An Optimization Modeler as an Efficient Tool for Design and Operation for City Energy Stakeholders and Decision Makers

Camille Pajot, Lou Morriet, Sacha Hodencq, Benoît Delinchant, Yves Maréchal, Frédéric Wurtz, Vincent Reinbold

OMEGAlpes: An Optimization Modeler as an Efficient Tool for Design and Operation for City Energy Stakeholders and Decision Makers

Camille Pajot[1], Lou Morriet[1], Sacha Hodencq[1]
Benoit Delinchant[1], Yves Marechal[1], Frederic Wurtz[1], Vincent Reinbold[2]
[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, G2Elab, 38000 Grenoble, France
[2]Sorbonne Univ, Univ Paris Saclay, Univ Paris Sud, Grp Elect Engn Paris, CNRS, Cent Supelec, 91192 Gif Sur Yvette, France

Abstract

Nowadays, urban energy projects are becoming more complex in order to meet operational, economic and environmental challenges. This requires using decision support tools in pre-study phases to easily design and adapt the energy system model several times in order to meet stakeholders requirements. This paper presents OMEGAlpes, a linear optimization tool designed to easily generate multi-carrier energy system models. Its purpose is to assist in developing district energy projects by integrating design and operation in pre-studies phases. OMEGAlpes is open-source and written in Python. A use example is described to illustrate the modeling and solving of optimization problems in OMEGAlpes.

Introduction

Context of energy design and operation in cities

Nowadays, cities consume more than half of the global primary energy use, most of which is produced from fossil fuel power plants. Concurrently, climate change issues require a drastic decrease of CO_2 emissions. The development of low-carbon and decentralized renewable energies as well as the increase in energy recovery potential suggests that districts or cities will be the next place for energy production. However, these low-carbon energies create new challenges such as intermittency.

Multi-carrier energy systems and flexibility appear to be answers to the intermittency issues but increase the complexity of the energy systems. Intermittency also requires taking into account the energy system operation in the pre-design studies. In addition, the shift from centralized production to district level production make new actors like local producers, prosumers or even consumers to join the incumbent actors in decision-making on energy project.

The previous considerations raise the following question: how to design an energy project and manage energy flows in order to be optimal from ecological, financial, and other relevant points of view, while guaranteeing constraints such as thermal comfort for occupants or industrial requirements? One way to help stakeholders to answer this question is to help them to formulate, and then solve such complex problems.

State of art and choices for an optimization modelling tool

In order to integrate the complexity presented before in district energy system projects, numerous models and decision support tools have been developed to answer issues ranging from technological system and building system design to policy assessment (Keirstead et al. (2012)). Mendes et al. (2011) and Allegrini et al. (2015) focus on decision support tools dedicated to district energy systems, some of them will be presented later.

Allegrini et al. (2015) focus on simulation tools and highlight the difficulty to use classical dynamic simulation software, as TRNSYS , at district scale. In fact, simulation tools are based on trial and error resolution repeated on various simulations to improve the solution. The pre-study phase and the scaling up of simulation models to the neighborhood or city level is drastically reducing the capacity to compute it successfully, due to memory and time issues. Optimization tools seem thus more dedicated for design and energy management for district energy projects.

The complexity of the projects and the involvement of various stakeholders with different objectives and constraints require the ability to easily generate and adapt several times the models of the project as modeling tools do. Thus, a modeling tool is considered as a tool to help in generating energy system models. It should also help in identifying the whole of possibilities in energy project pre-studies as it is easy to consider various models. And help decision makers to understand the challenges linked to their visions of the energy project modelled as constraints and objectives. TEASER is a widespread example of modeling tool, but is dedicated to simulation models generation (Remmen et al. (2018)).

Only few multi-carrier energy project optimization modeling tools have been highlighted and are compared (Table 1). Over half of them are proprietary tools or require the use of proprietary tools: HOMER,

REopt, Artelys Crystal Energy Planner, and Ehub Modeling tool. DER-CAM is also a proprietary tool although it is free to use. However, proprietary tools prevent easy contributions from third parties and thus are unable to quickly adapt the model to the stakeholders needs. Furthermore, as Lopion et al. (2018) highlight, using proprietary tools also prevents the modeling review by third parties. These reasons lead us to orient ourselves towards open-source tools.

