README.md

Resilionator

Resilionator is a small and portable Python application for identifying and mitigating potential weak points in networks. The application is built with Python's standard GUI toolkit Tkinter and NetworkX. The tool is intended primarily for university lecturers or students, but also for small and medium-sized companies and households.

Developer Documentation

For enhancing Resilionator we recommend a solid knowledge in Python as well as graph theory. Further we recommend some experience with Tkinter and NetworkX. Resilionator is split into frontend and backend. All calculations concerning networks are processed in the backend. The results are then forwarded to the frontend where they are subsequently displayed.

Now we introduce some functions in Resilionator, which can be used to out of the box:

Frames

- `createMainFrame` : Creates a frame that is appended to the root window.
- `createRightFrame` and `createLeftFrame` : Creates a paned window. If both frames are created, both will take up an equal amount of space. If only one frame is created it takes up the whole space.
- `createDualFrameView` : A more convenient way of creating the frames specified above.
- `createMenu` : Creates drop down menues.
- `createPopupWindow` : Creates a toplevel style window which is appended to the root window.
- `createCanvas` : Creates a canvas which is necessary form displaying figures which are generated from Matplotlib.
- `createScrollingFrame` : Creates a scrollable frame, which is appended to the main frame.

Garbage collection for frames

Frames in Tkinter need to be deleted manually, if not done correctly it may cause performance issues.

- `destroyFrame` : Destroys a specific frame.
- `destroyAllFrames` : This method is useful if you want to completely change views. For this method to work, frames need to be added to the `frames` list. However if you use any of the methods described above this will happend automatically.

Utility

- `importGraph` : Imports a graph from a file. Files formats that are currently supported are `.txt, .graphml and .gml` .
- `exportGraph` : Exports a graph to a file. Files formats that are currently supported are `.txt, .graphml and .gml` .
- `saveGraphAsImage` : Saves the graph as an image, currently supported formats are `.jpg and .png` .

