

ALEVIN Software User Guide

Authors:

Michael Duelli and Daniel Schlosser (University of Wuerzburg)

Juan Felipe Botero and Xavier Hesselbach (Universitat de Catalunya)

Andreas Fischer (University of Passau)

Table of Contents

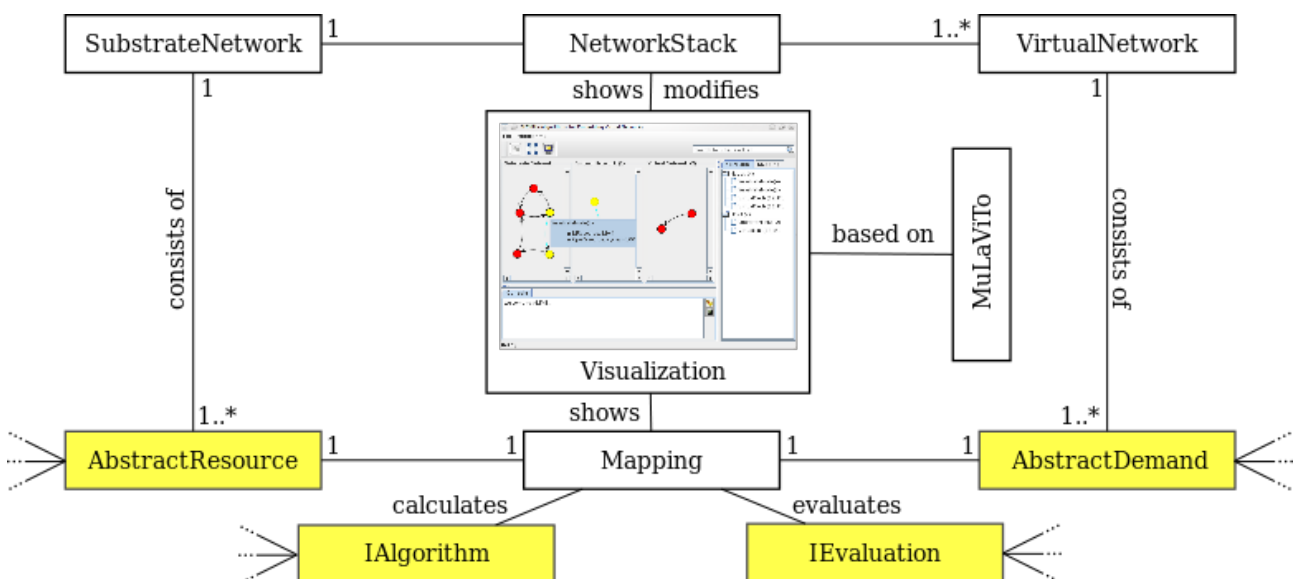
| | | |
|-------|--|----|
| 1 | Introduction..... | 2 |
| 2 | Installation Requirements..... | 3 |
| 2.1 | GLPK LP solver framework installation..... | 3 |
| 2.1.1 | Unpacking the distribution file..... | 4 |
| 2.1.2 | Configuring the package..... | 4 |
| 2.1.3 | Compiling the package..... | 4 |
| 2.1.4 | Check the package..... | 4 |
| 2.1.5 | Install the package..... | 4 |
| 3 | Editing..... | 6 |
| 3.1 | Create, Edit, Import, Export and Manage Scenarios..... | 6 |
| 3.1.1 | Scenario Creation From Scratch..... | 6 |
| 3.1.2 | Scenario Editing..... | 6 |
| 3.1.3 | Editing Nodes and Links..... | 8 |
| 3.1.4 | Scenario Import and Export..... | 9 |
| 3.1.5 | Scenario Generation..... | 10 |
| 4 | Currently Supported Algorithms..... | 15 |
| 4.1 | Algorithms brief description..... | 15 |
| 4.1.1 | Virtual Node Mapping Algorithms..... | 15 |
| 4.1.2 | Virtual Link Mapping Algorithms..... | 16 |
| 4.2 | Algorithms and Mappings..... | 17 |
| 4.2.1 | Mappings..... | 17 |
| 4.2.2 | Implemented Algorithms..... | 18 |
| 5 | GUI Features..... | 25 |
| 5.1 | The Graph Panel..... | 25 |
| 5.1.1 | Move Nodes..... | 25 |
| 5.1.2 | Zoom in and out..... | 25 |
| 5.2 | The Selection Panel..... | 25 |
| 5.3 | The Mapping Panel..... | 26 |
| 5.4 | The Console..... | 27 |
| 5.5 | Menu Options..... | 28 |
| 5.5.1 | File Menu..... | 28 |
| 5.5.2 | View Menu..... | 28 |
| 5.5.3 | Generators Menu..... | 29 |
| 5.5.4 | Algorithms Menu..... | 29 |
| 5.5.5 | Metrics Menu..... | 30 |

1 Introduction

The focus in the development of ALEVIN is on modularity and efficient handling of arbitrary parameters for resources and demands, and the support of integration of new and existing virtual network embedding algorithms and evaluation metrics.

For platform independency, ALEVIN is written in Java. ALEVIN's *graphical user interface* (GUI) and multi-layer visualization component is based on the MuLaViTo project MuLaViTo (<http://mulavito.sf.net>) which enables us to visualize and handle the substrate and arbitrary virtual networks as directed graphs.

The next figure depicts the architecture of ALEVIN and highlights the modular interaction of parameters for substrate as well as virtual networks, algorithms, and evaluation.



ALEVIN provides the ability to illustrate the deployment of resources in the substrate network and demands in an arbitrary number of virtual networks as well as the mapping of demands on resources calculated by a VNE algorithm. Moreover, ALEVIN can be used to create VNE scenarios as well as import and export them using an XML-based exchange format.

ALEVIN is completely modular regarding the addition of new parameters to the VNE model. Using the java *visitor design pattern* in a sophisticated way, we are able to avoid any casts to concrete demand/resource classes. Thus, the number of parameters is not performance-relevant and a convenient implementation of arbitrary parameters is possible.

To increase ALEVIN's modularity and to make it a flexible and extensible platform to compare existing and upcoming algorithms, the implementation of algorithms is kept independent of the resource/demand implementation. To that end, a simple interface is provided defining the rough structure of an algorithm and connecting its output to the GUI as illustrated in the previous picture.

2 Installation Requirements

Being implemented in Java, Alevin is platform independent and should work on MS Windows, Mac OS X and Linux. For proper operation of Alevin, however, the following software packages have to be installed:

- SUN Java JDK version 1.5 or later (recommended: version 6, update 21 or later). Do not use other Runtime Environments, as they are known to create problems!
- The GNU Linear Programming Kit (GLPK): <http://www.gnu.org/software/glpk/> - this is used by some algorithms.

2.1 GLPK LP solver framework installation

The GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library (<http://www.gnu.org/software/glpk/>).

Project GLPK for Java delivers a Java language binding (<http://sourceforge.net/projects/glpk-java/>).

Some algorithms of ALEVIN use linear programming to realize the VNE. ALEVIN implementation relies on GLPK to solve the VNE node and link mapping stages, when they require Linear Programming. As ALEVIN is developed in java, we have use the java biding interface provided in <http://sourceforge.net/projects/glpk-java/>.

The installation instructions are placed in the /doc folder of the glpk-java software. The file "glpk-java.pdf" explains how the installation must be done. However, here a step by step explanation will be provided for each platform:

1. Windows: The GLPK for Java JNI library can be compiled from source code. The build and make files are in directory w32 for 32 bit Windows and in w64 for 64 bit Windows. The name of the created library is glpk 4 45 java.dll for revision 4.45.

Recommended: A precompiled version of GLPK for Java is provided at <http://winglpk.sourceforge.net>

The library (the name of the library is glpk_4_45_java.dll) has to be in the search path for binaries, it can be found in the w64 or w32 folder (depending of your computer architecture) of the percompiled version (<http://winglpk.sourceforge.net>). Either copy the library to a directory that is already in the path (e.g. C:\windows\system32) or update the path in the system settings of Windows. The jar file, needed also for the execution of ALEVIN is stored also in the same w64 or w32 folder.

The library has to be in the CLASSPATH. Update the classpath in the system settings of Windows or specify the classpath upon invocation of the application, e.g. java -classpath ./glpk-java.jar;. MyApplication

1. Linux: To install the LINUX version, the original GLPK library (<http://www.gnu.org/software/glpk/>) must be compiled in first place. The GLPK for Java JNI library can be compiled from source code. The following instructions are also contained in the file INSTALL provided in the source

distribution (<http://sourceforge.net/projects/glpk-java/>):

2.1.1 Unpacking the distribution file

Copy the distribution file to a working directory. Check the MD5 checksum with the following command:

```
md5sum glpk-java-X.Y.tar.gz
```

Unpack the archive with the following command:

```
tar -xzf glpk-java-X.Y.tar.gz
```

Now change to the new directory glpk-java-X.Y

2.1.2 Configuring the package

Open swig/Makefile in a text editor, e.g. with the following command

```
vi swig/Makefile
```

Adjust the installation path "prefix".

Adjust the include path for glpk.h.

Adjust the version information concerning GLPK.

Save the text file.

