Specialisation Project (VT2)
HS2024

Enhanced Platform for Investment Analysis

Mixed-Integer Linear Programming Optimization Model for Integrated Energy Systems in Python

Submitted by
Rui Vieira
Institute of Product Development and Production Technologies (IPP)

Supervisor
Dr. Andrea Giovanni Beccuti
IEFE Model Based Process Optimisation

Study Program
Business Engineering, MSc in Engineering

July 13, 2025

Imprint

Project: Specialisation Project (VT2)
Title: Enhanced Platform for Investment Analysis
Author: Rui Vieira
Date: July 13, 2025
Keywords: mixed-integer linear programming, MILP, quantitative modeling, python, strategic planning, optimization, asset valuation, power-flow, platform, forecasting, energy trading

Study program:
Business Engineering, MSc in Engineering
ZHAW School of Engineering

Supervisor:
Dr. Andrea Giovanni Beccuti
IEFE Model Based Process Optimisation
Email: giovanni.beccuti@zhaw.ch

Abstract

This work presents an integrated investment framework for energy systems, focusing on optimal technology selection and placement of electrical generation, conversion, and storage assets. The core engine combines DC Optimal Power Flow (DC-OPF) simulations with linear programming (using the PuLP solver) to evaluate both technical feasibility and economic viability across a multi-scenario analysis. A 9-bus test network provides the backdrop for a reduced ten distinct cases, each featuring a unique mix of conventional (nuclear, gas) and renewable (solar, wind) power plants, supplemented by battery storage of varying capacities.

To balance computational efficiency with seasonal realism, the annual horizon is divided into three representative weeks (summer, winter, and spring/autumn), whose costs and operations are subsequently scaled to form a full-year analysis. This approach reveals significant seasonal differences in storage utilization even enabling clean assets to compensate for the cost of conventional generation.

In scenarios featuring abundant solar generation, the SoC frequently reaches its upper limits, highlighting the potential for upsizing or more flexible operational strategies—such as battery leasing or modular additions—to capture peak renewable output.

An economic sensitivity analysis underscores the strong influence of high-cost resources during extreme load conditions, causing a disproportionate rise in total costs when reliance on expensive generation escalates. Meanwhile, scenarios with nuclear-dominated baseload exhibit lower operational cost volatility but may still benefit from targeted storage deployment to manage residual demand swings. AI-assisted reporting consolidates these findings by identifying cost drivers, optimal technology mixes, and operational bottlenecks across all scenarios. Notably, Scenario7's balanced blend of nuclear, solar, and wind with moderate battery support emerges as the most cost-effective configuration, while Scenario4, featuring gas-fired generation and multiple storage units, proves the least favorable in terms of net present value (NPV).

Overall, the proposed framework bridges technical dispatch simulation and investment analysis, guiding stakeholders in designing resilient, economically viable energy systems. Future enhancements include broader maintenance modeling, real-time price integration for advanced arbitrage strategies, and further prompt-engineering improvements to refine AI-driven reporting and decision support.

Keywords: mixed-integer linear programming, MILP, quantitative modeling, python, strategic planning, optimization, asset valuation, power-flow, platform, forecasting, energy trading

Contents

1 Introduction

1.1 Context & Motivation

1.2 VT1 Recaps

1.2.1 Bottlenecks

1.3 Goals

2 Literature and Toolchain Review

2.1 Mixed-Integer Programming in Power-System Planning

Mixed-Integer Programming (MILP) is an improved version of Linear Programming (LP) that allows to solve problems with discrete variables for example whether a unit is on or off (unit commitment). Such binary decisions can not be modelled with pure linear programming. This allows to integrate investment (capital expenditure) and operational (dispatch, or operating cost) decisions into one optimization framework. One can co-optimize the true least-cost solution that considers both capex and opex together. [**?**, **?**].

Formulating generation expansion planning (GEP) as an MILP captures the binary nature of building decisions (build vs. not build) and can include operational details like unit commitment. Despite being possible to include operation details such as minimum on/off time, rampe rates, etc. in the model, it was not done in this project. Not only is it more realistic but it also avoids suboptimal decisions that could arise from treating planning and operation separately.

