Endesa Challenge:
Joint optimization of a Battery System and a Wind Farm

A hybrid quantum+classical MILP solver

Queen of fog

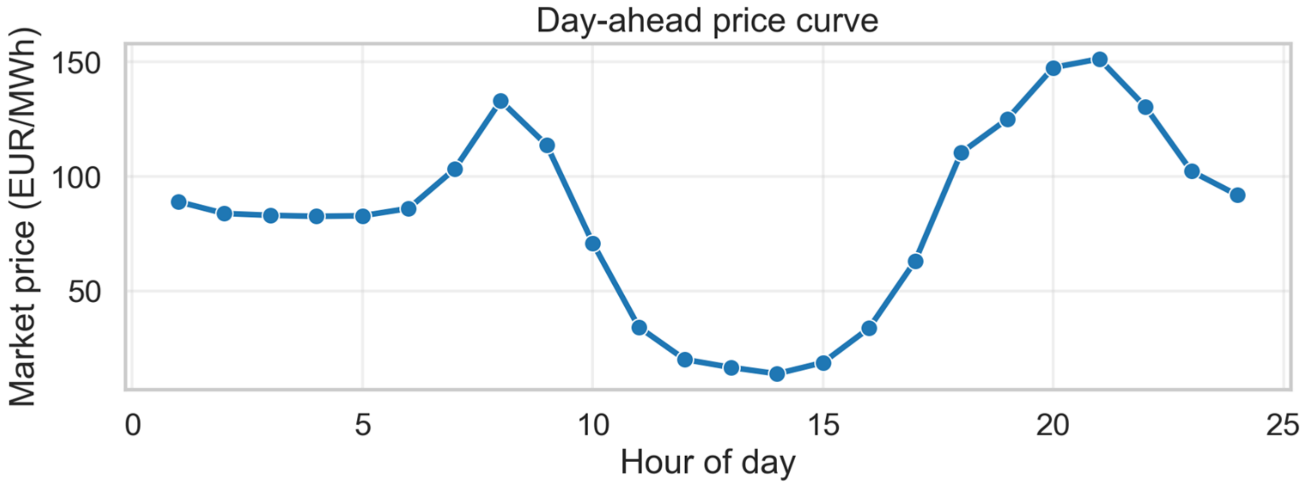
Cristian Tabares, Sergio Paniego, Jan Schneider,

