

EA GUI: Graphical User Interface for OpenEA

Nguyen Chi Thanh, Manuel Schildknecht

July 24, 2017

1 Introduction

1.1 Context

EA GUI is a part of the internship project of Bui Quang Minh, Nguyen Chi Thanh, Pham Nguyen Quang, Vo Huynh Quang Kiet and Manuel Schildknecht at the Frankfurt University of Applied Science, Germany. The project is about building a generic system for network optimization using evolutionary algorithm (EA). EA GUI provides an expandable graphical user interface as an add-on to the OpenEA Core Library to run customized EA using configuration file as input. This report describes the web version of the EA GUI.

1.2 Requirements

Basic information and requirements:

- Language: HTML5/CSS3/JavaScript (1.8.5)
- Platform: Ubuntu 64 bit (16.4), Firefox (54.0)
- Dependencies:
 - npm(3.5.2)
 - body-parser(1.17.1)
 - cookie-parser(1.4.3)
 - debug(2.6.3)
 - express(4.15.2)
 - fs-extra(3.0.1)
 - meminfo(0.0.0)
 - morgan(1.8.1)
 - randomstring(1.1.5)
 - serve-favicon(2.4.2)
 - vis(4.20.0)
 - NodeJS(6.11.0)
 - bower(1.8.0)
 - angular-route(1.6.4)
 - material-design-icons(3.0.1)
 - angular-material(1.1.4)
 - angular-sanitize(1.6.4)
 - angular-scroll-glue(2.2.0)
 - angular-messages(1.6.4)
 - angular-material-data-table(0.10.10)
 - underscore(1.8.3)
 - JQuery(3.2.1)
 - amCharts3(3.21.2)
 - AngularJS(1.6.4)
- Recommended IDE: Webstorm 2017.1.4

1.3 Structure

The EA-GUI folder will look like following if it was cloned from the repository:

📁 EA-GUI

- `app.js`: JS imports required modules, reads request and returns response

- `install.sh`: installation script
- `package.json`: JSON defines npm dependencies and start path
- 📁 `EA-GUI` ▶ `bin` ▶ `www`: the start up script of npm
- 📁 `EA-GUI` ▶ `etc`
- 📁 `EA-GUI` ▶ `node_modules`: where npm libraries will be installed into
- 📁 `EA-GUI` ▶ `proc`: stores information about running and previous process
- 📁 `EA-GUI` ▶ `public`: stores all necessary components for the client-side of the web-app
 - `bower.json`: defines bower dependencies
 - `index.html`: index page of the web-app
- 📁 `EA-GUI` ▶ `public` ▶ `app`: consists of HTML and JS files for the web-app
- 📁 `EA-GUI` ▶ `public` ▶ `bower_components`: where bower libraries will be installed into
- 📁 `EA-GUI` ▶ `public` ▶ `css`: stores the `css` file for the web-app
- 📁 `EA-GUI` ▶ `public` ▶ `etc`
- 📁 `EA-GUI` ▶ `public` ▶ `extra_js`: stores extra JS files that are not installable via bower
- 📁 `EA-GUI` ▶ `routes`
 - `exec.js`
 - `index.js`
 - `proc.js`

2 Installation

2.1 Automatic Installation

If the whole project was cloned from the online repository, the installation script should be located in `(project_root_folder)/EA-GUI/install.sh`. It must be executed in order to install all dependencies. The user can run the script by the following command lines:

```
$ cd (project_root_folder)/EA-GUI/
$ chmod +x install.sh
$ ./install.sh
```

2.2 Manual Installation

The user can manually run the following command lines:

```
$ sudo apt install python-software-properties
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
$ sudo apt install nodejs
$ sudo npm install -g bower
$ sudo runuser -l "$SUDO_USER" -c "npm --prefix\"$execpath/EA-GUI\" install
```

```
\"$execpath/EA-GUI\""
$ sudo runuser -l "$SUDO_USER" -c "npm --prefix \"$execpath/EA-GUI/public\"
$ install \"$execpath/EA-GUI/public\""
$ cd ./public
$ bower install
```

3 Usages

This section is written mainly for the user of the graphical user interface, which is shown in Figure 1. The appearance of the graphical user interface may vary depending on your screen resolution. The recommended screen resolution is 1280x720 or higher.

