EnCortex: A General, Extensible and Scalable Framework for Decision Management in New-age Energy Systems

Anonymous Submission #705

Abstract

With increased global warming, there has been a significant emphasis to replace fossil fuel-dependent energy sources with clean, renewable sources. These new-age energy systems are becoming more complex with an increasing proportion of renewable energy sources (like solar and wind), energy storage systems (like batteries), and demand side control in the mix. Most new-age sources being highly dependent on weather and climate conditions bring about high variability and uncertainty. Energy operators rely on such uncertain data to make different planning and operations decisions periodically, and sometimes in real-time, to maintain the grid stability and optimize their objectives (cost savings, carbon footprint, etc.). Hitherto, operators mostly rely on domain knowledge, heuristics, or solve point problems to take decisions. These approaches fall short because of their specific assumptions and limitations. Further, there is a lack of a unified framework for both research and production environments at scale.

In this paper, we propose `EnCortex` to address these challenges. `EnCortex` provides a general, easy-to-use, extensible, and scalable energy decision framework that enables operators to plan, build and execute their real-world scenarios efficiently. We show that using `EnCortex`, we can define and compose complex new-age scenarios, owing to industry-standard abstractions of energy entities and the modularity of the framework. `EnCortex` provides a foundational structure to support several state-of-the-art optimizers with minimal effort. `EnCortex` supports both quick developments for research prototypes and scaling the solutions to production environments. We demonstrate the utility of `EnCortex` with three complex new-age real-world scenarios and show that significant cost and carbon footprint savings can be achieved.

1 Introduction

With increasing global climate change, the energy industry is going through a paradigm shift [8, 36]. Electricity generation is currently responsible for 40% of global CO2 emissions [15]. Hence, there is an increasing focus on incorporating greener and renewable energy sources like solar, wind, and hydro in the energy generation mix. The share of renewables in global electricity generation jumped to 29% in 2020 and is expected to grow up to 69% by 2050 [15], [3]. Being dependent on the weather conditions like wind, cloud covers, and rains, these renewable energy sources induce high variability and intermittency in the generation mix [10]. To address such challenges, various types of energy storage systems (ESS) such as batteries, hydroelectric pumps, and flywheels are also introduced [37]. On the other side, energy demand still has an increasing trend with a 19% increase in energy demand between 2009 to 2019 [34]. There is also significant uncertainty in the energy demand depending upon seasonality, economic trends, weekday/weekend etc. There are also various types of demand-side management (DSM) approaches (including various control actions to reduce or increase energy demands) that are being executed by consumers or energy operators to reduce the peak demands and increase monetary savings [2].

Thus, new-age energy systems have become more complex with increasing uncertainty from the generation sources and demand sides. Various energy operators – like energy producers (i.e., solar/ wind farm operators), balancing authorities or consumers like (data centers, malls, universities, or households) need to make different planning and operations decisions periodically, and sometimes in near real-time, to maintain the grid stability and to optimize their objectives (maximize cost savings or reduce carbon footprint). Also, to support such a paradigm shift, new types of green energy markets and incentives are being introduced making the whole system dynamic and adapting to the changes rapidly.

Improving the efficiency of operating power plants has been a long-studied problem [29] in the academic literature. There are various optimization scenarios studied in the literature, including participation in multiple markets [6], demand response and surplus management [38], as well as other scenarios with multi-objective optimization for minimization of carbon emissions as well as monetization [21]. Various techniques of optimization, e.g., linear programming, robust optimization, reinforcement learning, fuzzy logic, etc. have been used to address the optimization problem [29]. However,

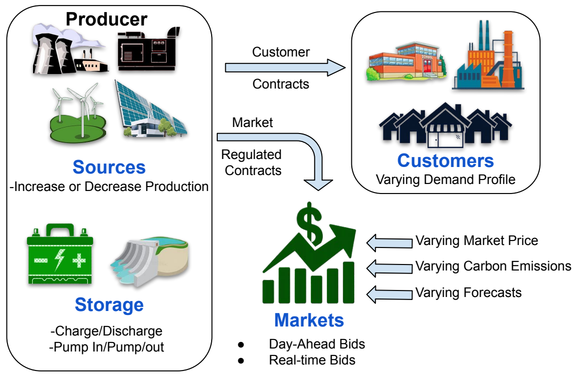


Figure 1: Illustration of an energy producer with the complexity involved in the decision-making.

industries and energy operators still rely on heuristics and/or manual methods or solving point problems for making their planning and operations decisions.

A primary reason for this gap stems from the lack of generalization, customizability, and ease of use with the existing approaches [23, 29]. Figure 1 shows a producer with a portfolio of generation sources like wind, solar, and storage. Different types of sources have different actions available, e.g., charge/discharge for batteries, increase/decrease production for nuclear/thermal plants, or, just curtail output for solar/wind farms. Producers also participate in various markets - long-term purchase agreements, week-ahead, day-ahead, intra-day and real-time markets [11]. The producer might also have contracts with different classes of consumers - data centers, residential, commercial, industrial, etc [16]. Each of these contracts has its own penalty structures and deadlines. In this context, the producer has to make decisions corresponding to the available actions while incorporating the varying generation profiles, varying consumer profiles, and the fluctuation in markets over a long-term as well as short-term bidding process. The uncertainty in prices, demand, and generation make this problem even harder to address without a generalized optimization framework. Further, the optimization decision can vary significantly when the scenario being optimized has an additional power plant or a long-term contract with variable demand throughout the day. This makes it difficult for existing formulations to be useful for a new scenario and end up being bespoke solutions that aren't applicable.

Therefore, there is a need for a general framework that provides - (i) *industry-standard abstractions* for various types of sources (modeling their actions and behaviors in a consistent way), (ii) *extensibility support* to introduce new types of energy entities (like hydrogen electrolyzer), (iii) *composibility* for developing new scenarios based on energy operators' needs, constraints and objectives, (iv) inherent ability to make *data-driven decisions and perform what-if analysis*, and most importantly, (v) *ease of use* for developers, business users, and energy operators.

To address the above challenges, we design `EnCortex`, a general, extensible, composable, and scalable framework for decision management in sustainable planning and operations of new-age energy systems. `EnCortex` provides industry-standard abstractions with realistic modeling and configurations for most new-age entities. `EnCortex` supports both the definition and composition of new scenarios in a structured way with very minimal overhead. `EnCortex` offers native support to model the variability and uncertainty in renewable sources and markets by incorporating data-centric decision-making. Finally, `EnCortex` is easy to use, programmable and scalable to run complex scenarios across multiple compute clusters and provides support for different optimizers to perform what-if analysis.

We summarize our contributions as follows: ❶ `EnCortex` is the first general, easy-to-use, extensible, and scalable energy decision framework that enables various energy operators to plan, build and execute their real-world energy decision scenarios efficiently. Energy operators can quickly express their objectives and constraints without worrying about the details of implementation and scaling (Section 3). ❷ `EnCortex` is modular and data-centric by design with support for public datasets and streaming data (Section 4). ❸ We demonstrate the support for modularity, extensibility, and composability features in `EnCortex`, by formulating three popular new-age energy scenarios with different levels of complexity (Section 5). ❹ We show that `EnCortex`'s core architecture provides a foundational structure and support for both traditional and neural-network-based optimizers. Our experiments demonstrate significant savings in both cost and carbon footprint across datasets/scenarios (Section 7). Finally, our experiments illustrate scenarios with industry partners in various geographies around the world. Thus giving confidence in the generality and applicability of `EnCortex` to a large number of practical new-age energy scenarios.

2 Motivation

`EnCortex` provides a foundational structure for generic scenario compositions as opposed to point formulation with support for ease of use, extensibility, composability, and scalability. In this section, we describe why such a system is needed to aid energy operators.

2.1 A Motivating Example

Most renewable energy resources such as solar and wind, being highly dependent on weather and climate conditions, bring about variability and uncertainty [10]. Hence, this has led to the rise of Energy Storage Systems (ESS), which provide a multitude of low-carbon emission-based ancillary services to the energy operators such as demand peak-shaving and grid stabilization [30].

Our partner, a global energy provider is deploying Li-ion batteries at their consumer premises to either maximize economic profit (increase cost savings by charging/discharging the battery at off-peak/peak price periods) or maximize environmental impact (reduce overall carbon footprint by charging/discharging the battery based on usage of renewable/non-renewable energy sources). Thus, it is important to optimize the management and scheduling of these batteries (when to charge or discharge) to maximize the objective. Intelligent optimization techniques are required to recommend optimal schedules that take into account numerous real-world factors such as available generation mix, state of charge of batteries, their efficiency, electricity purchase, and sale contracts, etc.

In order to determine the optimal schedule of the deployed batteries, our partner has to build this solution from scratch as prior solutions are mostly customer-specific and are not generalizable or reusable [7, 29]. Specifically, our partner has to define the different entities involved in the scenario such as the battery, grid, and renewable generation mix along with defining the objective function (price or carbon savings) and build an optimization service using either standard robust optimization techniques or newer neural network-based approaches. Developing an end-to-end solution for this scenario is non-trivial and majority of the energy operators find it significantly hard, and time-consuming.

Problem 1: Several point solutions exist, but fall short because of their specific assumptions and limitations. While the above scenario is pretty common and representative, there is no off-the-shelf tool/framework to solve this. Energy operators resort to developing ad-hoc tools to handle specific cases leading to inefficient and complex implementations. These bespoke solutions are mostly developed with specific assumptions on the abstractions and definitions of the sources and battery, and optimization algorithms employed. For instance, it is non-trivial to change or add new storage definitions say from Li-ion to another battery composition.

