**SCE13-0448**

# SIMULATION AND HUMAN-COMUTER INTERACTION TO ANALYZE GENE REGULATORY NETWORKS

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Computer Science
of the Nanyang Technological University

**BY**

**LINN HTET OO**

**U1120704F**

School of Computer Engineering

2014

# VERSION HISTORY

| Version # | Implemented By | Revision Date | Approved By | Approval Date | Changes |
|---|---|---|---|---|---|
| 0.1 | Linn Htet Oo | 10.02.2013 | | | Created outlines for report and added some contents |
| 0.2 | Linn Htet Oo | 10.03.2014 | | | Added updated version of content page and outlines |
| 0.3 | Linn Htet Oo | 11.03.2014 | | | Added contents for Time Series Visualization |
| 0.4 | Linn Htet Oo | 12.03.2014 | | | Added contents for Editing Network Structure |
| 0.5 | Linn Htet Oo | 13.03.2014 | | | Added contents for Time Series Visualization |
| 0.6 | Linn Htet Oo | 15.03.2014 | | | Added contents for Editing Network Structure and Time Series Visualization |
| 0.7 | Linn Htet Oo | 16.03.2014 | | | Added contents for Editing Network Structure and Time Series Visualization |
| 0.8 | Linn Htet Oo | 19.03.2014 | | | Added UML diagrams and contents for Time Series Comparison |
| 0.9 | Linn Htet Oo | 21.03.2014 | | | Added acknowledgements, recommendation and contents for Time Series Comparison |
| 1.0 | Linn Htet Oo | 22.03.2014 | | | Added abstract and conclusion |
| 1.1 | Linn Htet Oo | 23.03.2014 | | | Amended and added contents for conclusion |

# TABLE OF CONTENTS

## ABSTRACT

With rapid development in information technology, it has also become a great help to biological research and studies. Among the plentiful software applications developed to analyze Gene Regulatory Networks (GRN), GeneNetWeaver (GNW) is one of the most useful applications with its rich features and interactive Graphical User Interface (GUI). In order to make it the best out of the best ones, in this project, functionality of GNW has been enhanced by integrating more features into it. With these, the efficiency and effectiveness of the application will be fully utilized. In this project, we have integrated important features such as visualizing the gene expression datasets generated from GRN, which would greatly improve users' convenience. Another essential feature we have developed is allowing users to modify the structure of GRN by deleting nodes (genes) or edges (interactions), which could simulate 'knock-out' experiments. Moreover, to help users realize the effects of particular modification to GRN structure, we have implemented the feature to compare the gene expression data before and after the modification. These new features will offer users great help to gain further understanding into the function of GRN.

## ACKNOWLEDGEMENTS

Firstly, I would like to thank Professor Zheng Jie for giving me a chance to work on this bioinformatics project. This project gave me a fresh experience different from what we normally do for course project. I have gained many new knowledge and skills from this final year project. Therefore, I would like to express my special thanks to Professor Zheng for always inspiring me to work harder and learn more, and also praising and appreciating my productivity. However, it would not have been possible without support from his PhD student, Chen Haifen. Thus, my deep gratitude also goes to Haifen for being always there to help and give me guidance and for her great encouragement throughout the whole project period. Besides, I would like to thank everyone who gave me strength and support to accomplish this project, too.

# LIST OF FIGURES

# 1 PROJECT OVERVIEW

## 1.1 PROJECT OBJECTIVE

This project aims to integrate new features into a software application to analyze Gene Regulatory Networks (GRN) in order to understand the dynamics of cellular activities. Since most diseases are due to the dysfunction of GRN, this application could help to make mechanisms of human diseases more comprehensible, thereby assisting in research on new cure for diseases efficiently. With interactive Graphical User Interface (GUI) and rich data visualization, improved version of the application will clarify the complicated concepts and processes to ease the analysis of GRN.

## 1.2 PROJECT SCOPE

The existing application, GeneNetWeaver, is a useful tool which can easily generate gene expression datasets such as time series data for dynamic gene regulatory networks. Moreover, its powerful GUI also provides visualization of GRN to analyze the genes and interaction of those. However, it does not allow users to modify the structures of GRNs; and it does not provide the visualization of the time series data generated. Therefore, in this project, the application will be enhanced with new features which could improve users' convenience. New features such as modifying the structure of networks and visualization of time series data will be integrated into the current version.

# 2 GENENETWEAVER AT A GLANCE

With limited knowledge on bioinformatics, this project is definitely a challenge. Nevertheless, essential knowledge was acquired mainly from the source codes of GNW.

Since the basic understanding of source codes is crucial when it comes to integrating new functionalities into an existing application, the whole program was explored thoroughly. Therefore, much useful information such as how to create a sample network structure from scratch, how a network is represented as a graph, how to convert a static network to a dynamic network, and how datasets are generated from a dynamic network, were achieved for implementation tasks. In the following sections, those useful functions which help in improving the GNW will be explained in more details.

## 2.1 RUNNING GENENETWEAVER

To start running the application, the main class inside "ch.epfl.lis.gnwgui" package must be run. Section 6.2.1 shows the main method of the class executed when it starts running. Firstly, all the default settings are loaded and the preferences for the platform on which the application is running are set accordingly. Subsequently, the GUI is rendered. More details about the ***run*** method from **GnwGui** class will be introduced later on.

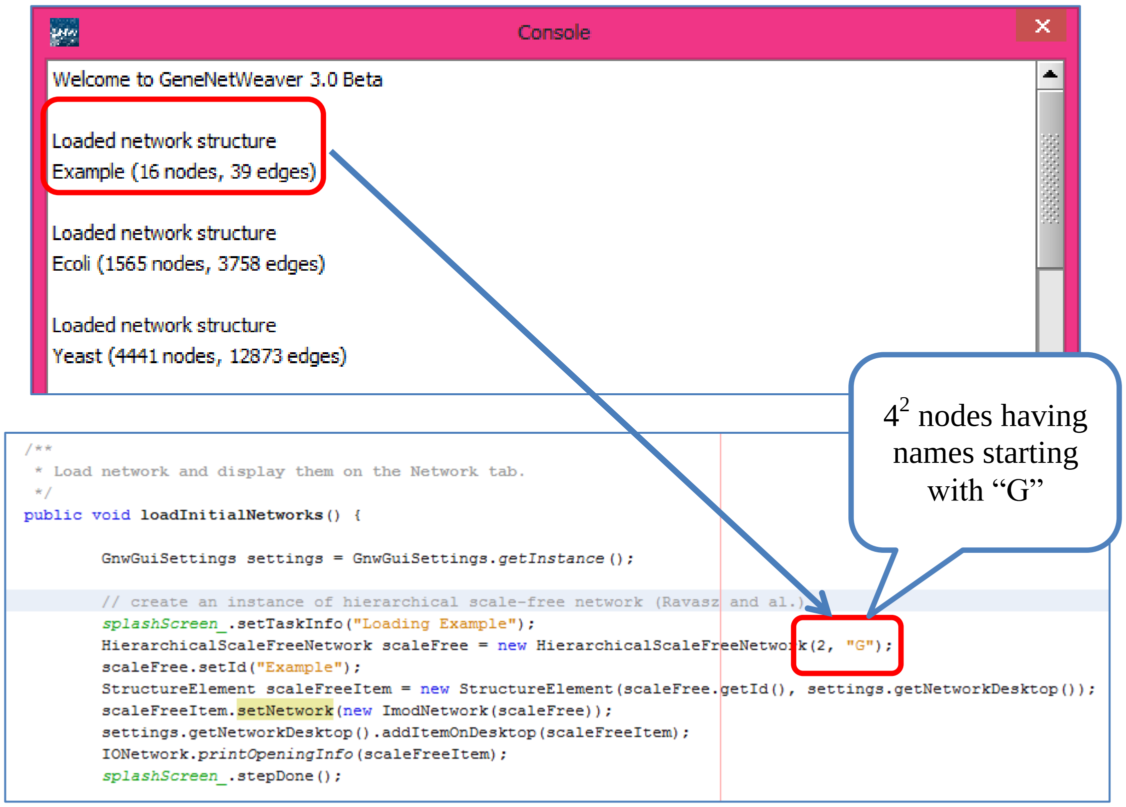
## 2.2 INITIALIZING THE GRAPHICAL USER INTERFACE

When the application is started, **GnwGui.*run*** method is executed in order to build the GUI. The splash screen is displayed first while loading the default networks. After all networks have been completely loaded, the main interface is displayed after the splash screen disappears.

As shown in Section 6.2.2, there are two methods for loading default networks: ***loadInitialNetworks*** and ***loadDREAMNetworks***. The former loads a sample network of specified size and predefined networks from **GnwGuiSettings** while the latter loads all DREAM networks. How a sample network has been constructed will be explored in details in the later section.

## 2.3 CREATING A SAMPLE NETWORK

How to create a sample Network can be seen in ***loadInitialNetworks*** method from **GnwGui** class. As shown below, a new hierarchical scale-free network is created, stating the number of nodes in exponent of 4 and the prefix for nodes. Then, a structure element for that network is defined to add an icon to represent the network onto the GNW desktop.



```
Welcome to GeneNetWeaver 3.0 Beta

Loaded network structure
Example (16 nodes, 39 edges)

Loaded network structure
Ecoli (1565 nodes, 3758 edges)

Loaded network structure
Yeast (4441 nodes, 12873 edges)
```

```
/**
 * Load network and display them on the Network tab.
 */
public void loadInitialNetworks() {

        GnwGuiSettings settings = GnwGuiSettings.getInstance();

        // create an instance of hierarchical scale-free network (Ravasz and al.)
        splashScreen_.setTaskInfo("Loading Example");
        HierarchicalScaleFreeNetwork scaleFree = new HierarchicalScaleFreeNetwork(2, "G");
        scaleFree.setId("Example");
        StructureElement scaleFreeItem = new StructureElement(scaleFree.getId(), settings.getNetworkDesktop());
        scaleFreeItem.setNetwork(new ImodNetwork(scaleFree));
        settings.getNetworkDesktop().addItemOnDesktop(scaleFreeItem);
        IONetwork.printOpeningInfo(scaleFreeItem);
        splashScreen_.stepDone();
```

Figure 2.3 – Creating a Sample Network

## 2.4 OPTIONS AVAILABLE FOR A NETWORK

When the **StructureElement** networks' icons are double-clicked or right-clicked, the options available for those networks will appear. Normally, static networks have fewer options compared to kinetic networks. In the following figure, the blue icons represent the static networks.

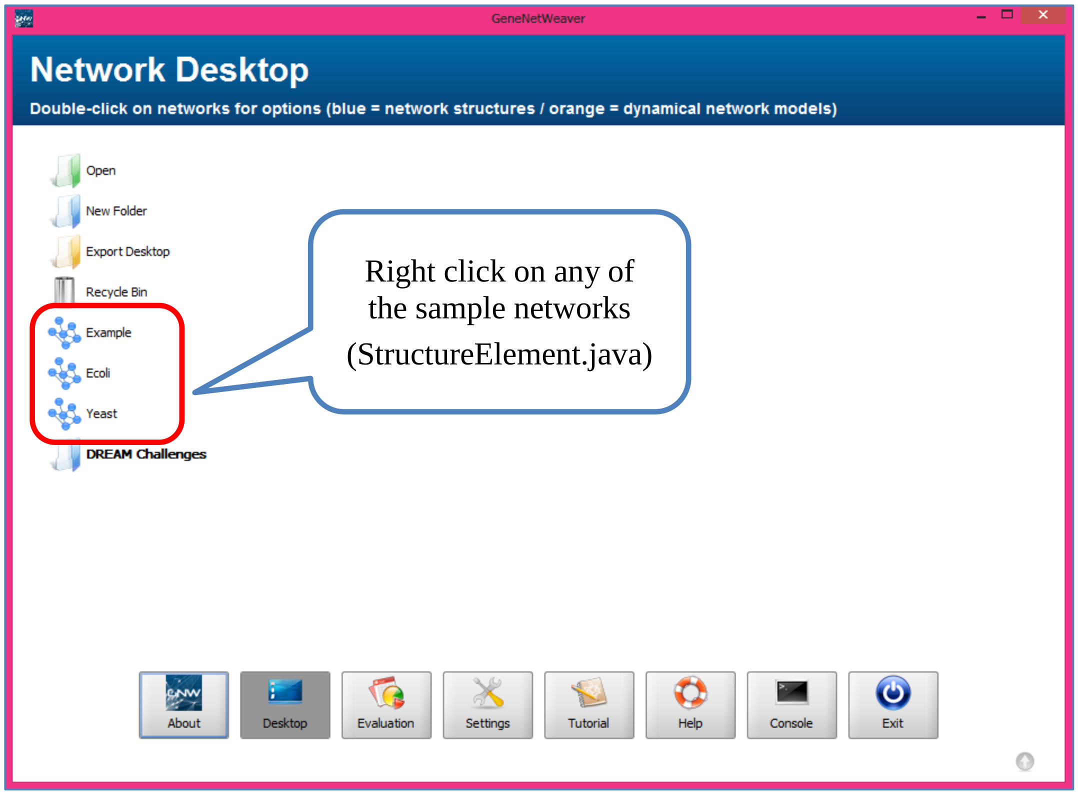

GeneNetWeaver
Network Desktop
Double-click on networks for options (blue = network structures / orange = dynamical network models)
Open
New Folder
Export Desktop
Recycle Bin
Example
Ecoli
Yeast
DREAM Challenges
Right click on any of the sample networks (StructureElement.java)
About
Desktop
Evaluation
Settings
Tutorial
Help
Console
Exit

Figure 2.4.1 – GeneNetWeaver Desktop

The following is the options for a static network. Every single button of this window is defined in **OptionsWindow** class while the action to perform when each of them is clicked is implemented in **Options** class.

Hence, in order to add a new option for newly developed feature, the button must be defined in **OptionsWindow** class first. Then, the function handling the button click must be stated inside ***setActions*** method of **Options** class; and, it should also be implemented in **Options** class.

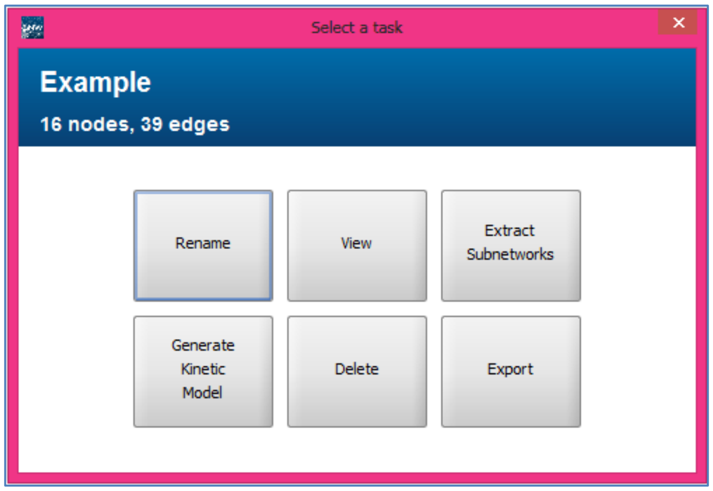


Figure 2.4.2 – Options Window for a Static Network

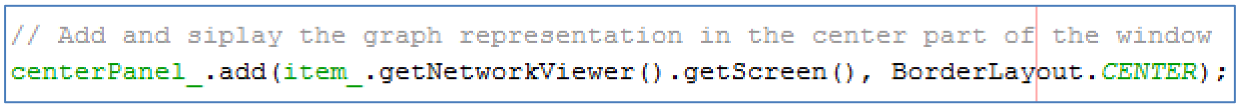
## 2.5 GRAPH REPRESENTATION OF A NETWORK

When **View** button from Figure 2.4.2 is clicked, a graph defining the structure of the network is displayed. The whole procedure from clicking button to displaying graph will be explained step by step.

In ***setActions*** method of **Options** class, how the button click will be handled is defined.

The ***viewNetwork*** method from **Options** class first checks if the network is large, i.e. the network is composed of more than 200 nodes. In that case, a message prompting that it can take longer to load the whole network for visualization will appear. Then, a new graph viewer will be created to display the graph.

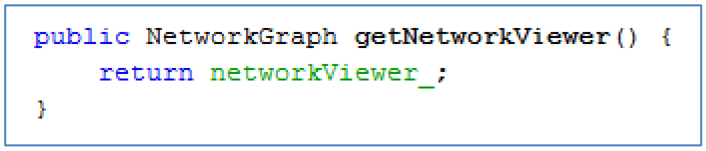
When a graph viewer is initialized, visualization controls are added on the left of the window and the graph in the center. The graph is displayed by using ***getScreen*** function of the network viewer. The object type of network viewer is discovered by navigating to its source code.



```
// Add and siplay the graph representation in the center part of the window
centerPanel_.add(item_.getNetworkViewer().getScreen(), BorderLayout.CENTER);
```

Figure 2.5.1 – Constructor of **GraphViewer**

As shown in the following piece of code, the network viewer is in fact a **NetworkGraph** object.



```
public NetworkGraph getNetworkViewer() {
    return networkViewer_;
}
```

Figure 2.5.2 – ***getNetworkViewer*** function of **NetworkElement**

When a **NetworkGraph** is initialized as shown in Section 6.2.5, the type of the network is identified first. Finally, the structure of the network is computed to display the graph using ***computeGraph*** method.

In this method, nodes and edges of the network are added first. Next, the layout of the graph is set up. In some cases, the positions of the nodes are predefined in the structure of the network. Then, the graph mouse is also added to the graph panel.

Finally, this is the resulted graph from the above procedure.

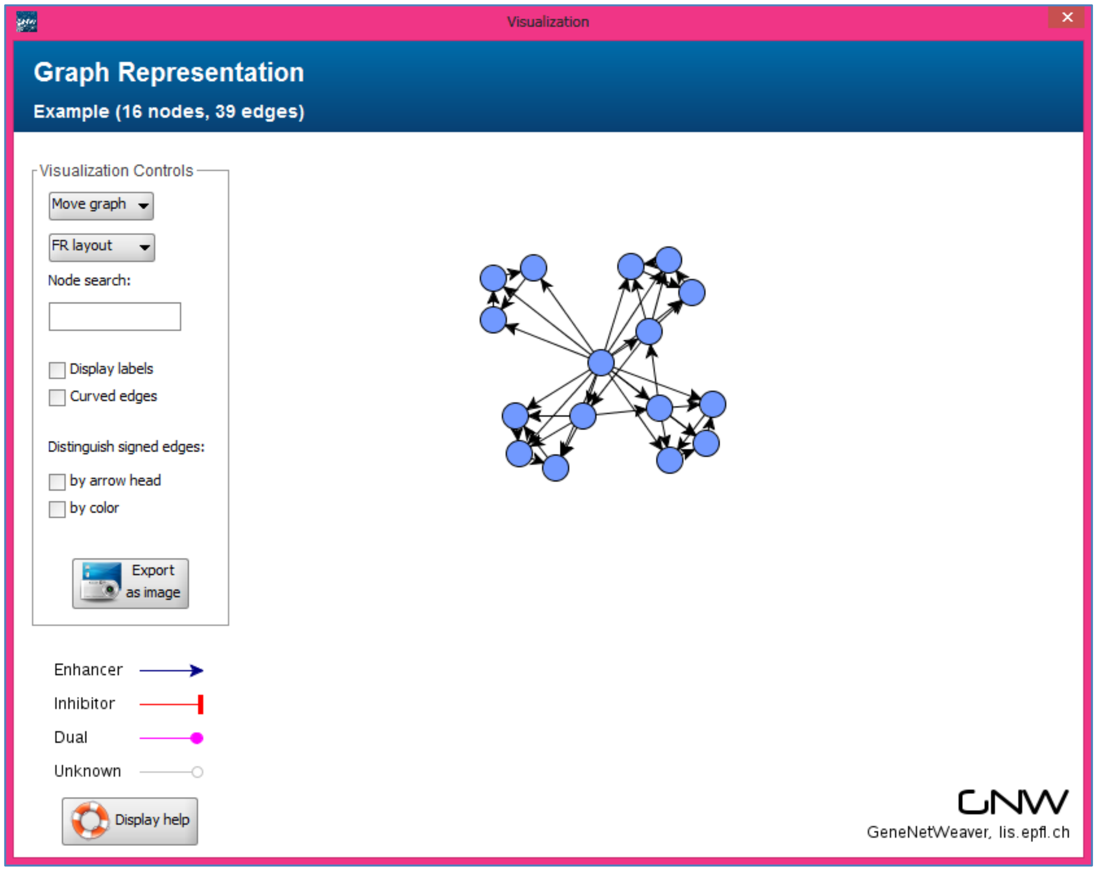


Figure 2.5.3 – Graph Representation of Example network

## 2.6 GETTING SELECTED NODES FROM A NETWORK GRAPH

The ***pickedState_*** object holds the set of nodes which are picked from the graph. Later, this part will be useful for implementing "Modifying Network Structure" feature.