The GUI can be started using the following command:

```
$ openea gui
```

Located on the left side of the web-app is the navigation bar, where the user can navigate him- or herself to the launch page, the graph viewer page or the currently running processes.

3.1 Homepage or Launch page

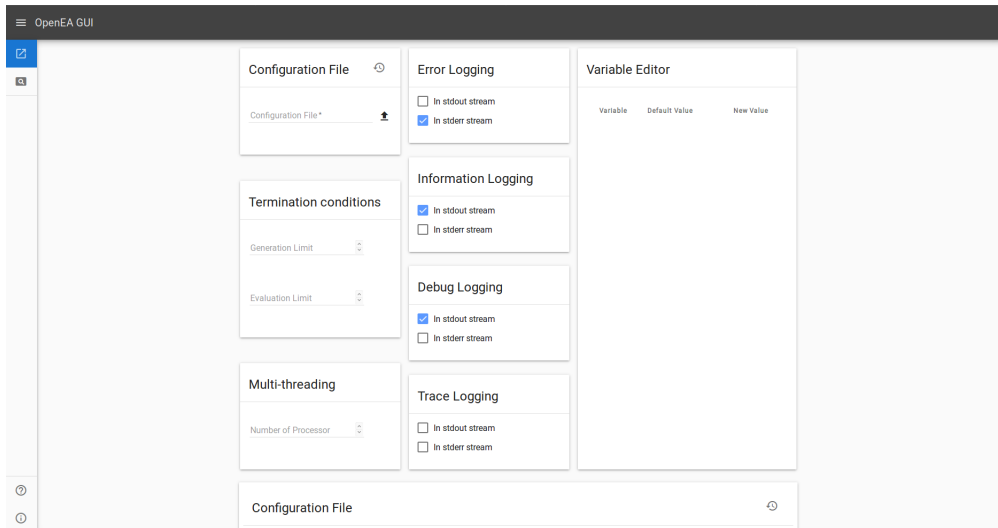


Figure 1: Graphical User Interface

The homepage (or launch page) is an configuration panel where the user can define the configuration file and relevant options and launch a new EA. The options are categorized so that the relevant options are located in the same **card**. Every card has two parts, the upper part is the *card title*, where the name of the card and relevant function buttons (if available) are located, and the other part is where the options in that category are located.

At the bottom of the page is the launch button to launch a new EA with the chosen configuration. By default, this button is disabled until all fields are valid.

3.1.1 File Chooser card

This card acts as a file uploader that helps the user to choose the desired configuration file. In order to choose a file from the local machine, the user can simply click in the area of the *Configuration File** field, then the internal file chooser will pop up and ask the user to choose the desired file. To clear the field, the user can press the reset button on the top-right corner of the card.

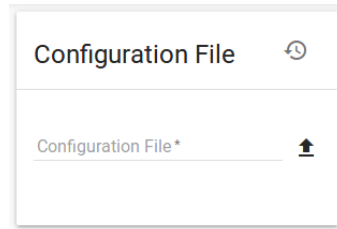


Figure 2: File Chooser card

3.1.2 Termination Condition card

This card consists of the fields for defining the limitation for generation or evaluation. The user can only input a positive integer into these two fields, otherwise the fields will be marked red indicating that the input is invalid and the user cannot start the EA with such input.

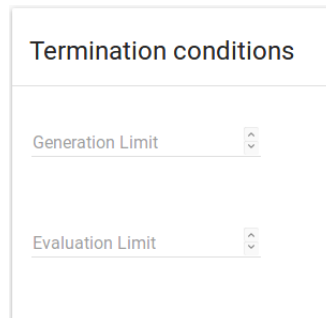
The image shows a card titled "Termination conditions". It contains two text input fields. The first field is labeled "Generation Limit" and the second is labeled "Evaluation Limit". Each field has a small icon to its right, consisting of a circle with a downward arrow and a circle with an upward arrow.

Figure 3: Termination Condition card

3.1.3 Multi-threading card

This field helps the user to define the number of threads for the EA. This field is blank by default.