Problem 2: The ad-hoc solutions are inefficient and usually cannot be reused. It is clear that the above point/ad-hoc solutions are not built to be applied to other configurations. The scenario definitions, energy purchase and sell contracts, and entity abstractions will likely be specific for that formulation. Since the design and implementation have to be swift, reusability, performance and scalability will likely be not the priorities. Thus, requiring significant understanding and time to re-purpose these solutions.

Problem 3: Existing solutions are not modular and do not support the addition of custom scenario definitions. Even if there exists a tool/solution to solve the above scenario, modifying the scenario objective and adding new entities/markets is still a huge challenge. For example, in the above scenario let's say the partner wants to develop a multi-objective function to maximize both cost and carbon footprint instead of just maximizing cost or carbon footprint. In such cases given the lack of modularity, abstractions, and extensibility of ad-hoc solutions it is impossible to compose and define such custom scenarios.

After observing these problems with our partners during planning and operations, we believe that a general, easy-to-use, extensible, and data-centric decision management framework can bring significant benefits to energy operators. It can help them construct, understand, and analyze different scenarios swiftly, and make decisions more efficiently.

2.2 Design Goals

Motivated by many real-world examples like the one in Section 2.1 (and more in Section 5), we derive the design goals of `EnCortex` to support numerous new-age energy scenarios.

1. New-age industry-standard entity abstractions. Hitherto, most of the solutions employ unstructured and disjoint definitions for energy entities such as sources, storage, consumer profiles, and markets [32]. These definitions are not standardized and cannot be reused for quick development. A key design goal is to build industry-standard abstractions that incorporate how the new-age entities such as sources, storage, and markets can be modeled realistically.

2. Extensibility support for entity abstractions. While entity abstractions are useful, it is even more important to support the extensibility of these abstractions. For example, a market abstraction should support extensions to address different types of markets such as intra-day, day-ahead, and real-time. Thus allowing extensions for the well-defined generic abstractions and support for definitions of new entities.

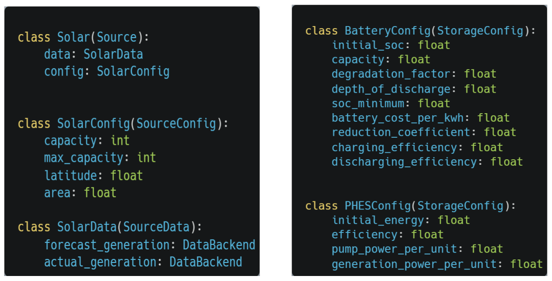
3. Composability of new-age energy scenarios. A scenario is comprised of numerous entities. For example, in the energy arbitrage scenario (Section 2.1), the entities include a battery, generation sources (solar, wind), and electricity purchase contracts. Currently, scenario formulation is not standardized and extensible. A key design goal is to support easy-to-use and intuitive ways to compose a scenario with entity abstractions. Further, we should be able to alter/modify these scenario definitions based on the application needs. For example, it should be trivial to extend single objective optimization (such as carbon footprint reduction) to multi-objective (i.e., joint optimization of both cost and carbon footprint).

4. Modular and support for custom scenario definitions. Current solutions are mostly bespoke and cumbersome to expand [7]. A general framework should be capable of handling a multitude of scenario compositions with different entities, objectives, and constraints.

5. Data-centric decision-making with support for real-time analysis. Data-based decision-making is a key element to optimal decision-making. Existing solutions either employ simulated data or work with small datasets, which may not capture the complete trend, variability, and uncertainties. Hence, framework design should provide native support to ingest large entity datasets of solar, wind, price information, etc., to perform detailed analysis and derive optimal decisions at different time granularity.

6. Scalable workflow to aid what-if analysis. Given the lack of modularity in the existing optimization solutions, they typically do not scale well, especially when evaluating on large datasets. Hence, it is important to provide a host of tools in the framework to support scalable workflows, offload compute to CPU/GPU clusters, and native support for different optimizers to do detailed what-if analysis.

7. Ease of use. Energy operators might have limited data science expertise and hence it is important for the framework to be easy to use wrt defining and composing scenarios, parameterizing and adding new abstractions, and supporting quick development for research prototypes to even scaling in production environments. Further, it should support visualizing the scenario and interactive result generation to understand the decision outcomes on different datasets and time periods.



```
class Solar(Source):
    data: SolarData
    config: SolarConfig


class SolarConfig(SourceConfig):
    capacity: int
    max_capacity: int
    latitude: float
    area: float

class SolarData(SourceData):
    forecast_generation: DataBackend
    actual_generation: DataBackend
```

```
class BatteryConfig(StorageConfig):
    initial_soc: float
    capacity: float
    degradation_factor: float
    depth_of_discharge: float
    soc_minimum: float
    battery_cost_per_kwh: float
    reduction_coefficient: float
    charging_efficiency: float
    discharging_efficiency: float


class PHESConfig(StorageConfig):
    initial_energy: float
    efficiency: float
    pump_power_per_unit: float
    generation_power_per_unit: float
```

(a) Solar entity (b) Battery entity

Figure 2: Example abstractions defined in `EnCortex`.

3 `EnCortex` Overview

`EnCortex` is an end-to-end framework for sustainable operations and planning in new-age energy systems. At a high level, to support `operations`, live data from entities such as solar, wind farms, consumer demands are connected to the framework and the corresponding actions for optimal control with respect to a specific objective, e.g., profit maximization are sent back to the entities, e.g., battery actions, curtailment actions for solar/wind, etc. These are then handled by the associated SCADA systems of these entities and executed on the appropriate hardware. For `planning` tasks, arbitrary entities like additional customers, and energy generating plants (solar, wind, hydro, etc.) can be added to the scenario and the effect on revenues/penalties can be estimated. We provide various functionalities to support realistic implementation and detailed what-if analysis for such planning scenarios. We now describe the components and the architecture of `EnCortex`.

3.1 Core components

`EnCortex` has three core components, *viz.,* entities, contracts, and decision units.

3.1.1 Entities

Entities are the backbone of our framework. An entity could be any component that consumes/generates electricity. At a high level, each entity has four key attributes: (i) Data: It is a data structure containing various data fields related to the component, (ii) Actions: defines how to implement supported actions associated with the component, (iii) Configuration: to support config-based initialization and (iv) Schedule: to define time-based deadlines for each component. Entities are:

❶ **`Sources:`** A source is any component that is capable of generating electricity. Solar, Wind, Thermal, Nuclear, Hydro, etc., are all examples of the source class. Sources have *forecast_generation* and optionally *actual_generation* attributes as a part of the data field. In some cases, actions might be associated with sources. Figure 2 shows an example and Appendix A has additional details.

❷ **`Consumers`**: A consumer is a component that consumes energy, e.g., a household, campus, shopping mall, etc. Similar to sources, consumers are a majorly data-only entity, but actions can also be associated with a consumer in case of demand response.

❸ **`Markets`**: A *market* is an entity where energy can be sold or purchased. Typically, this is done through a bidding/continuous auction process, which defines the action space of a market entity i.e. simultaneous price bidding and volume allocation. Different markets across the world serve different purposes, e.g., a day-ahead market is used to provide an estimate of energy availability one day ahead of time. A detailed discussion of the types of markets can be found here [29]. To model all these types of markets, we simplify the behavior down to a schedule definition. Corresponding to each market, there are periods of time where the bidding is open. The end of this window is the deadline before which certain values have to be committed to a market. We use a cron-style definition of schedule to define the number of variables that have to be predicted before the deadline. See Appendix B for an example of the schedule definition.

❹ **`Storage`**: The storage class is inherited from a source class. The only exception here is that negative generation values are allowed. The configuration includes factors related to charging rates, degradation rates and capacities. Battery Storage Systems (BSS), Pumped Hydro Systems (PHS) are all modeled as storage entities (see Figure 2) thus showcasing the extensibility feature of the provided abstractions.

❺ **`Producer`**: A producer is associated with multiple entities which have contracts with various consumers and markets.

These entities can be either, *offline* (has representative data in file/database), *online* (capture streaming data from end devices) or *simulated* (physics-based models to generate data).

3.1.2 Contracts

A *contract* is required to define the flow of energy between two entities in the framework. A source supplying energy to a consumer should have a contract with the fields - *min_supply*, *max_supply*, and a penalty function. A penalty function is a function of supply and demand over a time horizon. For example, a linear penalty is applied with the deviation in the promised supply at every instant, i.e., let's say for every 15 minutes, if the demand is more than the supply, a fixed cost of $P\$/kWh$ is levied on the source. The penalty function is a combination of the penalty values over several time horizons.

3.1.3 Decision Units

Based on the defined entities and contracts associated with a particular producer, we identify subsets of entities and contracts where the decision-making is dependent on each other. We use a graph representation of entities as nodes and contracts as edges to identify decision units. A decision unit

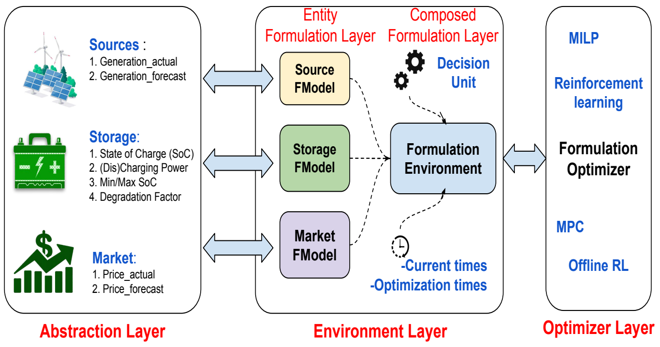


Figure 3: Overview of `EnCortex` architecture.

generates critical information on the schedule and the associated actions based on the included contracts/entities.

In Section 5 we show how these components are used to define and compose different new-age scenarios.