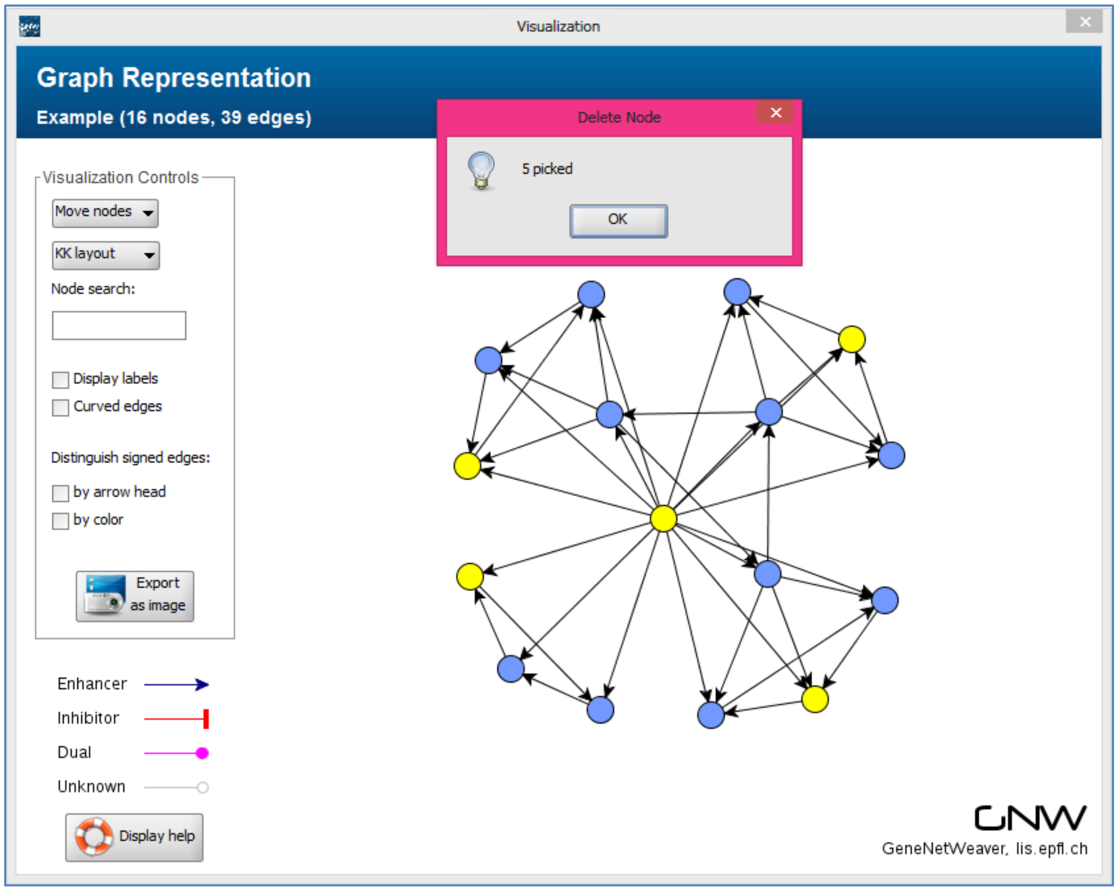


Figure 2.6.1 – Graph Representation of Example network

```
Set<Node> picked = pickedState_.getPicked();

JOptionPane.showMessageDialog(null,
     picked.size() + " picked",
     "Delete Node", JOptionPane.INFORMATION_MESSAGE);
```

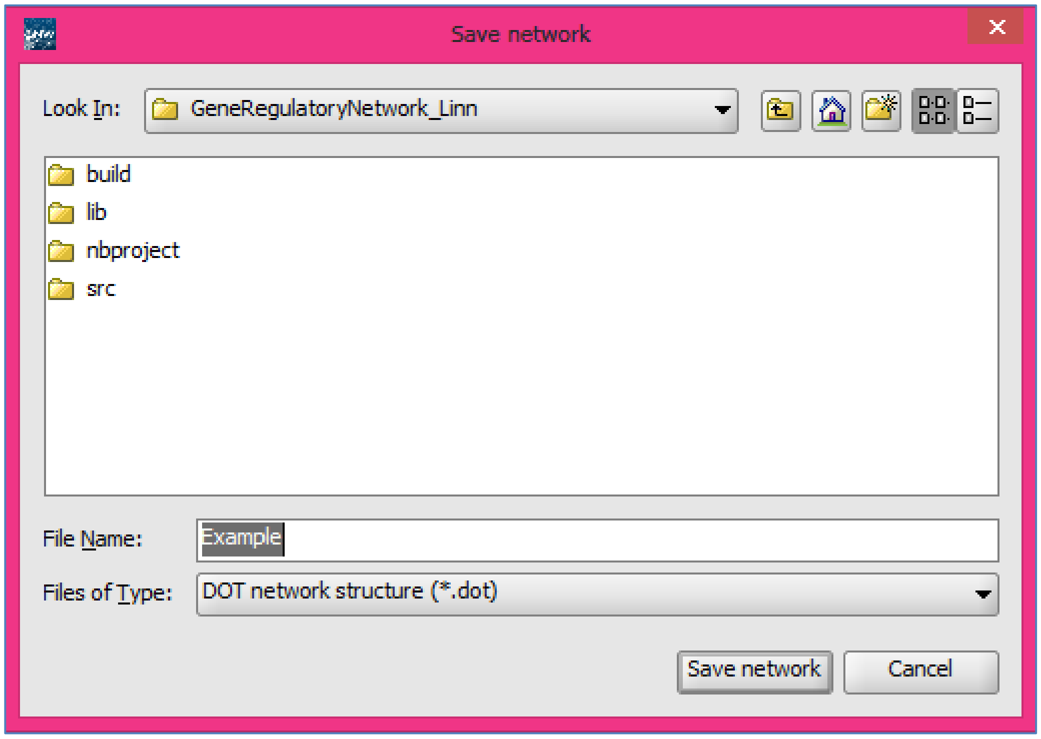
Figure 2.6.2 – Set of Picked Nodes

## 2.7 EXPORTING A NETWORK

When **Export** button from Figure 2.4.2 is clicked, the network can be saved as a DOT file which is a network structure file format.

As shown in Section 6.2.3, ***exportNetwork*** method handles the Export button click. In addition to exporting network, folders can also be exported. Options window will be closed and replaced with Save Network window.

The following figure shows Save Network window where the network can be exported as a DOT file which can later be opened using GNW again.



File Name: Example

Files of Type: DOT network structure (*.dot)

Figure 2.7 – Save Network Dialog

## 2.8 CONVERTING A STATIC NETWORK TO A DYNAMIC ONE

If **Generate Kinetic Model** button from Figure 2.4.2 is clicked, the static network will be transformed into a dynamic one. The transformation process is handled by ***conversion2DynamicalModel*** method as shown in Section 6.2.3.

A message will firstly prompt users, asking if users agree to remove auto-regulatory interactions from the current network structure. Upon users' agreement, those existing auto-regulatory interactions are removed. Subsequently, the blue icon on the GNW desktop will also change into a yellow icon which represents the kinetic model.

## 2.9 GENERATING DATASETS FROM A DYNAMIC NETWORK

This option is only available for dynamic networks as compared to the options window for static networks. When **Generate Datasets** button is clicked, a window is displayed to let users define the time series data, the number of time series, the duration of each time series, the location to save the time series data and so on.

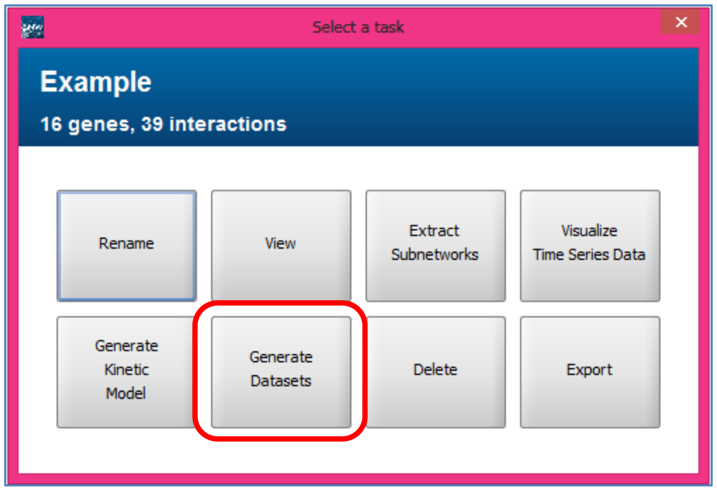
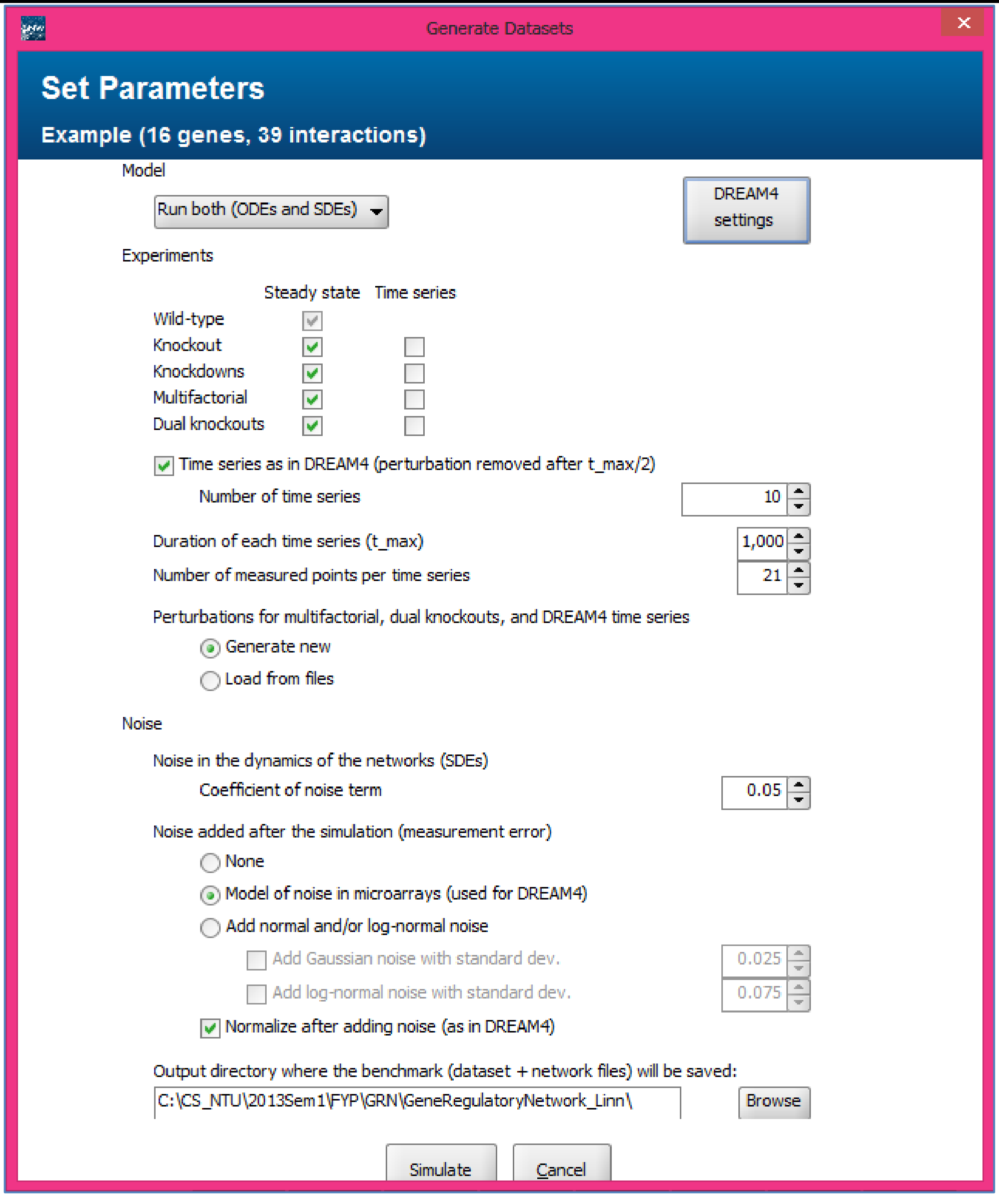


Figure 2.9.1 – Options Window for a Dynamic Network

As shown in Section 6.2.3, the options window is closed first and replaced by the parameters window instantiated in ***generateDREAM3GoldStandard*** method, where different settings mentioned above can be configured.



Model: Run both (ODEs and SDEs)

Experiments

| | Steady state | Time series |
|---|---|---|
| Wild-type | ✓ | |
| Knockout | ✓ | ☐ |
| Knockdowns | ✓ | ☐ |
| Multifactorial | ✓ | ☐ |
| Dual knockouts | ✓ | ☐ |

[✓] Time series as in DREAM4 (perturbation removed after t_max/2)

Number of time series: 10

Duration of each time series (t_max): 1,000

Number of measured points per time series: 21

Perturbations for multifactorial, dual knockouts, and DREAM4 time series: [✓ Generate new] Load from files

Noise

Noise in the dynamics of the networks (SDEs)

Coefficient of noise term: 0.05

Noise added after the simulation (measurement error): None [✓ Model of noise in microarrays (used for DREAM4)] Add normal and/or log-normal noise

[ ] Add Gaussian noise with standard dev. 0.025

[ ] Add log-normal noise with standard dev. 0.075

[✓] Normalize after adding noise (as in DREAM4)

Output directory where the benchmark (dataset + network files) will be saved: C:\CS_NTU\2013Sem1\FYP\GRN\GeneRegulatoryNetwork_Linn\

Figure 2.9.2 – Parameters Window for Generating Datasets

# 3 NEW FEATURES INTEGRATED INTO GENENETWEAVER

Additional features are integrated into GeneNetWeaver for better convenience of users and clearer information. Features added includes visualization of time series data generated from dynamic networks, editing network structure and comparison of time series data between different versions of dynamic networks.

## 3.1 EDITING NETWORK STRUCTURE

This new feature is the improvement upon the current feature of viewing network structure. It allows user to remove existing nodes and edges, and also add new edges. In this case, the terms “node” and “edge” are used since this feature is exclusively for static network structure.

In network graph viewer, there are two mouse modes: one for moving graph and another for moving nodes and edges. In the following figure, Move graph is using the first mode and the rest are using the second one. When the mode to move nodes and edges is chosen, the selected nodes will be portrayed in yellow and the picked edges, in green.

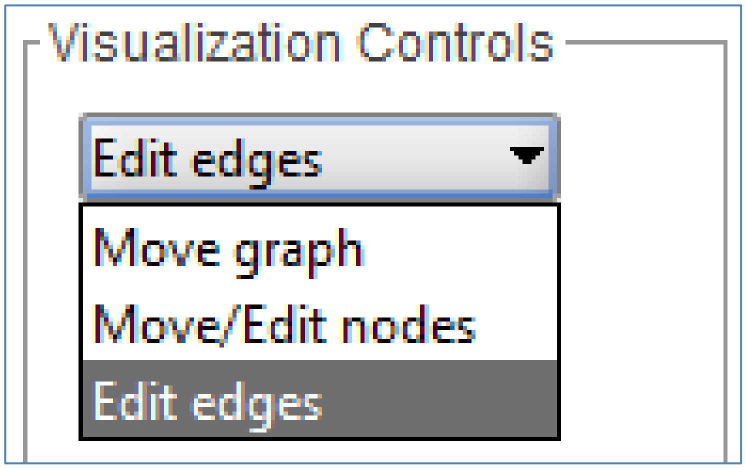


Figure 3.1 – Available Options from Network Graph Viewer

### 3.1.1 Deleting Nodes

**Move/Edit nodes** option should be chosen first in order to pick some nodes to delete. Then, when the nodes are picked by clicking, the color of the nodes changes to yellow so that they can be differentiated from those which are not selected.

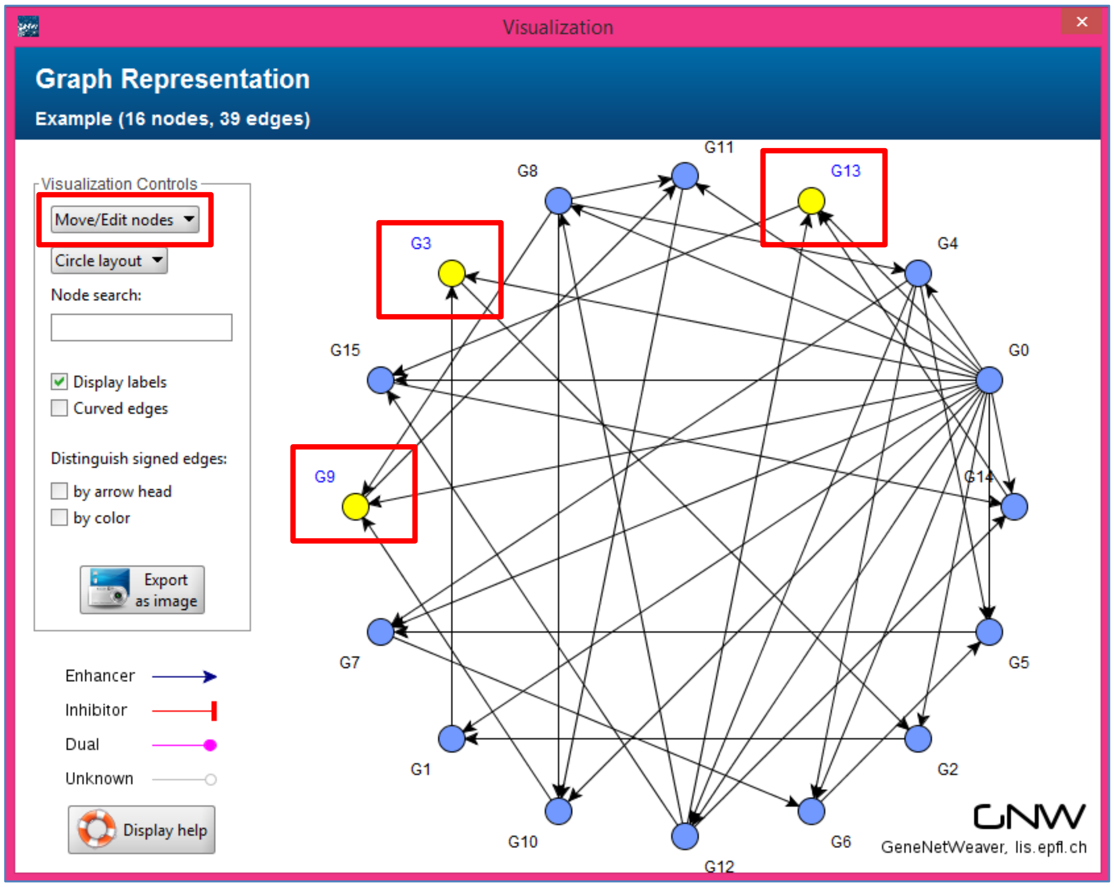
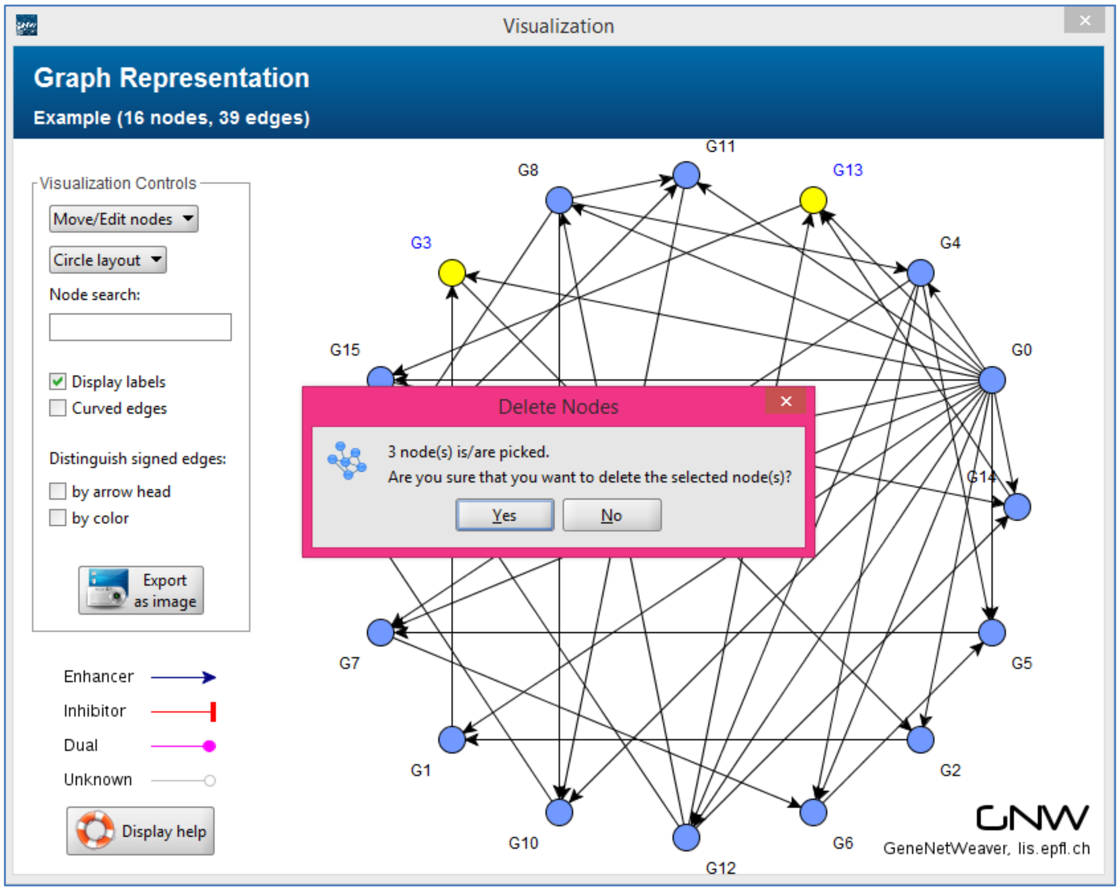


Figure 3.1.1.1 – Selecting Nodes from Structure of Network

While there are some nodes selected, by pressing Ctrl+D, those selected nodes will be deleted. Similarly, the edges between the deleted nodes will also be removed subsequently after user's confirmation to delete.

