
ExpertOp4Grid

Release 0.1.BETA

Antoine Marot, Mario Jothy, Nicolas Megel

Jan 07, 2022

STARTING KIT

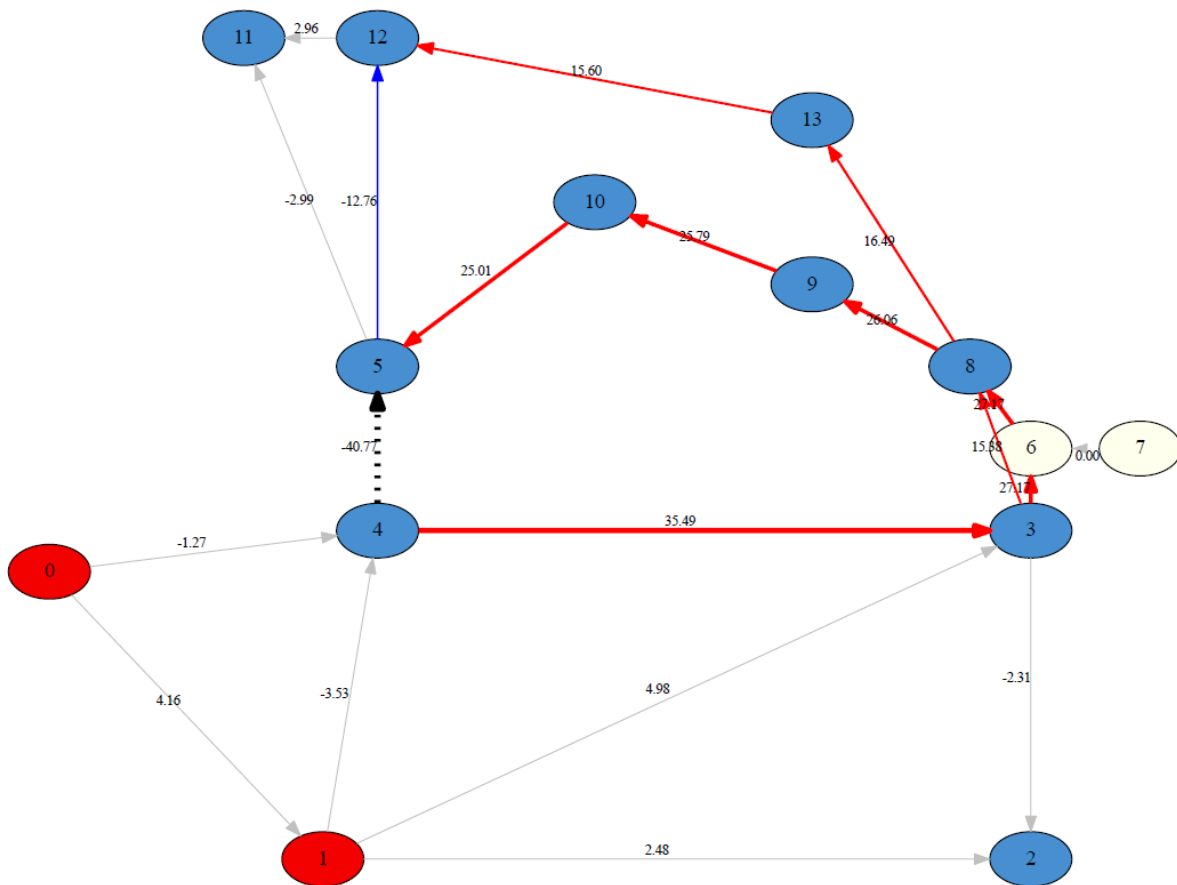
| | | |
|----------|---|-----------|
| 1 | Mentions | 1 |
| 1.1 | Quick Overview | 1 |
| 1.2 | Features | 2 |
| 1.3 | Contribute | 3 |
| 1.4 | Support | 3 |
| 1.5 | License | 3 |
| 2 | Installation | 5 |
| 2.1 | 1. (Optional)(Recommended) if you want to run in manual mode, install graphviz | 5 |
| 2.2 | 2. Install the package from Pypi | 5 |
| 2.3 | 3. (Optional) If you want to run simulation with pypowernet instead of Grid2op: | 5 |
| 2.4 | 4. (Optional) Compile and output the sphinx doc (this documentation) | 6 |
| 3 | Getting Started | 7 |
| 3.1 | Manual Mode | 7 |
| 3.2 | Agent Mode | 9 |
| 3.3 | Tests | 9 |
| 3.4 | Debug Help | 9 |
| 4 | Description | 11 |
| 4.1 | Introduction | 11 |
| 4.2 | Workflow overview | 11 |
| 4.3 | Workflow implementation | 12 |
| 4.4 | Outputs of the process | 13 |
| 4.5 | Didactic example | 15 |
| 4.6 | Important limitations | 18 |
| 5 | AlphaDeesp algorithm details | 19 |
| 5.1 | Call | 19 |
| 5.2 | Inputs | 19 |
| 5.3 | Outputs | 25 |
| 5.4 | Simulating AlphaDeesp suggestions | 26 |
| 5.5 | Last Note | 26 |

MENTIONS

1.1 Quick Overview

This is an Expert System which tries to solve a security issue on a power grid, that is on overload over a power line, when it happens. It uses cheap but non-linear topological actions to do so, and does not require any training. For any new overloaded situations, it computes an influence graph around the overload of interest, and rank the substations and topologies to explore, to find a solution. It simulates the top ranked topologies to eventually give a score of success:

4 - it solves all overloads, 3 - it solves only the overload of interest 2 - it partially solves the overload of interest 1 - it solves the overload of interest but worsen other overloads 0 - it fails. The expert agent is based It is an implementation of the paper: "Expert system for topological action discovery in smart grids" - https://hal.archives-ouvertes.fr/hal-01897931/file/_LARGE__bf_Expert_System_for_topological_remedial_action_discovery_in_smart_grids.pdf



Influence Graph example for overloaded line 4->5. The electrical paths highlighted there will help us identify interesting topologies to reroute the flows.

1.2 Features

- Analyse a power network when a line is in overflow
- Run simulations to understand the network constraints
- Return a ranking of topological actions that would solve the overflow, or reduce it
- If ran manually (through command line), can also output a series of graph to help visualise the state of the network

1.3 Contribute

- Issue Tracker: <https://github.com/marota/ExpertOp4Grid/issues>
- Source Code: <https://github.com/marota/ExpertOp4Grid>

1.4 Support

If you are having issues, please let us know. We have a discord located at: [\\$discordlink](#)

1.5 License

Copyright 2019-2020 RTE France

RTE: <http://www.rte-france.com>

This Source Code is subject to the terms of the Mozilla Public License (MPL) v2.

INSTALLATION

To install ExpertOp4Grid and AlphaDeesp execute the following lines:

2.1 1. (Optional)(Recommended) if you want to run in manual mode, install graphviz

This is for neato package, it allows to transform a dot file into a pdf file.

Warning: It is important to install graphviz executables before python packages

First install executable

- On Linux

```
apt-get install graphviz
```

- On Windows, use package finder (equivalent of apt-get on Windows)

```
winget install graphviz
```

Then ensure that graphviz and neato are in the path. You often have to set it manually. For example on windows you can use the following command line:

```
setx /M path "%path%; 'C:\Users\username\graphviz-2.38\release\bin"
```

Then you can move to python packages installation

2.2 2. Install the package from Pypi

```
pip install ExpertOp4Grid
```

2.3 3. (Optional) If you want to run simulation with pypowernet instead of Grid2op:

- Clone pypowernet somewhere else :

```
cd .. git clone https://github.com/MarvinLer/pypowernet.git
```

- Install from within that folder:

```
python setup.py install --user
```

or

```
cd ExpertOp4Grid pipenv shell cd ../pypownet python setup.py install
```

2.4 4. (Optional) Compile and output the sphinx doc (this documentation)

Run `./docs/make.bat html`

GETTING STARTED

3.1 Manual Mode

To execute in **manual mode**, type: `expertop4grid -l 9 -s 0 -c 0 -t 0`

-l **int** Integer representing the line to cut. For the moment, only one line to cut is handled

-snapshot **-s int** If 1, will generate plots of the different grid topologies managed by alphadeesp and store it in alphadeesp/ressources/output

-chronicscenario **-c string** Name of the folder containing the chronic scenario to consider By default, the first available folder will be chosen

