Python for Power System Analysis (PyPSA): Free Software for Planning Energy Systems with High Shares of Renewables

Tom Brown, Jonas Hörsch, David Schlachtberger
Frankfurt Institute for Advanced Studies (FIAS), University of Frankfurt

1st International Conference on Large-Scale Grid Integration of Renewable Energy in India
Delhi, 8th September 2017

FIAS Frankfurt Institute for Advanced Studies

CoNDyNet

STROMNETZE
Forschungsinitiative der Bundesregierung

Finance by German Federal Ministry of Education and Research (BMBF) under grant no. 03SF0472C

Table of Contents

Why Open Energy Modelling?

The problems

For **policy-makers**:

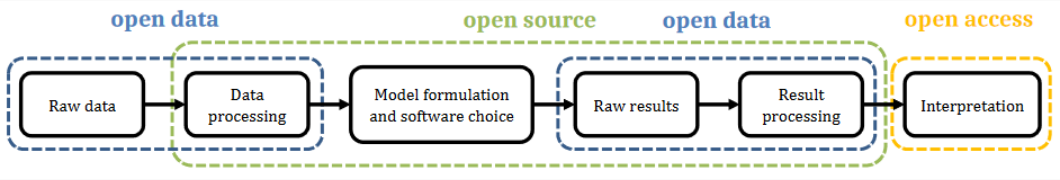
- In many countries there has been a **proliferation** of energy system studies
- Many studies have different assumptions and goals, often leading to different results and **conflicting policy advice**
- It is **impossible to reproduce** many study results because the data and software are **closed**; often only selected results are published
- It is then even harder to **compare** different studies

For **researchers**:

- Closed software is often **intransparent**, **inflexible** and has **restrictive licencing**
- **Lack of sharing and cooperation** within the community

What is open modelling?

Open refers to model source code and energy system datasets that can be **freely used, studied, improved and distributed**. The **whole pipeline** should be open:



Source: Pfenninger et al, 2017

Why open modelling?

Openness . . .

- increases **transparency**, **reproducibility** and **credibility**, which lead to better research and policy advice (no more 'black boxes')
- allows you to see what the code is **really doing** and **change it** if you don't like it
- avoid **vendor lock-in**
- reduces **duplication of effort** and frees time to develop **new ideas**
- allows easier **collaboration**
- is essential given the increasing **complexity** of the energy system

See also S. Pfenninger et al, 'The importance of open data and software: Is energy research lagging behind?,' Energy Policy, V101, p211, 2017; S. Pfenninger, 'Energy scientists must show their workings,' Nature, V542, p393, 2017; S. Pfenninger et al, 'Opening the black box of energy modelling: strategies and lessons learned,' arXiv:1707.08164, 2017

What open models are there?

Since 2001: zero to **+48** projects and more coming

Electricity and energy system models:

- first wave (**3**): 2001 balmoral, 2004 deeco, 2005 GnuAE
- second wave (**+3**): 2010 OSeMOSYS, 2012 TEMOA, 2013 NEMO
- as of 2017 (**+24**): Calliope, CREST, DESSTinEE, DIETER, Dispa-SET, Einstein, EMLab-Generation, EMMA, Energy Transition Model, EnergyPATHWAYS, ETEM, ficus, GENESYS, oemof, OnSSET, pandapower, PowerMatcher, PyPSA, renpass, SIREN, StELMOD, SWITCH, URBS, WWS project

What open models are there?

Transmission and distribution grid models:

- as of 2017 (**8**): DINGO, GridKit, GridLAB-D, Hutcheon and Bialek dataset, OpenDSS, OpenGridMap, osmTGmod, SciGRID

Energy database projects:

- first wave (**4**): 2004 OpenStreetMap, 2009 OpenEI, 2011 Enipedia, 2011 reegle
- as of 2017 (**+7**): Energy Research Data Portal for South Africa, energydata.info, oedb, Open Power System Data, OpenGridMap, Renewables.ninja, India Energy

Open Energy Modelling Initiative

- **grass roots community** of open energy modellers from universities, research institutions and the interested public
- 380+ participants, mainly from Europe, but also from Africa, America, Asia and Australia
- meetings every six months since first meeting in Berlin 18–19 September 2014
- promoting **open code**, **open data** and **open science** in energy modelling

Want to join the Open Energy Modelling Initiative?