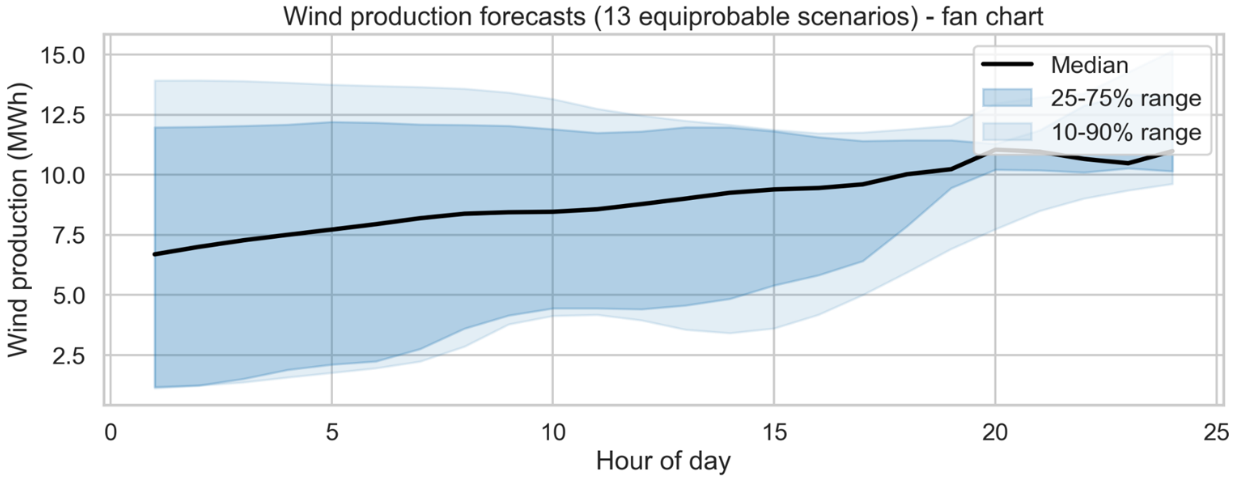
Joan Agustí, Yanis Le Fur

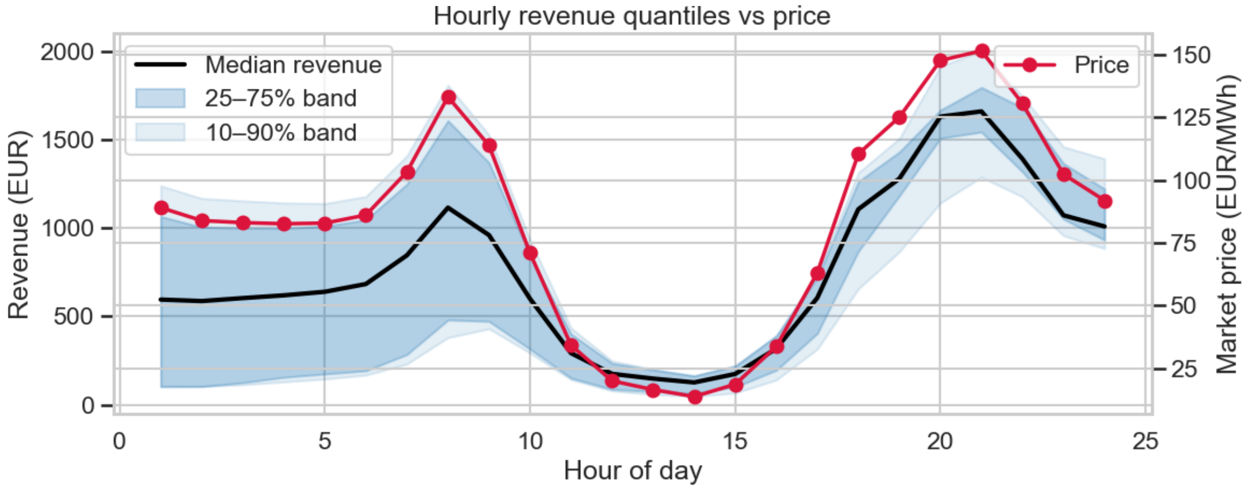
14/12/2025

The problem

Our objective: to maximize the joint revenue of a wind farm and a battery over a one-day horizon, modeling a battery energy storage system (BESS).
Our data: fixed electricty price & stochastic wind-production forecasts.







Expected revenue from wind farm: 17.645 €

Mixed-integer linear programs (MILP)

Our approach: the problem can be formulated as a MILP because maximizing revenue required optimizing continuous power flows while following strict discrete, binary decisions.

Expected revenue to maximize:

$$\max \sum_{s=1}^{13} \pi_s \sum_{t=1}^{24} p_t w_{t,s} + \sum_{t=1}^{24} p_t (d_t - c_t) - \lambda \sum_{t=1}^{24} (u_t^{\text{ch}} + u_t^{\text{dis}})$$

Continuous variable constraint:

$$E_t = E_{t-1} + \eta_{\text{ch}} c_t - \frac{1}{\eta_{\text{dis}}} d_t \qquad \forall t = 1, \ldots, 24$$