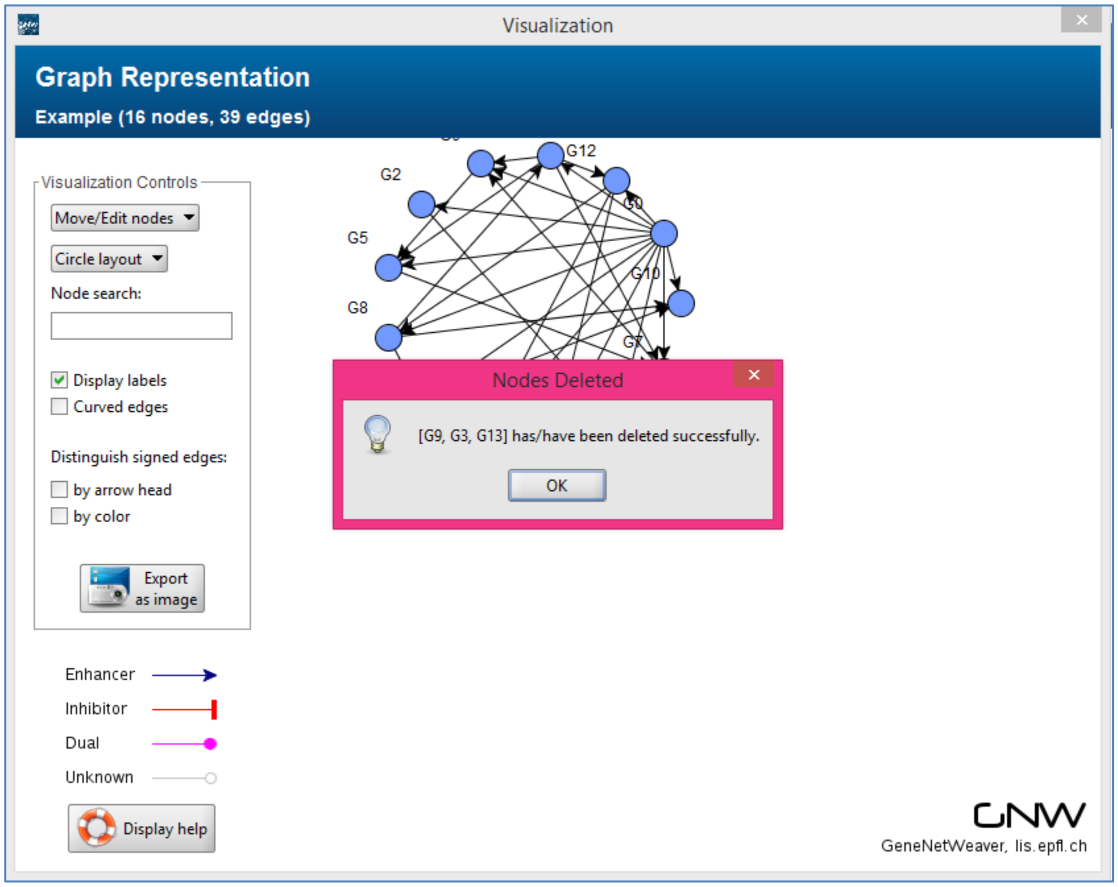


3 node(s) is/are picked.
Are you sure that you want to delete the selected node(s)?

Figure 3.1.1.2 – Deleting Selected Nodes from Structure of Network

When confirmed, the system will check whether the current network has been modified previously or not; if this deletion is the first modification made to the structure of the current network. If it has not been modified even once yet, the original structure will be saved temporarily. And then, the changes will be applied to the current structure so that the visualization will reflect the changes concurrently.

After deleting the chosen nodes successfully, the following message is displayed to inform user.



[G9, G3, G13] has/have been deleted successfully.

Figure 3.1.1.3 – Message after Successful Deletion

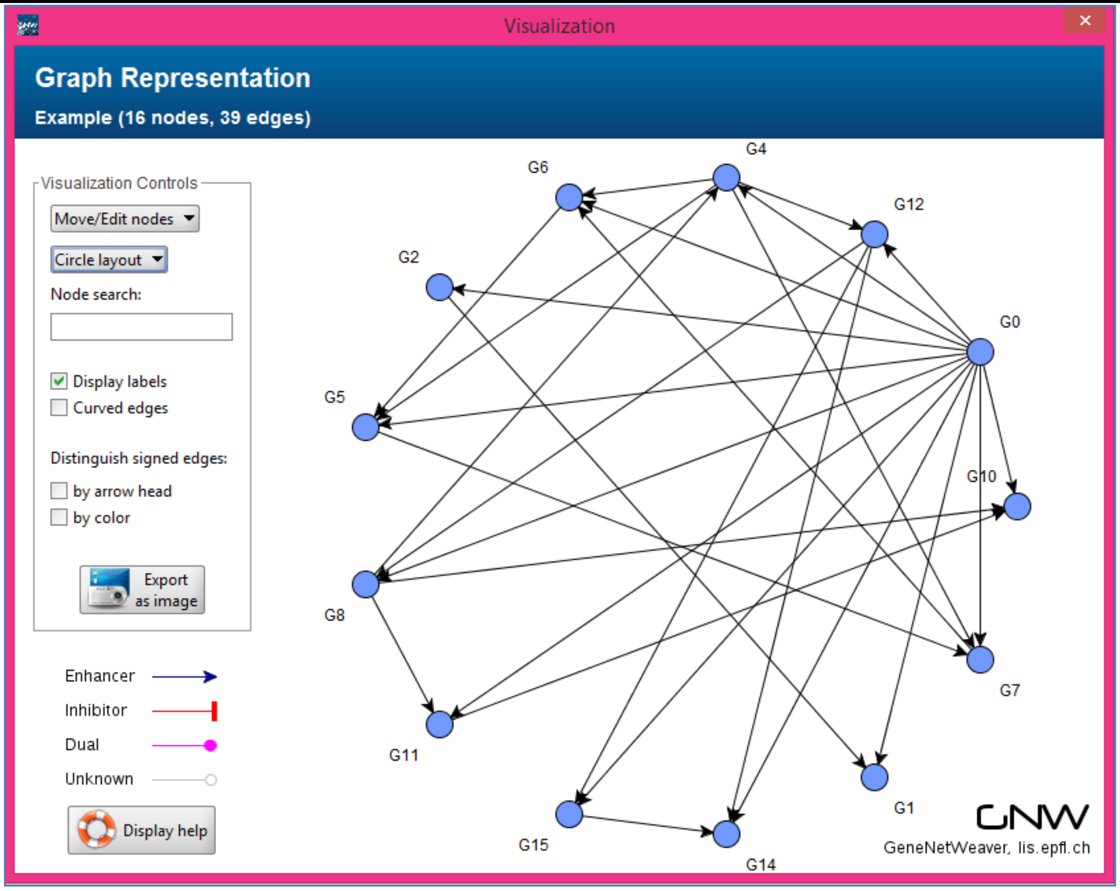


Figure 3.1.1.4 – Updated Structure of Network after Deletion

### 3.1.2 Deleting Edges

Existing edges can also be deleted like nodes. To delete the edges, **Edit edges** option need to be selected first. Then, after selecting one or more edges, the selected edges can be deleted by entering Ctrl+E keys.

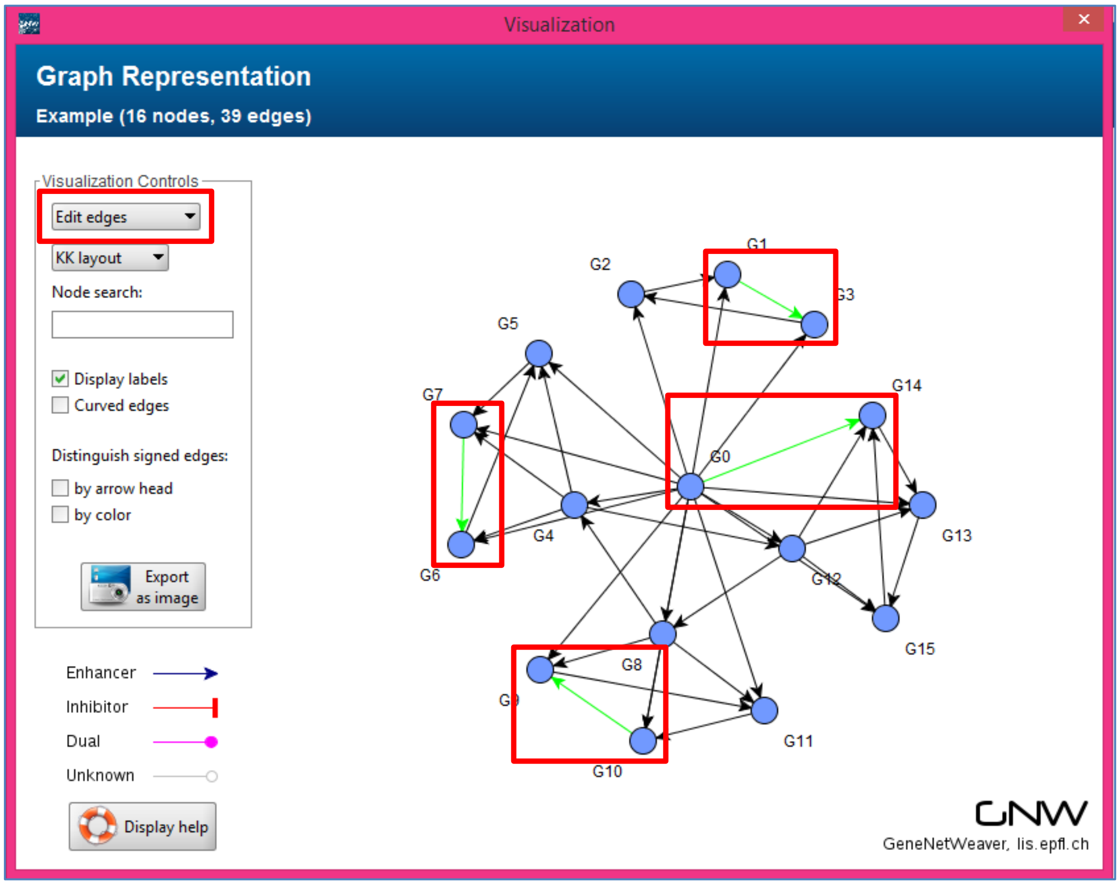
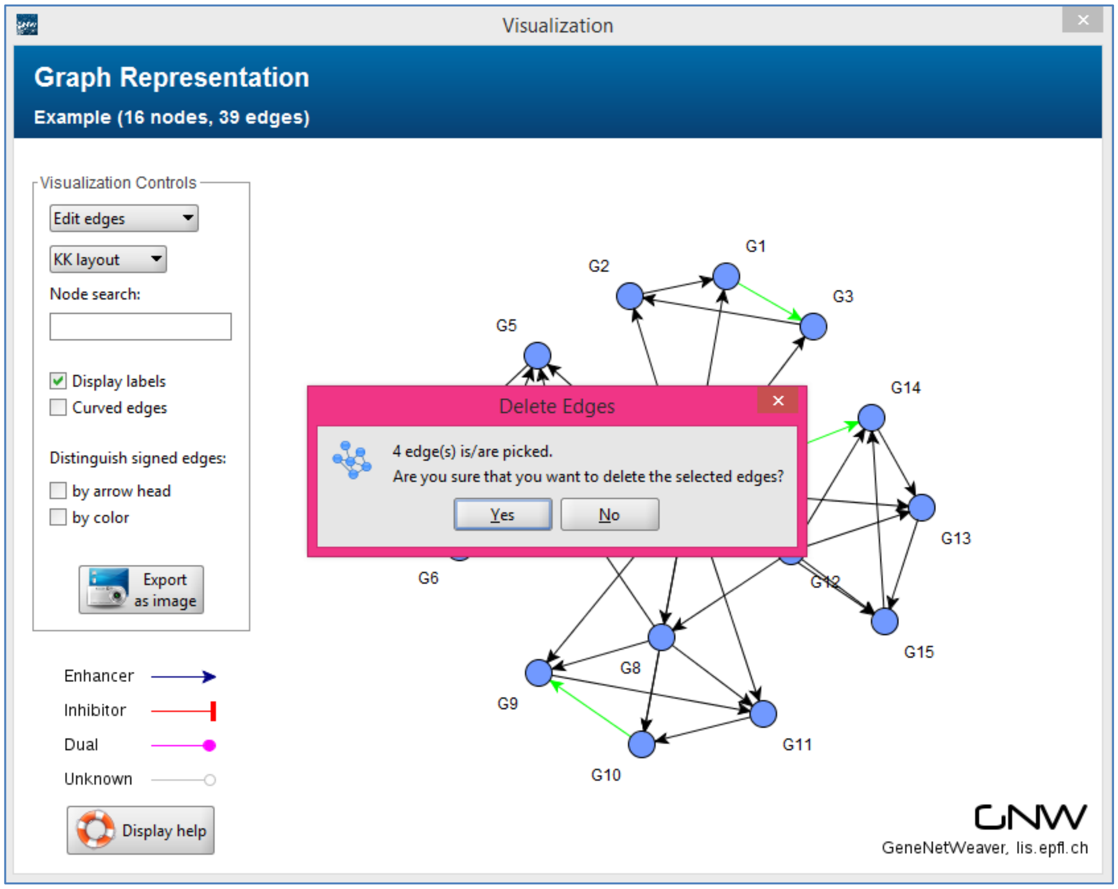


Figure 3.1.2.1 – Selecting Edges from Structure of Network

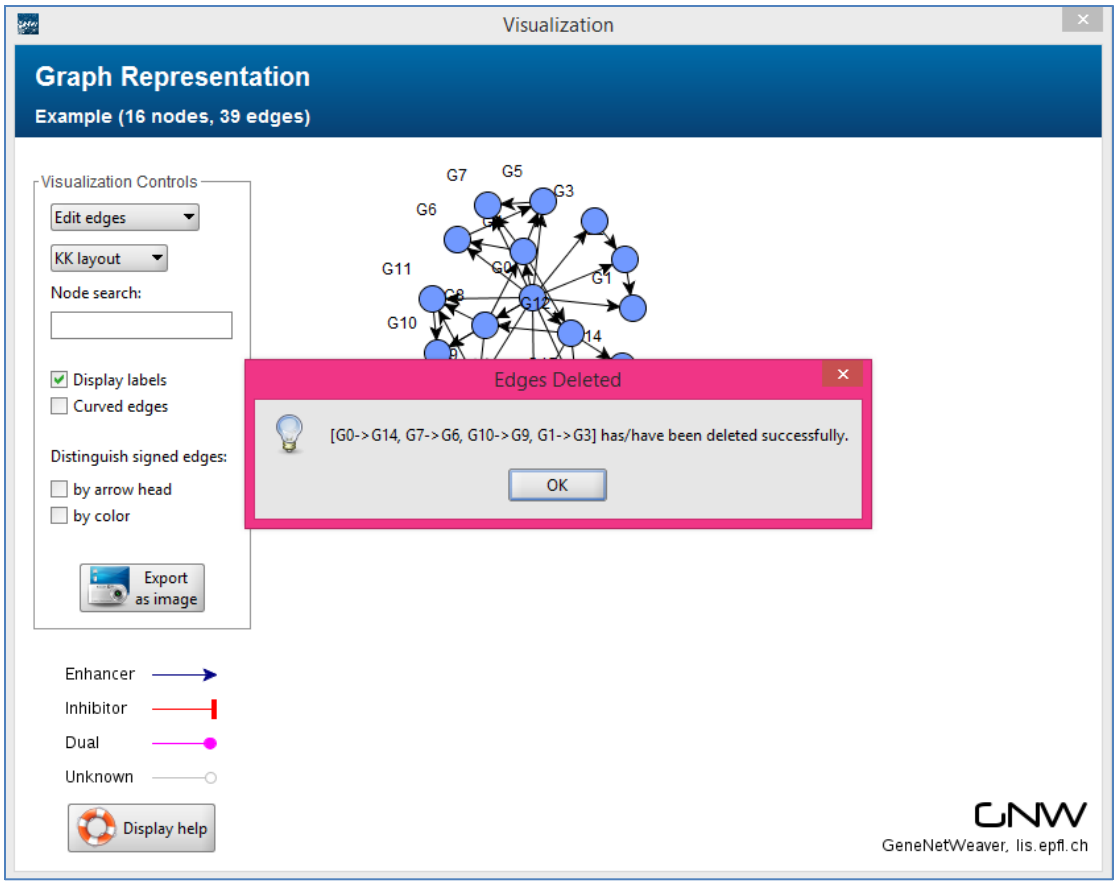
After that, the system will prompt user to confirm the deletion of the selected edges as follow.



4 edge(s) is/are picked.
Are you sure that you want to delete the selected edges?

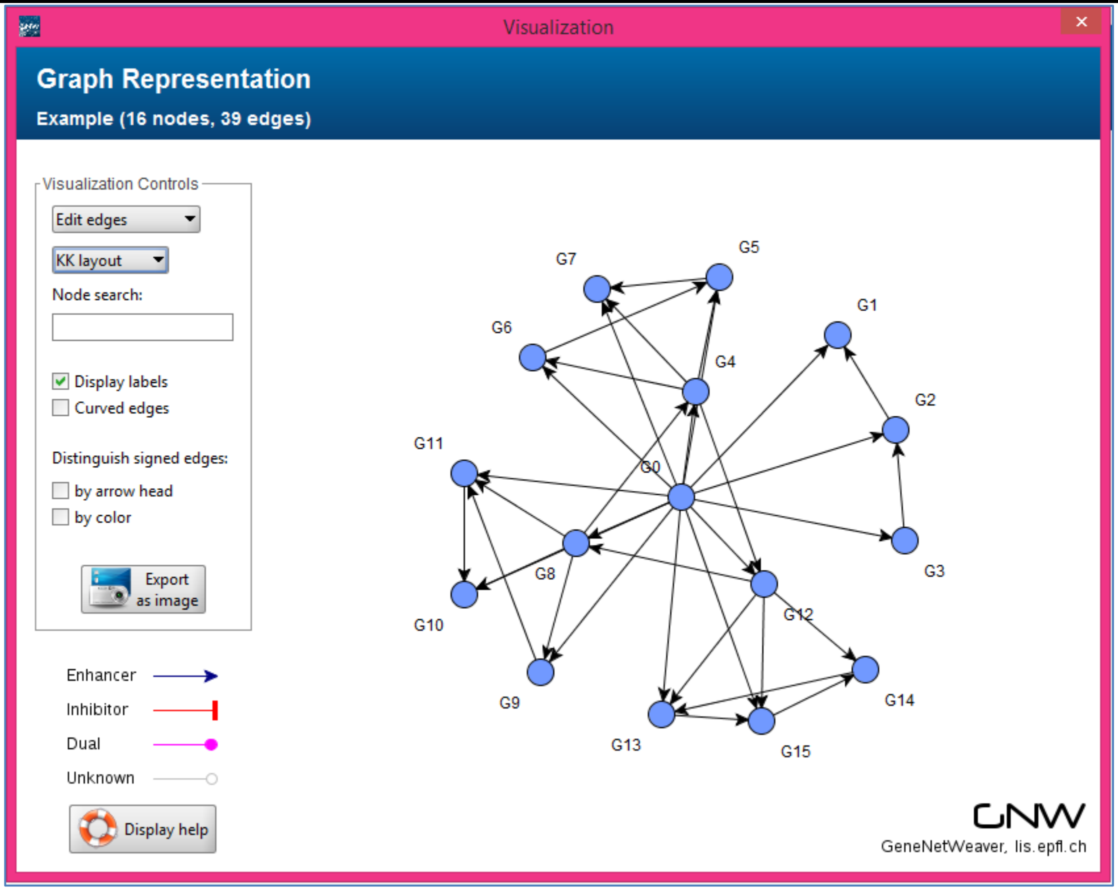
Figure 3.1.2.2 – Deleting Selected Edges from Structure of Network

Upon confirmation, the same procedure mentioned in Section 4.1.1 will be carried out. After deleting the chosen edges successfully, the following message is displayed to inform user.



