version

2.0

BILKENT UNIVERSITY

i-Vis Information Visualization Research Group

Chisio Editor User's Guide

BILKENT I-VIS RESEARCH GROUP

Chisio Editor User's Guide

iVis Info. Visualization Research Group

Computer Engineering Department • Bilkent University
Ankara 06800, TURKEY
Phone +90 312.290.1401 • Fax +90 312.266.4047

Table of Contents

Chapter 1

Overview

In this chapter, we describe the Chisio Editor tool, give its use cases and explain how to start the tool.

Chisio Editor is a graph visualization tool for creating, editing and layout of *compound* or *hierarchically structured* graphs. The tool features user-friendly, interactive creation and manipulation of compound graphs. In addition, a number of popular graph layout algorithms, including ones designed by our group, are available through the Chisio Layout library component [8].

Use Cases

Chisio can be used for different purposes. If you would like to simply use it as a generic graph editor, please continue reading this manual. However, if your goal is to:

- customize the graph editor for a specific application (e.g. one that is used to draw UML class diagrams or a tool for visualization of social networks), please refer to Chisio Editor Programmer's Guide.
- customize the tool for implementing a new layout algorithm or modify an existing one (e.g. an algorithm that you developed and would like to test in an interactive tool), please refer to Chisio Layout Programmer's Guide [8].

How to Download and Start Chisio

Chisio 2.0 is supported on both Windows and Linux operation systems. To download and set up Chisio 2.0, simply go to the tool Web site and follow the instructions provided:

http://www.cs.bilkent.edu.tr/~ivis/chisio.html

Chapter 2

Graph and Drawing Model

In this chapter, we present the graph and drawing model used in Chisio Editor.

The graph model used in Chisio Editor is managed by a compound graph. A compound graph manages a list of nodes (possibly compound) and edges. Compound nodes, in turn, manage a list of nodes and edges, potentially creating an arbitrary level of nesting hierarchies.

Graph Model

A *Chisio graph*, or simply a *graph*, is composed of a *root graph*, which contains a set of *nodes* and *edges*, which are called *graph objects*. Nodes are joined to or connected to each other by edges. For example, Figure 1 shows a Chisio graph with three nodes (A, B, and C) and three edges (A to B, A to C, and B to C).

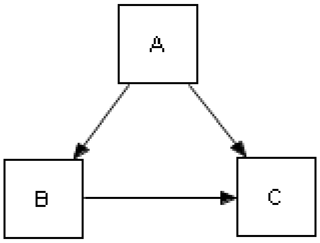


Figure 1 A Chisio graph with a root graph of 3 simple nodes (A, B, and C) and 3 edges (A-B, A-C, and B-C)

An edge has a *source* node and *target* node (Figure 2). If there are multiple edges between the same source and target nodes, then these edges are called *multi-edges* (Figure 3). If source and target nodes are the same then this edge is called a *reflexive* edge, simply a *loop* (Figure 4).

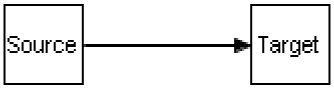


Figure 2 An edge with its source and target

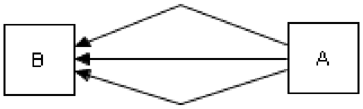


Figure 3 Example multi-edges

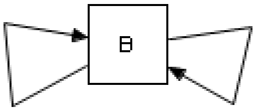


Figure 4 Example reflexive edges (loops)

A node maintains lists of its *incoming* and *outgoing* edges. Incoming edges are directed toward the node and outgoing edges are directed away from the node. In other words, node is the target node for its incoming edges and it is the source node for its outgoing edges (Figure 5).

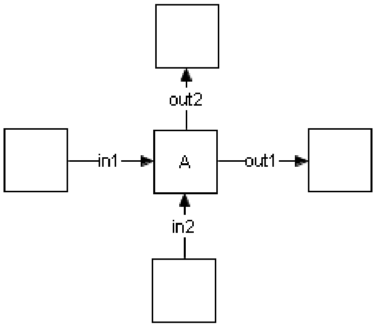


Figure 5 A node with incoming and outgoing edges

In a *directed graph*, each edge represents a one-way relationship from its source node to its target node. Directed edges use an arrow to indicate the direction of each edge. In an undirected graph, the directions of edges are ignored. Undirected edges are usually represented by lines without arrowheads (Figure 6).

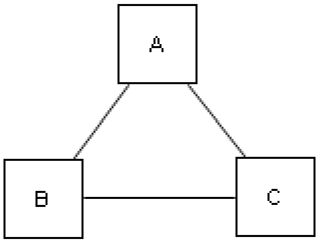


Figure 6 An undirected graph

Compound nodes are those that recursively contain a set of nodes and edges, composing its *child graph*, that are hierarchically below the compound node in the nesting tree. Edges can be categorized into two, depending on whether or not their end-nodes are in the same graph. If both source and target nodes of an edge are in the same graph, then the edge is said to be an *intra-graph edge*; it's called an *inter-graph edge*, otherwise.

For example, the graph in Figure 7 contains two compound nodes (G, H) and one simple node (J) in its root graph. Compound node G, in turn, has one compound (E) and two simple nodes (F, D) in its child graph. Finally, compound node E has one simple node (I). Compound node H has three nodes (A, B, C). Also, there are several inter- and intra-graph edges between these compounds and nodes as shown on the figure.

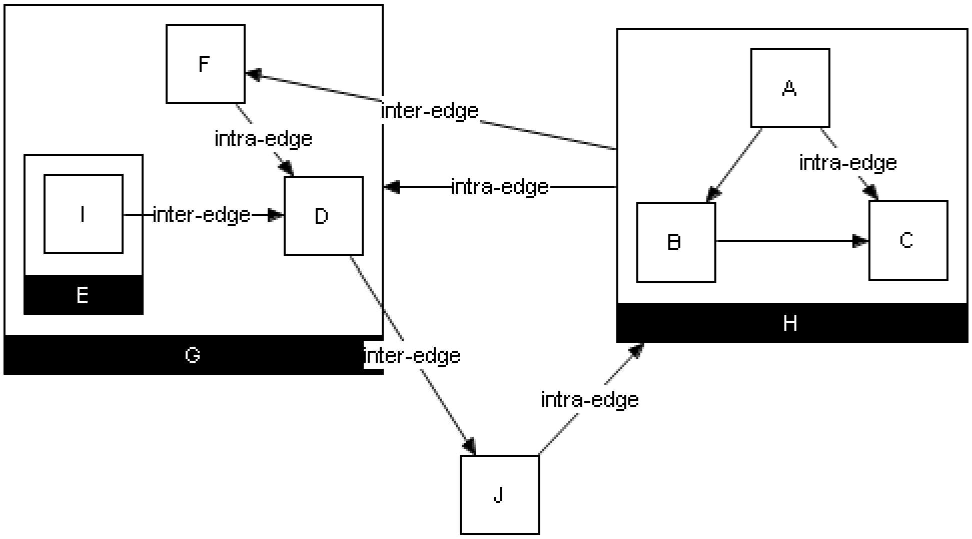


Figure 7 A compound graph with nesting of two levels, and intra and inter-graph edges

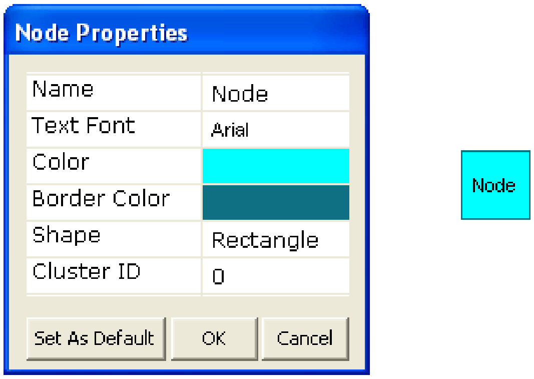
Drawings

Nodes, edges and compound nodes in Chisio Editor have distinct properties and UIs. These properties can be changed by using properties windows or inspectors for each graph objects. When you change your object's properties, these changes are transferred to the model only after you press the "OK" button.

Nodes

Nodes are represented by a *rectangle* shape as default. But there are other pre-determined shapes that can be added to the graph by user interaction. These are *ellipse* and *triangle*. If you would like to add more types of shapes, you can easily add them by following the instructions in the Chisio Editor Programmer's Guide.

Figure 8 shows a sample "Node Properties" window. By using this window, you can change your node's appearance. This window can be opened up by either double clicking the node or by using the node's popup menu.



Node Properties

Name	Node
Text Font	Arial
Color	🟦
Border Color	🟦
Shape	Rectangle
Cluster ID	0

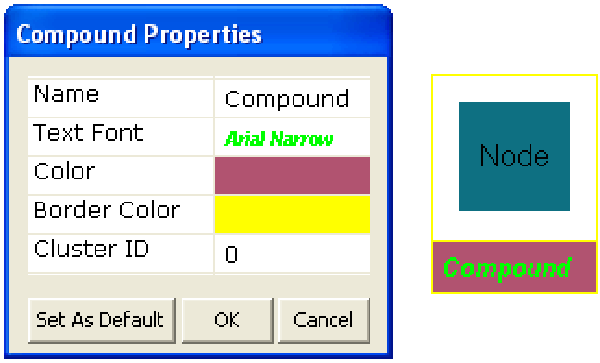
Figure 8 A node and its properties window or inspector

You can also change the label (font type, size, and color), color, border color, shape and cluster IDs of a node by using its "Properties" window.

"Set As Default" button is used for setting this node's properties as the default for the nodes that are to be created later on. Cluster IDs is ignored when default properties are set.

Compound Nodes

Similar to a simple node, a compound node's properties may be changed via the "Compound Properties" window (Figure 9). This window may also be opened up similar to the Node Properties window by double-clicking on it or through its pop-up menu.



Compound Properties

Name	Compound
Text Font	Arial Narrow
Color	🟥
Border Color	🟨
Cluster ID	0

Figure 9 A compound node (right) and its properties window

You can change the label, color and border color of a compound node through this window. The size of a compound node is auto-calculated by the geometry of its contents, and the compound node is always just large enough to tightly bound its contents plus some user-defined margins.

"Set As Default" button is used for setting this compound's properties as the default for the compounds that are to be created later on. Cluster ID part is ignored when default properties are set.

Edges

Both directed and undirected graphs can be visualized in Chisio Editor. Edges are assumed to be connected to the center of their source and target nodes. An edge is drawn clipped, according to the specific shape and position of its source and target nodes (Figure 10).

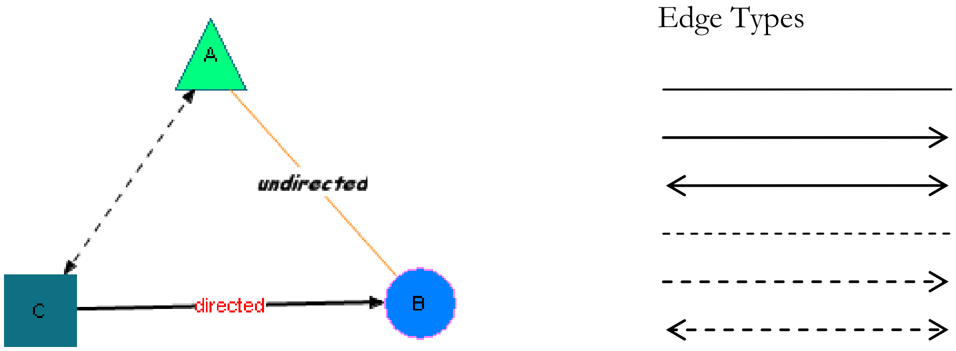
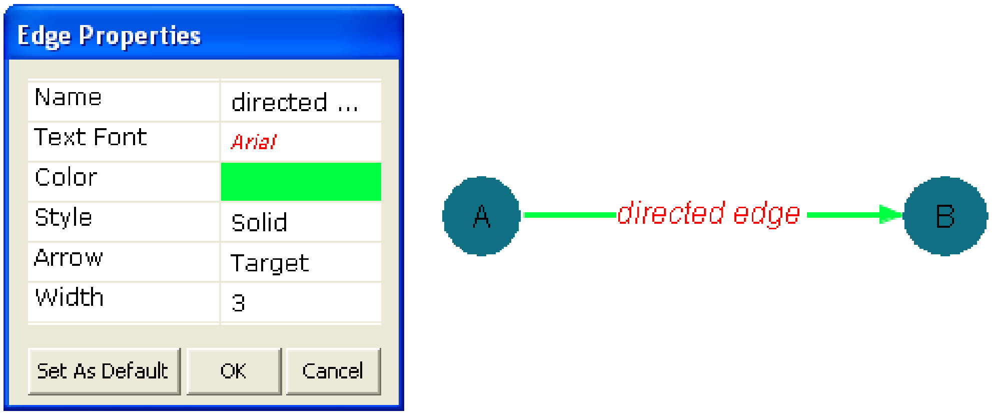


Figure 10 Examples of edges with different styles (left) and available edge types (right)

The appearance of edges can be changed by using the "Edge Properties" window. This window can be opened up by double clicking the edge or by the pop-up menu (Figure 11).

You can change the label, color, style (Solid, Dashed), arrow type (None, Source, Target, Both) and width (thickness) through this window.



Edge Properties

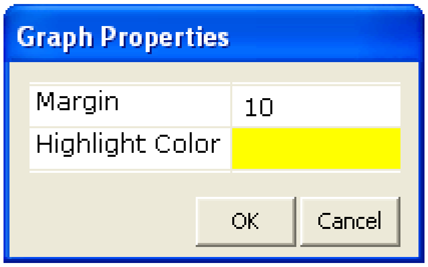
Name	directed ...
Text Font	Arial
Color	🟩
Style	Solid
Arrow	Target
Width	3

Figure 11 Edge properties window or edge inspector

"Set As Default" button is used for setting this edge's properties as default for the edges that are to be created later on.

Graphs

There are certain general properties independent of specific graph objects. These can be adjusted through the "Graph Properties" window (Figure 12).



Graph Properties

Margin	10
Highlight Color	🟨

Figure 12 Graph properties window

Specifically, margins and highlight color (page 14) can be customized through this window. Graph margins are used to separate graph objects from their bounding boxes (such as nodes in a compound graph). Change of this value affects the margins around Chisio Editor drawings when you fit them into the window (page 12).