Recently, open-source optimization and modeling tools like oemof, ficus and thus OMEGAlpes have also been developed. OMEGAlpes and oemof have been developed at the same time. Oemof (Open Energy Modeling Framework) (Hilpert et al. (2018)) is an energy framework aiming at linking data edition and energy system optimization. Solph is the internal library of oemof aiming to help in modeling the energy system model providing basic energy units or few specific energy components. OMEGAlpes aims to provide a panel of pre-built energy units with predefined operational options, and associated constraints and objectives. Ficus is also an optimization modeling tool but developed for factory energy systems (Atabay (2017)). It aims to help in finding the minimum cost energy system to satisfy given demand time-series. Because it is mainly based on Excel for the model definition, it may be difficult to integrate more complex energy projects in the tool. Finally, Ficus and oemof are open-source however they use the GNU General Public License which limits the use of the tools considering that each contribution has to be open-source with the same license.

Contribution

OMEGAlpes is a multi-carrier open-source generation model tool based on an intuitive and extensible object-oriented library. This tool aims to offer the possibilities to consider various energy study-cases taking into account stakeholders constraints and objectives focusing on the optimization. We aim at taking into account both energy management and design optimization. OMEGAlpes is based on the Apache License software, less restrictive than the GNU General Public License.

Considering the consequential amount of variables involved due to the district scale and the desire to investigate early stage projects, and also considering the type of the variables continuous and integer -, we decided to focus on Mixed-Integer Linear Programming (MILP). As we can see in Table 1, most of the other tools identified use also MILP formulation.

Paper structure

The paper presents an overview of OMEGAlpes. The first section describes the main characteristics of OMEGAlpes and details the structure of the library from the modeling step to the solving step. Additionally, the second section presents a use case in order to illustrate the potential of OMEGAlpes as a modeling tool.

Methods

MILP, object-oiented programming and MDA approaches and principles

The OMEGAlpes library aims to help the generation of MILP optimization models. The most basic form of an optimization problem consists in adjusting a set of decision variables $(x_1, x_2, ..., x_n)$ to minimize an objective function (1).

$$\text{Minimize } F = f(x_1, x_2, ..., x_n) \tag{1}$$

However, most of the optimization problems encountered in engineering applications have to respect physical laws bounding the values of some decision variables. In this case, the problems are denoted as constrained optimization problems and can be mathematically expressed as (2):

$$\begin{cases} \text{Minimize} & F = f(x) \text{ for } x \in E^n \\ \text{Subject to:} & a_i(x) = 0 \text{ for } i \in (1, 2, ..., p) \\ & c_j(x) \geq 0 \text{ for } j \in (1, 2, ..., q) \end{cases} \tag{2}$$

According to the nature of the search space (or choice set) E and function f, a_i and c_j, the optimization problem (2) is qualified as linear, integer, quadratic, etc. As explained before, only MILP formulations are considered in this paper. In order to be classified linear, an optimization problem should take the following form (3):

$$\begin{cases} \text{Minimize} & f(x) - c^T x \\ \text{Subject to:} & Ax = b \\ & x \geq 0 \end{cases} \tag{3}$$

Where c $\in R^{n*1}$, A $\in R^{p*n}$ and b $\in R^{p*1}$. However, many energy systems problems include binary and discrete in addition to continuous variables. While many variables (e.g., power, energy, costs, etc.) are continuous, integer variables are needed for the formulation of typical constraints. For instance, minimum continuous run time of generators can be implemented using binary variables. More generally, each time that the state of a system (on/off, high-/low, charge/discharge) has to be constrained, binary variables are needed and thus a MILP formulation is then required. However, the standard form of a MILP problem is more complex as the variables could be either continuous ($x \in R^n$) or integer ($y \in Z^m$) and can be expressed as (4):

$$\begin{cases} \text{Minimize} & f(x) - c^T x - h^T y \\ \text{Subject to:} & Ax + Gy \leq b \\ & x \geq 0 \text{ for } x \text{ in } R^n \\ & y \geq 0 \text{ for } y \text{ in } Z^m \end{cases} \tag{4}$$

With this formulation, it becomes obvious that the optimization solvers can only interpret specific formatting of the optimization models, composed of