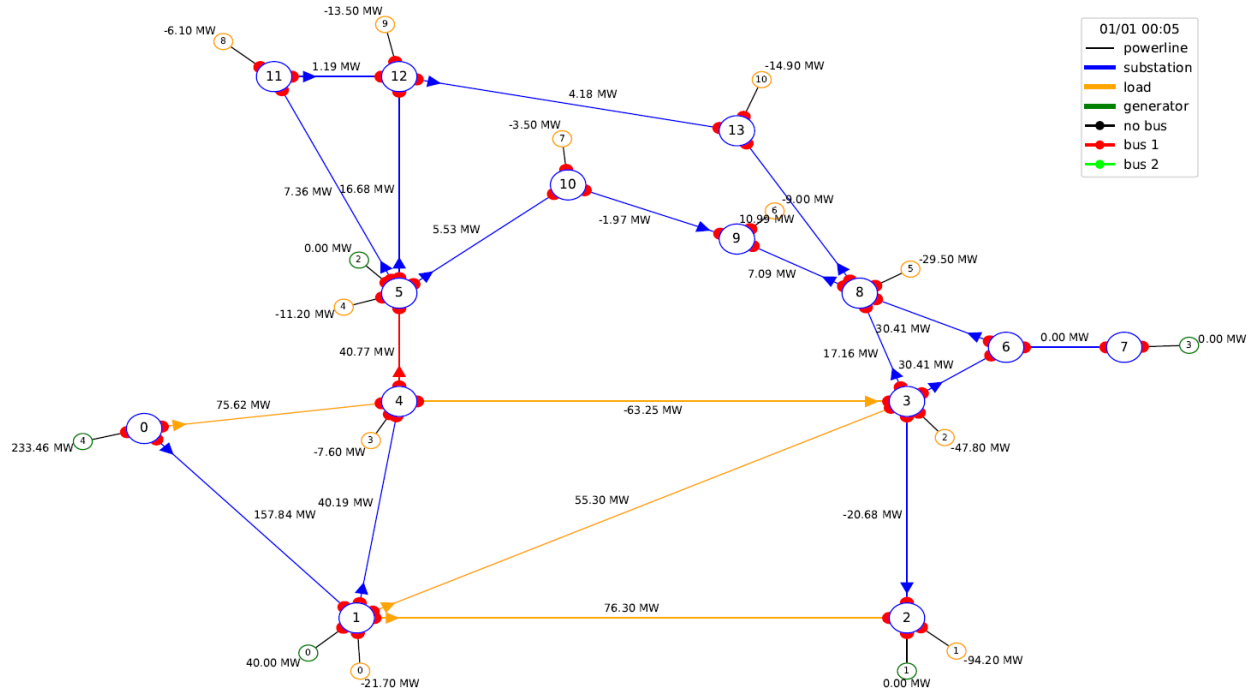
-timestep **-t int** Integer representing the timestep number at which we want to run alphadeesp simulation

-fileconfig **-f string** Path to .ini file that provides detailed configuration of the module. If none is provided, a default config.ini is provided in package

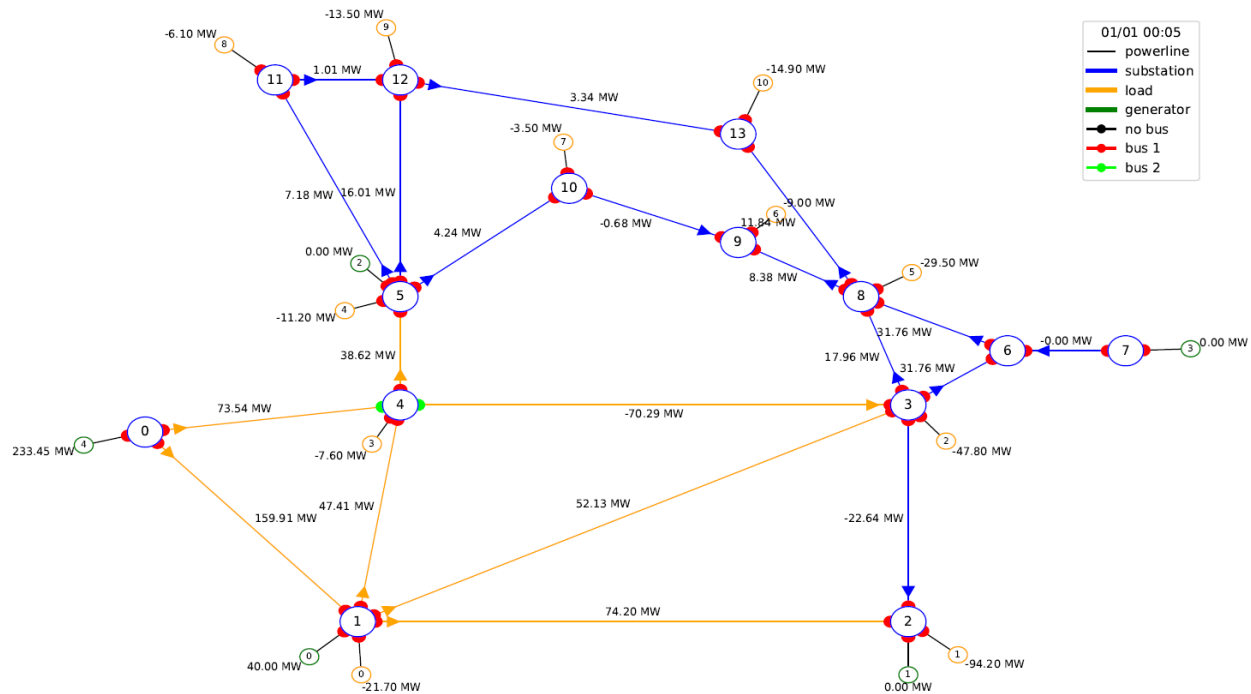
In any case, an end result dataframe is written in root folder.

If you run the same command with ‘-s 1’ to print the plots, you will indeed see that:

- On the initial state, you had an overflow to solve



- The expert system indeed finds a solution topology for it at substation 4



See **Algorithm Description** section to learn more about the workflow and results.

In manual mode, further configuration is made through `alphadeesp/config.ini`

- `simulatorType` - you can chose Grid2op or Pypownet
- `gridPath` - path to folder containing files representing the grid. If no value is provided, a default grid will be loaded (12rpn_2019) containing one chronic as a simple usecase example

- *outputPath* - path to write outputs in case snapshot mode is activated. If no path is provided, ExpertOp4Grid will write image results in the current working directory (folder named output/grid/linetocut/scenario/timestep)
- *CustomLayout* - list of couples representing coordinates of grid nodes. If not provided, grid2op will load grid_layout.json in grid folder
- *grid2opDifficulty* - “0”, “1”, “2” or “competition”. Be careful: grid datasets should have a difficulty_levels.json
- *7 other constants for alphadeesp computation* can be set in config.ini, with comments within the file

3.2 Agent Mode

To execute in **agent mode**, please refer to ExpertAgent available in l2rpn-baseline repository

https://github.com/mjothy/l2rpn-baselines/tree/mj-devs/l2rpn_baselines/ExpertAgent

Instead of configuring through config.ini, you can pass a similar python dictionary to the API

3.3 Tests

To launch the test suite in git repo: `pipenv run python -m pytest --verbose --continue-on-collection-errors -p no:warnings`

3.4 Debug Help

- To force specific hubs

in AlphaDeesp.compute_best_topo() function, one can force override the hubs result. Check in code, there are commented examples.

- To force specific combinations for hubs

If one wants a specific hub, a user can “force” a specific node combination. Check in the code, there are commented examples

