IESopt: A Modular Framework for High-Performance Energy System Optimization

1st Stefan Strömer
Center for Energy
AIT Austrian Institute of Technology GmbH
Vienna, Austria
orcid.org/0000-0002-5330-3318

2nd Klara Maggauer
Center for Energy
AIT Austrian Institute of Technology GmbH
Vienna, Austria
orcid.org/0000-0002-5994-3201

***Abstract*—Current climatic, political, and societal challenges pose increasingly complex questions, which in turn require comprehensive models of the real world, with rapidly growing complexity, to support decision makers with sound and reliable quantitative analyses. The energy system optimization framework IESopt may constitute one piece in filling this gap, by offering a modular and adaptable tool for modelers, that does not compromise on performance while still being user-friendly. This is enabled by reducing energy system assets to abstract building blocks, that are supported by specialized implementation, and can be combined into complex systems without the need of a detailed understanding of mathematical modeling or proficiency in any coding-language. IESopt's architecture and functionalities are laid out here, and demonstrated by the means of an illustrative example.**

***Index Terms*—energy system modeling, optimization framework, Julia, open-source-tool introduction**

I. Introduction

The rising challenges linked to climate change and the negative influence of the past and current energy system (and its emissions) on it, have put an ever increasing importance and interest on comprehensive depictions of the real world. Sound political and regulatory decisions require accurate studies and projections of the world's energy future. Many open-source community projects and frameworks are working on detailed collections and pre-processing of the vast amounts of required data (e.g., [1] for sector-coupled European datasets, [2] for technology-specific data across multiple modeling years, or [3] for high-resolution climate projections), which form the basis for many existing modeling tools. Many of those are built upon the existing Python ecosystem (c.f. [4] or [5]). Existing reviews [6], [7] provide indication of differences and similarities between these tools.

However, the rising complexity of the energy systems that are being modeled (e.g., driven by longer time-horizons and increased sector-coupling) poses computational challenges for existing tools. The implications and possible ways to tackle these problems have already been subject to increasing attention in recent years – [8] considers the interpretability of simplified models, [9] instead analyses explicit mathematical properties impacting performance. Further, many tools already take a considerable amount of time to even construct the necessary mathematical formulations. Workarounds for Python-based models (notably [10]) have emerged, with modern approaches, c.f. [11], explicitly stating their benchmark-to-beat: *"[...] comparable with JuMP.jl"*. These challenges and developments, amongst other factors, motivate why new energy system modeling frameworks that are fully implemented in Julia and make use of JuMP.jl [12] have emerged recently [13], [14].

With a similar motivation, the framework IESopt[1], *Integrated Energy System Optimization*, has been in development since 2021 at AIT Austrian Institute of Technology GmbH's *Center for Energy*. During the last three years, it has been successfully applied as main energy system optimization framework in various projects (e.g., optimizing Austrian hydrogen infrastructure in [15], operational planning for industrial sites using model-predictive-control in [16], or household scheduling in [17]). Moreover, it is currently employed in large ongoing funded research projects (for modeling hydrogen as part of a sector-coupled Europe across different layers of detail in [18]). In addition, IESopt has been used in publications (from energy communities in [19], to national hydrogen infrastructure in [20], up to models of future global energy systems in [21]) and resulted in technical talks [22].

While insights, discovered bugs, and implementations have been shared with the community "along the way", a final move to open-sourcing the full framework was a main goal to further foster collaboration. The following overview and introduction of IESopt is structured as follows: Section II summarizes the main software architecture, and provides a high-level insight into the mathematical formulations, whereas section III showcases some features on an exemplary use-case, while IV concludes.

II. Framework Description

The software framework IESopt consists of two main components: The modeling core, IESopt.jl[2], is implemented entirely in Julia and handles all critical functionality, while the Python wrapper IESopt acts as convenience layer on top of it. IESopt.jl applies an, as far as reasonable, non-opinionated

[1] See: github.com/ait-energy/IESopt

[2] Mostly abbreviated as *core* or *IESoptCore* to prevent confusion for users not familiar with Julia package naming conventions and/or the distinction between Python and Julia.

structure to enable full flexibility for any user in how they interact with it, while the Python wrapper abstracts some of it away to streamline the interaction with the framework based on widespread Python "conventions", like a direct conversion to pandas [23] DataFrames, or by wrapping all internal functionality into proper Python objects. Nonetheless, it allows full access to the core model (using juliacall [24]), e.g., exposing all optimization "objects" (variables, constraints, etc.), allowing the user to modify them from Python.

