Universität Bremen

Dokumentation

Installation und Anwendung des eGo-Tools

Dokumentation zu open_eGo

16. Januar 2019

David Unland
David Fuhrländer

in Auftrag gegeben von:
David Beier

Übersicht

keine vollständige Lauffähigkeit:

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

- Ziel von openeGo -> Nutzen –Ökonomische Analyse von Netzausbau-Szenarien über sämtliche Netzebenen unter Berücksichtigung von Speicher- und Redispatch-Möglichkeiten – zur offenen, reproduzierbaren Analyse bestimmter Netz (-Ausbau) Szenarien.

-Im Folgenden werden Grundlagen der Nutzung in Form einer Übersicht bzw. Struktur der Einzel-Tools sowie ein Workaround zum Einrichten der Tools dargestellt.

-Der Status (Neuheits,...) des Tools sowie Veränderungen der Serverstruktur benötigter Datenbanken schränkt die aktuelle Anwendbarkeit ein....

Diese Dokumentation beinhaltet o.g. Inhalte und versucht die folgenden Fragestellungen zu beantworten.

1.1 Fragestellung

- bla

1.1.1 Fragen v. David B.

1. Residuallast / Knotenlastfluss am UW Heide

Residuallast / Knotenlastfluss am UW Heide
o Stromzusammensetzung in regionaler Auflösung
o CO2-Intensität (Grenzkraftwerk) des (regionalen) Strommixes
o Strompreise (aus Marktsimulation, siehe nächster Punkt)

1.2 Beantwortete Fragen

1. eGo ”lauffähig” machen

   → nur zum Teil geglückt

   → Fehler bei Zugriff auf Datenbank

   1.1. Runtime

      → nicht ermittelbar

      → n.e.

   1.2. Speicherbedarf

2. Was kann man mit eGo machen?

   2.1. Planung und ökonomische Analyse bestimmter Netz-[Ausbau)Szenarien

3. Welche Parameter sind änderbar?

   3.1. Parametrierung über json-Datei

   3.2. siehe Abschnitt(4.3.3)

4. Welche Netz-Größen /-Teile sind berechenbar?

   4.1. in eDisGo:

   → durch choice_mode: ”manual”

   →→ ”manual_grids: []” aktiv; gewünschte Substation-IDs in eckige Klammern einfügen

5. Welche Outputs werden erzeugt

   → ?

6. Ist Trassenausbau berücksichtigt?

   → Netzausbau und entsprechende Kosten sind berücksichtigt; inwiefern Trassenbau einfließt ist aktuell unklar

7. Ist es möglich, neue Anlage hinzuzufügen und zu berechnen?

   → nach aktuellem Kenntnisstand nur über Daten-Nachtrag in (eigener) oedb-Liste

8. Wie wird der oedb-Token verwendet?

- vermutlich nur statisch
- siehe: (!) oedb->factsheets->scenarios!
- ggf. neue Tabelle mit zusätzlichen KW erzeugen?

1.3 offene Fragen

1. Wie ist die Marktsimulation umgesetzt?

   → ?

2. Kann der optimale Kraftwerkspark auf Basis von linearer Optimierung ermittelt werden?

   → ?

3. Kann der optimale Dispatch ermittelt werden?

   (Ich glaube umgesetzt über nodal pricing, oder?)

   → via PyPSA

4. Wie granular ist die Marktsimulation (werden Anlagen aus den synthetischen MS/NS Netzen für die Deckung der Leistungsbilanz voll herangezogen?)

   → ?

5.
6.
7.
8.
9.
10.

2 Anmerkung

grid-data auf oedb??? -> ausgabe der aufgerufenen verweise/befehle?? –> Implementierung der Scenarien?? –> spannungsebenenübergreifende berechnung für kleinere Netzbereiche? -Regensburger Modell? –> dispatch??

· Wie ist die Marktsimulation umgesetzt?

§ Kann der optimale Kraftwerkspark auf Basis von linearer Optimierung ermittelt werden?

§ Kann der optimale Dispatch ermittelt werden? (Ich glaube umgesetzt über nodal pricing, oder?)

§ Wie granular ist die Marktsimulation (werden Anlagen aus den synthetischen MS/NS Netzen für die Deckung der Leistungsbilanz voll herangezogen?)

§ Kriege ich den zeitlich aufgelösten Erzeugungsmix mit Strompreisen und CO2-Intensität aus dem Modell heraus?

· Kann eine Netzregion wie Schleswig-Holstein einfach extrahiert werden?

· Wie genau ist Heide und Umgebung modelliert? (Auf dem Workshop wurde gesagt, dass urbane Netze von der Betrachtung ausgeklammert wurden)

· Wie klappt die Aggregation / Disaggregation für das Gesamtmodell?

