Grosch *et al. Energy Informatics* 2022, **5**(Suppl 1):29
https://doi.org/10.1186/s42162-022-00207-6

Energy Informatics

RESEARCH Open Access

A framework for researching energy optimization of factory operations

Benedikt Grosch*, Heiko Ranzau, Bastian Dietrich, Thomas Kohne, Daniel Fuhrländer-Völker, Johannes Sossenheimer, Martin Lindner and Matthias Weigold

From The 11th DACH+ Conference on Energy Informatics 2022
Freiburg, Germany. 15-16 September 2022

*Correspondence:
b.grosch@ptw.tu-darmstadt.de

Technical University of Darmstadt, Institute for Production Management, Technology and Machine Tools, Otto-Berndt-Str. 2, 64287 Darmstadt, Germany

Abstract

Energy optimization of factory operations has gained increasing importance over recent years since it is understood as one way to counteract climate change. At the same time, the number of research teams working on energy-optimized factory operations has also increased. While many tools are useful in this area, our team has recognized the importance of a comprehensive framework to combine functionality for optimization, simulation, and communication with devices in the factory. Therefore, we developed a framework that provides a standardized interface to research energy-optimized factory operations with a rolling horizon approach. The optimization part of the framework is based on the OpenAI gym environment. The framework also provides connectors for multiple communication protocols including Open Platform Communication Unified Architecture and Modbus via Transmission Control Protocol. These facilities can be utilized to implement rolling horizon optimizations for factory systems easily and directly control devices in the factory with the optimization results. In this article, we present the framework and show some examples to prove the effectiveness of our approach.

Keywords: Industrial internet of things, Industrial demand side integration, Rolling horizon optimization

Motivation

Businesses are under increasing pressure to offer carbon-neutral products and services. The industrial sector, which consumes 26 % (eurostat 2021) of final energy in the European Union and 24 % (U.S. Energy Information Administration 2021) in the United States of America, has a special responsibility to reduce greenhouse gas emissions to counteract climate change. The proliferation of renewable power generation contributes to this reduction, but it comes with additional challenges, such as the stochastic availability of solar and wind energy (Degefa et al. 2021).

Demand response (DR), as part of Industrial Demand Side Integration, offers a solution by adjusting energy demand to the available generation capacity (Walther et al.

2022). Research regarding DR is ongoing and concerns many areas of factory operations. Shoreh et al. (2016) gives an overview of DR applications in industry. An example of a more specific approach to optimizing energy consumption of a factory is presented in Summerbell et al. (2017), which applies DR to a cement plant. Corinaldesi et al. (2020) introduces methods for modeling industrial equipment flexibility and uses a rolling horizon approach for optimization. Lu et al. (2021) implements machine learning based forecasts in a rolling horizon decision making process. A rolling horizon optimization approach is a recurring optimization of the system with shifting optimization horizons and starting values representing the system's current state (Sethi and Sorger 1991). In general, it is well suited to DR problems and the short overview shows that it is often utilized. However, most articles publish mathematical models, but software implementations of the demonstrated use cases are typically not available.

Our goal is thus, to publish a framework, which enables rolling horizon optimization and can easily be utilized for many use cases. To achieve this, the framework must fulfill the following requirements:

- The framework must offer a representation of the factory and of the devices or machines inside the factory, for example, through a mathematical or simulation model. We call this representation the environment.
- It should have an interface to perform the optimization, which should also support different algorithms. We refer to this as the agent.
- The agent and the environment need to interact during the rolling horizon optimization.
- The operation strategies determined by the agent have to be deployed to actual devices in the factory. For this, the framework must be able to communicate with the devices through industrial communication protocols such as Open Platform Communication Unified Architecture (OPC UA).
- The system must be able to handle time-series (or scenario) data. Time-series data can describe external variables such as the weather or energy prices (or forecasts thereof).

There are existing frameworks and libraries which fulfill some of the requirements; however, none of the existing solutions provide a sufficient basis to quickly implement rolling horizon energy-optimized factory operations in its entirety.

The *gym* framework by OpenAI specifies a standardized interface between agents and environments (Brockman et al. 2022). OpenAI developed the *stable_baselines* package based on this standard, which contains implementations of many different deep reinforcement learning (DRL) agents (Hill et al. 2018). The *stable_baselines3* package is based on *stable_baselines* and improves the agents (called algorithms in this case) (Raffin et al. 2021). The *gym* framework (Brockman et al. 2022) is very general and does not provide functions for integrating simulation models or connections to actual devices, which are required for energy-optimized factory operations. The *garage* framework (The garage contributors 2019) combines different environment interfaces, offers additional algorithms, and provides more customizable interfaces, but it remains specific to DRL and does not provide significant extensions for the

specification of environments. Other similar frameworks are *keras-rl* (Matthias Plappert 2016), *Coach* (Caspi et al. 2017), *ReAgent*, which Facebook uses for optimization based on batch data rather than simulators (Gauci et al. 2022), and *Acme* (Hoffman et al. 2022). *dopamine* (Castro et al. 2022) is meant for easy experimentation and supports only a subset of environments, *tensorforce* (Kuhnle et al. 2017) builds on tensorflow and offers multiple environment adapters (for example, with the OpenSim application programming interface (API)), *JuliaReinforcementLearning* (Tian et al. 2020) is a similar implementation for the Julia programming language. All of the above frameworks are specific to DRL and do not directly generalize to rolling horizon optimization. Therefore, they cannot readily be used with other optimization algorithms such as heuristics or linear and non-linear solvers. Most of the available frameworks and libraries are written in Python.

