RESEARCH

Open Access

An adapter-based architecture for evaluating candidate solutions in energy system scheduling

Malte Chlosta*, Jianlei Liu, Rafael Poppenborg, Richard Lutz, Kevin Förderer, Thorsten Schlachter and Veit Hagenmeyer

From Energy Informatics.Academy Conference 2022 (EI.A 2022)
Vejle, Denmark. 24-25 August 2022

*Correspondence:
Malte.Chlosta@kit.edu

Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen, Germany
Full list of author information is available at the end of the article

Abstract

Increasing shares of volatile generation and non-steerable demand raise the need for automated control of the Energy Systems (ESs). Various solutions for management and schedule-based control of energy facilities exist today. However, the amount and diversity of applications lead to a multitude of different automated energy management solutions. Different optimization algorithms have proven more or less effective for energy management. The multitude of optimization algorithms and energy management solutions require flexible, modular, and scalable integrations. We present a novel Optimization Service (OS) for easily integrating optimization algorithms while evaluating candidate solutions in the context of ESs applications. We propose an adapter-based architecture using metadata and domain knowledge to bridge between clients, e.g. smart grid applications and optimization algorithms. The architecture interfaces different clients with optimizers in a flexible and modular way. The clients provide metadata-based descriptions of optimization jobs translated by OS. OS then interacts with optimizers and evaluates candidate solutions. A consistent definition of interfaces for clients and optimization algorithms facilitates the modular evaluation of candidate solutions. OS's separation of client and optimization algorithms increases scalability by managing computational resources independently. We evaluate the presented architecture for scheduling a so-called Energy Hub (EH) as a test case describing a simulation scenario of a renewable EH embedded in grid scenarios from an industrial area in Karlsruhe, Germany. OS utilizes an Evolutionary Algorithm (EA) to optimize schedules for cost and strain on the electrical grid. The use case exemplifies OS's advantages in a proof-of-concept evaluation.

Keywords: Energy management systems, Optimization, Modularity, Flexibility, Smart energy system, Smart grid

© The Author(s) 2022. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Introduction

In order to limit global warming, the European Union (EU) (UN. BRUSSELS EUROPEAN COUNCIL 2015) and the German government formulated the long-term objective of reaching climate neutrality by 2045. Moreover, Germany aims to produce 100% of its electricity from Renewable Energy Sources (RESs) in 2050 (Klaus et al. 2010). High penetration of RESs leads to challenges in grid operation, energy security, and environmental sustainability. Challenges include the reduction of grid inertia (Fernández-Guillamón et al. 2019), shortage in high voltage line capacity (Moriarty and Honnery 2016), and RES volatility. One fundamental approach to cope with the rising challenges of volatile RES is to use the flexibility provided by the rising number of Distributed Energy Resources (DERs), such as combined heat and power plants, battery energy storage systems, and heat pumps. In this context, flexibility describes the ability of a DER to adapt the amount of consumed and provided power (Mauser et al. 2017). DERs often couple different sectors like electricity and heat in the case of heat pumps, or electricity and mobility in the case of electric vehicle supply equipment. Large amounts of heterogeneous DERs penetrate smart grids resulting in optimization problems that consider several sectors simultaneously. Moreover, in light of variations in energy demand and supply caused by weather and hard-to-predict human behavior, the coordination of these decentralized systems requires advanced energy management, scheduling, and control efforts. Especially for scheduling complex DERs, non-linear, multi-objective optimization problems with complex Boundary Conditions (BCs) arise.

Researchers have proposed a variety of approaches for the optimized control of DERs and smart- or micro-grids. Firstly, several Energy Management System (EMS) solutions are available, such as openEMS[1] or Organic Smart Home (Allerding 2014), which implement various features, including communication standards, data model standards or data visualization. Secondly, the literature proposes approaches and usually focuses on Energy System (ES) description, mathematical formulation of vast sets of BCs, or the general impact of optimized schedules on a given ES. A fixed optimization model is usually closely coupled to a specific solver or optimization algorithm in both cases. As stated by the "no free lunch theorems of optimization", there is no generally best-performing optimization algorithm (Wolpert and Macready 1997). Different algorithms (on average) perform better for different problem classes and structures (Wolpert and Macready 1997). Different applications call for different optimization algorithms for smart grid applications, ranging from simple monitoring tasks to complex market processes and scheduling tasks. Moreover, different problem instances may benefit (on average) from selecting different algorithms as search space; therefore, the problem structure differs. However, algorithm-specific requirements and labor-intensive implementation generally hinder switching between algorithms. With the help of a novel OS, we aim to alleviate this issue and simplify the switching of optimization algorithms for smart grid applications.

The present paper proposes an architectural design for the modular and flexible integration of diverse optimization solutions. The task of scheduling resources in smart

[1] https://openems.io/.

grids based on (meta-)data descriptions of the controllable resources functions as the first proof of concept. The architecture has an adapter-based design that interfaces with clients, such as an EMS, energy market platform, or any other applications that require optimized schedules and optimization algorithms. The architecture is especially suited to, but not limited to, the application of metaheuristics. In order to utilize metaheuristics, proposed solutions need evaluation and the return of objective value(s). If necessary, the proposed architecture automatically takes care of these steps and the interaction with the optimizer. Aside from switching between algorithms, modular integration allows for the parallel use of multiple algorithms. Future work might utilize flexibility regarding different algorithms for automated assessment and selection of the best available algorithm. Moreover, the novel approach decouples optimization tasks from specific computational resources. With the proposed architecture, it is possible to implement resource management to distribute the computational burden and use the available computational resources more efficiently.

The present paper is structured as follows: In the next section, we give an overview of the current state of the art regarding optimization in energy management. This is followed by the presentation of OS concept and architecture. Subsequently, the implemented use case, an example for the utilization of an EH (see also Poppenborg et al. 2021), and the results of using the proposed OS for this proof-of-concept are presented and discussed. Finally, we summarize the results and conclude the paper with an outlook.

State of the art

There is a vast amount of literature of the operational optimization of ESs with descriptions of different modeling techniques, optimization algorithms, and detailed analysis of the results. Most of the literature refers to explicit compositions of ESs. Following is a brief overview on research regarding use case-oriented and use case-independent setups.