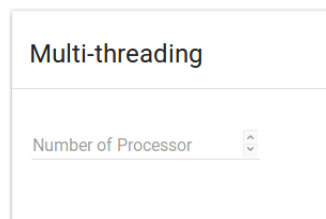
The image shows a card titled "Multi-threading". It contains a text input field labeled "Number of Processor" with a small icon to its right, consisting of a circle with a downward arrow and a circle with an upward arrow.

Figure 4: Multi-threading card

3.1.4 Logging cards

These cards provides the user with the options to show or hide specific logging levels, including *error*, *information*, *debug* and *trace*. There are two available logging streams for the user to choose, namely *standard output stream* (**stdout**) and *standard error stream* (**stderr**). The user can choose to show one logging level in two streams at the same time by checking both boxes, and the user can also choose not to show one logging level by un-checking both boxes. By default, the *Information* and *Debug* levels are shown in **stdout**, the *Error* level is shown in **stderr**, and the *Trace* level is hidden.

Error Logging	Debug Logging
<input type="checkbox"/> In stdout stream <input checked="" type="checkbox"/> In stderr stream	<input checked="" type="checkbox"/> In stdout stream <input type="checkbox"/> In stderr stream
Information Logging	Trace Logging
<input checked="" type="checkbox"/> In stdout stream <input type="checkbox"/> In stderr stream	<input type="checkbox"/> In stdout stream <input type="checkbox"/> In stderr stream

Figure 5: Logging cards

3.1.5 Variable Editor card

This card provides the user with the ability to edit the values of the available variables in the **eaml** file. The table will be filled after the user successfully chose a valid **eaml** file. There are three columns namely *Variable*, *Default Value* and *New Value*. The first two columns are used as references to set the new values, therefore they are read-only by default. There is no feature to check if the input value is valid or not, therefore the user must be careful when using this table.

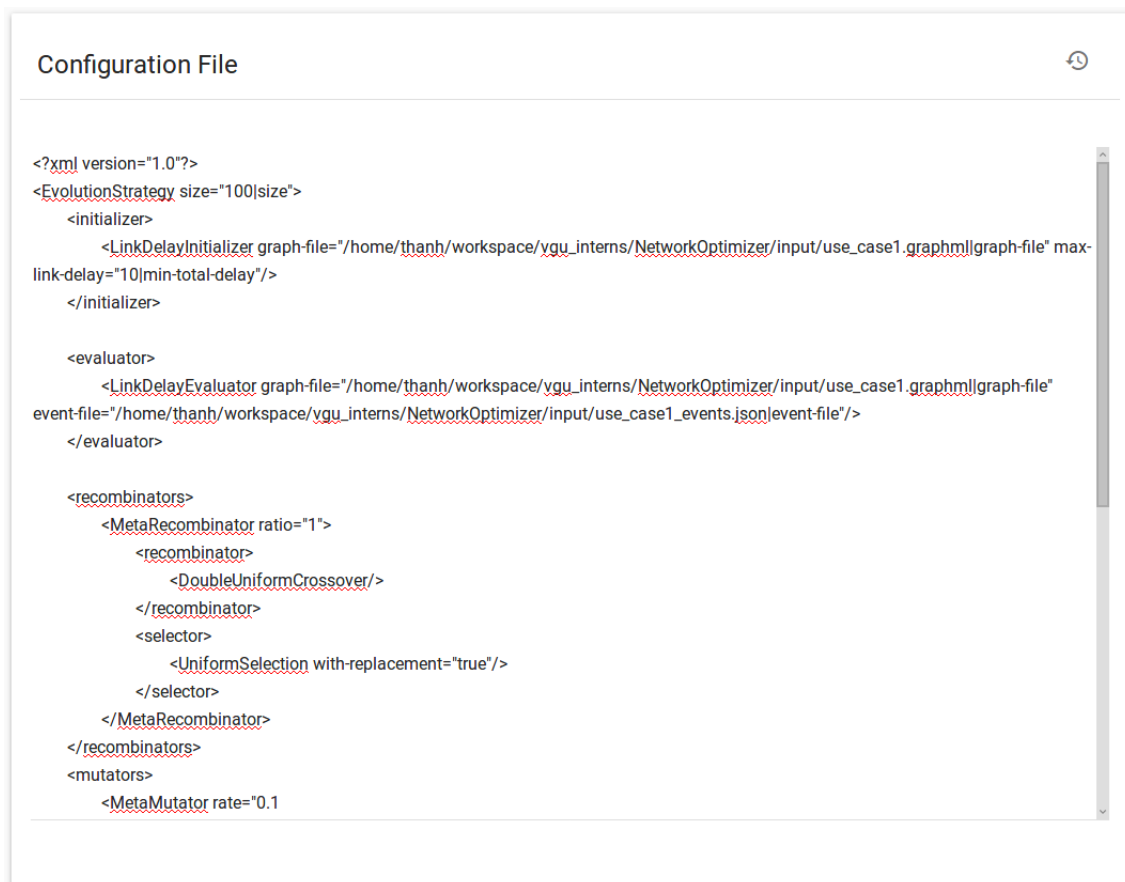
Variable Editor		
Variable	Default Value	New Value