While there are frameworks for DRL, other research on rolling horizon optimization often implements an individual approach, creating duplicate work, and often not allowing the deployment of optimization results to actual devices in the factory. The deployment requires connections to the devices, for example, using industrial communication protocols such as OPC UA and (Modbus TCP) or through APIs exposed by applications like Internet of Things (IoT) platforms. Some libraries which facilitate this process exist: The FreeOpcUa contributors (2021), for example, implements the OPC UA standard in Python, and Lefebvre (2018) is a Python library for Modbus TCP . IOT platforms, such as *ThingSpeak* (The MathWorks Inc 2022) or *Cumulocity IoT* (Software AG 2022), can collect data from devices and decrease the effort of data aggregation and pre-processing. However, these platforms need to operate continuously and are usually deployed to separate devices or as cloud services. In addition, the integration of scripts for energy optimization is often manufacturer-specific and limited. Due to these factors, they are not well suited for fast-changing research applications.

Currently there are no software frameworks available that combine the requirements of rolling horizon optimization for energy-optimized factory operations. Thus, we propose the *eta_utility* framework, which combines the functionality of a DRL framework based on *gym* and *stable_baselines3* with the ability to integrate other optimization algorithms like linear or non-linear programming solvers. The framework includes a standardized connector interface that enables communication with devices via industrial communication protocols. It enables factories' rolling horizon optimization and execution of DR measures. Links to the software repository are provided in the Availability of data and materials section.

In this article, we introduce the framework's structure and demonstrate its usage with some examples. A live energy forecast of a machine tool in the Energy Technologies and Applications in Production (ETA) research factory shows an application of the **connectors** module. A second example is the rolling horizon optimization of an industrial cleaning machine for DR. It includes a simulation model of the machine and a direct connection to the machine's programmable logic controller (PLC). The code for these examples is published within the *eta_utility* repository (see Availability of data and materials). Finally, we present an overview of other applications realized using the *eta_utility* framework. In the conclusion, a summary of the presented work as well as an outlook on future research is given.

Structure of the proposed framework

The *eta_utility* framework is developed in Python. Its structure is illustrated in Fig. 1 and consists of the following modules:

- The **eta_x** module, shown on the left in Fig. 1 contains the rolling horizon optimization functionality. It uses the environment interface specified by the *gym* framework (Brockman et al. 2022) and the agents provided by the *stable_baselines3* (Raffin et al. 2021) library. The module contains additional agents and base classes that enable the fast creation of new environments for energy-optimized factory operations.
- In the top right of Fig. 1, the **connectors** module defines a standardized interface for connections between Python and industrial communication protocols and APIs. It uses the concept of nodes to uniquely identify a specific data point or variable on a specific device.
- The **simulators** module, bottom right in the figure, implements interfaces to simulation models following the Functional Mock-up Unit (FMU) standard (Modelica Association 2022) and is based on the *fmpy* package (Dassault Systèmes 2018).
- The **timeseries** module, right-center in the figure, provides additional functionality to handle time-series data using the *pandas* package (Jeff et al. 2022; McKinney 2010).
- The **servers** can be used to publish optimization results and make output data, for example, from forecasting services, available to downstream services.
- Finally, the **util** module provides ancillary functions such as logging and data de-serialization.