Use case-oriented setups

The authors of Niknam et al. (2013) present a novel probabilistic approach to handling uncertainty in the energy management problem. Probabilistic approaches find the possible variation of the output caused by variation of the uncertain inputs and show which values of the output within the range are most likely. Also, in Cao et al. (2020), a robust optimization approach based on a modified grey wolf optimizer is proposed to determine the optimal energy management for a typical micro-grid concerning uncertainties. Zhang et al. (2013) introduce energy management to deal with the challenging constraint of the supply-demand balance raised by the intermittent nature of RES in a grid-connected microgrid. Obara et al. (2013) present a system that describes a microgrid for a particular use case, “the Lake Saroma (Hokkaido)”. Ignat et al. (2018) describe a microgrid consisting of several renewable energy sources, one energy storage system, and loads. It uses the Particle Swarm Optimization Algorithm to determine the optimal operation of the microgrid’s solar, geothermal, and biomass units while optimizing cost. Marzband et al. (2014) show an EMS based on a multi-period gravitational search optimization algorithm to solve such problem EMS in a microgrid including different types of distributed generation units, with particular attention to technical constraints. Elsied et al. (2015) present an EMS in a typical microgrid working in grid-connected mode. They formulate the established EMS as a non-linear optimization model with different equality and inequality constraints for proper solutions based on the AIMMS (Advanced Integrated Multidimensional Modeling Software). Regarding optimization algorithms, Fathima and Palanisamy (2015) state that (meta)heuristic optimization approaches have proven effective for scheduling hybrid RESs.

Other research focuses on more generic approaches by developing systems that, once initiated, can manage different compositions of ESs.

Use case-independent setups

Ayoub et al. (2018) propose one complex system of four main parts: energy semantic network databases; a simulation and optimization module; a risk-based LCA/LCC Module; and a user interface. As a two-level (detailed design level and client demonstration level) management system, it supports energy conservation options in buildings. Ingo Mauser has described a Building EMS in his dissertation (Mauser 2017). The proposed Building EMS realizes modular energy management and operation optimization of devices and systems in real-world and simulated buildings. The multicommodity optimization integrates into the Building EMS for multi-modal energy management. He provides an extensive foundation regarding heuristic optimization for Building facility scheduling with a modular simulation back end. Mauser (2017) presents architectural concepts of a building EMS couple optimization and simulation into a single system. Similar to other approaches in the literature, he combines optimization algorithm (EA), management tasks like distribution of schedules, and interpretation of optimization solutions (Energy Simulation Core) in one modular but a combined system (Mauser 2017; Mauser et al. 2016).

Conclusion

All the approaches described above focus only on modeling different EMSs or optimization methods to solve specific problems in microgrids. Some of them are limited to a particular scenario. They describe many implementation details for building energy management or optimization models, such as mathematical formulas and program code. Integration tests consisting of energy management and optimizer in a microgrid evaluated the models. However, none of them presents the definition of the interfaces between EMS and optimizer, namely how EMS and optimizer communicate with each other. Their EMS and optimizer are strongly coupled, have low cohesion, and are often not modular. Therefore, it is complex and time-consuming to test and maintain them, use them for other scenarios, and to scale them for new requirements. Van Beuzekom et al. (2015) concluded from 72 (13 more detailed) papers on combined planning and operations optimization that three main challenges arise:

(1) finding the right level of constraint detail while maintaining computational feasibility;
(2) combining short-term dynamics with long-term effects;
(3) developing heuristics, terminology, and system boundaries.

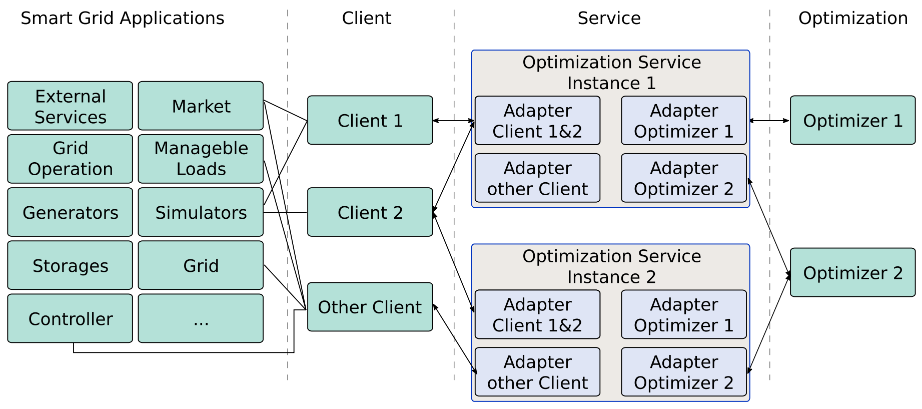

Fig. 1 System overview of an adapter-based OS

They conclude that no single tool exists to fit all requirements and more research needs to be done on combining different optimization tools and approaches. An extensive review (Fathima and Palanisamy 2015) concludes that there is a vast amount of tools for optimization in hybrid RES, but the flexibility of these tools needs to be increased, especially for control and energy management. Framinan and Ruiz (2009) and Framinan and Ruiz (2010) analyzed the literature on manufactory scheduling and concluded that there is an abundance of different scheduling models and solution procedures but a lack of work on practical implementation.

We conclude that the current literature leaves a gap for flexible and modular scheduling in ESs. A novel approach should be readily applicable to actual use cases.

Therefore, the presented article provides a novel optimization service as a communication interface between EMS and optimization solutions for separating them as independent components. The two parts are modular and have low coupling. The generic interfaces are defined to integrate different EMSs or optimization solutions easily using an adapter-based design. Therefore, the architecture of this optimization service can benefit the design of ESs by improving

- modularity: Clients, optimization algorithms, and evaluation methods should be independently replaceable;
- scalability: Architecture should allow scaling of computationally intensive parts of the system (Client, Optimization Service (OS), or optimization algorithms independently);
- flexibility: Developed architecture must apply to different client combinations, system configurations, and optimization services.

Concept

Figure 1 illustrates the central concept of OS as a bridge between the client and the optimization algorithm. EMS is an example for clients with access to various smart grid applications. For scheduling, clients need access to optimization algorithms and metadata-based evaluation of optimization suggestions. Based on information from the smart grid, a client formulates an optimization problem for OS. One adapter specification

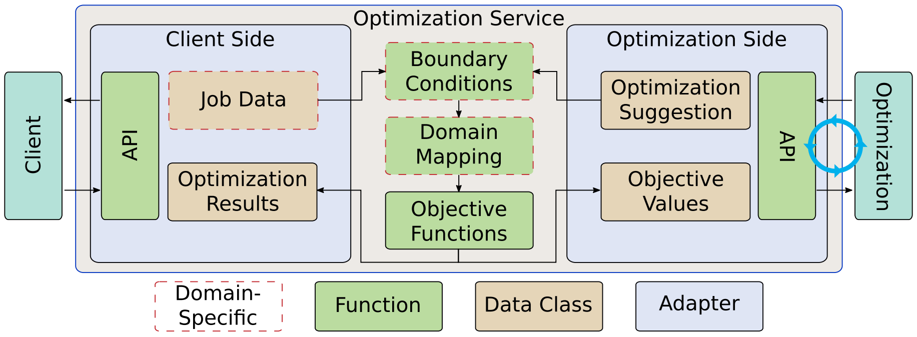

Fig. 2 Overview of OS architecture with a focus on Client and Optimization Side Adapter

implements a new client type that communicates from OS to the client and vice versa. Implementation of optimization algorithms functions works in the same way. Each new type of optimization algorithm needs a new adapter. Once implemented, the optimization algorithm is accessible to all clients.