3.2 `EnCortex` Architecture

Our framework has three layers (Figure 3), which are used for composing new-age scenarios of different complexities:

1. Abstraction Layer: An abstraction layer uses entity-specific configurations to define entity behavior required for a scenario. It uses the framework provided unified data handling capabilities (Section 4) to handle data from all types of underlying objects, be it file-based, a web endpoint, or custom instrumentation in a live plant. Essentially, the abstraction layer takes care of all the data associated with a particular entity and simulation of recommended action which is based on the physics of the entity.

2. Environment Layer: The key purpose of this layer is to provide data and state information from entities(called state space) that are needed to make a decision and a central point to orchestrate all the required decisions based on the schedule (termed as action space). The environment layer has an entity-specific component called the *Entity Formulation Layer* which exposes the micro variables like volume output, emission output, etc. in a uniform API to the environment. The decision unit identified (Section 3.1.3) is used as an input along with the entity formulation layer and a user-specified optimization horizon to generate the formulation environment. This layer identifies the time period for which models have to be built, maintains the current time in simulation and incorporates all the variables and actions that are required to solve the optimization for this particular time period. This environment composes the scenario based on the decision unit under consideration (called the *Composed Formulation Layer*) to expose the macro variables like revenue, penalty, transmission penalty, emissions, etc. When an action is executed in simulation, the agents receive a reward indicating how they've performed on a singular instance of the objective function that needs to be maximized. Appendix C includes details of entity formulation and composed formulation layer.

3. Optimizer Layer: An optimizer layer finally uses the environment layer outputs (revenue, penalty, transmission penalty, emissions, etc.) for every timestamp and runs the optimization based on the objective defined (e.g., monetization). `EnCortex` provides native support for various state-of-the-art algorithms apart from bringing your own optimizer code. The energy operator can run numerous experiments to determine which algorithm provides the best optimal schedules while maximizing the objectives. The supported algorithms include Simulated Annealing (SA), Mixed Integer Linear Programming (MILP), and Deep Q Network-based Reinforcement Learning (DQN-RL) (see Appendix D for details). *In this work, we have developed adaptations to the above algorithms to be compatible for new-age energy scenarios (Section 5).*

4 `EnCortex` Infrastructure Components

`EnCortex` is developed as a modular, easy-to-use, and ready-to-deploy framework. We now describe the infrastructure components developed to support these functionalities.

Storage Backends: The storage backend module helps keep a centralized data API to support data operations across various abstractions, entities, and scenarios. `EnCortex` supports two storage backends that inherit `DataBackend`- an abstract class to deal with various `CRUD` operations, *viz., (i)* *`DFBackend`:* This is built on top of `pandas`'s `DataFrame` and `Series` API to support offline, tabular datasets. *(ii)* *`DBBackend`:* The `DBBackend` helps connect to various relational databases - SQLite, MySQL, etc., to access data.

Pipelines: Each energy operator might have their own set of operational and deployment constraints. Depending on the data availability, frequency of incoming data, SLA requirements, and choice of optimizers, orchestrating pipelines becomes crucial. In `EnCortex` we have developed `Runners` to help in orchestrating time-based decisions and are designed to run independently on separate threads on the same machine or on different machines altogether. Broadly there are two types of runners, *(i) Scheduled Runner* enables running optimization on a fixed schedule, as a co-routine. Scheduled runners are best suited when data is *assumed* to be available or when SLAs must be strictly adhered to. *(ii) Event-based Runner:* In scenarios where data is received from a third-party, an event-based runner helps to run optimizations that are triggered when new data is received.

DataLoaders: DataLoaders provides a uniform API to load datasets available publicly or versioned datasets stored on the cloud. The dataloaders are built on top of storage backends to have consistent data handling across the framework. These data loaders are pythonic and support both iterators-based data loading for batch-based operations as well as a query-like interface for sequential data-loading. We also add cloud-based support to load datasets through various frameworks like Ray [28] and Dask [35].

Forecasting Models: Decisions within the framework are based on data received from forecasting models/services. We have developed a unified forecasting module, that supports numerous state-of-the-art time-series forecasting techniques and also operators can bring their own forecaster model and integrate it. We support a wide range of techniques from simple heuristics to state-of-the-art algorithms, i.e., *(i) Noise forecast:* gaussian noise is added to actual data , *(ii) Yesterday's forecast:* previous day's actual data as forecast for the next day, *(iii) Mean:* mean of actual data for previous N days as forecast, *(iv) LightGBM [25]*, a decision tree based algorithm that builds a tree-like structure to perform regression and *(v) N-BEATS [31]*, a purely neural network-based model.

EnCortex Configuration and CLI: To instantiate the abstractions supported by `EnCortex`, we have built a command-line utility (CLI) that takes configuration files as input. We utilize Hydra [40], which allows us to modularize configurations of different entities allowing us to define a config-based execution that can be overridden using configuration files.

Visualization: Visualizing and tracking various performance metrics and decisions helps showcase the decisions and capabilities of various agents to energy operators. Therefore, we provide a command-line interface to launch a browser-based dashboard to understand scenario composition, plot, and track various optimizers' performance and analyze the rewards/penalties on different datasets and time periods.

Machine Learning Operation: When deploying neural network-based optimizers, model deployment, reproducible pipelines, and other principles of Machine Learning Operations (MLOps) for efficient workflows are critical. We have built-in support for different MLOps such as model and dataset versioning, experiment tracking, and reusable pipelines.

Callbacks: Callbacks are designed in `EnCortex` to provide an additional level of control and modify various aspects of the pipeline. The key callbacks are: *(i) Logging and Monitoring callbacks* captures detailed logs to efficiently monitor and debug the pipelines. *(ii) Closed loop callbacks* allow energy operators to connect the decision/action derived by the optimizers with physical controllers to execute the decisions made. *(iii) Connect to other applications callback* facilitates other applications/scenarios to easily build on top of `EnCortex`. *(iv) Safety handling callbacks* provide easy-to-use callbacks that are triggered during potential safety warnings to help energy operators to analyze and debug the issues quickly.

5 Real-world New-age Energy Scenarios

We now present a few prevalent new-age energy scenarios based on discussions with our global industry partners. We show how `EnCortex` framework is employed to compose, analyze and derive optimal decisions.

5.1 Scenario 1 - Energy Arbitrage (EA)

The variability and uncertainty associated with renewable sources can cause reliability issues in the energy grid. Thus new-age scenarios now include an energy storage system to assist in supply and demand mismatch. Energy Arbitrage (EA) can be simply defined as purchasing more electricity during Off-peak periods, storing that electricity via energy storage systems, and discharging it during Peak periods.

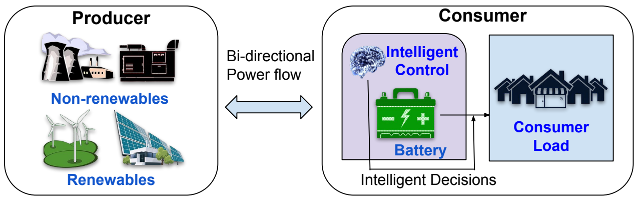


Figure 4: Overview of Energy Arbitrage Scenario.

As described in Section 2.1, our partner is deploying Li-ion batteries at their consumer premises for the following objectives (see Figure 4), **(i) maximize economic profit (cost savings):** exploit price variations to increase savings by charging/discharging the battery at off-peak/peak price periods; **(ii) maximize environmental impact (carbon footprint):** exploit generation mix (typically a combination of renewable and non-renewable sources are used to produce electricity) to reduce overall carbon footprint by charging/discharging the battery based on usage of renewable/non-renewable energy sources; **(iii) maximize both economic profit and environmental impact.** Thus, it is important to optimize the management and scheduling of these batteries, specifically, when to charge or discharge to maximize the given objective.

EA is emerging as a very crucial scenario to address the hour-to-hour variability in wind and solar, especially given the rapid increase in their adoption [3, 29]. EA can be framed as a maximization problem for cost and carbon savings. There exist several point solutions [12, 20, 32], that have mapped this to traditional optimizations such as Markov Decision Process (MDP) [12] and Mixed Integer Linear Programming (MILP) [32]. However, these are developed in silos, cannot be reused, and developing a solution from scratch is cumbersome and time-consuming. We now describe how `EnCortex` is employed to swiftly compose, analyze and derive optimal battery schedules. We describe this process by going through the three core layers.

5.1.1 Abstraction Layer

First, the energy operator determines the entities involved in this scenario and uses the framework provided abstractions (see Section 3.1.1). In EA there are two entity types,

Battery entity: We inherit the storage class to define a Li-ion battery entity. This includes the following core configurations: (i) battery initialization parameters such as charging rates, degradation rates, state of charge (SoC), capacity, and depth of discharge, etc., (ii) *get_state* returns the state of the battery i.e current SoC of the battery, (iii) *get_action_space* defines the possible actions a battery entity can take. In this scenario, we define three battery actions: *charge at max rate, discharge at max rate or stay idle.* The energy operator populates these values based on their battery configuration. In Section 6 we describe the battery configuration values used.

Battery degradation: Since batteries perform a limited number of cycles during their lifetime, we consider an accurate battery degradation model to model the battery's lifetime. We use a degradation coefficient which gets updated based on the capacity difference and the discharging cycles over a certain degradation period [12].

Simplified real-time market entity: Since EA does not require any bidding decisions in the market, we modify the real-time market entity to a simplified real-time market entity that captures just the real-time market prices along with the carbon footprint information. This shows the utility of our abstractions, which allow seamless modification/extension of the definitions based on the scenario.

5.1.2 Environment Layer

In this layer, the operator composes the actual EA scenario by defining the contract among the entities along with the decision unit to determine the optimized actions per entity.

Decision Unit: In this scenario, the decision unit includes the contract between the battery and the market entity. The decision unit here formulates state space as SoC of all involved storage entities and forecast price data of the market entity. In EA, we do not consider any penalty with the assumption that the consumer demand will always be met. Further, the decision unit orchestrates the actions that needs to be taken for the entities. The actions determine what the battery entity should do (i.e., either charge or discharge or stay idle) and also updates battery parameters such as SoC, degradation, etc., based on the action taken.