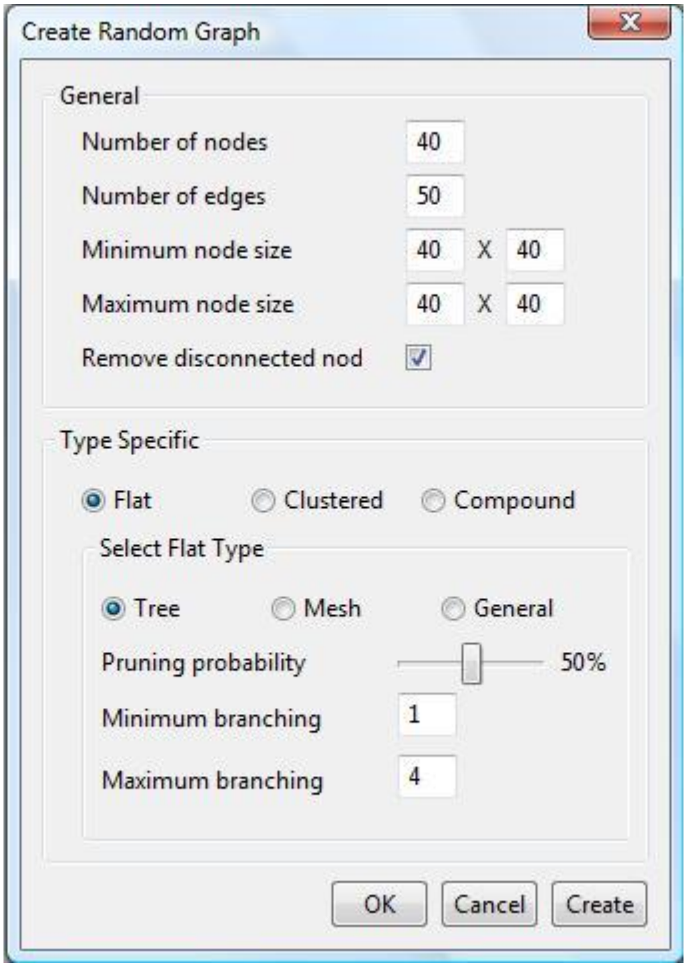
Random graph creation

Chisio Editor has a facility for randomly generating graphs of various classes using "File | Create Random Graph". This pops up a dialog for configuring the random graph to be created. General parameters (number of nodes and edges, minimum and maximum node dimensions, whether disconnected nodes should be removed after creation) and type specific parameters allow the user to configure the random graph to be created.

Three different types of graphs can be created through this facility as described below.

Flat graphs

Random flat (non-clustered, non-compound) graphs can be generated with Chisio Editor (Figure 13). The user has the option to create *tree-like*, *mesh-like* or *general* (has no restrictions) *graphs*



Create Random Graph

General

Number of nodes 40

Number of edges 50

Minimum node size 40 X 40

Maximum node size 40 X 40

Remove disconnected nod ☑

Type Specific

● Flat ○ Clustered ○ Compound

Select Flat Type

● Tree ○ Mesh ○ General

Pruning probability 50%

Minimum branching 1

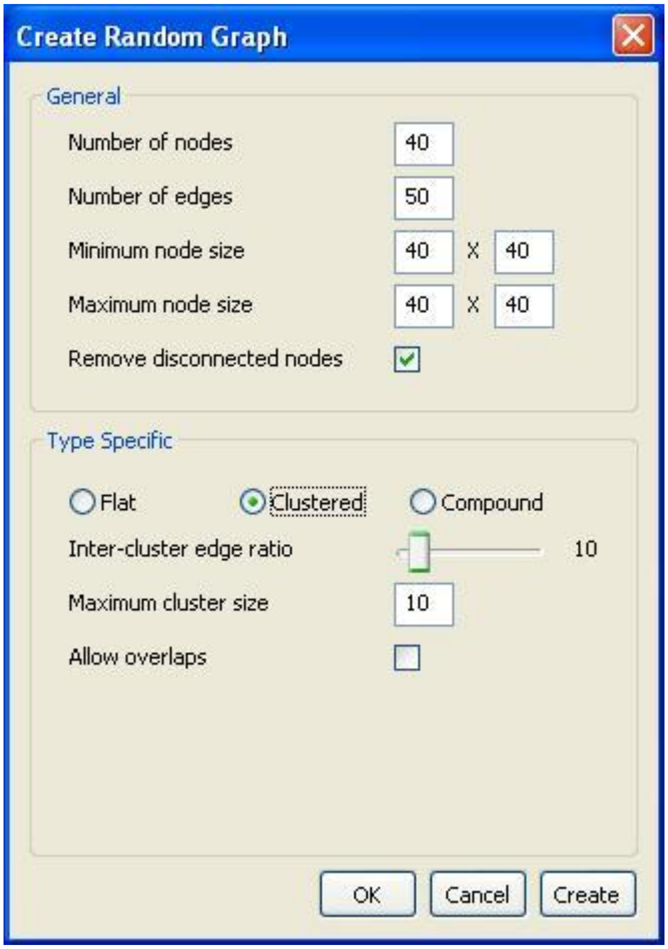
Maximum branching 4

Figure 13 Options for a random flat graph

There are additional parameters for random tree creation. *Pruning probability* is the probability of a node to have children. *Minimum* and *maximum branching* are respectively the minimum and the maximum number of children that a node can have.

Clustered graphs

Random clustered graphs can be generated with Chisio Editor (Figure 14). Through the options dialog, inter-cluster *edge ratio* (the ratio of number of inter-cluster edges to number of all edges) and *maximum cluster size* (maximum number of nodes in a cluster) may be set.


Create Random Graph

General

Number of nodes: 40

Number of edges: 50

Minimum node size: 40 X 40

Maximum node size: 40 X 40

Remove disconnected nodes ☑

Type Specific

○ Flat ● Clustered ○ Compound

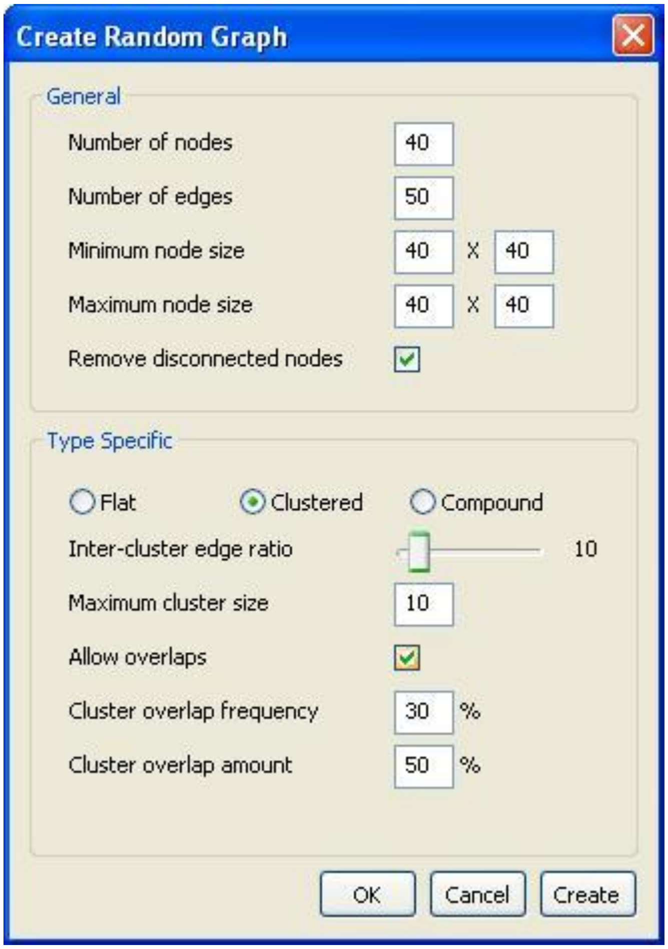
Inter-cluster edge ratio: 10

Maximum cluster size: 10

Allow overlaps □

Figure 14 Options for a random clustered graph where overlaps are not allowed

Moreover, random clustered graphs with overlaps may be created by selecting the *Allow overlaps* box and by setting *Cluster overlap frequency* (percentage of cluster pairs that have common nodes) and *Cluster overlap amount* (percentage of nodes that belong to both of the two overlapping clusters) properties (Figure 15).



Create Random Graph

General

Number of nodes: 40

Number of edges: 50

Minimum node size: 40 X 40

Maximum node size: 40 X 40

Remove disconnected nodes ☑

Type Specific

○ Flat ● Clustered ○ Compound

Inter-cluster edge ratio: 10

Maximum cluster size: 10

Allow overlaps ☑

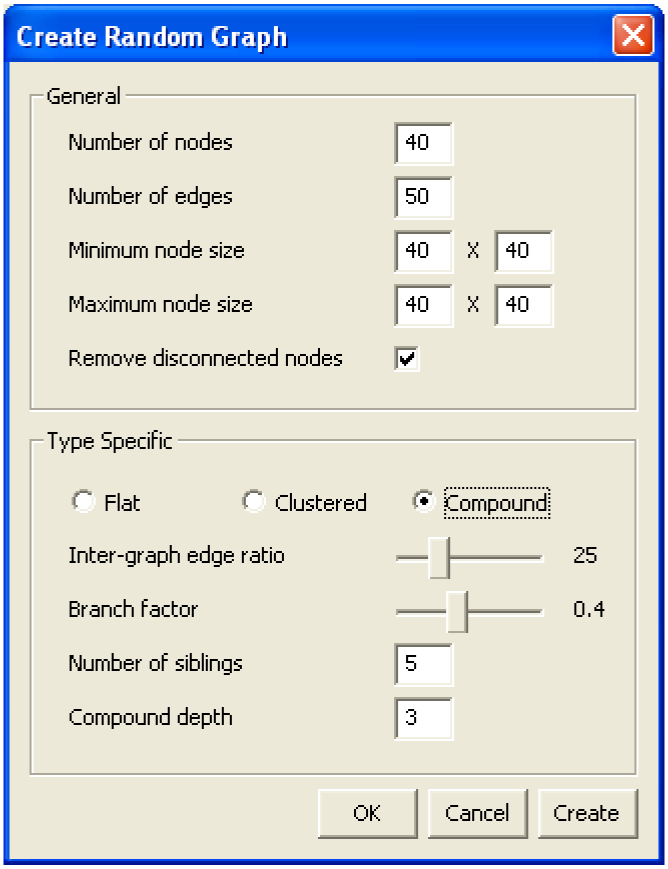
Cluster overlap frequency: 30 %

Cluster overlap amount: 50 %

Figure 15 Options for a random clustered graph with overlaps

Compound graphs

Random compound graphs can be generated with Chisio Editor (Figure 16). Through the options dialog, i*nter-graph edge ratio* (ratio of number of inter-graph edges to number of all edges), *branch factor* (probability of pruning a child in the nesting tree to avoid nesting trees that are too uniform in structure), *number of siblings* (maximum branching or number of children of a node in the nesting tree), and *compound depth* (maximum depth of a node or depth of the nesting tree) may be set.



Create Random Graph

General

Number of nodes 40

Number of edges 50

Minimum node size 40 X 40

Maximum node size 40 X 40

Remove disconnected nodes ☑

Type Specific

○ Flat ○ Clustered ● Compound

Inter-graph edge ratio 25

Branch factor 0.4

Number of siblings 5

Compound depth 3

Figure 16 Options for a random compound graph

Chapter

3

Editing in Chisio Editor

In this chapter, we describe how graphs can be edited interactively in Chisio Editor.

Graphs created with other tools and loaded into Chisio Editor or graphs created in Chisio Editor can be edited interactively as described below.

Chisio Editor Tools

There are several tools to interact with the graphs; namely: *Select Tools*, *Zoom Tools* and *Create Tools*.

Select Tools

The "Select Tool" can be chosen from the top menubar using "Edit | Select Tool" or from the toolbar menu (↖). This tool is used to select nodes, edges and compound nodes. Multiple selections are supported by holding the Shift or the Control key while left-clicking. When you select an object, handle points are drawn around the object.

Handle points for nodes are filled. This means that, node can be resized. Handle points for compound nodes are empty as they cannot be resized (Figure 17).

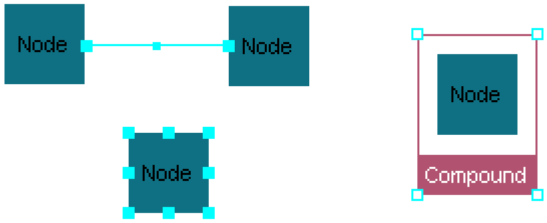


Figure 17 Example selection handles for all types of graph objects

When you press the left mouse button on the drawing canvas outside the boundaries of any graph objects, and drag your mouse, the select tool is automatically switched to the "Marquee Selection Tool". When you release the mouse button, all simple and compound nodes and edges that are *completely* included in the marquee selection area is selected (Figure 18).

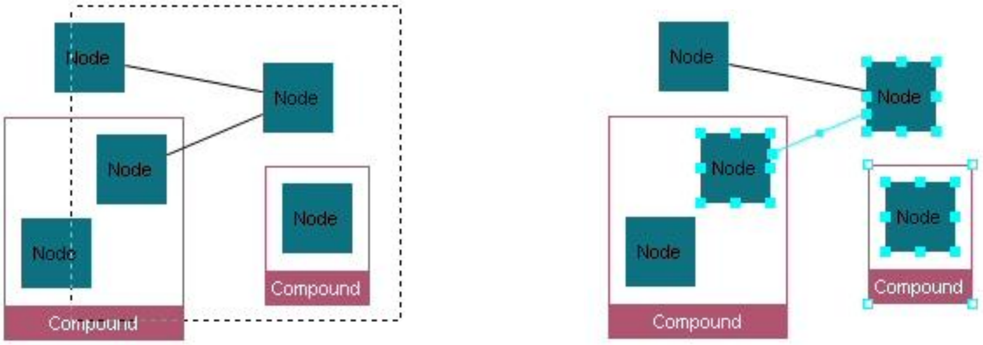


Figure 18 Example use of the Marquee Selection Tool

Zoom Tools

Chisio Editor has several zooming facilities: marquee zoom, zoom-in, zoom-out, zoom to specified level, and fit-in-window.

The "Marquee Zoom Tool" can be chosen from the top menubar using "Edit | Marquee Zoom Tool" or from the toolbar menu (🔍). This tool is used to zoom into a specified rectangular area of the graph (Figure 19 and Figure 20).