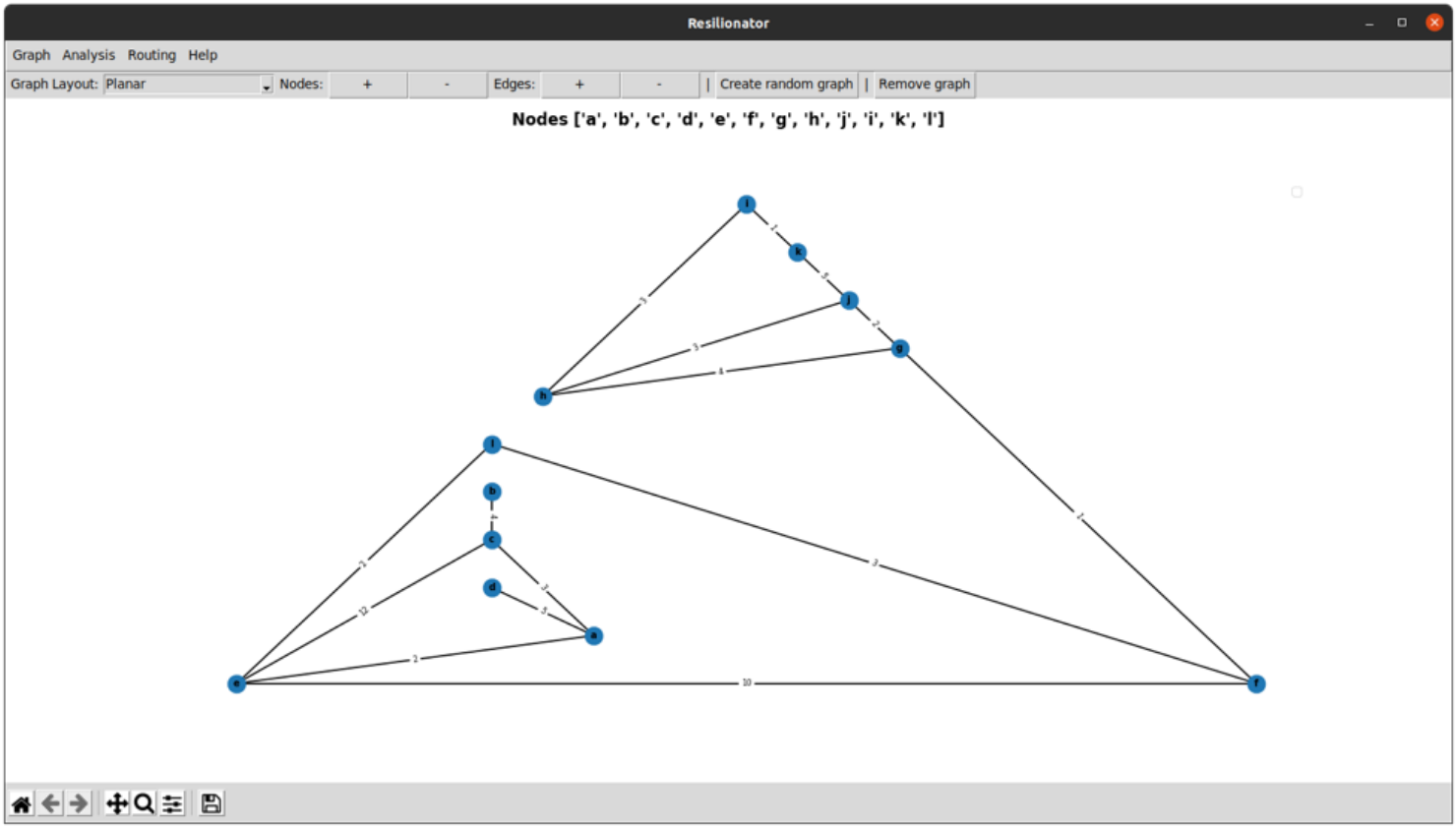
Graph

An empty graph is automatically created in the `init()` function of the Resilionator class.

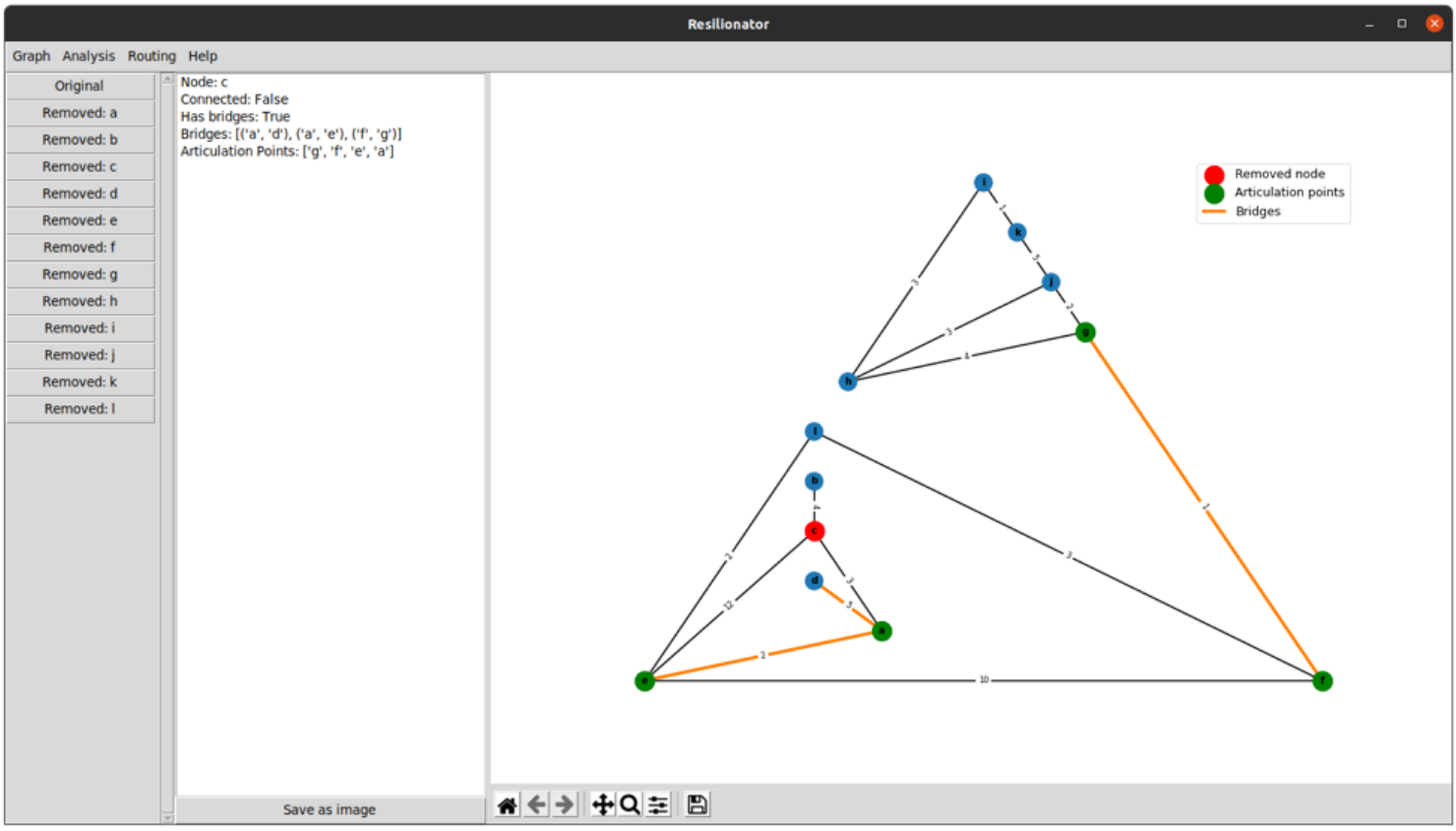
- `addNode` : Adds a new node to the graph.
- `addEdge` : Adds a new edge to the graph.
- `removeNode` : Removes a node from the graph.
- `removeEdge` : Removes an edge from the graph.
- `deleteGraph` : Removes the current graph and creates a new empty graph.
- `deleteGraph` : Checks if the current graph is empty - has no edges.