IESopt.jl supports and implements linear programming (LP) and mixed-integer linear programming (MILP) formulations, but is structured to also allow quadratic objective coefficients (for QPs), and even be extended by users using non-linear formulations (as far as supported by JuMP).

A. Foundation

IESopt.jl builds upon JuMP as its main optimization backend. JuMP allows efficient and performant interfacing with various commercial and open-source solvers and provides a convenient generalized syntax to define a mathematical optimization problem from Julia code. Besides various standard functionalities (e.g., reading/writing JSON, YAML, or CSV files), it makes use of existing JuMP extensions, as well as the existing Julia ecosystem (mainly to construct clear documentation, ensure continuous testing and quality checks of the code base, as well as ongoing performance monitoring and benchmarks).

B. Architecture

One of the main defining pillars of the IESopt framework is a complete no-code approach. The core, IESopt.jl, implements a "YAML + X" based configuration interface as only direct way of specifying a model. Here "X" summarizes additional file formats that can be used to better structure large models, e.g. CSV files, which are registered and controlled using the main configuration file (YAML format). This may be seen as limitation for other developers, which is why many tools define a "no-code-layer" on top of their initial/main code-based interface. However, without a sufficiently large team, and considerable resources dedicated towards project management, it may become increasingly hard to replicate each new functionality of the code-based interface also in the "no-code-layer", which often creates a "second-class citizenship" for the "no-code-layer". The deliberate decision to force any new functionality to be implemented in a no-code interface, prevents these issues.

To still support and encourage complexity and flexibility in a user's modeling approach, IESopt.jl follows a very high level of abstraction, requiring a single configuration file as its only mandatory input. Further, it does not implement common energy system assets directly, but specifies abstract *Core Components* that can be configured and/or combined into, e.g., power plants, heatpumps, or grid connections.

This section gives an overview about the main software- and model-architecture decisions. The terms *Carrier* and *Snapshot* are used to refer to "energy carrier" (e.g., electricity) and "temporal snapshot" (a specific point in time for which temporal quantities - e.g., demand - are known; the set of all Snapshots relates to the set T of all timesteps $t \in T$ over which the model is formulated).

1) Core Component: Five so-called *Core Components* are defined and implemented. These translate various configuration options into actual mathematical formulations as part of the optimization problem:

- **Node**: A single abstract "point of contact", bound to a specific Carrier. In its base configuration a Node only encapsulates a formulation of a "nodal balance equation": For each Snapshot, the total supply (feed-in) at a Node has to equal the total demand (withdrawal). Besides more complex variations of this (equality) constraint, it can however be configured to be *stateful*: A stateful Node constructs an additional state variable, that – combined with inter-temporal constraints linking consecutive[3] states – allows relaxing the nodal balance to shift energy between Snapshots, thereby enabling an abstract form of an energy storage.
- **Profile**: The general assumption of the core formulations do not explicitly allow the "creation" or "destruction" of energy - existing energy can only be converted (c.f. Unit) between different Carriers, or transported (c.f. Connection) to other locations (c.f. Node). This entails that initially no energy can "enter" or "leave" the model's scope, a limitation that is prevented through the use of Profiles: During each Snapshot, a basic Profile feeds a predefined amount of energy, of a specific Carrier, into a single Node (or withdraws from it), which can be used to model an exogenous supply (e.g., renewables) or demand (e.g., hourly electricity demand). Profiles can then be extended to represent other externalities without pre-defined quantities, e.g., buying natural gas from the market, or taxing CO_2 emissions.
- **Connection**: Represents an energy exchange between two Nodes and can be bidirectional, incorporate losses, and may be restricted to variable or fixed bounds, which may be asymmetric, of the energy flow.
- **Unit**: Most of the interesting or complex functionality of energy system models is often bound to certain "assets" or "technologies" that can mostly be reduced to the abstract thought of energy (or material, commodity, etc.) conversion: An electrolyzer can be seen (in a simplified way) as converting electricity and water into hydrogen, heat, and oxygen. That kind of conversion process can be modelled using a Unit. It allows various configurations that normally exist for such technologies, like availability, installed capacity, binary (or integral, or continuous) operation modes (which may, e.g., represent unit commitment

