 (interacts with) 204	interacts with	8	☑	☑	103 (interact…
102 (interacts with) 204	interacts with	8	☑	☑	102 (interact…
101 (interacts with) 204	interacts with	8	☑	☑	101 (interact…
104 (interacts with) 203	interacts with	8	☑	☑	104 (interact…
103 (interacts with) 203	interacts with	8	☑	☑	103 (interact…
102 (interacts with) 203	interacts with	8	☑	☑	102 (interact…
101 (interacts with) 203	interacts with	8	☑	☑	101 (interact…
104 (interacts with) 202	interacts with	8	☑	☑	104 (interact…
103 (interacts with) 202	interacts with	8	☑	☑	103 (interact…
102 (interacts with) 202	interacts with	8	☑	☑	102 (interact…

Figure 30: Edge table in In-Place Annotation.



Bipartite_Test_2.txt		3
Bipartite_Test_2.txt	17	52
Max Biclique 1	11	30
Max Biclique 2	11	30

Figure 31: New Child Networks added to the same Cytoscape Collection. They can now be interacted with the same way as the parent.


Max Biclique 1.txt Max Biclique 2.txt

```
105	204	8.0
104	204	8.0
103	204	8.0
102	204	8.0
101	204	8.0
105	203	8.0
104	203	8.0
103	203	8.0
102	203	8.0
101	203	8.0
105	202	8.0
104	202	8.0
103	202	8.0
102	202	8.0
101	202	8.0
105	201	8.0
104	201	8.0
103	201	8.0
102	201	8.0
101	201	8.0
105	200	8.0
104	200	8.0
103	200	8.0
102	200	8.0
101	200	8.0
100	200	8.0
100	201	8.0
100	202	8.0
100	203	8.0
100	204	8.0
Average edge weight: 8.0
```

Figure 32: Result Saved to File.

REFERENCES

[1] K. Ono, “What is Cytoscape?” https://cytoscape.org/what_is_cytoscape.html.

[2] K. Ono, “Download Cytoscape.” https://cytoscape.org/download.html.

[3] S. V. Vasaikar, P. Straub, J. Wang, and B. Zhang, “LinkedOmics: Analyzing multi-omics data within and across 32 cancer types,” *Nucleic Acids Res.*, vol. 46, no. D1, 2018, doi: 10.1093/nar/gkx1090.

[4] D. Ramsköld, E. T. Wang, C. B. Burge, and R. Sandberg, “An abundance of ubiquitously expressed genes revealed by tissue transcriptome sequence data,” *PLoS Comput. Biol.*, vol. 5, no. 12, 2009, doi: 10.1371/journal.pcbi.1000598.

[5] J. W. Rowley *et al.*, “Genome-wide RNA-seq analysis of human and mouse platelet transcriptomes,” *Blood*, vol. 118, no. 14, 2011, doi: 10.1182/blood-2011-03-339705.

[6] L. Han *et al.*, “The Pan-Cancer analysis of pseudogene expression reveals biologically and clinically relevant tumour subtypes,” *Nat. Commun.*, vol. 5, 2014, doi: 10.1038/ncomms4963.

[7] A. M. Mondal, C. A. Schultz, M. Sheppard, J. Carson, R. B. Tanvir, and T. Aqila, “Graph Theoretic Concepts as the Building Blocks for Disease Initiation and Progression at Protein Network Level: Identification and Challenges,” 2019, doi: 10.1109/BIBM.2018.8621417.

[8] C. SCHULTZ, “SAR: AN ALGORITHM FOR A STRUCTURAL ANALYSIS OF PROTEIN INTERACTION NETWORKS,” Claflin University, 2016.

[9] A. Levitin, *Introduction to The Design And Analysis of Algorithms*, 3rd ed. Pearson, 2011.

[10] C. Bron and J. Kerbosch, “Algorithm 457: Finding All Cliques of an Undirected Graph [H],” *Commun. ACM*, vol. 16, no. 9, 1973, doi: 10.1145/362342.362367.

[11] H. C. Johnston, “Cliques of a graph-variations on the Bron-Kerbosch algorithm,” *Int. J. Comput. Inf. Sci.*, vol. 5, no. 3, 1976, doi: 10.1007/BF00991836.

[12] yWorks, “yFiles Layout Algorithms.” https://apps.cytoscape.org/apps/yfileslayoutalgorithms.

[13] R. B. Tanvir and A. M. Mondal, “Cancer Biomarker Discovery from Gene Co-expression Networks Using Community Detection Methods,” 2019, doi: 10.1109/BIBM47256.2019.8982960.

[14] A. Clauset, M. E. J. Newman, and C. Moore, “Finding community structure in very large networks,” *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.*, vol. 70, no. 6, 2004, doi: 10.1103/PhysRevE.70.066111.

[15] M. Girvan and M. E. J. Newman, “Community structure in social and biological networks,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 99, no. 12, 2002, doi: 10.1073/pnas.122653799.

[16] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” *J. Graph Algorithms Appl.*, vol. 10, no. 2, 2006, doi: 10.7155/jgaa.00124.

[17] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 69, no. 2 2, 2004, doi: 10.1103/PhysRevE.69.026113.

[18] G. Csardi and T. Nepusz, “The igraph software package for complex network research,” *InterJournal Complex Syst.*, vol. Complex Sy, no. 1695, 2006.

[19] J. H. Ward, “Hierarchical Grouping to Optimize an Objective Function,” *J. Am. Stat. Assoc.*, vol. 58, no. 301, 1963, doi: 10.1080/01621459.1963.10500845.