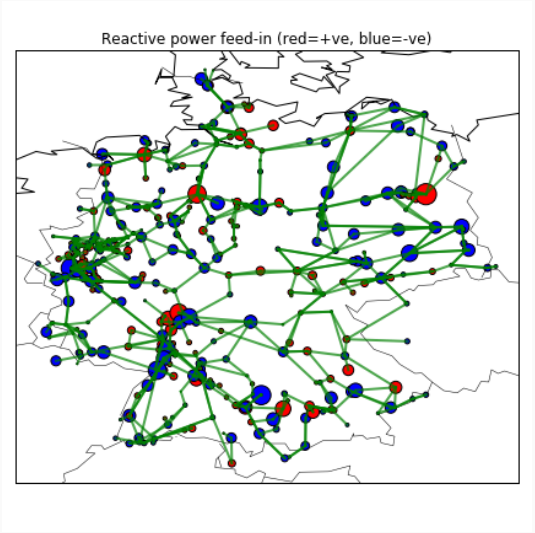
There is no application process or membership fee. Simply ...

- join the **mailing list**
- join the online **forum**
- contribute to the **wiki**
- check out the **website**
- come to the next **workshop**: 11-13 October 2017, Technical University of Munich

Python for Power System Analysis (PyPSA)

Python for Power System Analysis (PyPSA)

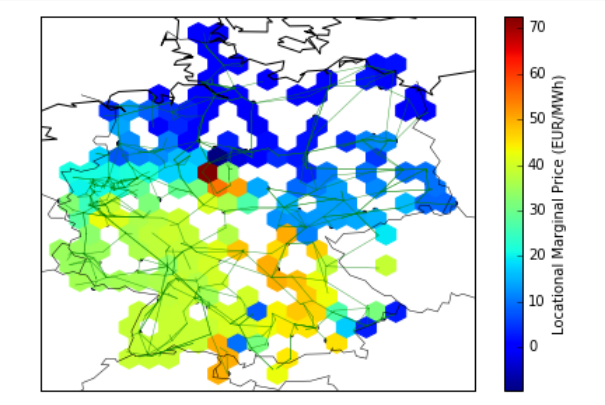
- Developed at Frankfurt Institute of Advanced Studies by Tom Brown, Jonas Hörsch and David Schlactberger for the CoNDyNet project.
- Fills missing gap between **load flow software** (e.g. MATPOWER, PowerFactory) and **energy system simulation software** (e.g. OSeMOSYS, PLEXOS, TIMES).
- Good grid modelling is increasingly important, for integration of renewables and possible electrification of transport and heating.



Python for Power System Analysis (PyPSA)

The FIAS software PyPSA is online at `http://pypsa.org/` and on github. It can do:

- Static **power flow**
- **Linear optimal power flow** (multiple periods, unit commitment, storage, coupling to other sectors)
- **Security-constrained linear optimal power flow**
- Total electricity system **investment optimisation**



Models

It has models for:

- meshed multiply-connected **AC and DC networks**, with controllable converters between AC and DC networks
- **standard types** for lines and transformers following the implementation in pandapower
- conventional **dispatchable generators** with **unit commitment**
- **generators with time-varying power availability**, such as wind and solar generators
- **storage units** with efficiency losses
- simple **hydroelectricity** with inflow and spillage
- coupling with **other energy carriers**
- **easy to extend**: basic components out of which more complicated assets can be built, such as Combined Heat and Power (CHP) units, heat pumps, resistive Power-to-Heat (P2H), Power-to-Gas (P2G), battery electric vehicles (BEVs)

Components

Network	Container for all other network components.
Bus	Fundamental nodes to which all other components attach.
Load	A consumer of energy.
Generator	Generator whose feed-in can be flexible subject to minimum loading or minimum down and up times, or variable according to a given time series of availability.
Storage Unit	A device which can shift energy from one time to another, subject to efficiency losses.
Store	A more fundamental storage object with no restrictions on charging or discharging power.
Shunt Impedance	An impedance in shunt to a bus.
Line	A branch which connects two buses of the same voltage.
Transformer	A branch which connects two buses of different voltages.
Link	A branch with a controllable power flow between two buses.

Technical details

PyPSA is written and tested to be compatible with Python 2.7 and Python 3.5.

It leans heavily on the following Python packages:

- pandas for storing data about components and time series
- numpy and scipy for calculations, such as linear algebra and sparse matrix calculations
- pyomo for preparing optimisation problems (currently only linear)
- networkx for some network calculations
- py.test for unit testing
- logging for managing messages

Design

- Network object is the overall container
- Components (buses, lines, transformers, links, generators, storage units) have static properties (nominal power, etc.) stored in pandas DataFrames
- Time-varying data (e.g. load, solar/wind availability) also stored in pandas DataFrames, indexed by a list of 'snapshots'
- Pyomo for optimisation
- Internal use of per unit power system quantities
- Set points (e.g. for loads, generators) are stored separately from actual dispatch points
- No GUI, use Jupyter notebooks

Power flow

In a power flow calculation, the user specifies the power dispatch of all dispatchable components (loads, generators, storage units, stores and links) and then PyPSA computes the resulting voltages in the network and hence the power flows in passive branches (lines and transformers) based on their impedances. For an AC network we must have:

$$S_n = V_n I_n^* = \sum_m V_n Y_{nm}^* V_m^*$$

where $S_n = P_n + jQ_n$ is the apparent power at node n, I_n is the complex current, V_n is the complex voltage and Y_{nm} is the bus admittance matrix.