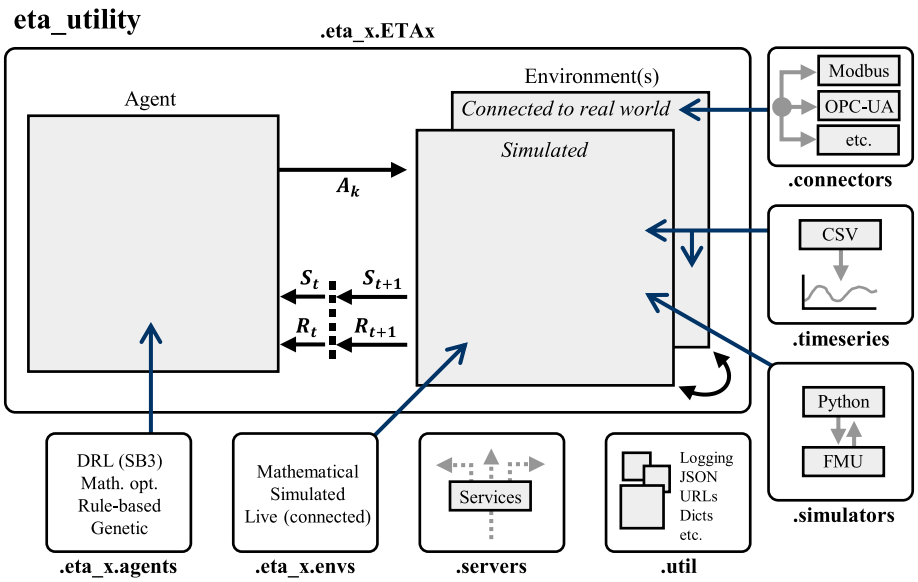

Fig. 1 Structure of the *eta_utility* framework. The rolling horizon optimization process is shown on the top left. Here, the algorithm selects actions A_k based on the current system state S_t and sends them to the environment, whereupon a new system state $S_t + 1$is established. In DRL algorithms, a reward signal R_t or $R_t + 1$is also transferred, which describes the quality of a system state. The modules are shown on the bottom and right

The design considerations for the **eta_x** and **connectors** modules are explained in the following sections. The remaining modules are less complex, and we point to the documentation provided with the framework for more information about them.

The eta_x module

eta_x combines functions from the other modules and provides the rolling horizon energy-optimization functionality. It is based on the OpenAI *gym* framework (Brockman et al. 2022) and the *stable_baselines3* package (Raffin et al. 2021). Some extensions for *stable_baselines3*, such as additional policies, extractors, schedules, and agents, are also included in the module. An initial version of **eta_x** was developed by Panten (2019), where it was applied for DRL-based energy optimization of factory supply systems. In addition to the DRL agents, hysteresis controllers were implemented. This concept has now been extended to generalized rule-based controllers. Furthermore, algorithms for solving linear programming models, initially proposed by Kohne et al. (2019), were added to **eta_x** in Panten et al. (2022).

eta_x is built on the concept of experiments. An experiment represents a specific environment, agent, and scenario configuration. In this context, the scenario could be a specific time of year combined with electricity market data. The configuration can be provided in JavaScript Object Notation (JSON) files. Depending on the type of agent used for the experiment, the agent must be trained with the **learn** function before an optimized operating schedule can be simulated or deployed with the **play** function. This functionality is combined within the **ETAx** class, but its parts can also be used standalone.

For the moment, three types of algorithms are available:

- A generalized, abstract rule-based algorithm (**RuleBased**) facilitates the creation of rule-based agents by specifying one or more control rules.
- An interface for mathematical solvers (**MathSolver**) integrates external solvers like *CPLEX* (IBM Corporation 2019). This agent is based on the *pyomo* (Hart et al. 2017) library.
- DRL algorithms from *stable_baselines3*, like Proximal Policy Optimization (PPO) (see Raffin et al. 2021 for more information).

Environments in **eta_x** are abstract subclasses of OpenAI *gym* (Brockman et al. 2022) environments and must be implemented to represent actual factories. The most important concept is the handling and configuration of the environment state. Each environment variable is represented by a **StateVar** object that contains all relevant information about the variable. For example, whether the variable is read from scenario data or a simulation model or whether the agent should set it as an action. Most other aspects of the environment can be determined and executed from the information contained in the **StateVar** objects.

As shown by the blue arrows in Fig. 1, the environments use most other modules in *eta_utility*. There are simulation environments that use the **simulators** module or live environments that directly interact with devices in the factory using the **connectors** module. The framework can handle two environments at once, which is helpful, for

example, when deploying an agent trained on a simulation model to the actual factory. The secondary environment could then be used to extend data available from the factory or to check actions taken by the agent. Data is shared between the two environments in this case, as indicated by the arrow in the lower right corner of Fig. 1.

The connectors module

The purpose of the **connectors** module is to enable communication between Python services and the actual environment. In the factory environment, these connections are often established using specific communication standards such as the standardized industrial automation framework OPC UA (OPC Foundation 2008) or the Modbus TCP (Modbus Organization, Inc 2006) communication protocol, which communicate via an Ethernet/IP network.

The **connectors** modules allows testing with real-time data and enables the deployment of energy-optimized operating strategies to the actual factory environment. For example, in the ETA research factory, we need to establish connections to read and write data from and to PLCs of production machines to implement an energy-optimized production schedule or the building automation system to generate optimized control strategies for energy supply systems. Connections to other systems, such as IoT platforms or energy management systems like EnEffCo (ÖKOTEC 2022), are suitable to obtain historical time-series data or to store optimization results.