[G0->G14, G7->G6, G10->G9, G1->G3] has/have been deleted successfully.

Figure 3.1.2.3 – Message after Successful Deletion


Visualization

Graph Representation

Example (16 nodes, 39 edges)

Visualization Controls

Edit edges

KK layout

Node search:

[✓] Display labels

[ ] Curved edges

Distinguish signed edges:

[ ] by arrow head

[ ] by color

Figure 3.1.2.4 – Updated Structure of Network after Deletion

### 3.1.3 Adding a New Edge

By selecting **Edit edges** option, user can also add new edges. In order to create a new edge, exactly two nodes to be connected together need to be chosen first. If selected nodes are either more than two or less than two, an error message will be displayed prompting user to select two nodes only.

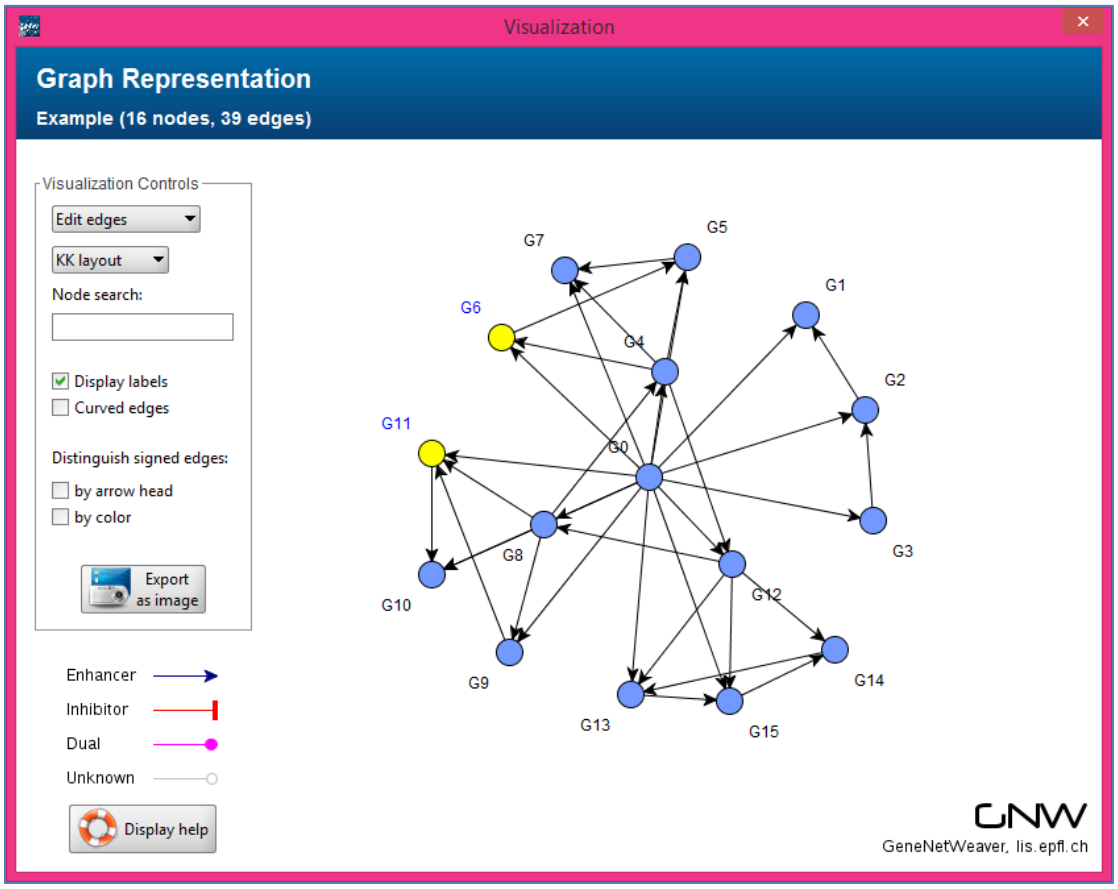
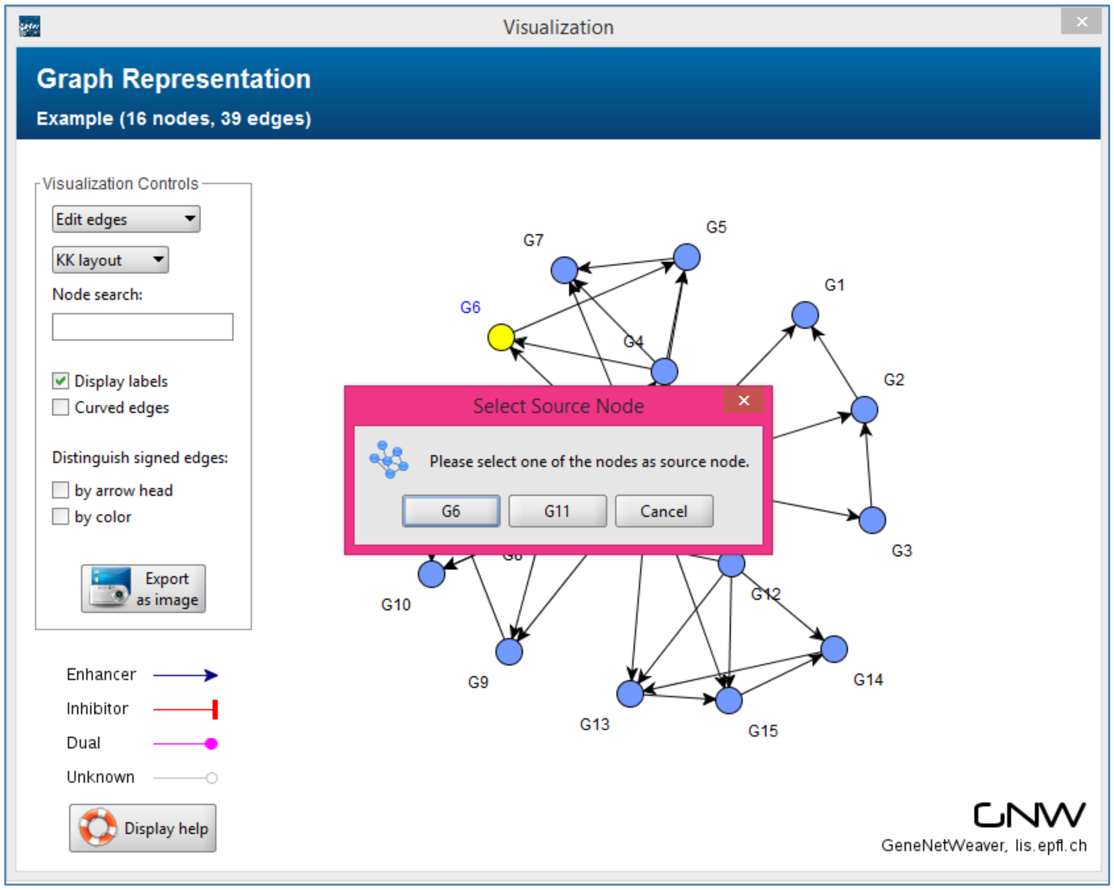


Figure 3.1.3.1 – Selecting Two Nodes from Structure of Network

After selecting two nodes, when user enters Ctrl+N, the system checks if there exists any edge between these two nodes. If so, a message will be displayed. Otherwise, user is prompted to choose one of them as the source node as follow.

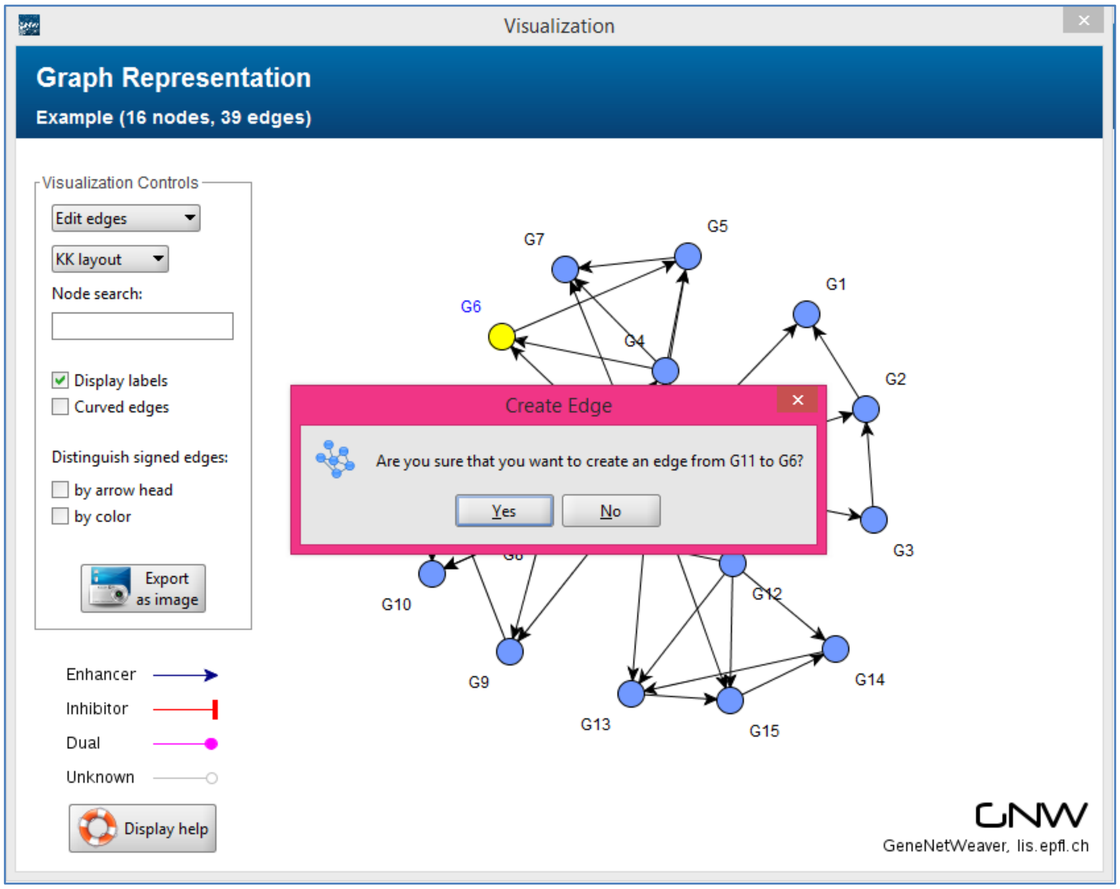


☑ Display labels
☐ Curved edges

Please select one of the nodes as source node.

Figure 3.1.3.2 – Selecting Source Node for the Edge to Create

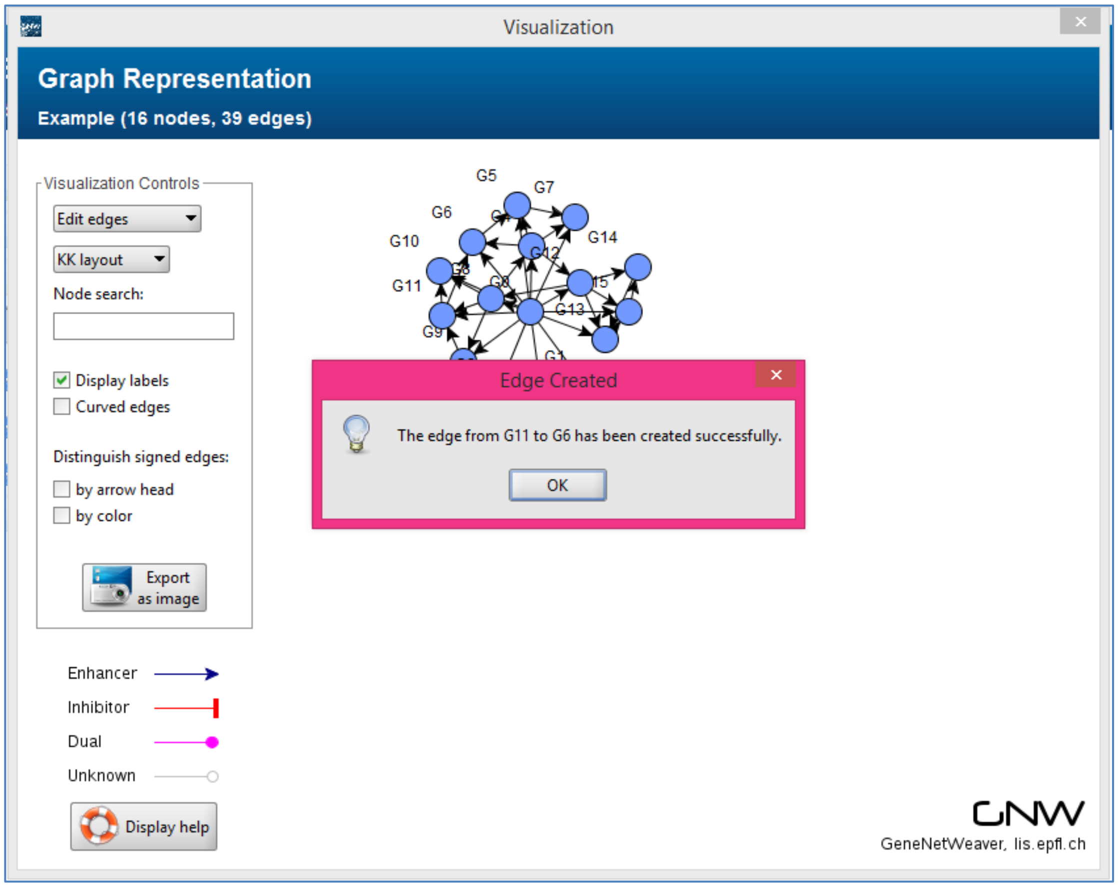
Afterward, the confirmation from the user is requested to create the specified edge.



Are you sure that you want to create an edge from G11 to G6?

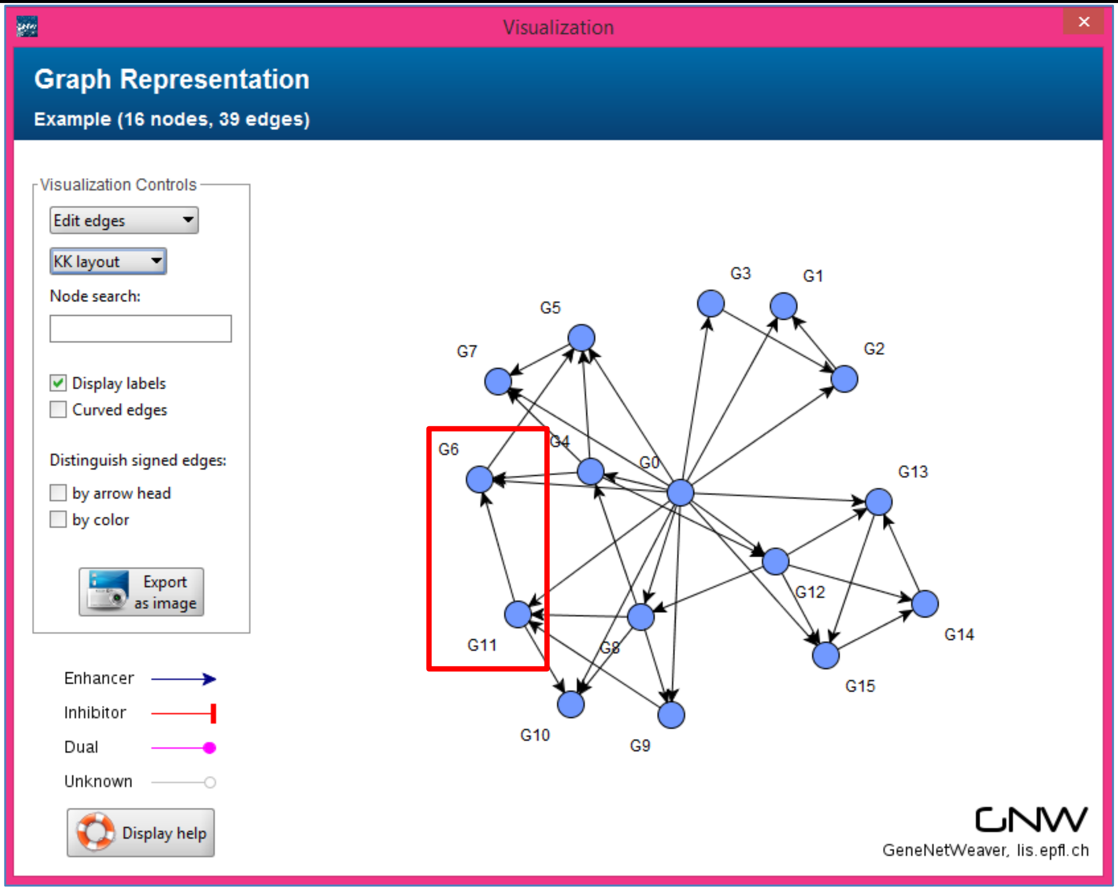
Figure 3.1.3.3 – Creating a New Edge between Two Selected Nodes

When user confirms, the specified edge is added to the structure of current network. A message saying that the edge is created successfully is displayed and the edge can then be found in the graph.



The edge from G11 to G6 has been created successfully.

Figure 3.1.3.4 – Message after Successful Creation


Visualization

Graph Representation

Example (16 nodes, 39 edges)

Visualization Controls

Edit edges

KK layout

Node search:

[✓] Display labels

☐ Curved edges

Distinguish signed edges:

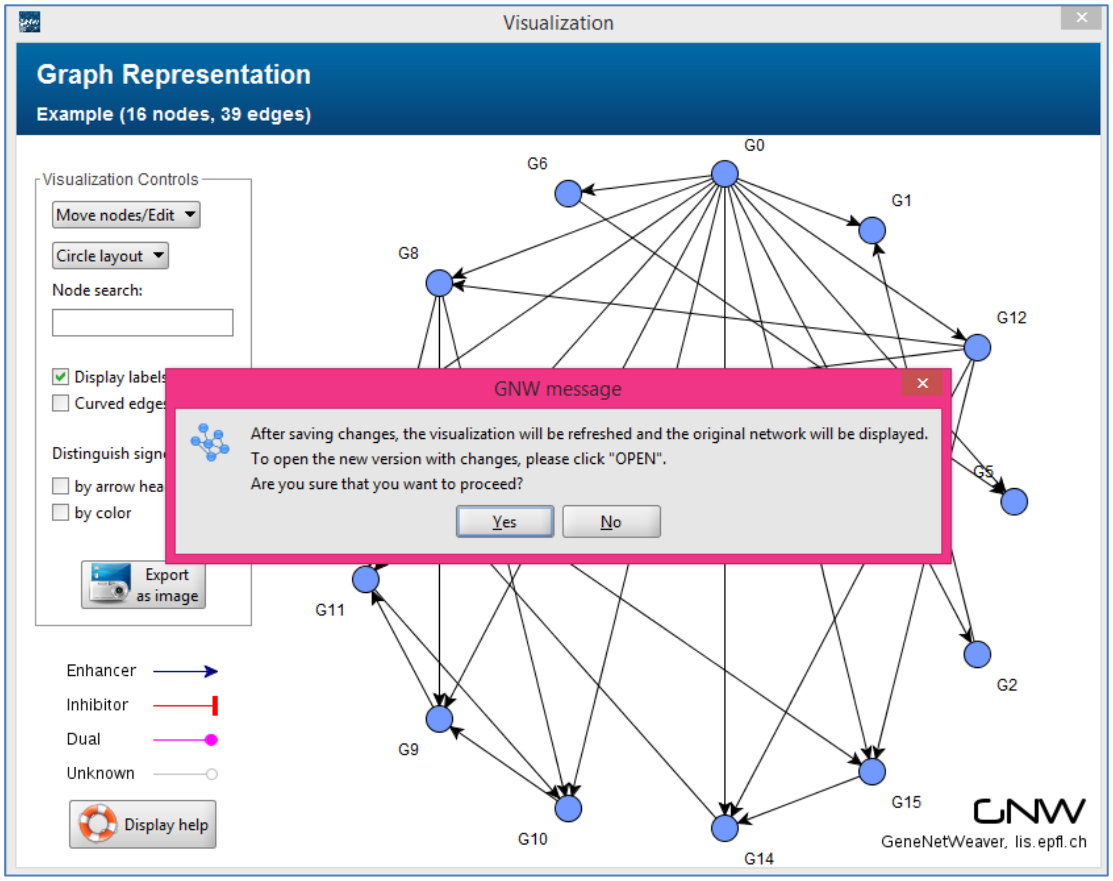
☐ by arrow head

☐ by color

Figure 3.1.3.5 – Updated Structure of Network after Deletion

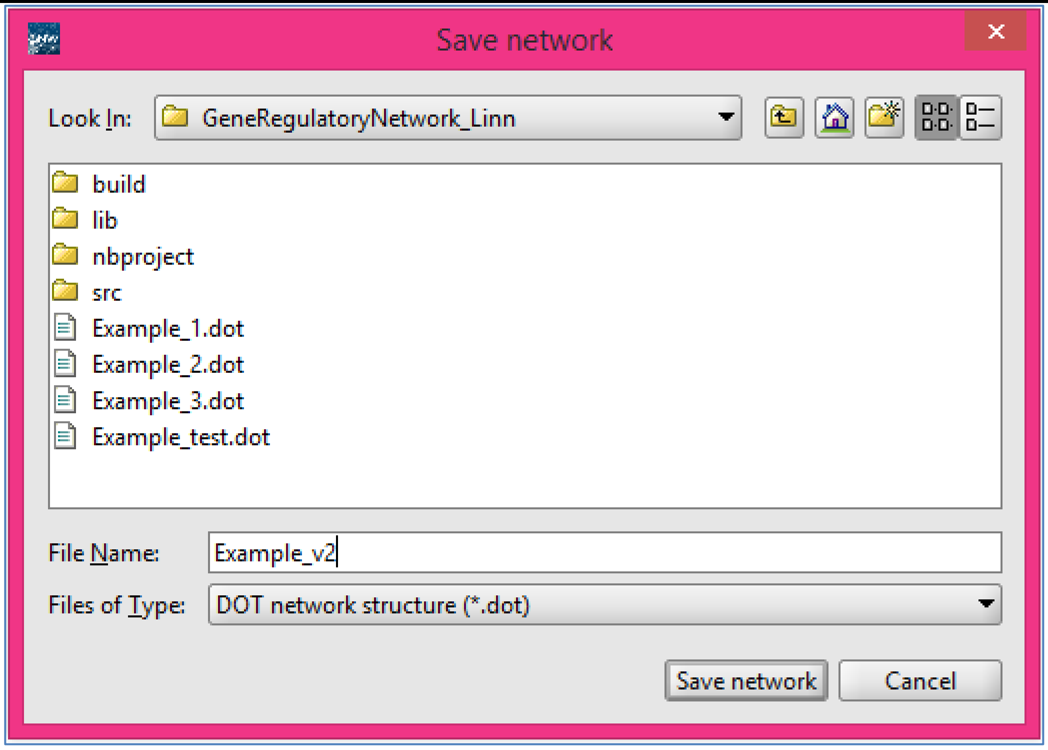
### 3.1.4 Saving Changes

While editing the structure of the current network, the changes will be reflected instantaneously. After modifying the structure, the changes can be saved by pressing Ctrl+S. A dialog to select the location to save the modified network will appear. After saving successfully, the graph viewer will be refreshed and display the original structure of the current network. To visualize the modified version, there is the icon Open on the GNW desktop which allows user to import any network into GNW.