On Mac OS X jni.h is in the following path:

```
/System/Library/Frameworks/JavaVM.framework/Headers
```

2.1.3 Compiling the package

To remove all files from prior compiling use the command

```
make clean
```

The package is compiled with the command

```
make
```

2.1.4 Check the package

To check if everything is built correctly use the command

```
make check
```

2.1.5 Install the package

To install the package you must be root or a sudoer. As sudoer use the command

```
sudo make install
```

1. Mac Os: The process to install GLPK-java in MacOS is the same as it is in linux. Remember that on Mac OS X jni.h is in the following path:

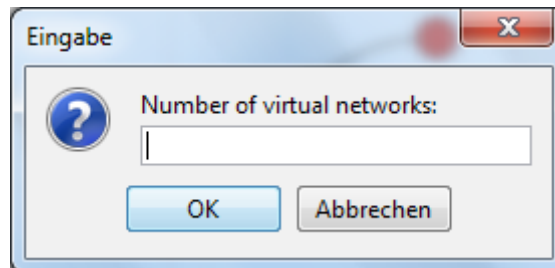
`/System/Library/Frameworks/JavaVM.framework/Headers`

3 Editing

3.1 Create, Edit, Import, Export and Manage Scenarios

3.1.1 Scenario Creation From Scratch

To create a new, empty scenario select “New empty scenario / layers” from the [File Menu](#) or use the “Ctrl + N” Hotkey. In the next step, you will be prompted for the number of virtual networks of the scenario to be created:

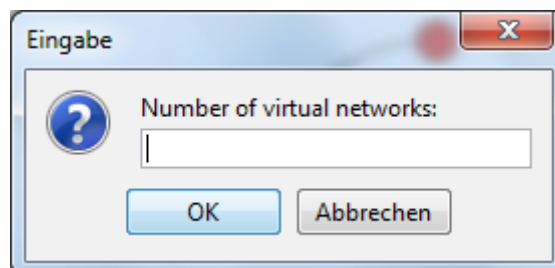


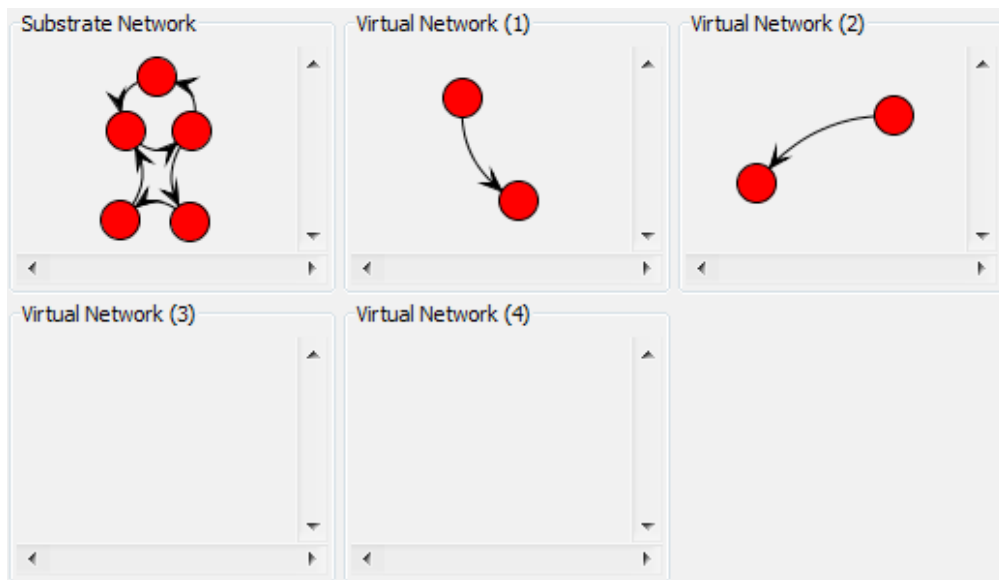
A scenario consisting of a substrate network and the desired number of virtual networks is created and shown in the graph panel. All created layers, substrate or virtual will be empty. To add nodes and links, follow the steps described in the Scenario Editing section below.

3.1.2 Scenario Editing

Adding Networks

To add virtual networks select “New empty scenario / networks” from the [File Menu](#). Next you will be prompted for the number of virtual networks to add:



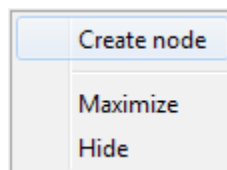


The desired number of virtual networks is added to the graph. If we add two more networks to our exemplary scenario, the graph will be as follows:

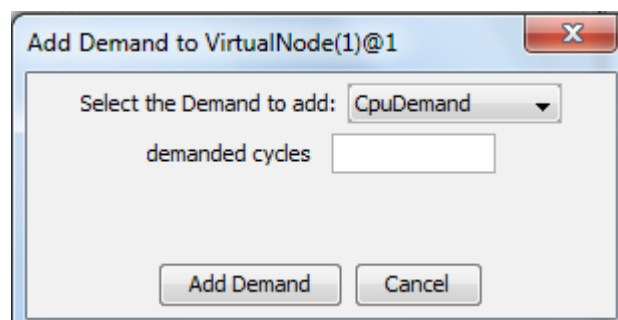
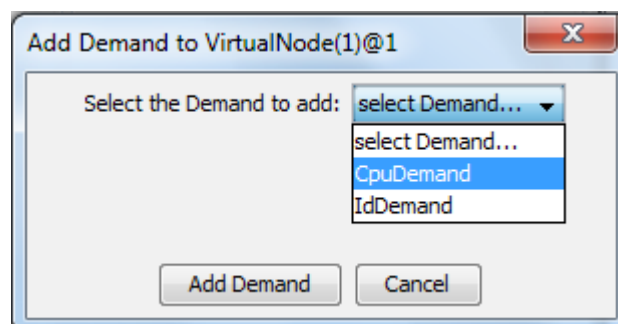
Note that the new networks are **empty**.

Creating Nodes

A node can be created in an existing network (substrate or virtual network), at a desired position. To create a node, right-click at the desired position and select "Create Node" from the pop-up menu:



Next you need to add a constraint to the node to be created:



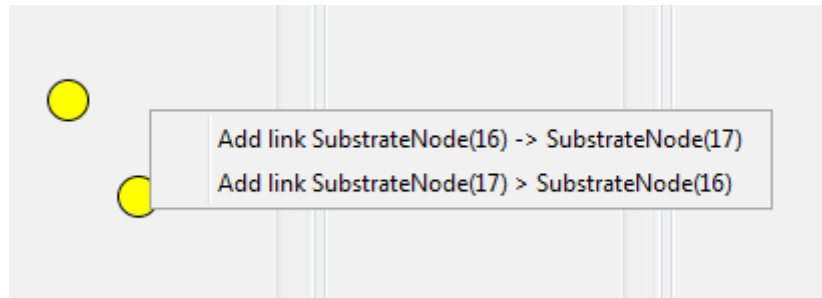
and set the parameter(s) for the selected constraint:

A constraint is **required** for the node to be created.

After following this steps, the node will be created at the desired position and shown in the graph panel.

Creating Links

To create a link select two nodes and right-click on the layer they belong to. You then have to choose source and destination of the link to be created. To make things simple, the two alternatives are displayed in the pop-up menu:



After the desired link is selected from the pop-up menu you need to add a constraint to the link, for it to be created. For this purpose, a dialog as described in the Creating Nodes section will be displayed. As in the case of nodes, a constraint is **required** for a link to be created. After adding a constraint, the link will be created and shown in the graph panel.

3.1.3 Editing Nodes and Links

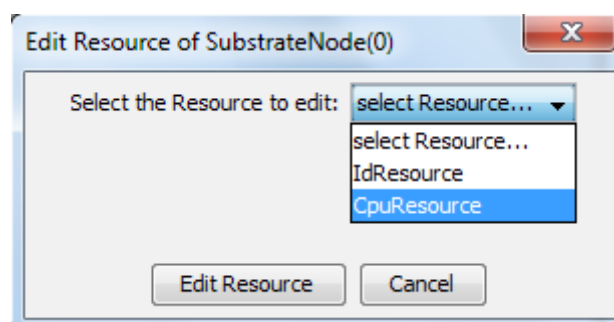
To edit a node or a link, right-click on it. You have the following options:

Add Constraint

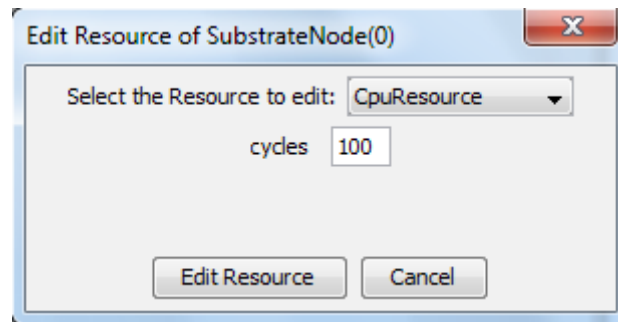
If you choose to add a constraint you will need to select the constraint type to add and set it's parameter(s), as described in the Creating Nodes section. Note that each node / link can only have a single resource of a given type.

Edit Constraint

If you select the edit constraint option, you can edit the parameter(s) of the node's / link's constraints. First you need to select the constraint to be edited:



then you can set the new values of the parameter(s):



Remove Constraint

To remove a constraint, select it as described in the Edit Constraint section above and click on the “Remove”-button.