The **connectors** module in *eta_utility* is built on two central concepts. First, **Node** objects are definitions of a single data point or variable. A **Node** contains all information to uniquely identify the **Node** and establish a connection to the data source. Second, **Connection** objects are used to establish a connection to a data source (server) and to read, write or subscribe to **Node**s from this server. All protocols and APIs can be treated equally for basic functionality like reading and writing data with these concepts. The **connectors** module also provides classes to handle data subscriptions and can treat multiple connections with an interface similar to a simulation model from the **simulators** module. The interface is provided by the **LiveConnect** class, which provides additional configuration options to automatically set the state of more complex systems of devices, which might require multiple **Node**s to be set to specific values to achieve a certain system state.

Application of the framework

To demonstrate the framework's capabilities, we present some examples of the usage of *eta_utility*. The live energy forecasting example illustrates capabilities of the **connectors** module. The cyber-physical production system example uses the framework to optimize the operation of an industrial cleaning machine. The code for these examples is included in the *eta_utility* software repository. The repository also contains some additional example code and documentation.

We also describe other applications which were implemented using the framework. These applications have been published elsewhere; thus, we only provide summaries.

Live energy forecasting

The forecasting example illustrates the usage of the connectors module. Forecasts of energy consumption or energy prices can be an essential element of the rolling horizon energy optimization. They provide the information required for subsequent optimization steps. Chang (2021) applies the **connectors** module to deploy an energy forecasting model on edge devices. The forecasting model, published in Dietrich et al. (2021), is a 100 s forecast of the electric load of a grinding machine in the ETA research factory based on a *keras* deep learning model. The forecast can be used for peak shaving or energy-optimal process scheduling. Input data for the forecast consists of nine signals corresponding to the total electric load, and the electric load of sub-components of the grinding machine. The data is a consecutive sequence of 100 s with a frequency of 1 Hz.

For deployment of the model, (Chang 2021) implements a loop of reading data, model inference, and publishing the forecast and executes it with a frequency of 1 Hz. An OPC UA server on the grinding machine's PLC and a Modbus TCP server on the energy metering device are the data sources.

The **connectors** module is used to connect to the data sources. The direct connection to the production machine's PLC and sensor gateway is advantageous due to the flexibility it provides. It facilitates data processing, integration with the energy-optimization and fast development times. Other solutions, for instance, using Telegraf to read OPC UA and Modbus TCP data directly to InfluxDB offer less flexibility and introduce additional points of failure.

Cyber-physical production system for demand response

The second use case takes advantage of the entire *eta_utility* framework for energy-flexible operation of an aqueous cleaning machine (Grosch et al. 2022). The aim is to execute DR measures on the cleaning machine MAFAC KEA in the ETA research factory by controlling its tank heater. Accordingly, the machine is extended to a cyber-physical production system that includes:

- the physical machine and its automation program (implemented as an **eta_x** environment with a **connectors** instance),
- an OPC UA server on the machine's PLC to publish current operation steps, conditions and energy consumption data (which we connect to with the **connectors** instance),
- a dynamic multi-physics simulation model of the machine (implemented as an **eta_x** environment with a **simulators** instance) and
- the DR service (agent) to control the machine that receives external energy prices for its optimization (implemented using the **timeseries** module).

The *eta_utility* framework manages the interaction between these components.

In Grosch et al. (2022), the DR service and the simulation model are executed on a PC connected to the machine's PLC via Ethernet. The *eta_utility* framework is used as a cyber-physical interface to connect different hierarchy levels of the Reference

Architecture Model Industry 4.0 (DIN 2016): The machine modules controlled by the automation program and executed on the PLC are located on the field or control device level. The DR service and the simulation are located on the station level.

The authors of Grosch et al. (2022) use an automation data model for the hierarchical connection between the machine automation and the DR service. The automation data model consists of the automation data specification and the automation data dictionary (Grosch et al. 2022): The automation data specification is located on the machine's PLC and its implementation leads to a hierarchically structured OPC UA server. The automation data dictionary is implemented using the **LiveConnect** class which is configured using a JSON file. It includes

- the name and the IP address of the OPC UA server(s),
- the user name and password to establish a connection,
- **Node** descriptions and data types for variables used by the DR service and
- a mapping between the **Node**s and equivalent variables of the DR service.

The DR service fetches external energy prices to determine the state of the tank heater. It is switched on when the energy price is below 100 €/MWh and switched off when the price is above 100 €/MWh, as shown in Fig. 2. The DR service also interacts with the simulation model of the cleaning machine which simulates the thermal and electrical behavior of the machine. A forecast of the tank heater state from the simulation model guarantees that temperature limits are not exceeded by DR measures. Overall, the field test shows a 19 % decrease in energy costs. The controller and the simulation model performed as expected and set safety values were obeyed.