The architecture provides flexibility for the whole system in the sense that

- OS receives jobs from multiple clients of the same or different kind(s), such as that multiple clients can use a single OS instance;
- OS communicates the optimization problem to multiple optimization algorithms of the same or different kind(s);
- optimization jobs from one client can be split into several OS's,
- or one optimization algorithm splits up suggestions to multiple OS instances.

Architecture

OS mediates between the optimization and client sides by implementing one adapter for each side. Fig. 2 illustrates this concept. The adapter is color-coded in light blue. Each adapter consists of functions (green) representing the Application Programming Interface (API) and data classes (brown) acting as data interfaces. The presented architecture is generic. Any domain-specific implementations are bordered in red and implement logic from the ESs domain. APIs translate information between internal data classes and external clients/optimization algorithms. The job data and optimization suggestion data classes provide the data interface to calculate the input for the objective values and optimization results data classes.

The API consists of several functions handling mapping and communication on the client side. The job data class holds domain-specific knowledge about ESs and acts as a data interface. The internal evaluation is domain-specific and interacts only with data interfaces as input and output. OS passes the final solution to the optimization job to the optimization results data class.

On the optimization side, the API consists of the same functions for communication as on the client side. n-D arrays represent optimization suggestions generically. Specifically for the domain of scheduling energy management, these n-D arrays can be interpreted as setpoints (floats), schedule (list of setpoints), schedule set (list of schedules, one for each facility), and list of schedule sets. Three sets of functions inside OS evaluate

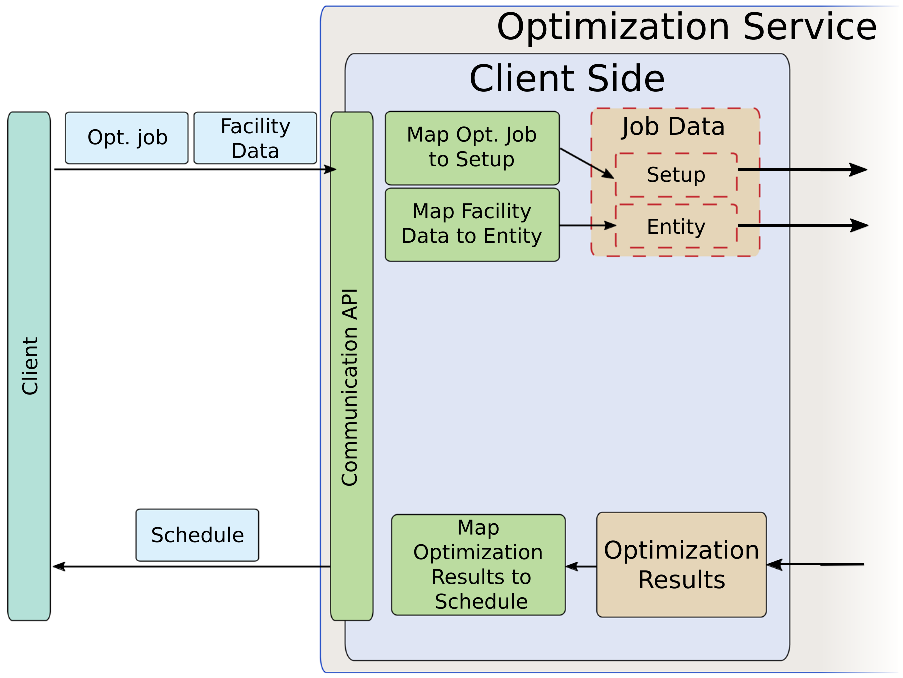

Fig. 3 Concept of the client side adapter with different mapping functions

each instance of an optimization suggestion data class. The evaluation returns so-called objective values that initiate the objective values data class instances.

Three sets of functions subdivide the evaluation into BCs, domain mapping, and objective functions. The sets of functions rely on specified data classes to provide fixed and pre-defined interfaces. Interfaces allow the implementation of domain-specific BCs while remaining independent of new adapters.

Adapters and Interfaces

The client (Fig. 3) and optimization adapters (Fig. 4) enable communication with external services. The combination of communication API, mapping functions, and data interfaces yields adapters. The communication API consists of functions that actively pull data from a communication service (e.g., Redis) or passively receive data from external services (e.g., HTTP requests). The communication API then passes messages to respective internal mapping functions. Mappings function as a translator between internal and external data standards.

The client adapter handles three types of messages by implementing respective mapping functions. For clarity, external data names differ from internal data classes. The client has to separate the optimization problem into two message objects: Job containing the optimization problem's metadata and Facility Data containing descriptive data about the facilities that are part of the optimization problem. When OS conducts a result to the optimization problem, OS returns schedules to the client.

Three functions map the incoming messages to the internal data classes. The Opt. Job creates one single instance of setup, a subclass of job data. Each facility in the facility data message initiates one instance of the entity subclass of job data. Both subclasses

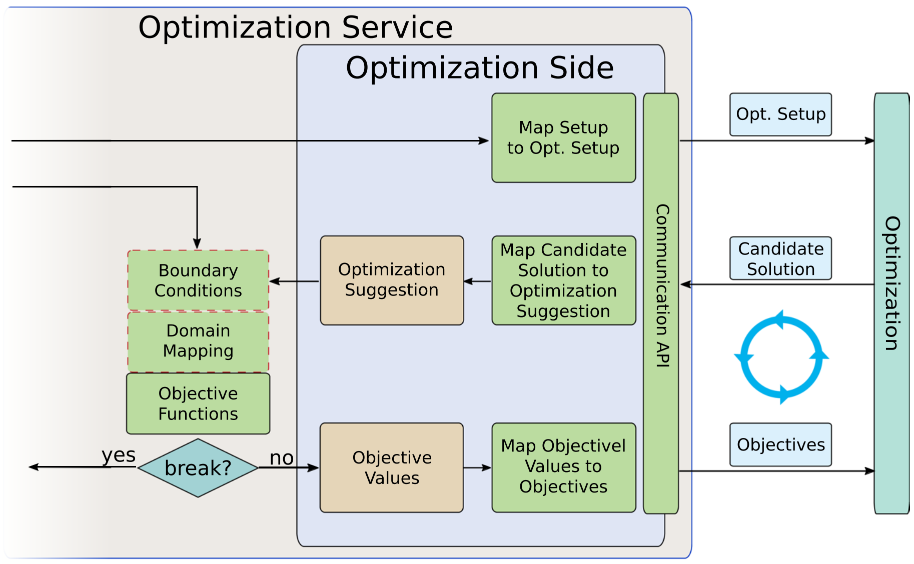

Fig. 4 Concept of the optimization side adapter with different mapping functions

are combined to describe one complete optimization problem. The result of the optimization problem contains a schedule for each entity. The set of schedules makes one instance of the optimization results data class. The respective mapping function must mediate between an internal entity (id and name-based) references and external (typically name-based) references.