5.1.3 Optimizer Layer

EA scenario is formulated as a multiple objective optimization problem, with weights associated to maximize different objectives wrt cost savings, carbon footprint, and degradation. *Thus, at every timestep the agent decides whether it should charge the battery or discharge the battery or do nothing based on the forecasts of price and carbon footprints.* The operator can now run either cost or carbon or both optimizations using any of the supported algorithms to derive the optimal battery schedules and analyze the overall savings. We now describe the overall methodology for EA.

(i) Uncertainties involved: In EA, uncertainties are involved in estimating future energy market prices and carbon footprint. This is very crucial information, which the optimizers will use to derive optimal schedules. As described in Section 4 we support multiple forecast data with different error rates to perform detailed analysis.

(ii) Optimization: The objective in EA is to either maximize cost savings or carbon footprint or both by taking into account battery degradation cost. We define R_t^{price}, R_t^{carbon}, R_t^{deg} as the cost, carbon footprint savings and degradation reward for a particular timeslot,

$$R_t^{price} = p_t * (C_t Pmax_{ch} + D_t Pmax_{dis}) * \Delta t \tag{1}$$

$$R_t^{carbon} = c_t * (C_t Pmax_{ch} + D_t Pmax_{dis}) * \Delta t \tag{2}$$

$$R_t^{deg} = \alpha_t * |C_t Pmax_{ch} + D_t Pmax_{dis}| * \Delta t \tag{3}$$

where p_t and c_t denotes the scaled actual price values and actual carbon footprint values, $Pmax_{ch}$ and $Pmax_{dis}$ are the maximum battery charging ($Pmax_{ch} <= 0$) and discharging ($Pmax_{dis} >= 0$) power values, C_t is the binary charging action taken at time t and D_t is the binary discharging action taken at time t. Δt signifies the time for which the battery delivers the power continuously. α_t is the degradation coefficient which gets updated based on the capacity difference and the discharging cycles over a certain degradation period.

Thus, the overall objective R_{net} can be written as:

$$R_{net} = max \sum_{t=1}^{T} (\omega_{carbon} R_t^{carbon} + \omega_{price} R_t^{price} - \omega_{deg} R_t^{deg}) \tag{4}$$

where $\omega_{carbon}, \omega_{price}$ denotes weights of the carbon and price optimization, respectively. ω_{deg} denotes the importance given to the degradation in the battery model.

The optimizers need to abide by certain constraints to update the battery SoC. If the resulting charging/discharging action decisions do not follow these, a hard penalty is observed. Eq. 5 validates if the formulated SoC for the next timestamp is within the defined limits and also maintains the SoC at the end of the day to an SoC level greater than SoC_{min}. Eq. 6 and 7 updates the SoC level for the next timestamp.

$$1 - DoD <= SoC_{t+1} <= SoC_{max}; \qquad SoC_{T+1} >= SoC_{min} \tag{5}$$

$$SoC_{t+1} = SoC_t - \Delta SoC \tag{6}$$

$$\Delta SoC = (\eta_{ch} C_t Pmax_{ch} + \frac{D_t}{\eta_{dis}} Pmax_{dis}) \frac{\Delta t}{S_t} \tag{7}$$

where SoC_{max} is the maximum achievable SoC of the battery, SoC_{min} is the minimum SoC that needs to be maintained at the end of the day, DoD or depth of discharge represents percentage of battery capacity discharged, T denotes the end of the day and, η_{ch} and η_{dis} are the charging and discharging efficiencies of the battery, S_t represents the current storage capacity, SoC_t is the current state of charge of the battery and SoC_{t+1} is the state of charge of the battery for the immediate next timestamp. Our optimizers such as SA, MILP, and DQN-RL use the above objective function and constraints definitions to derive optimal battery schedules (see Section 7).

5.2 Scenario 2 - Optimizing Demand-side Strategies in a Micro-grid (MG)

A microgrid (MG) is a local, self-sufficient energy system that allows you to generate your own electricity along with control capabilities [19]. Within microgrids are a group of interconnected loads (consumer demand) and distributed energy resources such as solar and wind to generate power. In addition, newer microgrids also include energy storage, typically batteries [17]. When the grid goes down or electricity prices peak, MGs respond. In this scenario, we consider each MG is associated with a solar farm, battery storage, and consumer demand (an industrial setup). **The objective is to efficiently utilize the MG resources to satisfy consumer demand while maximizing cost savings.** In other words, the optimizing agent should derive optimal schedules to determine when to use power from the MG (energy from solar and battery) or the utility grid to match consumer demand.

MG is one of the popular new-age scenarios with different optimization objectives such as achieving the lowest prices, cleanest energy, or greatest electric reliability, etc. There are numerous challenges that need to be tackled in terms of management, planning, and optimal use of MGs. We now describe how EnCortex is used to define and compose an MG scenario.

5.2.1 Abstraction Layer

EnCortex supports all the entities present in an MG. Specifically, in this scenario, MG is composed of Solar, Battery, and Consumer entities, and the grid is represented with a simplified real-time market for energy pricing.

Solar entity: This is inherited from the source class and includes the following configurations: *(i) maximum capacity:* Total capacity of the solar farm, *(ii) current capacity:* solar generation in a particular time slot, and *(iii) solar profile data:* provides solar generation data either actual or forecast.
Consumer entity: This abstraction includes the $demand\_forecast$ and $demand\_actual$ attributes.
Battery and simplified real-time market abstractions remain the same as described in Scenario 1.

5.2.2 Environment Layer

Decision Unit: Unlike in scenario 1, where there exists just one contract between market and battery, here there are several contracts namely between, real-time market and MG, consumer and MG, MG and solar farm, and MG and battery.

The decision Unit takes in all the contracts between the defined entities as input and determines the action space for every timestep. The action space includes battery charge/discharge.

5.2.3 Optimizer Layer

The MG scenario is formulated as a maximization problem of economic profits for the consumer. Thus, at every timestep the agent decides whether it should charge the battery or discharge the battery or do nothing based on the forecasts of price, solar generation, and consumer demand. We now describe the overall methodology:
(i) Uncertainties involved: The uncertainties are involved in solar generation data, market price, and consumer demand.
(ii) Optimization: The optimizers take care of the uncertainty in the forecasts by using the forecast values when training and employing the actual values to get the rewards on the objective. They learn the relation between the actuals and forecasts and generate optimal decisions for the MG. The objective in this scenario is to maximize profits for the consumer. Mathematically, we define R_{net} as the price savings for a particular time slot, $R_{net} = max\sum_{t=0}^{T} -(Price_t E_t^{Ugrid})$, where $Price_t$ is the scaled price value between [-1,1] and E_t^{Ugrid} is the energy supplied by the main Utility Grid. The constraints capture battery SoC updates (similar to scenario 1) and ensure instantaneous energy conservation i.e., the total energy supplied to the consumer should be equal to the total energy consumed every timeslot. Note that, one can easily extend scenario 1 to define the MG scenario (scenario 2) by adding additional entities and constraints, owing to the general abstraction, extensible, and composable design of EnCortex.

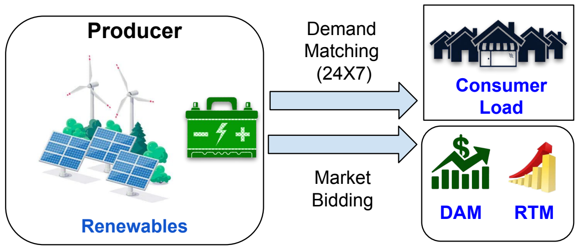


Figure 5: Demand matching and Bidding overview.

5.3 Scenario 3 - 24/7 Demand Matching and Bidding Optimization (BO)

The primary challenge with renewable sources is the uncertainty in the forecast for energy generation. Also traditionally, this uncertainty has existed at a longer time frame (a few days) due to supply chain dependencies in raw materials such as coal. This shift from a longer time frame uncertainty to higher uncertainty in a shorter time horizon has had several market implications. The market structure to support these uncertainties among others includes real-time and day-ahead markets, which offer the sale and purchase of energy for each hour of the following day. Further, for maintaining contracts with consumers with their own typical demand curves, there is also a need for use of storage sources - e.g., Li-ion batteries for a smaller time frame and Pumped Hydro Energy Storage (PHES) for a longer time frame.

As shown in Figure 5, we demonstrate a situation where weather-dependent sources - solar and wind - along with a PHES storage model are used to (i) meet the demands of a consumer contract with daily matching of supply-demand, (ii) monetization of excess energy available from the sources by bidding into energy markets. We consider two market scenarios, *viz.,* day-ahead (DAM) and real-time markets (RTM).

In the presence of the uncertainty, bidding into the multiple available markets based on the price and generation forecasts, and operating the battery storage systems has become a nontrivial operations problem for operators. While many bespoke solutions [18, 39] have been proposed in the literature, our primary distinction comes from the flexibility that the framework provides in defining and optimization for the scenario using the layered execution structure discussed in Section 3.2. In comparison, our framework automatically provides support for the bidding interval identification based on market definition and incorporates the contract penalty. The committed bids and the corresponding penalty variables are maintained as system states by the framework in the environment layer and can be accessed directly by the optimizer.

5.3.1 Abstraction layer

This scenario includes *two entity sources* - solar and wind, which simply generate power based on a forecast. A *battery entity* operates on fixed charging/discharging power. The decision has to be taken for every 15 mins timestep on whether the battery would charge/discharge at this time. The abstraction layer also contains *two market entities* - a day-ahead market entity has a bid window of 2 hours from 12 noon - 2 pm every day, where 96 slots of 15 mins each for the next day have to be bid. The real-time market (RTM) has a bid window every half hour where bidding for 2 slots 1-hr ahead of the current time is made. The two markets are used to monetize the excess power generated. A safer bid is made in the DAM market and then the balance is operated with the real-time market.