[3]Here *consecutive* does not always imply x_t and x_{t+1}, depending on whether or not representative Snapshots are used, and whether other constraints (like sliding window) are enabled that potentially link more than two states.

for power plants), costs per conversion or start-up, and varying efficiencies.

- **Decision**: All other Core Components operate (mostly) on a user-defined configuration and are limited in their interaction with each other. A Unit for example requires a defined "installed capacity". To also allow studying optimal designs of future energy systems, investment decisions - that, e.g., choose the installed capacity of a Unit model-endogenously - are the essential functionality represented by a Decision component. It features upper and lower bounds, as well as a diverse range of costs (continuous, fixed, piecewise-linear[4]), and can be plugged into most parameters of other Core Components. This allows re-using the formulations, no matter the target: Complex, linearized cost functions can be used with a single command to control the nominal power of a new power plant, or as chosen exploitation of a large-scale heat storage, or even the available capacity for maritime hydrogen transport.

These Core Components can be combined with each other to construct more complex technologies and assets. In combination with a set of parameters, these then form so-called *Templates*. A Template defines a new component type that can be used like any Core Component, with free choice of parameters, default values, and the possibility to arbitrarily nest and combine Templates within Templates[5].

2) Configuration Structure: The aforementioned Core Components and Templates are used to construct various assets of a given energy system. This is done based on a set of configuration files, which are structured in the following way (leaving out less important specific details):

- **.iesopt.yaml*, represents the main entry point of any model, and contains general settings (model and scenario name, number of Snapshots, solver choice and attributes, filepaths, ...), all registered Carriers[6], as well as components and parameters (either explicitly or as link to external files).
- **.iesoptparam.yaml*, contains a dictionary of globally available model parameters that can be set from the outside (or implement default values), which can be loaded from **.iesopt.yaml*.
- **.iesoptcomp.yaml*, specifies a Template, containing a user-defined (custom) component based on other components.

Similar to global parameters, all modeled components can initially be specified directly in the main YAML configuration file. While this may be an optimal way for small models, it can become hard to manage quite fast with a growing number of components. Therefore, tabular files (e.g., CSV) can be used instead, offering a fine-tuned control over how the model topology is structured (components can be freely "clustered", e.g., based on their type, by the user)[7].

C. Core Formulations

One of the main advantages of the previously sketched abstraction that the Core Components provide is the possibility to implement specialized mathematical formulations based on the configuration of the Core Components, that are then immediately available for all derived user-defined components. Since all modelled technologies are - after flattening potentially nested Templates - reduced to Core Components, this allows users that are not proficient with intricacies of mathematical modeling or the effect of different formulations on solver performance to easily implement new technologies without foregoing performance.

Using an intuitive example, a showcase of how the conversion of a Unit is impacted by activating "unit commitment"[8] (UC) is provided in the following. The example is based on a simplified gas turbine, consuming the Carrier *gas* and outputting the Carriers *elec* (short for electricity) and *co2*. The parameter η defines the efficiency when "converting" *gas* to *elec* (which is here, for simplicity, assumed to be constant across the whole operational range), p_{min} the minimum load, and ζ the emission factor. The nominal power p_{nom} limits the electricity output for a fixed size Unit, while p_{max} gives the maximum installable nominal power when accounting for investment.

In the following, an indication is given after each equation, specifying the type of element: *var* is implemented as variable definition (potentially with bounds), *exp* describes an internal expression which is not passed to the solver but used to simplify the creation of constraints, which are indicated by *con*. To increase readability, where applicable, the indication that a statement is constructed for all Snapshots, e.g., $\forall t \in T$, is not explicitly given[9].