Delete Node / Link

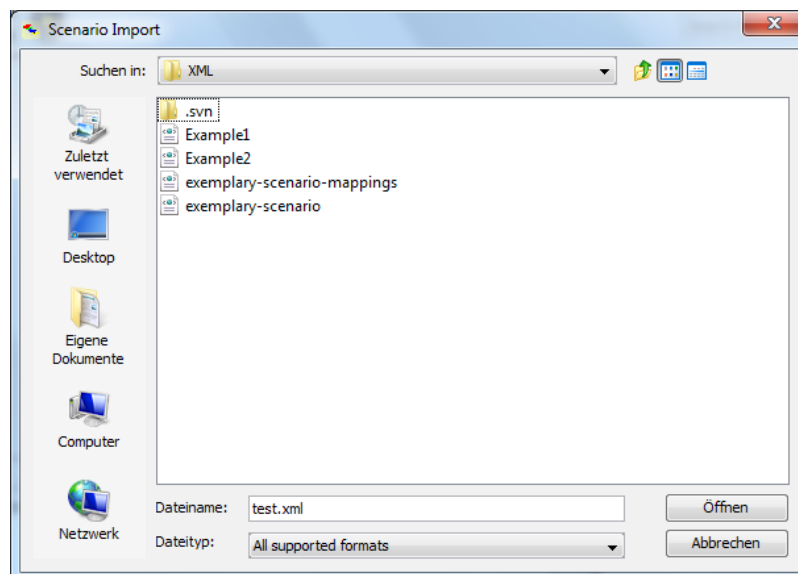
If you select this option, the network entity along with its constraints and mappings will be deleted.

If you delete a node, all its incident links will be deleted as well.

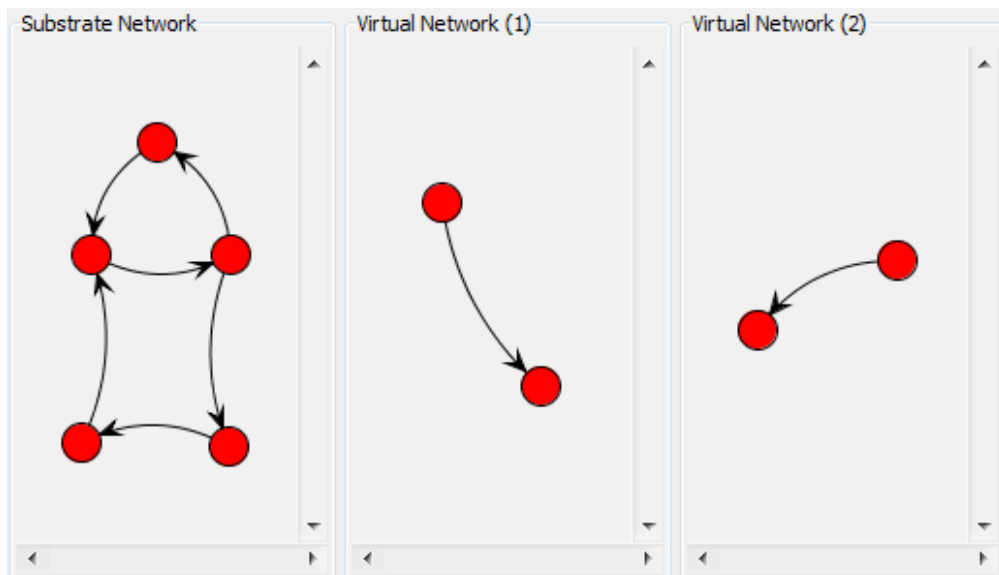
3.1.4 Scenario Import and Export

Scenario Import

To import a scenario select “Import” from the [File Menu](#) or use the “Ctrl + O” Hotkey. Next, select the file to import from:



Import file chooser



Exemplary scenario import

The imported scenario will then be displayed in the graph panel of the GUI. For our exemplary scenario this looks as follows:

Scenario Export

To export the current scenario, select “Export” from the [File Menu](#) or use the “Ctrl + S” hotkey. You will then need to choose the file to export to and select “Save”. The Scenario, including constraints and mappings will be saved as XML-data to the file specified.

3.1.5 Scenario Generation

Network Topology Generation

ALEVIN provides a random scenario generator, the scenario wizard. For the scenario generation the number of nodes in each network is set and the links are added randomly using a Waxman generator. It creates edges within a 1×1 square with probability $\frac{\alpha}{d_{ij}^\beta}$, where d_{ij} is the Euclidian distance between vertex i and vertex j , and α is the maximum distance between any two nodes. An increase in the parameter α increases the probability of edges between any nodes in the graph, while an increase in β yields a larger ratio of long edges to short edges.

For more information on the waxman generator please refer to

```
@article{Waxman88,
author = {Waxman, Bernard M.},
title = {Routing of Multipoint Connections},
journal = {IEEE Journal of Selected Areas in Communication},
pages = {1617--1622},
number = {9},
volume = {6},
month = dec,
year = {1988},
}
```

Prior to setting any parameters, the scenario wizard looks as follows.

The Scenario Wizard dialog box is shown with the following settings:

- Substrate Network:**
 - Substrate nodes: 1
 - Alpha: 1
 - Beta: 0.5
- Virtual Networks:**
 - number: 1
 - default number of virtual nodes: 1

| # | # nodes | alpha | beta |
|---|---------|-------|------|
| 1 | 1 | 1.0 | 0.5 |

Buttons: Create, Cancel

To generate a new scenario the following parameters must be set

- For the substrate network
 - The number of substrate nodes.
 - The value of the alpha and beta parameters.
- The number of virtual networks.
- The default number of virtual nodes per network.
- For each virtual network
 - The number of nodes, if different from the default.
 - The value of the alpha and beta parameters.

To set parameters for a specific virtual network, just click on the respective table row. If the mouse stands still over one of the table columns, a short info about the parameter is displayed, as shown in the graphic below.

After setting the parameters as needed, the aspect of the scenario wizard changes slightly.

Scenario Wizard

Substrate Network

Substrate nodes

Alpha

Beta

Virtual Networks

number default number of virtual nodes

| # | # nodes | alpha | beta |
|---|---------|-------|------|
| 1 | 5 | 1.0 | 0.5 |
| 2 | 4 | 0.9 | 0.6 |
| 3 | 5 | 1.0 | 0.5 |
| 4 | 6 | 1.0 | 0.5 |

alpha > 0
An increase in alpha will increase the number of edges in the graph.

Create Cancel

Constraints Generation

Generating constraints is possible using the constraints generator.

Generate Constraints

Resources

| Select | Resource | Maximum parameter values |
|--------------------------|-------------------------|-------------------------------------|
| <input type="checkbox"/> | BandwidthResource | max. bandwidth <input type="text"/> |
| <input type="checkbox"/> | CpuResource | max. cycles <input type="text"/> |
| <input type="checkbox"/> | EnergyResource | |
| <input type="checkbox"/> | IdResource | |
| <input type="checkbox"/> | MultiCoreEnergyResource | |
| <input type="checkbox"/> | StaticEnergyResource | |

Demands

VN 1 VN 2 VN 3

| Select | Demand | Maximum parameter values |
|--------------------------|-----------------------|-------------------------------------|
| <input type="checkbox"/> | BandwidthDemand | max. bandwidth <input type="text"/> |
| <input type="checkbox"/> | CpuDemand | max. cycles <input type="text"/> |
| <input type="checkbox"/> | EnergyDemand | |
| <input type="checkbox"/> | IdDemand | |
| <input type="checkbox"/> | MultiCoreEnergyDemand | |
| <input type="checkbox"/> | StaticEnergyDemand | |

Generate Cancel

Constraints Generator

It consists of two sections, one for the resources and one for the demands. The most important component of the constraints generator is the table used for selecting the constraints to add and setting the required parameters. The resources panel contains one such table, while the demands panel consists of one table for each virtual network, the tables being packed in different tabs.

For constraints generation, a constraint needs to be selected for the desired network (substrate or virtual) and the corresponding parameters set, if needed. This is done using the tables described above. Currently only CPU and bandwidth constraints have parameters to set. All other constraints need no further user action to be generated correctly. Demand generation can be done individually on a per virtual network basis.

After selecting some constraints and setting the needed parameters, the constraints generator will look as seen below. Notice that, as described above,

setting additional parameters is not required for all resource/demand types.

Generate Constraints

Resources

| Select | Resource | Maximum parameter values |
|-------------------------------------|-------------------------|--------------------------|
| <input checked="" type="checkbox"/> | BandwidthResource | max. bandwidth 20 |
| <input checked="" type="checkbox"/> | CpuResource | max. cycles 100 |
| <input type="checkbox"/> | EnergyResource | |
| <input checked="" type="checkbox"/> | IdResource | |
| <input type="checkbox"/> | MultiCoreEnergyResource | |
| <input checked="" type="checkbox"/> | StaticEnergyResource | |

Demands

VN 1 VN 2 VN 3

| Select | Demand | Maximum parameter values |
|-------------------------------------|-----------------------|--------------------------|
| <input checked="" type="checkbox"/> | BandwidthDemand | max. bandwidth 8 |
| <input checked="" type="checkbox"/> | CpuDemand | max. cycles 33.3 |
| <input type="checkbox"/> | EnergyDemand | |
| <input type="checkbox"/> | IdDemand | |
| <input checked="" type="checkbox"/> | MultiCoreEnergyDemand | |
| <input type="checkbox"/> | StaticEnergyDemand | |

Generate **Cancel**

Constraints Generator with selected constraints and parameter set

Removing All Constraints

It is also possible to remove all constraints of the current scenario, for example in order to regenerate them.

4 Currently Supported Algorithms

4.1 Algorithms brief description

In this section, a brief description of the virtual node and link mapping existing algorithms is given. The explanation will separate the virtual node mapping and link mapping.

4.1.1 Virtual Node Mapping Algorithms

Greedy Stress Approach (GS)

To understand the objective of this approach. Figure 2 defines the term stress.