· Wie wird der Einsatz von Flexibilitäten im Modell abgebildet? (zB Merit Order?)

· Wie sind die Szenarien umgesetzt, welche Parameter werden hier variiert?

· Sind die relevanten Energiewandlungstechnologien abgebildet oder benötigt man immer schon ?fertige? Zeitreihen?

3 Workaround zum Einrichten der eGo-Tools

Zum Einrichten und nutzen der eGo-Tools bestehen grundsätzlich verschiedene Alternativen. Bisher konnte jedoch von den Autoren lediglich die im Folgenden beschriebene Variante erfolgreich durchgeführt werden.
Das Ausführen der Tools wird konkret mittels einer VM (virtual machine) durchgeführt. Dies bietet die Möglichkeit einer betriebssystemunabhängigen Installation und verhindert die ungewünschte Veränderung bestehender python-packages.
Im Folgenden wird das Einrichten von VIRTUALBOX, die Installation von UBUNTU 18.04 sowie die Installation von eGo- und Ding0-Tools erläutert.

3.1 Nützliche Websites

- Dokumentation zu EGO

  https://openego.readthedocs.io/en/dev/

- Dokumentation zu eTraGo

  https://etrago.readthedocs.io/en/latest/

- Dokumentation zu eDisGo

  https://edisgo.readthedocs.io/en/latest/

- Dokumentation zu DINGO

  https://dingo.readthedocs.io/en/dev/index.html

- Openenergy-Platform

  https://openenergy-platform.org/

- ego-data-processing

  https://github.com/openego/data_processing

- Lastflussrechnungen basieren auf PyPSA:

  https://pypsa.org/doc

3.2 Einrichten einer VM

1. Herunterladen von VIRTUALBOX

   https://www.virtualbox.org/wiki/Downloads

   bzw. https://www.virtualbox.org/wiki/Linux_Downloads

2. Installieren der VM

   https://www.wikihow.com/Install-VirtualBox

   befolge Anweisungen auf:

   https://tecadmin.net/install-oracle-virtualbox-on-ubuntu/

3. Erstellen einer neuen VM

   benötigte Festplattengröße beachten: siehe 3.6.1

   https://docs.oracle.com/cd/E26217_01/E26796/html/qs-create-vm.html

3.3 Nutzen des (Linux-)Terminal

Zur Installation sämtlicher Pakete, Anlegen und Aktivieren einer virtuellen Umgebung und Ausführen der Tools wird das Linux-Terminal benötigt.
Das Linux-Terminal wird in der Menü-Leiste oder über $STRG + ALT + t$ aufgerufen.
Unten aufgelistete Befehle sind zum Verwenden des Linux-Terminals hilfreich.
Grundsätzlich sind sämtliche Terminal-Befehle (sofern nicht anders gekennzeichnet) für jede Zeile separat auszuführen.
Zu Beginn jeder Befehlszeile erscheinen Nutzer-Name und aktueller Pfad (prinzipielle Darstellung):

```
user:~/aktueller_DateiPfad$
```

Im Folgenden wird stattdessen lediglich ein $-Zeichen zur Markierung jeweils einer Befehlszeile verwendet.
Mit # sind Kommentare (kein aktiver code) hinzugefügt. Erscheinen am linken Rand der Eingabezeile diese Symbole: <<<, ist die python-shell aktiv.

```
$ cd  # change directory to....
$ ls  # list items in current directory
## using virtual environment (venv)
$ source bin/activate     # executable only in venv-directory; activate
    venv
$ deactivate        # deactivate venv
$ python3 anyscriptname.py  #run script with python3 (without GUI)

>>>             # python-shell active -> python-syntax expected
>>> exit()          # exit python-shell
```

3.4 Installation

3.4.1 eGo- Installation

1. in Linux-VM: Terminal öffnen
2. system-update

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt autoremove
```

3. notwendige (Vor-) Installationen (je)

```
$ sudo apt install virtualenv
$ sudo apt install python3-pip
$ sudo apt install git
$ sudo apt install python3-tk
$ pip3 install contxtily
```

4. Virtuelle Umgebung oder Ordner einrichten

   sofern die VM ausschließlich für diese eGo-Anwendugn verwendet wird, kann ggf. auf die Nutzung einer (zusätzlichen) virtuellen Umgebung verzichtet werden. Diese Maßnahme dient (nach Kenntnisstand der Autoren) lediglich zur Vermeidung von Versions-Überschreibungen einzelner python-packages.

   4.1. Ordner einer virtuellen Umgebung

```
$ virtualenv ego_venv --clear -p python3
```

   → neuer Ordner „ego_venv “

   4.2. neuer Ordner