Table 1: Available multi-carriers energy project optimization modelling tools

Tool	Description	Programming	License	Commercial or free
Artelys Crystal Energy Planner (Artelys (2016))	Fully configurable software based on a full library model of energy units and operating process	MILP	proprietary	Commercial
DER-CAM (Berkeley Lab (2018))	Decision support tool that primarily serves the purpose of finding optimal distributed energy resource (DER) investments in the context of either buildings or multi-energy microgrids	MILP	proprietary	Free
Ehub Modeling tool (Bollinger and Dorer (2017))	Set of Matlab scripts for creating, executing and visualizing the results of an energy hub model for a given case study and a set of technologies.	MILP	open-source	Free, but requires Matlab
ficus (Atabay (2017))	Mixed integer optimization model for local energy systems	MILP	GNU GLP3	Free, but based on Excel
HOMER (HOMER (2018))	Optimize microgrid design in all sectors, from village owner and island utilities to grid-connected campuses and military bases	-	proprietary	Commercial
oemof (Hilpert et al. (2018))	Modular open source framework to model energy supply systems	MILP and others	GNU GPL3	Free
OMEGAlpes	Energy systems modelling tool for linear optimization (LP, MILP)	MILP	Apache 2.0	Free
REopt (Simpkins et al. (2014))	Optimize the size and operating strategy of microgrids, storage, and energy/water systems	MILP	proprietary (exept REopt lite)	Commercial

equalities and inequalities. The four main formats and modeling languages used for linear programming are MPS, LP, AMPL and GAMS (Benson (2011)).

- MPS (Mathematical Programming System) is the oldest input format for linear problems. It is widely used by both commercial and academic solvers, but can be difficult for a user to read.
- LP (Linear Programming) format is easier for users to understand by describing the problem with readable algebraic expressions. However, it could be harder for solver to interpret this format.
- AMPL (A Mathematical Programming Language) was developed at Bell Laboratories and became the standard modeling language for optimization problems. Many solvers (such as CPLEX, Gurobi, Xpress or CBC) are supported by AMPL.
- GAMS (General Algebraic Modeling System) was the first algebraic modelling language for mathematical programming and supports the most famous optmization solvers (CPLEX, Gurobi, Xpress or CBC).

As explained before, one of the aims of OMEGAlpes consists in avoiding the creation of an optimization model for each study case, by using the concept of Model-Driven Architecture (MDA). Launched by the Object Management Group in 2001, this approach is used in software developments to switch between a very-high level of abstraction to specific models (Kleppe et al. (2003)). The high-abstraction level is more related to a human understanding and could sometimes be represented with diagrams, while the specific models are closer to a description for a particular software. Here, the very-high level of abstraction corresponds to the formulation of the study case of the problem on energy systems. For instance, it could be a building that consumes heat power from the heating network, while aiming to minimize its CO_2 emissions. In order to be solved, this optimization problem has to be translated into a specific language to be understood by an optimization solver.
For open-source purposes, OMEGAlpes was written in Python, an open-source and widely-used high-level programming language. Moreover, Python supports object-oriented programming, a crucial feature for the construction of multiple abstraction layers. The OMEGAlpes library relies on this concept by creating complex classes from elementary objects. More detailed explanations about the construction can be found in the next section.

Package structure of OMEGAlpes

OMEGAlpes is based on low-abstraction elements in order to create high-abstraction classes. For instance,