GNW message

After saving changes, the visualization will be refreshed and the original network will be displayed.
To open the new version with changes, please click "OPEN".
Are you sure that you want to proceed?

Figure 3.1.4.1 – Message before Saving Modified Structure of Network


Save network

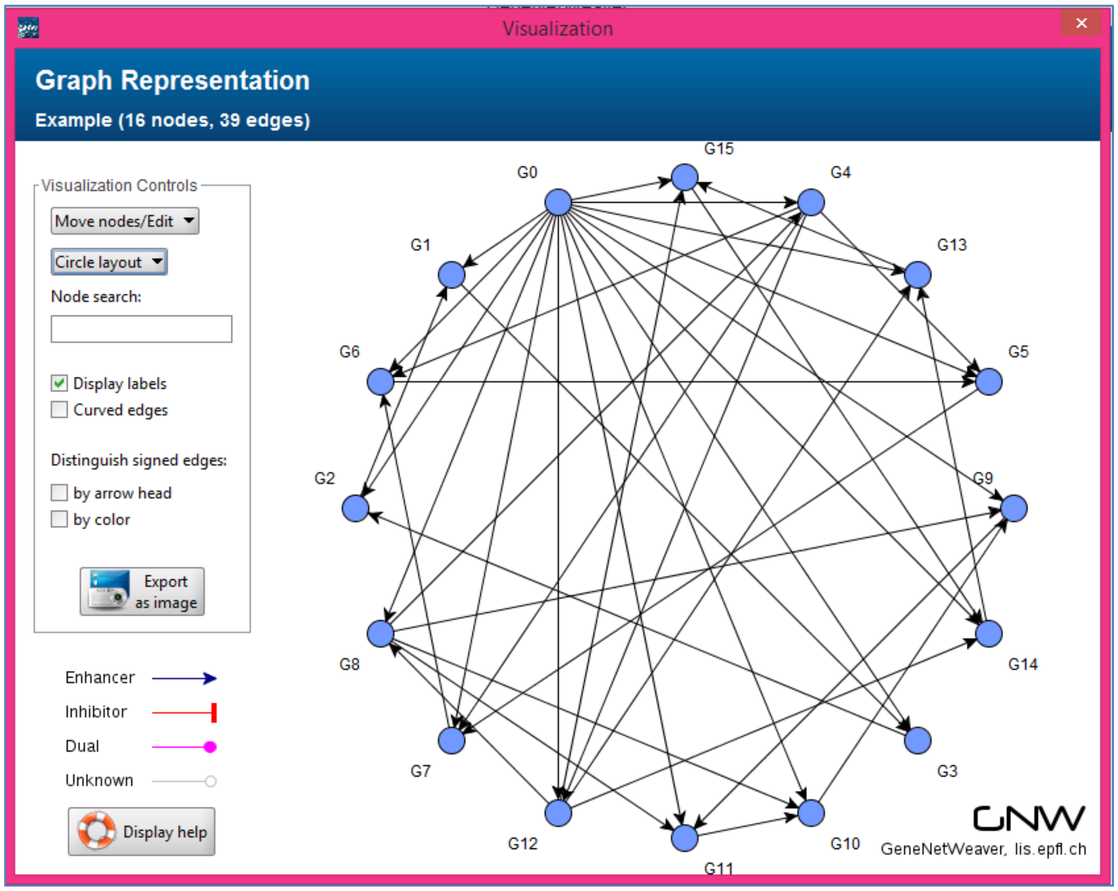
Look In: GeneRegulatoryNetwork_Linn

build
lib
nbproject
src
Example_1.dot
Example_2.dot
Example_3.dot
Example_test.dot

File Name: Example_v2

Files of Type: DOT network structure (*.dot)

Figure 3.1.4.2 – Saving Modified Structure of Network


Visualization

Graph Representation

Example (16 nodes, 39 edges)

Visualization Controls

Move nodes/Edit

Circle layout

Node search:

[✓] Display labels

[ ] Curved edges

Distinguish signed edges:

[ ] by arrow head

[ ] by color

Figure 3.1.4.3 – Original Structure of Network after Successfully Saving Changes

## 3.2 TIME SERIES VISUALIZATION

A new button, **Visualize Time Series Data**, is added for all dynamic networks. When the button is clicked, the system asks user to specify the directory where the generated dataset exists. Then, it checks if there really exists dataset generated for the current network in the specified directory. If not found, a proper error message prompting to generate dataset first will be displayed. Otherwise, the default time series graph of that network will be shown.

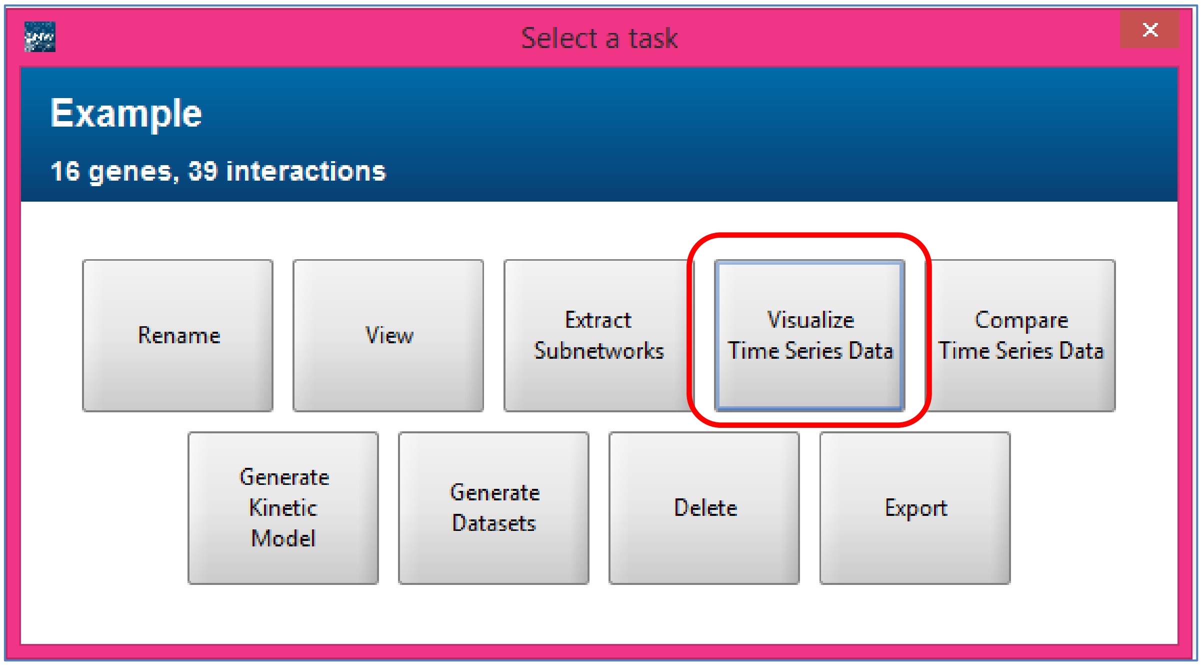
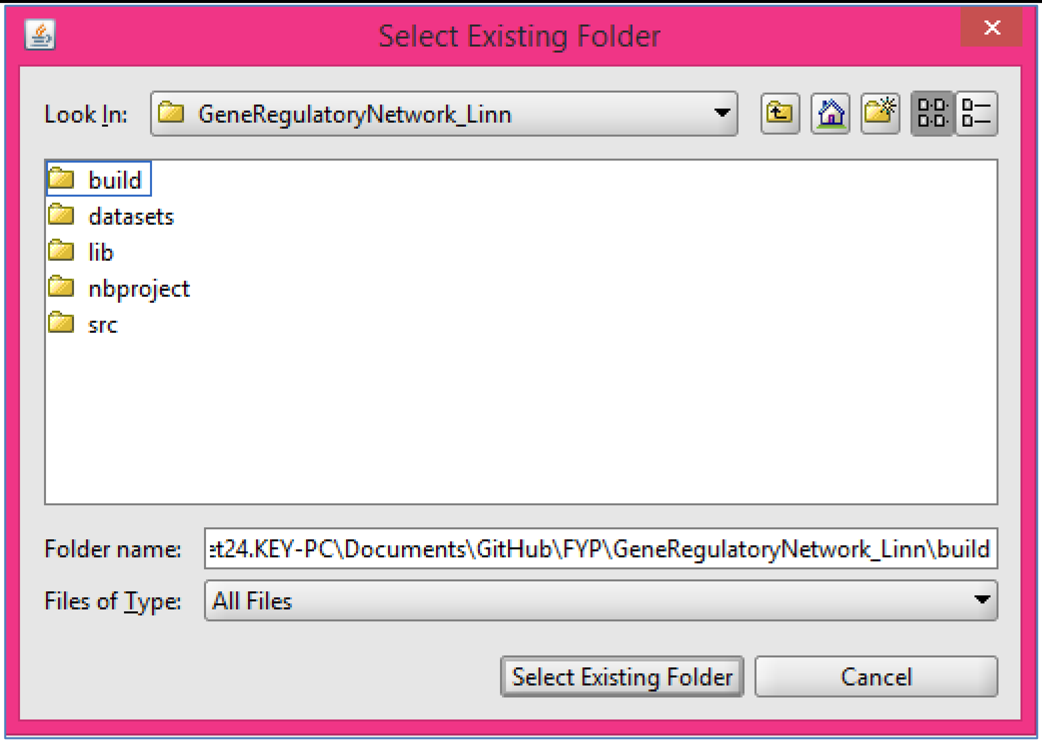


Figure 3.2a – Options for a Dynamic Network in New Version


Select Existing Folder

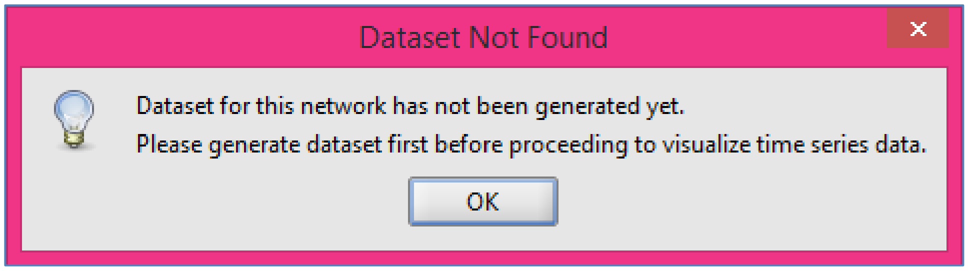
Look In: GeneRegulatoryNetwork_Linn

build
datasets
lib
nbproject
src

Folder name: ≥t24.KEY-PC\Documents\GitHub\FYP\GeneRegulatoryNetwork_Linn\build

Files of Type: All Files

Figure 3.2b – Selecting Directory where Generated Dataset Exists


Dataset Not Found

Dataset for this network has not been generated yet.
Please generate dataset first before proceeding to visualize time series data.

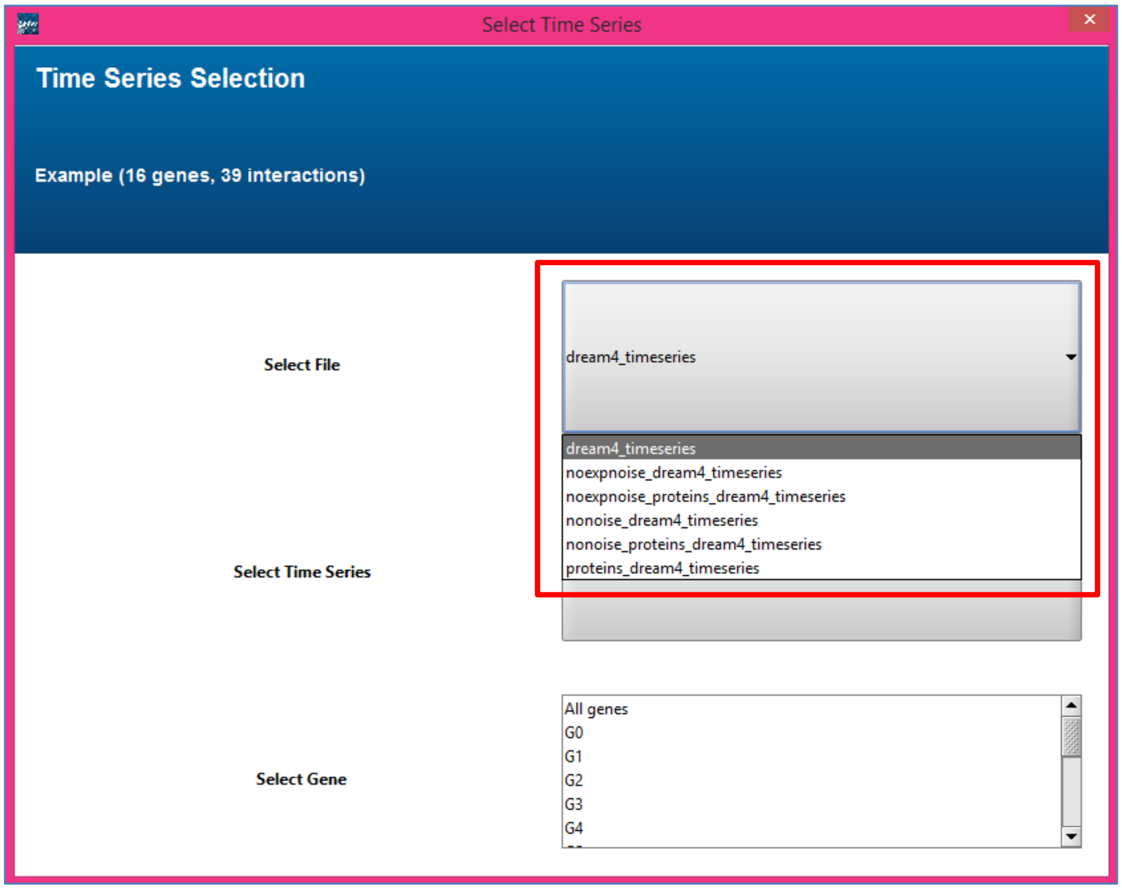
Figure 3.2c – Error Message Prompting to Generate Dataset First

### 3.2.1 Confirmation of Dataset's Existence

Currently, the types of time series data to visualize are specified in **GnwSettings** as stated in Section 6.2.10. When **Visualize Time Series Data** button is clicked, the system will identify if the data files for the current network exist in the specified folder as described in Section 6.2.11. However, not all files are necessarily to be found. Provided that any of those files are found, the system will consider that it is valid to visualize time series data of the current network since the available data list to visualize will be populated based on the number of existing data files later on.

### 3.2.2 Populating Available Time Series Data Files

In ***populateFileOptions*** function, using the types of time series data mentioned in Section 3.2.1, whether those types of time series data exist for the current network or not will be verified. Those available time series types will be added into the dropdown list as shown below.



dream4_timeseries

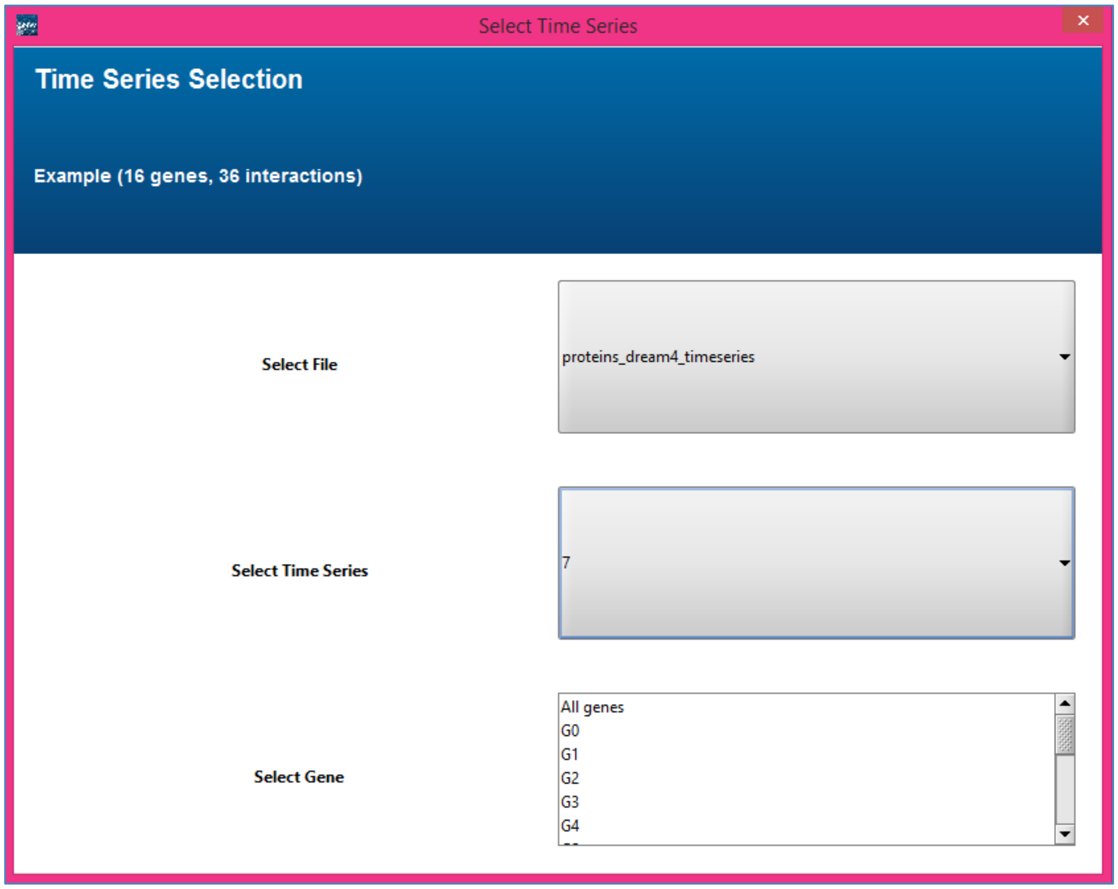
dream4_timeseries
noexpnoise_dream4_timeseries
noexpnoise_proteins_dream4_timeseries
nonoise_dream4_timeseries
nonoise_proteins_dream4_timeseries
proteins_dream4_timeseries

All genes
G0
G1
G2
G3
G4

Figure 3.2.2 – Different Options to Select Time Series to Visualize

### 3.2.3 Displaying All Genes

The following figures show how a default time series graph appears when "Visualize Time Series Data" button is clicked. Whenever the selected file is changed, the list of time series data for that particular file will be updated and the graph will be refreshed simultaneously. Either when **All genes** option from **Select Gene** is selected or when no gene is selected, time series data for all genes is visualized by default.