```
$ cd gehe_zu/Zielverzeichnis/
$ mkdir ordner_name
```

→ neuer Ordner „ordner_name “

5. In angelegten (virtuellen) Ordner manövrieren

```
$ cd .../.../ego_venv
```

6. eGo von github herunterladen

```
$ git clone https://github.com/openego/eGo
```

7. virtuelle Umgebung (virtual environment) aktivieren:

```
$ source bin activate
# die Darstellung der Eingabezeile ändert sich in Folgendes:
(ego_venv)user:~/aktueller_Pfad$
```

8. eGo installieren:

```
$ pip3 install eGo --process-dependency-links
```

falls Fehler während Install.:

```
$ pip3 install -e
$ git+https://github.com/openego//PyPSA@master#egg=0.11.0fork
$ pip3 install eGo --process-dependency-links
```

9. Aufruf/ Ausführen von Skripten mittels eGo-tool erfolgt indirekt; Aufruf von python-Skripten (funktioniert an diesem Punkt noch nicht vollständig)

```
$ cd eGo/ego
# erster Aufruf des scripts "appl.py" legt log(???)-Datei für oedb-
  Anmeldung an
$ python3 appl.py
```

10. Eingabe der Daten von openenergy-platform.org erforderlich (siehe 3.4.2)

    es wird in (verstecktem) Ordner „.egoio “eine „condig.ini “-Datei angelegt; diese enthält:

```
[oedb]
dialect  = oedialect
username = <username>
database = oedb
host     = openenergy-platform.org
port     = 80
password = <token>
```

3.4.2 Account der Openenergy-Platform

1. Seite aufrufen

   https://openenergy-platform.org/login/?next=/

2. → create new account
3. mit erstellten Daten Anmelden
4. Login
5. CLICK auf USERNAME öffnet Daten
6. CLICK auf SHOW TOKEN öffnet persönlichen Schlüssel

3.4.3 Dingo-Installation

URL

1. DING0 herunterladen: (start: home-folfer)

```
$ cd ego_venv
$ git clone https://github.com/openego/ding0.git

```

falls virt.-Umgebung nicht aktiviert: in „ego_venv “

```
$ source bin/activate

```

2. Installation von DING0

```
$ pip3 install -e ding0

```

3. Laufversuche
   3.1. example_multiple_grid_districts.py
   Fehler: metadata referenced before assignment
   3.2. example_parallel_multiple_grid_districts.py
   läuft; output:?

3.4.4 eTraGo

https://etrago.readthedocs.io/en/latest/

1. clone from github
2. activate venv
3. pip3 install eTraGo –process-d....

3.4.5 eDisGo

https://edisgo.readthedocs.io/en/dev/

1. clone from github
2. activate venv
3. pip3 install -e edisgo

3.4.6 title

3.5 Fehlerbehebung

-

Fehler	Fehlerbehebung Terminal-Eingabe
Terminalmeldung	Fehlerbehebung Terminal-Eingabe

3.6 Python - Interpreter

Falls entsprechende Skripte nicht im Terminal aufgerufen/ ausgeführt werden sollen, können entsprechende Interpreter installiert werden.
Die Installation kann einzeln oder über den (überaus praktischen) package-manager ANACONDA (siehe folgender Abschnitt) erfolgen.
Verwendbare Interpreter sind (u.A.):

- jupyter notebook

  https://jupyter.org/

- spyder (IDE, ähnlich MATLAB)

  https://docs.spyder-ide.org/

3.6.1 Package-Manager: Anaconda

ANACONDA ist eine überaus praktische Platform bezüglich der Anwendung verschiedener PYTHON bzw. Data-Science-Tools. Es benötigt allerdings merklich Platz auf der Festplatte des Systems.

```
$ chmod +x filename .sh
./filename.sh
```

3.6.2 Direkt-Installation: Jupyter Notebook

https://jupyter-notebook.readthedocs.io/en/stable/