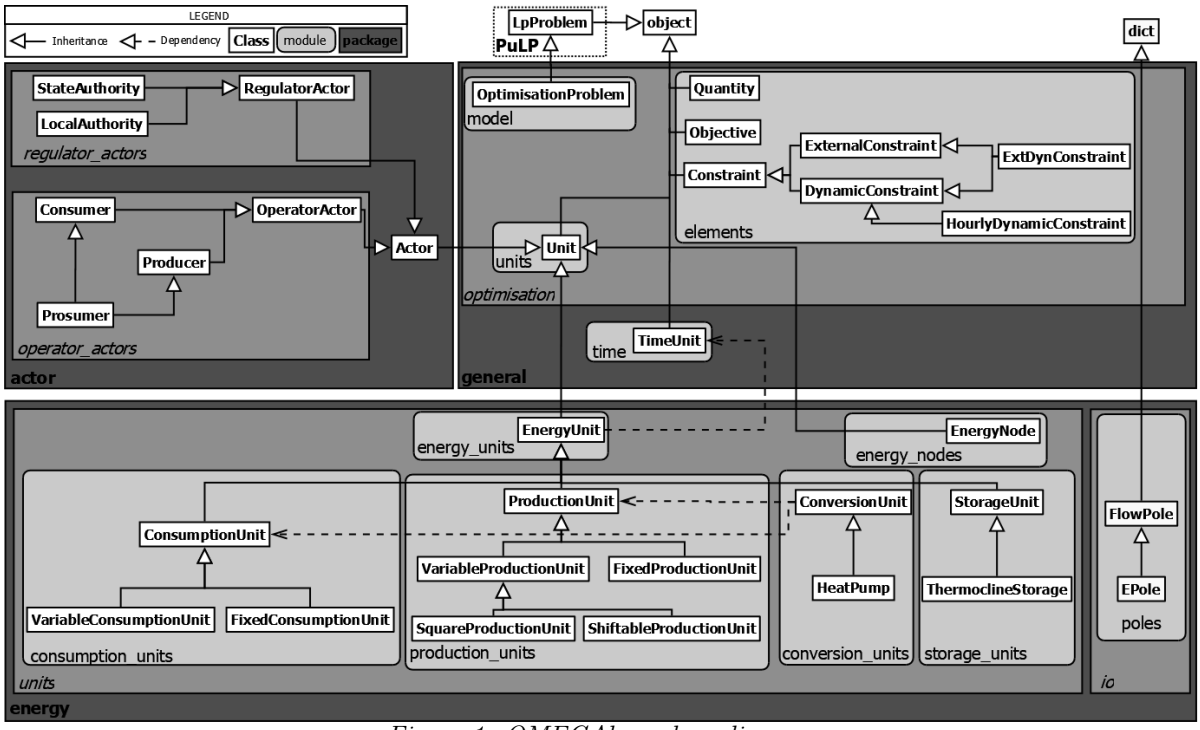

Figure 1: OMEGAlpes class diagram

a low abstraction element `Unit` creates a high abstraction element `HeatPump`. The available classes are represented on Figure 1 and belong to one of the three main sub-packages: general (e.g. optimization and time related elements), energy (e.g. production, consumption, storage and conversion) and actor (e.g. regulator and operator actors).

To translate the models related to energy and actors concepts into an optimization model, we used the python package PuLP, which will be detailed later in this paper (Mitchell et al. (2011)).

The general sub-package represents the first abstraction layer, by providing all the classes needed for the optimization problem. Indeed, the `OptimisationProblem` class inherits from the `LpProblem` class provided by PuLP to project the Python model into optimization models. The *elements* module contains the optimization elements needed to formulate a constrained optimization problems:

- `Quantity` defines a decision variable or a parameter
- `Objective` defines an objective (the objective of the optimization problem will be the sum of all objectives)
- `Constraint` defines a constraint
  - `DynamicConstraint` defines a constraint with a time-dependency
    - `HourlyDynamicConstraint` defines a constraint that repeats each time for an hour range (for instance from 7am to 9am)
  - `ExternalConstraint` defines a constraint which does not reflect a physical equation (for instance, a power plants that does not operate during the night).

In the *general.optimisation.units* module, a `Unit` is defined to represent the elementary object of an optimization problem. A `Unit` is a set of the three optimization elements mentioned above (`Quantity`, `Objective` and `Constraint`). An other elementary class is included in the general package: `TimeUnit`. This class is needed to define the dynamic of the study case (i.e. the time step, the duration, ...). Moreover, the time module contains several methods based on the time series data analysis package: Pandas[1], in order to help the user to link the index of the decision variables (low-abstraction) to dates and hours (high-abstraction).

The *energy* package gathers all the models used in OMEGAlpes to describe an energy system. The Figure 2 shows an example of an energy system that links various configured energy units: `ProductionUnit`, `ConsumptionUnit` and `StorageUnit` through an energy node, with constraints and an objective. In this example, the objective is to minimize the capacity of a storage unit while providing energy to a fixed consumption unit, thanks to a variable production unit with maximum (pmax) and minimum (pmin) power boundaries. The storage unit also has maximal charging and discharging power (pc_max and pd_max). Energy nodes allow to link energy units of the same energy type while ensuring the power balance.