Screenshots

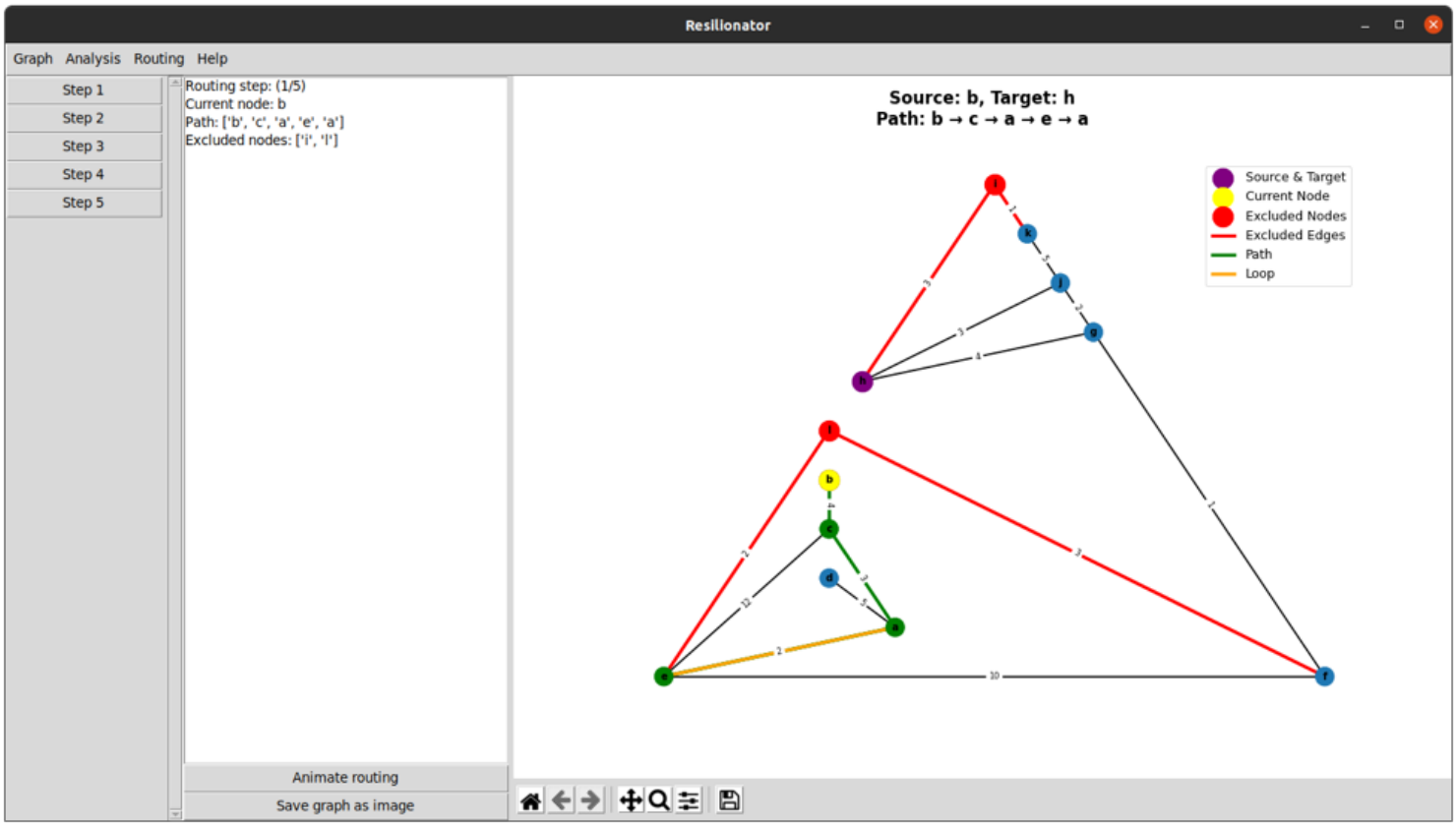
Homescreen



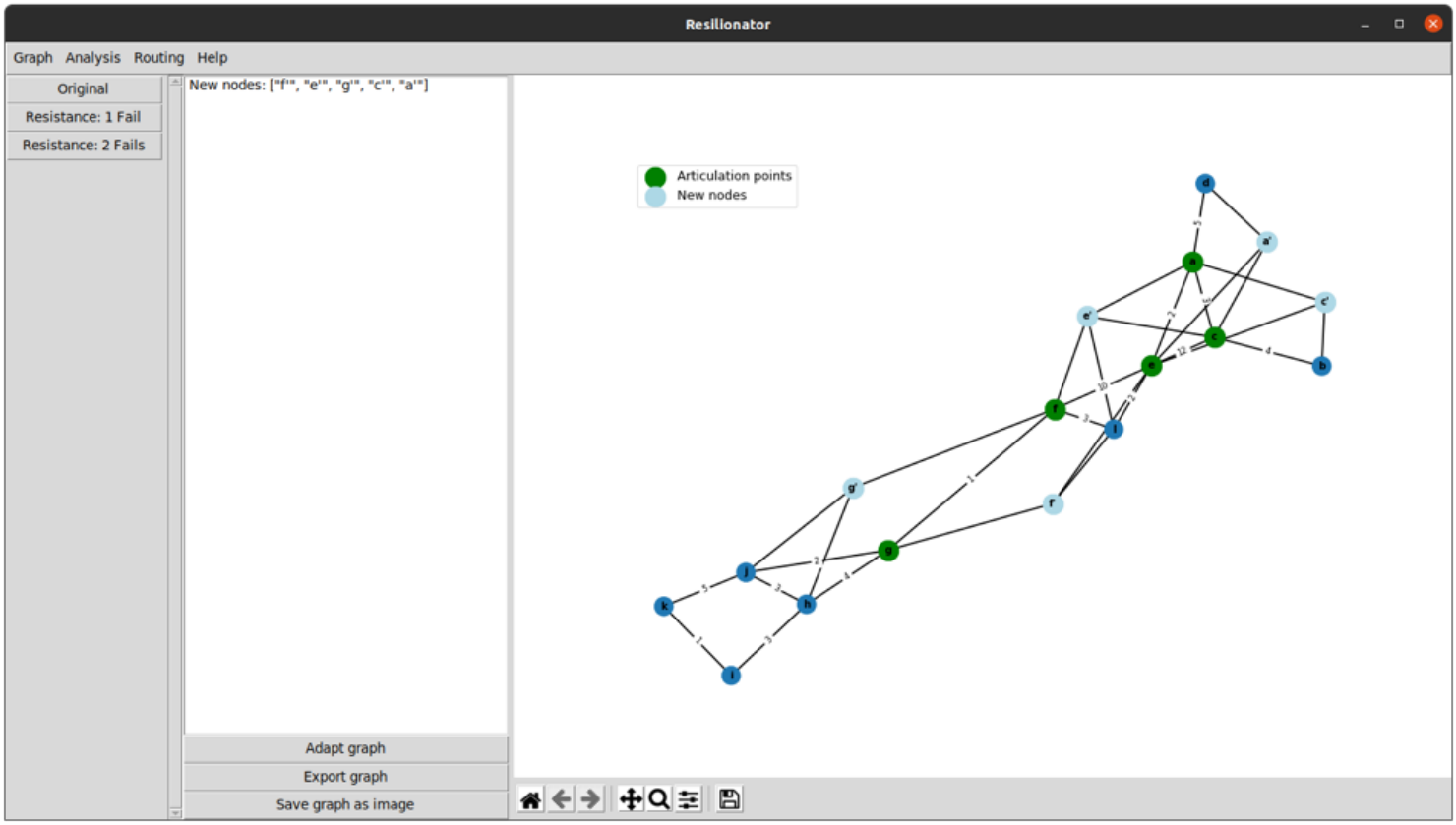
Node Connectivity



Dijkstra Routing



Failure Resistance



Resilionator Project Website

Link to the website: https://www.netidee.at/resilionator

The project Resilionator is funded by netidee.