$$0 \le E_t \le E^{\max} \qquad \forall t = 1, \ldots 24$$

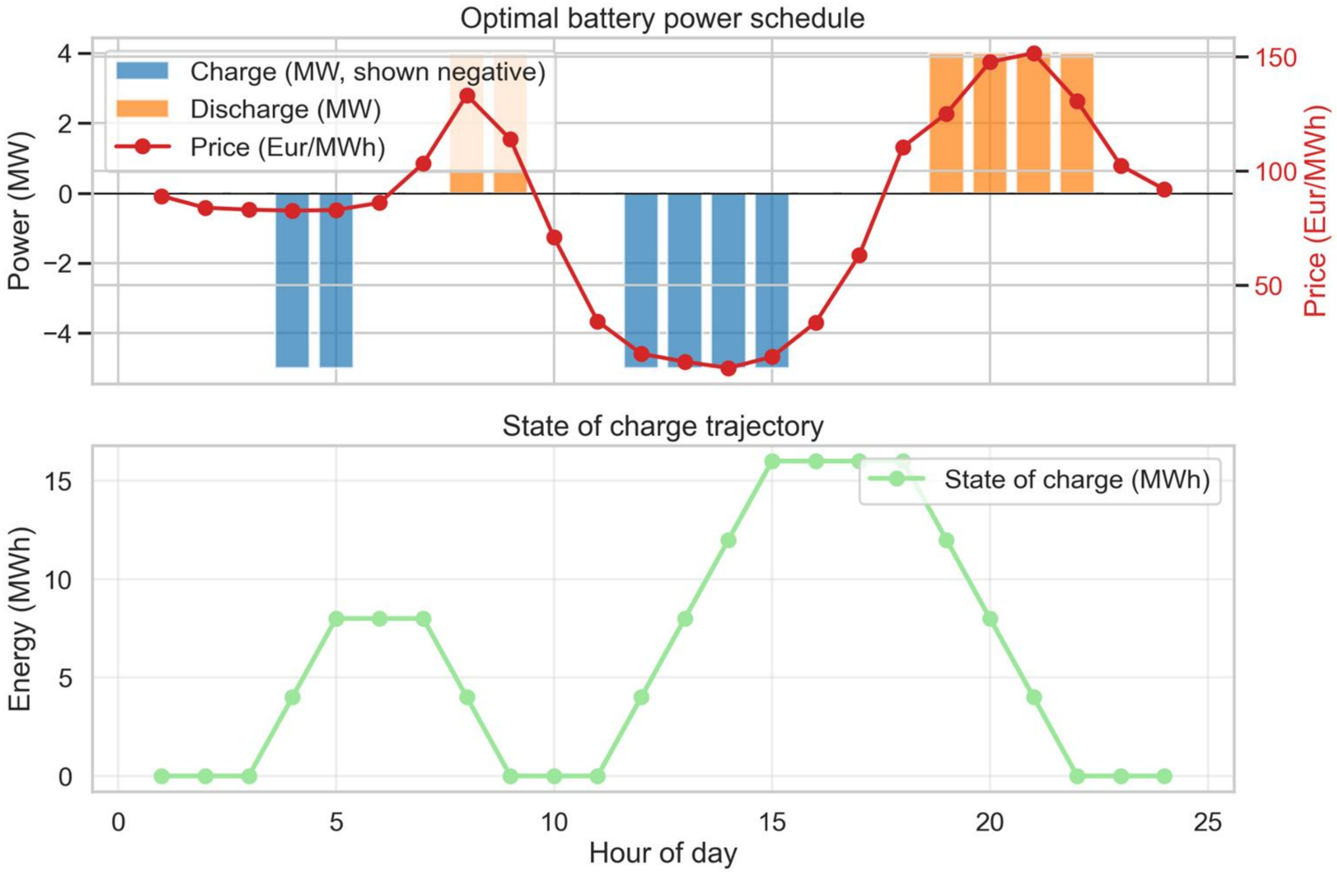
$$E_0 = 0, \qquad E_{24} = 0$$

Discrete variable constraint:

$$y_t^{\text{ch}} + y_t^{\text{dis}} + y_t^{\text{id}} = 1, \quad \forall t$$

$$0 \le c_t \le P_{\text{ch}}^{\max} y_t^{\text{ch}}, \quad y_t^{\text{ch}} \in \{0,1\}$$

$$0 \le d_t \le P_{\text{dis}}^{\max} y_t^{\text{dis}}, \quad y_t^{\text{dis}} \in \{0,1\}$$

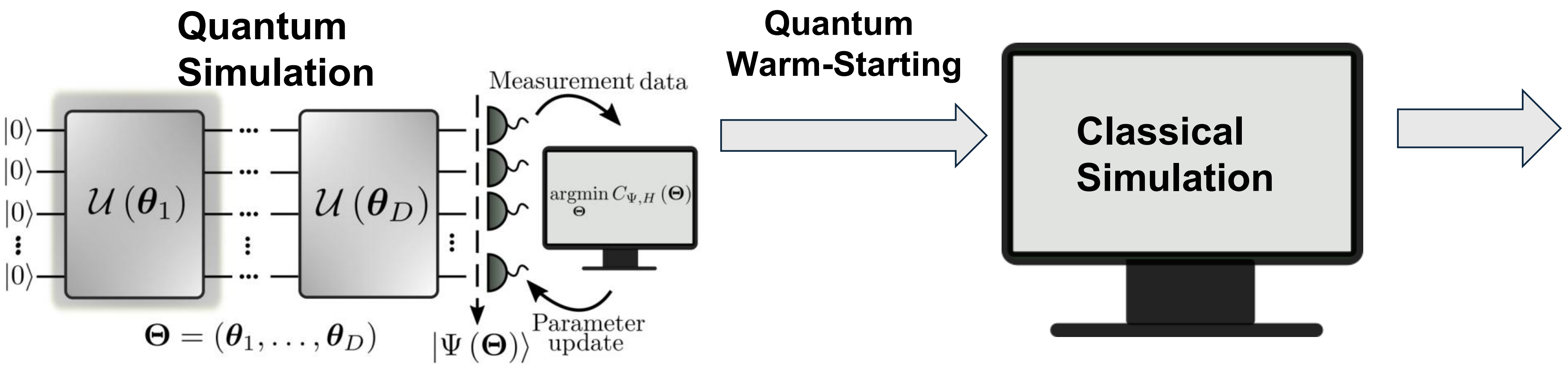


Pozo, David. 'Linear battery models for power systems analysis'. Electric Power Systems Research, 212 (2022)

Hybrid quantum+classical MILP algorithm

Motivation: For large systems (many time steps, many batteries, complex constraints, etc.), the total solution space for a MILP is massive and usual approaches are <u>inefficient</u>.