2.2 Short-term PV/Wind Forecasting Methods

Short-term forecasting of photovoltaic (PV) or wind power is vital for grid operations. Approaches include:

- **Persistence/Empirical**: Simple baselines, e.g., assuming tomorrow equals today.
- **Physical/NWP**: Use weather forecasts and physical models for power prediction.
- **Statistical**: ARIMA/SARIMA and ARIMAX models learn from historical data and exogenous variables [**?**]. They are interpretable and data-efficient but limited for nonlinearities.
- **Machine Learning**: Methods like neural networks, SVR, and especially gradient boosting (e.g., XGBoost) capture complex patterns and often outperform statistical models when sufficient data is available [**?**, **?**, **?**]. Gradient boosting is noted for its accuracy and speed in PV/wind forecasting.
- **Hybrid/Ensemble**: Combine models (e.g., ARIMA+ANN) for improved robustness, though added complexity may not always yield better results.

Given these findings, we used gradient boosting (XGBoost) for forecasting, with SARIMA as a baseline.

2.3 Solver Landscape and Selection

Solving large MILPs requires robust solvers. Commercial options (CPLEX, Gurobi, Xpress) are state-of-the-art, offering fast solve times and reliability [**?**, **?**]. Open-source solvers (CBC, GLPK, SCIP, HiGHS) are free but generally slower and less robust. Benchmarks show Gurobi and CPLEX are typically 12–100x faster than CBC, and commercial solvers solve more instances to optimality [**?**].

For this project, CPLEX was chosen for the sake of understanding the solver and for having a academic license. Additionally, CPLEX offers a Python API (docplex), which made integration with our Python-based workflow straightforward

2.4 Python Ecosystem for Optimisation and ML

This interoperability and abundance of libraries is a major reason Python is so dominant in these fields today. Our literature review also confirmed that many recent research works in energy systems adopt Python for similar tasks, citing its balance of user-friendliness and powerful capabilities

Below is a list of the most relevant libraries for optimization and machine learning:

- For optimization, libraries like PuLP and Pyomo allow flexible model formulation and solver switching [**?**]. We used the CPLEX Python API (docplex) for direct integration.
- For ML, scikit-learn and XGBoost provide powerful tools for data processing and forecasting. Others librairies such as PyTorch, TensorFlow, and Keras are the reference ones for deep learning and neural networks [**?**].
- Statsmodels was used for time-series (SARIMA) modeling as a baseline model.

3 Problem Definition and Scope

This project extends the first-year LP framework in two directions:

- **MILP investment-dispatch model** – replaces the single-year LP with a multi-year mixed-integer formulation that chooses both what to build and how to run it, so CAPEX and OPEX are minimised in one pass.
- **Independent ML forecasting module** – delivers day-ahead predictions of the exogenous time series (load and variable renewables). It is stand-alone for now but provides the data that a rolling short-term optimisation would need.

This section outlines what is inside the study, what stays outside, and which indicators we will track.

3.1 Planning Horizon and System Boundaries

The scope of the study is defined by several key elements:

Strategic horizon: The model considers a user-defined list of years, normally between 1 and 10. This allows the optimisation to capture the timing of asset builds and retirements.

Operational resolution: Each year is represented by three typical weeks (winter, summer, spring-autumn). This approach keeps the MILP size modest while preserving essential seasonal detail.

Forecast horizon: The main forecasting target is 24 hours ahead, with additional checks at 48, 72, and 168 hours. This aligns with day-ahead operational needs and allows us to observe how forecast quality degrades with longer horizons.

Network footprint: The optimisation is run on a fixed test grid, including buses, lines, and assets. This lets us attribute changes in results to the model itself, not to changes in the underlying data.

Everything behind the connection point, as well as retail tariffs and ancillary services, is excluded from the study.