The optimization adapter handles the initial optimization setup based on the initiated job data setup with the respective mapping function. Once initialized, the optimization algorithm responds with a candidate solution to the optimization problem. From OS point of view, these solutions are further referred to as an optimization suggestions that need further evaluation. A mapping function maps these to the optimization suggestion data class. Based on a break condition, the optimization suggestion is either returned as an optimization result to the client adapter or the optimization adapter as objective optimization values. The whole process repeats until the break condition indicates the final result.

Evaluation

The evaluation divides into BCs, domain mapping, and objective functions. The three parts are a short form of other optimization procedures in literature. Optimization suggestions are the input for BCs that are formulated to apply directly to these abstract outputs from the optimization algorithms. The optimization algorithms output the schedules in the example of ESs consisting of setpoints relative to maximum output. A BC, thereby, has to condition the setpoint. The input for each BC is an entity and individual part of the optimization suggestion data object. The novel architecture's implementation of BCs follows five simple steps guaranteeing a generic and expandable application.

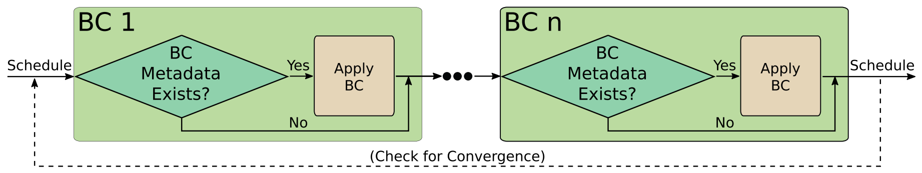

Fig. 5 Metadata-based BCs in a pipeline

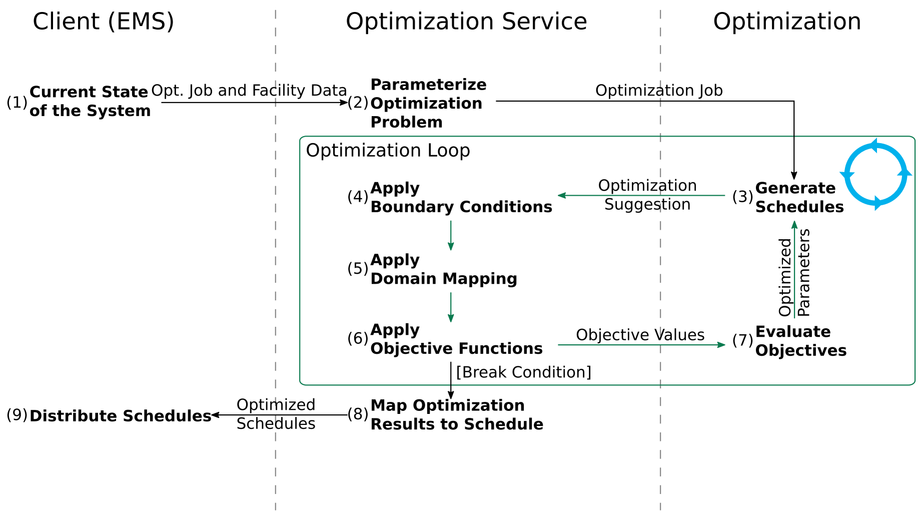

Fig. 6 Execution procedure of OS

1. BC checks if necessary metadata exists in the entity data object. If not, the function skips further calculation.
2. The conditions syntax directly applies to the optimization suggestion.
3. BCs formulate a statement. When the condition holds, the statement applies.
4. Both statement and condition apply to an entity's optimization suggestion.
5. The schedule is returned.

A pipeline is constructed when describing the system with many BCs (Fig. 5). The pipeline can be applied once or until it converges.

Once all BCs are applied to the optimization suggestion, the conditioned suggestions are mapped from the domain of optimization suggestions to the domain where evaluation takes place. In the example of ESs, optimization suggestions are in the information and communication technologies domain and mapped into the physical domain by calculating the entity's physical outputs based on setpoints. The input for the domain mapping functions are abstract suggestions like setpoints, and the output is relevant physical values like electric power.

A set of objective functions evaluates time-series from the physical domain. Typically, functions include cost calculations, Root Mean Square Error, or others.

Operation

Figure 6 shows a high-level overview of OS process, client interactions, and optimization algorithm interactions. The optimization process starts with an optimization job containing the current state of facilities and optimization setup variables. The client's information is sent as opt. job and facility data messages to OS. The client side adapter initializes instances of the job data class to parameterize the optimization problem and maps it to the optimization algorithms syntax via mapping functions. Resulting opt. setup is communicated to the optimization algorithm. Once initialized, the optimization algorithm iterative generates candidate solutions that OS evaluates. The results of the objective functions are returned as objective values to the optimization algorithm, which then evaluates the return and generates new candidate solutions until a break condition is met. The final optimization suggestion is mapped to optimization results and mediated to the client. The client can request the subsequent optimization by sending a new opt. job and new facility data.

Implementation

The presented paper implements three exemplary adapters to connect OS with

- a custom EMS client that implements OS in a co-simulation scenario,
- General Learning Evolutionary Algorithm and Method (GLEAM), an evolutionary optimization algorithm,
- and Monte Carlo, a random sampling algorithm for development purposes.

GLEAM is an evolutionary algorithm that has proven to be efficient in scheduling tasks. All adapters use Redis[2] as communication infrastructure.

Domain knowledge regarding ESs is implemented into the job data class, the BCs, and the domain mapping functions (compare Fig. 2), before implemented adapters interact. The implemented domain knowledge is based on Geidl and Andersson (2007)'s EH work. EH is one generic way of describing ESs.

Since EH is a generic concept for mathematical description, implemented domain knowledge does not restrict OS appliance to specific compositions of ESs, ensuring the system's flexibility.

Converters and storages are controllable units that condition, transform, or store energy, whereas forecasts (sometimes called generic/active consumer) are any non-controllable generations or loads. The combination of the three facility types allows a general description of any composition of ESs.

EH further allows the implementation of topological connections between facilities. The most basic topological arrangement of EH components is the connection of any energy carrier to a grid. The grid connections do not restrict the EH components in consumption or generation. Consequences of certain facilities' consumption or generation must be considered in the objective function, e.g., by a high price of energy carrier grid. The presented implementation uses these simplified topological assumptions.