Figure 6: Variable Editor card

3.1.6 EAML Editor card

This card provides the user with the ability to edit the whole **EAML** file without opening the local file viewer. The text field will be filled with the content of the **EAML** file once the user successfully chose a valid one. Changes in file content will only be available in the scope of the web-app and do not (or cannot) affect the original file in the local machine. The user can get the edited **EAML** file belonging to a previous run EA by browsing the **proc** folder.

The user can reset this field to the original content by pressing the top-right *reset* button.



The screenshot shows a web-based editor titled "Configuration File" with a reset button in the top right. The editor contains an XML configuration for an evolutionary strategy. The XML code is as follows:

```
<?xml version="1.0"?>
<EvolutionStrategy size="100|size">
  <initializer>
    <LinkDelayInitializer graph-file="/home/thanh/workspace/vgu_interns/NetworkOptimizer/input/use_case1_graphml|graph-file" max-link-delay="10|min-total-delay"/>
  </initializer>

  <evaluator>
    <LinkDelayEvaluator graph-file="/home/thanh/workspace/vgu_interns/NetworkOptimizer/input/use_case1_graphml|graph-file" event-file="/home/thanh/workspace/vgu_interns/NetworkOptimizer/input/use_case1_events.json|event-file"/>
  </evaluator>

  <recombinators>
    <MetaRecombinator ratio="1">
      <recombinator>
        <DoubleUniformCrossover/>
      </recombinator>
      <selector>
        <UniformSelection with-replacement="true"/>
      </selector>
    </MetaRecombinator>
  </recombinators>
  <mutators>
    <MetaMutator rate="0.1">

```

Figure 7: EAML Editor card

EAML Modification Priority

There are three ways that the user can modify a **EAML**. The first way is using a local script editor on the current machine, the second way is modifying the EAML editor field in the web-app and the last way is modifying the variable table. Out of these three ways, the variable editor has the most priority when it comes to modifying the EAML files, and the first way - modifying the **EAML** file itself - has the least priority. In the other words, the modifications in the variable editor will be considered first, then the eaml editor, then lastly the eaml file itself. Please bear in mind that after successfully choosing an eaml file and the eaml field is already ready, any local change in the eaml file does not affect this field, unless the user loads this file again.

3.2 Graph Viewer page

This page contains one card, called *Graph Visualizer*. This card displays the graphML-files with help of nodes (dots) and edges (lines), to create a network graph.

In the top-left corner of the card is a button which will open the internal file chooser and ask the user to choose the desired graphML-file. After the visualized graphML-file shows up at the middle of the card, a new button gets visible as well. This new button opens a drop-down list, which represents the node attributes of the graphML-file. With the help of these attributes and the button "cluster" at the top-right corner of the card, it is possible

to cluster the network. For this purpose the cluster-button takes the selected attribute from the drop-down list and bind the nodes which correspond to this attribute to one node. One click at a clustered node will release all the bounded nodes to their original places at the card. The cluster-button will not work if there is no attribute selected.

It is possible to move the nodes in the visualization and hovering over a node or edge will display a tooltip in which every attribute of the hovered node or edge is shown.