DESCRIPTION

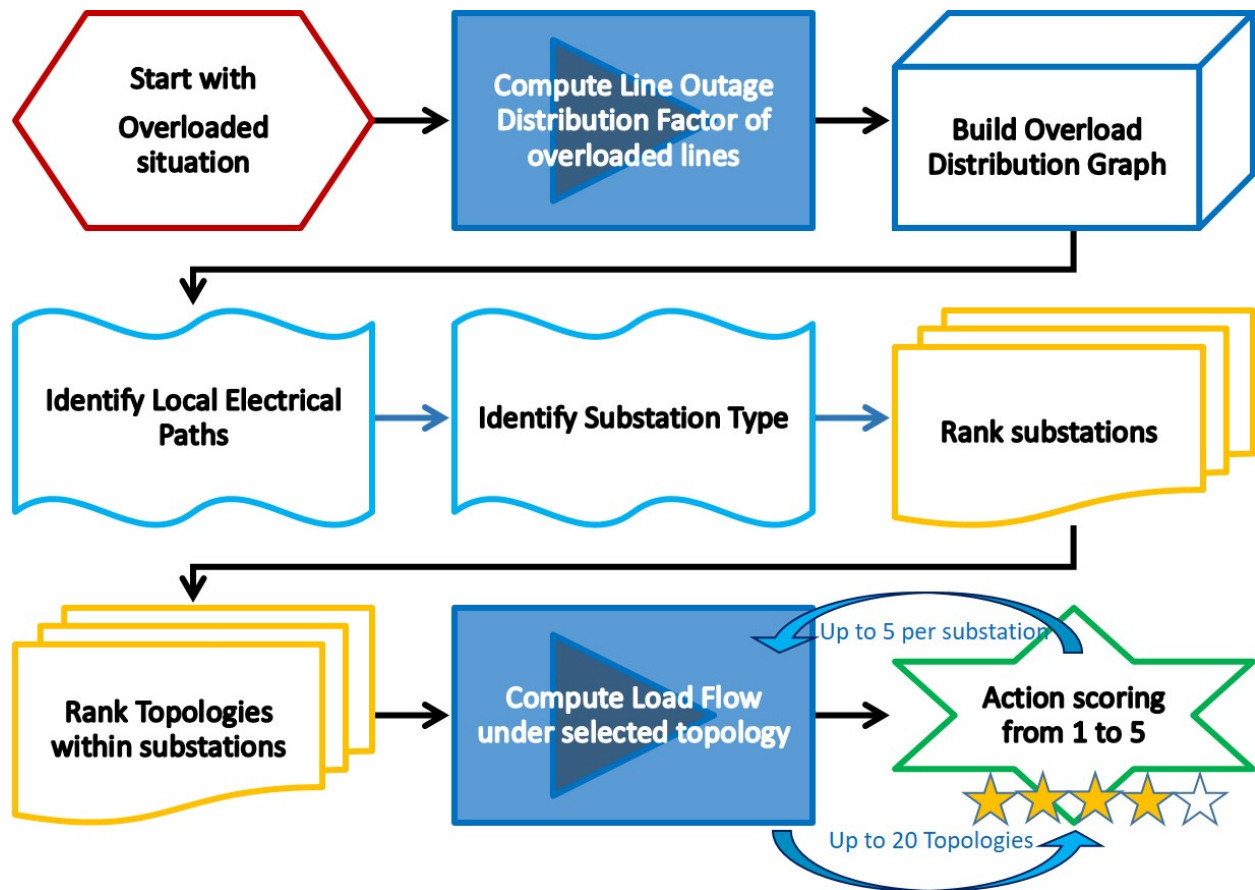
4.1 Introduction

This module represents an expert agent that finds solutions to optimize a power network. The expert agent is based on a research paper (https://hal.archives-ouvertes.fr/hal-01897931/file/_LARGE__bf_Expert_System_for_topological_remedial_action_discovery_in_smart_grids.pdf)

Given a power grid and a line in overflow (referred as *Line to cut*) the expert agent will run simulations on the network and try to find and rank the best topological actions (changing elements of the graph from one bus to the other) to hopefully solve the overflow.

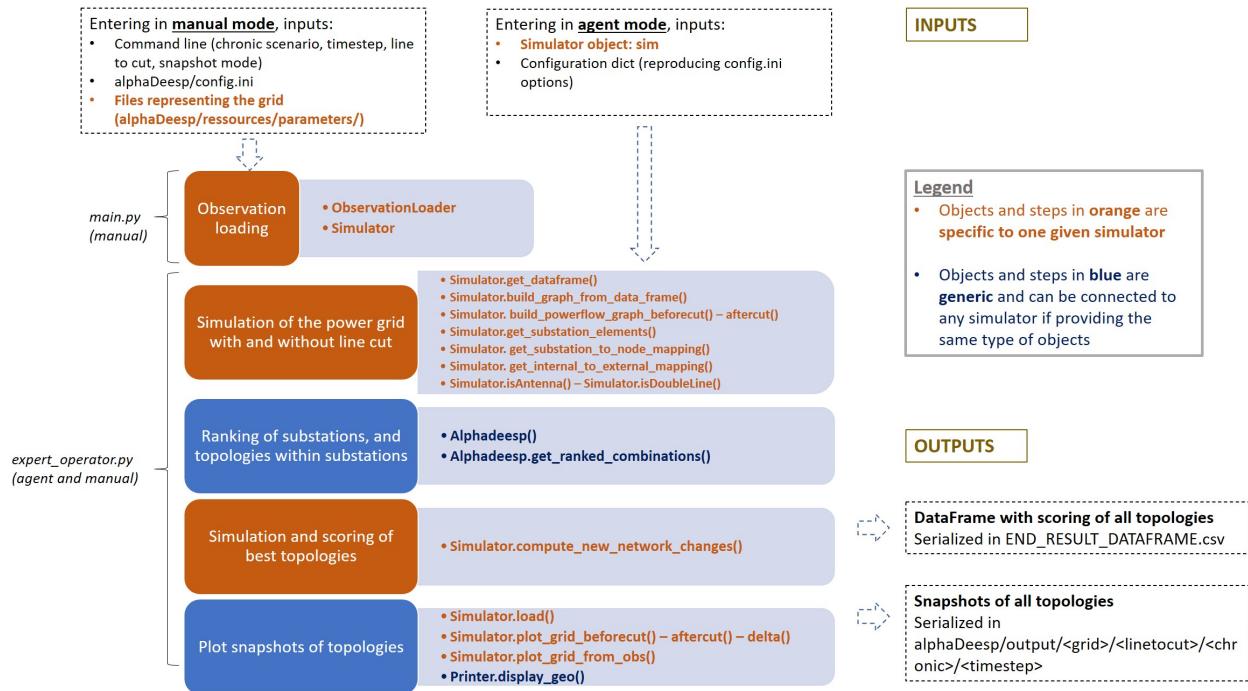
4.2 Workflow overview

We can decompose the Expert System algorithm along those successive steps. Based on the overload distribution graphs it builds, it tries to identify relevant patterns in it described by expert knowledge, to eventually find good spots to reroute the flows. It ranks those substations apriori by relevance and then test them by simulation to get a final score of success. Notice that 2 of those steps involves running simulations: it indeed relies on a simulator backend to work.



4.3 Workflow implementation

The following picture shows an overview of the different inputs and outputs, and of the major modelisation steps.



Whether entering in manual mode or agent mode, different sets of inputs are provided.

Objects and steps in orange are specific to one given simulator. Two objects are manipulated as such

- `ObservationLoader` object
- `Simulator` object

See examples of implementation with Grid2op simulations and Pypowernet simulations, in following scripts

- `alphaDeesp/core/grid2op/Grid2opObservationLoader.py`
- `alphaDeesp/core/grid2op/Grid2opSimulation.py`
- `alphaDeesp/core/pypowernet/PypowernetObservationLoader.py`
- `alphaDeesp/core/pypowernet/PypowernetSimulation.py`

It can be substituted by your favorite simulator if it provides the same interface methods and returns the same type of objects to be able to work with AlphaDeesp

4.4 Outputs of the process

When called, the `main.py/expert_operator.py` will return three objects : `ranked_combinations`, `expert_system_results`, `action`

4.4.1 ranked_combinations

This dict contains all topological configurations per node, sorted from best to worst by the simulation.

4.4.2 expert_system_results

The main dataframe presenting quantitative information for all simulated topologies