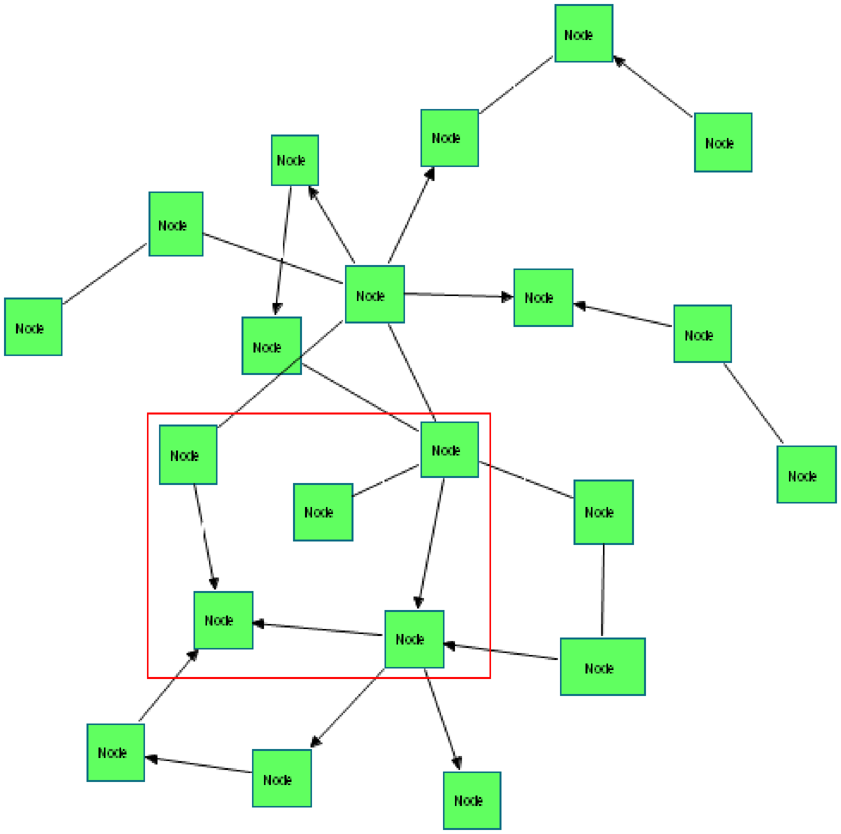


Figure 19 Before marquee zoom to the area specified by the red rectangle

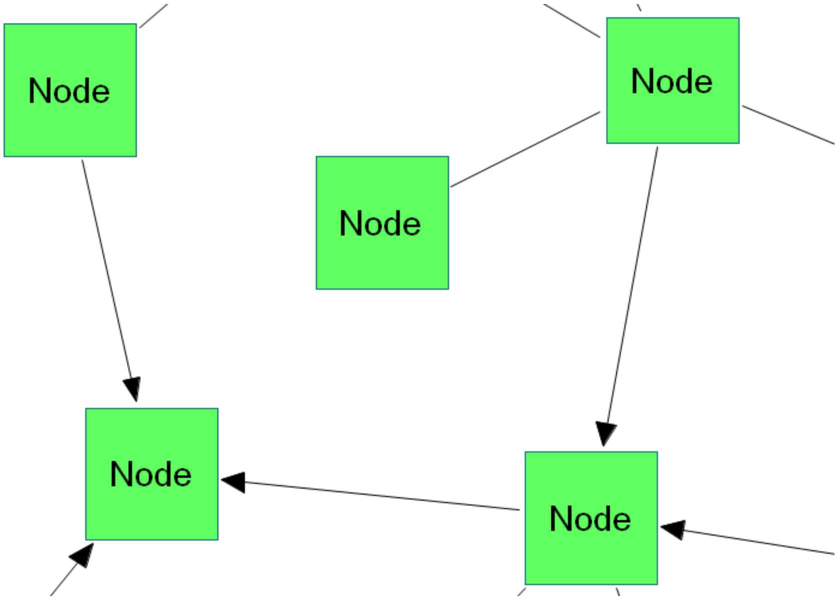


Figure 20 After marquee zooming into the area as described by Figure 19

Other zoom operations like zoom-in (), zoom-out () and zoom to specified level are also supported. These operations can be found in the top menubar under "View | Zoom" and in the toolbar menu. In addition, graph popup menu provides zoom-in and zoom-out capabilities (Figure 21).

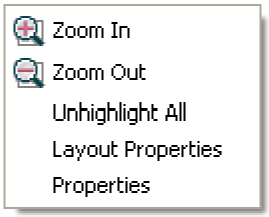


Figure 21 Graph pop-up menu includes zoom-in and zoom-out operations

Fit-in-window operation is another useful zoom operation, which shows the whole graph in the window by properly scaling it. You can fit your graph in window from the top menu bar using "View | Fit in window" or from the toolbar menu ().

Changing Topology

You can interactively change the topological information of your graphs. You can create and delete graph objects easily. In addition, nodes and compounds can be transferred from one graph (root or child) to another (root or child). Finally edges can be reconnected by changing their source and/or target nodes.

Creating Graph Objects

Creating a new node or a new compound can be done from the top menubar under the "Edit" menu. Also using the toolbar menu, you can choose (create a node) or (create a compound node) items. When you select creation tools, the cursor is changed to (create a node) or (create a compound node), respectively. When you click any place on the drawing canvas, the creation is performed (Figure 22).

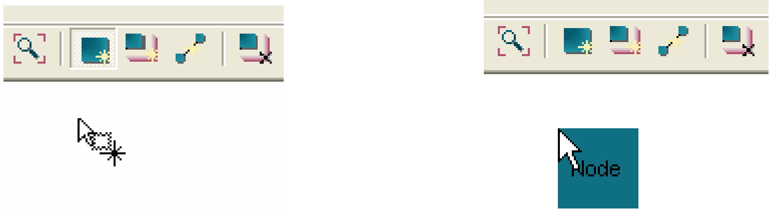


Figure 22 Create a node by simply clicking on the drawing canvas, where you would like your new node to be placed; before the creation (left) and after the creation (right).

Creating a new edge can be done from the top menubar under the Edit menu. Also using the toolbar menu, you can choose item. When you select the edge creation tool, the cursor is changed to . You must click first on the source node for this new edge. Your second click must be on to the target node for this edge to complete the creation of the new edge (Figure 23).

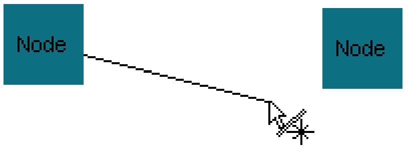


Figure 23 Create an edge by first clicking on the source node and then on the target node

Deleting Graph Objects

To delete nodes, compound nodes or edges, you must select them first. Then you can delete them in one of the following ways:

- Pressing "DEL" button on your keyboard;
- Using the toolbar menu item ✕;
- Using "Edit | Delete Selected" in the top menubar;
- Using "Delete" item under the node popup menu.

Multiple graph objects may be deleted at once using selection of these objects together.

One may also remove existing compound nodes while keeping their child nodes and edges at their current locations. This can be done from the top menubar using "Edit | Remove Compound" or by using the toolbar menu ( ) after selecting the compound node(s) to be removed (Figure 24).

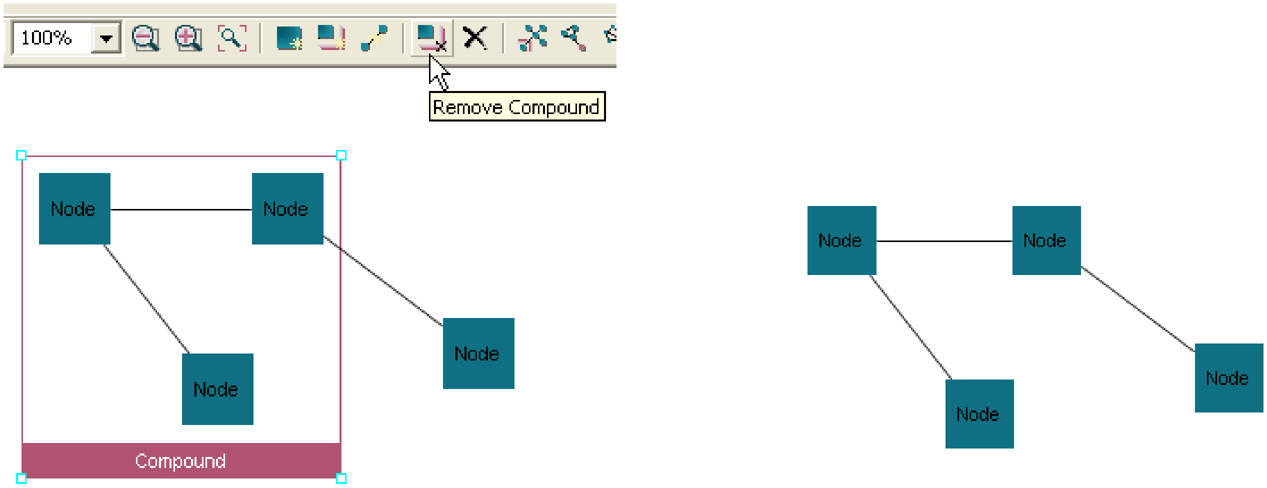


Figure 24 Remove compound; before and after

Transferring Graph Objects

Another way to change the topology is to move a node from one compound node (or root graph) to another compound node (or root graph). In other words, you can change the parent of a node.

This can only be done when the *mode* of the selection tool is "Transfer Mode". You can change the mode to "Transfer Mode" from the toolbar menu (Figure 25) or from the top menubar using "Edit | Transfer Mode".

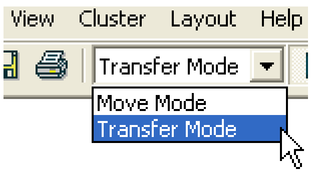


Figure 25 Move/Transfer Mode combo

When in the transfer mode, drag operations might mean change of ownership or transfer. So you can transfer a node/compound node A into another compound node B by dragging and dropping A onto (within the bounds of) the compound node B. When you select a node and move it on to a compound node, compound nodes change background color to cyan to indicate that you will be transferring the selected node into the highlighted compound node should you release it at this location (Figure 26).

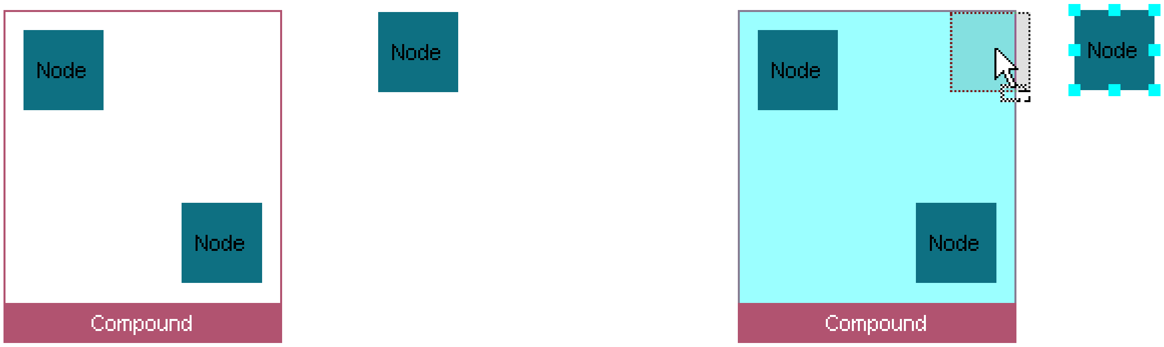


Figure 26 When transferring a node into a compound node, the compound node's background color changes to indicate the transfer; initially (left), when the transfer of the selected node is about to be performed.

Also when you are in the transfer mode, you can clone the selected graph objects by pressing the CTRL key while dragging them. When you press the CTRL key, the cursor changes to indicate the cloning operation (Figure 27). In cloning is not possible in the "Move Mode".

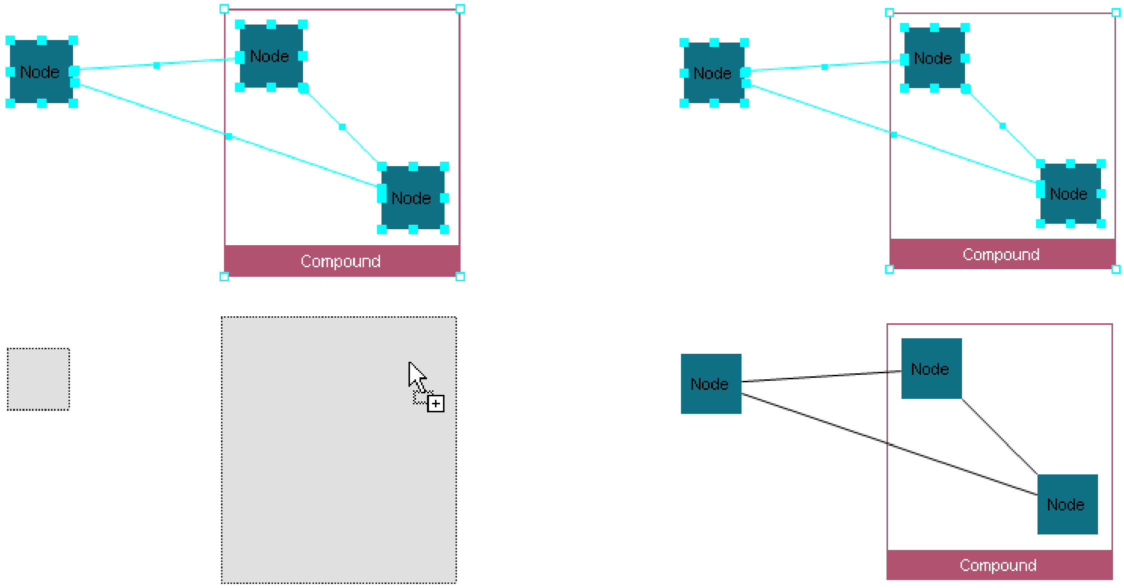


Figure 27 Clone operation; drag selected nodes pressing the CTRL key in the transfer mode and click to clone.

Another transfer operation is via the "Create Compound from Selected" operation in the node pop-up menu. You can create a compound node from a set of selected nodes and compounds easily. This operation is also available through the top menubar using "Edit | Create Compound from Selected" (Figure 28).

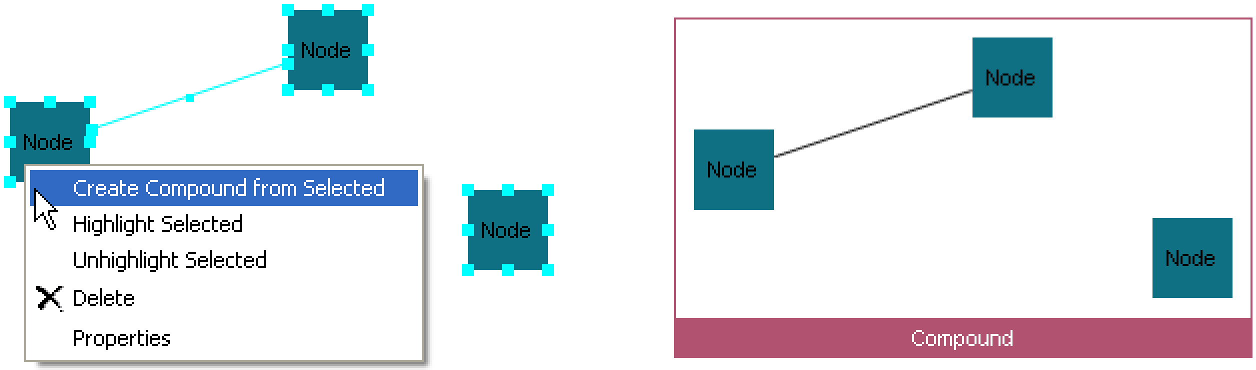


Figure 28 Create compound (right) from selected nodes (left)

Reconnecting an Edge

You can change the source or target of an edge after creation of it. When an edge is selected, its handle points become visible. When you move your mouse to the handle point associated with the target or source node, the cursor is changed into a plus icon. You can click, drag and drop this point onto its new target or source. While dragging, the cursor indicates potential new target or source for this edge (Figure 29).

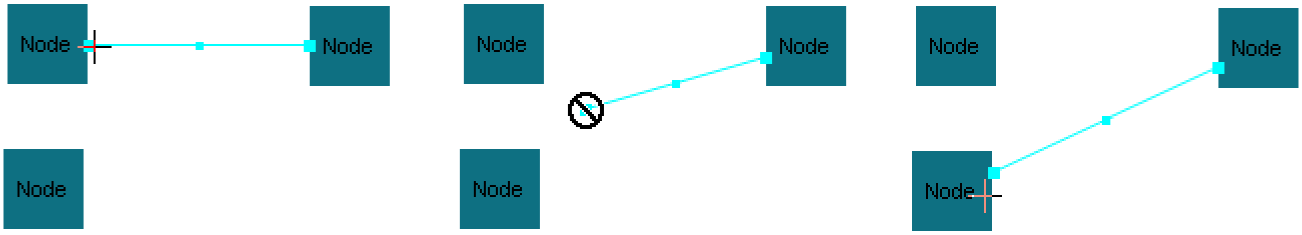


Figure 29 Edge reconnection; after selection of the edge (left), during dragging (middle) and after reconnection (right).