After a graphML-file is selected and shown at the screen a new file-chooser button will show up next to the first one. It asks the user to select a event.json-file. When a event-file is chosen the event will start and be shown with help of the visualized graph. Next to this file-chooser a button exist which toggles the Event so that it can be paused and started again.

At the very bottom of this card the events will be displayed as text. A scrollbar exists for the events so that the card stays at a normal size.



Figure 8: Graph Viewer

3.3 Process page

This page will be created once all configurations have been set and a new EA is started. The page consists of a console on the top, a fitness plot card on the bottom left and a graph visualization card on the right. If the EA is running, the console and the fitness plot is updated in real-time. If the EA has finished, the console will store the old information, and the fitness plot will stop updating itself. Once the EA has finished and the user tries to navigate between the sites, the fitness plot will not work when he or she get back to that same page again.

The graph visualization is exactly the same to the one seen in the graph viewer page, please see Section 3.2 for the description.

3.3.1 Console card

This card imitates the linux terminal and shows the information on the current EA. If the EA is running, the console is updated real-time. There are two buttons and two switches on the card header, including the show/hide button for hiding and showing the console, the stop button for stopping the EA and the switches for toggling show/hide the `stdout` and `stderr` streams.

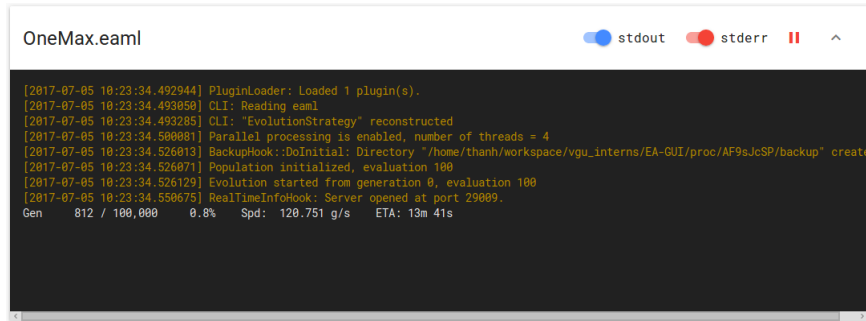


Figure 9: Console card

3.3.2 Fitness Plot card

This card displays the fitness values of the EA in real-time to a statistical diagram. In the top-right corner of the card are three buttons. Two of them the "Select" and the "Pan" button are select-buttons. If "Select" is pressed a preciser area of the statistical diagram can be selected. To move freely in the diagram "Pan" should be chosen. The third button is an update-button, which will stop newer values to enter the diagram. A second click on this button will rescind the first one and the diagram will draw new values again. At the bottom-left corner of this card, the values of all lines in the statistic are displayed as the sum of all values that are currently displayed at the diagram. As a result the sum of the values will decrease if a preciser area is selected. If the mouse is hovered over an area at the diagram, a tooltip will be displayed which shows the values of this area. At last there exist a button at the top-right corner of the diagram, which will only be visible if not the whole statistic is selected. Then the button "show all" can be pressed and the complete statistic will be shown.

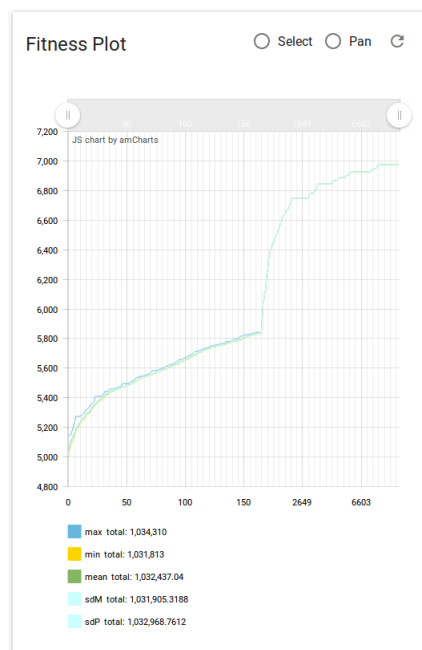


Figure 10: Fitness Plot card