Select File: proteins_dream4_timeseries

Select Time Series: 7

Select Gene: All genes, G0, G1, G2, G3, G4

Figure 3.2.3.1 – Options for Time Series Visualization: No option selected

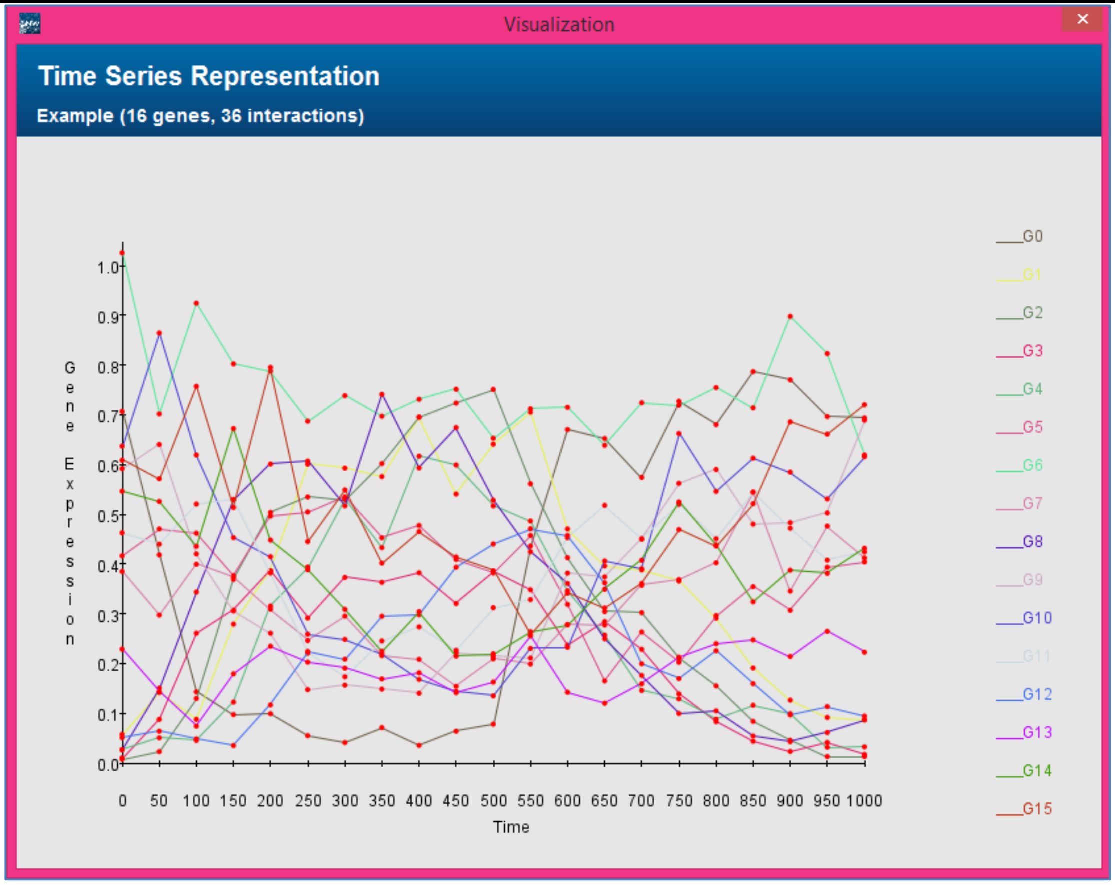
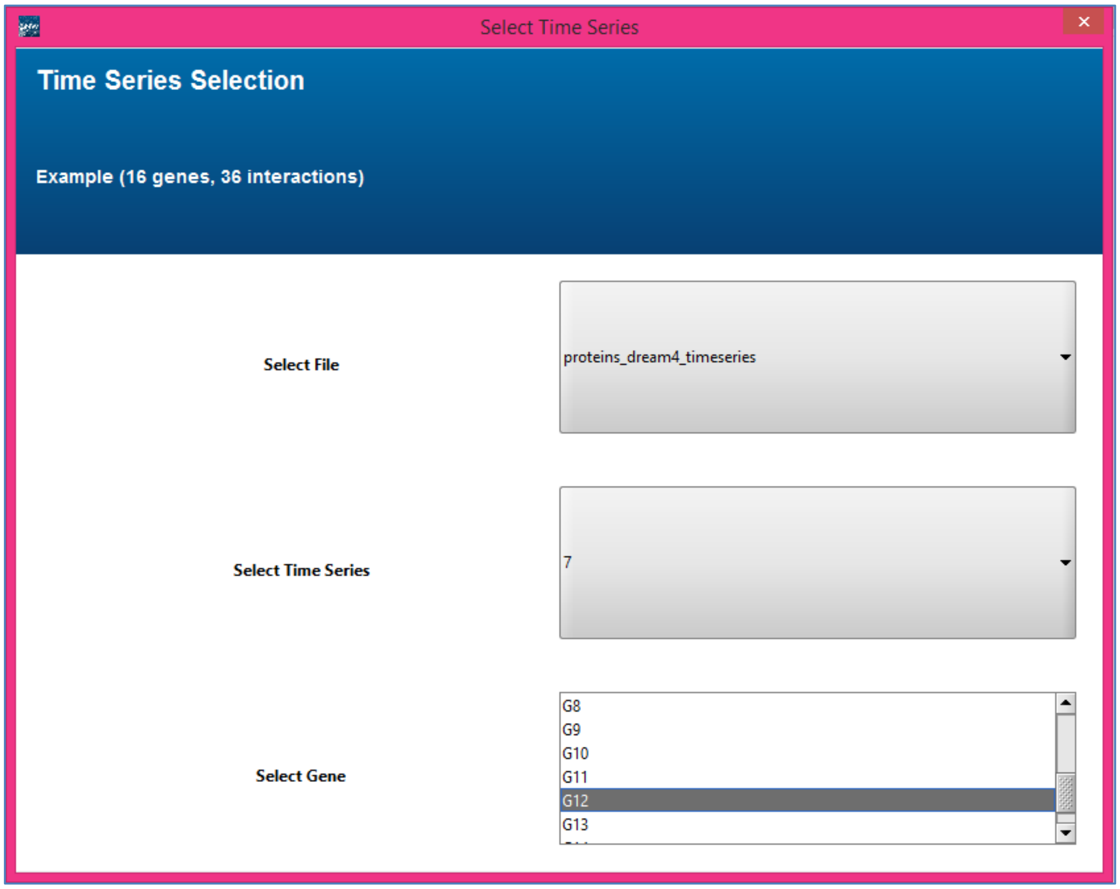


Figure 3.2.3.2 – Time Series Graph Displaying All Genes

### 3.2.4 Displaying One or More Genes

Despite the fact that viewing all genes at the same time is not a problem for networks with a few genes, it could be a concern for those composed of a large number of genes. Hence, for better convenience, not only each gene can be visualized separately but also more than one genes yet manageable number of genes can be displayed all together at the same time.



Select File: proteins_dream4_timeseries

Select Time Series: 7

Select Gene: G8, G9, G10, G11, G12 (selected), G13

Figure 3.2.4.1 – Options for Time Series Visualization: G12 selected

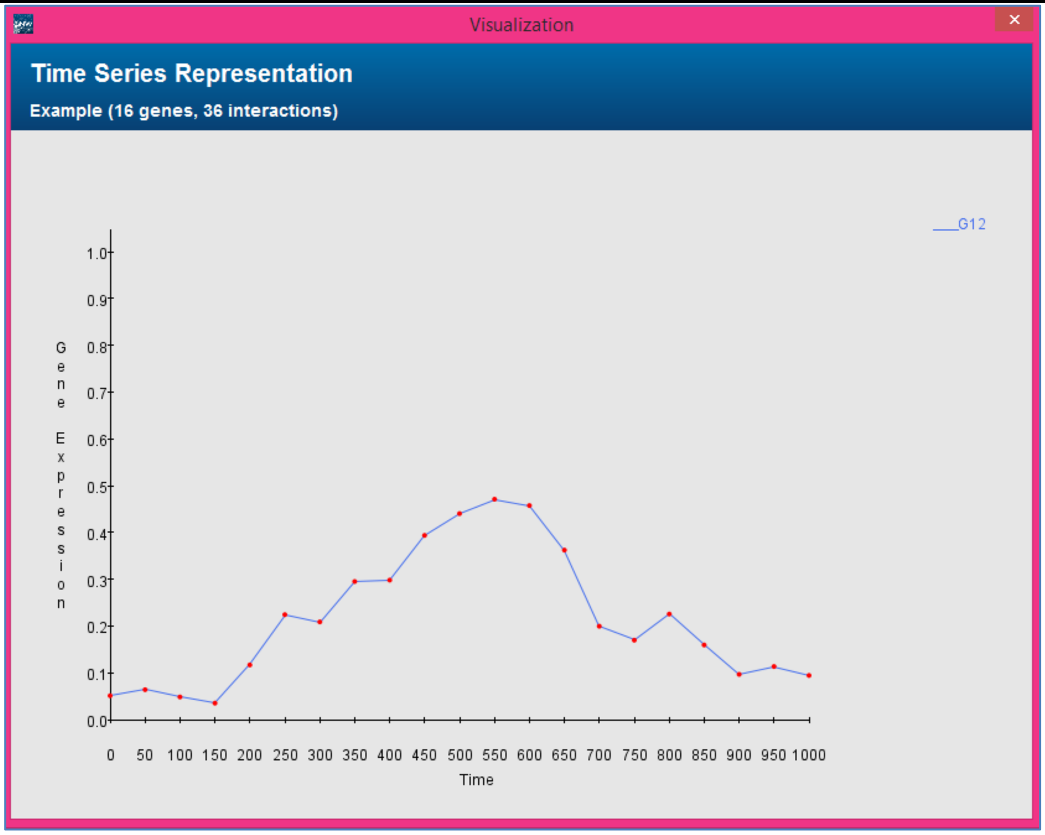


Figure 3.2.4.2 – Time Series Graph Displaying a Specific Gene

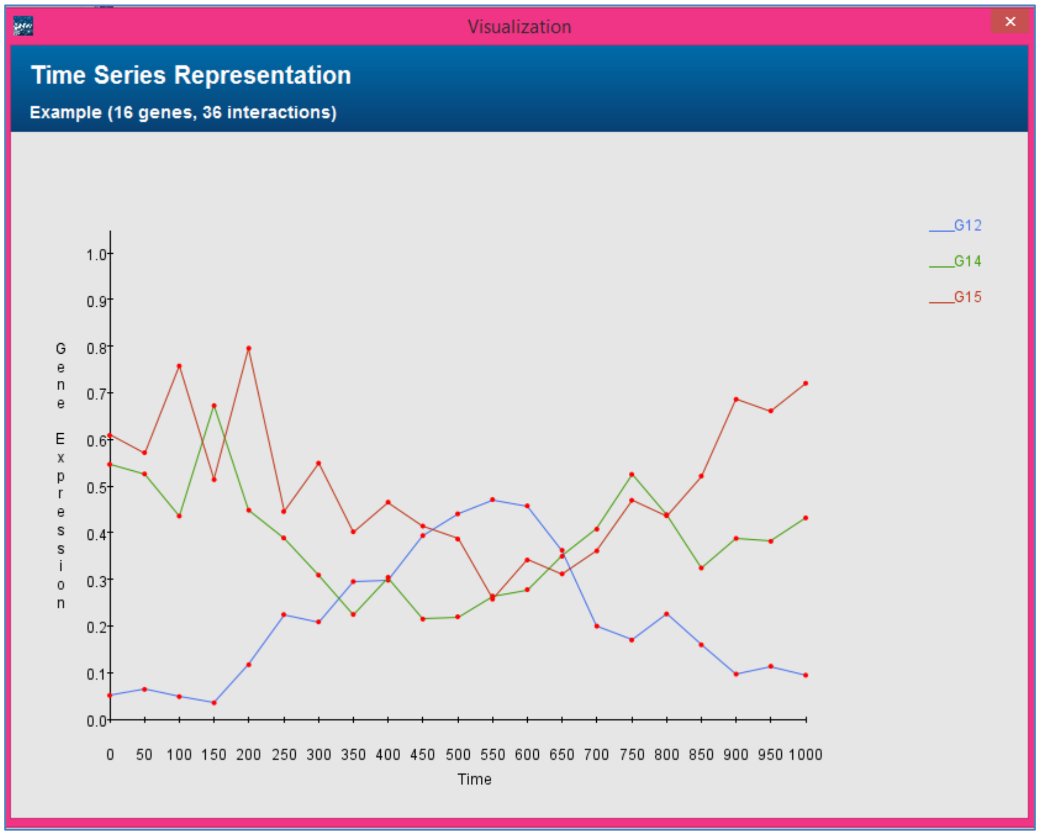
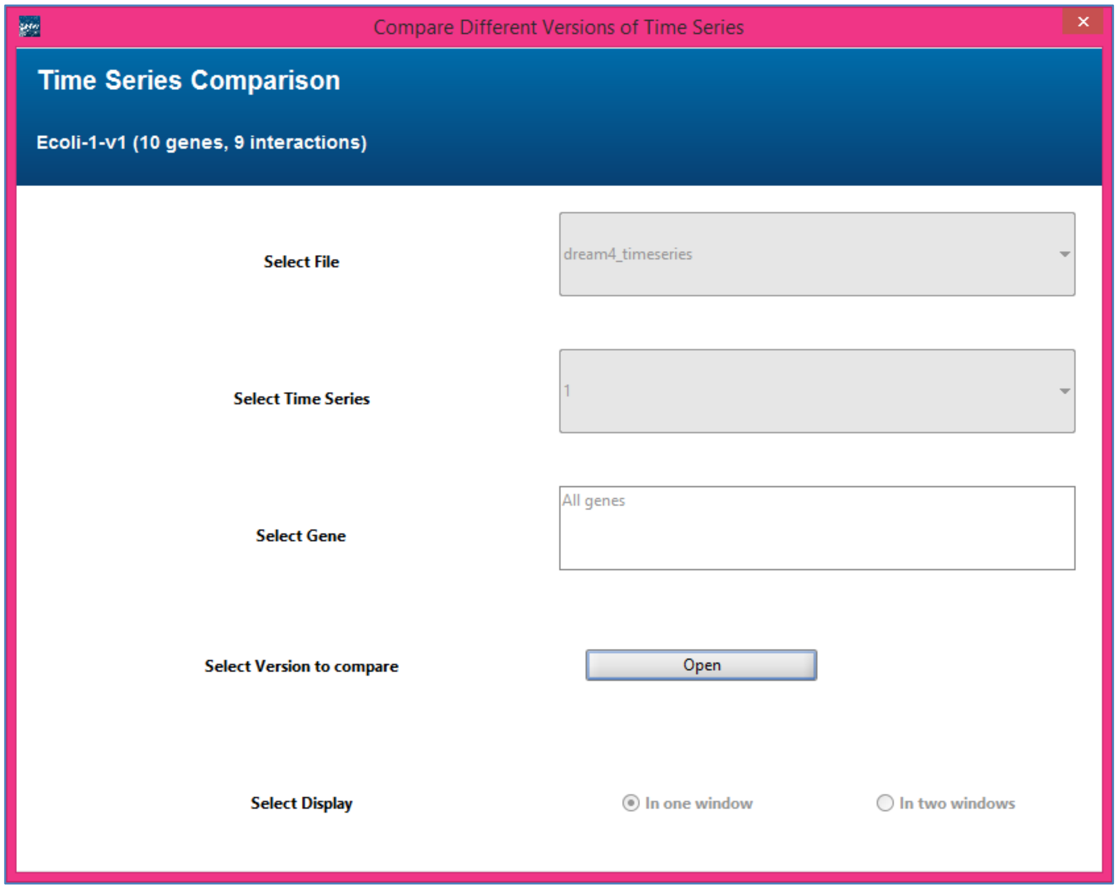


Figure 3.2.4.3 – Time Series Graph Displaying Three Genes Together

## 3.3 COMPARISON OF TIME SERIES BETWEEN DIFFERENT VERSIONS

When the button, **Compare Time Series Data**, from Figure 3.2a is clicked, the system will confirm the existence of dataset for the current network in the same way mentioned in Section 3.2.1.

After populating the list of time series data files for the current network, all the options to select are disabled until another version of the network to compare has been selected as shown in the following figure.



Compare Different Versions of Time Series

Time Series Comparison

Ecoli-1-v1 (10 genes, 9 interactions)

Select File: dream4_timeseries

Select Time Series: 1

Select Gene: All genes

Select Version to compare: Open

Select Display: [✓ In one window] In two windows

Figure 3.3a – Disabled Options for Comparing Time Series Data

The network to compare can be selected by clicking **Open** button. A dialog to choose an .xml file will appear. Only the modified versions of the current network are recommended to be chosen since comparison will be based on similarity between two versions. In other words, comparing two totally different networks will not give effective results at the end.

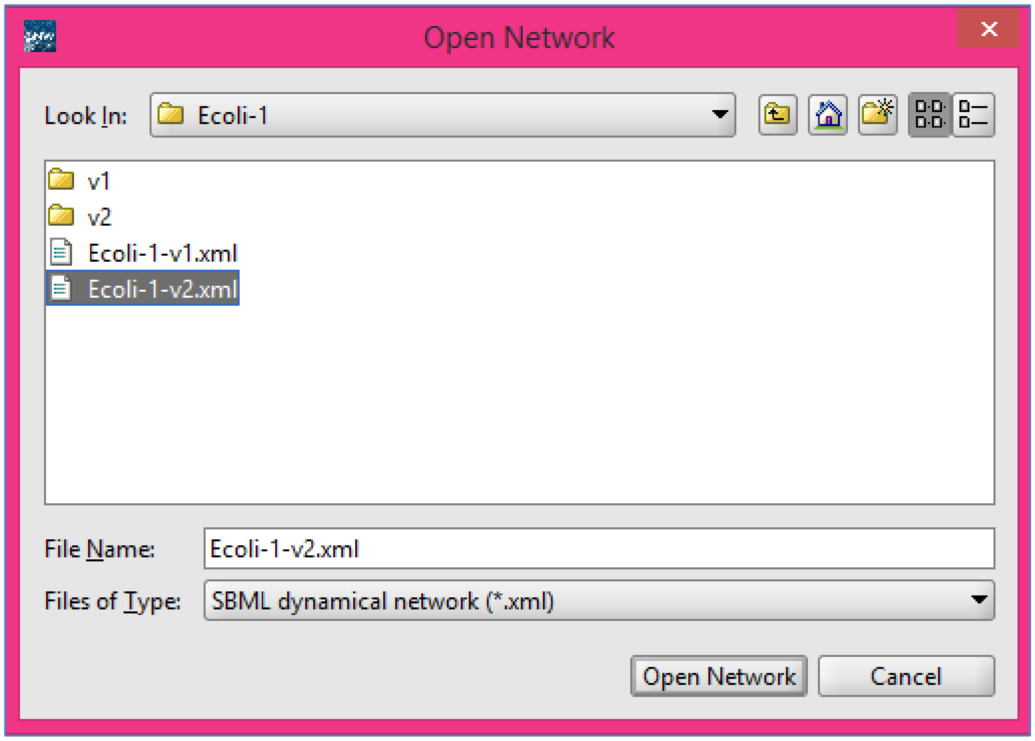


Figure 3.3b – Dialog to Select a Different Version of Network to Compare

After opening the version to compare, the directory where its generated dataset exists should also be specified. The same process will be gone through in order to identify if the dataset exists in that folder.

### 3.3.1 Populating Available Time Series Data to Compare

Unlike time series visualization, comparing time series filters all the common data between two versions of the network to compare. Common data means common time series and common genes. For instance, if the data file of one version holds 10 time series and another has only 5 time series, there will be 5 common time series between these two networks.