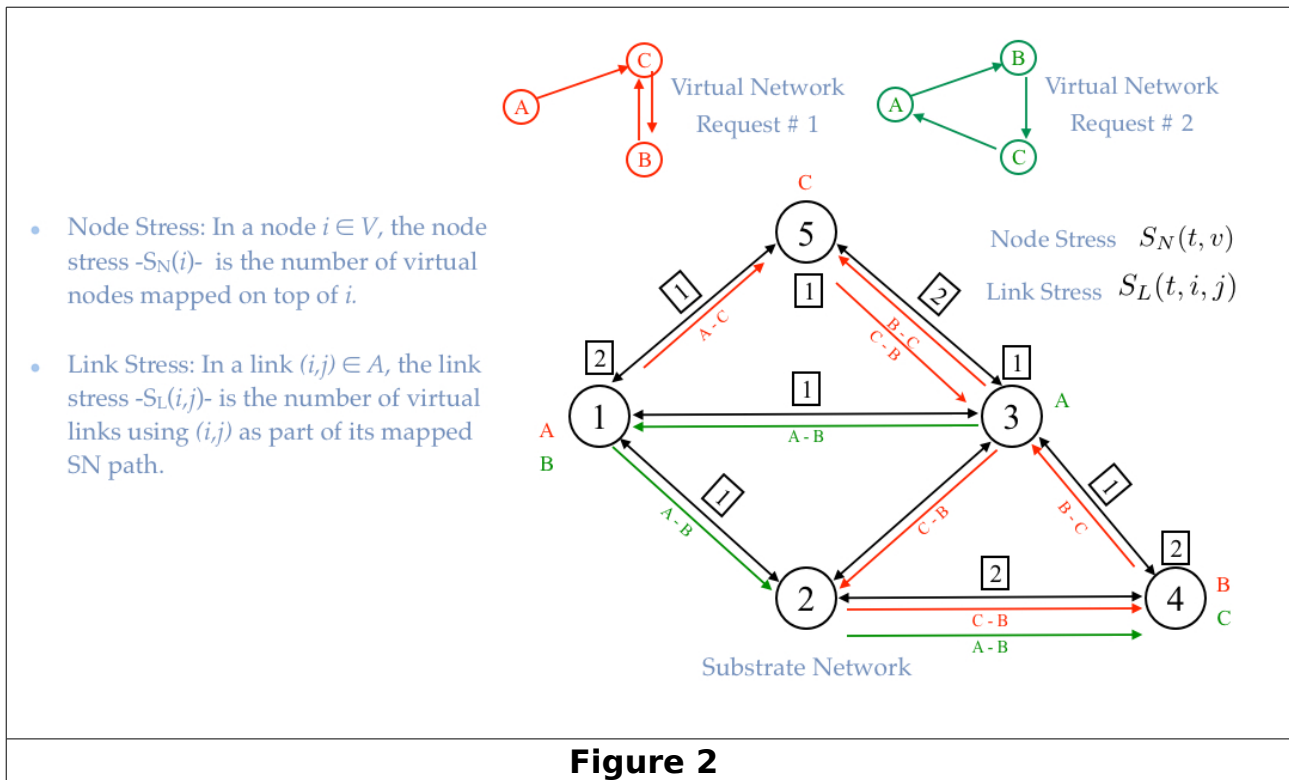


Figure 2

The numbers above the links and nodes are the respective stresses. The objective of this approach is to minimize the balanced stress (weighted sum of node and link stresses) in the network. To reach this objective, the virtual node mapping stage is performed by assigning the nodes with greater degree in virtual network to the nodes with greater *potential* (for each node in the SN, the potential is the multiplication of the node stress by the sum of adjacent link stresses) in the substrate network.

Greedy Available Resources (GAS)

This virtual node mapping approach is very similar to the GS. The available resources concept is defined in Figure 3.

* CONCEPTS DEFINED IN EXISTING APPROACHES

- Available Resources (H): Is a substrate node parameter. It measures node capacity, taking into account the capacity of its incident links.

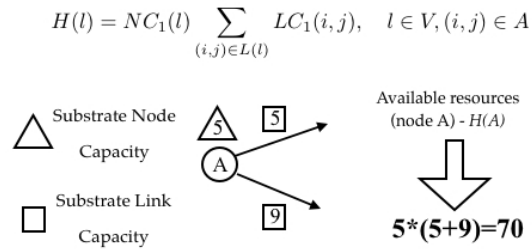


Figure 3

The virtual node mapping GAS is a greedy algorithm. Nodes with greater demand are assigned with greater available resources.

Mixed Integer Programming Solution

Substrate Network Graph is augmented creating Meta-nodes (representing the virtual nodes) and meta-edges (joining meta-nodes with the candidate nodes to be mapped in the SN), they are added to the SN graph as indicated in Figure 4.

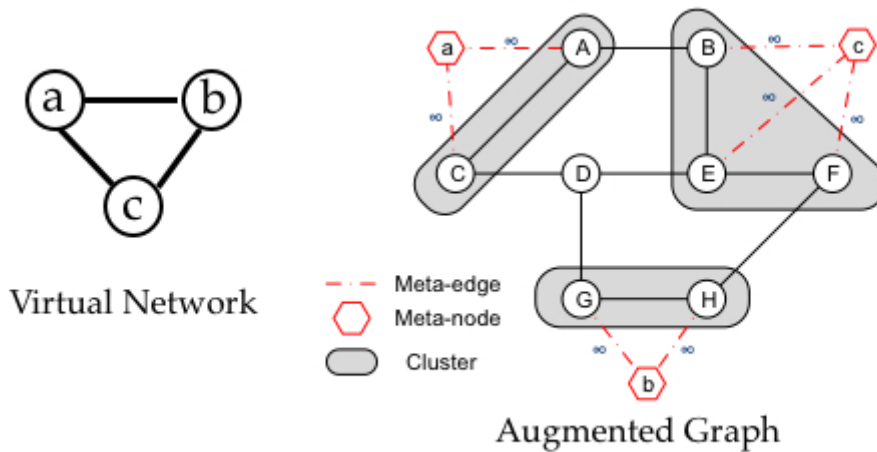


Figure 4

In figure 4, a will be mapped to either A or C. (A and C are the candidates nodes to map a). After the following steps, the map is decided.

- The problem is solved using linear programming (LP) -> The relaxed [BIP](#) formulation is solved.
- The LP solution contains rational value for each of the meta-edges (joining the meta-nodes with nodes in the substrate network). A randomized or deterministic rounding is performed, among the meta-edges of each meta-node, to choose one of them; the SN connected to the chosen meta-edge is then chosen as the mapped node in the SN.

4.1.2 Virtual Link Mapping Algorithms

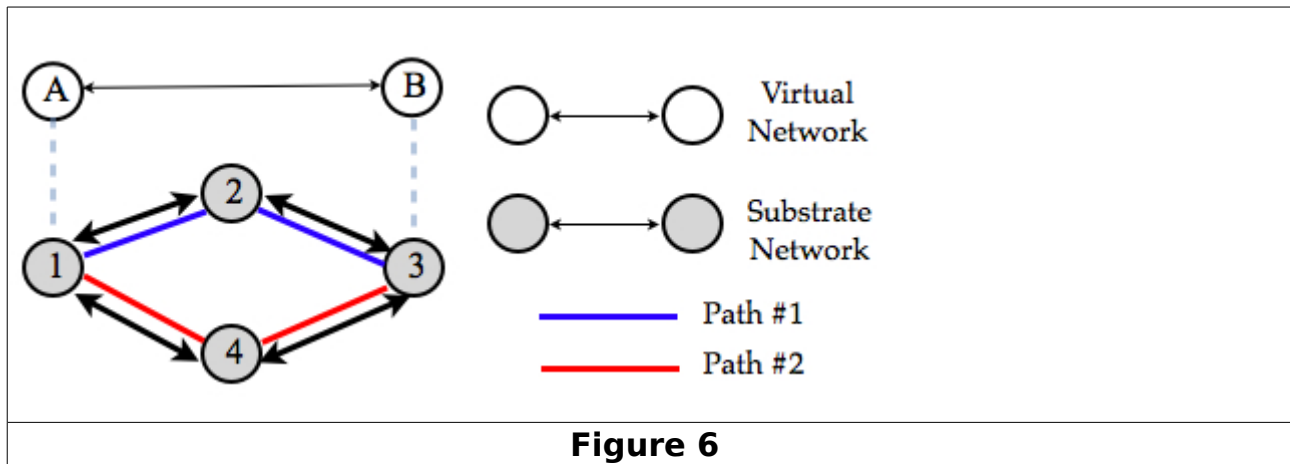
It is very important to state that the virtual link mapping stage starts after virtual node mapping.

K-Shortest Paths Algorithm (KSP)

This algorithm calculates, for each virtual link, the k-shortest paths from one mapped virtual node to the other end (in the substrate network). The mapping is performed by assigning, to each virtual link, the first shortest path that accomplishes the bandwidth and CPU demands.

Multi-Path Algorithm (MP)

This algorithm solves a LP formulation of the virtual link mapping stage (the same formulation as the relaxed [BIP](#)). This formulation is equivalent to the multi-commodity flow problem and is solved using the traditional LP methods (SIMPLEX, IPM, etc.) A example of this approach is shown in Figure 6.



Rounding Multi-Path (RMP)

The problem is formulated as the multi-commodity flow problem and is solved using optimal linear programming algorithms (SIMPLEX, IPM, etc.). Then, the multi-path solution, each path will have a rational percentage of the demand, is rounded (in a deterministic or randomized way) and just one directed path is used to map each virtual link.