1) Conversion without UC: Without any specific configuration, this Unit models a simple, and "dispatchable" conversion of the form: "$1\ gas \rightarrow \eta\ elec + \zeta\ co2$", which is the basic way that the conversion rule is defined in the YAML configuration (assuming that units for all Carriers are properly chosen). It constructs:

$$\mathbf{conv}_t \in [0, p_{nom}] \qquad (var) \tag{1}$$

The variable $\mathbf{conv}_t$ hereby encodes the "amount of conversion" during each Snapshot. Further, an expression is constructed for each in-/output ((3) and (4) are identical for the

[4]While other types of cost functions (quadratic, or general non-linear terms) are currently not made available in the configuration syntax, the framework is designed in a generalized way to also support these, and may be extended as soon as necessary.

[5]The Templates used in the later presented use-case are available at github.com/sstroemer/OSMSES2024.

[6]IESopt does not impose any assumptions about energy carriers (or commodities) being modeled; the user can register "usual" ones like *electricity* in the same way as more exotic ones like, e.g., sulphur dioxide (to track emissions).

[7]More complex functionality for expert users exists, but was removed here to adhere with length limits, consult github.com/sstroemer/OSMSES2024 for the long version.

[8]The formulations have been slightly simplified during the conversion from Julia code to equations, to better communicate the reasoning.

[9]Consult github.com/sstroemer/OSMSES2024 for the long version of this work, which includes additional information regarding native links of the presented formulations to Benders decomposition and stochastic optimization.

other formulations, and will not be repeated there since they do not cover any sort of functionality, and are only included for sake of completeness here):

$$\mathbf{out\_elec}_t := \mathbf{conv}_t \quad (exp) \quad (2)$$
$$\mathbf{in\_gas}_t := \mathbf{out\_elec}_t / \eta \quad (exp) \quad (3)$$
$$\mathbf{out\_co2}_t := \mathbf{in\_gas}_t \cdot \zeta \quad (exp) \quad (4)$$

The variable $\mathbf{conv}_t$ is normally bound to whatever in- or output is specified as "conversion limit" (the installed capacity, herein given as p_{nom}, here explicitly constraining the electricity output; all other in-/outputs are implicitly also constrained due to their fixed relation). However, this choice is entirely arbitrary and may be automatically changed, e.g., to prevent numerical issues. The expressions (given in (2)-(4)) are not required in the Unit itself, but are used to connect the in-/outputs properly to the respective Nodes.

2) Conversion with UC: A first extension to the Unit's functionality is activating unit commitment, resulting in:

$$\mathbf{conv}_t \in [0, p_{nom}] \quad (var) \quad (5)$$
$$\mathbf{is\_on}_t \in \{0, 1\} \quad (var) \quad (6)$$
$$\mathbf{conv}_t \leq \mathbf{is\_on}_t \cdot p_{nom} \quad (con) \quad (7)$$
$$\mathbf{conv}_t \geq \mathbf{is\_on}_t \cdot p_{min} \quad (con) \quad (8)$$
$$\mathbf{out\_elec}_t := \mathbf{conv}_t \quad (exp) \quad (9)$$

After adding the binary variable $\mathbf{is\_on}_t$, that encodes whether or not a Unit is "online", it is used to constrain $\mathbf{out\_elec}_t$ either to 0 (if the Unit is offline) or to the range between minimum load and the total nominal power.

However, a slight modification is possible, by defining $\mathbf{conv}_t$ as only the variable/dispatchable part of the overall conversion (everything above p_{min}):

$$\mathbf{conv}_t \in [0, p_{nom} - p_{min}] \quad (var) \quad (10)$$
$$\mathbf{is\_on}_t \in \{0, 1\} \quad (var) \quad (11)$$
$$\mathbf{conv}_t \leq \mathbf{is\_on}_t \cdot (p_{nom} - p_{min}) \quad (con) \quad (12)$$
$$\mathbf{out\_elec}_t := \mathbf{is\_on}_t \cdot p_{min} + \mathbf{conv}_t \quad (exp) \quad (13)$$

This makes the bounds on available conversion more implicit (ensuring that $\mathbf{out\_elec}_t \leq p_{nom}$ always holds), resulting in only a single constraint being constructed. However, it leads to $\mathbf{is\_on}_t$ explicitly occurring in the expression that encodes that electricity output of the Unit (see (13)). Depending on the choice of solver, and specifics of the full model, this may or may not positively impact solution times. While the formulations presented here pose an intuitive view of the approach, extensive literature exists on different unit commitment formulations [25][10].