For short overhead transmission lines that are close to their natural loading, these equations can be linearised to:

$$P_n = \sum_m (KBK^T)_{nm}\theta_m - \sum_\ell K_{n\ell} b_\ell \theta_\ell^{\text{shift}}$$

where K is the incidence matrix of the network, B are the series susceptances, θ_n are the voltage angles and $\theta_\ell^{\text{shift}}$ are transformer phase-shifts.

PyPSA: Full non-linear power flow with Newton-Raphson

Benchmarked against MATPOWER/PYPOWER and IEEE test cases:

```
import pypsa
from pypower.api import case118 as case

ppc = case()
network = pypsa.Network()
network.import_from_pypower_ppc(ppc)
network.pf()

print(network.generators_t.p)
print(network.generators_t.q)
print(network.buses_t.v_ang)
print(network.buses_t.v_mag)
```

Optimal power flow

PyPSA minimises the costs of generator dispatch $g_{n,s,t}$ at each node at each time t, and investment in generator $G_{n,s}$ and transmission capacity F_ℓ:

$$\min_{\substack{G_{n,s},F_\ell,\\ g_{n,s,t},f_{\ell,t}}} \left[\sum_{n,s,t} w_t o_{n,s} g_{n,s,t} + \sum_{n,s} c_{n,s} G_{n,s} + \sum_{\ell} c_\ell F_\ell \right]$$

subject to physical constraints (meeting energy demand, generator limits, storage consistency, physical flow conditions, thermal limits of branches).

Main constraints 1/2

- Demand $d_{n,t}$ is always met by generation or transmission

$$d_{n,t} = \sum_{s} g_{n,s,t} + \sum_{\ell \in n} f_{\ell,t}$$

- Dispatch $g_{n,s,t}$ cannot exceed availability $\bar{g}_{n,s,t}$

$$0 \leq g_{n,s,t} \leq \bar{g}_{n,s,t} \leq \bar{g}_{n,s}$$

- Installed capacity cannot exceed the installable potential $\hat{g}_{n,s}$

$$\bar{g}_{n,s} \leq \hat{g}_{n,s}$$

- CO_2 constraint is respected

$$\sum_{n,s,t} \frac{1}{\eta_{n,s}} g_{n,s,t} e_{n,s} \leq CAP$$

Main constraints 2/2

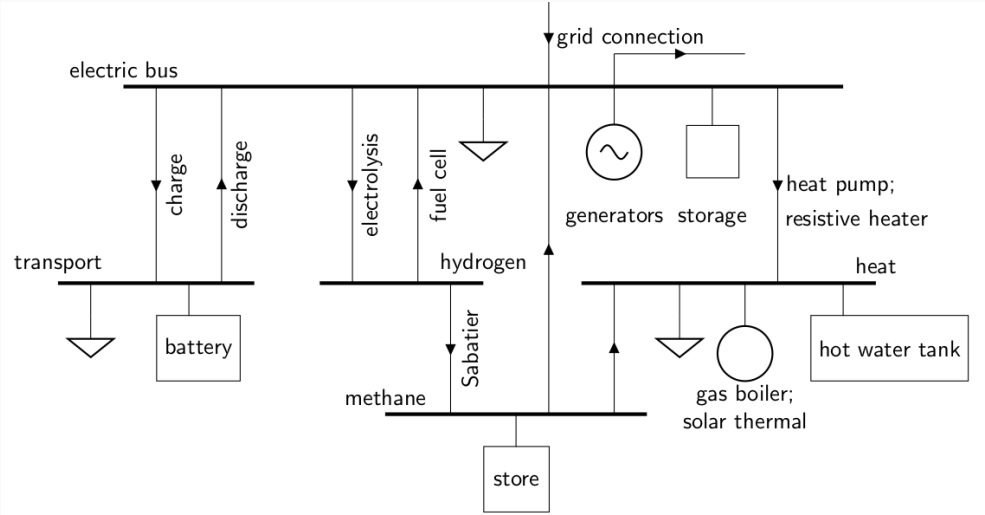
- Storage consistency

$$e_{n,s,t} = \eta_0^{w_t} e_{n,s,t-1} - \eta_1 w_t \left[g_{n,s,t}\right]^- + \eta_2^{-1} w_t \left[g_{n,s,t}\right]^+$$