Data interfaces

Implemented data interfaces are separated into four classes: job data, optimization suggestions, objective value, and optimization results (Compare Figs. 2 and 4).

[2] https://redis.io//.

The job data consists of two subclasses: setup and entity. The setup data interface is shown in Listing 1 and provides all data needed to initialize and set up the optimization process, including initialization parameters for the optimization problem, static information regarding the simulation system, or the number of objective values returned after evaluation.

- First, the number of different facilities *number_facilities* that expect schedules is described as an integer. The *power_fraction* represents the range for each set point. Typically the range is [0, 1] for converter units and $[-1, 1]$ for storage units. The *power_fraction* is implemented as a dictionary with the keys representing the names of the entities. *start_time* and *stop_time* are integers representing the internal time for the start and stop of the optimization.
- Next, the so-called sitting variables implement *interval_steps* as the number of set points that assemble one schedule. The *names_facilities* is a list of all facility names, including forecasts being not part of the optimization. All names used as keys in the *power_fraction* must be listed here. The difference between *stop_time* − *start_time* represents the optimization time range *calculation_time*. The *interval_time* describes the time between two optimization steps. In the current implementation, all set points have to be equidistant. Further research on non-equidistant ES scheduling is currently taking place based on the novel architecture.
- The third group consists of the objective functions and the constraints. Each objective function is given a representative name in a list *obj_functions*. The same applies to the *constraints*.

Listing 1 Setup data class (python oriented syntax).

```
class Setup:
    number_facilities:          int
    power_fraction:             Optional[Dict[str, List[float]]]
    start_time:                 int
    stop_time:                  int

    interval_steps:             int
    names_facilities:           List[str]
    calculation_time:           float
    interval_time:              float

    obj_functions:              Optional[List[str]] = []
    constraints:                Optional[List[str]] = []
```

The second subclass entity is the data class regarding data modeling of facilities and is shown in Listing 2. All later implemented BCs rely on the respective entity data objects. The parent class entity consists of a *name*, *controlling* that flags whether the entity can receive schedules from the optimization algorithm, the outputs that the relative facility has, and the Lookup Table (LUT) with a link for later domain mapping.

Based on EH work, entities are further divided into three subclasses implementing respective metadata for each. Converters are described by *P_min*, the minimal power, the maximal power *P_max*, the current power *P_0*, and the maximal power increase *RampRate*.

Storage units additionally use parameters for calculating the current energy of the facility. *E_max* is the maximum energy, the current energy *E_0*, the minimal energy *E_min*, and the efficiency term *Eta*.

Forecasts are not controllable facilities based purely on time-series data. As they receive no schedules, they do not need any metadata description.

Listing 2 Entity data class with sub classes Converter, Storage, and Forecast (python oriented syntax).

```
class Entity:
    name:                        str
    controlling:                 True
    Outputs:                     List[str]
    LUT:                         Optional[str]

class Converter(Entity):
    P_min:                       Optional[float]
    P_max:                       Optional[float]
    P_0:                         Optional[float]
    RampRate:                    Optional[float]

class Storage(Converter):
    E_max:                       float
    E_0:                         Optional[float]
    E_min:                       Optional[float] = 0.0
    Eta:                         Optional[float] = 1.0

class Forecast(Entity):
    controlling:                 False
```

An optimization suggestion contains one schedule per controllable instance. The evaluation of one optimization suggestion is stored in the goal value data class.

Based on previously discussed data interfaces, the actual evaluation of schedule suggestions takes place, starting with the BC.

Boundary conditions

For readability, the implemented BCs are presented in mathematical syntax. Exemplary consider the BC that conditions for the minimal power output of a converter unit. The power output $P^i_\alpha(t)$ of an entity i and an energy carrier α at a given time point t are described by the nominal setpoint $s^i(t)$ times the maximal power output. This power output has to be greater or equal to a minimal power $P^i_{\alpha,min}$. If not, the converter unit shuts down, producing an output of $0 \cdot P^i_{\alpha,max}$.

$$P^i_\alpha(t) = s^i(t) \cdot P^i_{\alpha,max} \geq P^i_{\alpha,min}. \tag{1}$$

For this explicit BC, two variables have to be provided in the metadata: $P^i_{\alpha,max}$ and $P^i_{\alpha,min}$. The actual condition can be implemented when both variables are presented in the converter instance. All boundaries are rearranged to mathematically condition the setpoint $s^i(t)$. The presented BC that reviews $P^i_{\alpha,min}$ is formulated as

$$s^i(t) = \begin{cases} 0, & \text{if } s^i(t) \leq \frac{P^i_{\alpha,min}}{P^i_{\alpha,max}} \\ s^i(t), & \text{otherwise.} \end{cases} \tag{2}$$

Additionally BC for P_{max}

$$s^i(t) = \begin{cases} \frac{P^i_{\alpha,max}}{|P^i_{\alpha,max}|}, & \text{if } s^i(t) \geq \frac{P^i_{\alpha,max}}{P^i_{\alpha,max}} \\ s^i(t), & \text{otherwise,} \end{cases} \tag{3}$$

and *RampRate*

$$s^i(t) = \begin{cases} s^i(t-\Delta t) + \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}}, & \text{if } s^i(t) \geq s^i(t-\Delta t) + \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}} \\ s^i(t-\Delta t) - \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}}, & \text{if } s^i(t) \leq s^i(t-\Delta t) - \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

are added to describe a converter facility. Storage units apply the same BC as converter, *P_min* though changes to

$$s^i(t) = \begin{cases} \frac{P^i_{\alpha,min}}{|P^i_{\alpha,max}|}, & \text{if } s^i(t) \leq \frac{P^i_{\alpha,min}}{|P^i_{\alpha,max}|} \\ s^i(t), & \text{otherwise.} \end{cases} \tag{5}$$

Additionally BC for energy of a storage facility E is added as

$$s^i(t) = \begin{cases} s^i(t-\Delta t) + \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}}, & \text{if } s^i(t) \geq s^i(t-\Delta t) + \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}} \\ s^i(t-\Delta t) - \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}}, & \text{if } s^i(t) \leq s^i(t-\Delta t) - \frac{\Delta P^i_{\alpha,max}}{P^i_{\alpha,max}} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Forecasts are not subject to any BCs. The principle of BCs leaves room for future applications of forecast calculations like uncertainty.

The mapping is currently separated into two functions:

- Interpolation based on LUT for all three types of entities, and
- integration of the setpoint to gain a state of charge of storage units.

Literature suggests other approaches that can be integrated as additional functions here.