[10]A further interesting extension of the formulation is related to endogenous investment with active unit commitment (using a tight big-M notation), which was removed here to adhere with length limits, consult github.com/sstroemer/OSMSES2024 for the long version.

III. Use-Case: Hydrogen Production

To showcase the application of IESopt, a small-scale use-case is given in this section[11]. It is comprised of:

- Renewable generation, wind and solar, each installable between 0 and 20 MW.
- A battery energy storage system, of up to 25 MWh linked to up to 2.5/7.5 MW of charge/discharge power.
- Either a proton exchange membrane electrolyzer (PEMEL) or alkaline water electrolyzer (AEL) that feeds into a local hydrogen storage, limited to 185 MWh (which is roughly the average daily demand of the connected industry), with a loading/unloading time of eight/four hours, respectively.
- The possibility to substitute up to 10% of the annual hydrogen demand that needs to be covered with natural gas.

Further, the system is located near an existing grid connection with a limit of 20 MW. The solar PV, battery storage, and electrolyzer moreover require AC/DC inverters, as the local grid operates in AC.

A. Data

The input data is mostly comprised of historical time series and available technical information from commonly used sources and applicable literature. Cost data is based on "real 2020 euros", while 2030 is chosen as the modeling year. It is carefully selected to ensure consistency, but may not fully represent the latest domain-specific expert projections on future developments of single technologies. Even so, its purpose is first and foremost intended as an intuitive showcase of the framework's application.

Inflation adjusted day-ahead prices, and renewable availability profiles (based on generation data) for Austria during 2023 are used. Information from an Austrian distribution system operator (DSO) from 2020 is used to account for occurring grid fees, comprised of 35.55 EUR_{2020}/MWh for consumption, 8.27 EUR_{2020}/MWh for generation fed back into the grid, and 5051.18 EUR_{2020}/MW for monthly peak demand power from the grid (grid level 5). Prices for natural gas (used as substitution) are estimated at 26.91 EUR_{2020}/MWh. The industrial demand profile is based on the default configuration of [26], scaled to adjust the maximum peak demand to 10 MW. All power/energy related data, variables, and results are expressed in MW, respectively MWh. Where unambiguous, the specification of the carrier (e.g., $\text{MWh}_{hydrogen}$) is dropped. The nominal power of the electrolyzers is defined at the (electricity) input.

B. Setup

The model is solved over the course of a full year with a two-hourly resolution (to limit solve times of the MILP) and consists in the base version of roughly 100k variables and 225k

[11]The full model configuration, including all necessary input data, and sources for all parameter choices, can be found at: github.com/sstroemer/OSMSES2024

constraints. All runs are executed on a standard laptop using a i7-1185G7 processor running at 3.00 GHz, utilizing up to eight threads (on four physical cores). To showcase a more complete set of IESopt's functionality, some formulations are explicitly defined using an Addon: The power-related cost term of the grid connection and the unit commitment (both implemented in the core formulation), as well as the limitation on total number of cycles for the battery storage and the operational hours of the electrolyzer (both are currently not part of the core formulation) are all manually added into the overall model.

C. First Results

First, the model is executed as-is, based on the developed configuration files, and optimizing total system costs. Optimal investment slightly favors utilizing a PEMEL, resulting in (converted from EUR/MWh) levelized costs of hydrogen (LCOH) of 6.57 EUR/kg, while an AEL results in 6.62 EUR/kg. Wind and solar PV are fully (20 MW) installed for both technologies, but no battery is used. Further, PEMEL results in slightly more installed nominal power (18.3 MW versus 17.6 MW for AEL) and utilizes a larger hydrogen storage (95.7 MWh versus 84 MWh for AEL). Meanwhile, the PEMEL's annual operating hours (4225 hours) are slightly below the AEL's at 4367 hours.