Changing Geometry

You can change the geometry of nodes by moving or resizing them. The geometry of the edges, on the other hand, can be changed by re-routing them by creating new bend points, by moving or deleting existing ones. The layout operation also changes the geometry of graph objects as will be explained later on.

To move a node, first you must select it. Then click anywhere on the node and start dragging; you will see a "ghost shape" of your node as you drag it. You can drop it to the location you like. Upon release of the mouse the operation is completed (Figure 30). We assume the "Move Mode" here; during "Transfer Mode" this might mean a transfer operation as explained earlier in Section on Transferring Graph Objects on page 15.

Figure 30 Move a node by selecting and dragging it

To resize a node, first you must select it upon which its handle points will be visible. By using these points, you can resize the node. Just select a handle point and move it; a ghost shape will appear indicating the new size of the node upon release of the mouse (Figure 31).

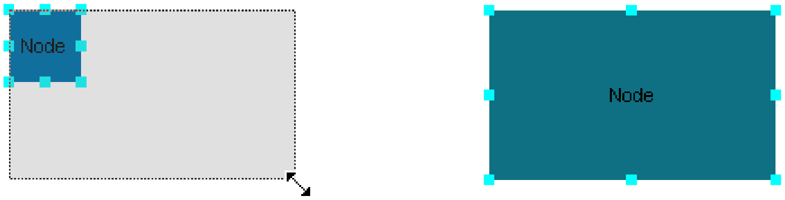


Figure 31 Resize a node; before (left) and after (right)

You can not resize compound nodes as their geometry is auto-calculated by the geometry of its contents as explained earlier. This is indicated by empty handles upon selection of compound nodes (Figure 32).

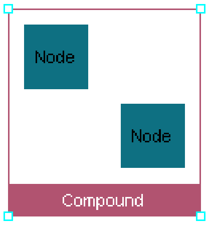


Figure 32 Compound nodes are automatically resized.

Both move and resize operations may be applied to multiple objects simultaneously for convenience.

Chisio Editor supports bend points to freely route edges around other graph objects. You can create, delete and relocate bend-points. When an edge is selected two handles of two different sizes becomes visible. The large ones (except for the end points of the edge used for reconnecting edges) correspond to the bends of the edge. The smaller ones, on the other hand, can be clicked on and dragged to create new bend points (Figure 33). Initial position of the newly created bend point will be right in between the two nearest existing bends on the edge but it will move with the mouse.

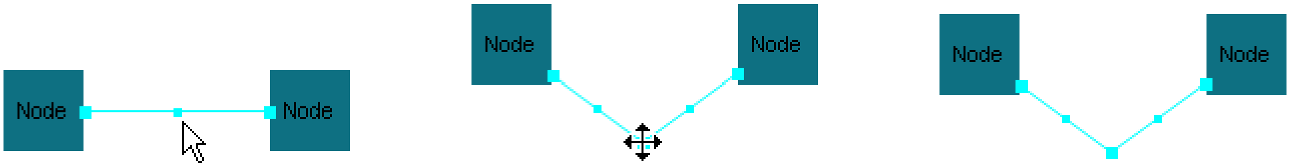


Figure 33 Bend point creation; after selection (left), during drag of the small handle (middle), and upon completion with the release of the mouse (right)

You can relocate existing bend points by dragging them to new locations as you like. If the final location of a bend point upon release of the mouse is to be aligned with its current neighbors of the bend point to make a straight line (or very near such a position), then the bend point is automatically deleted (Figure 34).

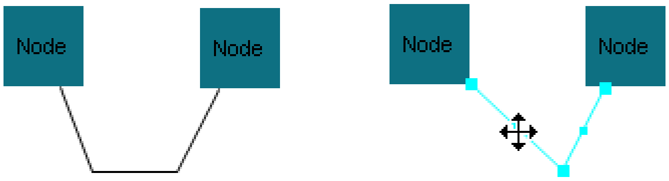
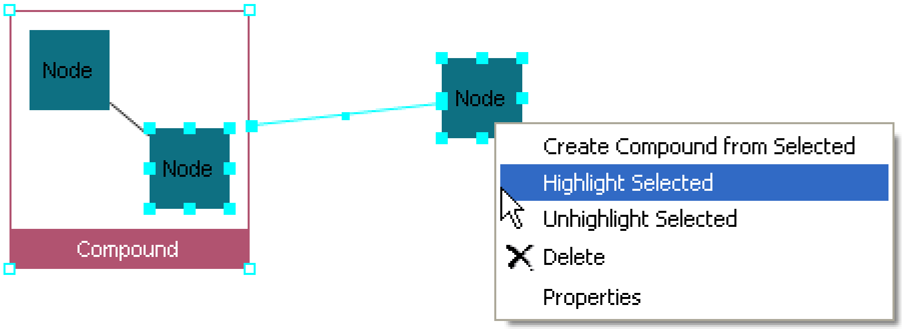


Figure 34 Bend point deletion; initially the edge has two bends (left); upon drag and release of the left bend, it is deleted to leave the edge with only one bend (right).

Highlighting

Graph objects may be highlighted using the highlight facility to differentiate them from others (e.g. a path, a cycle or a particular subgraph of interest). Selected objects may be highlighted through the "Highlight Selected" item under the node pop-up menu or using "View | Highlight Selected" in the top menubar. Multiple objects may be highlighted together (Figure 35).



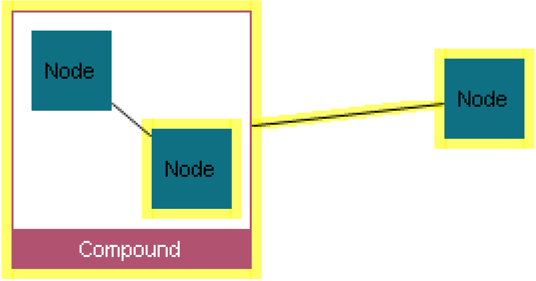


Figure 35 Highlighting objects

You can change the highlight color from the graph pop-up menu. Thus different objects may be highlighted with different colors if desired.

You can remove highlights of all objects by selecting "Unhighlight All" item in the top menubar or from the graph pop-up menu.

Alternatively, a subset of currently highlighted objects may be unhighlighted by first selecting them and then choosing "View | Unhighlight Selected" in the top menubar or "Unhighlight Selected" in the node pop-up window.

Cluster IDs

You can group or cluster a set of nodes in a Chisio graph. To do that, simply select the nodes you would like to assign a new cluster to, and then select "Cluster | Assign Selected to New Cluster" from the top menubar (Figure 36). This operation gives a new, unused cluster ID to selected nodes.

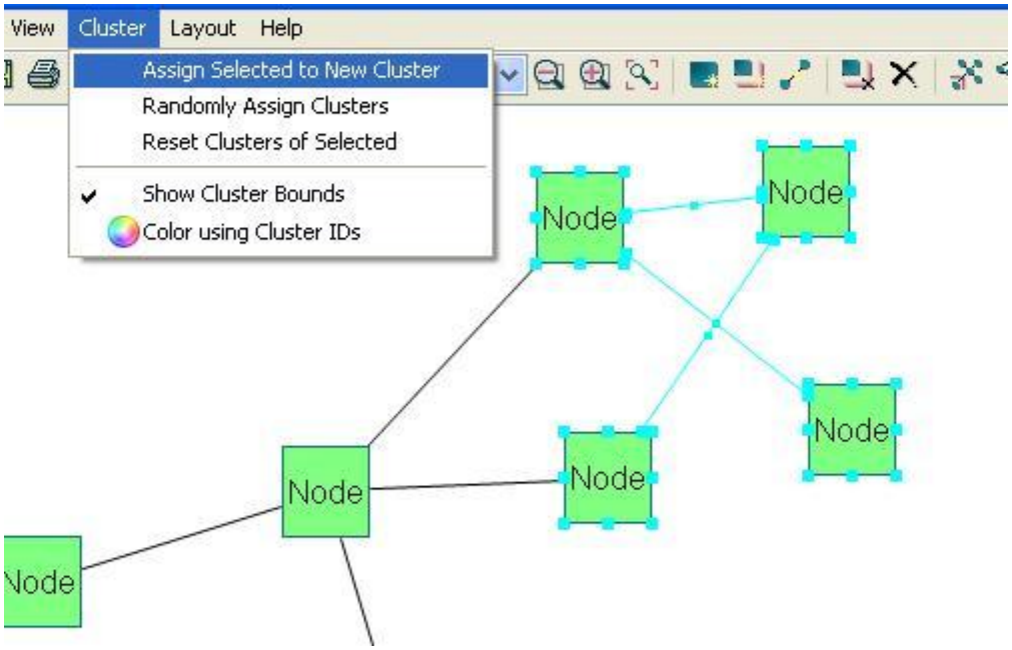


Figure 36 Creating a new cluster

Cluster Bounds

You can show cluster bounds by enabling "Cluster | Show Cluster Bounds" from the top menubar. For every cluster, a polygon with different background color will be drawn (Figure 37).

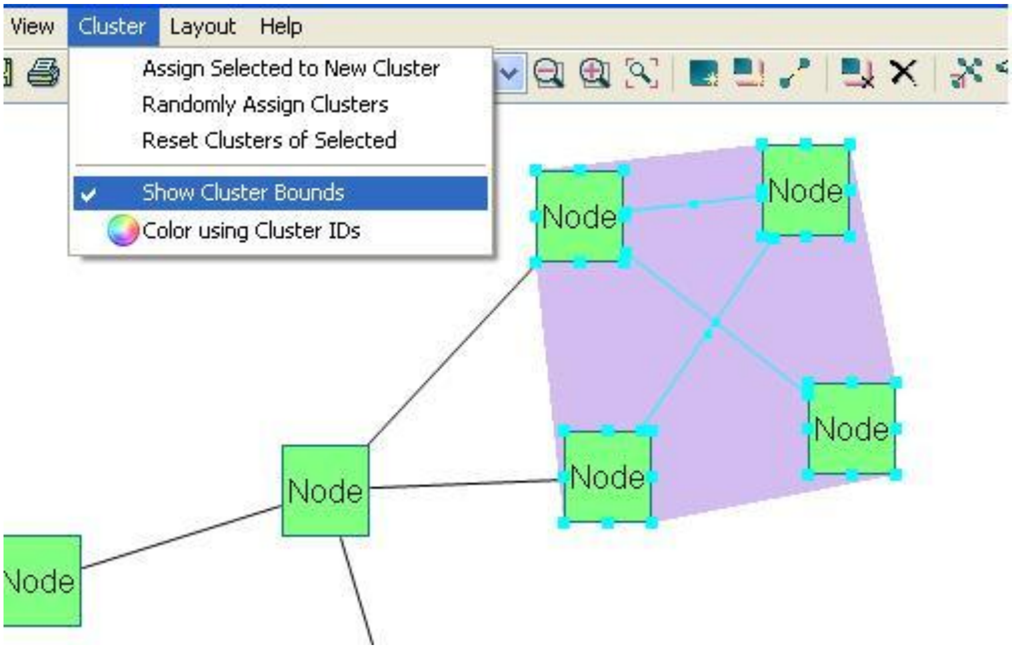


Figure 37 Show cluster bounds

You can also hide cluster bounds by disabling "Cluster | Show Cluster Bounds" from the top menubar (Figure 38). Cluster polygons will disappear.

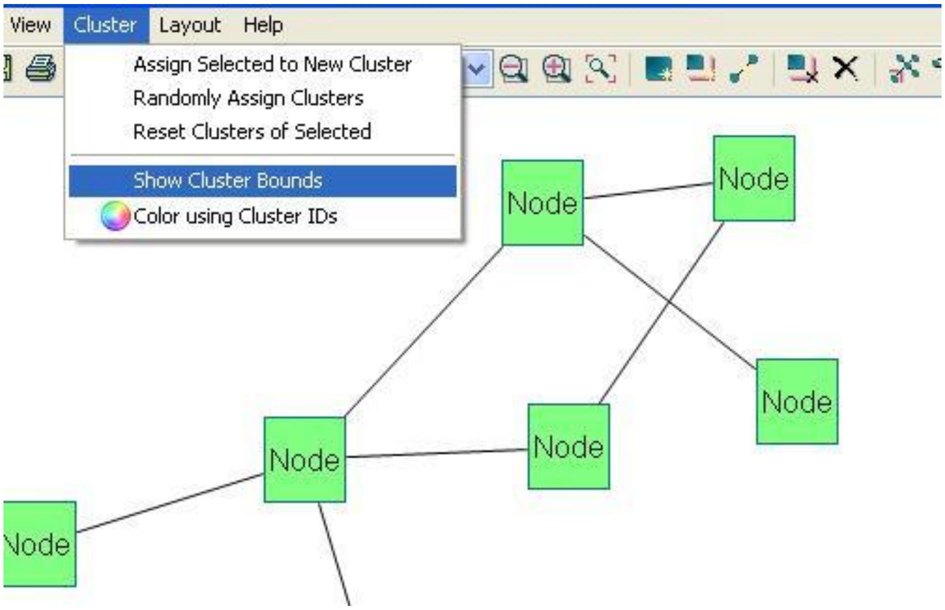


Figure 38 Hide cluster bounds

Random cluster assignment

You may also assign random clusters to the current graph using "Cluster | Randomly Assign Clusters" from the top menubar. This operation re-colors the graph using cluster IDs that are randomly assigned (Figure 39).