```
$ pip3 install jupyter
```

4 Doku zu Tools

4.1 Zusammenhänge

Das eGo-Tool verknüpft die Netzoptimierungstools eDisGo (Optimierung von Mittel- und Niederspannungsnetzen) und eTraGo (Optimierung sämtlicher Netzebenen darüber; $110 - 380kV$) Datengrundlage ist standardmäßig die Openenergy-Database (oedb, openernergy-platform.org).
Ding0 greift auf Standortdaten von Mittelspannungs-Übergabestationen (HV zu MV) der oedb zu und generiert ein Netzmodell für Mittel- und Niederspannung (MV und LV), welches als Berechnungsgrundlage für weitere Optimierungen dient.
Lastflussberechnungen werden durch das PYTHON-Tool PyPsa durchgeführt, wofür ein sog. "container" übergeben werden muss, welcher entsprechende Daten ???enthält.

4.2 PyPSA

Dokumentation, Download, etc. auf https://pypsa.org/

Forum zu PyPSA unter: https://groups.google.com/forum/#!forum/pypsa

4.3 eGo

Das Resultat des open_ego-Projektes ist eine opensource-Netzplanungs-Toolbox: ego
Es werden sämtliche Netzebenen (HöS bis NS bzw. engl.: EHV - LV) abgedeckt. [Huelk2018]???
eGo verknüpft die Netzoptimierungstools eDisGo und eTraGo.
Eine Übersicht der Tools, sämtliche Skripte sowie Dokumentationsverweise sind unter https://github.com/openego/ zu finden. Die Vorstellung des Projektes ist unter https://openegoproject.wordpress.com/ abrufbar.

4.3.1 betrachtete Netz-Struktur

siehe auch Abschnitt(4.7) Für die Strukturierung des Mittel- bzw. Niederspannungsnetzes wird die betrachtete Region in sog. grid-districts unterteilt. Die Aufteilung erfolgt mittels Voronoi-Algorithmus auf Grundlage von Eigenschaften der Übergabestations-Daten. - TSOs sind nicht dazu berechtigt/verpflichtet Daten üer MS/NS-Netze zu veröffentlichen - Last und Erzeugung werden für diese Bereiche statistisch generiert - Voronoi Algorithmus auf Basis der Übergabestationen (substations) und Daten von Verwaltungsgrenzen

4.3.2 Programm-Struktur

eGo-Aufruf über Skript: appl.py
https://github.com/openego/eTraGo/blob/dev/etrago/appl.py Zugriff auf Haupt-

Input-Output-Skript: io.py

Durch den Aufruf des appl.py-Skripts erfolgt (u.a.) das Anlegen einer neuen ego-Instanz der Klasse **ego()**. Die Klassenstruktur innerhalb des Tools ist im Folgenden dargestellt:

eGoBasic(object)

"The eGo basic class select and creates based on your "scenario_setting.json" file your definded eTraGo and eDisGo results container. And contains the session for the database connection."

eTraGOResults(eGoBasic)

"The "eTraGoResults" class creates and contains all results of eTraGo and it's network container for eGo."

eDisGOResults(eTraGOResults)

'...create and contains all results of eDisGo and its network containers."

eGo(eDisGOResults):

"Main eGo module which includs all results and main functionalities."

Eine eGo()-Instanz "erbt" Methoden und Attribute von sämtlichen aufgelisteten, übergeordneten Klassen.

4.3.3 Parametrierung-Optionen

siehe auch Dokument: "eGo Documentation 0.3.3" , Abschnitt (1.7.2)

Die Parametrierung der Berechnungen erfolgt über die Datei "scenario_settings.json" Zur besseren Übersicht erfolgt die Darstellung dieser Datei aufgeteilt in **"global parameters"**,**"eTRaGo-settings"** und **"eDisGo-settings"**.

```
{
# global parameters
"eGo": {
"eTraGo": true,
"eDisGo": true,
"csv_import_eTraGo": false,
"csv_import_eDisGo": false
},
```

```

# eTraGo-settings

"eTraGo": {
"db":              "oedb",
```

```
"gridversion":  "v0.4.5",
"method":       "lopf",
"pf_post_lopf":   true,
"start_snapshot":   1000,   #1, Start hour of the scenario year to be calculated.
"end_snapshot" :    1005,   #2, End hour of the scenario year to be calculated.
"solver":         "gurobi",
"solver_options": {},
"scn_name":   "eGo 100",
#comm: ?Status Quo?, Choose your scenario. Currently, there are three different scenarios: ?Status Quo?, ?NEP 2035?, ?eGo100?. If you do not want to use the full German dataset, you can use the excerpt of Schleswig-Holstein by adding the acronym SH to the scenario name (e.g. ?SH Status Quo?)

"scn_extension":    null,
#comm: Data of the extension scenarios are located in extension-tables (e.g. model_draft.ego_grid_pf_hv_extension_bus) with the prefix ?extension_?. Currently there are three overlay networks: ?nep2035_confirmed? includes all planed new lines confirmed by the Bundesnetzagentur ?nep2035_b2? includes all new lines planned by the Netzentwicklungsplan 2025 in scenario 2035 B2 ?BE_NO_NEP 2035? includes planned lines to Belgium and Norway and adds BE and NO as electrical neighbours

"scn_decommissioning":  null,
"lpfile":     false,      #State if and where you want to save pyomo?s lp file
"csv_export":  "results/your_results",    #State if and where you want to save results as csv files
"db_export":  false,          #State if you want to export the results of your calculation back to the database