These differences are all in line with the configured technological parameters: The PEMEL is able to run at as low as 5% of its installed nominal power, while the AEL has to respect a 20% boundary. This entails a more flexible operation of the PEMEL (and less operational hours), requiring slightly more nominal power, which results in more exchange with the external grid, and the requirement of a larger storage to smooth the more volatile production in alignment with the demand. With similar total costs and comparable efficiencies, the AEL's only advantage, namely higher total operational hours, could thus not be exploited (the enforced limit on operational hours was not binding). A more constant (e.g., more wind potential) and cheap (e.g., not directly coupled to spot prices) electricity supply could, however, change that.

Even so, the results clearly motivate the following section: The comparison of PEMEL and AEL illustrates that while an optimal decision is evident, two results can exhibit almost identical objective values, yet still differ structurally to a significant degree. The small differences in the objective values could presumably be easily eliminated due to uncertainty in economic (e.g., technology costs), technological (e.g., exact efficiencies of electrolyzers), and scenario-related (e.g., weather profiles) model parameters. Consequently, it may be of high interest to investigate alternative solutions that either:

1) Diversify the initial question: While total system costs and resulting LCOH may be of high importance, other factors like emissions, or impact on the power grid may represent crucial aspects.
2) Better understand structural characteristics of solutions (decisions) that are almost identical, resulting in a range of recommendations instead of a single "optimal" point.
3) Directly account for uncertainties in various input data, by not imposing perfect foresight of the future and respecting the non-deterministic nature of modeling future systems.

While (3.) can be tackled using stochastic optimization approaches (IESopt.jl plays nicely with either two-stage Benders based stochastic optimization or stochastic dual dynamic programming based on SDDP.jl [27]), the following results showcase how (1.) can be approached using a multi-objective (MO, e.g., utilized in [28]) approach [12].

D. Multi-Objective Application

Without any additional configuration, IESopt.jl by default builds a single objective function, called *total_cost*, which contains all cost terms that are added by the components of the model. A user can further specify custom objectives, comprised of variables or expressions (or their combination) available in the model. In the chosen use-case, an additional objective *grid_connection_power*, abbreviated by **gcp** is constructed, given as:

$$\mathbf{grid}_{cap,m} \geq \mathbf{grid}_{flow,buy,t} \qquad \forall m \in M, t \in T_m \quad (14)$$

$$\mathbf{gcp} := 5051.18 \cdot \sum_{m \in M} \mathbf{grid}_{cap,m} \quad (15)$$

Here, (14) constrains the utilized grid power ($\mathbf{grid}_{cap,m}$) of each month $m \in M := \{1, \ldots, 12\}$ to be greater or equal to all occurring consumption (= "from the grid") flows $\mathbf{grid}_{flow,buy,t}$ across all Snapshots of month ($t \in T_m$). Then, the objective grid_connection_power (gcp) is calculated as sum over all twelve months in (15), accounting for the grid fee of 5051.18 EUR/MW. Since this objective will be minimized, (14) is always binding as equality.

The most simple way would be to now scalarize these two objectives into one, by minimizing $\alpha \cdot \mathbf{total\_cost} + \beta \cdot \mathbf{gcp}$. However, this requires a prior choice of the weighting factors, which may not be doable in an informed fashion. Instead, a MO algorithm (like *EpsilonConstraint* as defined in MultiObjectiveAlgorithms.jl) can be activated in the main configuration of IESopt. It first minimizes by **total_cost**, which results in an objective value obj_1, then by **gcp**, which results in a corresponding value obj_2 of **total_cost**, evaluated at the second optimal solution point. The range $[obj_1, obj_2]$ now contains all total costs between an optimal solution based on **total_cost** and one based on **gcp**. This range is now split into multiple sections (e.g., equidistant). For each section's end-point c, the model is extended by an additional constraint

$$\mathbf{total\_cost} \leq c \quad (16)$$

and then solved by minimizing only **gcp**. Further repeating this for each section, this results in a list of points, each related to the minimal grid connection power that can be achieved, while