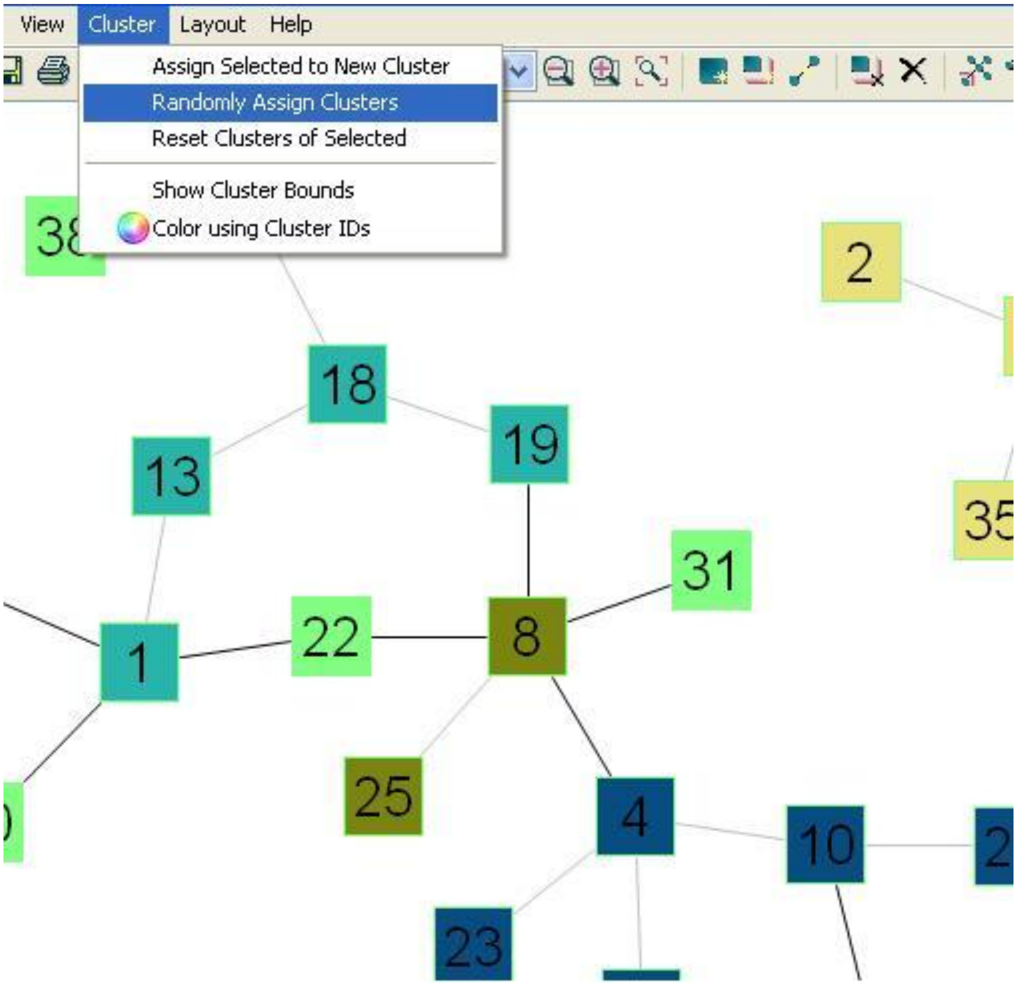


Figure 39 Randomly assigning clusters

You may enable "Cluster | Show Cluster Bounds" from the top menubar, in order to see newly assigned cluster's bounds (Figure 40).

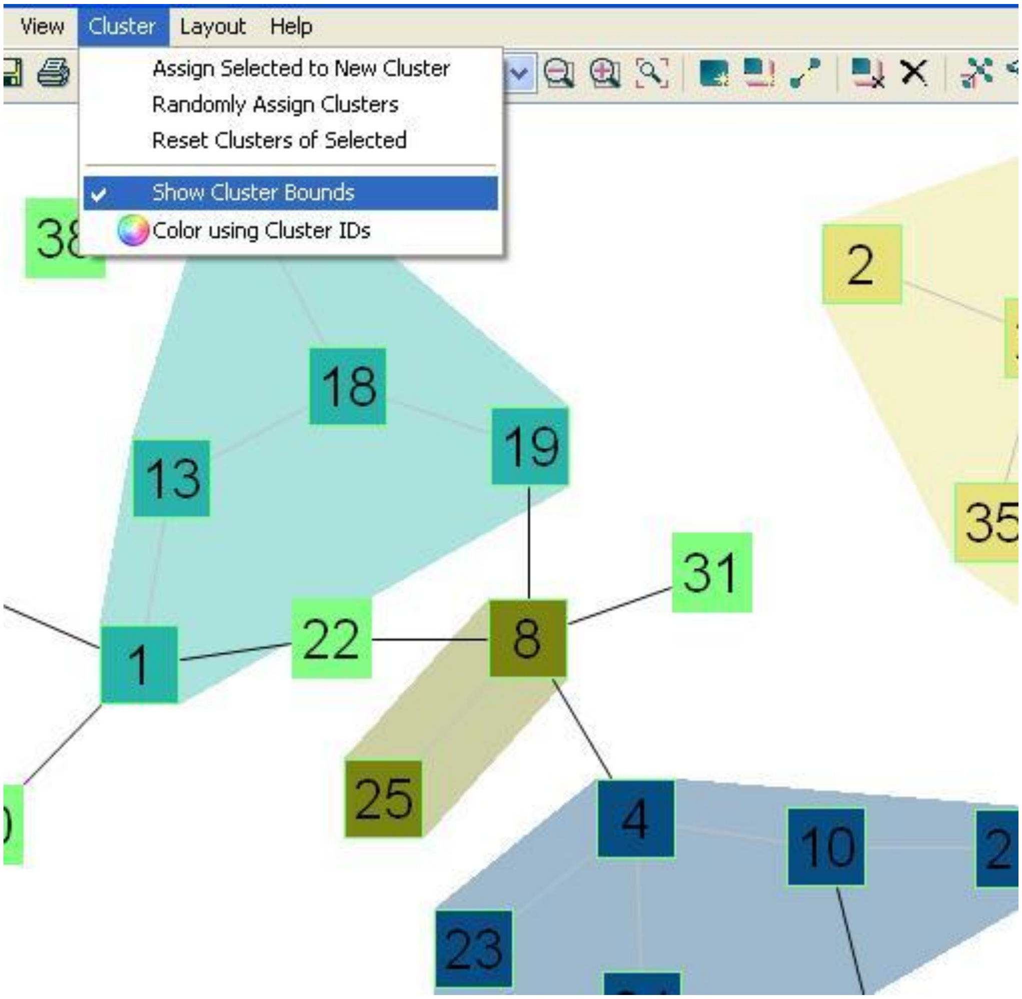


Figure 40 It is optional whether or not to show cluster bounds

You can reset the cluster information of selected nodes. If ClusterID of a node is "0", this means that node does not belong to any cluster. Select the nodes you want to reset their cluster information, and then select "Cluster | Reset Clusters of Selected". When clusters of nodes are reset, involved cluster bounds are updated accordingly (Figure 41). Affected cluster bounds are also updated accordingly.

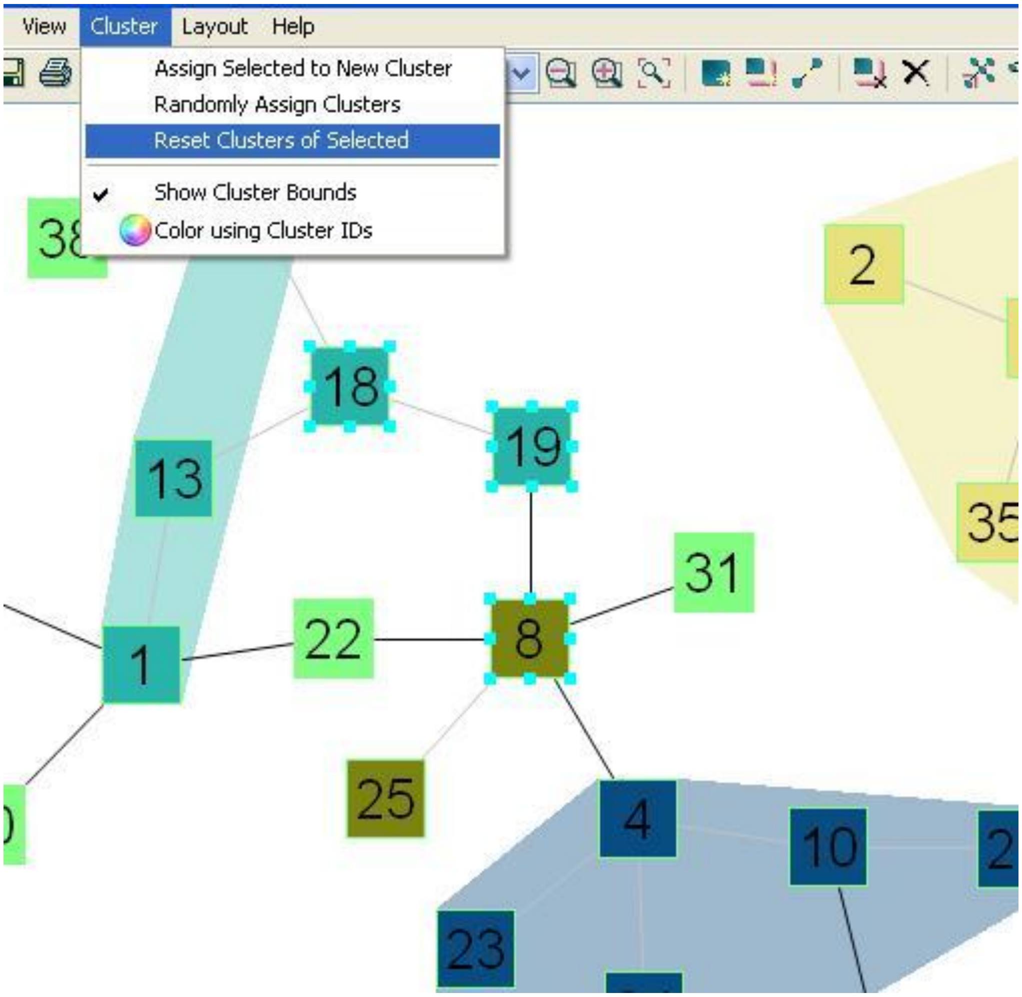


Figure 41 Resetting clusters of nodes 18, 19 and 8 of the graph in Figure 40 results in new cluster bounds.

You can also color-code the nodes in your Chisio graph according to their cluster IDs. Nodes in the same cluster are colored with the same, unique color. In addition, intra-cluster edges are colored to be gray (a little lighter than the default edge color, black, used for inter-cluster edges), so they can be better differentiated from inter-cluster edges. Simply select "Cluster | Color using ClusterIDs" from the top menubar. Figure 42 shows new colors of nodes 18, 19 and 8, cluster information of which were previously reset in Figure 41. Moreover, other clustered nodes are also re-colored with random colors.

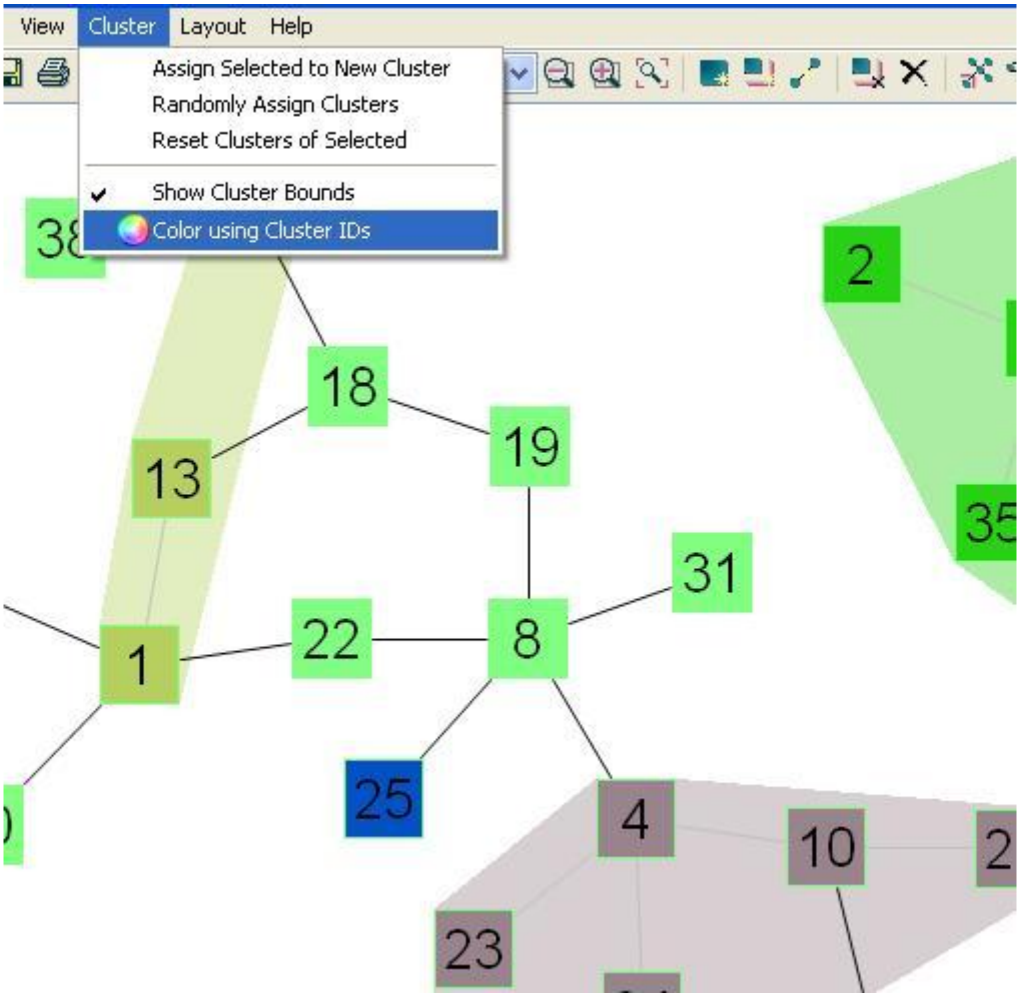


Figure 42 Coloring uniquely with respect to Cluster IDs

Multiple Cluster IDs

As stated earlier, nodes may belong to multiple clusters. Figure 43 shows an example graph, where nodes 1 and 2 belong to cluster with ID 1, nodes 3 and 4 belong to cluster with ID 2, and nodes 5 and 6 belong to cluster with ID 3. The node in the center belongs to all 3 clusters. Since, the node in the center belongs to multiple clusters, cluster bounds also overlap.