| overflow ID | Flows before | Flows after | Delta flows | Worsened line | Prod redispatch | Load redispatch | Internal Topo | Topology app | Substation ID | Rank Substati | Topology scor | Topology simi | Efficacy |
|-------------|--------------|---------------|---------------|---------------|-----------------|-----------------|---------------|-----------------------------|---------------|---------------|---------------|---------------|---------------------|
| 0 9.0 | 40.768272395 | -7.599999904 | 48.368270874 | [6] | 4.1723480224 | 0.0 | [0 0 1 1 1] | [2, 2, 2, 1, 1] | 4.0 | 1.0 | 40.29 | 1.0 | 249.5369873046875 |
| 1 9.0 | 40.768272395 | 48.569595334 | -7.801322937 | [9] | 0.8322448736 | 0.0 | [0 0 0 1 1] | [1, 2, 2, 1, 1] | 4.0 | 1.0 | 39.02 | 0.0 | -6.4611897468566895 |
| 2 9.0 | 40.768272395 | 38.617652895 | 2.1506195064 | [] | 0.0117034914 | 0.0 | [0 0 1 0 1] | [2, 1, 2, 1, 1] | 4.0 | 1.0 | 36.76 | 4.0 | 2.2398364543914795 |
| 3 9.0 | 40.768272395 | 51.023971557 | -10.25569915 | [9] | 1.2664031982 | 0.0 | [0 1 1 0 0] | [2, 1, 1, 2, 1] | 4.0 | 1.0 | 31.42 | 0.0 | -8.08720874786377 |
| 4 9.0 | 40.768272395 | 40.5486221310 | 2.196502684 | [] | 0.1283416744 | 0.0 | [0 1 0 1 0] | [1, 2, 1, 2, 1] | 4.0 | 1.0 | 29.16 | 2.0 | 0.2179497927427292 |
| 5 9.0 | 40.768272395 | 58.262042995 | -17.49377059 | [2, 3, 9] | 3.6772613525 | 0.0 | [0 1 1 1 0] | [2, 2, 1, 2, 1] | 4.0 | 1.0 | 27.89 | 0.0 | -12.063469886779785 |
| 6 9.0 | 40.768272395 | 35.204250335 | 5.564022064 | [2, 3] | 12.354400634 | 3.0517578125 | [0 0 1 1 0] | [2, 2, 1, 1, 1] | 4.0 | 1.0 | 4.8 | 1.0 | 6.346175193786621 |
| 7 9.0 | 40.768272395 | 35.5310478215 | 2.372245784 | [2, 3] | 10.698944091 | 3.0517578125 | [0 1 0 0 1] | [1, 1, 2, 2, 1] | 4.0 | 1.0 | -2.8 | 1.0 | 5.9294657707214355 |
| 8 9.0 | 40.768272395 | 39.739013671 | 1.0292587284 | [] | 0.4824066162 | 0.0 | [0 1 1 0 1] | [2, 1, 2, 2, 1] | 4.0 | 1.0 | -4.07 | 4.0 | 1.032004952430725 |
| 9 9.0 | 40.768272395 | 39.099594114 | 1.6686782834 | [2] | 6.2888183595 | 0.0 | [0 1 0 1 1] | [1, 2, 2, 2, 1] | 4.0 | 1.0 | -6.33 | 1.0 | 1.7099021673202515 |
| 10 9.0 | 40.768272395 | -2.886579864 | 40.768272395 | [6] | 4.3804016113 | 3.0517578125 | [0 1 1 1 0] | [1, 2, 2, 2, 1, 2, 5, 0] | 1.0 | 1.0 | 51.96 | 1.0 | 227.79396057128906 |
| 11 9.0 | 40.768272395 | 26.340831754 | 14.427440644 | [] | 2.8236846923 | 0.0 | [0 1 1 0 1 0] | [1, 2, 1, 2, 1, 2, 5, 0] | 1.0 | 1.0 | 48.97 | 4.0 | 21.288515090942383 |
| 12 9.0 | 40.768272395 | 26.315893175 | 14.452379224 | [] | 2.1344146724 | 0.0 | [0 0 0 1 0 1] | [2, 1, 2, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 48.97 | 4.0 | 22.668455123901367 |
| 13 9.0 | 40.768272395 | 11.199999805 | 29.568271634 | [6] | 1.9164123535 | 3.0517578125 | [0 0 1 1 1 0] | [1, 2, 2, 2, 1, 1, 5, 0] | 1.0 | 1.0 | 40.76 | 1.0 | 85.57772064208984 |
| 14 9.0 | 40.768272395 | 11.199999805 | 29.568271634 | [6] | 1.5085296630 | 0.0 | [0 1 0 0 0 1] | [2, 1, 1, 1, 1, 2, 5, 0] | 1.0 | 1.0 | 40.76 | 1.0 | 88.72957611083984 |
| 15 9.0 | 40.768272395 | 33.153594974 | 7.6146774291 | [] | 1.1479797363 | 0.0 | [0 0 1 0 0 1] | [2, 2, 1, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 39.2 | 4.0 | 9.455564498901367 |
| 16 9.0 | 40.768272395 | 33.207462314 | 7.5608100891 | [] | 1.5565948484 | 0.0 | [0 1 0 1 1 0] | [1, 1, 2, 2, 1, 1, 2, 5, 0] | 1.0 | 1.0 | 39.2 | 4.0 | 8.970056533813477 |
| 17 9.0 | 40.768272395 | 30.302978515 | 10.465293884 | [] | 1.0005798335 | 3.0517578125 | [0 1 0 1 0 1] | [2, 1, 2, 1, 1, 1, 2, 5, 0] | 1.0 | 1.0 | 37.77 | 4.0 | 13.942999839782715 |
| 18 9.0 | 40.768272395 | 30.532775874 | 10.235496524 | [] | 1.1595611572 | 0.0 | [0 0 1 0 1 0] | [1, 2, 1, 2, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 37.77 | 4.0 | 13.114059448242188 |
| 19 9.0 | 40.768272395 | 30.418809894 | 10.349462504 | [] | 0.8561859134 | 0.0 | [0 0 1 1 0 1] | [2, 2, 2, 1, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 36.21 | 4.0 | 13.747495651245117 |
| 20 9.0 | 40.768272395 | 32.245273594 | 8.5229988094 | [] | 2.0540618894 | 0.0 | [0 0 1 1] | [2, 2, 1, 1] | 12.0 | 1.0 | 29.1 | 4.0 | 10.400897026062012 |
| 21 9.0 | 40.768272395 | 42.203769684 | -1.4354972834 | [] | 0.9725799564 | 0.0 | [0 1 1 0] | [2, 1, 2, 1] | 12.0 | 1.0 | 16.46 | 0.0 | -1.3525135517120361 |
| 22 9.0 | 40.768272395 | 40.057800295 | 0.71047211064 | [] | 0.0380249024 | 0.0 | [0 1 0 1] | [1, 2, 2, 1] | 12.0 | 1.0 | 15.6 | 2.0 | 0.7036463022232056 |

- Simulated flows on the target line (line to cut) before and after topological actions is operated. The delta flow is the difference between both of them
- Worsened lines: new lines that got overloaded or initially overloaded lines which overload increased
- Redispatched Prod: sum of all the production increase or decrease at each generator
- Redispatched Load: difference between the total demand and the actual power supply in all loads (production - losses)
- Internal Topology applied: topology list as used in AlphaDeesp. Represents the bus of each element at the substation (column Substation ID)
- Topology applied: topology list as used in the simulator. Represents the bus of each element at the substation (column Substation ID)
- Substation ID: ID of the substation on which the topology is applied
- Topology score: quantitative score returned by Alphadeesp for this topology
- Topology simulated score: integer score (from 0 to 4) given by the simulator when computing powerflow on the grid after applying the topology
- Efficacy: flexible quantitative score to be returned by the simulator. It can for instance take into account the reward after applying the topological action

In manual mode, if option `-snapshot` is set to 1, plots of all simulated topos graphs are generated to dig into their consequences on the grid powerflow distribution. Plot names are meant to facilitate links between the snapshot and its correspondign line in the dataframe. See plots in the following didactic example.

4.4.3 action

This list contains all actions (generated by and for the chosen backend) that represent the topological states chosen by Alphadeesp. Sorted from best to worst, they are a syntactic sugar to give users a direct action to apply to their network. It is still recommended to parse the main dataframe to understand better what solutions are available.

4.5 Didactic example

We launch the expert operator in manual mode on the grid l2rpn_2019, with the Grid2opSimulator, on scenario a, at first timestep. There is an overflow on line 9 (between substation 4 and 5), so we provide $l_{tc} = 9$. We want to see snapshots of the grid.

- Command line

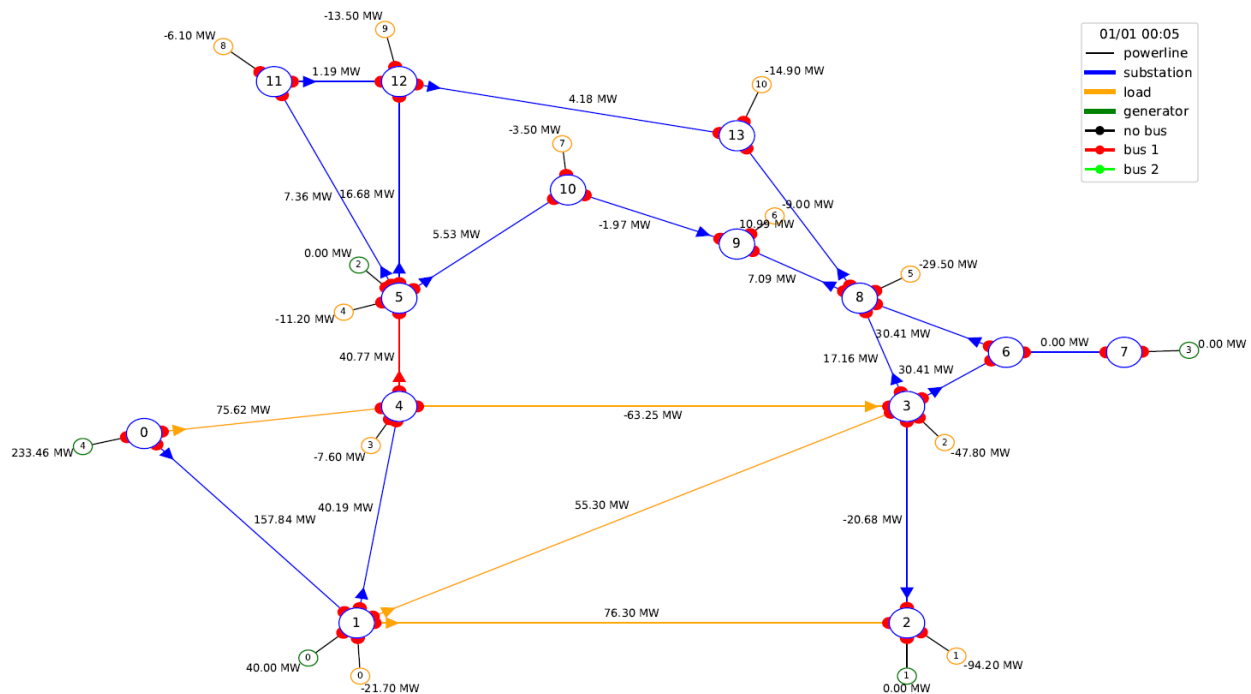
```
pipenv run python -m alphaDeesp.main -l 9 -s 0 -c 0 -t 0
```