Solution: Quantum Warm-Starting. To provide the classical solver with an initial, high-quality guess for the Discrete Variables using a quantum algorithm



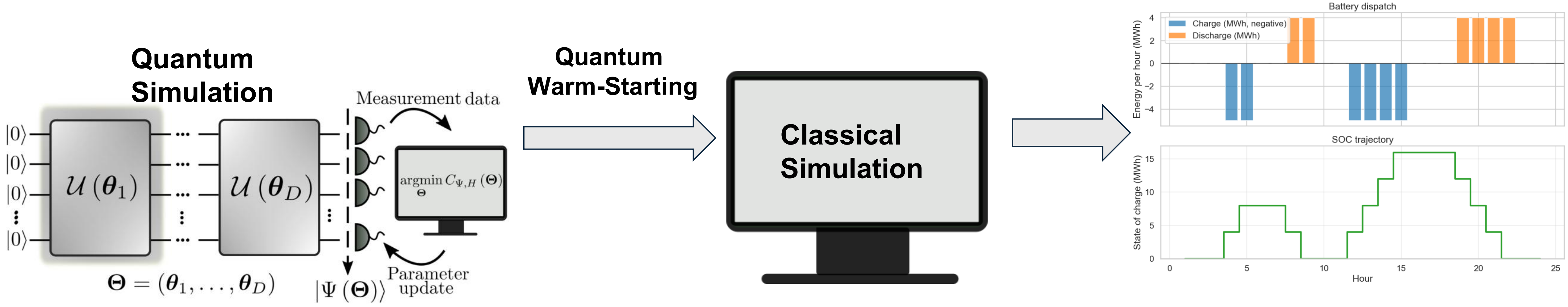
Efficient solution

Conclusion: By using the quantum-obtained guess, we transform a problem of <u>inefficient exploration</u> to one of <u>focused, rapid exploitation</u>, unlocking faster and higher-quality solution for large-scale energy systems.

Hybrid quantum+classical MILP algorithm

Motivation: For large systems (many time steps, many batteries, complex constraints, etc.), the total solution space for a MILP is massive and usual approaches are <u>inefficient</u>.

Solution: Quantum Warm-Starting. To provide the classical solver with an initial, high-quality guess for the Discrete Variables using a quantum algorithm



Conclusion: By using the quantum-obtained guess, we transform a problem of <u>inefficient exploration</u> to one of <u>focused, rapid exploitation</u>, unlocking faster and higher-quality solution for large-scale energy systems.

Hybrid quantum+classical MILP algorithm: QAOA

Our proposal: QAOA is a algorithm to find near-optimal solutions for discrete optimization tasks. It uses quantum effects to explore complex spaces.

We seek to optimize two angles: γ, β

$$|\psi(\boldsymbol{\gamma},\boldsymbol{\beta})\rangle = U_B(\beta_p)U_C(\gamma_p)\ldots U_B(\beta_1)U_C(\gamma_1)|\psi_0\rangle$$

We rewrite a simplified version of the problem in QUBO form including the constraints as const function penalties. This can be solved with standard quantum optimization routines to recover a quantum guess of the optimal schedule.

Step 1 — Skeleton decision variables (24 qubits)

Define one binary variable per hour: $x_t \in \{0,1\}$ for $t = 1, \ldots, 24$.

Interpretation:

- $x_t = 1$ means "discharge during hour t"
- $x_t = 0$ means "do not discharge during hour t"

To keep the QUBO minimal, we use a coarse discharge magnitude: $\tilde{d}_t := 4x_t$ (MWh).

This yields a binary timing skeleton for discharge, which is the warm-start signal we want.

Step 2 — Revenue term in QUBO form

Battery discharge revenue under the skeleton approximation is: $\sum_{t=1}^{24} p_t\tilde{d}_t = \sum_{t=1}^{24} 4p_tx_t$.

A QUBO is typically a minimization, so we minimize negative revenue: $E_{\text{rev}}(x) := -\sum_{t=1}^{24} 4p_tx_t$.

The final QUBO is: $\min_{x\in\{0,1\}^{24}} E(x)$

with: $E(x) = E_{\text{rev}}(x) + E_{\text{sw}}(x) + E_{\text{card}}(x)$

i.e.: $E(x) = -\sum_{t=1}^{24} 4p_tx_t + \gamma\sum_{t=2}^{24}(x_t - x_{t-1})^2 + A\left(\sum_{t=1}^{24} x_t - K\right)^2.$

This uses exactly 24 binary variables.