Other applications

eta_utility permits comfortable implementation of various use cases, leading to substantially reduced time to obtain research results for energy-optimized factory

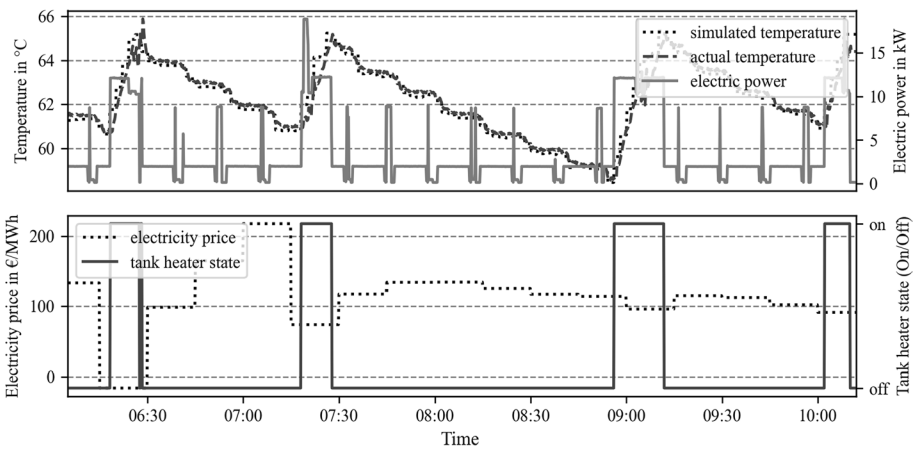

Fig. 2 Tank temperature and machine's electrical power consumption (top). Electricity price and tank heater state (bottom). The upper temperature limit of 65 °C is exceeded for 160 s. The tank heater is switched four times during low prices. Grosch et al. (2022)

operations. In the following we present a selection of additional use cases built on the functionality of *eta_utility*.

Low-cost energy monitoring based on offline trained prediction models

In order to reduce the costs of monitoring energy flows in the factory, Hybrid Virtual Energy Metering Points (VMPs) can be used. VMPs are offline trained models that predict the energy consumption of energy consumers like industrial production machines. They are set up empirically by correlating a temporary power consumption measurement with machine-internal process and state signals (Sossenheimer et al. 2020). Further research shows how other data sources can be used for training and deploying VMPs in case of insufficient machine data availability (Sossenheimer et al. 2021). The overall goal of using VMPs is to save costs by deploying trained black-box machine learning models to predict the energy consumption instead of installing physical metering devices. The **connectors** module is used to read the necessary machine data to the VMPs and the **servers** module to publish the predicted power consumption via OPC UA.

Optimized control of a central cooling system

In Weigold et al. (2021), the **eta_x** module is used and the DRL algorithm PPO is successfully applied to a simulation of an industrial cooling supply system. Significant reductions in electricity costs by 3 % to 17 % as well as reductions in CO_2 emissions by 2 % to 11 % are achieved. The DRL-based control strategy is interpreted and three main reasons for the performance increase are identified. The DRL controller reduces energy cost by utilizing the storage capacity of the cooling system and moving electricity demand to times of lower prices. Additionally, the DRL-based control strategy for cooling towers (CT) as well as compression chillers (CC) reduces electricity costs and wear-related costs alike (compare Fig. 3). To achieve these results, the cost function to be minimized was designed as a weighted sum of temperature restriction cost (C_T), energy cost (C_E), switching cost (C_S) and other cost (C_O), each multiplied by individual weights (w):

$$C = \mathrm{w_T C_T} + \mathrm{w_E C_E} + \mathrm{w_S C_S} + \mathrm{w_O C_O}. \tag{1}$$

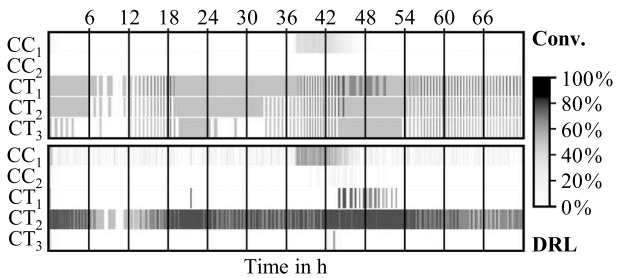

Fig. 3 Control signals of a 3 day test episode. In comparison to the conventional (Conv.) controller, the DRL controller preferably utilizes one cooling tower pair up to its highest power level before activating the next. This keeps unneeded water pumps inactive, reducing energy cost. This approach also allows for fewer total switching operations, improving switching cost (Weigold et al. 2021)

Comparative study of algorithms for optimized control of energy supply systems

In Kohne et al. (2020), the **eta_x** module is used to obtain a standardized comparative study of different controllers (rule-, model- and data-based) for optimized operation strategies by connecting them to dynamic simulation models of two industrial energy supply systems of varying complexity. The first energy supply system consists of a heating, gas and electricity grid which are supplied by a combined heat and power unit, a gas boiler and an immersion heater. In the second energy supply system, a cooling grid with a cooling tower, a compression chiller and a heat pump between heating and cooling grid are added. The rule-based controller activates or deactivates the respective energy converters based on the temperatures in the top and bottom of the thermal storages depending on a priority list. The objective function of the model-based controller (mixed-integer linear programming (MILP)) is explained by Eq. (2), which contains costs for gas (C_G) and electricity (C_{El}) as well as taxes (C_P) on procured energy and charges for peak loads. Additionally, non-direct costs for switching (C_S) are added. As not every optimization run might result in feasible solutions due to grid constraints, infinite sinks and sources (C_{SS}) are modeled to ensure system stability of the optimization process, resulting in

$$\min C = \mathrm{C_G} + \mathrm{C_{El}} + \mathrm{C_P} + \mathrm{C_S} + \mathrm{C_{SS}}. \tag{2}$$