[12](2.) can be – with a similar algorithmic approach – investigated as part of the topic of modeling-to-generate-alternatives, which is contained as additional section in the long version, which is available at github.com/sstroemer/OSMSES2024.

staying below a predefined total system cost, which directly relates to a pareto-frontier, shown in Figure 1. As evident,

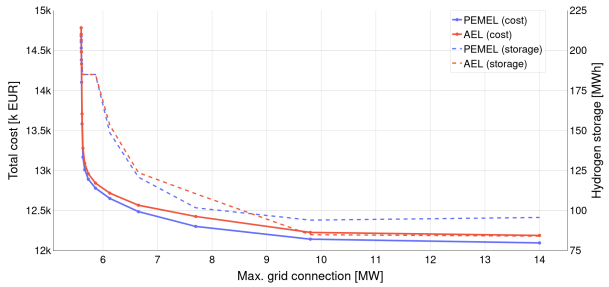

Fig. 1: Course of total system costs, shown over the utilized maximum (over the full year) grid connection power, comparing PEMEL (blue) and AEL (red) systems. The related optimal hydrogen storage size is given (dashed).

extremely limited grid connection utilization leads to higher system costs, as well as a close match of PEMEL and AEL results. Starting with at least 7 MW, the grid offers enough flexibility to properly utilize the wider range of operation that the PEMEL offers. At the same time, the optimal hydrogen storage size – initially offsetting the missing flexibility of the grid – roughly halves from 200 MWh to around 100 MWh for both PEMEL and AEL.

IV. Conclusion

The presented work motivates the necessity of highly performant and detailed mathematical optimization models for energy system analyses. IESopt, based on a core model (IESopt.jl) written entirely in Julia and rounded off with an easy-to-use Python wrapper, allows energy system modelers from diverse backgrounds to construct sophisticated models, without any prerequisites regarding mathematical optimization, specific coding languages, or complex workflows. IESopt will undergo significant and crucial improvements to complete the transition from a closed-source to a fully open-source project. Continuous development efforts will expand the feature set further, with a strong emphasis on algorithmic enhancements and improved built-in interfaces to existing large open-source datasets.

References

[1] M. Victoria, E. Zeyen, and T. Brown, "Speed of technological transformations required in Europe to achieve different climate goals," *Joule*, vol. 6, no. 5, pp. 1066–1086, May 2022.

[2] "Technology catalogues," Aug. 2016. [Online]. Available: https://ens.dk/en/our-services/technology-catalogues

[3] H. Formayer, I. Nadeem, D. Leidinger, P. Maier, F. Schöniger, D. Suna, G. Resch, G. Totschnig, and F. Lehner, "SECURES-Met: A European meteorological data set suitable for electricity modelling applications," *Scientific Data*, vol. 10, no. 1, p. 590, Sep. 2023, publisher: Nature Publishing Group.

[4] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for Power System Analysis," *Journal of Open Research Software*, vol. 6, no. 1, Jan. 2018.

[5] S. Pfenninger and B. Pickering, "Calliope: a multi-scale energy systems modelling framework," *Journal of Open Source Software*, vol. 3, no. 29, p. 825, Sep. 2018.

[6] H. C. Gils, H. Gardian, M. Kittel, W.-P. Schill, A. Zerrahn, A. Murmann, J. Launer, A. Fehler, F. Gaumnitz, J. van Ouwerkerk, C. Bußar, J. Mikurda, L. Torralba-Díaz, T. Janßen, and C. Krüger, "Modeling flexibility in energy systems — comparison of power sector models based on simplified test cases," *Renewable and Sustainable Energy Reviews*, vol. 158, p. 111995, Apr. 2022.

[7] M. Groissböck, "Are open source energy system optimization tools mature enough for serious use?" *Renewable and Sustainable Energy Reviews*, vol. 102, pp. 234–248, Mar. 2019.

[8] M. M. Frysztacki, V. Hagenmeyer, and T. Brown, "Inverse methods: How feasible are spatially low-resolved capacity expansion modelling results when disaggregated at high spatial resolution?" *Energy*, vol. 281, p. 128133, Oct. 2023.