Hybrid quantum+classical MILP algorithm: QAOA

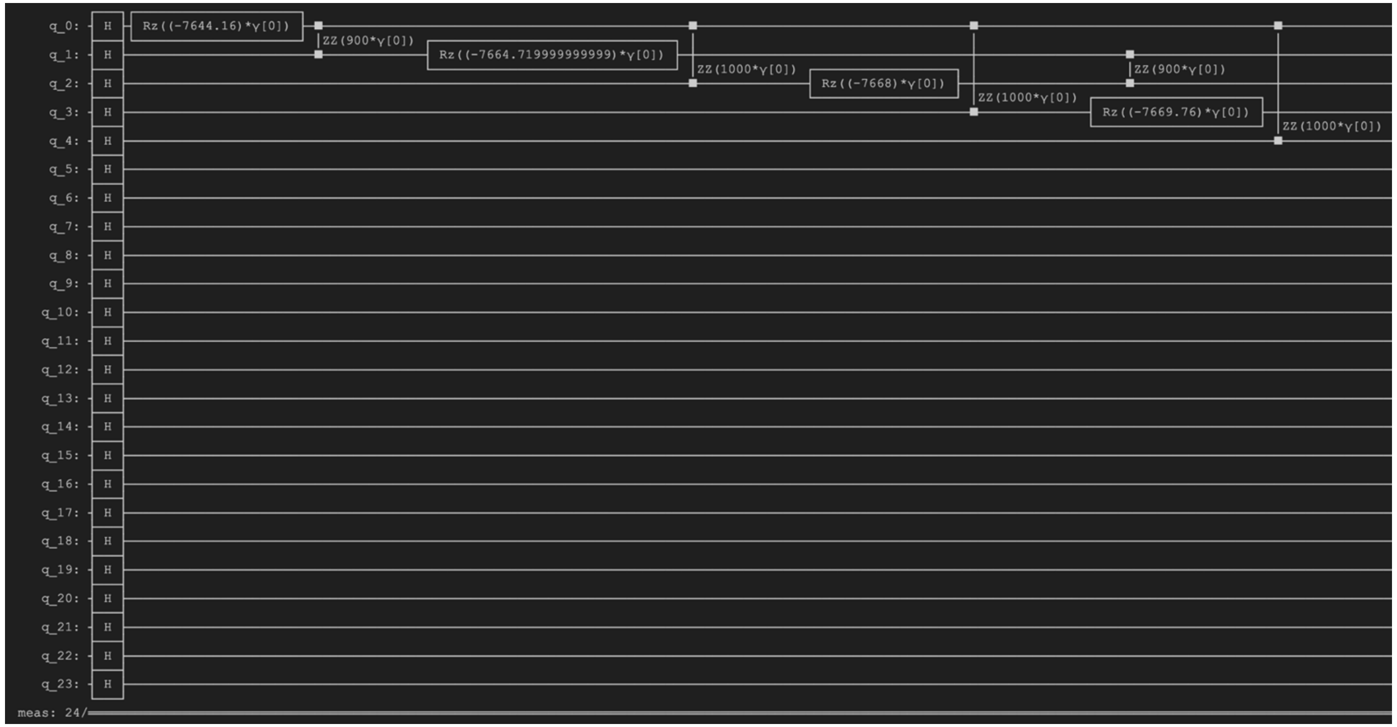
Our proposal: QAOA is a algorithm to find near-optimal solutions for discrete optimization tasks. It uses quantum effects to explore complex spaces.

We seek to optimize two angles: $\boldsymbol{\gamma}, \boldsymbol{\beta}$

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U_B(\beta_p)U_C(\gamma_p)\ldots U_B(\beta_1)U_C(\gamma_1)|\psi_0\rangle$$

We rewrite a simplified version of the problem in QUBO form including the constraints as const function penalties. This can be solved with standard quantum optimization routines to recover a quantum guess of the optimal schedule.

In practice:



Hybrid quantum+classical MILP algorithm: COIN-OR CBC

Our proposal: QAOA is a algorithm to find near-optimal solutions for discrete optimization tasks. It uses quantum effects to explore complex spaces.

We seek to optimize two angles: $\boldsymbol{\gamma}, \boldsymbol{\beta}$

$$|\psi(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = U_B(\beta_p)U_C(\gamma_p)\ldots U_B(\beta_1)U_C(\gamma_1)|\psi_0\rangle$$

After this, we solve the full MILP (constraints+objective) problem using standard classical solvers:

- PuLP package in Python.
- Problem is dispatched to a COIN-OR CBC algorithm, which solves it using a branch-and-cut approach (LP relaxations + branching + cuts) and returns the optimal schedule.