3.2 Decision Variables and Constraints

Variable	Type	Status
$b_{g,y},\ b_{s,y}$	binary	new – build decisions for generators g and storage s
$u_{g,y,t}$	binary	new – on / off for thermal generators
$p_{g,y,s,t}$	continuous	carried over – dispatch per generator
$p^{\text{ch}}_{s,y,s,t},\ p^{\text{dis}}_{s,y,s,t}$	continuous	carried over – storage charge, discharge
$e_{s,y,s,t}$	continuous	carried over – state of charge
$f_{l,y,s,t}$	continuous	carried over – DC line flow

New or changed constraint families include:

- **Build–lifetime link:** Sums of $b_{g,y}$ or $b_{s,y}$ set the installed flag for each year and forbid overlapping rebuilds.
- **Unit commitment:** Minimum up/down times, start costs, and ramp limits are tied to $u_{g,y,t}$.
- **CAPEX annuity:** Annual cost terms are based on build binaries and a capital recovery factor.
- **Storage relaxed final SoC:** The end-of-week state of charge may float within 10% of capacity to speed up the solve.

All other LP constraints—such as power balance, line limits, storage energy balance, and renewable profiles—remain in place, but now reference the new binary variables where needed. These elements turn the LP into a MILP, giving the solver the freedom to co-optimise when to invest and how to operate.

3.3 Forecast Horizon and Accuracy Targets

The forecasting module generates:

- **Primary target** – 24 hourly values for the next day for each time series.
- **Additional checks** – 48 h, 72 h and 7-day horizons to stress-test model degradation.
- **Features** – calendar flags, recent lags, rolling means, simple weather proxies.
- **Models** – linear baseline, random forest, gradient boosting, LSTM; all treated with identical split and scaling rules.

Accuracy targets (set after an initial back-test):

Metric	**Day-ahead target**
MAE	$< 5\%$ of mean load
RMSE	$< 7\%$ of mean load

Longer horizons have looser bands, logged but not optimised against.

3.4 Key Performance Indicators

The following indicators are tracked to assess the value of the MILP and the forecasting module:

- **Cost:** Total discounted system cost, including CAPEX annuities and weighted OPEX, is the main optimisation objective.
- **Solver:** MILP wall-clock time and optimality gap are reported for each run.
- **Forecast:** MAE and RMSE are tracked for each horizon and time series; lower values are better.
- **Robustness:** The number of hours with unmet demand or line overload is monitored and should be zero.
- **Transparency:** The share of total cost by asset class is reported to support sensitivity analysis.

4 Methodology

as mentioned before, the code base has been upgraded to a multi-year MILP and a new forecasting module. The codebase relies on that strucutre hence the methodology used.

The change on the code base happen mostly between these three blocks:

```
investment-model/
  +-- data/                       # data layer
  |   \-- ...
  |
  +-- forecast/                   # day-ahead forecasting module
  |   \-- ...
  |
  \-- scripts/                    # MILP investment model
  |   \-- ...
  ...
```

Figure 1: Code architecture - showing the main components: data layer for grid and time-series inputs, forecasting module, and optimization scripts.

4.1 Data layer

The data layer looks similar to the previous by splitting static part (network topology and the assets specifications) and asset profiles. (defining profiles for the demand and assets capabilities) However, switching to a multi-year MILP removed the need for per-scenario network files ! hence creation of a new configuration file `analysis.json`, which defines the parameters of the investment problem (optimization) such as the planning horizon, annual load growths, and representative weeks selection.

In the new version the static grid data files have been simplified, still replicating the metadata of the previous version using the matlab method (SEARCH FOR REF) but only the necessary columns were kept -¿ made use to clean Representative-week slicing (three 168h blocks) is carried over from the LP version and therefore not described here again.

```
data/
  +-- grid/
  |   +-- analysis.json           # configuration file : horizon, growth, ...
  |   +-- buses.csv               # network topology
  |   +-- generators.csv          # assets specifications, lifetime and CAPEX
  |   +-- lines.csv               # network topology
  |   +-- loads.csv               # demand profile
  |   \-- storages.csv            # assets specifications, lifetime and CAPEX
  +-- processed/
      +-- load-2023.csv           # load time-series
      +-- solar-2023.csv          # solar time-series
      \-- wind-2023.csv           # wind time-series
```

Figure 2: Data layer - showing the main components: static grid data and generation and load profiles.