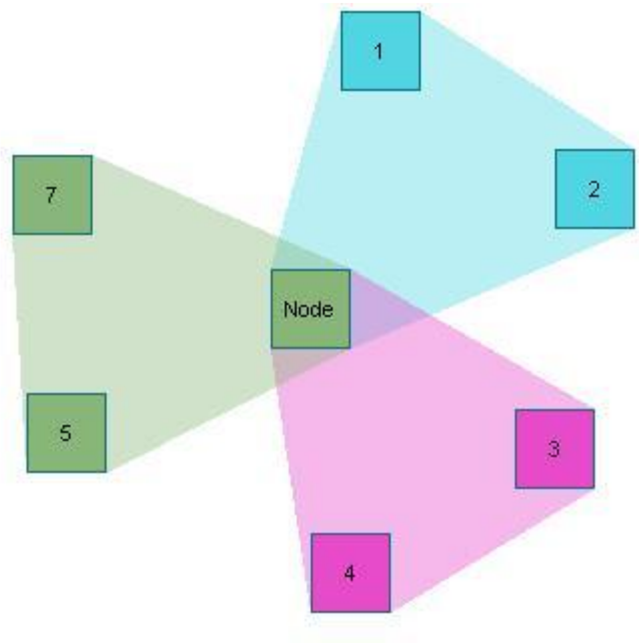
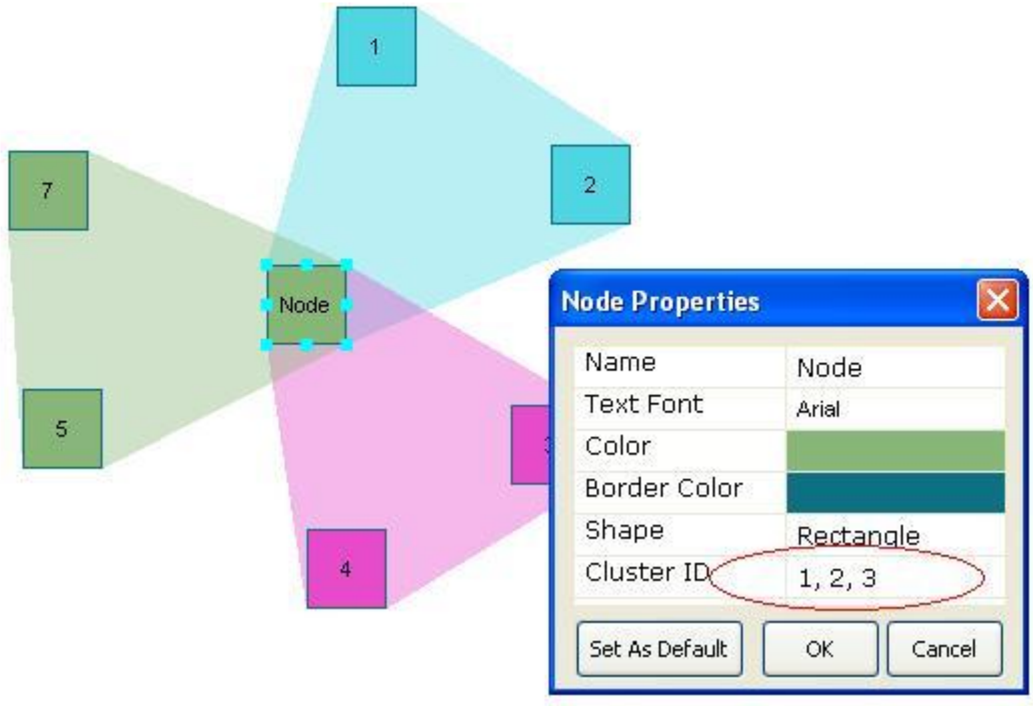


Figure 43 An example graph with overlapping clusters

From the node inspector, you can find out cluster IDs of the clusters that a particular node belongs to (Figure 44).

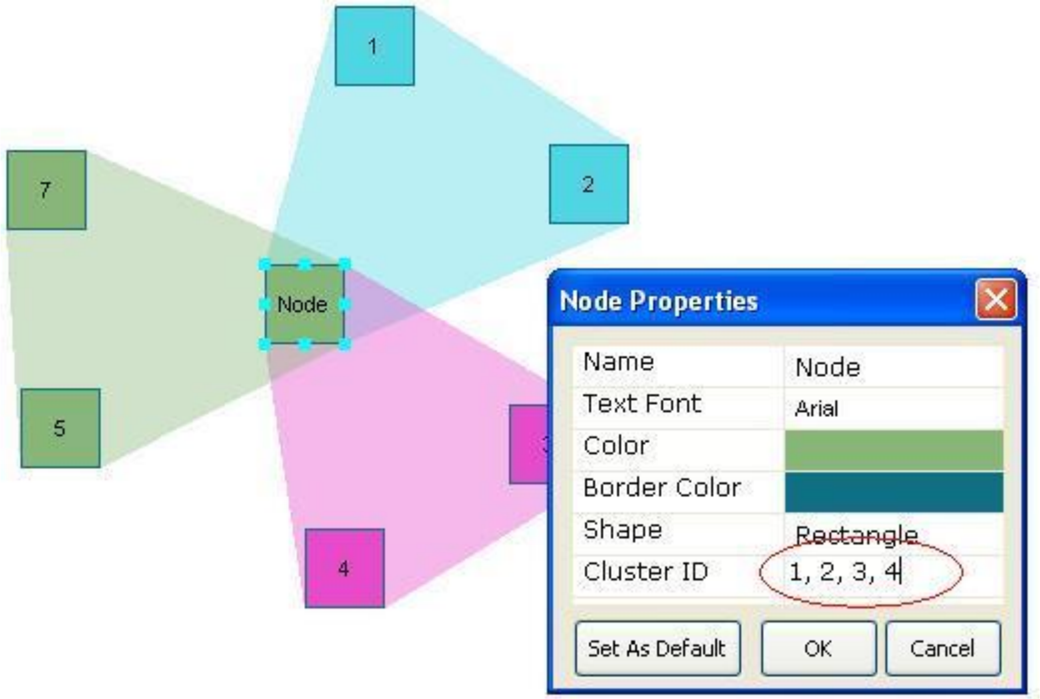


Node Properties

Name	Node
Text Font	Arial
Color	🟩
Border Color	🟦
Shape	Rectangle
Cluster ID	1, 2, 3

Figure 44 Cluster ID(s) of a node are separated by commas; "0" (zero) signifies no clustering.

You may edit cluster IDs of the node with the node inspector. Simply enter cluster ID's using commas or spaces (Figure 45). A new cluster will be created, if you enter a cluster ID that was not used before in graph.



Name	Node
Text Font	Arial
Color	🟩
Border Color	🟦
Shape	Rectangle
Cluster ID	1, 2, 3, 4

Figure 45 Edit cluster information

Persistent Storage

You can save your graphs into a file for later use. Graphs are saved in GraphML file format [1], which is an XML-based graph format. In addition, your own files in GraphML format, possibly created by other programs may be loaded up into Chisio Editor. Drag and drop is supported for convenient file loading as well.

Every property of a graph object is written into the file in a regular format. Below you can find parts of a GraphML file for specific examples.

Example 1: A highlighted node (Figure 46). Multiple cluster information of a node is represented by separating cluster IDs with a vertical bar. The current color of the polygon representing cluster bounds is not persisted but re-assigned each time the graph is loaded.

```
...
<node id="n0">
  <data key="x">108</data>
  <data key="y">90</data>
  <data key="height">40</data>
  <data key="width">40</data>
  <data key="color">14 112 130</data>
  <data key="borderColor">14 112 130</data>
  <data key="text">Node</data>
  <data key="textFont">1|Arial|8|0|WINDOWS|1|-11|0|0|0|0|0|0|0|1|0|0|0|0|Arial</data>
  <data key="textColor">0 0 0</data>
  <data key="clusterID">4|2</data>
  <data key="highlightColor">255 255 0</data>
  <data key="shape">Rectangle</data>
</node>
...
```

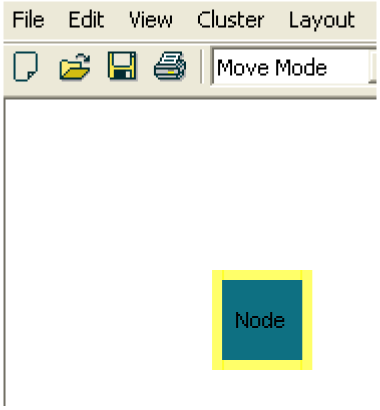


Figure 46 A Highlighted node for example GraphML format

Example 2: A dashed edge (Figure 47)

```
...
<edge id="e0" source="n0" target="n1">
  <data key="color">255 128 192</data>
  <data key="text">edge</data>
  <data key="textFont">1|Comic Sans MS|12|2|WINDOWS|1|-16|0|0|0|400|1|0|0|0|3|2|1|66|Comic
Sans MS</data>
  <data key="textColor">0 0 255</data>
  <data key="style">Dashed</data>
  <data key="arrow">Target</data>
  <data key="width">2</data>
</edge>
...
```

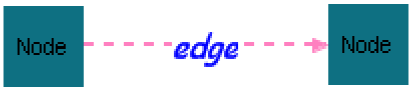


Figure 47 An edge for example GraphML format

Static Images and Printing

You may save the current drawings of your graphs as static images as well. Supported image formats are BMP and JPEG. You have the option of recoding the entire drawing or only the currently viewable part of the drawing (Figure 48).

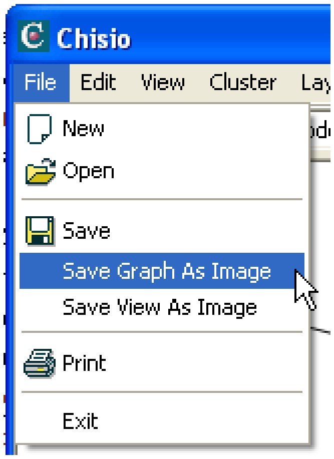


Figure 48 Save as image menu item

In addition, you can print your graphs from the top menubar "File | Print" or the toolbar menu (🖨). Print operation prints the whole graph in regardless of the currently viewable part of the drawing.

Chapter 4

Layout

In this chapter, we present automatic layout operations available in Chisio Editor.

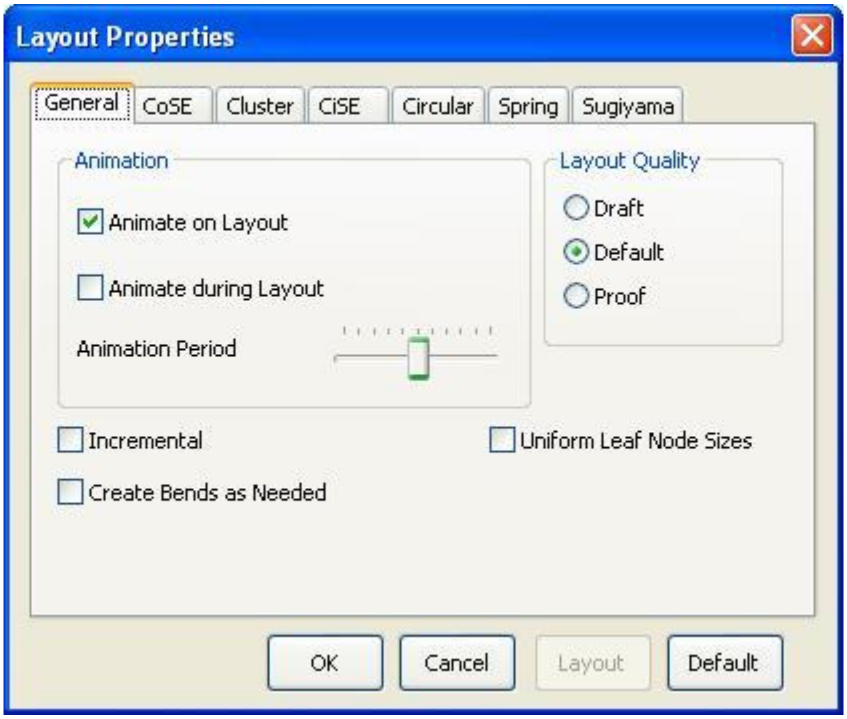
Automatic layout of graphs is extremely important with most graphs as it becomes a cumbersome operation, if not impossible, to manually layout the nodes of a graph and route the edges to produce aesthetically pleasing drawings. Chisio provides a number of popular and useful layout styles as described here.

Each layout style can be customized through the "Layout Properties" window, accessible through "Layout | Properties" in the topmenu bar or under the graph pop-up menu. Each style gets a tab of its own for its available options.

Layout operation is run on a separate thread in Chisio Editor. This facilitates a smooth animation as well as allowing the layout operation to be cancelled using "Layout | Stop Layout" menu item or from the toolbar menu (●).

General Layout Options

Options common to all layout styles are gathered on a separate tab in Layout Properties dialog (Figure 49).



Animation

☑ Animate on Layout

□ Animate during Layout

Animation Period

Layout Quality

○ Draft

● Default

○ Proof

□ Incremental

□ Uniform Leaf Node Sizes

□ Create Bends as Needed

Figure 49 Common layout options

Animate on Layout: Whether the results of layout should be displayed in an animated fashion or not.

Animate during Layout: Whether the results of layout should be displayed as the operation is progressing or not. This is especially useful for debugging purposes for a layout developer.

Animation Period: How often the canvas should be refreshed for animation during layout.

Incremental: Should layout start from current positions (respecting current layout) or not (usually starts from random positioning).

Create Bends as Needed: Should the algorithm create bend points for edges as needed (e.g. for multi-edges).

Layout Quality: Quality of layout; usually the higher the quality, the slower the layout will be.

Uniform Leaf Node Sizes: Whether nodes (except for any compound nodes) are of uniform dimensions or not. If nodes are of uniform dimensions, setting this option might speed up the layout process since, then, force calculations will be based on node centers as opposed to clipping points.

CoSE Layout

CoSE (**Co**mpound graph **S**pring **E**mbedder) layout is an algorithm specifically designed for compound graphs [2]. It has been designed by our group, based on the traditional force-directed layout scheme with extensions to handle multi-level nesting, varying node sizes, and possibly other application-specific constraints.

An expanded node and its associated nested graph are represented as a single entity, similar to a "cart", which can move freely in orthogonal directions (no rotations allowed). Multiple levels of nesting is modeled with smaller carts on top of larger ones.

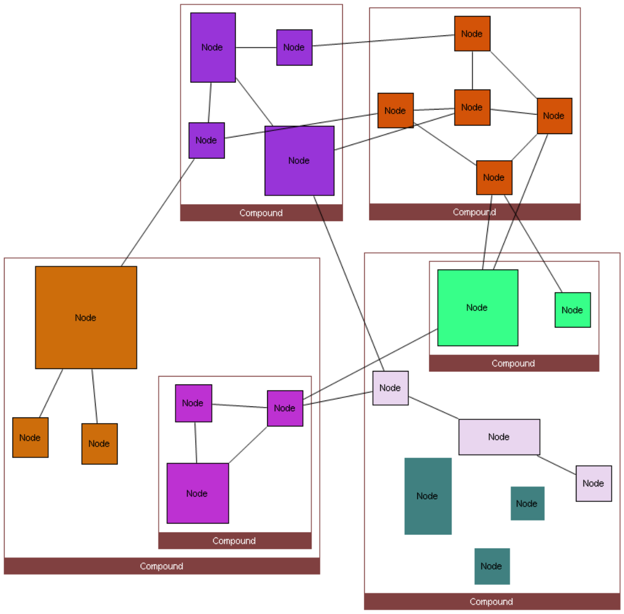


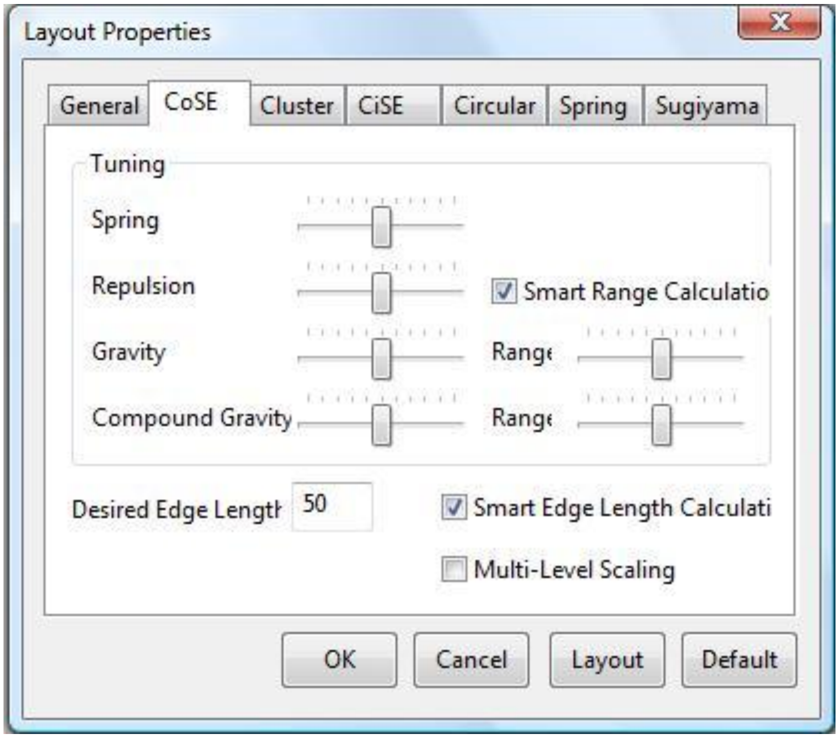
Figure 50 Sample CoSE layout produced by Chisio

CoSE Layout Steps

- Initialize data structures, perform random positioning for static layout and reduce trees for efficiency;
- Lay out the remaining, "skeleton graph" using a modified spring embedder as described earlier;
- Grow trees and continue iterating over the spring embedder model;
- Polish layout during this stabilization phase using a cooling schema for convergence.