In practice:

```
solver = ClassicalMILPSolver(charge_pattern=charge_pattern, lambda_switch=00.0)

schedule, status, battery_profit, total_revenue = solver.solve(df, wind_rev_exp)
```

✓ 0.1s

```
print(f"Solver status: {status}")
print(f"Battery profit: EUR {battery_profit:,.2f}")
print(f"Total expected revenue (wind + battery): EUR {total_revenue:,.2f}")
print(schedule.head())
```

✓ 0.0s

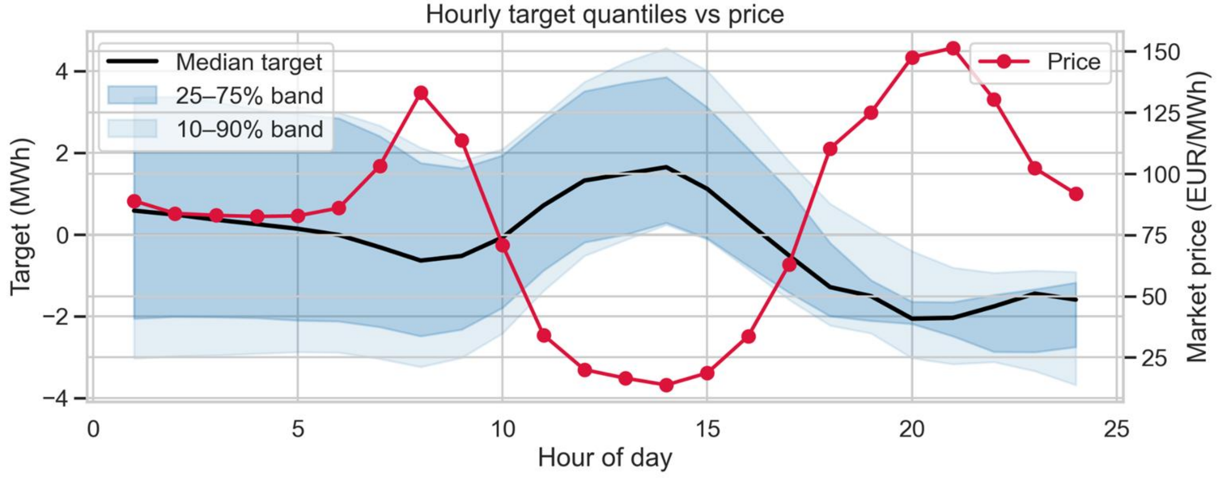
```
Solver status: Optimal
Battery profit: EUR 2,034.54
Total expected revenue (wind + battery): EUR 19,680.02
```

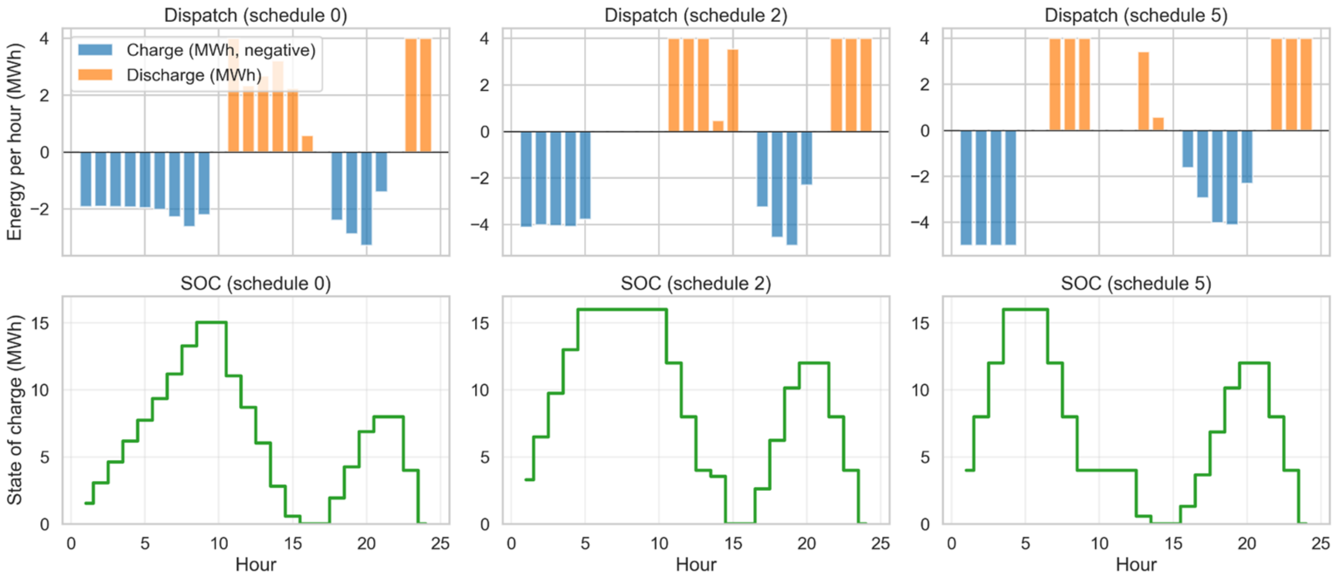
A more realistic formulation of the problem

More realistic models: to take into account effects from the wind farm such as the variance of all 13 predictions, which did NOT affect the previous state of the battery