- Flows $f_{\ell,t}$ have to by physical, i.e. obey Kirchhoff's Current and Voltage Laws for the linear load flow.
- New cycle-based formulations of the linear power flow are provided which are up to 20 times faster than standard angle-based formulations, see arXiv:1704.01881.
- Transmission flows cannot exceed capacities

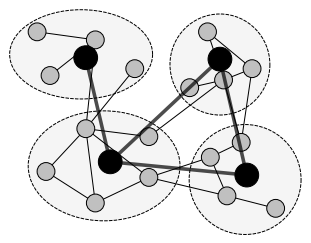
$$|f_{\ell,t}| \leq \bar{P}_\ell$$

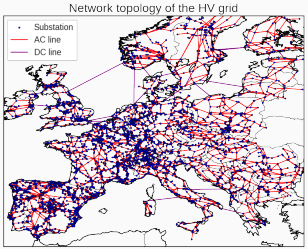
Coupling to Other Energy Sectors

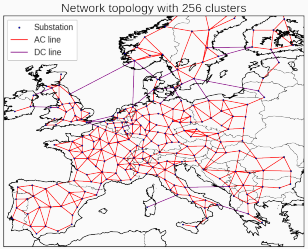


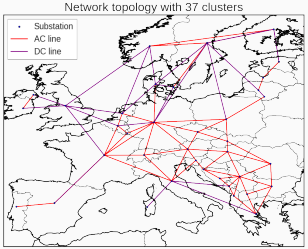
Network Clustering

Can cluster down the network, to reduce resolution while retaining important transmission lines:









Comparison to Other Software Tools

					Grid Analysis			Economic Analysis							
	Software	Version	Citation	Free Software	Power Flow	Continuation Power Flow	Dynamic Analysis	Transport Model	Linear OPF	SCLOPF	Nonlinear OPF	Multi-Period Optimisation	Unit Commitment	Investment Optimisation	Other Energy Sectors
Power system tools	MATPOWER	6.0	[?]	✓	✓	✓		✓	✓		✓				
	NEPLAN	5.5.8	[?]		✓		✓	✓	✓	✓	✓				✓
	pandapower	1.4.0	[?]	✓	✓			✓	✓		✓				
	PowerFactory	2017	[?]		✓		✓		✓	✓	✓				
	PowerWorld	19	[?]		✓		✓	✓	✓	✓	✓				
	PSAT	2.1.10	[?]	✓	✓	✓	✓		✓		✓	✓	✓		
	PSS/E	33.10	[?]		✓		✓		✓	✓	✓				
	PSS/SINCAL	13.5	[?]		✓		✓				✓				✓
	PYPOWER	5.1.2	[?]	✓	✓			✓	✓		✓				
	PyPSA	0.9.0		✓	✓			✓	✓	✓		✓	✓	✓	✓
Energy system tools	calliope	0.5.2	[?]	✓				✓				✓		✓	✓
	minpower	4.3.10	[?]	✓				✓	✓			✓	✓		
	MOST	6.0	[?]	✓	✓	✓		✓	✓	✓	✓	✓	✓		
	oemof	0.1.4	[?]	✓				✓				✓	✓	✓	✓
	OSeMOSYS	2017	[?]	✓				✓				✓		✓	✓
	PLEXOS	7.400	[?]					✓	✓	✓		✓	✓	✓	✓
	PowerGAMA	1.1	[?]	✓				✓	✓			✓			
	PRIMES	2017	[?]					✓	✓			✓	✓	✓	✓
	TIMES	2017	[?]					✓	✓			✓	✓	✓	✓
	urbs	0.7	[?]	✓				✓				✓	✓	✓	✓