[1] pandas.pydata.org

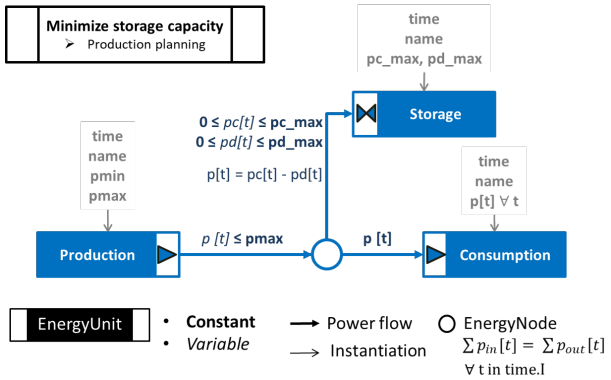

Figure 2: Principle diagram of an energy system optimization problem modelled with OMEGAlpes

The various parameters and objectives of the main energy unit classes are detailed in the Figure 3.

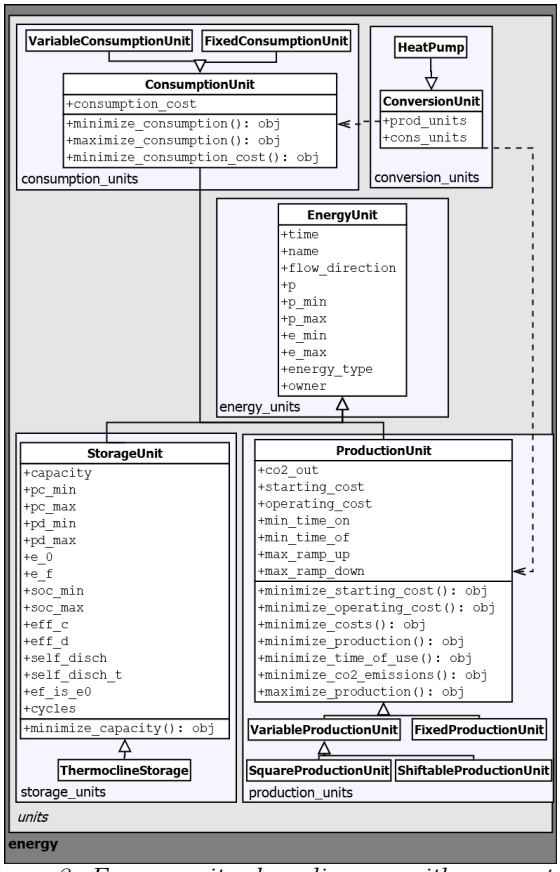

Figure 3: Energy units class diagram with parameters and objectives

In OMEGAlpes, inputs and outputs of energy units are defined as poles: the generic pole is `FlowPole`, a pole with a directed flow, and `EPole` defines an energy pole with a power flow and an energy type. In order to help modelers to take into account stakeholders objectives and constraints in the design process, we developed an actor modeling layer . This actor layer enables a bigger panel of modeling possibilities. However, an energy project model can be generated without integrating the actor modeling. We identified two main actor categories and divided them into the following packages:

- The *operator_actors* package focuses on stakeholders who operates energy units and have a scope of responsibility on these energy units. `Consumer`, `Producer` and `Prosumer` are operator actors and are defined as classes. Using the oriented object modeling, the `Prosumer` class inherits from the `Consumer` and the `Producer` ones. Modeling an operator actor enables to add the actors constraints and objectives to all or part of the energy units in the area of responsibility.
- The *regulator_actors* package focuses on stakeholders who do not operate energy units but influences final decisions via network (economical, values) and resource regulation.

In Figure 4, the previous example is described from an actors point of view. The two main actors are represented with their own scope of responsibility: one for the production unit, the other for the storage and consumption unit. The objectives, which may be contradictory, are associated to each actor. Additionally, the producer imposes a CO2 limit constraint on its production unit.

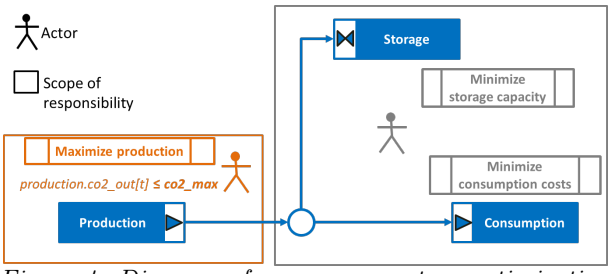

Figure 4: Diagram of an energy system optimization problem that considers actors constraints and objectives