4.2 Linear to Mixed-Integer Programming Transition

Below is a concise-but-thorough "diff" of the two generations of the code base, written from an optimisation & applied-mathematics standpoint.

1. Big-picture methodology shift

Aspect	vt1 ("OLD")
Problem class	Pure LP (continuous), single representative week per season, one year at a time.
Decision space	Dispatch – generation, line flows, storage charge/ discharge. Asset mix is exogenous (read fron
Temporal layers	One 168-h week × 3 seasons cost scaled by {13, 13, 26}.
Cost handling	Variable cost only (generator gencost · p) + tiny cycling penalty for storage. Investment anal
Solver / package	PuLP + CBC.
Scenario management	"Run many LPs with different inputs; compare afterwards."

2. New variables and why they matter Let

- G = set of candidate generators,
- S = set of candidate storage units,
- $Y = \{1 \dots Y\}$,
- Σ = seasons.

Variable	Type	Size	Interpretation
gen_build[g,y]	binary	$\lvert G\rvert \cdot \lvert Y\rvert$	$1 \Leftrightarrow$ unit g is first commissioned (or replaced) in year y.
gen_installed[g,y]	binary	$\lvert G\rvert \cdot \lvert Y\rvert$	$1 \Leftrightarrow$ unit g is available in year y. Determined by sliding-sum of
p_gen[g,y,s,t]	≥ 0	...	Dispatch (MW). Capped by p_{nom}· installed and by weather pro
p_line[ℓ,y,s,t]	free	...	DC line flow (MW).
p_charge, p_discharge, soc[s,y,s, t]	≥ 0	...	Storage power & state of charge.

Storage variables existed before; what is new is the ±year index and the installation gating by storage_installed[s,y].

3. Key new constraints (mathematical form) Below Δ denotes what is new vs. vt1.

3.1 Build / installed linking Δ

For every asset a with lifetime L_a (years) and for every target year y:

$$\text{installed}[a, y] = \sum_{y' \leq y,\, y - y' < L_a} \text{build}[a, y']$$

$$\sum_{y' \leq y,\, y - y' < L_a} \text{build}[a, y'] \leq 1 \quad \text{(no double-build within life window)}$$

Gives a neat "at-most-one build inside the sliding lifetime window" logic in two linear rows.

3.2 Capacity coupling Δ

$$\begin{aligned} p\_gen[g, y, s, t] &\leq p\_nom_g \cdot \text{profile}[g, s, t] \cdot \text{installed}[g, y] \\ p\_charge[s, y, s, t] &\leq p\_nom_s \cdot \text{installed}[s, y] \\ p\_discharge[s, y, s, t] &\leq p\_nom_s \cdot \text{installed}[s, y] \\ soc[s, y, s, t] &\leq p\_nom_s \cdot \text{max\_hours} \cdot \text{installed}[s, y] \end{aligned}$$

3.3 Storage dynamics (unchanged except for year index)

$$\begin{aligned} soc_{t+1} &= soc_t + \eta_{in}\, p\_charge_t - (1/\eta_{out})\, p\_discharge_t \\ soc_0 &= 0 \\ soc_T &\leq 0.1 \cdot E_{nom} \cdot \text{installed} \end{aligned}$$

Seasonal SoC is forced to start (and loosely end) at 0, removing cross-season coupling (vt1 required cyclical SoC equality).

3.4 Nodal power balance (extended)

For each bus b, season s, year y, time t:

$$\sum_{g \in G_b} p\_gen[g,y,s,t] + \sum_{s \in S_b} (p\_discharge - p\_charge)[s,y,s,t] + \sum_{\ell \in In_b} p\_line[\ell,y,s,t] = \text{growth}[y] \cdot \text{Load}[b,s,t] + \sum_{\ell \in Out_b} p\_line[\ell,$$

(vt1 had the same balance but without the growth multiplier and without year index).