Set-point tracking problem (**SPT**)

$$\min \sum_{t \in T} \left(\sum_{n \in N} (p^{\mathrm{d}}_{nt} - p^{\mathrm{c}}_{nt}) - p^{\mathrm{sig}}_t \right)^2$$





Pozo, David. 'Linear battery models for power systems analysis'. Electric Power Systems Research, 212 (2022)

A more realistic formulation of the problem

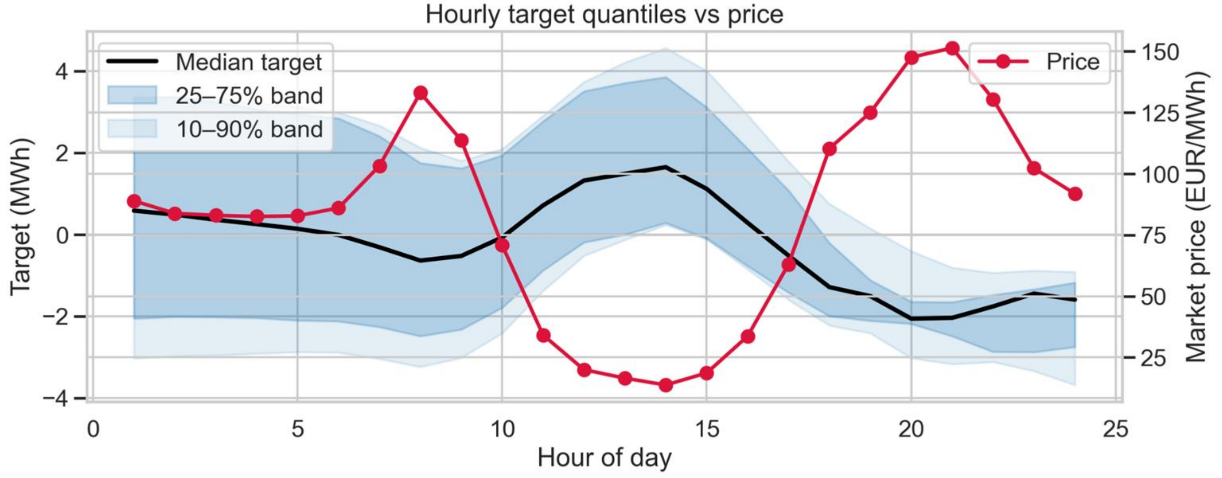
More realistic models: to take into account effects from the wind farm such as the variance of all 13 predictions, which did NOT affect the previous state of the battery