The cost function of the data-based DRL controller is designed as a sum of weighted terms, similar to Eq. (1) with costs for energy, switching, temperature limits and other cost (Panten 2019). The results indicate that controllers based on DRL and MILP have significant potential to reduce energy-related costs of up to 50 % for less complex (Fig. 4) and around 6 % for more complex systems.

Conclusion

In this paper, we presented a software framework for research on energy-optimized factory operations. The *eta_utility* framework is based on the OpenAI *gym* environment and follows a rolling horizon optimization approach. It is developed in Python and consists of six modules: **eta_x**, **connectors**, **simulators**, **timeseries**, **servers**, and

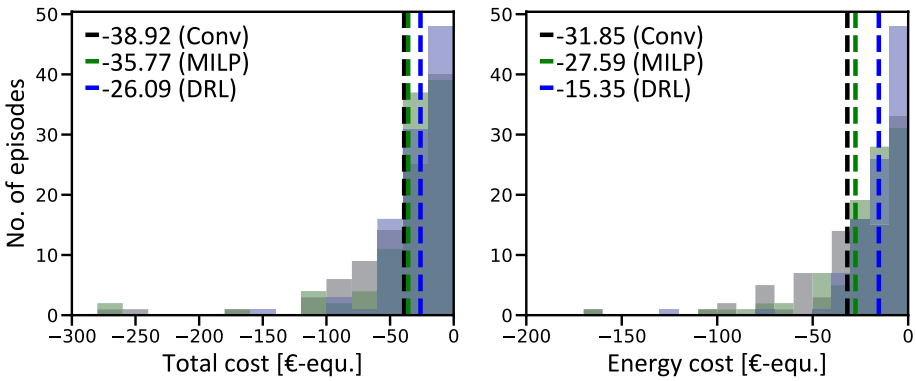

Fig. 4 Distribution and mean value of overall and specific cost per episode for the smaller system. Cost in this case are interpreted as negative rewards. Thus, increasing performance is indicated by higher values (Kohne et al. 2020)

util. As shown in the examples, *eta_utility* allows fast and simple implementation of rolling horizon optimizations for energy-optimized factory operations, and to deploy the results by controlling devices in the factory. For this, the framework provides **connectors** for multiple communication protocols including Open Platform Communication Unified Architecture and Modbus via Transmission Control Protocol. It also integrates simulation models through the Functional Mock-up Unit standard. Within the paper, we explained the overall structure, functionality of *eta_utility* and the design decisions for the main modules **eta_x** and **connectors**.

To demonstrate the capabilities of the framework we introduced several examples for the usage of *eta_utility*. Code for the two main examples is included in the *eta_utility* software repository (see Availability of data and materials). The examples show that all modules of the software framework are applicable for different use cases and applications. Use cases for transparency such as forecasting are shown as well as the energy-optimization of factories. The latter is carried out both in simulation and in the application to real systems within the ETA research factory. The implementation of the examples proves the effectiveness of our approach.

Nevertheless, there are still some useful expansion options that could necessitate further research. Agents in *eta_utility* can utilize vectorized environments, however it is currently not possible to check multiple solutions in a single thread. This functionality would be needed for more efficient execution of heuristics. Additionally, the framework does not yet offer multi-agent optimization.

By publishing the *eta_utility* software framework we hope to contribute to further research on energy-optimized factory operations outside our research group. We will continue our research and to improve the framework to make it applicable for broader use cases in energy-optimized factory operations.

Abbreviations

API	Application programming interface
DR	demand response
DRL	Deep reinforcement learning
ETA	Energy technologies and applications in production
FMU	Functional mock-up unit
IoT	Internet of things
JSON	JavaScript object notation
MILP	Mixed-integer linear programming
Modbus TCP	Modbus via transmission control protocol
OPC UA	Open Platform Communication Unified Architecture
PLC	Programmable logic controller
PPO	Proximal policy optimization
VMPs	Hybrid virtual energy metering points

Acknowledgements

We would like to thank all those who contributed to the *eta_utility* software framework, in particular: Niklas Panten, Nina Strobel and Thomas Weber as well as Guilherme Fernandes Gonçalves Silva, Tobias Lademann, Saahil Nayyer, Magdalena Patyna and Jerome Stock.