Subgraph isomorphism detection heuristic (SID)

The mapping in nodes and links is done simultaneously by trying to find a subgraph inside the substrate network being isomorphic to the virtual network request. Graph isomorphism is explained in figure 5.

4.2 Algorithms and Mappings

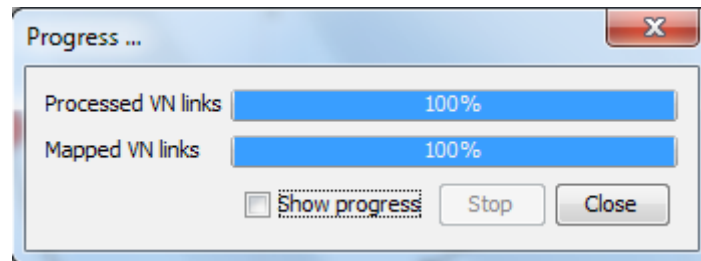
4.2.1 Mappings

Compute Mappings

To compute mappings for the current scenario you need to run an algorithm from the Algorithms Menu. To do so, just select it from the menu. Algorithms can not be run on scenarios that already have mappings.

To better understand the input parameters requested by each algorithm, see a brief and detailed explanation of the algorithms.

While an algorithm is running, a progress dialog is shown:



Algorithms progress dialog

4.2.2 Implemented Algorithms

Until now, there are five algorithms available:

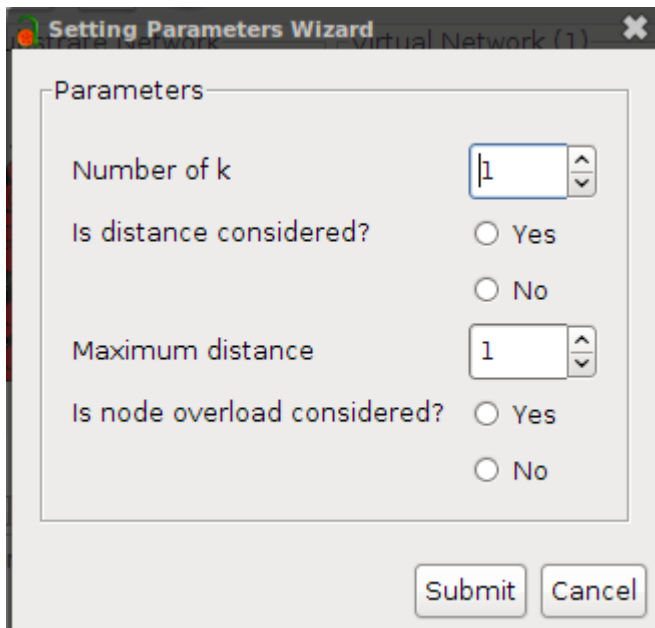
- Simple Dijkstra: Test virtual link mapping algorithm (with node mapping assumed as performed) mapping each virtual link to the shortest path.
- Greedy Available with K-Shortest Path: Virtual node and link mappings for each VNR are performed separately. Node mapping is performed using a greedy algorithm that maps virtual nodes with higher revenues to SN nodes with higher “available resources” (GAS). Link mapping is accomplished by mapping each virtual link to the shortest-path, accomplishing capacity constraints, between the corresponding mapped nodes in the SN.
- Greedy Available with Path Splitting: Virtual node and link mappings for each VNR are performed separately. Node mapping is also performed with a greedy available resources (GAS). Link mapping is accomplished with a multi-path (MP) optimal solution that avoids the NP-completeness of the problem.
- Coordinated Node and link mapping with Path Splitting: The node mapping stage is performed by defining an augmented graph over the substrate network; introducing a set of meta-nodes, one per virtual node, each connected to a cluster of candidate SN nodes obeying location and capacity constraints. The algorithm solves the VNE problem by using a Mixed Integer Programming (MIP). Its objective is to minimize the cost, i.e. the weighted sum of the bandwidth and CPU allocated in the SN links to fulfill VNR demands, of embedding a VNR. To avoid the NP-completeness of the MIP, its linear programming relaxation is solved, and the obtained solution is rounded in two ways: deterministically or randomly. The link mapping stage is performed using a Multi-Path approach (MP).
- Coordinated Node and link mapping with k-Shortest Path: The same virtual node mapping stage is performed in the same way (MIP) as the previously described algorithms. The link mapping stage is performed using a K-Shortest paths approach.

Input Parameters

Each algorithm has different input parameters. For the implemented algorithms, the parameters are as follows:

- **Greedy Available Resources (Node Mapping) + k-Shortest paths (Link Mapping):** The only parameter needed in this algorithm is the number of shortest paths to be calculated (the parameter k). An input wizard is

shown:



Setting Parameters Wizard Virtual Network (1)

Parameters

Number of k

Is distance considered? ☐ Yes ☒ No

Maximum distance

Is node overload considered? ☐ Yes ☒ No

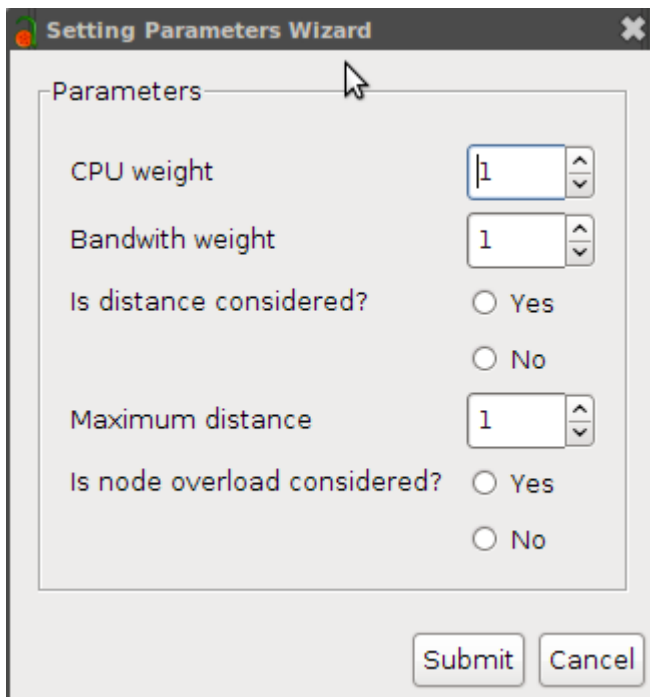
The parameters presented in the algorithm's wizard are the following:

k: Is the number of shortest-paths (in terms of hops) used in the algorithms for the virtual link mapping stage.

Distance Considered: The default value of this boolean value is “No”. This is not a value presented in the original algorithm. This value is included to provide the possibility of comparing this algorithm with subsequent proposals. If the value is chosen to “No” the normal behavior of the algorithm is presented. The value refers to the maximum distance that a substrate node can be from a virtual node to be considered a candidate node to be mapped. This value must be between 0 and 100 (the area of the space where each network is plotted is 100×100).

Node Overload: As the previous parameter the node overload is not a value presented in the original algorithm. Current algorithms do not allow that more than one virtual node, belonging to a virtual network, are mapped in one substrate node. This value modifies the algorithm to avoid this constraint. Again, If the “No” option is chosen, the algorithm will have its normal behavior.

- **Greedy Available Resources with Path Splitting (Link Mapping):** To realize the link mapping algorithm it is needed to provide the weight that user gives to the CPU and BW.



The image shows a 'Setting Parameters Wizard' dialog box. It has a title bar with a close button. The main area is titled 'Parameters' and contains several settings:

- CPU weight:** A numeric input field with the value '1' and up/down arrows.
- Bandwidth weight:** A numeric input field with the value '1' and up/down arrows.
- Is distance considered?:** Two radio buttons, 'Yes' and 'No', with 'No' selected.
- Maximum distance:** A numeric input field with the value '1' and up/down arrows.
- Is node overload considered?:** Two radio buttons, 'Yes' and 'No', with 'No' selected.

At the bottom right, there are 'Submit' and 'Cancel' buttons.

The parameters presented in the algorithm's wizard are the following:

Cpu and Bandwidth Weights: These values allow to give more revenue value to Cpu or Bw when the virtual network embedding is being performed. i.e., if Cpu weight is higher than bandwidth weight, the algorithm will perform a mapping trying to minimize the CPU consume in the substrate network.