Optimization model generation and resolution

Once created with the packages detailed above, the study case has to be translated into an optimization model. As already explained, several formats are commonly used to model an optimization problem (MPS, LP, AMPL and GAMS). In order to write this optimization model from the Python description of the study case, we use the PuLP package. Mitchell et al. (2011) defined PuLP as a "linear programming toolkit for Python", focused on supporting linear and mixed-integer models. Similarly to PuLP, the Pyomo framework can also be used to describe optimization problems in Python. Where Pyomo benefits from the ability to express non-linear models, PuLP has the advantage of keeping the formulation simple, while guaranteeing the interoperability with many solvers as well as Pyomo does. Indeed, two standard formats are available in PuLP for the model generation: LP and MPS. Thus, once generated, the optimization model can be solved by a large range of

solvers. The LP and MPS formats allow the interfacing with popular commercial solvers such as CPLEX or Gurobi. For a fully open-source utilization of the OMEGAlpes library, a CBC solver is included in the PuLP library (COIN-OR Foundation (2018)), allowing OMEGAlpes to handle all the steps from the study case description to the optimization problem resolution.

Open source and collaborative development

OMEGAlpes has been developed with the ambition of creating a community of users and developers, and efforts were made to facilitate the involvement and contributions of this community. First, OMEGAlpes development was driven with an open-source philosophy. Thus, OMEGAlpes is coded with Python, based on the open-source license Apache Software License 2.0 (Apache Software Foundation (2004)), and can be freely downloaded through The Python Package Index (PyPI). Then, a detailed documentation[2] and energy study cases[3] allow the users to easily pick up OMEGAlpes. Finally, a versioning process of OMEGAlpes development based on a GitLab project[4] enables to track the contributions and to facilitate collaborative development. This ensures OMEGAlpes ambition to create a community and enrich the model library, but also to perpetuate the study cases presented in scientific papers.

Example application

The aim of this subsection is to show how to use OMEGAlpes on a study case. For this purpose, we use demand-side management to maximize self-consumption of photovoltaic (PV) generation. We do so by shifting the energy consumption of a clothes washing machine and a dryer, and by storing heat into a water tank (see Figure 5). However, the focus here is not on the example in itself but how to easily study a complex problem thanks to OMEGAlpes.

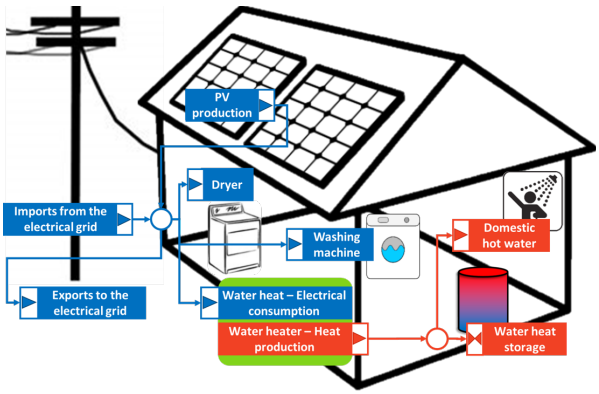

Figure 5: Representation of the study case

[2] https://omegalpes.readthedocs.io

[3] https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes/omegalpes-examples

[4] https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes

Our goal is to move from the representation of the example drawn in Figure 5 to its formulation with OMEGAlpes, step by step (available online[5]).

A typical study case can begin with two code lines creating an `OptimisationModel` and the time characteristics.

```
model = OptimisationModel(name="example")
```

The time characteristics are gathered in the class `TimeUnit`, that may include:

- the start (start)/ the end (end)
- the time step in hours (dt)
- the number of time steps (periods)

```
time = TimeUnit(periods=24*12, dt=1/12)
```

In this example, the study is realized on a 5-minutes time step (1/12 hours) during a day (the number of periods equals 24 hours*12 time steps, i.e 288 periods).
Then, the energy package can be used in order to describe the study case. In this example, there is two types of load:

1. The electrical loads whose starting time is not defined but whose consumption profile is known, such as the clothes dryer. This type of unit is available with the class `ShiftableConsumtpionUnit`, where the `power_values` corresponds to its consumption profile, that can be shifted.
   ```
   ShiftableConsumptionUnit( time,
   "dryer", power_values=dryer_load,
   energy_type="Electrical")
   ```
2. Non-shiftable heating load whose consumption profile cannot be changed, such as the domestic hot water. The corresponding model can be found under the class `FixedConsumptionUnit` and can be instantiated by the entire consumption profile on the time period (p).
   ```
   FixedConsumptionUnit(time, "dhw", p=
   dhw_load, energy_type="Heat")
   ```

When the production profile is known, OMEGAlpes provides a `FixedProductionUnit` class. In addition to the fixed electric production from the PV panels, the house is connected to the electrical grid. The power exported to the electrical grid is modeled as consumption, while the imported power is represented as production. As the power profiles of these units are not fixed by advance, the classes `VariableProductionUnit` and `VariableConsumptionUnit` are used.