Objective functions

Implemented Objective Functions are oriented on three of the most common functions in literature: minimization of 1) system operational cost, 2) emissions, and 3) deviation from original demand. For the ability to optimize multiple functions collectively, all functions are connected through a penalty variable. The penalty variable for each function is selected so that the resulting unit is priced in euros. In general, both system's operational cost and system emission minimizations are the same, as they punish an energy or mass flow with a price p_α resulting in the initialization of operational cost.

$$c_{operational} = \sum_t \sum_i \sum_\alpha P^i_\alpha(t) \cdot p_\alpha(t). \tag{7}$$

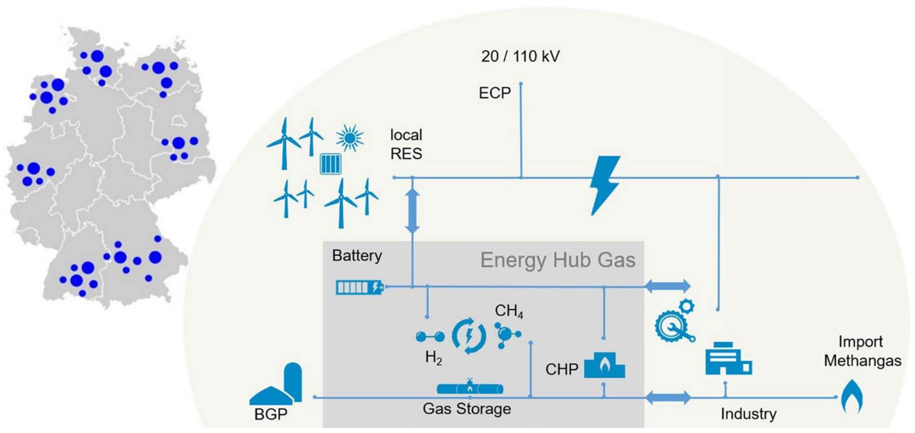

Fig. 7 Co-simulation setup of an EH in a gas and electricity grid scenario near Karlsruhe, Germany

It is common in the literature to separate the emission and operational costs.

The minimization of system operational costs is formulated as the sum over all time steps *t*, the entity's *i* and energy carriers α

$$\begin{aligned} c_{operational} &= \sum_t c_{operational}(t) \\ &= \sum_t \sum_i \sum_\alpha P^i_\alpha(t) \cdot p_\alpha(t), \end{aligned} \tag{8}$$

with $P^i_\alpha(t)$ the power flow of an entity i and an energy carrier α, and p_α the price of an energy carrier α.

In the same way, the total cost of emission is phrased as

$$\begin{aligned} c_{CO2} &= \sum_t c_{CO2}(t) \\ &= \sum_t \dot{m}_{CO2}(t) \cdot p_{CO2}(t). \end{aligned} \tag{9}$$

Last, the deviation from the original demand is integrated quadratic in order to account for the sign of P^i_α as

$$\begin{aligned} d_{dev} &= \sum_t \left(P^{dev}(t)\right)^2 \\ &= \sum_t \sum_i \sum_\alpha \left(P^i_\alpha(t)\right)^2. \end{aligned} \tag{10}$$

Results

The presented result showcases an exemplary proof of concept with one specific client and two implemented optimization algorithms. Results focus on an in-depth discussion of one use case rather than displaying multiple superficial implementations.

The EH concept was introduced by Geidel (2007) and describes a central unit that can transform, convert, and store energy carriers in decentralized plant networks. EH is a modular approach and provides flexibility for balancing local generation and demand, as stated by Mohammadi et al. (2017). The modular approach and the provision of flexibility make EH an excellent candidate to support the local integration of RES by relieving the grid infrastructure, as shown in Poppenborg et al. (2021).

The EH model, illustrated in Fig. 7, bi-directionally couples the electricity and gas sectors. Electrolysis with coupled methanation converts electricity to methane.

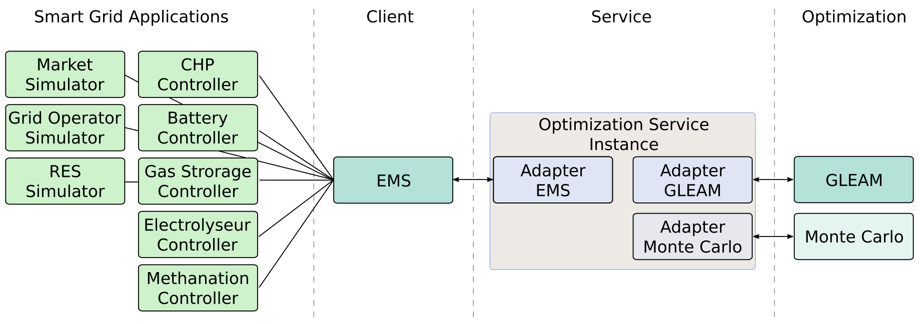

Fig. 8 Implemented setup of EH

Combined Heat and Power (CHP) converts methane to electricity. The grid operator sends schedules for the EH as one unit to the EMS based on the local RES generation and demand to operate the presented EH as a grid-supporting asset. Subsequently, the EMS has to fulfill the received schedule at its Electrical Connection Point (ECP) by aggregating included components. Providing a schedule for each facility formulates a complex optimization problem. In more detail, the EMS controls five facilities (battery, gas storage, CHP, electrolysis, methanation) while constantly receiving (meta)data on their current state. Schedule optimization follows two objective functions: (1) Follow the grid operator's target value at the electrical connection point; (2) Cost efficiency considering CO_2 emissions and operational costs. Objective functions require additional price information. The EMS receives price information as (predicted) time-series data from the market simulator (historical time-series data). The use case implementation considers the day-ahead spot market price information for 2021. Weather data for Karlsruhe, Germany, and the peak power of installed RES complete the scenario information. The considered RES accumulates up to 4MW. The grid operator simulation provides the resulting schedules for the entire EH by taking complete information from the scenario into account as a day-ahead prediction.

With the information on the five facilities, price signals, and total schedule, the EMS formulates the optimization problem for OS. The system setup from OS's point of view is represented in Fig. 8. In return, the EMS receives one schedule for each facility, respectively. These schedules are then sent to respective controllers, and the simulation proceeds. The EMS formulates optimization problems for 24 h with an interval of 15 min. Every 24 h, the EMS submits a new optimization problem to OS with current facility states. Over the simulation of one year, 365 optimization problems are formulated and solved by OS.

The EMS can switch between implemented optimization algorithms by configuration. Two optimization algorithms are implemented. The monte Carlo optimization process results are not visualized as they do not differ from random chance recognizably, even after long optimization runs. Following focuses on the simulation run that utilized GLEAM as an optimizer.