Distance Considered: It has the same meaning than in the previous algorithm

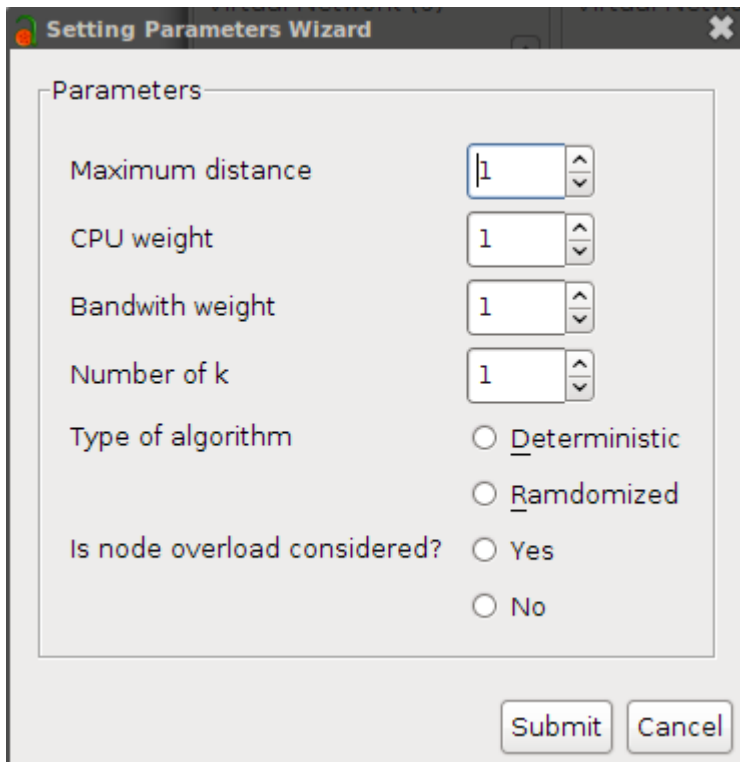
Node Overload: It has the same meaning than in the previous algorithm

Previous two algorithms are proposed in:

```
@article{Yu08,
  author = {Yu, Minlan and Yi, Yung and Rexford, Jennifer and Chiang, Mung},
  title = {Rethinking virtual network embedding: substrate support for path
splitting and migration},
  journal = {SIGCOMM Comput. Commun. Rev.},
  volume = {38},
  issue = {2},
  month = {March},
  year = {2008},
  issn = {0146-4833},
  pages = {17--29},
  numpages = {13},
  url = {http://doi.acm.org/10.1145/1355734.1355737},
  doi = {http://doi.acm.org/10.1145/1355734.1355737},
  acmid = {1355737},
  publisher = {ACM},
  address = {New York, NY, USA},
  keywords = {network virtualization, optimization, path migration, path
splitting, virtual network embedding},
}
```

- **Coordinated Node (Node Mapping) and Link Mapping with k-Shortest Paths (Link Mapping):** This solution uses the heuristic solution of the Mixed Integer Programming approach in the node mapping phase and the k-

shortest path approach in the link mapping phase.



The image shows a 'Setting Parameters Wizard' dialog box. It contains several parameters for configuration:

- Maximum distance:** A numeric input field with the value '1' and up/down arrows.
- CPU weight:** A numeric input field with the value '1' and up/down arrows.
- Bandwidth weight:** A numeric input field with the value '1' and up/down arrows.
- Number of k:** A numeric input field with the value '1' and up/down arrows.
- Type of algorithm:** Two radio button options: 'Deterministic' (selected) and 'Randomized'.
- Is node overload considered?:** Two radio button options: 'Yes' and 'No' (selected).

At the bottom right, there are 'Submit' and 'Cancel' buttons.

The parameters presented in the algorithm's wizard are the following:

k: Is the number of shortest-paths (in terms of hops) used in the algorithms for the virtual link mapping stage.

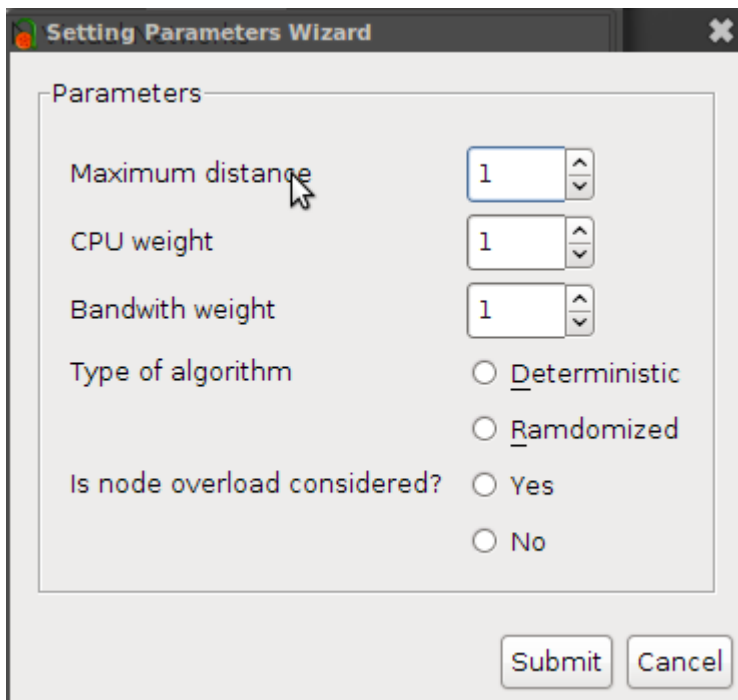
Cpu and Bandwidth Weights: These values allow to give more revenue value to Cpu or Bw when the virtual network embedding is being performed. i.e., if Cpu weight is higher than bandwidth weight, the algorithm will perform a mapping trying to minimize the CPU consume in the substrate network.

Distance Considered: It refers to the maximum distance that a substrate node can be from a virtual node to be considered a candidate node to be mapped. This value must be between 0 and 100 (the area of the space where each network is plotted is 100×100).

Node Overload: The node overload is not a value presented in the original algorithm. Current algorithms do not allow that more than one virtual node, belonging to a virtual network, are mapped in one substrate node. This value modifies the algorithm to avoid this constraint. Again, If the “No” option is chosen, the algorithm will have its normal behavior.

Algorithm Type: This algorithm works in the node mapping phase, by solving a relaxed version of the NP-Complete Unsplittable Flow Problem (UFP). After that, the relaxed solution is rounded by two methods (deterministic or randomized). This parameter contains the rounding type.

- **Coordinated Node (Node Mapping) and Link Mapping with Path Splitting (Link Mapping):** This solution uses the heuristic solution of the Mixed Integer Programming approach in the node mapping phase and a path splitting approach in the link mapping phase.



The parameters presented in the algorithm's wizard are the following:

Cpu and Bandwidth Weights: These values allow to give more revenue value to Cpu or Bw when the virtual network embedding is being performed. i.e., if Cpu weight is higher than bandwidth weight, the algorithm will perform a mapping trying to minimize the CPU consume in the substrate network.

Distance Considered: It refers to the maximum distance that a substrate node can be from a virtual node to be considered a candidate node to be mapped. This value must be between 0 and 100 (the area of the space where each network is plotted is 100×100).

Node Overload: The node overload is not a value presented in the original algorithm. Current algorithms do not allow that more than one virtual node, belonging to a virtual network, are mapped in one substrate node. This value modifies the algorithm to avoid this constraint. Again, If the “No” option is chosen, the algorithm will have its normal behavior.

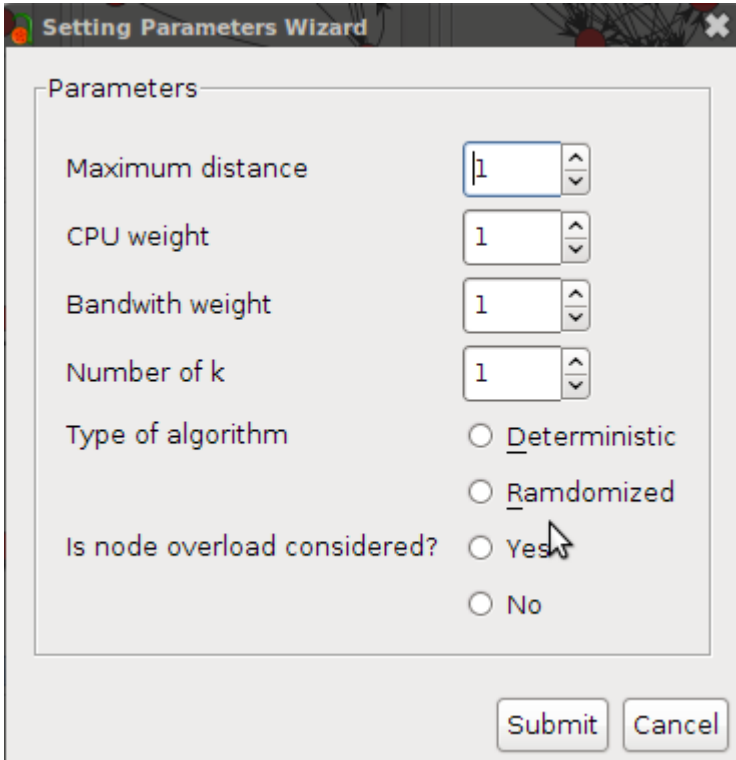
Algorithm Type: This algorithm work, in the node mapping phase, by solving a relaxed version of the NP-Complete Unsplittable Flow Problem (UFP). After that, the relaxed solution is rounded by two method (deterministic or randomized). This parameter contains the rounding type.

To see more details of previous two algorithms, please refer to:

```
@INPROCEEDINGS{CH09,
author={Chowdhury, N.M.M.K. and Rahman, M.R. and Boutaba, R.},
booktitle={INFOCOM 2009, IEEE}, title={Virtual Network Embedding with
Coordinated Node and Link Mapping},
year={2009},
month={april},
volume={},
number={},
pages={783 -791},
keywords={Internet;coordinated node;heuristic-based algorithms;link
mapping;mixed integer program;multiple heterogeneous virtual networks;network
virtualization;substrate network augmentation;virtual network
embedding;Internet;embedded systems;virtual machines;},
```

doi={10.1109/INFCOM.2009.5061987},
ISSN={0743-166X},}

- **Coordinated Node (Node Mapping) and Rounding Multipath (Link Mapping):** This solution uses the heuristic solution of the Mixed Integer Programming approach in the node mapping phase and a new approach in link mapping to realize the virtual link mapping using single path. Single path is reached by rounding the multi-path solution in the same way than in the previous proposal (Deterministically or randomly).



