OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF
COMPUTER SCIENCE

# Open problems in energy modelling

Stephan Günther
Martin Glauer

Institut für Intelligente Kooperierende Systeme

18.12.2018 & 08.01.2019

# Characterisation of Problem Domains

Optimisation problem classes

Optimisation problem characteristics

Model transparency

Modelling process

# Optimisation Problem Classes

- Models are currently mostly formalized as linear programs (LP)
- But they are pushing what's possible with LPs
- *Investment* models already lead to mixed integer constraints
- *Unit commitment* models can cross the threshold to combinatoric optimization
- Non-simplified *ramping constraints* would make the optimization problems full partial differential equations

# Optimisation Problem Characteristics

- Optimisation problems can easily become too complex to even be solved as LPs
- Complexity reduction sometimes necessary:
- Spatial or temporal clustering
  - doesn't help with complexity arising from considering different levels of technological detail
  - mapping the clustered solution to the original problem is non-trivial
- Rolling time horizon
- Use alternative solution methods:
- heuristics, machine learning, agent based modelling

# Model Transparency

- Repeatable: The model can be used to by other parties than the model creators to repeat the experiments done with the model
- That means the model's source code has to be available and the model must be documented well enough so that other's can build and run the code.
- Reproducibility: The results obtained using the model can be reproduced by others using different approaches.
- That means the model's assumptions and methods have to be documented well enough for others to write a different model using the same assumptions and formulations to reproduce the model's results.
- Applicability: The model's results have to be understood by others in order for the model to aid in decision making

# Modelling Process

- Data acquisition: Gathering the data needed to run an energy system model is a tedious process. Even if successful, the data is more often then not, not under a clear license, limiting its usefulness greatly.
- Data integration: The data used to run an energy system model consists of a multitude of data sources in different formats, which are non-trivial to integrate with each other.
- Workflow: More often than not, modelling not only involves one model, but also lots of different unversioned scripts using the model to obtain a multitude of results. That leads to situations, where even the modeller himself has difficulties keeping track of how exactly his results where produced.

# Demand Prediction

- Energy demand is crucial for energy modelling
- Data is rarely disclosed by TSOs -> hard to acquire
- Need for demand prediction
- Many approaches have been applied - often including machine learning
- **But:** Results must remain explainable to ministries
- Conceptors? Neuro-symbolic integration?

# Merge Polygons

- Geographical regions are partitioned into smaller regions and depicted as polygons
- e.g.: Demand regions, administrative regions
- These polygons are displayed on the OEP but are rendered client side
- They form a (close-to) partitioning of a regions
- **Idea 1:** Simply merge them
- **Idea 2:** Convex hull

# Merge Polygons

Definition
An $\alpha$-**cocave hull** is a polygon such that all interior angles are less that or equal to $180 + \alpha$ degrees. [1]

- This is just one definition
- Concave hulls are a known tool in data science

# Units in a database

- Values in a database may have different units
- e.g. Euro in 2018 vs Euro in 2017

| id | value | unit | reference |
|---|---|---|---|
| 1 | 30 | EUR | 2017 |
| 2 | 50 | EUR | 2018 |

# Data Integration for Oemof

`Oemof`'s Workflow: Write a Python script which

- Creates a graph using `oemof.classes`.
- Generates an optimisation problem from this graph via `oemof.solph`.
- Solves the optimisation problem with the solver of your choice.

The last two steps are short and can usually be handled in one go using `oemof`.