- Beginning of config.ini

```
[DEFAULT]
# Simulator choice, either: "Pypownet" or "RTE"
simulatorType = Grid2OP
#simulatorType = Pypownet
#simulatorType = RTE

# Path to grid representation files
gridPath = ./alphaDeesp/ressources/parameters/l2rpn_2019
#gridPath = ./alphaDeesp/ressources/parameters/rte_case14_realistic
#gridPath = ./alphaDeesp/ressources/parameters/default14_static
```

- Layout of the grid in its current state (also called g_{pow})

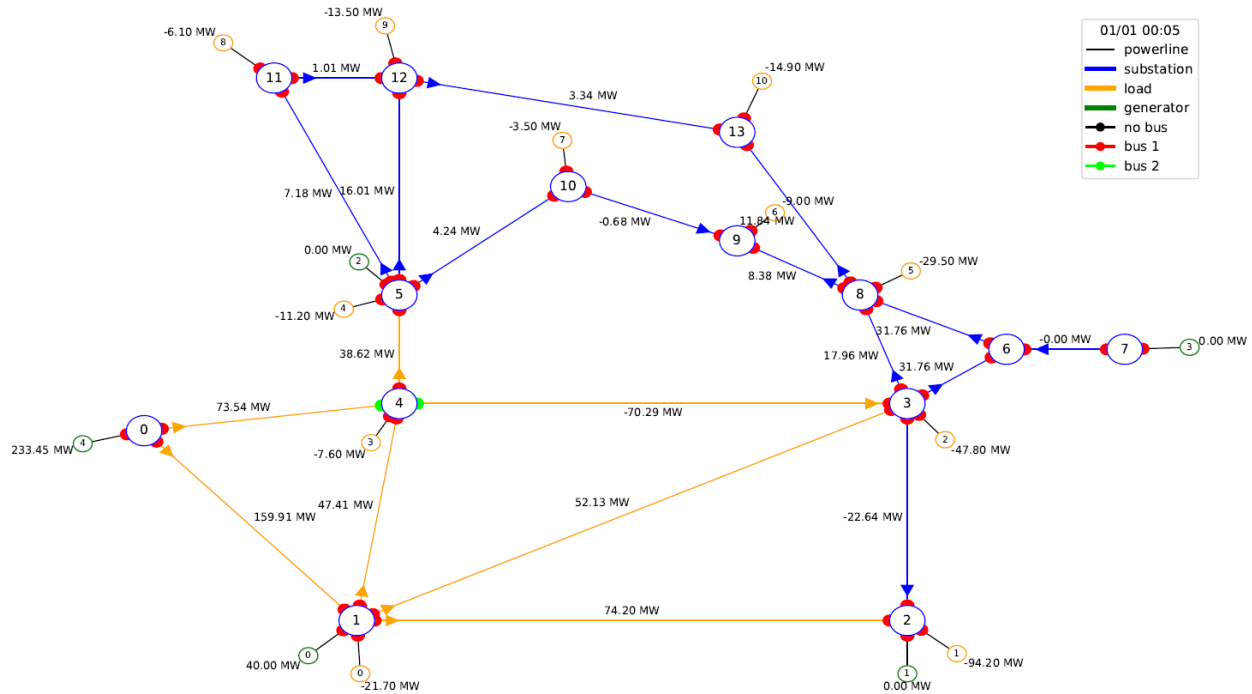


The simulator will then compute several objects to provide to AlphaDeesp, which will run a greedy algorithm to determine the best topological action to solve the overload. For more details, see the section *Algorithm Details*

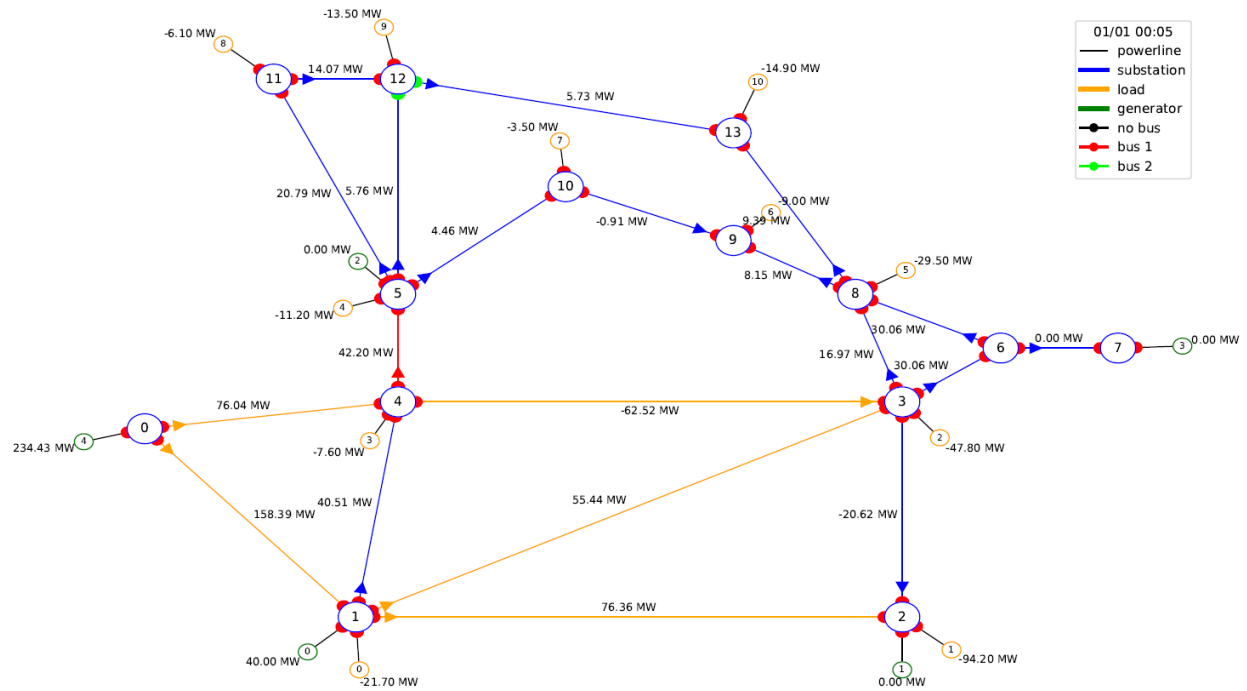
- AlphaDeesp will then output a dataframe with all computed details about the best topologies found.