Setting Parameters Wizard

Parameters

Maximum distance: 1

CPU weight: 1

Bandwidth weight: 1

Number of k: 1

Type of algorithm: ☐ Deterministic ☐ Randomized

Is node overload considered? ☐ Yes ☐ No

Submit Cancel

k: Is the number of shortest-paths (in terms of hops) used in the algorithms for the virtual link mapping stage in case that rounded paths do not accomplish the virtual link demands.

Cpu and Bandwidth Weights: These values allow to give more revenue value to Cpu or Bw when the virtual network embedding is being performed. i.e., if Cpu weight is higher than bandwidth weight, the algorithm will perform a mapping trying to minimize the CPU consume in the substrate network.

Distance Considered: It refers to the maximum distance that a substrate node can be from a virtual node to be considered a candidate node to be mapped. This value must be between 0 and 100 (the area of the space where each network is plotted is 100×100).

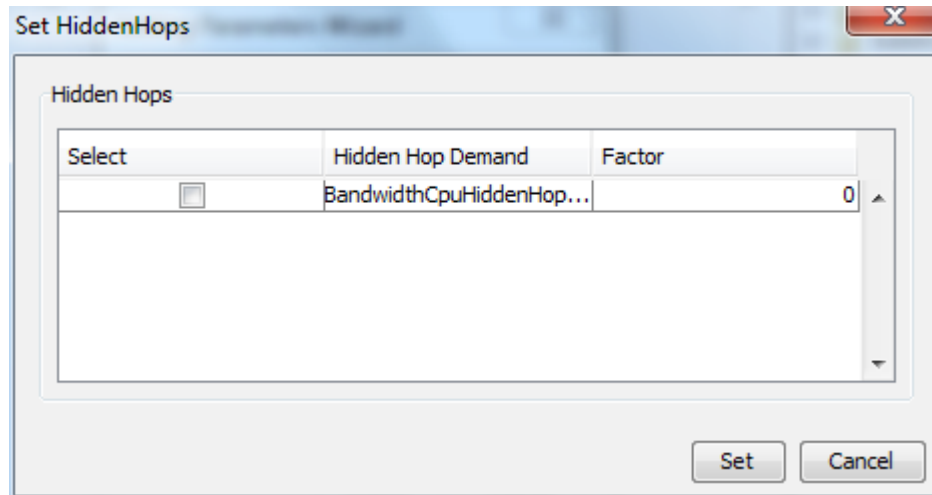
Node Overload: The node overload is not a value presented in the original algorithm. Current algorithms do not allow that more than one virtual node, belonging to a virtual network, are mapped in one substrate node. This value modifies the algorithm to avoid this constraint. Again, If the “No” option is chosen, the algorithm will have its normal behavior.

Algorithm Type: This algorithm works, in the node mapping phase, by solving a relaxed version of the NP-Complete Unsplittable Flow Problem (UFP). After that, the relaxed solution is rounded by two methods (deterministic or

randomized). This parameter contains the rounding type.

Hidden Hops Selection

In the next step, you can select the hidden hop demands that are to be considered when computing the mapping. This is done using the Hidden Hops Dialog.



Hidden Hops Dialog

It consists of a table that enables you to select the HH Demands to use and set the factor used for computing them.

Output Parameters

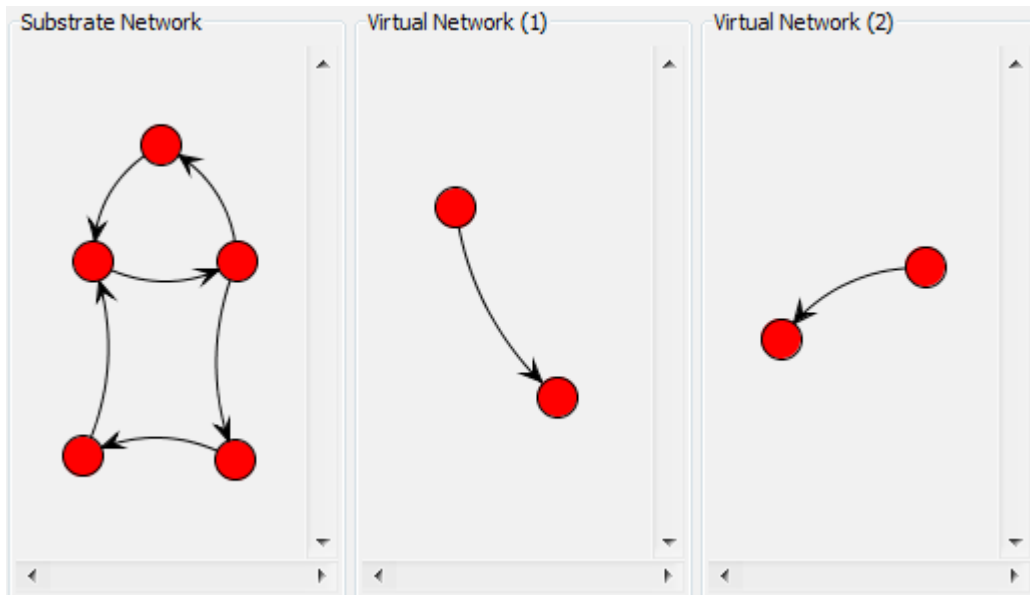
After termination some information messages are displayed in the Console. If the algorithm finishes normally, the computed mappings are added to the scenario and displayed in the Mapping Panel.

Remove All Mappings

To remove all mappings of the current scenario, select the “Remove all mappings” option from the Algorithms Menu. This action will also free up all resources in the substrate network.

5 GUI Features

5.1 The Graph Panel



The graph panel showing our exemplary scenario.

The graph panel is taken from [MuLaViTo](#) and is the central part of the GUI, where the scenarios are visualized. For each layer of the scenario, substrate or virtual, the graph of the network is displayed.

The graph panel enables you to:

5.1.1 Move Nodes

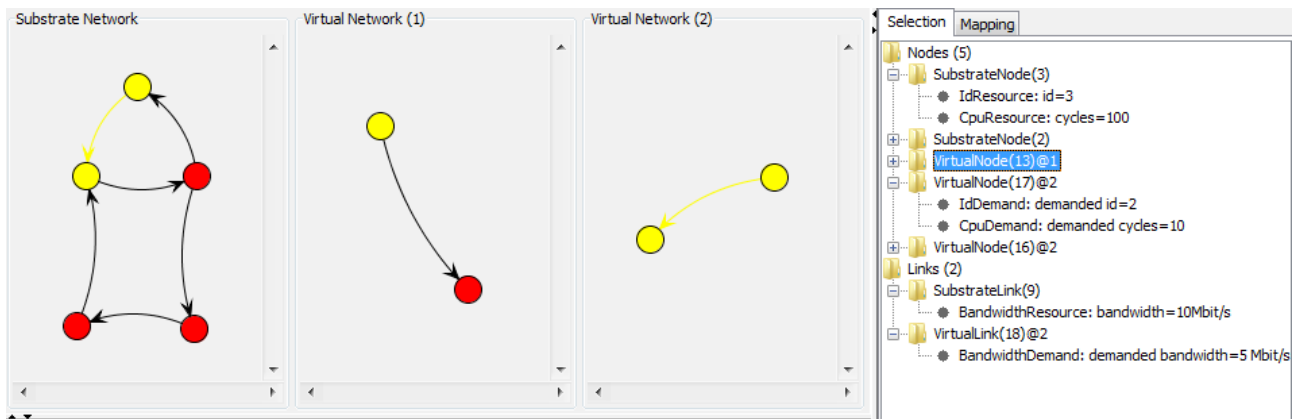
To move nodes, simply drag them with the mouse. Note that this action will have an effect on the node's coordinates.

5.1.2 Zoom in and out

To zoom in or out hold the “Ctrl” key and use the mouse wheel.

5.2 The Selection Panel

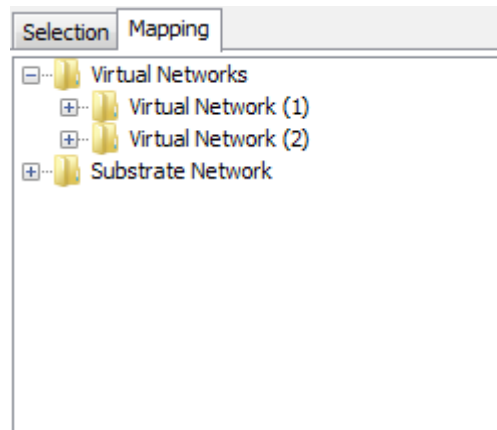
The selection panel shows the network entities currently selected as well as their constraints in a tree structure. Here is an example using our exemplary scenario. Note that some elements of the selection tree are not shown because their parent element are not popped up.