5.3.2 Environment Layer

Decision Unit: In this scenario, there exist contracts between the sources, markets, and batteries. The entity formulation layer of the markets contains decision variables/actions as per the number of slots to bid for. The *update* routines update the number of values that are committed, scheduled, or delivered. Based on this information, the contract penalty and the DSM penalty are estimated and used for optimization. The optimization slots for which the models must be built for all entities before composition are determined through the decision unit. For example, at 2 pm, the decisions taken, affect the 96 slots for the next day as well as the slots for 3, 3:15 pm in the same day. The environment automatically identifies these as non-overlapping decisions and builds models for the current day and the next day appropriately. The composed environment layer now exposes the revenue, penalty, and transmission penalty variables to the optimization layer.

5.3.3 Optimization Layer

The optimization layer maximizes the profit for the producer based on the exposed revenue, penalty, and variables from the environment layer for every set of optimization times. The uncertainties include price variations in DAM and RTM markets, and energy generation from solar and wind.

Market bidding optimization is a prevalent new-age energy scenario and we showed that the modular design of `EnCortex` framework enables the composition of such complex scenarios with multiple sources and markets.

6 Experimental Setup & Implementation

We implemented `EnCortex`, including core components (entities, contracts, decision units) and core layers (abstraction, environment, optimizers) in >10K lines of python code. The codebase along with ready-to-consume notebooks is made available[1]. We will now describe the datasets and configurations used to evaluate the three scenarios.

1. Energy Arbitrage Scenario

Datasets: We use public datasets from UK and US (California) to obtain hourly price (in euros or dollars) and carbon footprint (in gCO2eq) information. The UK data is obtained from National Grid ESO [5] for the years 2018, 2019, and 2020, and US data from CAISO [1] for 2018 and 2019.

Entity configurations: The battery configurations obtained by our industry partners are below. For UK setup, we use a 10kWh capacity, of efficiency 1.0, with a depth of discharge

Table 1: Variations and complexity of `EnCortex` for pre-defined scenarios with E= Energy Source, S= Storage, M= Market, C= consumer demand.

Scenario	E	S	M	C	Env	Max actions	State space
Scenario 1- BA	–	✓	✗	–	Fixed	S	H
Scenario 2 - MG	✓	✓	✗	✓	Fixed	$S+\frac{M*T}{d}$	H
Scenario 3- BO	✓	✓	✓	✓	Dynamic	$S+\frac{M*T}{d}+C$	$H*(1+M+C)$

of 90%, having an initial state of charge of 0.5. For CAISO, we use 50kWh capacity with an efficiency of 1, having an initial state of charge of 20%, with a 100% depth of discharge.

2. Microgrid Scenario

Datasets: We use the public Schneider Electric Microgrid dataset to evaluate scenario 2. The dataset includes 70 industrial sites data, with each site having a solar farm, battery, and consumer demand data every 15 minutes. The dataset also provides battery configurations per site along with electricity price information. More details can be found here [17]. *Entity configurations:* We use the same configurations provided in the dataset to populate solar, battery, market, and consumer entity abstractions.

3. Market Bidding Optimization

Datasets: We use India Energy Exchange (IEX) dataset [4] to obtain day-ahead (DAM) and real-time markets (RTM) price information.

Entity configurations: The source and battery configurations are similar to scenarios 1 and 2. The market configurations are similar to IEX regulations [24].

7 Results

We now present detailed results to showcase the core features of `EnCortex` framework to support new-age scenarios.

7.1 Support for composabilty

To demonstrate the power of abstractions and composability of `EnCortex`, we have developed a few real-world pre-defined scenarios using the framework as discussed in Section 5. Table 1 demonstrates the variations and increasing level of complexity of these scenarios. As shown in the Table, the different scenarios compose different combinations of Energy Sources-like solar, wind etc., Storage like battery vs. hydro storage, different forward bidding markets, and consumer side participation or control. Also, for the scenarios of battery arbitrage and Microgrid optimization, there is single optimization is performed per time step. However, in Market bidding scenario, there are multiple levels of cascading optimization decisions that needs to be performed for committing to the day-ahead, intra-day and real-time markets, battery actions and allocation decisions (dynamic environment), thereby expanding the state-space and action space of decisions. This is illustrated by the state and action space sizes per Scenario. H represents the historical time intervals data included in the state space, T indicates the time interval over which decisions are made, and d indicates the optimization time step.

Summary: `EnCortex` can be used to quickly formulate and compose increasingly complex scenarios efficiently.

[1] `EnCortex` Code available at https://anonymous.4open.science/r/encortex/

Table 2: Savings observed for scenarios 1-3 using different optimizers supported by `EnCortex`.

Optimizers	Scenario 1 - EA		Scenario 2 - MG	Scenario 3- BO
	Cost savings (K EUR)	Carbon savings ($kgCO_2eq$)	Cost savings (K EUR)	Profits (M INR)
SA	54.5	102.5	4732.5	NA
MILP	**88.1**	**163.3**	4738 .1	581.3
DQN-RL	85.1	140.2	**4738.2**	NA

7.2 Support for different optimizers

Table 2 shows the savings obtained for all three scenarios using the optimizers supported by `EnCortex` Optimizer layer (discussed in Section 3) with forecast computed using the mean of the previous 10 days' data. For scenario 1, we used 1 year of test data (year 2020) across all optimizers and for DQN-RL we used 2 years of data (years 2018-19) for training. The result shows over 88K EUR in cost and 163 $kgCO_2eq$ carbon reduction can be achieved over a period of one year, which is equivalent to greenhouse gas emissions from 35000+ gasoline-powered passenger vehicles driven for one year. For scenario 2, we use 305 days of test data and 455 days of train data for DQN-RL. We show average cost savings of 4738K EUR across all 70 sites over the test period. For scenario 3, we use 1 year of day-ahead market data (year 2020) from IEX Market and use MILP to generate cost savings for a renewable farm. The results show a net income of 581.3M INR using MILP, with a profit of approximately 20.4M INR as compared to a heuristics baseline, which assigns volumes to markets with a greedy strategy to maximize profits after meeting certain consumer fulfillments without looking for the longer contract reward. In general, both MILP and DQN-RL have similar performances. In the next section, we show how the optimizer's performance changes as the forecast errors increase. Note, for the remaining sections, we will focus on Scenario 1 - energy arbitrage results and similar results can be derived using our framework for scenarios 2 and 3.

Summary: With native support for different optimizers `EnCortex` allows energy operators to analyze the decisions without worrying about the implementation details.

7.3 Support for uncertainty in forecasts

We explore the impact of uncertainty in the forecasts while optimizing for cost and carbon savings. Using MILP and DQN-RL as our optimizers on six forecasting methods, discussed in 4. From Table 3, we observe that the forecasting errors for price forecasting are relatively low (MAE<11) for all forecasting methods compared with the forecasting errors for carbon forecasting (with MAE >40). This is due to the fact that the time-series of carbon emissions lack seasonality and is often harder to predict. DQN-RL outperforms MILP in cost savings, owing to the training routine to learn and adapt to forecast errors. However, it performs similarly to MILP in carbon savings, mainly due to the fact that accounting for high uncertainty is a much more difficult task for DQN-RL and requires additional training data.

Table 3: Cost and carbon savings on different forecasting methods with varying mean average error (MAE).

Forecast Method	Forecasting MAE		Cost Savings (K EUR)		Carbon Savings ($kgCO_2eq$)	
	Price	Carbon	MILP	DQN	MILP	DQN
Accurate	0	0	116.1	**117.73**	**227.27**	211.59
Noise	19.82	19.08	44.64	**57.86**	123.93	**163.17**
Yesterday	9.70	42.19	77.74	**83.08**	**138.384**	129.37
Mean	9.94	48.13	**88.2**	83.24	**163.34**	135.06
LightGBM	11.88	38.33	55.89	**62.14**	124.4	**128.73**
N-BEATS	10.13	38.49	73.3	**76.71**	**145.82**	130.94

Summary: Neural network-based formulations outperform traditional algorithms in presence of forecast errors and sufficient training data to learn from. `EnCortex` provides easy support to evaluate outcomes of different optimizers on various available forecasting functions/algorithms.

7.4 Support for Extensibility

Table 4 demonstrates the extensibility support of `EnCortex` through ease of composition from the single objective (either cost or carbon optimization) to multi-objective (having joint cost and carbon optimization) results for scenario 1 using MILP and DQN-RL. Given the composable & extensible formulation as shown in earlier section, it is trivial for energy operators to extend the single objective functions to multi-objective through configuration. To solve the objectives, we leverage the market price data and carbon emissions data available from the UK grid. The DQN-RL agent is then trained on 2 years (2018-19) UK data (training data) and results are inferred on 1 yr (2020) data of the same UK grid (test data). We utilize the same 1 year's test data to produce results for MILP. Here, we show results for both accurate forecasts as well as forecasts by taking the mean of previous 10 days' actual data. We observe that when the forecasts are accurate, joint optimization results in cost and carbon savings that are close to the optimization results obtained by solving individual objectives. In both carbon as well as cost savings, the joint-optimization savings are lower by approximately 30% as compared to the respective individual objectives. Similarly with the forecast data, the numbers significantly reduce by approximately 43 - 47% while using joint optimization as opposed to the single objectives. In these experiments of joint optimization, we are using a weight factor ($\alpha = \frac{\omega_{carbon}}{\omega_{price}}$) of 1.155 and 1.28 for MILP and RL respectively. While any number can be used as a configuration parameter for α, to determine this optimal value we perform a detailed what-if analysis, described in the next section.