| overflow ID | Flows before | Flows after | Delta flows | Worsened line | Prod redispatch | Load redispatch | Internal Topology | Topology app | Substation ID | Rank | Substati | Topology score | Topology simi | Efficacy |
|-------------|--------------|--------------|--------------|---------------|-----------------|-----------------|-------------------|--------------------------|---------------|------|----------|----------------|---------------------|----------|
| 0 9.0 | 40.76827239 | -7.59999904 | 48.36827087 | [6] | 4.172348022 | 0.0 | [0 0 1 1 1] | [2, 2, 2, 1, 1] | 4.0 | 1.0 | 40.29 | 1.0 | 249.5369873046875 | |
| 1 9.0 | 40.76827239 | 48.56959533 | -7.801322937 | [9] | 0.832244873 | 0.0 | [0 0 0 1 1] | [1, 2, 2, 1, 1] | 4.0 | 1.0 | 39.02 | 0.0 | -6.4611897468566895 | |
| 2 9.0 | 40.76827239 | 38.61765289 | 2.150619506 | [1] | 0.011703491 | 0.0 | [0 0 1 0 1] | [2, 1, 2, 1, 1] | 4.0 | 1.0 | 36.76 | 4.0 | 2.2398365453914795 | |
| 3 9.0 | 40.76827239 | 51.02397155 | -10.25569915 | [9] | 1.266403198 | 0.0 | [0 1 1 0 0] | [2, 1, 1, 2, 1] | 4.0 | 1.0 | 31.42 | 0.0 | -8.08720874786377 | |
| 4 9.0 | 40.76827239 | 40.548622131 | 0.219650268 | [1] | 0.128341674 | 0.0 | [0 1 0 1 0] | [1, 2, 1, 2, 1] | 4.0 | 1.0 | 29.16 | 2.0 | 0.2179497927427292 | |
| 5 9.0 | 40.76827239 | 58.26204299 | -17.49377059 | [2, 3, 9] | 3.677261352 | 0.0 | [0 1 1 1 0] | [2, 2, 1, 2, 1] | 4.0 | 1.0 | 27.89 | 0.0 | -12.063469886779785 | |
| 6 9.0 | 40.76827239 | 35.20425033 | 5.564022064 | [2, 3] | 12.35440063 | 3.051757812 | [0 0 1 1 0] | [2, 2, 1, 1, 1] | 4.0 | 1.0 | 4.8 | 1.0 | 6.346175193786621 | |
| 7 9.0 | 40.76827239 | 35.53104782 | 15.237224578 | [2, 3] | 10.69894409 | 3.051757812 | [0 1 0 0 1] | [1, 1, 2, 2, 1] | 4.0 | 1.0 | -2.8 | 1.0 | 5.9294657707214355 | |
| 8 9.0 | 40.76827239 | 39.73901367 | 11.029258728 | [1] | 0.482406616 | 0.0 | [0 1 1 0 1] | [2, 1, 2, 2, 1] | 4.0 | 1.0 | -4.07 | 4.0 | 1.032004952430725 | |
| 9 9.0 | 40.76827239 | 39.09959411 | 11.66867828 | [2] | 6.288818359 | 0.0 | [0 1 0 1 1] | [1, 2, 2, 2, 1] | 4.0 | 1.0 | -6.33 | 1.0 | 1.7099021673202515 | |
| 10 9.0 | 40.76827239 | -2.88657964 | 40.76827239 | [6] | 4.380401611 | 3.051757812 | [0 1 1 1 0] | [1, 2, 2, 2, 1, 2, 5, 0] | 1.0 | 1.0 | 51.96 | 1.0 | 227.79396057128906 | |
| 11 9.0 | 40.76827239 | 26.34083175 | 14.42744064 | [1] | 2.823684692 | 0.0 | [0 1 1 0 1] | [1, 2, 1, 2, 1, 2, 5, 0] | 1.0 | 1.0 | 48.97 | 4.0 | 21.288515090942383 | |
| 12 9.0 | 40.76827239 | 26.31589317 | 14.45237922 | [1] | 2.134414672 | 0.0 | [0 0 0 1 0] | [2, 1, 2, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 48.97 | 4.0 | 22.668455123901367 | |
| 13 9.0 | 40.76827239 | 11.19999980 | 29.56827163 | [6] | 1.916412353 | 3.051757812 | [0 0 1 1 0] | [1, 2, 2, 2, 1, 1, 5, 0] | 1.0 | 1.0 | 40.76 | 1.0 | 85.57772064208984 | |
| 14 9.0 | 40.76827239 | 11.19999980 | 29.56827163 | [6] | 1.508529663 | 0.0 | [0 1 0 0 1] | [2, 1, 1, 1, 2, 5, 0] | 1.0 | 1.0 | 40.76 | 1.0 | 88.72957611083984 | |
| 15 9.0 | 40.76827239 | 33.15359497 | 7.614677429 | [1] | 1.147979736 | 0.0 | [0 0 1 0 0] | [2, 2, 1, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 39.2 | 4.0 | 9.455564498901367 | |
| 16 9.0 | 40.76827239 | 33.20746231 | 7.560810089 | [1] | 1.556594848 | 0.0 | [0 1 0 1 0] | [1, 1, 2, 2, 1, 2, 5, 0] | 1.0 | 1.0 | 39.2 | 4.0 | 8.970056533813477 | |
| 17 9.0 | 40.76827239 | 30.30297851 | 10.46529388 | [1] | 1.000579833 | 3.051757812 | [0 1 0 1 0] | [2, 1, 2, 1, 1, 2, 5, 0] | 1.0 | 1.0 | 37.77 | 4.0 | 13.942999839782715 | |
| 18 9.0 | 40.76827239 | 30.53277587 | 10.23549652 | [1] | 1.159561157 | 0.0 | [0 0 1 0 1] | [1, 2, 1, 2, 1, 1, 5, 0] | 1.0 | 1.0 | 37.77 | 4.0 | 13.114059484242188 | |
| 19 9.0 | 40.76827239 | 30.41880989 | 10.34946250 | [1] | 0.856185913 | 0.0 | [0 0 1 1 0] | [2, 2, 2, 1, 1, 1, 5, 0] | 1.0 | 1.0 | 36.21 | 4.0 | 13.747495651245117 | |
| 20 9.0 | 40.76827239 | 32.24527359 | 8.522998809 | [1] | 2.054061889 | 0.0 | [0 0 1 1] | [2, 2, 1, 1] | 12.0 | 1.0 | 29.1 | 4.0 | 10.400897026062012 | |
| 21 9.0 | 40.76827239 | 42.20376968 | -1.435497283 | [1] | 0.972579956 | 0.0 | [0 1 1 0] | [2, 1, 2, 1] | 12.0 | 1.0 | 16.46 | 0.0 | -1.3525135517120361 | |
| 22 9.0 | 40.76827239 | 40.05780029 | 0.710472106 | [1] | 0.038024902 | 0.0 | [0 1 0 1] | [1, 2, 2, 1] | 12.0 | 1.0 | 15.6 | 2.0 | 0.7036463022232056 | |

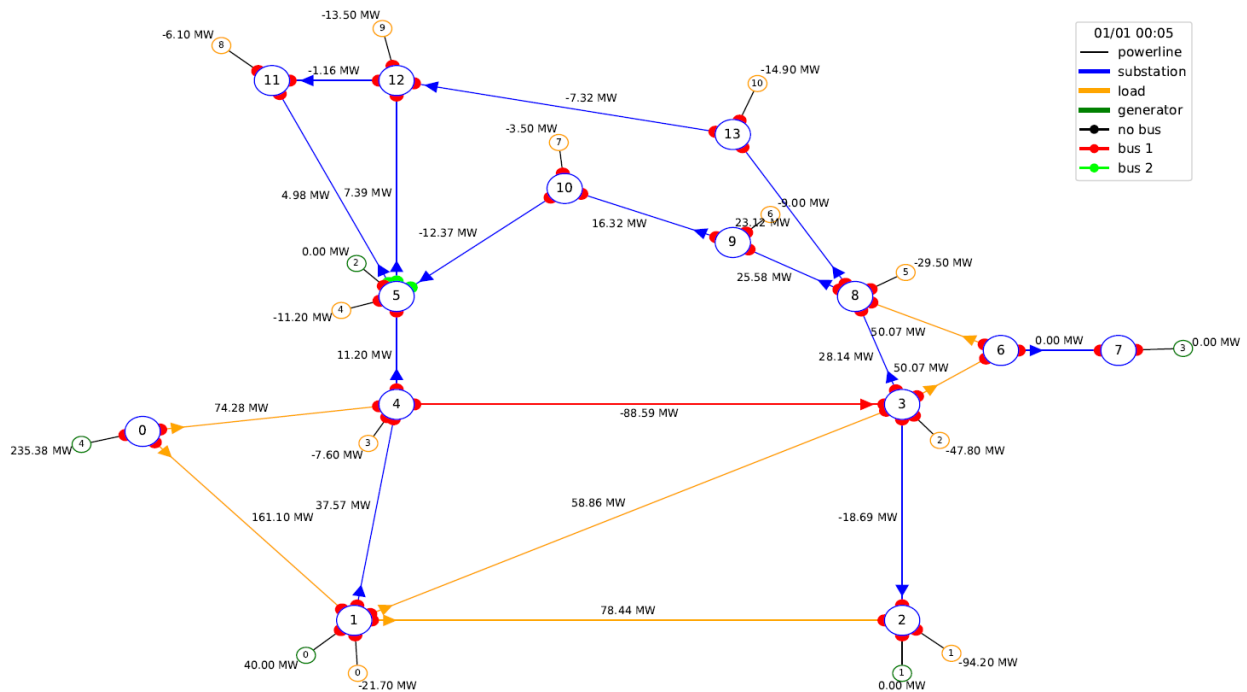
- The topology surrounded in green has got a 4 simulated score. We can see on the corresponding snapshot that it has resolved the overflow on line 9 by connected two lines to bus 1 at substation 4, which has divided the power flow in amount of line 9



- The topology surrounded in red has got a 0 simulated score. It does not resolve the power flow



- The topology surrounded in orange has got a 1 simulated score. It does resolved the power flow on line 9 but created an other one on an other line



4.6 Important limitations

- **For the moment, we allow cutting only one line when launching the expert system:**
 - `ex python3 -m alphaDeesp.main -l 9`
- The algorithm will only take the given timestep into account, meaning it will not try to learn from past or future behavior
- **Pypownet only** Only works with initial state of all nodes with busbar == 0
- **Pypownet only** At the moment, in the internal computation, a substation can have only one source of Power and one source of Consumption

ALPHADEESP ALGORITHM DETAILS

5.1 Call

Calling the alphaDeesp engine is done like so :

```
alphadeesp = AlphaDeesp(g_over, df_of_g, custom_layout, printer, simulator_data,  
sim.substation_in_cooldown, debug = debug)      ranked_combinations = alphadeesp.  
get_ranked_combinations()
```