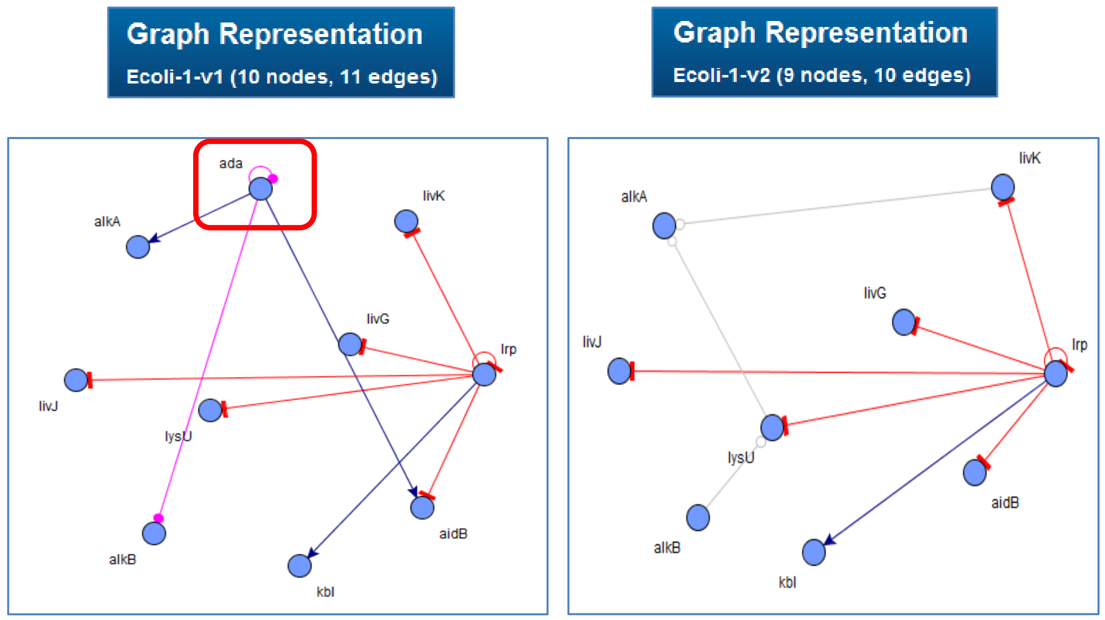


Figure 3.3.1 – Before and After Deleting **ada** Gene

In the above example, Ecoli-1-v1 network is modified by deleting ada gene and adding three more edges, and saved as Ecoli-1-v2. Therefore, all genes except ada are considered as common genes between these two networks. In the following sections, this case will be used to compare the gene expression data and analyze the effect of modifications done on Ecoli-1-v1 network.

### 3.3.2 Comparison using One Window

By default, two networks are compared in one window. This option works best when there are only few common genes. The changes of expression data of same genes but from different networks can be analyzed instantly. Each pair of two genes with the same name is differentiated by marking the first one (from current network) with *.

As shown in Figure 3.3.1, after deleting ada gene from the network, the genes alkA, alkB and aidB have been affected. Thus, let's compare the gene expression data of these three genes.

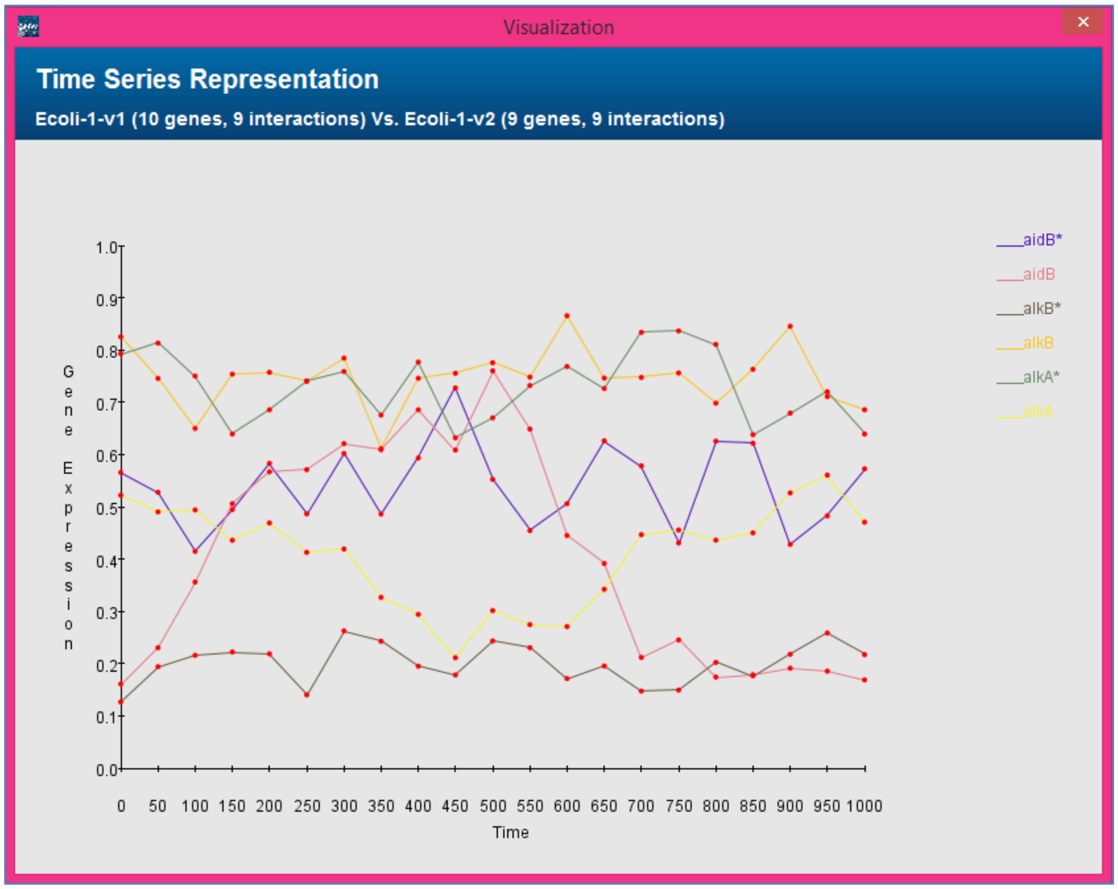
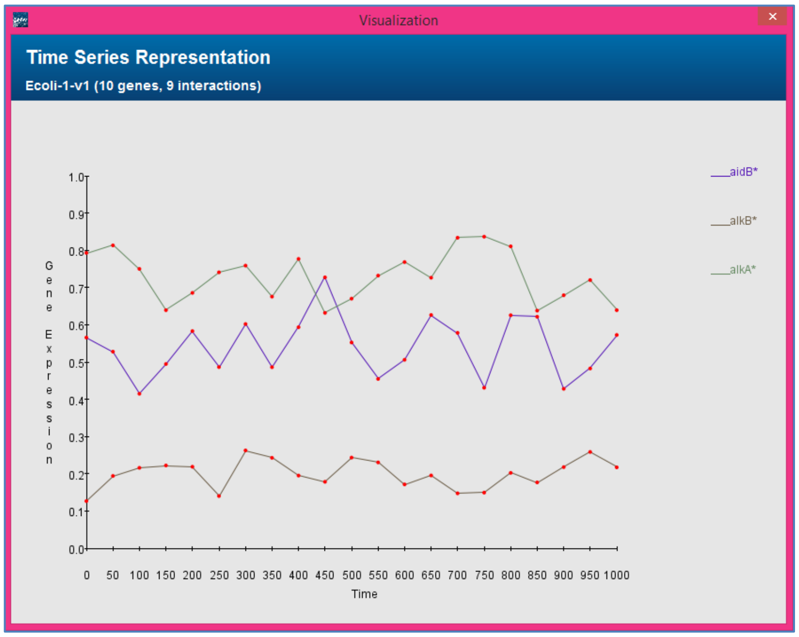


Figure 3.3.2 – Comparison between Two Networks using One Window

### 3.3.3 Comparison using Two Windows

Although using one window to compare might be suitable for networks with only a few genes, it would be inconvenient for user when the number of common genes between two networks is quite enormous. Therefore, this option helps user to visualize in a clearer way. However, even using one window, selecting a few particular genes to compare is an alternative for a clearer view.



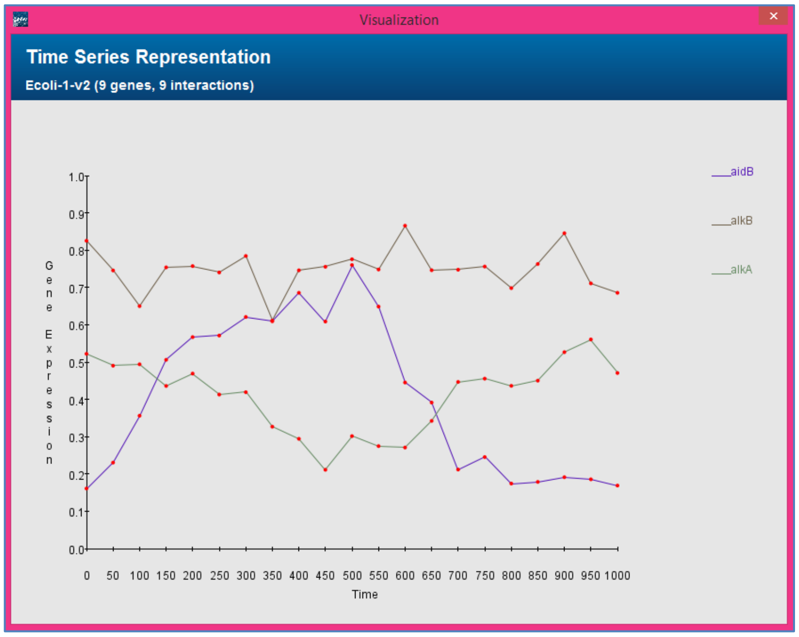


Figure 3.3.3 – Comparison between Two Networks using Two Windows

# 4 CONCLUSION

In this project, GNW has been enhanced and more potential improvement in the future has also been discovered at the same time. In this new version of GNW, generated dataset of GRN can be easily visualized and the structure of GRN can also be modified with a few mouse clicks. After modification, the effect of changes made can be analyzed by comparing the gene expression data before and after changes with the colorful GUI. With these newly integrated features, GNW is believed to help users gain a better understanding on the structure and functions of GRN, hence become a great tool assisting in research of new cure for diseases.

# 5 RECOMMENDATION FOR FURTHER IMPROVEMENT

Currently, editing structure feature is available for static networks only. Therefore, it can further be improved by implementing the same feature for dynamic networks, too.

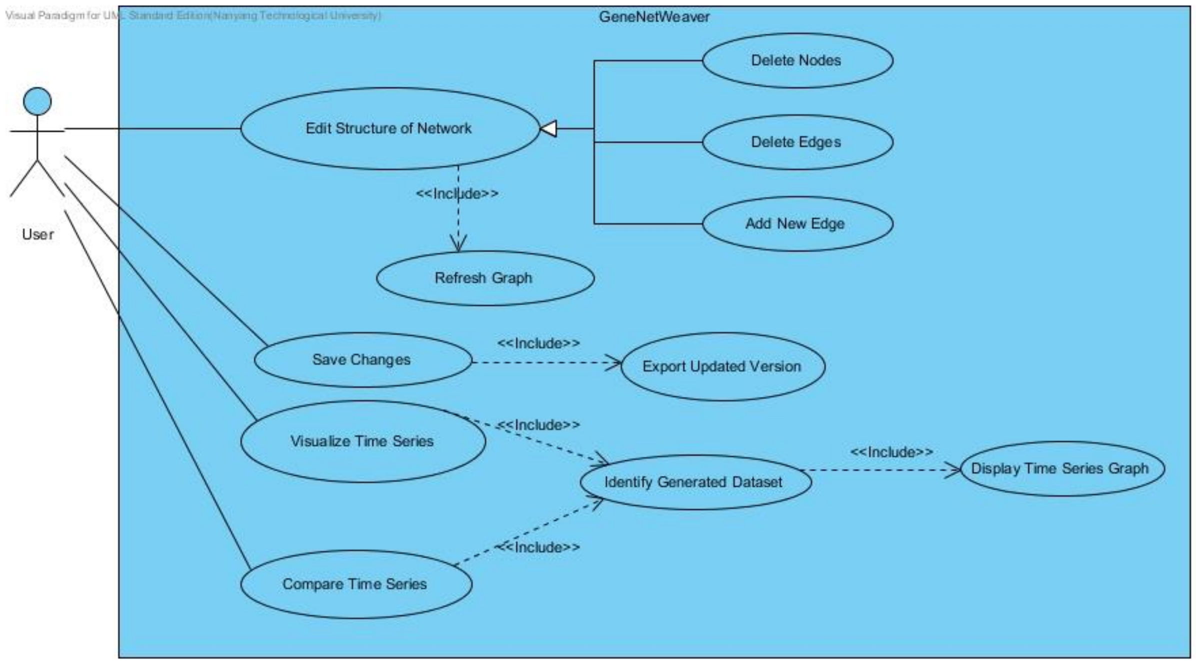
In addition, generating time series expression data does not take the time delay for each interaction between each pair of genes into consideration while calculating expression values. This can also be considered to further enhance GNW.

Apart from that, comparison feature can be enhanced by adding more detailed analysis based on the differences, such as which genes are the most affected ones by the modification done on the original network.

# 6 APPENDIX

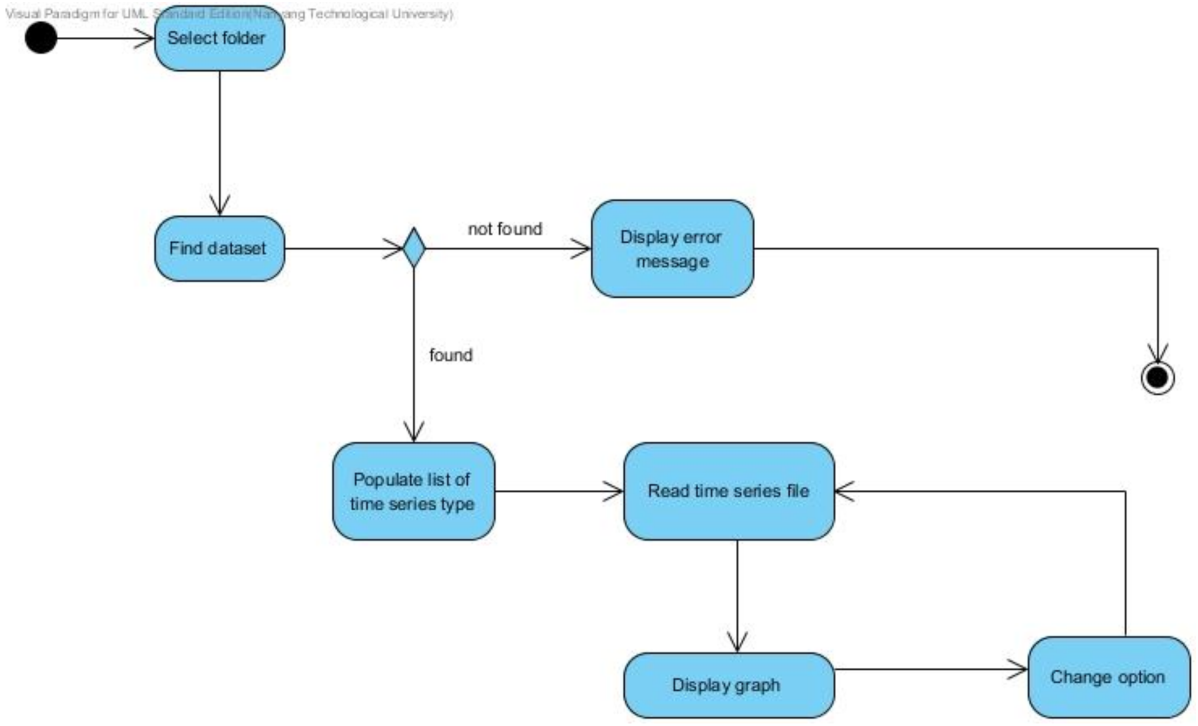
## 6.1 UML DIAGRAMS

### 6.1.1 Use Case Diagram

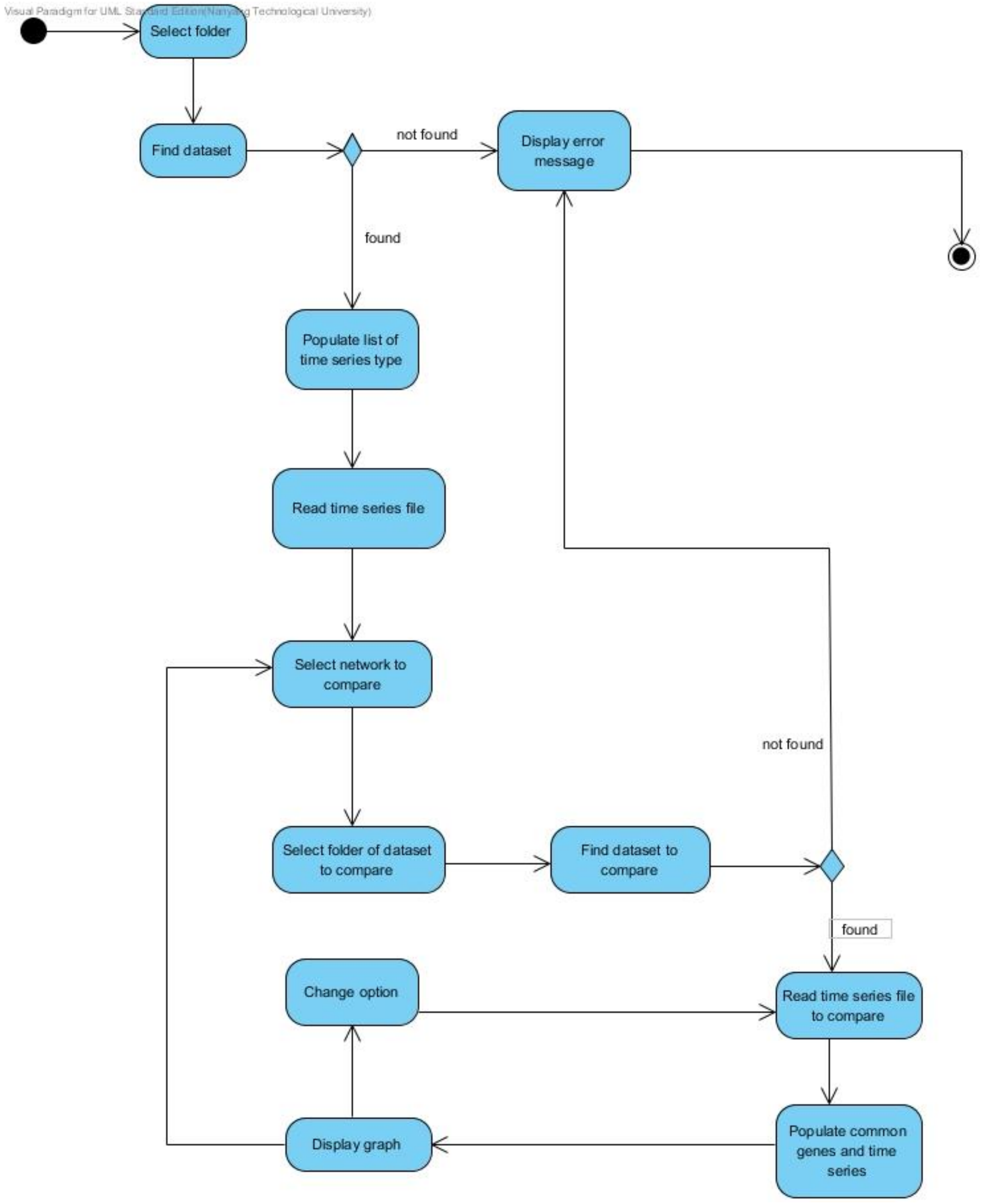


### 6.1.2 Activity Diagrams

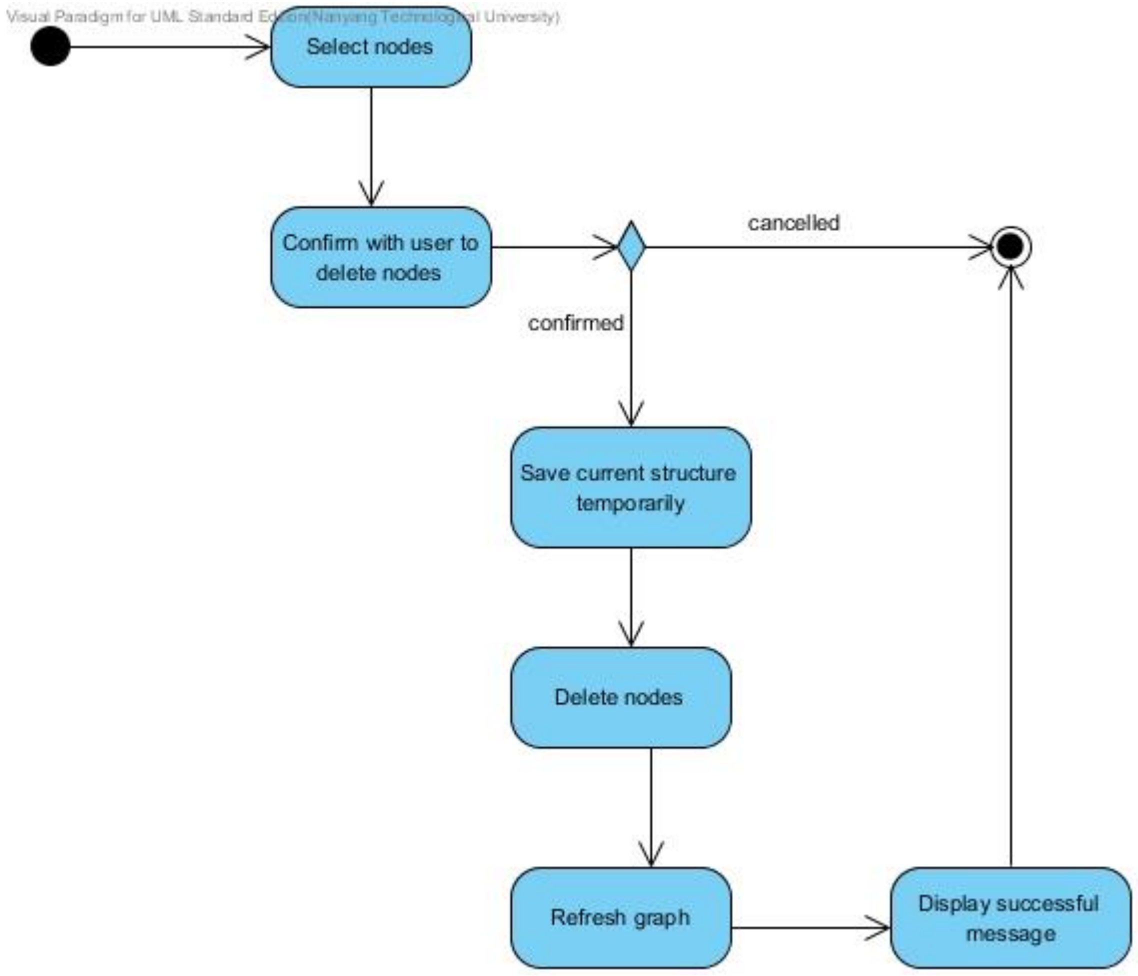
#### 6.1.2.1 Visualize Time Series


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
Select folder
Find dataset
not found
Display error message
found
Populate list of time series type
Read time series file
Display graph
Change option