About this supplement

This article has been published as part of Energy Informatics Volume 5 Supplement 1, 2022: Proceedings of the 11th DACH+ Conference on Energy Informatics. The full contents of the supplement are available online at https://energyinformatics.springeropen.com/articles/supplements/volume-5-supplement-1.

Author contributions

BG: Conceptualization, Software, Validation, Writing—Original Draft, Visualization HR: Conceptualization, Software, Writing—Original Draft, Visualization BD: Conceptualization, Software, Writing—Original Draft TK: Conceptualization, Software, Writing—Original Draft DF-V: Conceptualization, Writing—Original Draft JS: Conceptualization, Writing—Original

Draft ML: Conceptualization, Writing—Original Draft MW: Writing—Review & Editing, Supervision, Project Administration, Funding Acquisition. All authors read and approved the final manuscript.

Funding
The authors gratefully acknowledge financial support of the Project "KI4ETA" (Grant Number 03EN2053A) by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) and thank the Projektträger Jülich (PtJ) for the project supervision.

Availability of data and materials
The *eta_utility* framework is available on Github: https://github.com/ptw-tuda/eta_utility and on PyPi: https://pypi.org/project/eta_utility.

Declarations

Competing interests
The authors declare that they have no competing interests.

Published: 7 September 2022

References

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2022) OpenAI Gym. http://arxiv.org/pdf/1606.01540v1
Caspi I, Leibovich G, Shadi Endrawis Novik G (2017) Reinforcement Learning Coach. Zenodo
Castro PS, Moitra S, Gelada C, Kumar S, Bellemare M, Dopamine G (2022) A research framework for deep reinforcement learning. http://arxiv.org/pdf/1812.06110v1
Chang P (2021) Deployment of ai algorithms for industrial energy prognosis on edge devices. Master thesis, TU Darmstadt, Darmstadt
Corinaldesi C, Schwabeneder D, Lettner G, Auer H (2020) A rolling horizon approach for real-time trading and portfolio optimization of end-user flexibilities. Sustain Energy Grids Netw 24:100392
Dassault Systèmes: FMPy. GtiHub (2018). https://github.com/CATIA-Systems/FMPy Accessed 2018-10-21
Degefa MZ, Sperstad IB, Sæle H (2021) Comprehensive classifications and characterizations of power system flexibility resources. Electric Power Syst Res 194:107022
Dietrich B, Walther J, Chen Y, Weigold M (2021) A deep learning approach to electric load forecasting of machine tools. MM Sci J 2021(5):5283–5290
DIN - Deutsches Institut für Normung e.V.: Referenzarchitekturmodell Industrie 4.0 (RAMI4.0). Beuth Verlag GmbH, Berlin (2016). https://www.beuth.de/en/technical-rule/din-spec-91345/250940128 Accessed 27 Jan 2021
eurostat: Final energy consumption by sector: Online data code TEN00124 (2021). https://ec.europa.eu/eurostat/databrowser/view/TEN00124/default/table Accessed 04 Apr 2022
Gauci J, Conti E, Liang Y, Virochsiri K, He Y, Kaden Z, Narayanan V, Ye X, Chen Z, Fujimoto S (2022) Horizon: Facebook's Open Source Applied Reinforcement Learning Platform. http://arxiv.org/pdf/1811.00260v5
Grosch B, Fuhrländer-Völker D, Stock J, Weigold M (2022) Cyber-physical production system for energy-fl exible control of production machines. Procedia CIRP 55
Hart WE, Laird CD, Watson J-P, Woodruff DL, Hackebeil GA, Nicholson BL, Siirola JD (2017) Pyomo—Optimization Modeling in Python. 2nd edn. Springer Optimization and Its Applications, vol. 67. Springer International Publishing, Cham
Hill A, Raffin A, Ernestus M, Gleave A, Traore R, Dhariwal P, Hesse C, Klimov O, Nichol A, Plappert M, Radford A, Schulman J, Sidor S, Wu Y (2018) Stable Baselines. GitHub
Hoffman M, Shahriari B, Aslanides J, Barth-Maron G, Behbahani F, Norman T, Abdolmaleki A, Cassirer A, Yang F, Baumli K, Henderson S, Novikov A, Colmenarejo S.G, Cabi S, Gulcehre C, Le Paine T, Cowie A, Wang Z, Piot B, Freitas N.d (2022) Acme: A Research Framework for Distributed Reinforcement Learning. http://arxiv.org/pdf/2006.00979v1 Accessed 27.05
IBM Corporation: IBM ILOG CPLEX Optimization Studio (2019). https://www.ibm.com/docs/en/icos/ Accessed 05 Apr 2022
Jeff R, Jbrockmendel, Wes M, Van den Bossche J, Tom A, Phillip C, Simon H, Matthew R, Gfyoung, Sinhrks, Adam K, Patrick H, Terji P, Jeff T, Chang S, William A, Shahar N, Darbyshire JHM, Marc G, Richard S, Jeremy S, Andy H, Daniel S, Marco EG, Fangchen L, Matthew Z, Vytautas J, Ali M, Pietro B. Skipper Seabold: pandas-dev/pandas: Pandas 1.4.1. Zenodo (2022)
Kohne T, Becker PA, Weber T, Panten N, Abele E (2019) Modeling approach for thermal dependencies in complex industrial energy supply system. Energy Informat 2(2):13–17
Kohne T, Ranzau H, Panten N, Weigold M (2020) Comparative study of algorithms for optimized control of industrial energy supply systems. Energy Informat 3(1):1–19
Kuhnle A, Schaarschmidt M, Fricke K (2017) tensorforce. GitHub. https://github.com/tensorforce/tensorforce
Lefebvre L (2018) pyModbusTCP. GitHub. https://github.com/sourceperl/pyModbusTCP/
Lu R, Bai R, Huang Y, Li Y, Jiang J, Ding Y (2021) Data-driven real-time price-based demand response for industrial facilities energy management. Appl Energy 283:116291
Matthias Plappert: keras-rl. GitHub (2016). https://github.com/keras-rl/keras-rl. Accessed 10 Jun 2022
McKinney W (2010) Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference. In: Proceedings of the Python in Science Conference, pp. 56–61. SciPy