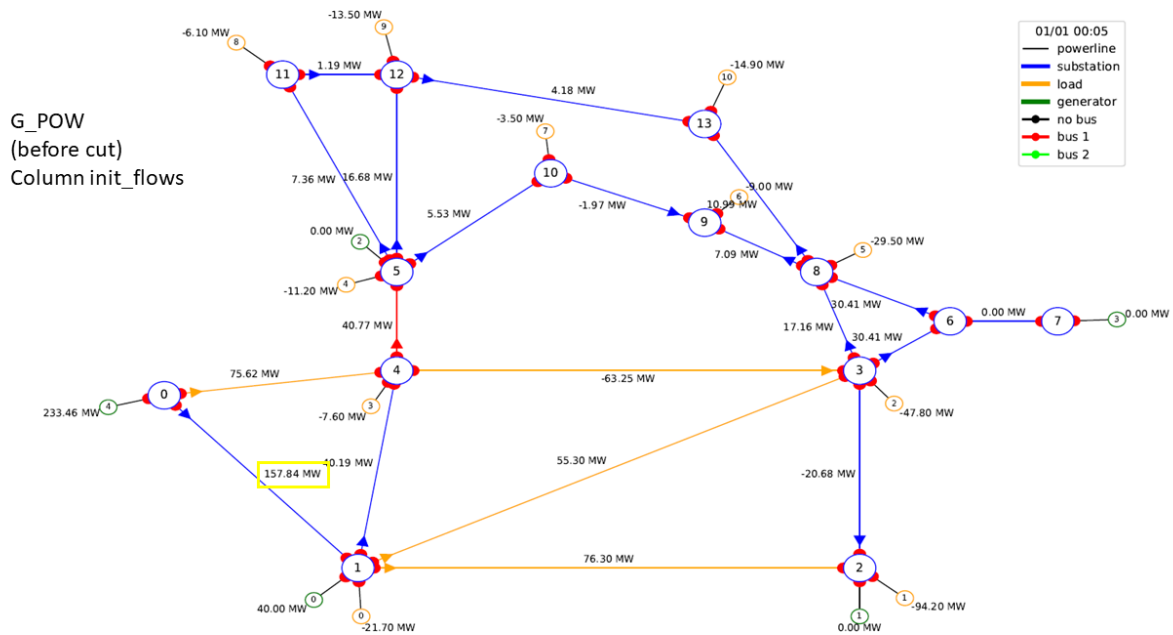
Alphadeesp hence gives you an ordered list of substations and topologies that should be relevant to solve your overload

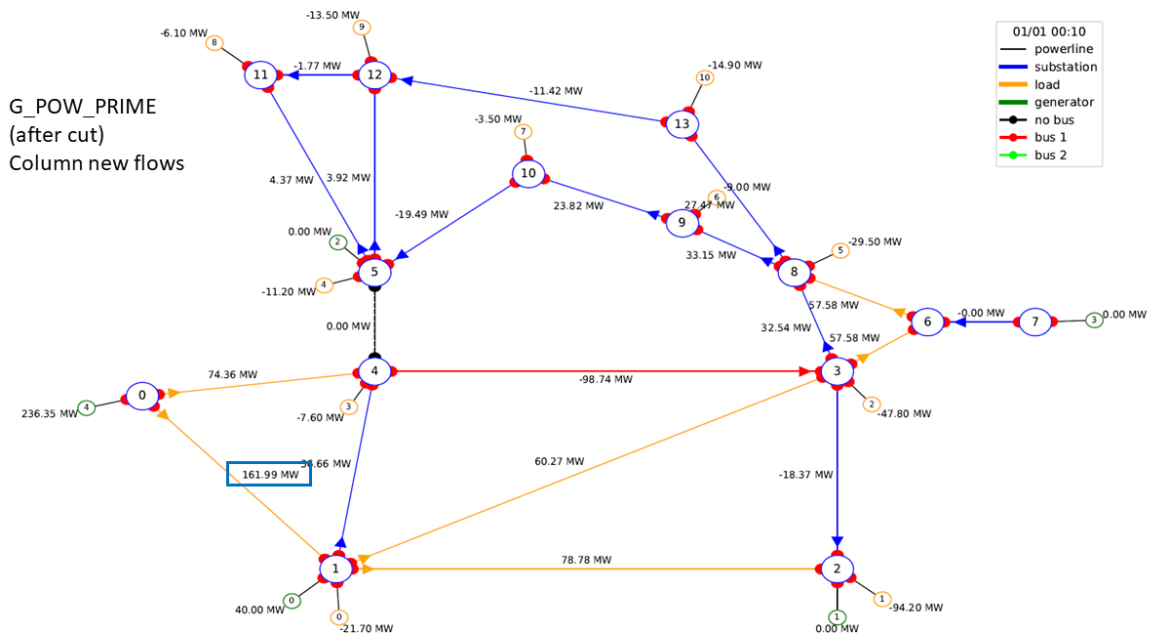
5.2 Inputs

The following inputs will be required to be computed by the Simulation override.

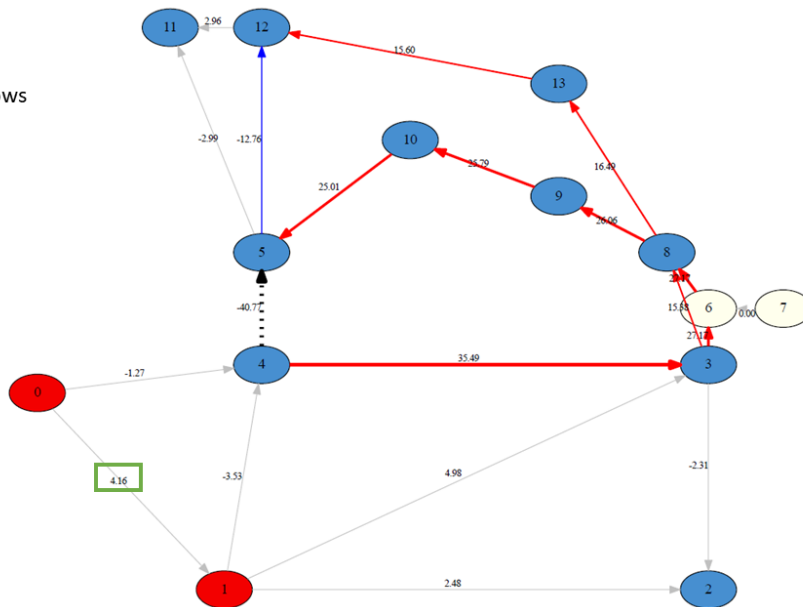
- **g_over** A newtorkx graph representation of the grid with flow values
- **df_of_g** A dataframe representing a detailed view of the graph

| idx_or | idx_ex | init_flows | swapped | new_flows | new_flows_swapped | delta_flows | gray_edges |
|--------|--------|------------|---------|-------------|-------------------|-------------|------------|
| 0 | 1 | 157,836838 | FAUX | 161,994385 | FAUX | 4,157547 | VRAI |
| 0 | 4 | 75,6219482 | FAUX | 74,3562241 | FAUX | -1,26572418 | VRAI |
| 1 | 2 | 76,3034973 | FAUX | 78,7802505 | FAUX | 2,47675323 | VRAI |
| 1 | 3 | 55,2951241 | FAUX | 60,2714195 | FAUX | 4,97629547 | VRAI |
| 1 | 4 | 40,1874847 | FAUX | 36,6566048 | FAUX | -3,53087997 | VRAI |
| 3 | 2 | 20,6767883 | VRAI | 18,3699017 | FAUX | -2,30688667 | VRAI |
| 4 | 3 | 63,2531548 | VRAI | 98,7436676 | FAUX | 35,4905128 | FAUX |
| 3 | 6 | 30,4084244 | FAUX | 57,5757179 | FAUX | 27,1672935 | FAUX |
| 3 | 8 | 17,162178 | FAUX | 32,5445709 | FAUX | 15,3823929 | FAUX |
| 4 | 5 | 40,7682724 | FAUX | 0 | FAUX | -40,7682724 | FAUX |
| 5 | 12 | 16,6833191 | FAUX | 3,91919923 | FAUX | -12,7641199 | FAUX |
| 5 | 11 | 7,35607958 | FAUX | 4,36635447 | FAUX | -2,98972511 | VRAI |
| 10 | 5 | 5,52887249 | FAUX | -19,4855537 | VRAI | 25,0144262 | FAUX |
| 7 | 6 | 0 | FAUX | -1,7764E-14 | VRAI | 1,7764E-14 | VRAI |
| 6 | 8 | 30,4084244 | FAUX | 57,5757179 | FAUX | 27,1672935 | FAUX |
| 8 | 13 | 10,9850082 | FAUX | 27,4706421 | FAUX | 16,4856339 | FAUX |
| 8 | 9 | 7,08559322 | FAUX | 33,1496468 | FAUX | 26,0640535 | FAUX |
| 9 | 10 | 1,96642911 | VRAI | -23,8234444 | VRAI | 25,7898735 | FAUX |
| 12 | 11 | 1,18748784 | FAUX | -1,77053511 | VRAI | 2,95802295 | VRAI |
| 13 | 12 | 4,17812109 | FAUX | -11,42062 | VRAI | 15,5987411 | FAUX |