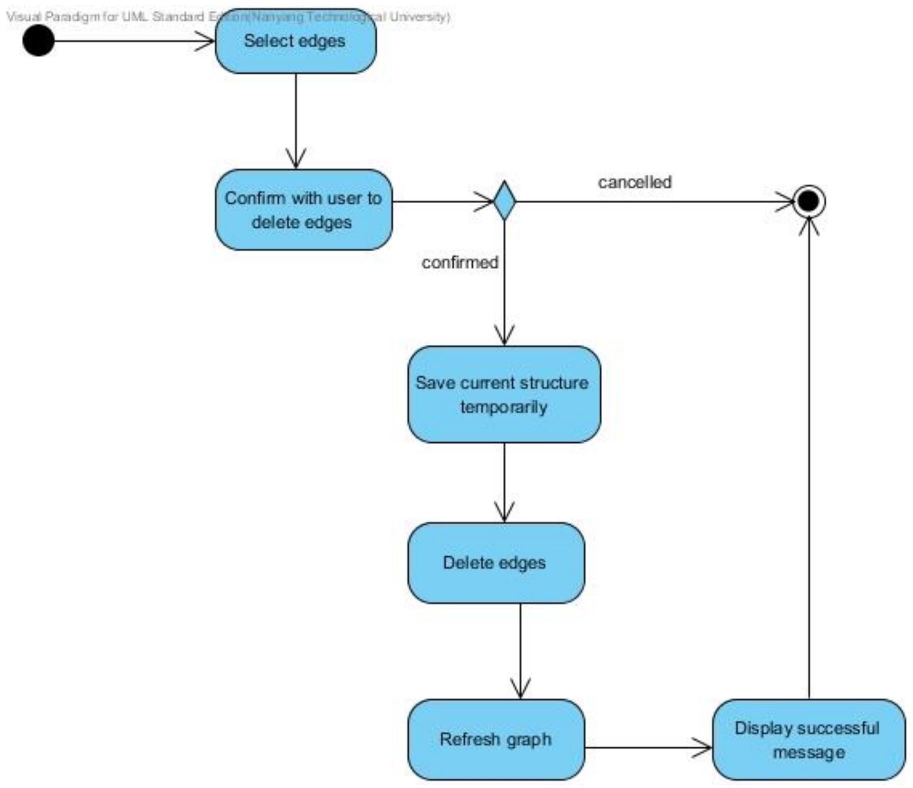
#### 6.1.2.2 Compare Time Series


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
Select folder
Find dataset
not found
Display error message
found
Populate list of time series type
Read time series file
Select network to compare
not found
Select folder of dataset to compare
Find dataset to compare
found
Change option
Read time series file to compare
Display graph
Populate common genes and time series

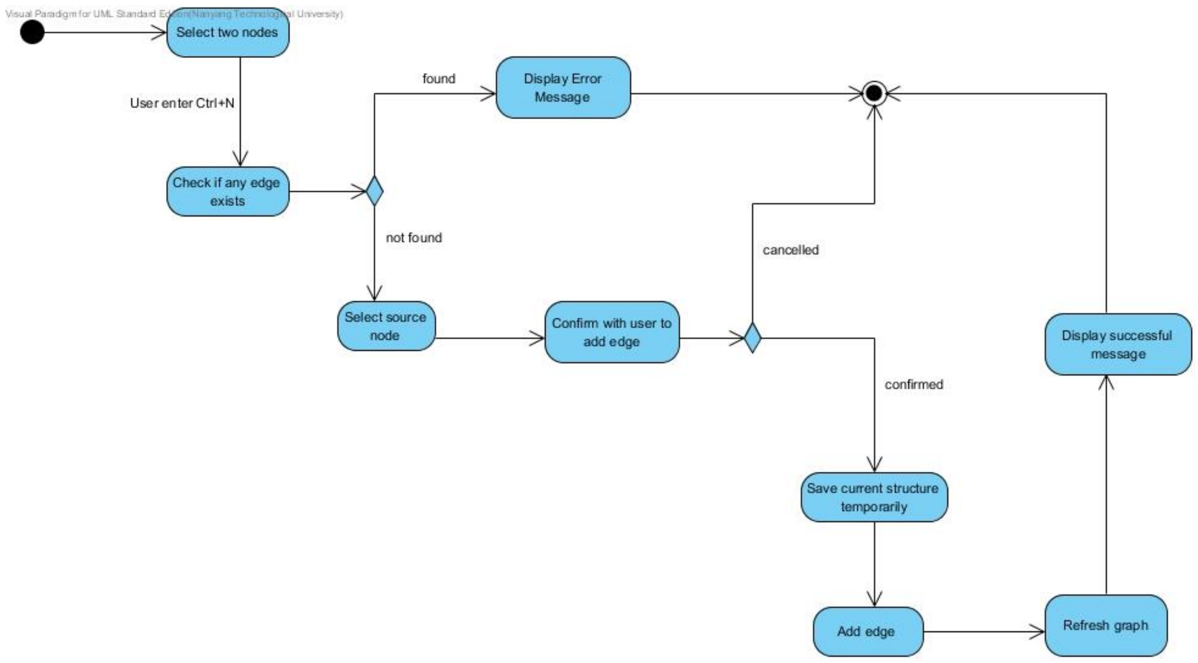
### 6.1.2.3 Delete Nodes


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
Select nodes
Confirm with user to delete nodes
cancelled
confirmed
Save current structure temporarily
Delete nodes
Refresh graph
Display successful message

#### 6.1.2.4 Delete Edges


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
Select edges
Confirm with user to delete edges
cancelled
confirmed
Save current structure temporarily
Delete edges
Refresh graph
Display successful message

#### 6.1.2.5 Add New Edge


Visual Paradigm for UML Standard Edition(Nanyang Technological University)
Select two nodes
User enter Ctrl+N
Check if any edge exists
found
Display Error Message
not found
Select source node
Confirm with user to add edge
cancelled
confirmed
Save current structure temporarily
Add edge
Refresh graph
Display successful message

## 6.2 PROGRAM LISTINGS

### 6.2.1 Main Method

```
// ==============================================================================
// MAIN METHOD

public static void main(String[] args) {

	try {
		setPackageLoggers(); // load input stream gnwguiLogSettings.txt
		parseArguments(args); // Loading default settings file
		setPlatformPreferences(); // osname is windows 8
		setLookAndFeel();

		GnwGui gui = new GnwGui();
		gui.run();

	} catch (Exception e) {
		e.printStackTrace();
	}
}
```

### 6.2.2 GnwGui.run

```
// ------------------------------------------------------------------------------

/**
 * Build and initialize the GUI.
 */
public void run()
{
	setGnwConsole();

	// display the splash
	splashScreen_ = new SplashScreen(new Frame(),
	                GnwGuiSettings.getInstance().getSplashScreenImage(),
	               SplashScreen.NORMAL, true, true, false);
	splashScreen_.setVisible(true);

	initialize();

	loadInitialNetworks();
	loadDREAMNetworks();

	setMessages();

	// hide the splash and display the loaded GUI interface
	splashScreen_.setVisible(false);
	this.getFrame().setVisible(true);
}
```

### 6.2.3 Options.setActions

```
private void setActions()
{
	bView_.addActionListener(new ActionListener()
	{
		public void actionPerformed(final ActionEvent e)
		{
			dispose();
			viewNetwork((NetworkElement) item_);
		}
	});

	bExport_.addActionListener(new ActionListener()
	{
		public void actionPerformed(final ActionEvent arg0)
		{
			if (item_ instanceof NetworkElement)
				exportNetwork();
			else if (item_ instanceof Folder || item_ instanceof Multi)
			{
				dispose();
				((NetworkDesktop)
GnwGuiSettings.getInstance().getNetworkDesktop()).displaySaveDialog();
			}
		}
	});

	bKinetic_.addActionListener(new ActionListener()
	{
		public void actionPerformed(final ActionEvent arg0)
		{
			conversion2DynamicalModel();
		}
	});

	bExtract_.addActionListener(new ActionListener()
	{
		public void actionPerformed(final ActionEvent arg0)
		{
			dispose();
			subnetworkExtraction((NetworkElement) item_);
		}
	});

	bDelete_.addActionListener(new ActionListener()
	{
		public void actionPerformed(ActionEvent e)
		{
			//CLOSE WINDOW
			dispose();

			//DISPLAY DELETE WINDOW
			if (item_ instanceof NetworkElement || item_ instanceof Folder
|| item_ instanceof Multi)
			{
```

```
                              GnwGuiSettings global = GnwGuiSettings.getInstance();
                              Delete dd = new Delete(global.getGnwGui().getFrame());
                              dd.setVisible(true);
                       }
                }
        });

        bRename_.addActionListener(new ActionListener()
        {
                public void actionPerformed(final ActionEvent arg0)
                {
                       renameAction();
                }
        });

        bDatasets_.addActionListener(new ActionListener()
        {
                public void actionPerformed(final ActionEvent arg0)
                {
                       try {
                              dispose();
                              generateDREAM3GoldStandard((NetworkElement) item_);

                       } catch (Exception e) {
                              log_.log(Level.WARNING, "Options::Options(): " +
e.getMessage(), e);
                       }
                }
        });

        bTimeSeries_.addActionListener(new ActionListener()
        {
                public void actionPerformed(final ActionEvent arg0)
                {
                       visualizeTimeSeries(false);
                }
        });

        bCompare_.addActionListener(new ActionListener()
        {
                public void actionPerformed(final ActionEvent arg0)
                {
                       visualizeTimeSeries(true);
                }
        });
}
```

### 6.2.4 Options.viewNetwork

```
// ----------------------------------------------------------------------------

/**
 * Open a dialog to visualize the network graph.
 * @param item
 */
public static void viewNetwork(NetworkElement item)
{

        int numNodes = 0;

        if (item instanceof StructureElement)
                numNodes = ((StructureElement)item).getNetwork().getSize();
        else if (item instanceof DynamicalModelElement)
                numNodes = ((DynamicalModelElement)item).getGeneNetwork().getSize();

        if (numNodes >= 200)
        {
                String msg = "Large networks (> 200 nodes) can take some time to
display. Continue ?";

                int n = JOptionPane.showConfirmDialog(
                                new Frame(),
                                msg.toString(),
                                "GNW message",
                                JOptionPane.YES_NO_OPTION,
                                JOptionPane.QUESTION_MESSAGE);

                if (n == JOptionPane.NO_OPTION)
                        return;
        }

        GraphViewer dialog = new GraphViewer(new Frame(), item);
        dialog.setVisible(true);
}
```

### 6.2.5 NetworkGraph.initialize

```
// ---------------------------------------------------------------------------

public void initialize() {

       if (item_ instanceof StructureElement) {
             structure_ = ((StructureElement)item_).getNetwork();
       } else if (item_ instanceof DynamicalModelElement) {
             structure_ = ((DynamicalModelElement)item_).getGeneNetwork();
       }

       netSize_ = structure_.getSize();
       control_ = new MyGraphViewerController();
       screen_ = new JPanel();
       screen_.setLayout(new BorderLayout());
       computeGraph();
}
```

### 6.2.6 NetworkGraph.computeGraph

```
/**
 * Create the graph from the structure of a network.
 */
@SuppressWarnings({ "unchecked", "rawtypes" })
public void computeGraph() {

    if (structure_.isDirected())
        g_ = new DirectedSparseMultigraph<Node,Edge>();
    else
        g_ = new UndirectedSparseMultigraph<Node,Edge>();

    // The layout is instantiated after all nodes are added to the graph.
    // Otherwise the nodes are misplaced!
    addVertices();
    addEdges();

    //Layout v= new FadingNodeLayout( 10, new SpringLayout( g ));
    layout_ = new FRLayout<Node,Edge>(g_);
    // sets the initial size of the layout space
    layout_.setSize(new Dimension(250,250));
    vv_ = new VisualizationViewer<Node,Edge>(layout_); // Graph layout
    pickedState_ = vv_.getPickedVertexState(); //Get information from the nodes picked
    pickedEdges_ = vv_.getPickedEdgeState(); //Get information from the edges picked

    // If node positions are present in the Structure object, e.g. loaded from
    // DOT files, the layout is remodelled.
    loadStructureLayout();

    // LABEL
    vs_none_ = new ConstantTransformer(null);
    //vs_ = new ToStringLabeller();
    vs_ = new NodeLabelLabeller();
    // NODES, EDGES AND ARROW HEADS
    // By default no distinction of arrow head
    arrowTransformer_ = new ArrowShapeTransformer<Node,Edge>(false);
    // By default no distinction of edge/outside arrow color and inside arrow color
    edgeArrowColorTransformer_ = new EdgeArrowColorTransformer<Edge>(false,
pickedEdges_);
    insideArrowColorTransformer_ = new EdgeArrowColorTransformer<Edge>(false,
pickedEdges_);
    // By default no distinction of stroke edges
    edgeTransformer_ = new EdgeTransformer<Edge>(false);
    // Defines the color of the nodes
    nodeFillColorTransformer_ = new NodeFillColorTransformer<Node>(pickedState_);

    vv_.setPreferredSize(new Dimension(400, 250));
    vv_.setBackground(Color.white);

vv_.getRenderContext().setVertexFillPaintTransformer(nodeFillColorTransformer_);
    vv_.getRenderContext().setVertexLabelTransformer(vs_none_/*new
ToStringLabeller()*/);
    vv_.getRenderer().getVertexLabelRenderer().setPosition(Position.AUTO);
    vv_.getRenderContext().setEdgeShapeTransformer(new EdgeShape.Line<Node,Edge>());

    vv_.getRenderContext().setEdgeArrowTransformer(arrowTransformer_);
    vv_.getRenderContext().setEdgeDrawPaintTransformer(edgeArrowColorTransformer_); //
edge color
    vv_.getRenderContext().setArrowFillPaintTransformer(insideArrowColorTransformer_);
// arrow inside color
    vv_.getRenderContext().setArrowDrawPaintTransformer(edgeArrowColorTransformer_);
// arrow outside color
    vv_.getRenderContext().setEdgeStrokeTransformer(edgeTransformer_); // edge
transformer



    // Create a graph mouse and add it to the visualization component
    gm_ = new DefaultModalGraphMouse();
    gm_.setMode(ModalGraphMouse.Mode.TRANSFORMING);
    vv_.setGraphMouse(gm_);

    setScreen(vv_); // Draw the viewer into the panel displayed
    addPrintAction(screen_); // Add the key action ALT-P to print the JPanel screen_
    addDeleteAction(screen_); // Add the key action Ctrl-D to delete selected nodes
    addDeleteEdgeAction(screen_); // Add the key action Ctrl-E to delete selected edge
    addCreateEdgeAction(screen_); // Add the key action Ctrl-N to create a new edge
    addSaveAction(screen_); // Add the key action Ctrl-S to save the changes

    // Finally, add the signature at the low-bottom corner of the graph visualization
    // addSignature(vv_);
    signature_ = new GraphSignature(vv_, new
ImageIcon(GnwGuiSettings.getInstance().getGnwWatermarkImage()));
    try {
        signature_.addSignatureToContainer();
    } catch (Exception e) {
        log_.log(Level.WARNING, "NetworkGraph::computeGraph(): " + e.getMessage(), e);
    }
}
```

### 6.2.7 Options.exportNetwork

```
// ------------------------------------------------------------------------------

/** Open a dialog to export the present network. */
public void exportNetwork()
{
        escapeAction();
        IONetwork.saveAs((NetworkElement) item_);
}
```

### 6.2.8 Options.conversion2DynamicalModel

```
// ------------------------------------------------------------------------------

/**
 * Convert a structure into a dynamical model. The item of the structure on the
 * desktop is also transformed into a dynamical model item.
 */
public void conversion2DynamicalModel() {

        JOptionPane optionPane = new JOptionPane();
        Object msg[] = {"Do you want to remove autoregulatory interactions ?"};
        optionPane.setMessage(msg);
        optionPane.setMessageType(JOptionPane.QUESTION_MESSAGE);
        optionPane.setOptionType(JOptionPane.YES_NO_CANCEL_OPTION);
        JDialog dialog = optionPane.createDialog(this, "Dynamical model");
        dialog.setVisible(true);
        Object value = optionPane.getValue();

        int size = 0;

        if (value == null || !(value instanceof Integer)) {
                //log.log(Level.WARNING, "Wrong returned value.");
                return;
        }else {
                int i = ((Integer)value).intValue();
                if (i == JOptionPane.CLOSED_OPTION || i == JOptionPane.CANCEL_OPTION) {
                        return;
                }else if (i == JOptionPane.OK_OPTION) {
                        // Close the options dialog
                        enterAction();
                        // Remove autoregulatory interactions
                        if (item_ instanceof StructureElement)
                        {
                                size = ((StructureElement)
item_).getNetwork().getSize();


                                ((StructureElement)
item_).getNetwork().removeAutoregulatoryInteractions();
                                log_.info("Autoregulatory interactions are removed from
network " + ((StructureElement) item_).getNetwork().getId() + "!");
                        }else if (item_ instanceof DynamicalModelElement)
                        {
                                size = ((DynamicalModelElement) 
item_).getGeneNetwork().getSize();

                                ((DynamicalModelElement) 
item_).getGeneNetwork().removeAutoregulatoryInteractions();
                                log_.info("Autoregulatory interactions are removed from 
network " + ((DynamicalModelElement) item_).getGeneNetwork().getId() + "!");
                        }
                }
                else if (i == JOptionPane.NO_OPTION)
                        enterAction();
        }

        Wait wait = new Wait(GnwGuiSettings.getInstance().getGnwGui().getFrame(), 
true);
        wait.setTitle("Kinetic model");
        KineticModelGeneration kmg = new KineticModelGeneration(wait);
        kmg.execute();

        // For networks with less than 1000 nodes, it's possible that the generation of 
the
        // kinematic model is fast (process in thread finish before the wait dialog is 
display!)
        if (size > 1000)
                wait.start();
}
```

### 6.2.9 Options.generateDREAM3GoldStandard

```
// ----------------------------------------------------------------------------

/**
 * Open a dialog to generate benchmarks.
 * @throws Exception
 */
public static void generateDREAM3GoldStandard(NetworkElement item) throws Exception {
        Simulation rd = new Simulation(new Frame(), item);
        rd.setVisible(true);
}
```

### 6.2.10 GnwSettings.timeSeriesDataFiles

```
// File list for timeseries data
private String[] timeSeriesDataFiles = { "dream4_timeseries"
                    , "noexpnoise_dream4_timeseries"
                    , "noexpnoise_proteins_dream4_timeseries"
                    , "nonoise_dream4_timeseries"
                    , "nonoise_proteins_dream4_timeseries"
                    , "proteins_dream4_timeseries"};
```

### 6.2.11 Options.dataExists

```
// ----------------------------------------------------------------------------
public boolean dataExists(String path, String[] files){
        File file;
        for (int i = 0; i < files.length; i++){
            file = new File(path + "\\" + item_.getLabel() + "_" + files[i] + ".tsv");
            if(file.exists()){
                return true;
            }
        }
        return false;
}
```

### 6.2.12 TimeSeriesSelectionWindow.populateFileOptions

```
private Vector<String> populateFileOptions(){
    Vector<String> vFiles = new Vector<String>();
    String[] files = GnwSettings.getInstance().getTimeSeriesDataFiles();
    File file;

    for (int i = 0; i < files.length; i++){
        file = new File(path_ + "\\" + item_.getLabel() + "_" + files[i] + ".tsv");
        if(file.exists()){
            vFiles.add(files[i]);
        }
    }
    return vFiles;
}
```

# 7 REFERENCES

- GeneNetWeaver Website, http://gnw.sourceforge.net/
- Schaffter T, Marbach D, and Floreano D. (2011) GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods. *Bioinformatics, 27(16):2263-70*