Layout Options

CoSE layout options are shown in Figure 51.



Desired Edge Lengtł 50

☑ Smart Range Calculatio

☑ Smart Edge Length Calculati

□ Multi-Level Scaling

Figure 51 CoSE layout options

Spring Strength: Constant for spring forces corresponding to edges; the higher this value is the stronger spring forces will be.

Repulsion Strength: Constant for repulsion forces that are applied to node pairs; the lower this value is, the closer the nodes will be.

Gravity Strength: Factor for the gravity of each graph; the higher this value is, the closer disconnected parts of the graph will be.

Gravity Range: This defines the range, outside which gravity will be effective. The default range is calculated using the size of the graph, with the hope that within the range no gravity is needed. The larger the percentage of this range is the larger the area in which gravity will not be applied.

Compound Gravity Strength: The factor for the gravity inside compound nodes; the higher this value is, the closer disconnected parts of the graph inside compound nodes will be.

Compound Gravity Range: This is the compound version of Gravity Range defined above.

Desired Edge Length: Desired length of an intra-graph edge; inter-graph edges are allowed to be longer as they need to span their owner compound bounds.

Smart Range Calculation: Whether or not FR-Grid Variant [9] method is used during repulsion force calculation. This calculation eliminates unnecessary repulsion calculations and checks each node's repulsion with its immediate nearby neighbors using a grid.

Smart Edge Length Calculation: Whether or not smart calculation of ideal edge length should be performed for inter-graph edges. When this is enabled, we calculate the number of nesting levels each edge spans and calculate the desired edge length accordingly.

Multi-Level Scaling: Whether or not multi-level scaling [10] should be used to speed up the layout process. Use of multi-level scaling captures the structure of the graph better, often yielding better layouts.

Cluster Layout

Many applications group or cluster the nodes in graphs to refer to say molecules with similar functionality or network devices in a LAN. Naturally users of such graphs would like the nodes in the same cluster to be placed near each other. Cluster Layout in Chisio works for graphs with overlapping clusters as well, but the algorithm is still under construction.

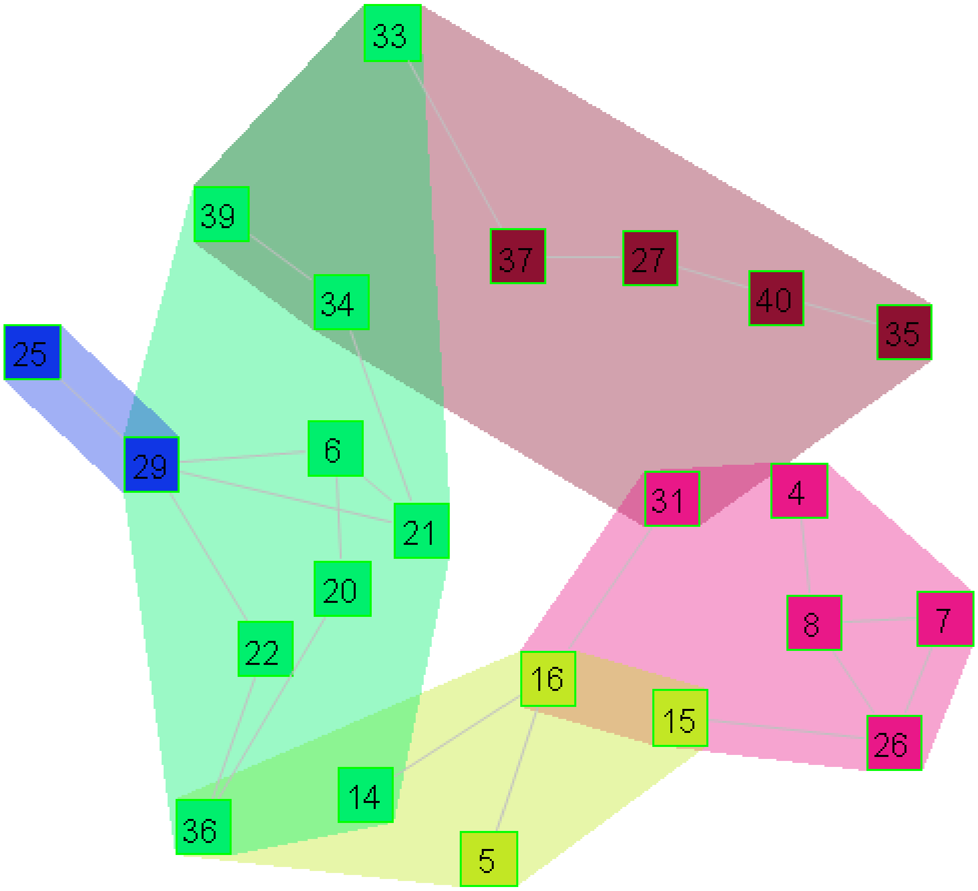


Figure 52 Sample Cluster layout produced by Chisio Editor, where nodes are color-coded by their clusters

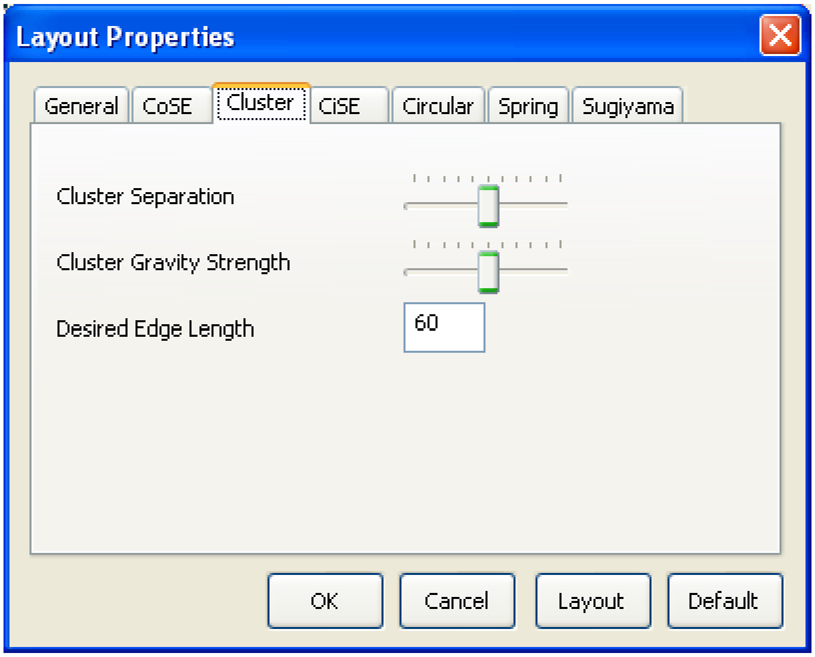
Cluster layout uses the CoSE layout algorithm to establish this as follows:

Cluster Layout Steps

- Create a dummy compound node for each region, where nodes belonging to a single cluster will form one region, whereas nodes in the same cluster as well as a second other cluster will be in a different region;
- Move clustered nodes into these dummy compounds according to their cluster information (cluster IDs);
- Add edge between two dummy compound nodes if corresponding two regions are neighbors (e.g. one region represent cluster 1 and a second one is for clusters 1 & 2). Two regions are considered as neighbors, if one of them belongs to all clusters that the other one belongs to.
- Run CoSE layout for this graph;
- Remove dummy compounds and leave nodes in these compounds in their current absolute locations.

Layout Options

Cluster layout options are shown in Figure 53.



Cluster Separation

Cluster Gravity Strength

Desired Edge Length 60

Figure 53 Cluster layout options

Cluster Separation: Distance between neighboring clusters.

Cluster Gravity Strength: The gravity inside clusters; the higher this value is, the closer the nodes in each cluster will be.

Desired Edge Length: Desired length of an intra-graph edge; inter-graph edges are allowed to be longer as they need to span their owner compound/cluster bounds.

CiSE Layout

A popular way to draw clustered graphs is in a circular fashion. In other words, a circle of appropriate size is created for each cluster and the nodes in that cluster are placed around this circle trying to minimize edge crossings. Circular layout algorithms address the issue of placing nodes of a cluster nicely around a circle, trying to minize the number of crossing and edge lengths; however most of these algorithms do not address the issue of how the cluster graph should be laid out (i.e. how the individual circles should be placed with respect to each other to minimize crossing and edge lengths of inter-cluster edges. CiSE (**Ci**rcular **S**pring **E**mbedder) layout algorithm has been developed by our group and uses a spring embedder based approach to do this [11].

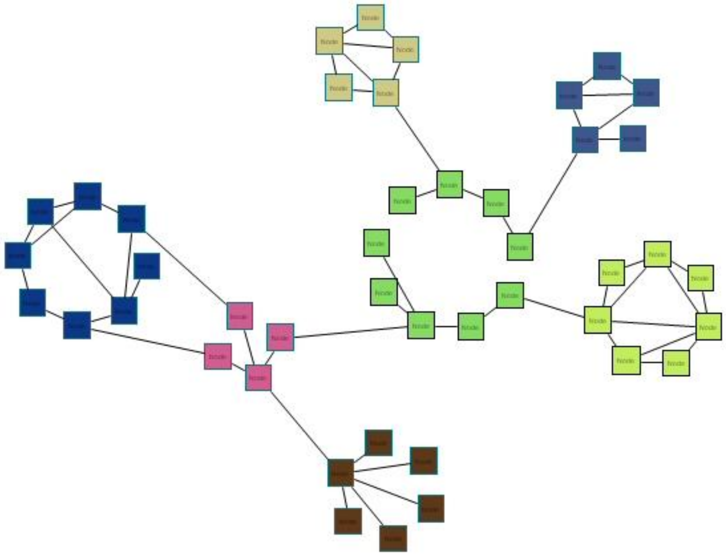


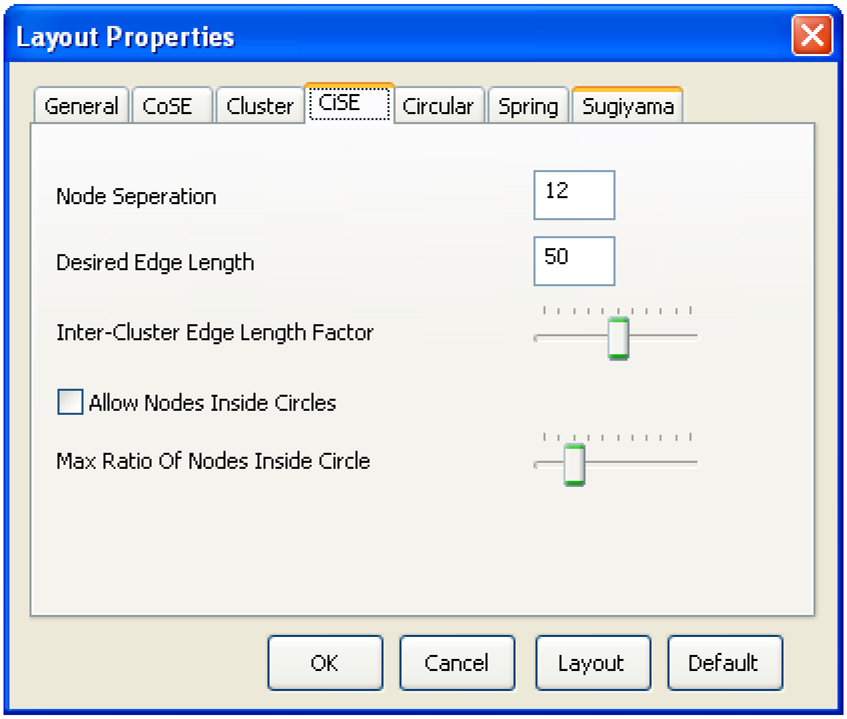
Figure 54 Sample CiSE layout produced by Chisio, where nodes are color-coded by their clusters

CiSE Layout Steps

- Lay out individual clusters using Circular (AVSDF) layout as described on page 33;
- Do an initial placement of the cluster graph using a basic spring-embedder;
- Perform a modified spring embedder, where clusters/circles are allowed to move and rotate;
- Perform a modified spring embedder, where nodes in each cluster are also allowed to move (swap).

Layout Options

CiSE layout options are shown in Figure 55.



Node Seperation 12

Desired Edge Length 50

Inter-Cluster Edge Length Factor

□ Allow Nodes Inside Circles

Max Ratio Of Nodes Inside Circle

Figure 55 CiSE layout options

Node Separation: Distance between neighboring nodes in a cluster.

Desired Edge Length: Desired length of an intra-graph edge.

Inter-Cluster Edge Length Factor: The proportion of inter-cluster edge length to regular edges. The higher this value is over 1.0, the longer inter-cluster edges will be with respect to intra-cluster edges.

Allow Nodes Inside Circles: Whether or not nodes should be placed inside the circles if they are unlikely to introduce new edge crossings to save space (Figure 56).

Max Ratio of Nodes Inside Circles: Maximum ratio of the nodes to be placed inside each circle to the nodes on the circle.

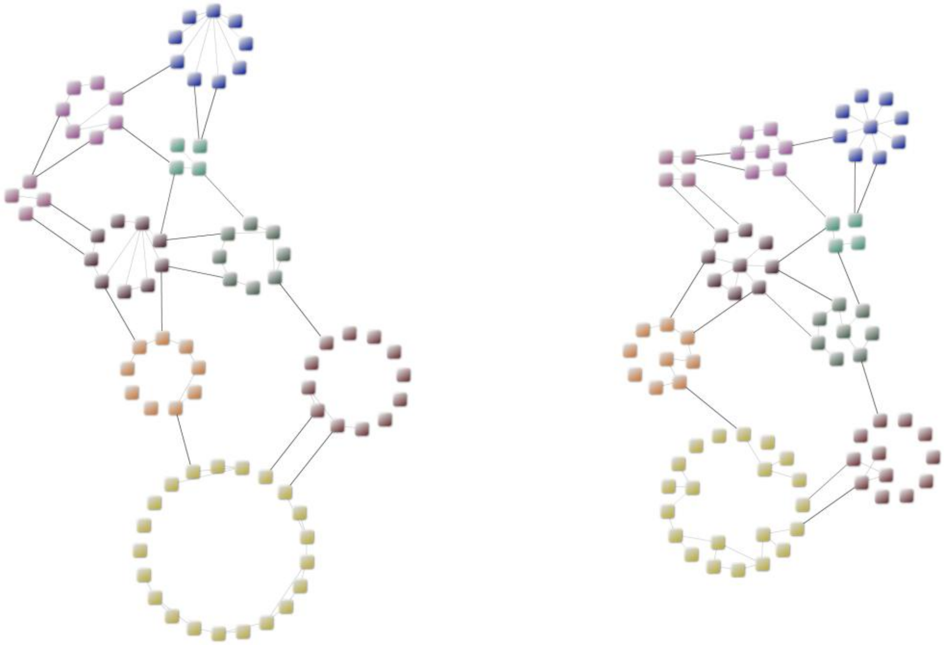


Figure 56 An example where allowing nodes inside circles allows us to save significant space, reducing drawing area.