Besides the electrical appliances' consumption, electricity is used by a water heater to meet the domestic hot water demand. The object `ElectricalToHeatConversionUnit(time, "water_heater", elec_to_heat_ratio=0.9)` represents the water heater's electrical consumption and

[5] https://tinyurl.com/OMEGAlpes-Basic-Example

the heat production. The parameter elec_to_heat ratio (90%) is the efficiency of the water heater.

Finally, the heated water can be partly stored into a water tank.

```
StorageUnit(time, "water_tank", capacity=
6000, ef_is_e0=True, self_disch=0.05,
soc_min=0.2, energy_type ="Heat")
```

In this case, the capacity of the thermal storage is 6000 kWh. The initial state of charge (SoC) has to be equal to the SoC at the end of the period (ef_is_e0=True). Moreover, the SoC is not allowed to go below 20% and the storage has a self-discharge of 5% per hour.

We add an external constraint to model the fact that the dryer cannot be launched before the end of the washing machine cycle. This constraint is considered to be an `ExternalConstraint` because it represents an operational, rather than a physical constraint. Moreover, this constraint is time-dependent and is created as a `DynamicConstraint`. For these reasons, the `ExtDynConstraint`, inheriting from both classes is used as follows:

```
cst = ExtDynConstraint(name= "wait_to_dry",
exp_t ="dryer_start_up[t] ≤ lpSum(
clothes_washer_switch_off[k] for k in
range(0, t))", parent=clothes_dryer)
```

Then, the constraint is added to the clothes dryer through the following command:

```
setattr(dryer, "wait_to_dry", cst)
```

To specify the objective, we apply a function available for the class `ProductionUnit`: `minimize_production` to the imports from the electrical grid:

```
imports.minimize_production()
```

The method sets the objective corresponding to the minimization of the electrical imports from the grid, i.e. the maximization of the self-consumption. Finally, the energy nodes are created using the following code:

```
EnergyNode(time, "heat_node", energy_type=
"Heat")
```

The units have to be connected to the corresponding node:

```
heat_node.connect_units(dhw, water_tank,
water_heater.heat_production_unit).
```

Now that the energy model is set, it can be added to the empty optimization model by adding the nodes:

```
model.add_nodes(elec_node, heat_node).
```

Finally, the optimization is launched by the command: `model.solve_and_update()`.

With 6922 variables (2890 continuous and 4032 binary) and 79172 non-zeros, this optimization problem is generated within 1,2 seconds on an Intel bicore i5 2.4 GHz CPU. Two MILP solvers are compared in term of performances. For the free CBC solver available in the PuLP package, the optimal solution was found in 43,6 seconds, while the commercial Gurobi solver provided an optimal result in 2,5s.

Regarding the results of the study case, the reference case with no thermal storage and electrical appliances not starting before 6pm, only 3% of the electrical consumption is self-consumption while the optimization reaches 53% of self-consumption with a demand-side management strategy. Self-consumption can be significantly improved with optimal planning, so that a quick and easy formulation of the optimization problem can be game changing for prosumers.

More generally, the generation of optimization models through the combination of pre-defined units can help many actors to study optimal energy planning. While we have focused on a simple example, OMEGAlpes was also used for optimization problems with hundreds of thousands of variables, solved by Gurobi in several minutes.

Conclusion and Perspectives

Conclusion

In many urban contexts, cities and districts face numerous challenges in terms of energy projects. Besides technical and financial issues, considering the environmental and social aspects is required more and more, which renders energy systems very complex problems. The optimization tool OMEGAlpes (Optimization Models Generation As Linear Programming for Energy Systems) was created to help the stakeholders to address these new challenges.

As indicated by its name, OMEGAlpes generates linear optimization models for energy systems. Indeed, this tool can be described as a MILP modeling tool, with a focus on multi-carriers energy projects at the district level. Written in Python, OMEGAlpes relies on the object-oriented concept to gather energy and actors models for a quick design of various energy study cases. A low-abstraction layer is based the the PuLP package in order to translate the study cases into MILP models with two available formats (MPS and LP). Then, the model is sent to various solvers (CBC, CPLEX, Gurobi, etc.) for its resolution. Available at https://gricad-gitlab.univ-grenoble-alpes.fr/omegalpes, this Python library is fully open-source with a permissive license for a non-restrictive contribution.