Selection panel example

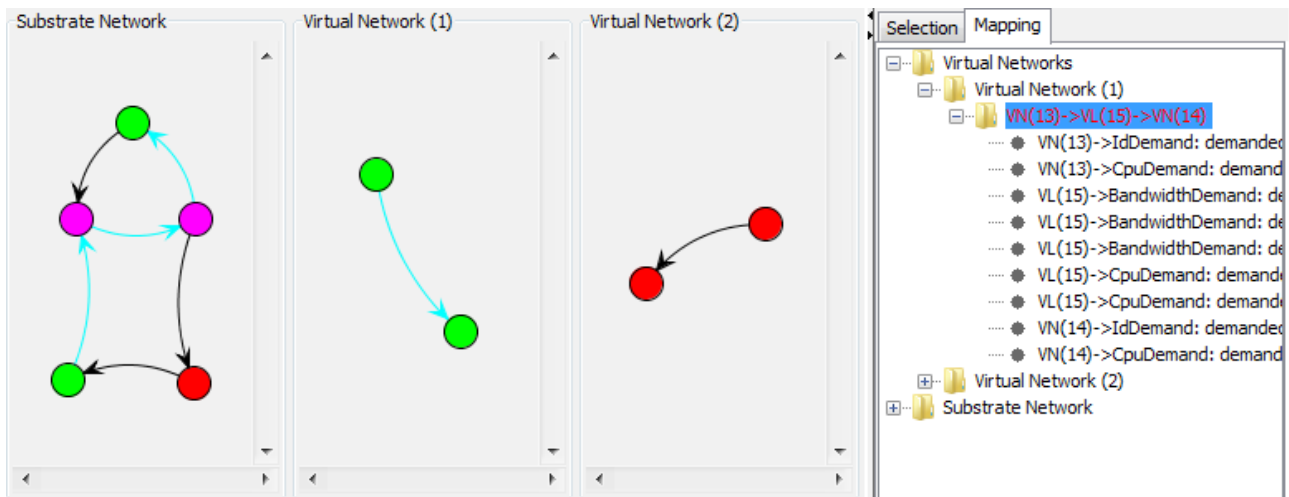
5.3 The Mapping Panel

The mapping panel shows the mappings of the current scenario in a tree structure. If no mappings exist, the virtual links will be the leaf elements of the tree:



Mapping panel showing only the virtual links as no mappings exist

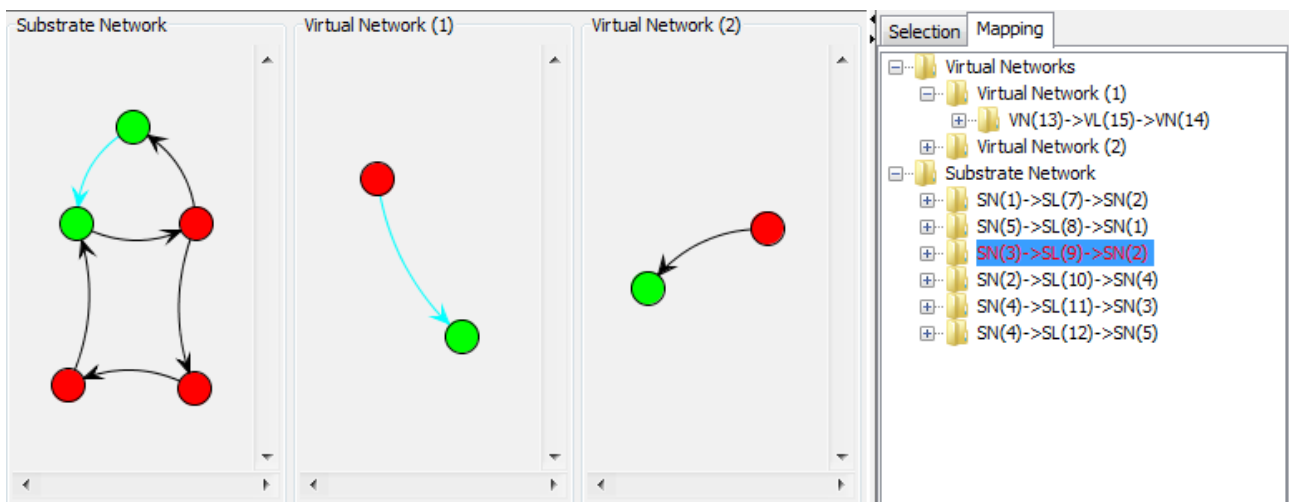
If mappings do exist, they are shown as child elements of the virtual links. When selecting a virtual link from the mappings tree, it is highlighted along with the substrate links it is mapped to. Hidden hops are displayed using a different color (magenta).



Mapping panel with mappings and highlighting

Note that the “Virtual Network (2)” node is not popped up in this example.

The Mapping panel also enables reverse highlighting. If you select a substrate link, the virtual network entities mapped on the link, or its incident nodes are highlighted.



Mapping panel with mappings and highlighting

5.4 The Console

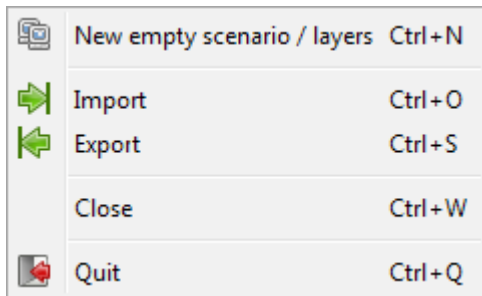
The Console is taken from MuLaViTo and displays information, debug and error messages.

The Console can be cleared or redirected to the normal terminal console.

It can also be closed and then restored by using the “Hidden Panels” option in the Views Menu.

5.5 Menu Options

5.5.1 File Menu

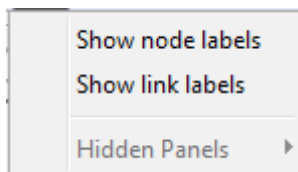


File Menu

From the File Menu you can select one of the following options

- Creating a new empty scenario or adding new layers to an existing scenario
- Importing a scenario
- Exporting a scenario
- Closing the Scenario
- Exiting the Application

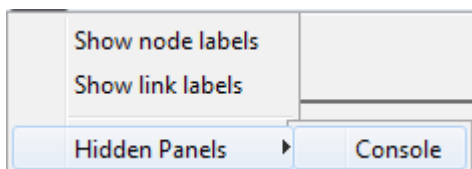
5.5.2 View Menu



*view menu
without hidden
panels*

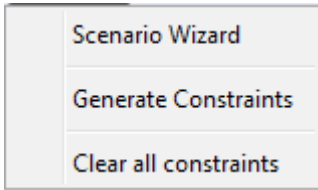
You can check the “Show node labels” or “Show link labels” options. This will display the labels of nodes / links in the graph panel.

The “Hidden Panels” menu entry provides a means to restore closed or hidden GUI components. Currently only the console can be closed and restored. This functionality is provided by MuLaViTo and is only available if there are closed GUI components available for restoration.



*view menu with hidden
console*

5.5.3 Generators Menu

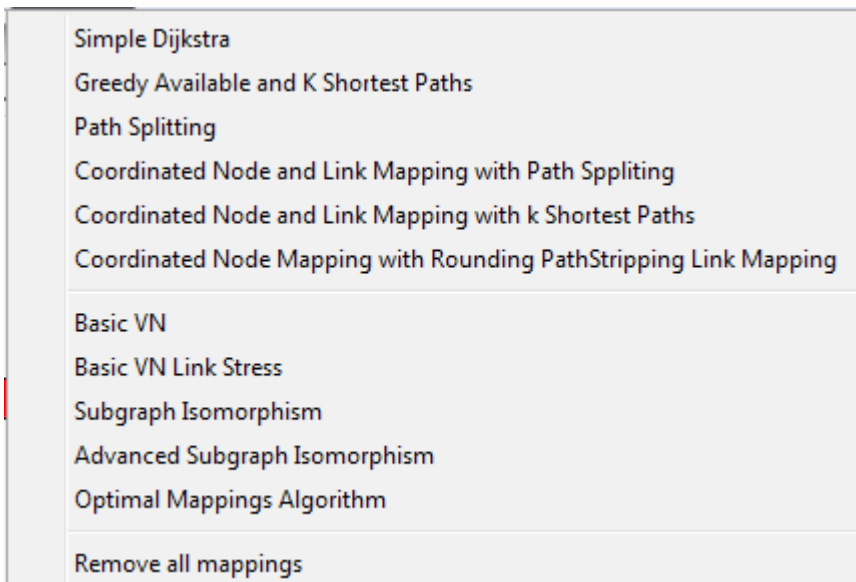


Generators Menu

The Generators Menu provides the following three options

- Generate a scenario using the Scenario Wizard.
- Generate constraints for an existing scenario.
- Remove all constraints of the current scenario.

5.5.4 Algorithms Menu



Algorithms Menu

From the algorithm menu you can either select an algorithm to be run or remove all mappings of the current scenario.

Run an Algorithm

You can run an algorithm in order to create mappings between demands and resources.

See the graphics above for a list of the currently available algorithms.

An algorithm can only be run if the current scenario has no mappings.

Remove all Mappings

This option removes all existing mappings of the current scenario and frees up all resources in the substrate network.

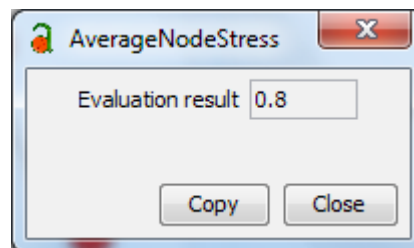
5.5.5 Metrics Menu

| |
|------------------------|
| AcceptedVnrRatio |
| AvActiveNodeStress |
| AvNodeStress |
| Cost |
| CostRevTimesMappedRev |
| CostRevenue |
| LinkCostPerVnr |
| MappedRevenue |
| RatioMappedRevenue |
| RejectedNetworksNumber |
| RemainingLinkResource |
| RevenueCost |
| RunningTime |
| TotalRevenue |

Metrics Menu

The Metrics Menu enables the evaluation of the current scenario using one of the available metrics. It is generated automatically using reflection, to ensure that all available evaluations are displayed.

To perform an evaluation on the current scenario select it from the menu. The result will be displayed in a dialog that also enables copying it. An example for the average node stress is shown below.



Metrics Dialog