Modbus Organization, Inc.: MODBUS Messaging on TCP/IP Implementation Guide V1.0b (2006). https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf Accessed 16 Mar 2022
Modelica Association: (2022) Functional Mock-up Interface 2: FMI for Model Exchange and Co-Simulation. https://github.com/modelica/fmi-standard/releases/download/v2.0.3/FMI-Specification-2.0.3.pdf. Accessed 23 Mar 2022
ÖKOTEC Energiemanagement GmbH: Energieeffizienz Controlling (2022). https://www.oekotec.de/unser-portfolio/energieeffizienz-controlling/ Accessed 16 Mar 2022
OPC Foundation: OPC Unified Architecture (2008). https://opcfoundation.org/about/opc-technologies/opc-ua/ Accessed 16 Mar 2022
Panten N (2019) Deep Reinforcement Learning zur Betriebsoptimierung Hybrider Industrieller Energienetze: Dissertation. Innovation Fertigungstechnik. Shaker, Aachen . https://shop.falter.at/detail/9783844070361 Accessed 28 Feb 2020
Panten N, Ranzau H, Kohne T, Moog D, Abele E, Weidner E (2022) Simulation und Optimierung kombiniert
Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N (2021) Stable-baselines3: reliable reinforcement learning implementations. J Mach Learn Res 22(268):1–8
Sethi S, Sorger G (1991) A theory of rolling horizon decision making. Ann Oper Res 29:387–415
Shoreh MH, Siano P, Shafie-khah M, Loia V, Catalão JPS (2016) A survey of industrial applications of demand response. Electric Power Syst Res 141:31–49
Software AG: Cumulocity IoT, Darmstadt (2022). https://www.softwareag.cloud/site/product/cumulocity-iot.html. Accessed 04 Apr 2022
Sossenheimer J, Vetter O, Abele E, Weigold M (2020) Hybrid virtual energy metering points—a low-cost energy monitoring approach for production systems based on offline trained prediction models. 21st CIRP Conference on Life Cycle Engineering 93, 1269–1274
Sossenheimer J, Vetter O, Stahl T, Weyand A, Weigold, M (2021) Hybrid virtual metering points—a low-cost, near real-time energy and resource flow monitoring approach for production machines without plc data connection. 21st CIRP Conference on Life Cycle Engineering 98, 452–457
Summerbell DL, Khripko D, Barlow C, Hesselbach J (2017) Cost and carbon reductions from industrial demand-side management: study of potential savings at a cement plant. Appl Energy 197:100–113
The FreeOpcUa contributors: python-opcua. GitHub (2021). https://github.com/FreeOpcUa/python-opcua
The garage contributors: garage. GitHub (2019). https://github.com/rlworkgroup/garage
The MathWorks Inc.: ThingSpeak. The MathWorks Inc. (2022). https://thingspeak.com/
Tian J et al. (2020) ReinforcementLearning.jl: A Reinforcement Learning Package for the Julia Programming Language. GitHub. https://github.com/JuliaReinforcementLearning/ReinforcementLearning.jl
U.S. Energy Information Administration: Monthly Energy Review April 2021: 7. Electricity, Washington, DC. https://www.eia.gov/totalenergy/data/monthly/ Accessed 09 May 2021
Walther J, Dietrich B, Grosch, B, Lindner M, Fuhrländer-Völker D, Strobel N, Weigold M (2022) A methodology for the classification and characterisation of industrial demand-side integration measures. Energies 15(3)
Weigold M, Ranzau H, Schaumann S, Kohne T, Panten N, Abele E (2021) Method for the application of deep reinforcement learning for optimised control of industrial energy supply systems by the example of a central cooling system. CIRP Ann Manuf Technol 70(1):17–20

Publisher's Note