An example of utilization for PV self-consumption with electric load shifting and water tank management according to domestic hot water needs is available online at https://tinyurl.com/OMEGAlpes-Basic-Example.

Besides the energy package used on this study case, actors models are also integrated in OMEGAlpes to explicit the actors' objectives and constraints. However, OMEGAlpes have been designed for pre-studies

and does not integrate very-detailed models such as those required for real-time management.

Perspectives

One perspective of our work is thus to link OMEGAlpes with simulation tools, based on more detailed models. This would enable one to use OMEGAlpes for model predictive control.

Alternatively, thermal building models have been integrated to OMEGAlpes in order to enable the modeling of heat building consumption curves based on an RC model.

Aiming to facilitate the use of the library, a Graphical User Interface may be developed based on the package presented before.

Finally, as presented before, OMEGAlpes aims to be developed by a community of developers and so integrate new contributions.

Acknowledgment

This work has been partially supported by the CDP Eco-SESA receiving fund from the French National Research Agency in the framework of the "Investissements davenir program (ANR-15-IDEX-02).

References

Allegrini, J., K. Orehounig, G. Mavromatidis, F. Ruesch, V. Dorer, and R. Evins (2015). A review of modelling approaches and tools for the simulation of district-scale energy systems. *Renewable and Sustainable Energy Reviews 52*, 1391 – 1404.

Apache Software Foundation (2004). Apache License, Version 2.0.

Artelys (2016). Artelys crystal energy planner.

Atabay, D. (2017). An open-source model for optimal design and operation of industrial energy systems. *Energy 121*, 803 – 821.

Benson, H. Y. (2011, January). Interior-Point Linear Programming Solvers. In *Wiley Encyclopedia of Operations Research and Management Science*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Berkeley Lab (2018). Distributed Energy Resources - Customer Adoption Model (DER-CAM) | Building Microgrid.

Bollinger, L. A. and V. Dorer (2017). The ehub modeling tool: A flexible software package for district energy system optimization. *Energy Procedia 122*, 541 – 546. CISBAT 2017 International ConferenceFuture Buildings & Districts Energy Efficiency from Nano to Urban Scale.

COIN-OR Foundation (2018). COIN-OR: Computational Infrastructure for Operations Research.

Hilpert, S., C. Kaldemeyer, U. Krien, S. Gnther, C. Wingenbach, and G. Plessmann (2018). The open energy modelling framework (oemof) - a new approach to facilitate open science in energy system modelling. *Energy Strategy Reviews 22*, 16 – 25.

HOMER (2018). HOMER - Hybrid Renewable and Distributed Generation System Design Software.

Keirstead, J., M. Jennings, and A. Sivakumar (2012). A review of urban energy system models: Approaches, challenges and opportunities. *Renewable and Sustainable Energy Reviews 16*(6), 3847 – 3866.

Kleppe, A. G., J. Warmer, J. B. Warmer, and W. Bast (2003). *MDA Explained: The Model Driven Architecture : Practice and Promise*. Addison-Wesley Professional.

Lopion, P., P. Markewitz, M. Robinius, and D. Stolten (2018). A review of current challenges and trends in energy systems modeling. *Renewable and Sustainable Energy Reviews 96*, 156 – 166.

Mendes, G., C. Ioakimidis, and P. Ferro (2011). On the planning and analysis of integrated community energy systems: A review and survey of available tools. *Renewable and Sustainable Energy Reviews 15*(9), 4836 – 4854.

Mitchell, S., M. OSullivan, and I. Dunning (2011). PuLP: A Linear Programming Toolkit for Python. pp. 12.

Remmen, P., M. Lauster, M. Mans, M. Fuchs, T. Osterhage, and D. Mller (2018, January). TEASER: an open tool for urban energy modelling of building stocks. *Journal of Building Performance Simulation 11*(1), 84–98.

Simpkins, T., D. Cutler, K. Anderson, D. Olis, E. Elgqvist, M. Callahan, and A. Walker (2014). REopt: A Platform for Energy System Integration and Optimization. (45875), V002T03A006.