Circular Layout

This layout algorithm can be used as the base of such a clustered graph layout algorithm. It is based on that of He and Sykora [3] and places all nodes of the current graph around a single circle, taking into account non-uniform node dimensions and desired node separation parameter.

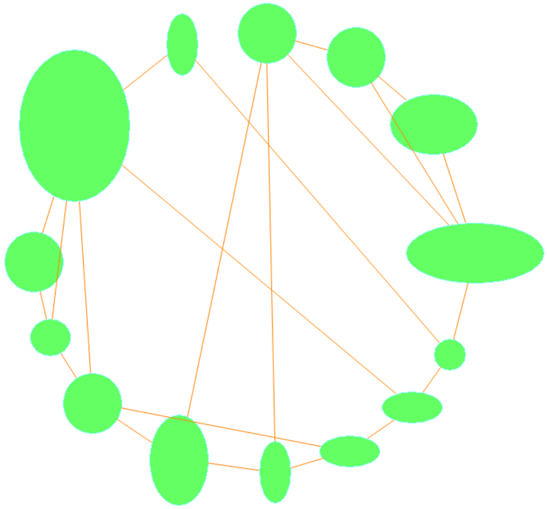


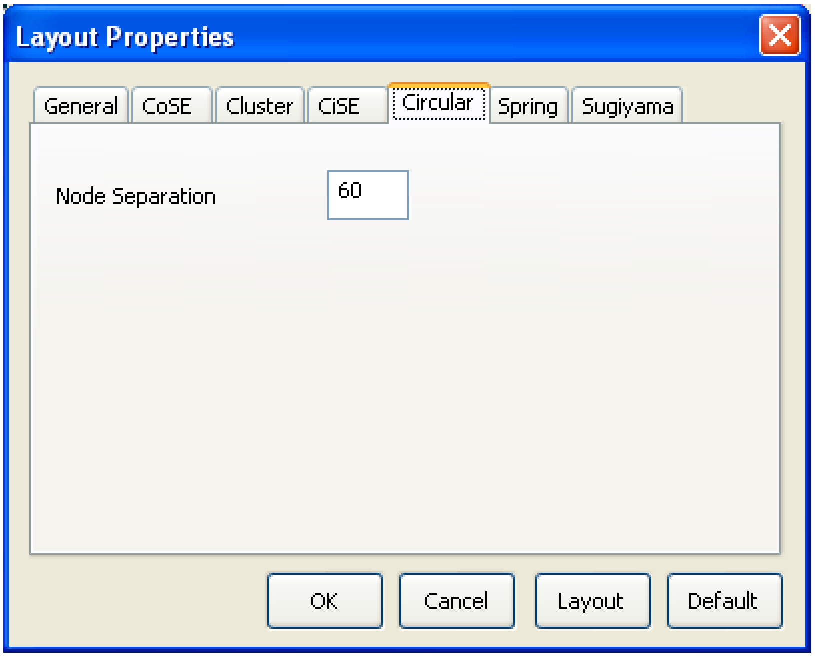
Figure 57 Sample Circular layout

Circle Layout Steps

- Radius of the circle is calculated by using the total number of nodes, their dimensions and desired spacing between neighboring nodes;
- Each node is located onto the circle by AVSDF (adjacent vertex with smallest degree first) approach;
- Numbers of edge crossings are reduced with a post processing step.

Layout Options

Circular layout options are shown in Figure 58.



Node Separation 60

Figure 58 Circular layout options

Node Separation: Desired distance between two neighboring nodes on the circle

Spring Layout

The spring layout is perhaps the simplest force-directed layout. Here the edges of the network are represented as springs. All the springs have a natural length, measured in the same units as the screen co-ordination system, which they attempt to achieve constantly. If the spring is shorter than its natural length it extends, pushing the nodes at either end of the edge apart. If the spring is longer than its natural length it contracts, pulling the nodes at either end together. The force exerted by the spring is proportional to the difference between its current length and its natural length. Nodes linked together tend to form cluster so a repulsive force is also added. The lengths are changed iteratively to obtain a well spaced out layout by minimizing the total energy.

We use an implementation by the GINY graph library [4], based on the Kamada-Kawai Spring Layout Algorithm [5] for this style.

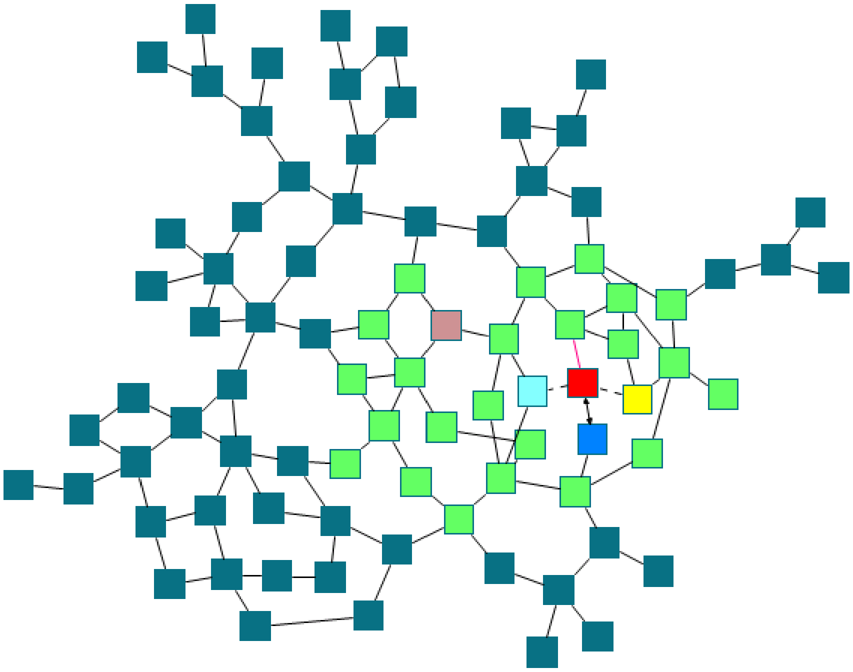


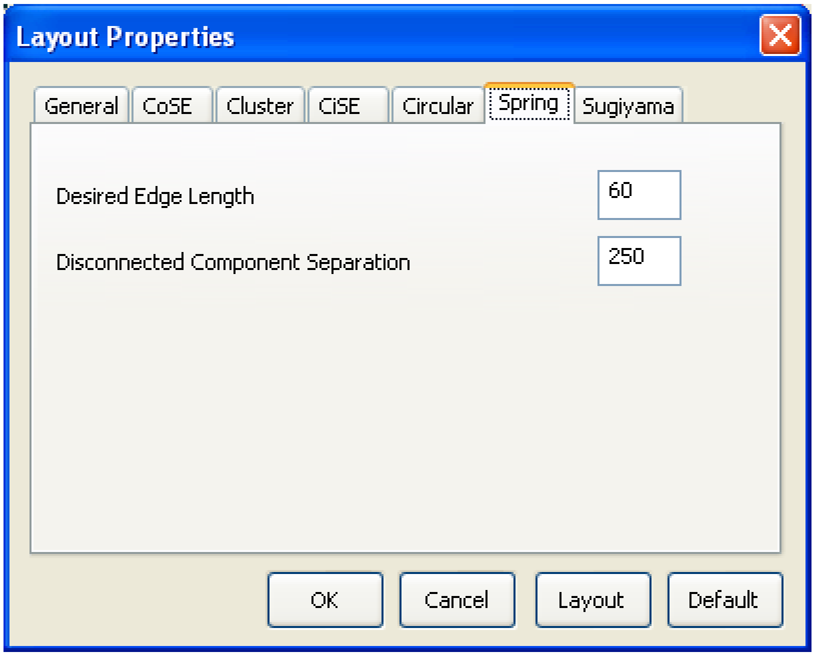
Figure 59 Sample Spring layout produced by Chisio

Understanding the Spring Layout Algorithm

- Compute distances between nodes by using shortest paths algorithm;
- At each iteration, move a node in the direction, which decreases the total energy most.

Layout Options

Spring layout options are shown in Figure 60.



Desired Edge Length: 60

Disconnected Component Separation: 250

Figure 60 Spring layout options

Desired Edge Length: Desired length of an edge

Disconnected Component Separation: Adjusts the separation of disconnected parts of the graph

Hierarchical Layout

The hierarchical layout is bay far the most popular layout style for directed graphs, emphasizing the precedence relationships among graph objects. This layout can represent organizational and information management system dependencies, as well as process models, software call graphs, and workflows.

Our hierarchical layout is an implementation of the Sugiyama algorithm [7], taken from JGraph [5]. We have extended the implementation to support compound graphs and variable node dimensions.

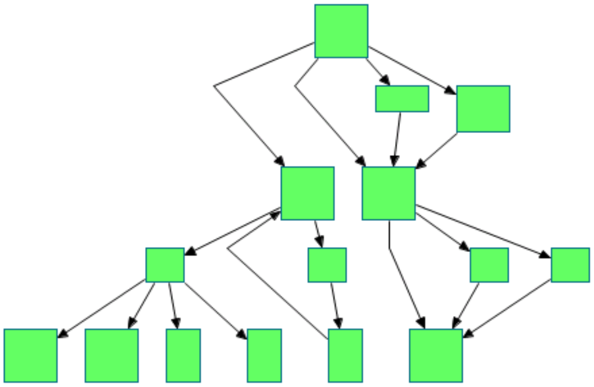


Figure 61 Sample Hierarchical layout

Hierarchical Layout Steps

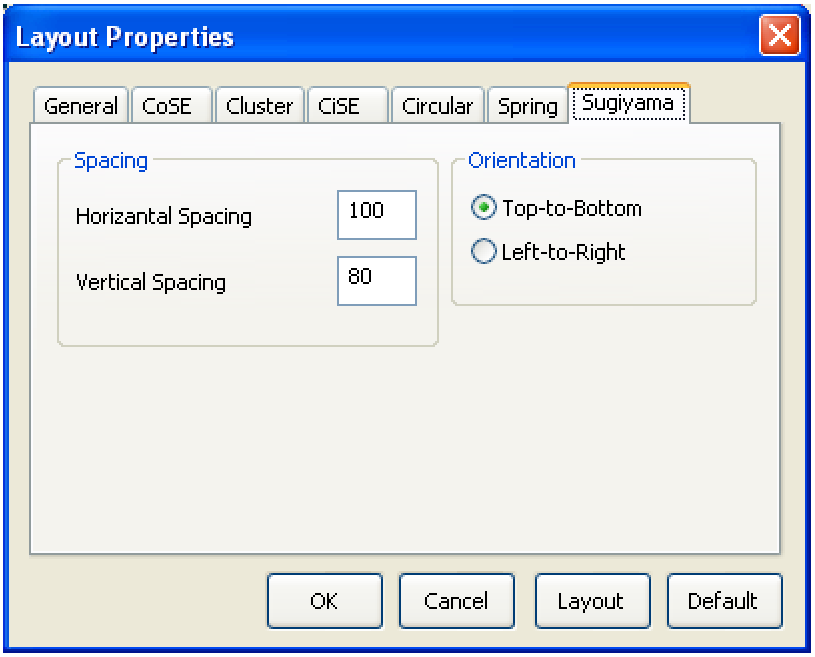
The hierarchical layout algorithm divides the nodes of a graph into levels and directs edges from top to bottom in a drawing oriented top-to-bottom as follows:

- For compound node support, inter-graph edges are converted into intra-edges;
- All roots in the graph are identified;

- By using these roots, nodes are topologically sorted with a DFS. If the graph contains cycles, the edges causing the cycles are reversed;
- Calculated topological information is used to create levels;
- If bend points are allowed to be created, bend points are created for multi level edges;
- Number of edge crossings between neighboring levels are reduced via the barycenter method;
- Levels of nodes are adjusted to minimize edge lengths;
- Spacing between levels and nodes are calculated;
- These values are converted into specific positions.

Layout Options

Hierarchical layout options are shown in Figure 62.



Spacing

Horizantal Spacing 100

Vertical Spacing 80

Orientation

● Top-to-Bottom

○ Left-to-Right

Figure 62 Hierarhical layout options

Horizantal Spacing: The space between two nodes in the same level

Vertical Spacing: The space between two subsequent levels

Orientation: Whether the drawing should from left-to-right (roots at the left, leaves at the right of the drawing) or top-to-bottom.

Chapter 4

Third-Party Software License Agreements

In this chapter, we list any third-party software license agreements.

Chisio Editor makes use of Eclipse Graph Editing Framework (GEF) version 3.1 distributed under license Eclipse Public License – v 1.0, for drawing and editing pathways. Silk icons, which are distributed under Creative Commons Attribution 2.5 License, have been used to design icons for the toolbar. In addition, Chisio Layout 2.0 library, which is part of Chisio Editor 2.0 makes use of certain third-party libraries. Please refer to Chisio Layout 2.0 Programmer's Guide [8] for licensing terms of those libraries.

Component	Version	By	License
Eclipse Graph Editing Framework (GEF)	3.1	Eclipse	Eclipse Public License - v 1.0
Silk Icons	-	Mark James	Creative Commons Attribution License, v 2.5
XMLBeans	-	The Apache Software Foundation	Apache License

References

[1] GraphML, http://graphml.graphdrawing.org

[2] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir, "A Compound Graph Layout Algorithm for Biological Pathways", In Graph Drawing (Proc. GD '04), LNCS, vol. 3383, pp. 442-447, 2005.

[3] H. He and O. Sýkora, "New circular drawing algorithms", In Proc. Workshop on Information Technologies - Applications and Theory (ITAT'04), 2004.

[4] GINY, http://csbi.sourceforge.net

[5] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs", Information Processing Letters, 31, pp. 7-15, 1989.

[6] jgraph, http://www.jgraph.com

[7] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for Visual Understandings of Hierarchical System Structures", IEEE Transactions in Systems, Man, and Cybernetics, vol. smc-11, no.2, pp. 109-125, February 1981.

[8] Chisio Layout, https://sourceforge.net/projects/chilay/

[9] T.M.J. Fruchterman, E.M. Reingold, "Graph drawing by force-directed placement", Software Practice and Experience 21(11), pp. 1129–1164, 1991.

[10] C. Walshaw, "A Multilevel Algorithm for Force-directed Graph Drawing", (Proc. Graph Drawing) LNCS, 1984, pp. 171-182, 2001.

[11] U. Dogrusoz, M.E. Belviranli, and A. Dilek, "CiSE: A Circular Spring Embedder Layout Algorithm", under review.