[9] M. Bröchin, B. Pickering, T. Tröndle, and S. Pfenninger, "Harder, better, faster, stronger: understanding and improving the tractability of large energy system models," Nov. 2022, arXiv:2211.12299 [physics].

[10] F. Hofmann, "Linopy: Linear optimization with n-dimensional labeled variables," *Journal of Open Source Software*, vol. 8, no. 84, p. 4823, Apr. 2023.

[11] Y. Yang, "metab0t/PyOptInterface," May 2024, original-date: 2023-06-13T16:11:33Z. [Online]. Available: https://github.com/metab0t/PyOptInterface

[12] M. Lubin, O. Dowson, J. D. Garcia, J. Huchette, B. Legat, and J. P. Vielma, "JuMP 1.0: recent improvements to a modeling language for mathematical optimization," *Mathematical Programming Computation*, vol. 15, no. 3, pp. 581–589, Sep. 2023.

[13] M. Ihlemann, I. Kouveliotis-Lysikatos, J. Huang, J. Dillon, C. O'Dwyer, T. Rasku, M. Marin, K. Poncelet, and J. Kiviluoma, "SpineOpt: A flexible open-source energy system modelling framework," *Energy Strategy Reviews*, vol. 43, p. 100902, Sep. 2022.

[14] D. A. Tejada-Arango, G. Morales-España, L. Clisby, N. Wang, A. Soares Siqueira, S. Ali, L. Soucasse, and G. Neustroev, "Tulipa Energy Model," Mar. 2024. [Online]. Available: https://zenodo.org/records/10895406

[15] "HyTechonomy." [Online]. Available: https://projekte.ffg.at/projekt/3915332

[16] "H2Factory." [Online]. Available: https://projekte.ffg.at/projekt/4468737

[17] "Plan4.Energy - Methodenset für die Planungsbegleitung der Plusenergiequartiere." [Online]. Available: https://nachhaltigwirtschaften.at/de/sdz/projekte/plan4-energy.php

[18] www.ait.ac.at, "H2Real - AIT Austrian Institute Of Technology." [Online]. Available: https://www.ait.ac.at/en/research-topics/integrated-energy-systems/projects/h2real

[19] I. Mariuzzo, B. Fina, S. Stroemer, and M. Raugi, "Economic assessment of multiple energy community participation," *Applied Energy*, vol. 353, p. 122060, Jan. 2024.

[20] S. Reuter, S. Strömer, M. Traninger, and A. Beck, "Optimizing the Domestic Production and Infrastructure for Green Hydrogen in Austria for 2030: 9th International Conference on Smart Energy Systems," *Book of Abstracts: 9th International Conference on Smart Energy Systems*, pp. 278–279, Sep. 2023, place: Copenhagen. [Online]. Available: https://smartenergysystems.eu/

[21] P. Ortmann, S. Reuter, and S. Strömer, "Entwicklung eines globalen Marktmodells für Wasserstoff."

[22] "How JuMP enables abstract energy system models JuliaCon 2023." [Online]. Available: https://pretalx.com/juliacon2023/talk/C7DFSF/

[23] The pandas development team, "pandas-dev/pandas: Pandas," May 2024. [Online]. Available: https://github.com/pandas-dev/pandas

[24] "JuliaPy/PythonCall.jl," May 2024, original-date: 2020-11-03T16:11:11Z. [Online]. Available: https://github.com/JuliaPy/PythonCall.jl

[25] D. A. Tejada-Arango, S. Lumbreras, P. Sánchez-Martín, and A. Ramos, "Which Unit-Commitment Formulation is Best? A Comparison Framework," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 2926–2936, Jul. 2020, conference Name: IEEE Transactions on Power Systems.

[26] "Energy Transition Model." [Online]. Available: https://energytransitionmodel.com/

[27] O. Dowson and L. Kapelevich, "SDDP.jl: A Julia Package for Stochastic Dual Dynamic Programming," *INFORMS Journal on Computing*, vol. 33, no. 1, pp. 27–33, Jan. 2021, publisher: INFORMS.

[28] J. Finke and V. Bertsch, "Implementing a highly adaptable method for the multi-objective optimisation of energy systems," *Applied Energy*, vol. 332, p. 120521, Feb. 2023.