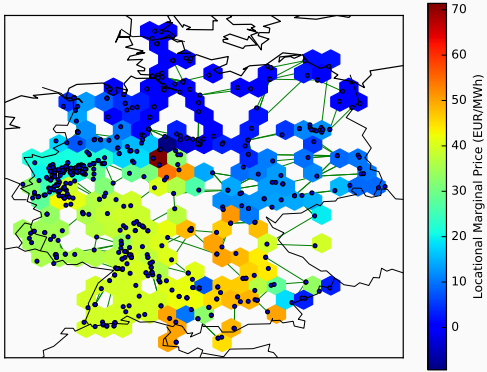
Example Usage of PyPSA

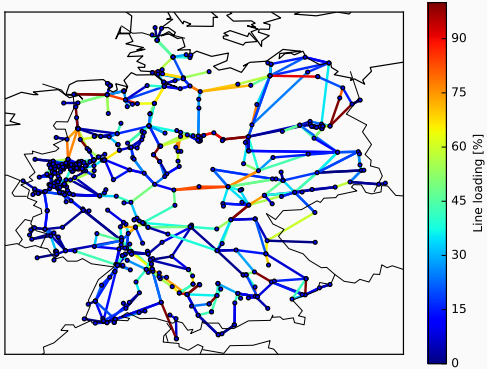
Example: linear OPF for a mixed AC-DC network

```
import pypsa

csv_folder_name=‘‘path/to/ac-dc-meshed/ac-dc-data’’

network = pypsa.Network(csv_folder_name)

solver_name = ‘‘glpk’’

network.lopf(snapshots=network.snapshots,solver_name)

print(network.generators.p_nom)  #series for each gen

print(network.generators_t.p)  #dataframe for each gen, snapshot
```

Example of LOPF in Germany

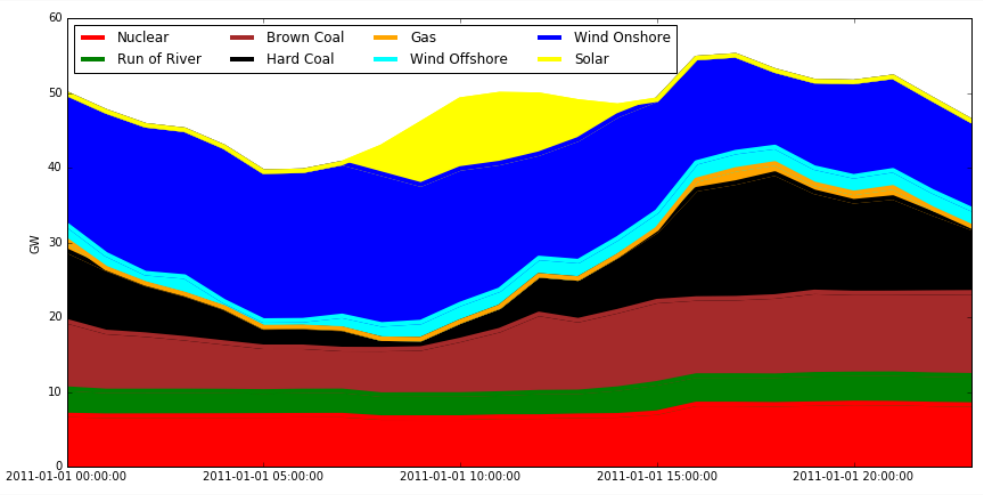
Using the **SciGRID open dataset**, nodal pricing for a day of high wind shows divergent prices in North and South Germany due to line overloading:



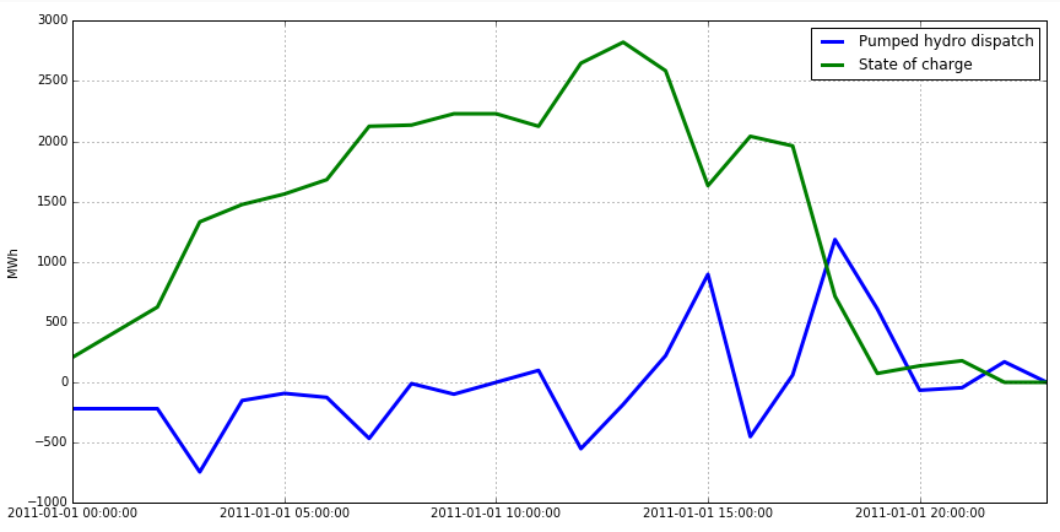


Dispatch over 24 hours includes pumped hydro

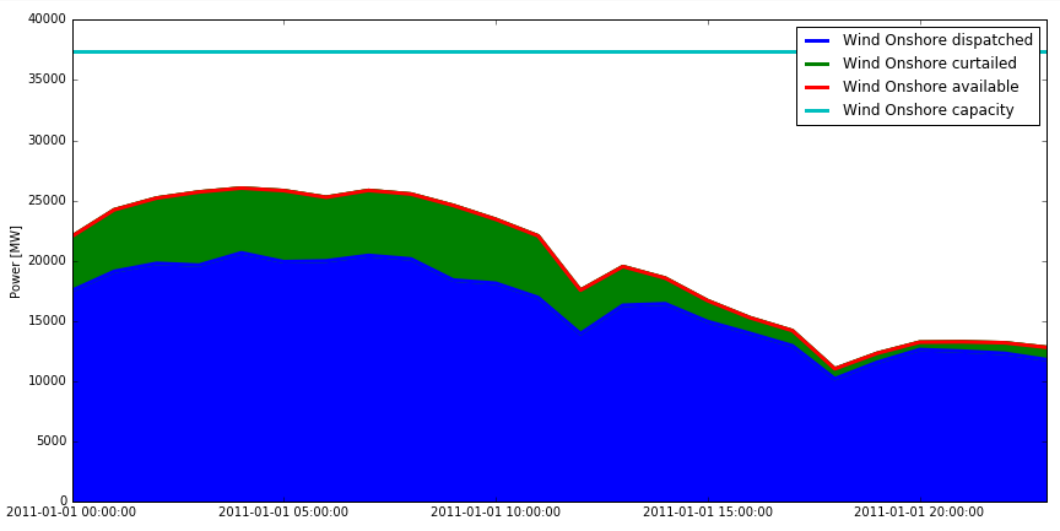
Dispatch is optimised over multiple periods (24 hours in this case), including storage and unit commitment (although latter ignored in this example).



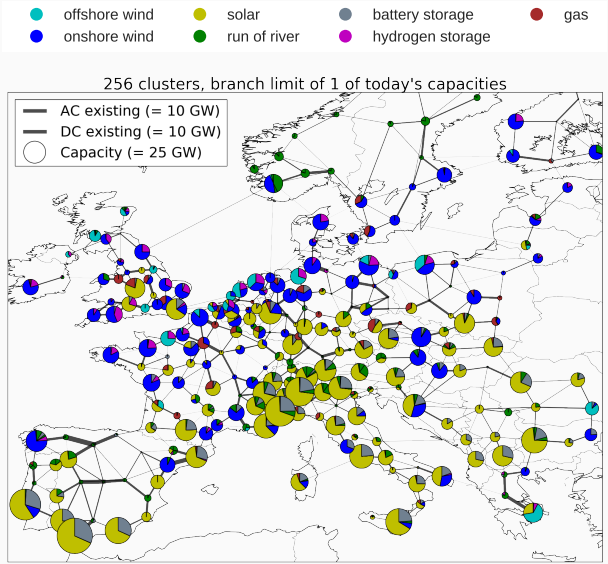
Pumped hydro power and energy



Wind curtailed because of grid bottlenecks



Example of highly renewable investment scenario

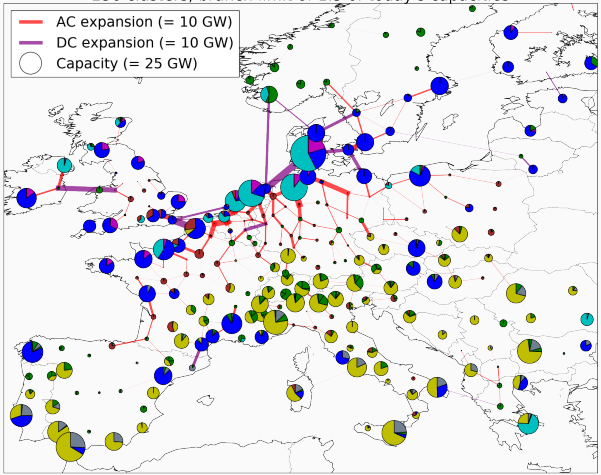


- Consider **joint optimisation** of transmission, generation and storage
- Force a **95% CO_2 reduction**
- With **no grid expansion**, lots of storage required, costs are high
- Such research is **complementary** to the TYNDP process
- Uses only **open data**, e.g. GridKit dataset