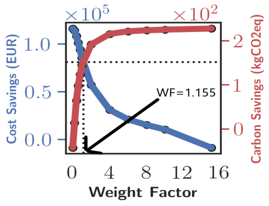
Summary: The savings reduce with the increase in the uncertainty of forecasts. `EnCortex` enables operators to extend existing scenarios through configurations.

7.5 Support for what-if analysis

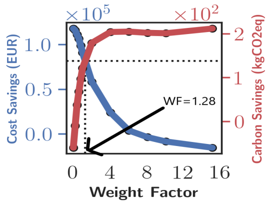
Pareto intersection results. As seen in the previous subsection, we use an optimum value of weight factor for joint optimization of carbon and price savings. To find this optimal point, `EnCortex` supports very easy configuration-based alternatives to search through different α values as a hyper-

Table 4: Savings comparison using single and multi-objectives for Energy Arbitrage scenario.

Optimizers	With Accurate forecasts		with Mean forecasts	
	MILP	DQN-RL	MILP	DQN-RL
Cost savings (K EUR)	116.1	**117.7**	85.1	**88.1**
Carbon savings ($kgCO_2eq$)	**227.2**	211.5	**163.3**	140.2
Joint OPT - Cost (K EUR)	74.8	81.5	46.2	24.9
Joint OPT - Carbon ($kgCO_2eq$)	164.7	137.7	135.8	92.6



(a) Pareto - MILP



(b) Pareto - DQN-RL

Figure 6: Optimal points for multi-objective problem setting

parameter in one go. The user is supposed to enter the configuration values or range required to be searched for, and the system runs and provides the final optimal value. Figure 6 shows the optimal point being determined to solve the multi-objective joint optimization of cost and carbon savings using MILP and RL respectively. The pareto-intersection of the price and carbon savings' curves varied across different α values to generate the optimal point.

Adding degradation and varying the associated cost. The carbon and price savings results for battery arbitrage scenario discussed in earlier sub-sections did not include the battery degradation over multiple charging/discharging cycles. We now use degradation as a part of what-if analysis. In Figure 7, we vary the degradation cost and see how the savings change. We performed the experiment across 3 different seeds using DQN-RL and have the confidence interval plotted in the figure. The plot shows that cost savings almost remains constant when degradation cost increases. This is due to the fact that the DQN-RL optimizer learns to perform actions that doesn't quickly degrade the battery capacity since we include the degradation coefficient in the reward function (see Eq. 4). As the battery degradation cost increases, the degradation coefficient decreases. The degradation coefficient being proportional to the change in capacity, nullifies the effect of degradation. Figure 8 shows how the capacity degradation varies for a 10KWh capacity battery over a year with the increasing degradation cost. As we can see, there is less capacity degradation with increasing degradation cost. Further, the number of frequent charging and discharging cycles drops by 13% when degradation cost is considered as compared to no degradation (477 cycles vs 551 cycles).

Summary: `EnCortex` supports various types of what-if analysis for energy operators through easy configurations, in order to evaluate and determine the optimal decisions.

7.6 Support for MLOps workflows

We now present a few key results showcasing the MLOps workflows supported by `EnCortex`. MLOps workflow allows

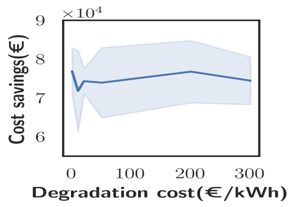


Figure 7: Savings vs Degrad.

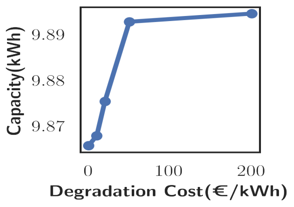
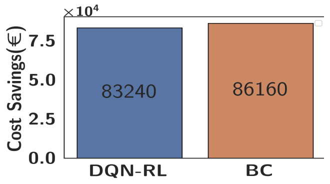
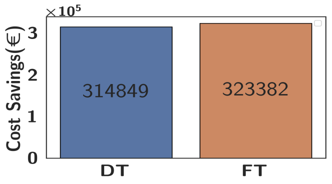


Figure 8: Capacity degrad.



(a) Behaviour Cloning



(b) Pre-training vs Direct

Figure 9: MLOps workflows.

energy operators to save different models and their trajectories to do what-if analysis or re-train on additional data, etc. To this end, we employ our public Azure Machine Learning (AML) service to build, deploy and perform various MLOps.

Expert RL agents to support lack of data: Often energy operators have multiple agents trained on several scenarios that are experts in their respective tasks. These expert agents with learned trajectories can be easily utilized for a new scenario. Let's say an energy operator requires installation of their energy resources in a new region/geography where the available data is sparse. In such cases, the expert agents trained on a different dataset can be re-used in neural network-based solutions for decision making and planning.

Specifically, the expert agent, instead of providing the child agent with the reward value functions, sets a list of demonstrations by sampling from expert transitions. The new agent tries to learn the optimal policies by imitating the expert's decisions. To support this, `EnCortex` employs imitation learning library [27] and behavior cloning (BC) models. We use 2 years of price data (2018 and 2019 from UK) to train both a DQN-RL and our expert agent. Then, we use 1-year data (2020 from UK) to show the savings obtained using DQN-RL (naive without any additional learning) and BC model (which uses information from trained expert trajectories). In Figure 9a, we can see that the BC model outperforms traditional DQN-RL, owing to the expert trajectories used to learn the observation-action pairs.

Use pre-trained models for new configurations and dataset: Energy Operators can also move from learning in one environment to another. Here, environment changes refer to testing the existing agents on a different grid with very different abstraction configurations. The intuition here is to bring out generalization across environments, even if the agents use different state representations or action spaces. Specifically, we use a pre-trained RL agent trained on UK data and retrain it for CAISO (US) data, and compare its performance against an RL agent that is trained only on CAISO data. Note that, the dataset size and battery configurations in these two datasets are different (see Section 6). Figure 9b shows that with pre-trained models we can improve savings by 3% on CAISO data as opposed to direct training. The pre-trained models help energy operators especially when there is sparse data. Appendix E discusses other workflows supported.

Summary: EnCortex supports several MLOps workflows, which allows operators to use and analyze different datasets and configurations for their planning and operations tasks.

7.7 EnCortex latency benchmarking

Within EnCortex, the optimizers are either exact solvers (MILP), search-based solvers (SA) or learnable optimizers (DQN-RL). We now present online latency values, which measure the speed of execution runtime on the hardware. The benchmarks are performed on a VM running AMD EPYC 7742 64-Core Processor with 996GB RAM and 4 A6000 RTX GPUs. The online latency for DQN-RL on CPU and GPU are $600\mu s$ and $4.4\mu s$, respectively. Further, the latency for MILP and SA are 0.42s and 195s, respectively. This benchmark provides runtime information required when choosing different optimizers for offline and real-time scenarios.

8 Limitations and Future work

With EnCortex, we have taken the first step towards building a general, extensible decision management framework for energy operators. We recognize that there are a few limitations/challenges in our current framework and addressing these would be an interesting future direction.

Seamless support for new state-of-the-art optimizers. EnCortex supports most of the traditional and neural network-based optimizers in the same environment. There may be a few new optimizers, which can be applied for energy scenarios as described in [9, 33]. That said, we believe one can extend the current environments to support more optimizers with fewer lines of code using existing framework utilities.

Recommendation engine to determine the best optimizer. While EnCortex supports running detailed what-if analysis across different optimizers, for application users it would be easier if the framework could recommend the best optimizer for a given scenario, objective, and constraints. It is non-trivial to build such a general recommendation engine as it requires significant data and decisions knowledge across scenarios and is an interesting future direction.

Safe-RL approaches for new-age energy scenarios. All new-age entities such as batteries, renewable sources, etc., support control features either to charge/discharge battery, curtail generation, etc [14]. Further, the decisions obtained in EnCortex are eventually sent to the physical entity hardware via their SCADA system. Any safety violation can lead to reliability issues in the power grid. For example, neglecting the crucial energy balance constraint could result in either under- or over-production leading to exceeding maximum power capacity [13]. Similarly violating minimum SoC constraints for batteries would degrade them significantly quicker than expected. Thus it is important for newer formulations to incorporate such safety constraints explicitly in the objective.

9 Related Work

Historically, bespoke solutions have been devised to tackle specific problems in optimizing energy systems. These methods are often times inextensible and hard to adapt to newer systems with different requirements. Previously, frameworks like ElecSim [26] and pymgrid [22] have tried to tackle individual scenarios like market bidding optimization and microgrid optimization respectively. These frameworks provide individual instances of implementing some variants of the scenario but individually suffer some drawbacks. Firstly, it is not trivial to extend ElecSim to scenarios that don't involve the market, because of the rigid structure it adopts when defining the market bidding scenario. On the other hand, while pymgrid supports variations of a microgrid, but it is not straightforward to extend it to other types of entities and forecasts, especially entities like Market that involve complex scheduling and decision requirements. Our proposed framework addresses each of these drawbacks by providing modular, extensible entities that are customizable based on requirements. Here, entities are designed around the principle that the core components are disentangled i.e. various domain experts (domain-experts, developers, etc.) can individually customize required modules without modifying unrelated components, enabling code reuse and better functionality. For example, we show it is easy to add a new storage device with a completely different underlying mechanism based on a different physics model. Furthermore, we also support composing a multitude of scenarios and optimizers to enable planning and decision-making.

10 Conclusion

Renewable energy sources are non-deterministic and pose additional complexity in periodic decision-making. While there are several optimization scenarios and approaches studied in the literature, those are designed and deployed as bespoke solutions and lack generality and extensibility. The whole ecosystem (including new energy resources, markets, incentives, and business models) is evolving fast. Hence, there is an ongoing demand for an easy-to-use system that would allow energy operators to compose, build, validate and deploy their real-world scenarios rapidly. In this work, we presented EnCortex, a generic, modular, extensible and scalable framework for new-age energy systems that enables data-driven decision-making and provides ease of use for development and production scale deployments. Through the evaluation on three real-world scenarios, we demonstrate the power and effectiveness of EnCortex.