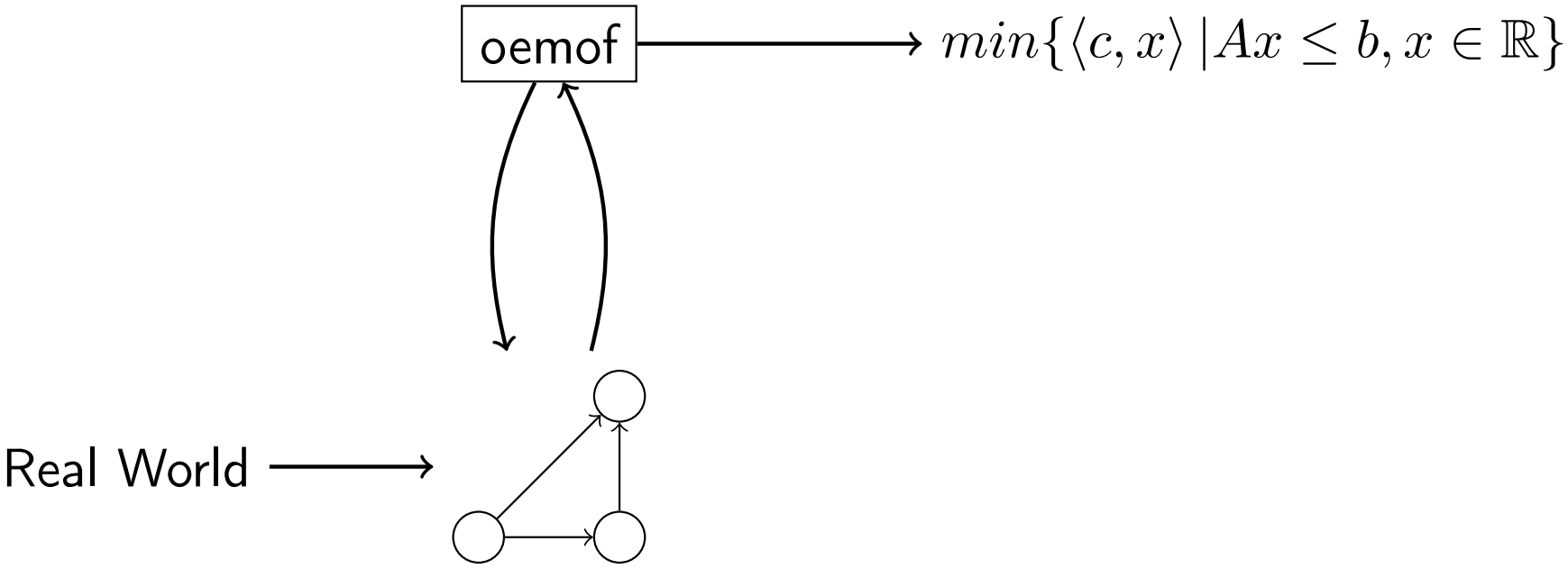
# Data Integration for Oemof

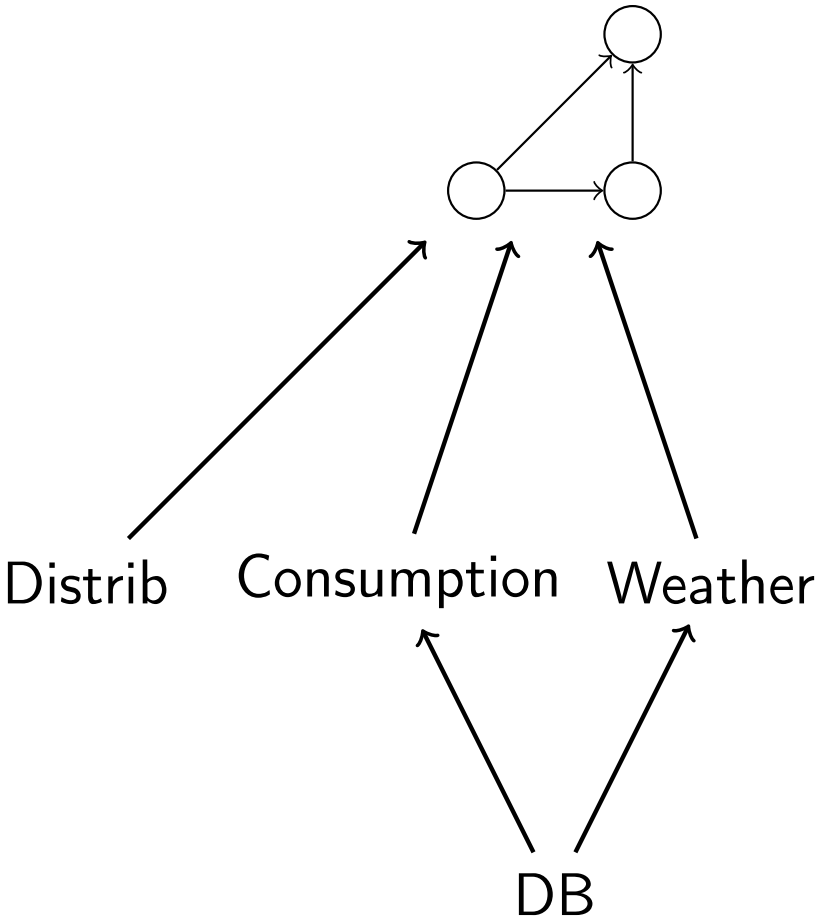
`Oemof`'s Workflow: Write a Python script which

- Creates a graph using `oemof.classes`.
- Generates an optimisation problem from this graph via `oemof.solph`.
- Solves the optimisation problem with the solver of your choice.

The first step though is rather time consuming and automating it is a usual feature request.

- Case in point: I'm currently doing something like this in "my other job".


oemof
$min\{\langle c, x\rangle \,|Ax \leq b, x \in \mathbb{R}\}$
Real World


Distrib
Consumption
Weather
DB