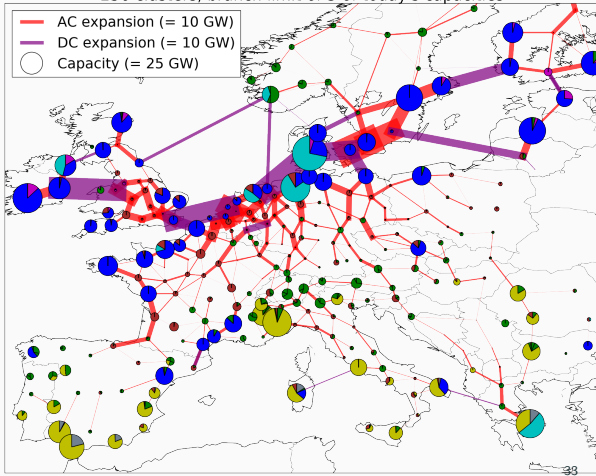
When grid expansion allowed: avoid costly storage

offshore wind | onshore wind | solar | gas | hydro | hydrogen storage | battery storage

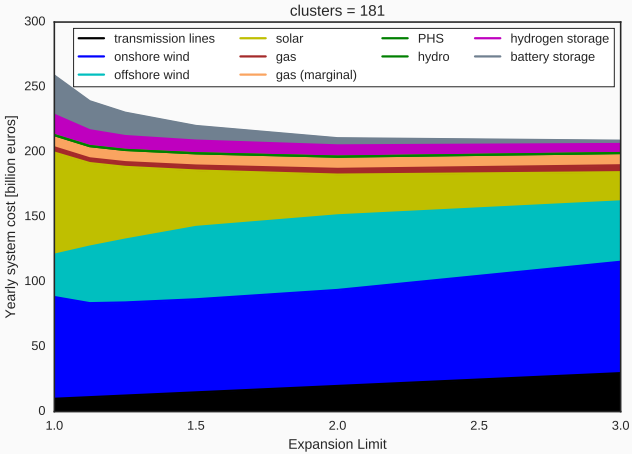
256 clusters, branch limit of 1.5 of today's capacities



256 clusters, branch limit of 3 of today's capacities



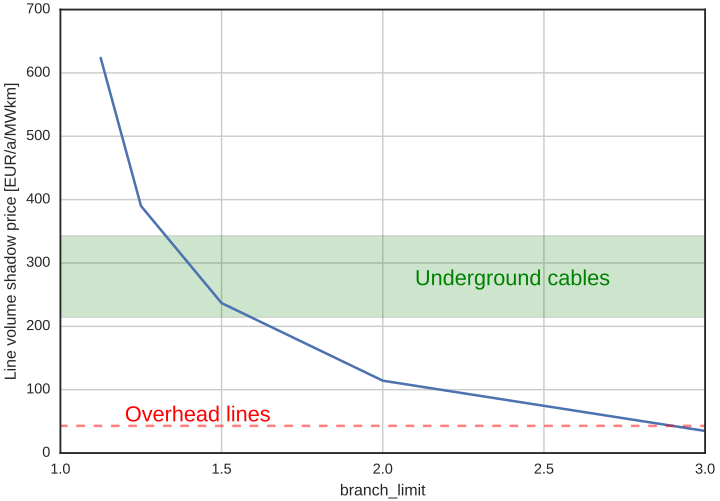
Behaviour as transmission expansion is allowed



- Big **non-linear cost reduction** as grid is expanded, from 82€/MWh to 66€/MWh (drop of 50 bill. €/a)
- Most of cost reduction happens with **25% grid expansion** compared to today's grid; costs rather flat once capacity has doubled
- Need for solar and batteries decrease significantly as grid expanded; with cost-optimal grid, system is dominated by wind

Source: Schlachtberger et al, 2017, Hörsch et al, 2017

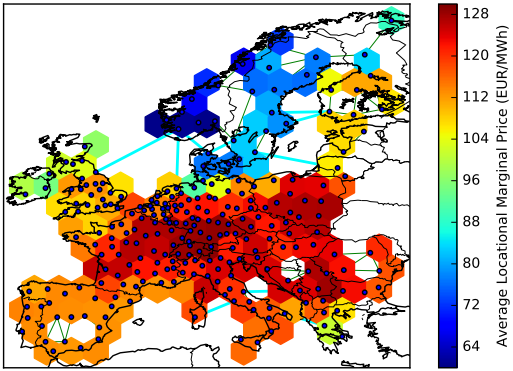
Grid expansion CAP shadow price for 181 nodes as CAP relaxed



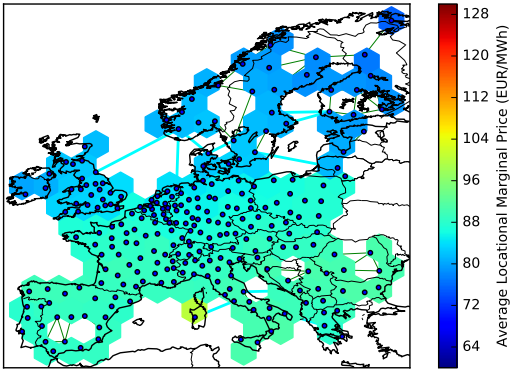
- With overhead lines the optimal system has around 3 times today's transmission volume
- With underground cables (5-8 times more expensive) the optimal system has around 1.3 to 1.6 times today's transmission volume