Figure 9 presents the simulation results, with the black line being the grid operator's target value that the EH has to fulfill. At the same time, the green dashed line represents the actual electric output of the EH at its connection point. This electrical

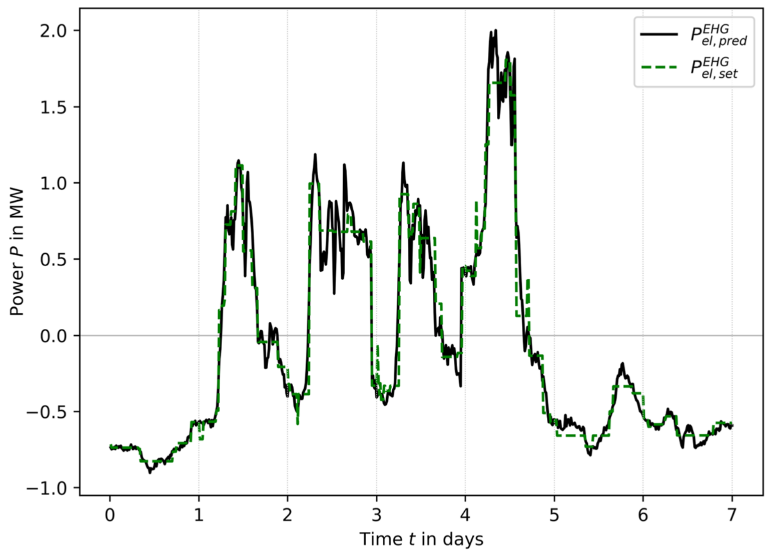

Fig. 9 Results of the optimization for a period of 7 days

output consists of the aggregated electrical power of the considered EH components. The obtained results for an exemplary period of 7 days show an excellent approximation between the target value and EH's effectual power output, as depicted in Fig. 9.

With the main computational effort of evaluating every schedule proposed by the optimization algorithm, OS provides the best results as the schedule sends it to the EMS. The Simulation and EMS then calculate the component models' results with significantly less computational effort.

Discussion

Current research suggests a gap regarding modularity, scalability, and flexibility of scheduling approaches for ES. Additionally, a more straightforward application of scheduling approaches to actual use cases is desired.

Novel OS approaches these gaps by implementing an adapter architecture to separate clients, evaluation logic, and optimization algorithm. OS provides data interfaces for the evaluation logic of candidate solutions from optimization algorithms. The interfaces are exemplary filled with rudimentary data models, boundary conditions, and topology information from ESs. Two optimization algorithms and one client are connected to one instance of OS in an EH simulation test case. The test case provides insights into which aspects of flexibility, modularity, scalability, and implementation to actual use cases can be fulfilled.

The presented architecture and implementation increase the modularity of scheduling in ES by separating data management on the client side, generating candidate solutions from optimization algorithms, and evaluating methods through the consequent implementation of data interfaces. Separated parts thereby increase independence and enable separate development.

The presented data interfaces allow the modular design of the evaluation logic. The three evaluation functions rely solely on data interfaces. Development and implementation of evaluation functions are independent of clients or optimization service data. Only when changing the structure of the data interfaces the modules have to be adjusted. The interfaces for implementing optimization algorithms through adapters help quickly implement different optimization algorithms for different optimization tasks. The presented architecture provides many possible applications, especially in comparing different approaches. Modular design facilitates the comparison of different approaches.

The implemented use case showed the separation of computational resources between the three instances, allowing the distribution of computational expenses on different machines. The evaluation of candidate solutions has shown to be the computationally most intensive part of the presented optimization problem. OS's architecture successfully decoupled client, OS, and optimization algorithms, thereby increasing the system's scalability.

OS communicates the optimization problem to multiple optimization algorithms, thereby demonstrating flexibility regarding optimization algorithms. Use case-specific data has been injected into OS, making OS a generic and flexible optimization tool. The architecture increased the flexibility of the system in the sense that changes in the simulation system only have to be implemented on the client side.

The construction of BCs in a pipeline also holds advantages. It creates independence between the client that supplies data and the internal library of BCs. The client can supply less detailed data in return for lower calculation accuracy. Modeling through BCs thereby becomes independent of data modeling. The independence is one key feature of the presented OS's flexibility, as the system does not require adjustments for clients with varying detail of data models.

The presented architecture can move the edge of current knowledge to more modular, scalable, and flexible ES scheduling shown in a realistic use case scenario.

Though, the selected test case cannot cover all architectural advantages. The setup limits the significance of evaluating the stated property of flexibility. OS received jobs from one single client that represented optimization problems of one single type of scenario. The stated increase in flexibility due to the ability to connect multiple clients with varying optimization problems can not be confirmed nor refuted. Similarly, implementing one OS instance allows no further conclusions regarding the client's flexibility to split optimization problems onto multiple OS instances. A test case combining clients with more diverse optimization problems, numerous instances of OS, and more advanced optimization algorithms is an extensive exercise left for ongoing research.

Additionally, to architectural limitations, the implemented facility logic is still elementary though functioning. This especially applies to the topological information regarding connections of different entities. The topological model is sufficient to represent the presented use case but has to be improved to apply to more diverse use cases. Apart from topology, the level of detail of data modeling and BCs was kept low for the proof of concept and might limit the accuracy of results. Especially under higher time resolution of the optimization, more detailed modeling might become noticeable or even necessary.

Conclusions

The presented paper first derives the necessity for an OS's flexible, modular, and scalable architecture from the literature. OS implements an adapter pattern that generically mediates between clients and optimization algorithms. Domain-specific logic evaluates candidate solutions and solves the client's optimization problems. Domain knowledge of ESs fills the presented architecture exemplary. A co-simulation scenario based on real-world scenario data from near Karlsruhe, Germany, implements OS for evaluation purposes. An EMS inside the co-simulation acts as a client, while the GLEAM, an exemplary metaheuristic evolutionary algorithm, acts as an optimization algorithm. The benefits and limitations are motivated based on the proof of concept implementation and conceptual work. The presented paper concludes with a brief outlook on future work and thoughts on overcoming current limitations.

Further research focuses on the implementation of topologies into the presented architecture. The literature presents different methods, but the most fitting must be identified and implemented. The additional implementation work focuses on the domain-specific modeling of the entity data class and BC. Both offer the potential to increase the level of detail and standardization.

The presented architecture can separate tasks in the ES scheduling into parts of different computational expenses. One improvement lies in a concept and implementation to efficiently manage resources of OS and optimization algorithms. Automated parallelization of computationally expensive parts is one promising approach in resource management. Presented OS receives an optimization problem from the client before initializing an optimization algorithm and holds all information for meta-optimization. Future implementations of OS could select a fitting optimization algorithm for the presented optimization problem to enhance optimization efficiency greatly.