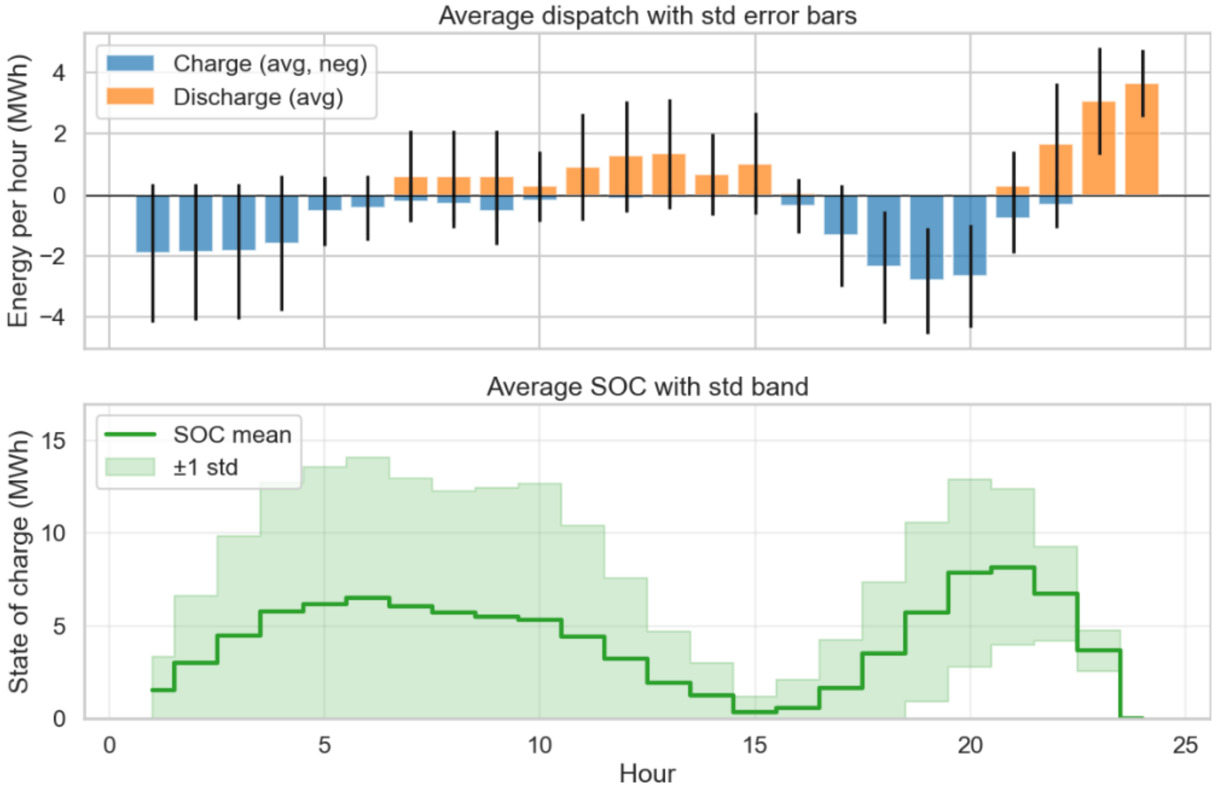
Set-point tracking problem (**SPT)**

$$\sum_{t=1}^{24} |f_t| \quad ; \quad f_t = d_t - c_t - p_t^{\mathrm{sig}}$$

$$p_t^{\mathrm{sig}} = D_t - W_t$$

$$D_t = A p_t^{-\epsilon}$$





Pozo, David. 'Linear battery models for power systems analysis'. Electric Power Systems Research, 212 (2022)

Our final deliverable

We have created a **library implementing and applying our algorithm** to the wind **farm+battery joint optimization**. The library can be used to reproduce our results following the **self-contained tutorials**.

README   Apache-2.0 license

A hybrid quantum+classical MILP solver

This repository holds the code developed by the **queen_of_fog** team for the [IQM Hackathon 2025 Endesa Challenge](). It tackles a day-ahead battery arbitrage and grid-tracking problem that combines classical optimisation (MILP) with a quantum-inspired warm start (QUBO formulation+ QAOA solution). The project works with a deterministic 24-hour price profile and 13 equiprobable wind scenarios stored in `data/input_data.csv`, and packages the workflow into reusable Python modules plus tutorial notebooks.

Tutorials

The `tutorials/` directory walks through the full workflow:

1. 1_Data_analysis.ipynb - load `data/input_data.csv`, explore price and 13 wind scenarios with `PlotDataUtils`, compute wind and revenue statistics, and visualise correlations between wind, revenue, and price.
2. 2_QAOA_Hardware.ipynb - build the battery-dispatch QUBO, run QAOA (both simulators and real IQM hardware) with `QAOAGuessSolver`, and export the most probable bitstring/discharge pattern.
3. 3_Hybrid_quantum+classical_MILP.ipynb - decode the QAOA bitstring into a warm start for `ClassicalMILPSolver`, solve the deterministic arbitrage MILP with the PuLP package, and plot the resulting dispatch and state-of-charge trajectory.
4. 4_Introducing_weather_forecasting.ipynb - extend the MILP to multiple wind scenarios using `ClassicalMILPSolverWithWeatherData`, build a price-elastic demand target, evaluate tracking penalties, and compare or aggregate schedules across scenarios.

Code description

Core modules live in `src/` and mirror the notebook workflow:

```
.
|-- data/
|   |-- input_data.csv              # Hourly price + 13 wind scenarios
|   |-- qubo_matrix_symmetric.csv   # Pre-built QUBO matrix (symmetric form)
|   |-- qubo_matrix_upper.csv       # Upper-triangular QUBO matrix
|   `-- qubo_solution.csv           # Stored QAOA bitstring/discharge guess
|-- src/
|   |-- __init__.py
|   |-- plot_data_utils.py          # Load data, compute wind/revenue stats, and plot curves/fan chart
|   |-- classical_MILP_solver.py    # Deterministic battery arbitrage MILP with warm starts and plotti
|   |-- classical_MILP_solver_with_weather_data.py  # Scenario-based arbitrage/tracking MILP and visual
|   `-- qaoa_guess_solver.py        # Build QUBO matrix and solve via QAOA to seed the MILP
|-- tutorials/                      # Main notebooks (see above)
`-- test/                           # Scratch notebooks and solver outputs (MPS/solution files)
```

Conclusions and outlook

- We introduce a hybrid quantum+classical mixed linear integer programming algorithm that leverages quantum computers to generate initial guesses for the solution.
- The quantum part is implemented solving a QUBO problem with the QAOA. The classical part is a standard MILP routine.
- We apply the algorithm to jointly optimize a battery and a wind farm, demonstrating its advantage when the systems are both coupled and uncoupled.
- QAOA is expected to give superpolynomial advantage in optimization problems [S. Boulebnane and A. Montanaro, PRX Quantum **5**, 030348 (2024)]. This makes the approach promising when scaling to bigger systems.
- Quantum error mitigation (QEM) and correction (QEC) techniques could be included to mitigate current hardware noise rates.