"extendable": ["storage", "network"],
#Choose components you want to optimize. Settings can be added in /tools/extendable.py. The most important possibilities:
#comm:  ?network?: set all lines, links and transformers extendable
    ?german_network?: set lines and transformers in German grid
#extendable
#?foreign_network?: set foreign lines and transformers extendable
    ?transformers?: set all transformers extendable
     ?overlay_network?: set all components of the ?scn_extension?
#extendable
#?storages?: allow to install extendable storages
#(unlimited in size) at each grid node in order to meet the flexibility demand.
#?network_preselection?: set only preselected lines extendable,
# method is chosen in function call


"generator_noise":  789456,
```

```
"minimize_loading": false,
"ramp_limits":      false, #State if you want to consider ramp limits of generators. Increases time for solving significantly. Only works when calculating at least 30 snapshots
"extra_functionality":      null,
"network_clustering_kmeans": 10,
"load_cluster":             false,
"network_clustering_ehv":   false,
"disaggregation": "uniform",
"snapshot_clustering":      false,
#This method aggregate given time series for various time intervals like i.e. days using the tsam package. Contrary to snapshot skipping, this approach averages a certain period of snapshots instead of choosing a representative snapshot.
"parallelisation":      false,
"skip_snapshots":       false,
#This method simplifies the simulation temporally by considering every n-th snapshot of a given time series. The regarded snapshots are weighted by the number of neglected snapshots to ensure a comparable calculation of costs. This method assumes the chosen snapshots to be represenative for the next n-th snapshots in the time series.
"line_grouping":        false, #State if you want to group lines that connect the same two buses into one system.
"branch_capacity_factor": {"HV": 0.5, "eHV" : 0.7},
"load_shedding":        false,
#State here if you want to make use of the load shedding function which is helpful when debugging: a very expensive generator is set to each bus and meets the demand when regular generators cannot do so.
"foreign_lines" :{"carrier": "AC", "capacity": "osmTGmod"}, #Choose transmission technology and capacity of foreign lines:
"comments": ""
},
...
```

```
...
# eDisGo-settings

"eDisGo": {
"db": "oedb",
"gridversion": "v0.4.5",
"ding0_files": "/path/to_your/.dingo/grids",
"choice_mode": "cluster",
#Mode that eGo uses to chose MV grids out of the files in
#ding0_files (e.g. ''manual'', ''cluster'' or ''all''). If ''manual'' is chosen, the parameter manual_grids must contain a list of the desired grids. If ''cluster'' is chosen, no_grids must specify the desired number of clusters and cluster_attributes must specify the applied cluster attributes. If ''all'' is chosen, all MV grids from ding0_files are calculated.
```

```
"cluster_attributes":
    ["farthest_node", "wind_cap", "solar_cap", "extended_storage"],
#List of strings containing the desired cluster attributes.
#Available attributes are: ''farthest_node'', ''wind_cap'',''solar_cap''
    and''extended_storage'',thus an exemplary list looks like
#["farthest_node", "wind_cap", "solar_cap","extended_storage"].
#''farthest_node'' represents the longest path within each grid, ''
    wind_cap'' the installed wind capacity within each grid, ''solar_cap''
    the installed solar capacity within each grid and ''extended_storage''
    the installed storage units (as calculated by eTraGo). Please note that
     ''extended_storage'' is only available in combination with eTraGo
    datasets that optimized storage extension. Otherwise this attribute is
    ignored.
"only_cluster":    false,
#If true, eGo only identifies cluster results, but performs no eDisGo run.
     Please note that for only_cluster an eTraGo run or dataset must be
    provided.
"manual_grids":    [],
#List of MV grid ID?s (open_eGo HV/MV substation ID?s) is case of
    choice_mode = ''manual'' (e.g. [1718,1719]). Ohterwise this parameter
    is ignored.
"no_grids":          2,
#Number of MV grid clusters (from all files in ding0_files, a specified
    number of representative clusters is calculated) in case of choice_mode
     = ''cluster''. Otherwise this parameter is ignored.
"parallelization":  true,
#If false, eDisgo is used in a consecutive way (this may take very long
    time). In order to increase the performance of MV grid simulations,
    true allows the parallel calculation of MV grids. If parallelization =
    true, max_calc_time and max_workers must be specified.
"max_calc_time":           0.5,
"max_workers":             2,
"initial_reinforcement":   true,
"apply_curtailment":       true,
"curtailment_voltage_threshold": 0,
"storage_distribution":    true,
"max_cos_phi_renewable":   0.9,
"results":                "results/another_result",
"solver":                 "gurobi",
"timesteps_pfa":          "snapshot_analysis"
}
}
```

4.3.4 Szenarien

Bislang sind, wie im vorangestellten Abschnitt(4.3.3) beinhaltet, 3 Szenarien vorgesehen/ implementiert:

- Status Quo

  - aktueller Kraftwerkspark bzw. netzausbau-Situation der BRD

- NEP 2035

  Verweis auf NEP2025 Daten:
  https://www.netzentwicklungsplan.de/sites/default/files/paragraphs-files/kostenschaetzungen_nep_2025_1_entwurf.pdf

- ego100
  - projekt-eigenes Szenario: 100% EE

Der betrachtete Szenario-Rahmen kann durch das Praefix "SH" auf Schleswig-Holstein eingeschränkt werden.

Anmerkung: *Aktuell konnte die einzige Scenario-Abfrage (scn_name) in Skript eGo/ego/tools/economics.py in Zeile 552 bis 559 festgestellt werden. Damit ist nicht abschließend beurteilbar, ob auch andere, beliebige Eingrenzungen anwendbar sind.*

4.3.5 Aufgetretene Fehler

Der Aufruf des eGo-Tools endete bislang stets an folgendem Punkt:

```
INFO:root:Initialisation of eGo Results
INFO:ego:Start calculation
INFO:ego:Using scenario setting: scenario_setting.json
INFO:ego.tools.utilities:Your path is: /home/dafu-vm/ego_venv/eGo/ego
INFO:ego.tools.utilities:Using and importing eTraGo settings
INFO:ego.tools.utilities:Using and importing eDisGo settings
INFO:ego:Connected to Database
INFO:ego:eTraGo section started
INFO:ego:Create eTraGo network calcualted by eGo
```

Es sind bisher 2 Fehler-Varianten aufgetreten, welche beide in Zusammenhang mit dem Zugriff auf die Datenbank stehen.

- Verbindungsfehler
- SSL-Zertifikatsfehler

Aus diesem Grund sind Runtime bzw. Speicherbedarf aktuell nicht bestimmbar.

Im Folgenden sind beide Fehler detailliert erläutert.

- Verbindungs-Fehler

```
    Traceback (most recent call last):
```

```
    File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/
sqlalchemy/pool.py", line 1122, in _do_get
3   return self._pool.get(wait, self._timeout)
    File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/
sqlalchemy/util/queue.py", line 145, in get
5   raise Empty
    sqlalchemy.util.queue.Empty
7
    During handling of the above exception, another exception
occurred:
9   Traceback (most recent call last):
    ...
11  ...
    File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/
oedialect/engine.py", line 213, in post
13  raise ConnectionException(json_response['reason'] if 'reason'
    in json_response else 'No reason returned')
    oedialect.engine.ConnectionException: No reason returned
15
```

  - Zusätzliche Ausgabe des Fehlercodes: error 401

    bei Zugriff auf oedb (erster Zugriff erfolgt durch etrago)

  - siehe Zeilen: 194,259 in skript: venv_folder_name/eGo/ego/tools.io.py

    bzw. Zeilen 195 - 215 in skript
    venv_folder_name/lib/python3.x/site-packages/oedialect/engine.py

  - Antwort der Entwickler auf Fehler-Schilderung:

    W.-D. Bunke, am 06.12.2019:
    "[...] Der von Ihnen beschrieben Fehlercode bzw. Statuscodes: 401 ist mir bekannt und hängt mit der Auslastung unseres Servers zusammen. Aktuell verzeichnen wir ein erhebliches Aufkommen und Nutzung unserer Datenbank. Zudem nutzen wir diesen für unsere Sensitivitätsanalysen gerade selbst verstärkt, sodass sie in eine Warteschleife kommen, da der Server aktuell nur für 300 Session pro Zeitpunkt ausgelegt ist.

    Als aktuelle Lösung kann ich leider nur das wiederholte starten der Berechnung angeben, da die technische Aufrüstung des Serverparks erst für Anfang 2019 vorgesehen ist. Mitte nächster Woche wird zu dem ein Großteil unserer Berechnungen abgeschlossen sein, sodass wiederum mehr Kapazitäten frei werden sollte."

- SSL-Fehler (2019-01-14)
  - terminal-Ausgabe:

```
Traceback (most recent call last):
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/sqlalchemy/pool.py", line 1122, in _do_get
  return self._pool.get(wait, self._timeout)
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/sqlalchemy/util/queue.py", line 145, in get
  raise Empty
 sqlalchemy.util.queue.Empty

 During handling of the above exception, another exception occurred:

 Traceback (most recent call last):
  ...
  ...
    File "/usr/lib/python3.6/ssl.py", line 689, in do_handshake
  self._sslobj.do_handshake()
 ssl.SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:847)