3.5 Objective (linearised annuity cost) Δ

$$\min \underbrace{\sum_{s \in \Sigma} W_s \sum_{y \in Y} \sum_{g,t} c_g \, p_{g,y,s,t}}_{\text{operational}} + \underbrace{\sum_{y \in Y} \Big(\sum_{g} \text{CRF}_g \cdot \text{CapEx}_g \cdot \text{installed}_{g,y} + \sum_{s} \text{CRF}_s \cdot \text{CapEx}_s \cdot \text{installed}_{s,y} \Big)}_{\text{annualised capital}}$$

No slack / load-shedding term – unmet demand is infeasible.

4. What disappeared

Old element	Reason for removal / replacement
Scenario CSV controlling asset mix	Investment is now endogenous via binary variables.
Separate NPV/annuity spreadsheet	Annuity built straight into objective through CRF × installed.
Final-SoC = initial constraint per season	Replaced by → $\text{soc}_0 = 0$, $\text{soc}_T \leq 10\%$ cap. Eliminates hard coupling that n
Slack variables (load_shedding)	Model now enforces demand exactly (power-balance equality).
Storage "must finish with same SoC"	See above.
Discounting of OpEx	Dropped (weights cover season length only); simplifies objective coefficients

5. Impact on solvability & scale

- **Problem size:** Binaries: $|G| + |S|$ per year – small relative to dispatch variables.
- **Continuous vars:** identical order of magnitude as before $\times |Y|$.
- **Complexity jump:** MILP is NP-hard vs. LP polynomial. However, lifetime & build constraints are totally unimodular "staircase" structures → usually easy cuts for CPLEX.
- **Numerical tightness:** Removing slack makes the model brittle if profiles & capacities cannot cover peak load. The "$\leq 0.1 E_{nom}$" final-SoC relaxation avoids cyclic infeasibility due to net-export / net-import imbalance inside a season.
- **Interpretability:** With installed extracted per asset/year, the post-processor can draw Gantt-style timelines and compute replacement patterns exactly (see plot_implementation_timeline).

6. Summary for practitioners The new framework internalises what used to be a Monte-Carlo-over-scenarios exercise into a single MILP. Binary "build" decisions, lifetime-aware replacement, annuitised capex and load-growth scaling are now first-class constraints. Everything else – DC power flow, storage physics, seasonal weighting – is inherited from the old LP.

For a planning team this means:

- One optimisation run = one least-cost expansion plan, instead of manual enumeration of candidate mixes.
- Costs are directly comparable across assets thanks to CRF-based linearisation.

- No external NPV spreadsheets; sensitivities (discount rate, lifetime) change only numerical coefficients, not model structure.
- Infeasibility becomes a diagnostic – you see immediately when the candidate fleet cannot meet future demand without new builds.

From a mathematical optimisation viewpoint the step from LP $\Rightarrow$ MILP is moderate in size yet profound in capability: the added 0/1 variables permit lumpy investment choices while preserving a linear objective and constraints, so commercial solvers (CPLEX, Gurobi) remain efficient on realistic medium-scale instances.

4.3 Forecasting module

The forecasting module is new and is based on a statistical and machine learning models that predicts the profile of a given generation asset.

5 Implementation

5.1 Code Walk-through

5.2 Data Structures & File Formats

5.3 Performance Profiling and Optimisation

5.4 Testing & Validation Strategy

6 Results

6.1 Forecasting Accuracy

6.2 MILP vs Legacy LP

6.3 Solver Impact (CPLEX vs GLPK)

6.4 Sensitivity and Scenario Analysis

6.5 Integrated Workflow Demo

7 Discussion

7.1 Interpretation of Key Findings

7.2 Trade-offs Analysis

7.3 Limitations

8 Conclusions and Outlook

8.1 Achievements Relative to Goals

8.2 Near-term Tasks

8.3 Long-term Vision

Acknowledgements

For the redaction of this report, I would like to acknowledge the use of artificial intelligence to improve the clarity and structure of my sentences. The core observations, analyses, and personal reflections are entirely my own, drawn from my experiences during the field trip and subsequent research. The LLM usage was employed primarily for language refinement, code formatting, and orthographic corrections. Its integration helped communicate complex concepts clearly and effectively.