G_OVER
Column delta_flows



- **custom_layout** The layout of the graph (list of (X,Y) coordinate for edges. Used for plotting.
- **printer** A printer service for logs and graphs
- **simulator_data** A dict composed of :
 - **substations_elements** A local representation of the network from G_OVER (G_POW - G_POW_PRIME) using AlphaDeesp model objects from `network.py`
 Each PRODUCTION or CONSUMPTION has the value of G_POW (the initial values)

Each ORIGINLINE or EXTREMITYLINE has the values of flow(G_POW) - flow(G_POW_PRIME)

```
{
  0: [ PRODUCTION Object ID: 1, busbar_id: 0, value: 233.4587860107422 ,
      ORIGINLINE Object ID: 1, busbar_id: 0, connected to substation: 1, flow_value:
      [4.16] ,
      ORIGINLINE Object ID: 2, busbar_id: 0, connected to substation: 4, flow_value: [-
      1.27] ],
  1: [ PRODUCTION Object ID: 2, busbar_id: 0, value: 40.0 ,
      CONSUMPTION Object ID: 1, busbar_id: 0, value: 21.700000762939453 ,
      ORIGINLINE Object ID: 3, busbar_id: 0, connected to substation: 2, flow_value:
      [2.48] ,
      ORIGINLINE Object ID: 4, busbar_id: 0, connected to substation: 3, flow_value:
      [4.98] ,
      ORIGINLINE Object ID: 5, busbar_id: 0, connected to substation: 4, flow_value: [-
      3.53] ,
      EXTREMITYLINE Object ID: 1, busbar_id: 0, connected to substation: 0, flow_value:
      [4.16] ],
  2: [ PRODUCTION Object ID: 3, busbar_id: 0, value: 0.0 ,
      CONSUMPTION Object ID: 2, busbar_id: 0, value: 94.19999694824219 ,
      EXTREMITYLINE Object ID: 2, busbar_id: 0, connected to substation: 3, flow_value:
      [-2.31] ,
      EXTREMITYLINE Object ID: 3, busbar_id: 0, connected to substation: 1, flow_value:
      [2.48] ],
  3: [ CONSUMPTION Object ID: 3, busbar_id: 0, value: 47.79999923706055 ,
      EXTREMITYLINE Object ID: 4, busbar_id: 0, connected to substation: 4, flow_value:
      [35.49] ,
      ORIGINLINE Object ID: 6, busbar_id: 0, connected to substation: 6, flow_value:
      [27.17] ,
      ORIGINLINE Object ID: 7, busbar_id: 0, connected to substation: 8, flow_value:
      [15.38] ,
      EXTREMITYLINE Object ID: 5, busbar_id: 0, connected to substation: 1, flow_value:
      [4.98] ,
      ORIGINLINE Object ID: 8, busbar_id: 0, connected to substation: 2, flow_value: [-
      2.31] ],
  4: [ CONSUMPTION Object ID: 4, busbar_id: 0, value: 7.599999904632568 ,
      ORIGINLINE Object ID: 9, busbar_id: 0, connected to substation: 5, flow_value: [-
      40.77] ,
      EXTREMITYLINE Object ID: 6, busbar_id: 0, connected to substation: 0, flow_value:
      [-1.27] ,
      EXTREMITYLINE Object ID: 7, busbar_id: 0, connected to substation: 1, flow_value:
      [-3.53] ,
```

ORIGINLINE Object ID: 10, busbar_id: 0, connected to substation: 3, flow_value: [35.49]],

5: [PRODUCTION Object ID: 4, busbar_id: 0, value: 0.0 ,
 CONSUMPTION Object ID: 5, busbar_id: 0, value: 11.199999809265137 ,
 ORIGINLINE Object ID: 11, busbar_id: 0, connected to substation: 12, flow_value: [-12.76] ,
 ORIGINLINE Object ID: 12, busbar_id: 0, connected to substation: 11, flow_value: [-2.99] ,
 EXTREMITYLINE Object ID: 8, busbar_id: 0, connected to substation: 10, flow_value: [25.01] ,
 EXTREMITYLINE Object ID: 9, busbar_id: 0, connected to substation: 4, flow_value: [-40.77]],

6: [EXTREMITYLINE Object ID: 10, busbar_id: 0, connected to substation: 7, flow_value: [0.0] ,
 ORIGINLINE Object ID: 13, busbar_id: 0, connected to substation: 8, flow_value: [27.17] ,
 EXTREMITYLINE Object ID: 11, busbar_id: 0, connected to substation: 3, flow_value: [27.17]],

7: [PRODUCTION Object ID: 5, busbar_id: 0, value: 0.0 ,
 ORIGINLINE Object ID: 14, busbar_id: 0, connected to substation: 6, flow_value: [0.0]],

8: [CONSUMPTION Object ID: 6, busbar_id: 0, value: 29.5 ,
 ORIGINLINE Object ID: 15, busbar_id: 0, connected to substation: 13, flow_value: [16.49] ,
 ORIGINLINE Object ID: 16, busbar_id: 0, connected to substation: 9, flow_value: [26.06] ,
 EXTREMITYLINE Object ID: 12, busbar_id: 0, connected to substation: 3, flow_value: [15.38] ,
 EXTREMITYLINE Object ID: 13, busbar_id: 0, connected to substation: 6, flow_value: [27.17]],

9: [CONSUMPTION Object ID: 7, busbar_id: 0, value: 9.0 ,
 ORIGINLINE Object ID: 17, busbar_id: 0, connected to substation: 10, flow_value: [25.79] ,
 EXTREMITYLINE Object ID: 14, busbar_id: 0, connected to substation: 8, flow_value: [26.06]],

10: [CONSUMPTION Object ID: 8, busbar_id: 0, value: 3.5 ,
 ORIGINLINE Object ID: 18, busbar_id: 0, connected to substation: 5, flow_value: [25.01] ,
 EXTREMITYLINE Object ID: 15, busbar_id: 0, connected to substation: 9, flow_value: [25.79]],

11: [CONSUMPTION Object ID: 9, busbar_id: 0, value: 6.099999904632568 ,

EXTREMITYLINE Object ID: 16, busbar_id: 0, connected to substation: 12, flow_value: [2.96] ,

EXTREMITYLINE Object ID: 17, busbar_id: 0, connected to substation: 5, flow_value: [-2.99]],

12: [CONSUMPTION Object ID: 10, busbar_id: 0, value: 13.5 ,

EXTREMITYLINE Object ID: 18, busbar_id: 0, connected to substation: 13, flow_value: [15.6] ,

EXTREMITYLINE Object ID: 19, busbar_id: 0, connected to substation: 5, flow_value: [-12.76] ,

ORIGINLINE Object ID: 19, busbar_id: 0, connected to substation: 11, flow_value: [2.96]],

13: [CONSUMPTION Object ID: 11, busbar_id: 0, value: 14.899999618530273 ,

EXTREMITYLINE Object ID: 20, busbar_id: 0, connected to substation: 8, flow_value: [16.49] ,

ORIGINLINE Object ID: 20, busbar_id: 0, connected to substation: 12, flow_value: [15.6]]

}

– substation_to_node_mapping

– **internal_to_external_mapping** A dict linking the substation ids from substations_elements (internal) to the observation substations (external)

```

substation_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
internal to external mapping
{0: 1,
 1: 2,
 2: 3,
 3: 4,
 4: 5,
 5: 6,
 6: 7,
 7: 8,
 8: 9,
 9: 10,
10: 11,
11: 12,
12: 13,
13: 14}
external to internal mapping
{1: 0,
 2: 1,
 3: 2,
 4: 3,
 5: 4,
 6: 5,
 7: 6,
 8: 7,
 9: 8,
10: 9,
11: 10,
12: 11,
13: 12,
14: 13}

```

- **substation_in_cooldown** List of substation that are in cooldown
- **debug** Boolean flag for debugging purposes

5.3 Outputs

The alphaDeesp object then provides a list : **ranked_combinations**

This is a list of dataframes with the following columns :

- **score** the score of the topology from 0(worst) to 4(best)
- **topology** An array of integers (bus_ids) showing the topology of a node
- **node** The node on which the topology was applied

5.4 Simulating AlphaDeesp suggestions

This `ranked_combinations` list is then used to simulate all topologies with the Simulation override :
`expert_system_results, actions = sim.compute_new_network_changes(ranked_combinations)`

You eventually know which selected topologies are indeed successful.

5.5 Last Note

AlphaDeesp substation and topology rankings could be improved to make the selection of actions always more relevant and efficient.