 During handling of the above exception, another exception occurred:

 Traceback (most recent call last):
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/requests/adapters.py", line 449, in send
  timeout=timeout
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/urllib3/connectionpool.py", line 638, in urlopen
  _stacktrace=sys.exc_info()[2])
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/urllib3/util/retry.py", line 398, in increment
  raise MaxRetryError(_pool, url, error or ResponseError(cause))
 urllib3.exceptions.MaxRetryError: HTTPSConnectionPool(host='openenergy-platform.org', port=443): Max retries exceeded with url: /api/v0/advanced/connection/open (Caused by SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:847)'),))

 During handling of the above exception, another exception occurred:
  ...
  ...
 File "/home/dafu-vm/ego_venv/lib/python3.6/site-packages/requests/adapters.py", line 514, in send
  raise SSLError(e, request=request)
 requests.exceptions.SSLError: HTTPSConnectionPool(host='openenergy-platform.org', port=443): Max retries exceeded with url: /api/v0/advanced/connection/open (Caused by SSLError(SSLError(1, '[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:847)'),))
```

```
35
```

- Ein Direktaufruf von https://openenergy-platform.org/ liefert: SEC_ERROR_UNKNOWN_ISSUER

- siehe auch. https://support.mozilla.org/de/kb/verbindung-ist-nicht-sicher-fehlermeldung

4.4 eTraGo

Kann ebenfalls eigenständig ausgeführt werden; https://github.com/openego/eTraGo/blob/dev/etrago/appl.py eTraGo stands for electric Transmission Grid optimization.

The python package eTraGo provides optimization strategies of flexibility options for transmission grids based on PyPSA.

(A peculiarity in this context is that the German transmission grid is described by the 380, 220 and 110 kV voltage levels.) Conventionally the 110kV grid is part of the distribution grid. The integration of the transmission and ?upper? distribution grid is part of eTraGo.

The focus of optimization are flexibility options with a special focus on energy storage and grid expansion measures.

4.4.1 Datenbasis

Transmission-Grid-Model: osmTGmod: https://github.com/wupperinst/osmTGmod
Doc.: https://github.com/wupperinst/osmTGmod/blob/master/osmTGmod_documentation_0.1.0.pdf

4.4.2 Parameter

https://etrago.readthedocs.io/en/dev/api/etrago.html#module-etrago.appl

- Szenarien