Furthermore, as part of developing the presented architecture, a comparison of different optimization algorithms for (real) case ES scheduling will be provided. A promising research topic also explores the need for detailed modeling through extensive sets of BCs. The presented use case utilized merely five boundary conditions but still resulted in approximations close enough to schedule EH facilities successfully. Different sets of BCs can be easily compared with the presented architecture to research what detail level of modeling is sufficient for different use cases. Last, the setup allows optimizing problems from different domains, like a combination of operating and planning optimization. The generic structure should allow utilizing OS for different optimization problems.

Abbreviations

API	Application programming interface
BC	Boundary condition
CHP	Combined heat and power
DER	Distributed energy resource
EA	Evolutionary algorithm
ECP	Electrical connection point
EH	Energy hub
EMS	Energy management system
ES	Energy system
EU	European Union
GLEAM	General learning evolutionary algorithm and method
LUT	Lookup table
OS	Optimization service
RES	Renewable energy source

Acknowledgements
Special thanks to Jan Schweikert for his assistance in generalizing and analyzing the presented architecture.

Author contributions

Conceptualization and methodology: MC and JL; formal analysis: MC and JL; validation: MC; writing-original draft: MC and JL; writing-review and editing, MC, JL, RP, RL, KF, TS, and VH; supervision: KF, TS, and VH; funding acquisition: TS and VH. All authors read and approved the final manuscript.

Funding

This paper is part of the funded German project 'Energy System Design' (ESD) https://www.helmholtz.de/forschung/forschungsbereiche/energie/energiesystemdesign/ subtopic 2 (digitalization and system technology) by the Helmholtz Association https://www.helmholtz.de/. The implemented use case is part of the German ***Sektorkopplung*** https://www.esd.kit.edu/deutsch/85.php (SEKO, english: sector coupling) project. Use case is also part of the Kopernikus Project ENSURE ('New ENergy grid StructURes for the German Energiewende') https://www.kopernikus-projekte.de/projekte/ensure from the German Federal Ministry of Education and Research (BMBF) https://www.bmbf.de/.

Declarations

Competing interests

The authors declare that they have no competing interests.

About this supplement

This article has been published as part of Energy Informatics Volume 5 Supplement 4, 2022: Proceedings of the Energy Informatics. Academy Conference 2022 (EI.A 2022). The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-5-supplement-4.

Published: 21 December 2022

References

Allerding F (2014) Organic Smart Home—Energiemanagement für Intelligente Gebäude. KIT Scientific Publishing, Karlsruhe

Ayoub N, Musharavati F, Pokharel S, Gabbar HA (2018) An Approach for Energy Conservation Management Systems in Buildings. In: 2018 IEEE International Conference on Smart Energy Grid Engineering (SEGE); p. 9–13

Cao Y, Chen T, Sun L, Sun Y, Wei Z, Amaratunga GAJ (2020) Robust Energy Management for Uncertain Microgrid Using Modified Grey Wolf Optimizer. In: 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA); p. 1515–1519

Elsied M, Oukaour A, Gualous H, Hassan R (2015) Energy management and optimization in microgrid system based on green energy. Energy 84:139–151

Fathima AH, Palanisamy K (2015) Optimization in microgrids with hybrid energy systems—a review. Renew Sustain Energy Revi 45:431–446

Fernández-Guillamón A, Gómez-Lázaro E, Muljadi E, Molina-García A (2019) Power systems with high renewable energy sources: a review of inertia and frequency control strategies over time. Renew Sustain Energy Rev 115: 109369

Framinan JM, Ruiz R (2009) Guidelines for deploying and implementing scheduling systems. In: XIII Congreso de Ingeniería de Organización; p. 677–686

Framinan JM, Ruiz R (2010) Architecture of manufacturing scheduling systems: literature review and an integrated proposal. Eur J Opera Res 205(2):237–246

Geidel M (2007) Integrated modeling and optimization of multi-carrier energy systems. ETH Zürich. Available from: https://www.research-collection.ethz.ch/handle/20.500.11850/123494

Geidl M, Andersson G (2007) Optimal power flow of multiple energy carriers. IEEE Trans Power Syst 22(1):145–155

Ignat A, Lazar E, Petreus D (2018) Energy Management for an Islanded Microgrid Based on Particle Swarm Optimization. In: 2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging (SIITME); p. 213–216

Klaus T, Vollmer C, Werner K, Lehmann H, Mueschen K (2010) Energieziel 2050: 100% Strom aus erneuerbaren Quellen. Dessau-Rosslau: Umweltbundesamt; Available from: https://www.umweltbundesamt.de/publikationen/energieziel-2050

Marzband M, Ghadimi M, Sumper A, Domínguez-García JL (2014) Experimental validation of a real-time energy management system using multi-period gravitational search algorithm for microgrids in islanded mode. Appl Energy 128:164–174

Mauser I (2017) Multi-modal Building Energy Management. Karlsruher Institut für Technologie (KIT); 37.06.01; LK 01

Mauser I, Müller J, Allerding F, Schmeck H (2016) Adaptive building energy management with multiple commodities and flexible evolutionary optimization. Renew Energy 87:911–921

Mauser I, Müller J, Förderer K, Schmeck H (2017) Definition, Modeling, and Communication of Flexibility in Smart Buildings and Smart Grids. In: ETG-Fb. 155: International ETG Congress 2017. VDE; p. 605–610

Mohammadi M, Noorollahi Y, Mohammadi-ivatloo B, Yousefi H (2017) Energy hub: From a model to a concept—a review. Renew Sustain Energy Rev 80:1512–1527

Moriarty P, Honnery D (2016) Can renewable energy power the future? Energy Policy 93:3–7

Niknam T, Golestaneh F, Shafiei M (2013) Probabilistic energy management of a renewable microgrid with hydrogen storage using self-adaptive charge search algorithm. Energy 49:252–267

Obara S, Kawai M, Kawae O, Morizane Y (2013) Operational planning of an independent microgrid containing tidal power generators, SOFCs, and photovoltaics. Appl Energy 102:1343–1357

Poppenborg R, Ruf J, Chlosta M, Liu J, Hotz C, Düpmeier C, et al (2021) Energy Hub Gas : A Multi-Domain System Modelling and Co-Simulation Approach. In: Proceedings of the 9th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES '21). Association for Computing Machinery (ACM); p. Art.Nr. 12. 37.12.03; LK 01
UN, Brussels european council, 29/30 october 2009, presidency conclusions. United Nations Treaty Collection, Chapter XXVII 7 d. 2015; Available from: https://www.consilium.europa.eu/uedocs/cms_data/docs/pressdata/en/ec/110889.pdf
Van Beuzekom I, Gibescu M, Slootweg JG (2015) A review of multi-energy system planning and optimization tools for sustainable urban development. In: 2015 IEEE Eindhoven PowerTech; pp 1–7
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
Zhang Y, Gatsis N, Giannakis GB (2013) Robust energy management for microgrids with high-penetration renewables. IEEE Trans Sustain Energy 4(4):944–953

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com