References

[1] California iso outlook. http://www.caiso.com/outlook/SP/History/20220908/co2.csv?_=1663324325660.

[2] Demand-side management programs save energy and reduce peak demand. Homepage - U.S. Energy Information Administration (EIA) https://www.eia.gov/todayinenergy/detail.php?id=38872.

[3] IEA(2021) *Global Energy Review 2021*, IEA, Paris . https://www.iea.org/reports/global-energy-review-2021.

[4] Iex data portal. https://www.iexindia.com/marketdata/areaprice.aspx.

[5] Uk national grid eso portal. https://data.nationalgrideso.com/.

[6] Razan AH Al-Lawati, Jose L Crespo-Vazquez, Tasnim Ibn Faiz, Xin Fang, and Md Noor-E-Alam. Two-stage stochastic optimization frameworks to aid in decision-making under uncertainty for variable resource generators participating in a sequential energy market. *Applied Energy*, 292:116882, 2021.

[7] Muhammad Ali, Krishneel Prakash, Md Alamgir Hossain, and Hemanshu R Pota. Intelligent energy management: Evolving developments, current challenges, and research directions for sustainable future. *Journal of Cleaner Production*, 314:127904, 2021.

[8] Ariel Bergmann, Nick Hanley, and Robert Wright. Valuing the attributes of renewable energy investments. *Energy policy*, 34(9):1004–1014, 2006.

[9] Yuexin Bian, Ningkun Zheng, Yang Zheng, Bolun Xu, and Yuanyuan Shi. Demand response model identification and behavior forecast with optnet: a gradient-based approach. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 418–429, 2022.

[10] Lueder Von Bremen. Large-scale variability of weather dependent renewable energy sources. In *Management of weather and climate risk in the energy industry*, pages 189–206. Springer, 2010.

[11] Markus Burger, Bernhard Graeber, and Gero Schindlmayr. Managing energy risk: An integrated view on power and other energy markets. 2014.

[12] Jun Cao, Dan Harrold, Zhong Fan, Thomas Morstyn, David Healey, and Kang Li. Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model. *IEEE Transactions on Smart Grid*, 11(5):4513–4521, 2020.

[13] Glenn Ceusters, Luis Ramirez Camargo, Rüdiger Franke, Ann Nowé, and Maarten Messagie. Safe reinforcement learning for multi-energy management systems with known constraint functions. *arXiv preprint arXiv:2207.03830*, 2022.

[14] Shuai Chen, Chengpeng Jiang, Jinglin Li, Jinwei Xiang, and Wendong Xiao. Improved deep q-network for user-side battery energy storage charging and discharging strategy in industrial parks. *Entropy*, 23(10):1311, 2021.

[15] John E Doerr. *Speed & Scale: An Action Plan for Solving our Climate Crisis Now*. Penguin Audio, 2021.

[16] EIA. Eia: Renewables source. https://www.eia.gov/energyexplained/what-is-energy/sources-of-energy.php, 2022.

[17] Schneider Electric. The emsx dataset: historical photovoltaic and load scenarios and forecasts for 70 industrial sites. https://zenodo.org/record/5510400#.Yybc_ddBwZt, 2021.

[18] Pary Fazlalipour, Mehdi Ehsan, and Behnam Mohammadi-Ivatloo. Risk-aware stochastic bidding strategy of renewable micro-grids in day-ahead and real-time markets. *Energy*, 171:689–700, 2019.

[19] Kaiye Gao, Tianshi Wang, Chenjing Han, Jinhao Xie, Ye Ma, and Rui Peng. A review of optimization of microgrid operation. *Energies*, 14(10), 2021.

[20] Pedro Luis Camuñas García-Miguel, Jaime Alonso-Martínez, Santiago Arnaltes Gómez, Manuel García Plaza, and Andrés Peña Asensio. A review on the degradation implementation for the operation of battery energy storage systems. *Batteries*, 8(9), 2022.

[21] Shahrzad Hadayeghparast, Alireza SoltaniNejad Farsangi, and Heidarali Shayanfar. Day-ahead stochastic multi-objective economic/emission operational scheduling of a large scale virtual power plant. *Energy*, 172:630–646, 2019.

[22] Gonzague Henri, Tanguy Levent, Avishai Halev, Reda Alami, and Philippe Cordier. pymgrid: An open-source python microgrid simulator for applied artificial intelligence research. *arXiv preprint arXiv:2011.08004*, 2020.

[23] IET. Energy management: the foremost challenge for facilities managers. https://electrical.theiet.org/media/1162/energy-management-the-foremost-challenge-for-faciliti pdf, 2021.

[24] IEX. Regulations and procedures. https://www.iexindia.com/regulations_procedures.aspx?id=OZixKt2MNIs%3D&mid=Gy9kTd80D98%3D, 2021.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[26] Alexander Kell, Matthew Forshaw, and A Stephen McGough. Elecsim: Monte-carlo open-source agent-based model to inform policy for long-term electricity planning. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, pages 556–565, 2019.

[27] Imitation learning. Imitation learning baseline implementations. https://pypi.org/project/imitation/, 2021.

[28] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018.

[29] Natalia Naval and Jose M Yusta. Virtual power plant models and electricity markets-a review. *Renewable and Sustainable Energy Reviews*, 149:111393, 2021.

[30] AG Olabi. Renewable energy and energy storage systems, 2017.

[31] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[32] Andrés F. Peñaranda, David Romero-Quete, and Camilo A. Cortés. Grid-scale battery energy storage for arbitrage purposes: A colombian case. *Batteries*, 7(3), 2021.

[33] Luis Pineda, Taosha Fan, Maurizio Monge, Shobha Venkataraman, Paloma Sodhi, Ricky Chen, Joseph Ortiz, Daniel DeTone, Austin Wang, Stuart Anderson, Jing Dong, Brandon Amos, and Mustafa Mukadam. Theseus: A Library for Differentiable Nonlinear Optimization. *arXiv preprint arXiv:2207.09442*, 2022.

[34] REN21. Renewable energy data in perspective. https://www.ren21.net/wp-content/uploads/2019/05/GSR2022_Key_Messages.pdf, 2021.

[35] Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th python in science conference*, volume 130, page 136. Citeseer, 2015.

[36] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[37] Oliver Ruhnau and Staffan Qvist. Storage requirements in a 100% renewable electricity system: Extreme events and inter-annual variability. *Environmental Research Letters*, 17(4):044018, 2022.

[38] Fei Wang, Xinxin Ge, Peng Yang, Kangping Li, Zengqiang Mi, Pierluigi Siano, and Neven Duić. Day-ahead optimal bidding and scheduling strategies for der aggregator considering responsive uncertainty under real-time pricing. *Energy*, 213:118765, 2020.

[39] Fei Wang, Xinxin Ge, Peng Yang, Kangping Li, Zengqiang Mi, Pierluigi Siano, and Neven Duić. Day-ahead optimal bidding and scheduling strategies for der aggregator considering responsive uncertainty under real-time pricing. *Energy*, 213:118765, 2020.

[40] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

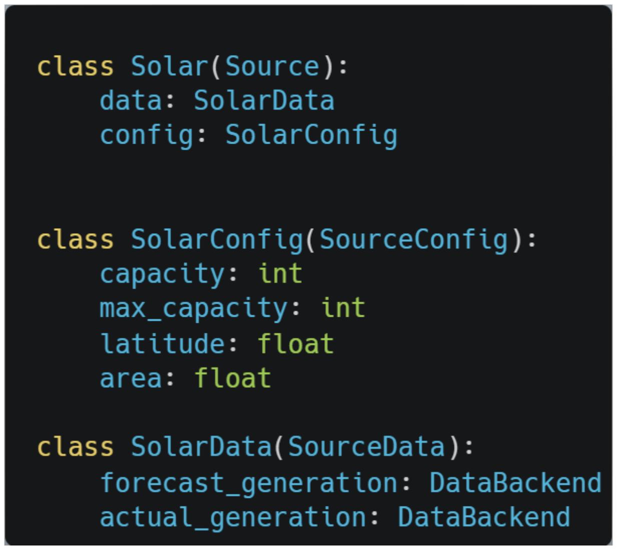
Appendix

A Entity Abstraction

Here, we briefly cover how to implement different energy sources. In Figure 10, the `Solar` class overrides the data and config attributes to support additional configuration parameters to implement the underlying phsyics-based formulations and contains a data attribute to access various solar specific power generation attributes. Figure 11 further illustrates how configuration parameters vary across storage devices(Battery and PHES) and the requirement of an easy-to-configure, disentangled module.