Locational Marginal Prices CAP=1 versus CAP=3

With today's capacities:



With three times today's grid:



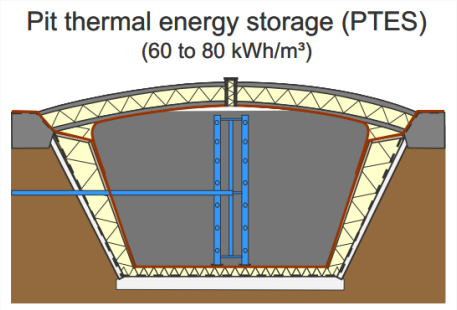
Coupling to Other Energy Sectors

Idea: Couple the electricity sector to heating and mobility.

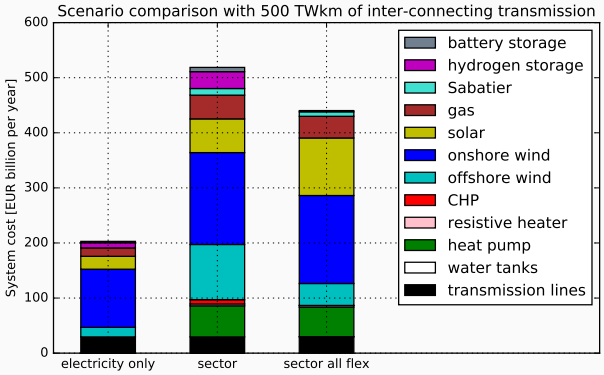
This enables decarbonisation of these sectors **and** offers more flexibility to the power system.

Battery electric vehicles can change their charging pattern to benefit the system and even feed back into the grid if necessary

Heat is much easier and cheaper to store than electricity, even over many months



Coupling to Other Energy Sectors



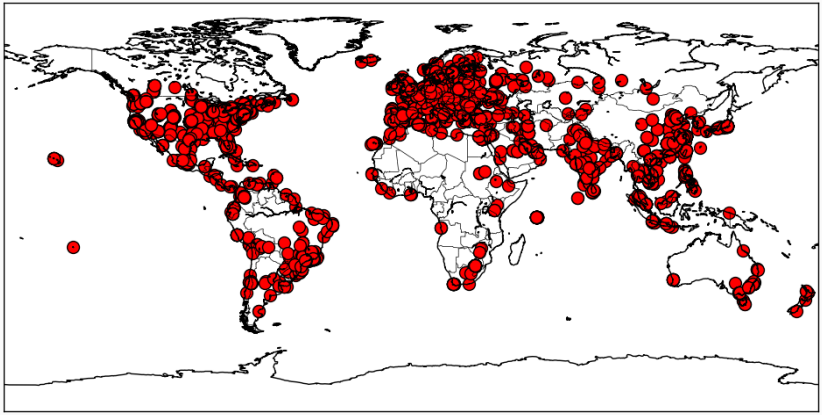
- Not just electricity: transport, heating and industrial must be considered to meet **Paris targets**
- Electrifying land-based transport and low-temperature heating would **increase electricity demand** by up to 60%, with strong seasonality
- However, these other sectors also offer significant **flexibility**: smart battery electric vehicle charging and thermal storage can play a big role

Python for Power System Analysis (PyPSA)

- Documentation and example Jupyter notebooks showcasing open data, like SciGRID: `http://pypsa.org/`
- Github: `https://github.com/FRESNA/PyPSA`
- Mailing list: `https://groups.google.com/forum/#!forum/pypsa`
- Research paper description: `https://arxiv.org/abs/1707.09913`

PyPSA users

PyPSA is being actively used by around a dozen institutions (that we know of...) and the website has been visited by people from 120+ countries:



Conclusions

Conclusions

- **Open energy modelling** increases **transparency**, **reproducibility** and **credibility**, which lead to better research and policy advice (no more 'black boxes')
- The **Open Energy Modelling Initiative** is an existing, friendly community with lots of combined experience to share
- There are many **existing open modelling frameworks** to choose from
- We have focused on our free software **Python for Power System Analysis (PyPSA)**: a toolbox for simulating and optimising modern power systems
- Many research results already obtained with PyPSA: **joint optimisation of generation and transmission** is important for cost-effectiveness, as is **coupling energy sectors**

Copyright