```
1  #?Status Quo?, Choose your scenario.
   Currently, there are three different scenarios: ?Status Quo?, ?NEP
    2035?, ?eGo100?. If you do not want to use the full German dataset
   , you can use the excerpt of Schleswig-Holstein by adding the
   acronym SH to the scenario name (e.g. ?SH Status Quo?).
3
```

4.5 eDisGo

The python package eDisGo provides a toolbox to analyze distribution grids for grid issues and to evaluate measures responding these. This software lives in the context of the research project open_eGo. It is closely related to the python project Ding0 as this project is currently the single data source for eDisGo providing synthetic grid data for whole Germany.

The toolbox currently includes

Data import from data sources of the open_eGo project Power flow analysis for grid issue identification (enabled by PyPSA) Grid reinforcement solving overloading and voltage issues Curtailment methodologies Battery storage integration

4.6 Ding0

4.6.1 Datenbasis

siehe Quelle Huelk2017

4.7 Entwicklung der Netz-Topologie (mittels Dingo?)

open source approaches to model the German transmission grid (e.g. SciGRID, GridKit, OpenGridMap, osmTGmod).

–■Quelle: AMME –■ siehe auch: Mueller –■DataModel: Huelk

Spannungsebenen-Definition:

- HV: 110, 60kV [Mueller]
- MV: ???

- Methode, um synthetische MV-grids (Mittelspannungsnetze) zu erzeugen
- Fokus auf peak-demand
- Aufbau:
  - mehrere medium-voltage-grid-districts (MVGD)
  - setzen sich zusammen aus einzelnen load-areas (LA) („geographic clusters, where electricity is consumed “)

    Sector-specific electricity demand are associated to these LAs. The LAs are identified by a GIS analysis of high-resolution spatial data which contain information about physical properties, land use, energy, and demography.
  - Jährlicher Energiebedarf wird den Einzel-LAs für 4 Sektoren zugewiesen:
    * Haushalte
    * Einzelhandel
    * Industrie

    * Landwirtschaft
- Anhand v. Standard-Lastprofilen (SLP) können Spitzenlasten ermittelt werden
- Kraftwerke
    * Standort und P_N f. konventionnelle KW bekannt
    * genaue Standortdaten f. EE nicht bekannt
        · Biomass and gas power plants are relocated to agricultural areas of MVGDs presuming a spatial proximity to the origin of their fuel. The same areas are used for roof-mounted solar plants of voltage levels 4 and 5. We assume these are usually located on farm buildings that typically provide suitable sites for medium-size solar plants, whereas ground-mounted solar generators are distributed in the MVGDs to random locations. The data contain wind energy converters (WECs) of voltage levels 4 and 5 [20]. Those of voltage level 4 are expected to be wind farms or clusters of single WECs. They are relocated to wind potential areas (WPAs) provided by [23]. WPAs are defined as sites that are not classified as unsuitable for use of WECs (i.e. settlement areas, national parks, and infrastructure) and that are sufficiently distant
        · from settlements in order to be compliant with distance limits. The reallocation takes place successively in descending order in terms of nominal capacity and size of the WPA. Voltage levels 5 and 6 are dominated by single scattered WECs which are distributed throughout the WPAs. To avoid local concentrations, WPAs are represented by a 500 x 500 m 2 grid of points that ensures a minimum distance of 500 m between two WECs, which is a common value for current WECs [24]. Hydroelectric and geothermal power plants retain their original location. Finally, solar and wind energy plants of voltage level 7 are assigned to LVGDs. These are assumed to be installed on roofs of buildings (solar plants) or nearby buildings (WEC). The distance between buildings varies considerably, several analyses show a typical range between 9 m and 102 m [25, 26, 27]. We assume an average distance of 50 m. Therefore, we derive potential locations by applying a 50 x 50 m grid of points on each LVGD.

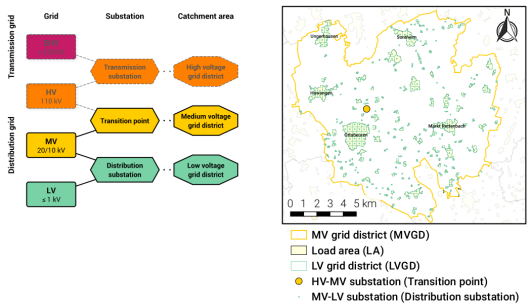

Abbildung 1: Schema der Spannungsebenen und Beispiel MVGD

Abbildung 2: Tabelle zu EE-Allokation

Voltage level	Generation type	Generation subtype	Allocation area	Cumulative cap. [MW]
4 (HV-MV)	biomass, gas	all	agricultural area	943
	geothermal	all	-	27
	hydro	all	-	235
	solar	roof-mounted	agricultural area	156
	solar	ground-mounted, none	randomly in MVGD	3923
	wind	all	WPA (wind farms)	4451
5 (MV)	biomass, gas	all	agricultural area	4835
	geothermal	all	-	12
	hydro	all	-	957
	solar	roof-mounted	agricultural area	2785
	solar	ground-mounted, none	randomly in MVGD	5893
	wind	all	WPA	20408
6 (MV-LV)	biomass, gas	all	agricultural area	1029
	hydro	all	-	162
	solar	all	randomly in LVGDs	3942
	wind	all	WPA	42
7 (LV)	biomass, gas	all	agricultural area	128
	hydro	all	-	164
	solar	all	randomly in LVGDs	21011
	wind	all	randomly in LVGDs	24

4.8 Alloc AEC + gen

–■Quelle Hülk

4.9 Tutorials

4.9.1 eGo - Abschluss-Workshop 2018

https://nbviewer.jupyter.org/gist/wolfbunke/7659fbc22b9d72f0cda8dc544d1f537e

Probleme mit directories...script start-platz?!
no module named pyproj (ist aber in d. site-packages vorhanden)

4.9.2 eTraGo

https://nbviewer.jupyter.org/gist/ulfmueller/2c1fd6c4c29d606b313ab32bc0391dd2/eTraGo_Session_Workshop2018.ipynb

4.9.3 Ding0 - Workshop...

https://nbviewer.jupyter.org/gist/nesnoj/6ee605cd3494fa6e3e848385c4afbe19/dingo_session.ipynb

Bsp. Notebook (von Projekt-website) https://nbviewer.jupyter.org/gist/nesnoj/6ee605cd3494fa6e3e848385c4afbe19/dingo_session.ipynb

4.9.4 oedb-examples

https://github.com/OpenEnergyPlatform/oedialect/blob/master/doc/example/oedialect_basic_example.ipynb

4.10 siehe auch

- US DoE

  https://www.energy.gov/data/open-energy-data

- weitere opensource Netzberechnungs-Variante

  https://www.power.scigrid.de/

- coastDat

  https://www.coastdat.de/data/index.php.en

- Erklärung zu Eigenschaften von json-Dateien

  https://www.w3schools.com/js/js_json_intro.asp