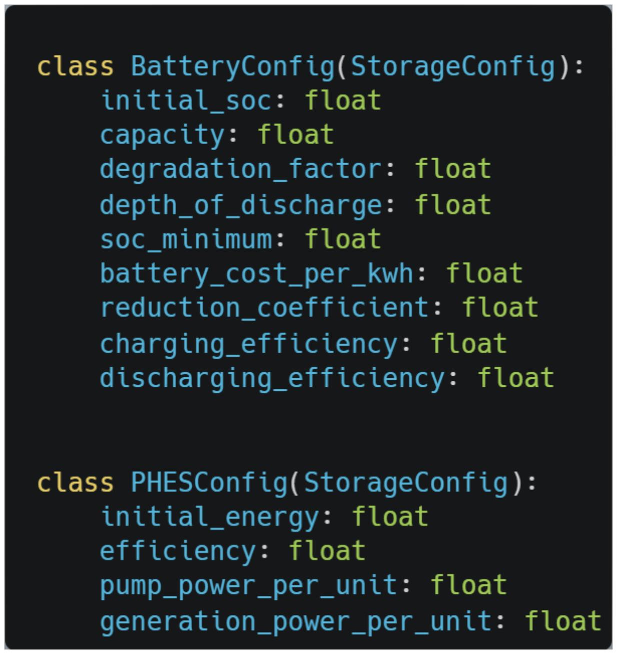
B Market Definition example

To model all these types of markets, we simplify the behavior down to a schedule definition (Figure 12). Corresponding to each market, there are periods of time where the bidding is open. The end of this window is the deadline before which certain values have to be committed to a market. Using the cron-style definition of schedule in Figure 12, we define the number of variables that has to be predicted before the deadline. For example, a day-ahead market has a bidding window from 12 noon - 2 pm every day where 96 values (bid volume, bid price) corresponding to 15-minute slots of the next day have to be defined. The schedule for this example is shown in



```python
class Solar(Source):
    data: SolarData
    config: SolarConfig


class SolarConfig(SourceConfig):
    capacity: int
    max_capacity: int
    latitude: float
    area: float

class SolarData(SourceData):
    forecast_generation: DataBackend
    actual_generation: DataBackend
```

Figure 10: Solar entity abstraction



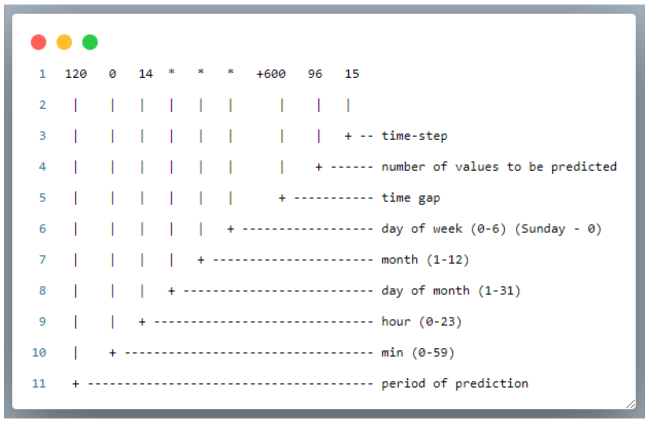
```python
class BatteryConfig(StorageConfig):
    initial_soc: float
    capacity: float
    degradation_factor: float
    depth_of_discharge: float
    soc_minimum: float
    battery_cost_per_kwh: float
    reduction_coefficient: float
    charging_efficiency: float
    discharging_efficiency: float


class PHESConfig(StorageConfig):
    initial_energy: float
    efficiency: float
    pump_power_per_unit: float
    generation_power_per_unit: float
```

Figure 11: Battery entity abstraction

Figure 12. The market models available in our framework currently assume that the bidders are price-takers, however, the framework supports extension of markets to a bid-sensitive price formulation. Continuous auction markets are currently not modeled and are a part of future work in the framework.

C EnCortex Environment Layer details

We now present the details of Entity and composed formulation layer by taking MILP as an example.



```
120   0   14  *   *   *   +600   96   15
 |    |   |   |   |   |      |    |   |
 |    |   |   |   |   |      |    |   + -- time-step
 |    |   |   |   |   |      |    + ------ number of values to be predicted
 |    |   |   |   |   |      + ----------- time gap
 |    |   |   |   |   + ------------------ day of week (0-6) (Sunday - 0)
 |    |   |   |   + ---------------------- month (1-12)
 |    |   |   + -------------------------- day of month (1-31)
 |    |   + ------------------------------ hour (0-23)
 |    + ---------------------------------- min (0-59)
 + --------------------------------------- period of prediction
```

Figure 12: A cron-style schedule definition is used for defining markets that support double-blind bids. The demonstrated values are for a market with a bid window of 120 mins ending at 1400 hours where the bid must be made for 96 slots of 15 mins each starting 400 mins after 1400 hours, i.e., 0000hrs+1D.

Entity Formulation Layer: MILP Example Let us consider an example of the layers for the MILP formulation. In this case, each of the entities expose *volume_out_var* functions. In the case of entities like solar/wind, this is simply the generation forecast numbers. In case of a battery entity, let C_t and D_t represent the binary variables whether the battery is charged/discharged at the time t, then the entity formulation layer consists of the following:

1. *model_build*: The model build routine initializes the binary variables, imposes constraints,

$$0 \leq C_t \leq 1, 0 \leq D_t \leq 1 \tag{8}$$

$$0 \leq C_t + D_t \leq 1 \tag{9}$$

   Since the model is built for $t \in [1, T]$, this function also specifies the operating rules, i.e.,

$$E_t = E_i + \sum_{t=1}^{t} (\eta_C C_t P_C - \frac{1}{\eta_D} D_t P_D) \tag{10}$$

   where η_C and η_D are charging and discharging efficiencies, P_C and P_D are the charging/discharging powers along with their constraints, e.g.,

$$SOC_{min} * cap \leq E_t \leq SOC_{max} * cap \tag{11}$$

   where SOC_{min} and SOC_{max} are based on the operational usage recommendations from the battery manufacturer.

2. *volume_out_var*:

$$\text{volume\_out\_var} = (P_D D_t - P_C C_t) \tag{12}$$

   where P_D and P_C are the discharging/charging power respectively.

3. *update*: An update routine interfaces with the underlying abstraction to update the underlying abstraction with solutions of the current optimization. For example, in this case, the battery state of charge is updated int the battery abstraction.

$$batt.soc_t = soc_{t-1} + (\eta_C P_C C_t - \frac{1}{\eta_D} D_t P_D) \quad (13)$$

Composed Formulation Layer: MILP Example With all the entities exposing the same interface, it is possible to write a generic composed formulation layer. Let **D** be the decision unit under consideration. Let **E** be the set of entities and **C** be the set of contracts in the decision unit. Let $c.C_r$ be the contractors for contract c, which is a list of sources for a particular contract and $c.C_e$ be the contractee for the contract c, i.e., market or consumer. Then,

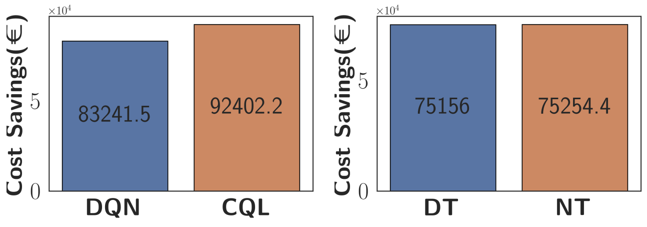
$$\text{revenue}_t = \sum_{c \in \mathbf{C}} (\sum_{s \in c.C_r} s.vout_t) * c.C_e.price_t \quad (14)$$

$$\text{penalty} = \sum_{c \in \mathbf{C}} [c.penalty\_fn((\sum_{s \in c.C_r} s.vout_t), c.C_e.cv_t)] \quad (15)$$

where *vout* and *cv* are volume_out (as obtained from the volume_out_var function in the entity formulation layer) and committed volumes respectively. The penalty_fn is a user specific function per contract based on which the differences in committment vs supply are penalised. In the framework, we provide provisions for accuring this for different time intervals, e.g., daily or for a fixed subset of hours per day.

D `EnCortex` supported optimizers

`EnCortex` supports the following optimizers:
(i) ***Simulated Annealing (SA)*** is a traditional probabilistic optimization technique, which approximates the global optimum for a given function.
(ii) ***Mixed Integer Linear Programming (MILP)*** is also one of the traditional optimization techniques, which solves a given objective based on certain constraints over a fixed given Horizon.
(iii) ***Model Predictive Control (MPC)*** is also a traditional technique for optimization which, unlike MILP, solves a sequence of multistage deterministic optimization problems over a given fixed horizon H.
(iv) ***Deep Q Network based Reinforcement Learning (DQN-RL)*** is a neural network-based algorithm. The environment, agent, state, and reward form the building blocks for any RL problem. The agent interacts with the environment by taking actions in the current state to proceed to the next state. Agents get rewarded based on the actions taken in the environment. They learn how to navigate through the environment based on the experience gained to reach the end goal by maximizing the cumulative reward for every episode. DQN, being model-free, solves the RL task directly using the samples provided by the environment, without constructing explicit estimates of them.



(a) Offline RL (b) Pre-training vs Direct
Figure 13: MLOps workflows.

E `EnCortex` supported MLOps workflows

We now present two specific workflows: **Exploit domain knowledge or other agents to learn better policies:** Energy operators have significant historical data in terms of decision-making either manually or using any traditional techniques. In cases where the data available to train the RL agent is sparse, one can use this historical decision dataset along with the new incoming data to train an offline RL agent. Offline-RL in principle leverages large, previously collected datasets to learn effective policies without the need to constantly interact with the environment as required by traditional RL.

To show this, we first use MILP to solve the objective functions of the scenario by maintaining the constraints on the historically available datasets. For example, here, we run MILP for the first 2 years of UK data and then, make our own Markov Decision Process-based dataset (MDPDataset) for the data-driven offline RL approach which is used like a supervised learning dataset. We leverage *conservative Q-learning* (CQL) provided by *d3rlpy*, a python-based offline RL library. Figure 13a shows that using Offline-RL the savings increases by 11% (83K to 92K EUR) as compared to training an RL agent from scratch (DQN-RL).

Use pre-trained models for new configurations: We now show how a pre-trained model with a specific action space can be re-used with a scenario with different action spaces. Specifically, we take two battery configurations, battery-1: 3 action spaces (charge at max, discharge at max or stay idle) and battery-2: 5 action spaces, in addition to the above three we have two new actions, i.e., (charge at half rate and discharge at half rate). Since the task is the same we finetune the pre-trained model with a 3-state battery configuration to target the task with a 5-state configuration. Figure 13b shows that with pre-training and near-transfer we can achieve significant savings as opposed to direct training.