RACE for 2030

RELIABLE
AFFORDABLE
CLEAN
ENERGY

N3 Research Report

# Behind the meter forecasting and optimisation

## Final Report

Australian Government
Department of Industry, Science and Resources

AusIndustry
Cooperative Research Centres Program

## RACE for Networks

**Research Theme N3**: Local DER network solutions

ISBN: 978-1-922746-49-8

Industry Report

Behind the Meter Forecasting & Optimisation

November 2023

## Citations

Bergmeir, C., Bui, Q., De Nijs, F., Stuckey, P.J., (2023) Behind the Meter Forecasting & Optimisation. Prepared for RACE for 2030 CRC.

## Project team

### Monash University

- Dr Christoph Bergmeir
- Mr Quang Bui
- Dr Frits de Nijs
- Professor Peter Stuckey

### SwitchDin

- Dr Alan Gore
- Ms Alison Washusen
- Mr Yuji Ishikawa

## Projectpartners

## Acknowledgements

The authors would like to thank the stakeholders involved in the development of this report, in particular the industry reference group members who have given so generously of their time, including Gridsight, Synergy, and Ausgrid. Whilst their input is very much appreciated, any views expressed here are the responsibility of the authors alone.

## Acknowledgement of Country

The authors of this report would like to respectfully acknowledge the Traditional Owners of the ancestral lands throughout Australia and their connection to land, sea and community. We recognise their continuing connection to the land, waters and culture and pay our respects to them, their cultures and to their Elders past, present and emerging.

## What is RACE for 2030?

RACE for 2030 CRC is a 10-year co-operative research centre with AUD350 million of resources to fund research towards a reliable, affordable and clean energy future. racefor2030.com.au

## Disclaimer

# Executive Summary

## Project purpose and overview

The electricity grid has seen a significant increase in the number of prosumers, households and small business connections acting as both a producer and a consumer of electricity, who can benefit from mitigating rising energy costs with reduced net consumption.

The wider grid can also benefit from prosumers' load fl exibility. However, left uncoordinated their increasing involvement in the grid introduces inherent instabilities, particularly in relation to grid frequency. To maintain stability, a concerted response is required and Virtual Power Plants (VPPs), being a single software-defined entity from the perspective of the market, action those responses by internally distributing controls to the individual prosumer assets. This arrangement benefits both the wider grid through increased flexibility and the individual prosumers through reduced costs for services provided.

The real-time control system of such a virtual power plant faces a difficult prediction and optimisation problem due to the large number of entities. This project develops and evaluates the potential of solutions that jointly tackle the prediction and optimisation problem, to improve the effectiveness of virtual power plant c ontrol. It consists of three core activities:

**Work package 1: Creating benchmark dataset** Despite the flurry of activity in virtual power plant research, there is a lack of suitable shared datasets with which to work. Such datasets can help lower the barrier to entry into the field and create a common ground to evaluate s olutions. Other fields of computer science such as computer vision have benefited greatly from access to open and established benchmark data, where the MNIST dataset of handwritten digits and the ImageNet object recognition database are widely used. The goal of this work package is to construct such a benchmark dataset for the virtual power plant problem by curating real-world, anonymised data from SwitchDin for use by the wider community. This data, together with other publicly available datasets such as historical weather time series and distribution network layouts, allows others to build on this research project.

**Work package 2: Baseline forecasting and optimisation** In addition to common data, for the evaluation of progress in the field, it is also essential to have access to common baselines against which to compare new technical solutions. Therefore, in this work package, we develop *separate* baseline solutions to the forecasting and optimisation problems and combine them in the usual sequential approach. This will produce a credible state-of-the-art baseline for future innovations to build on.

**Work package 3: Predict plus optimise solution** Subsequently, we develop advanced solutions to the *joint* predict plus optimise problem, that improve upon the baseline solution in terms of scalability of the virtual power plant controller to handle large aggregations, as well as accuracy of control by integrating the optimisation objective into the prediction engine.

## Methodology

Developing any forecasting system for solar generation or consumer load profiles requires input data. Therefore, the first step of our methodology is to collect and analyse the exisiting data with an exploratory data analysis aimed at finding relevant features and repairing any anomalies. Subsequently, we use the prepared data to develop a strong baseline forecast, using state-of-the-art gradient-boosted decision tree regression algorithm LightGBM (Ke et al., 2017). We fit direct, global models, such that there is one model for every forecast

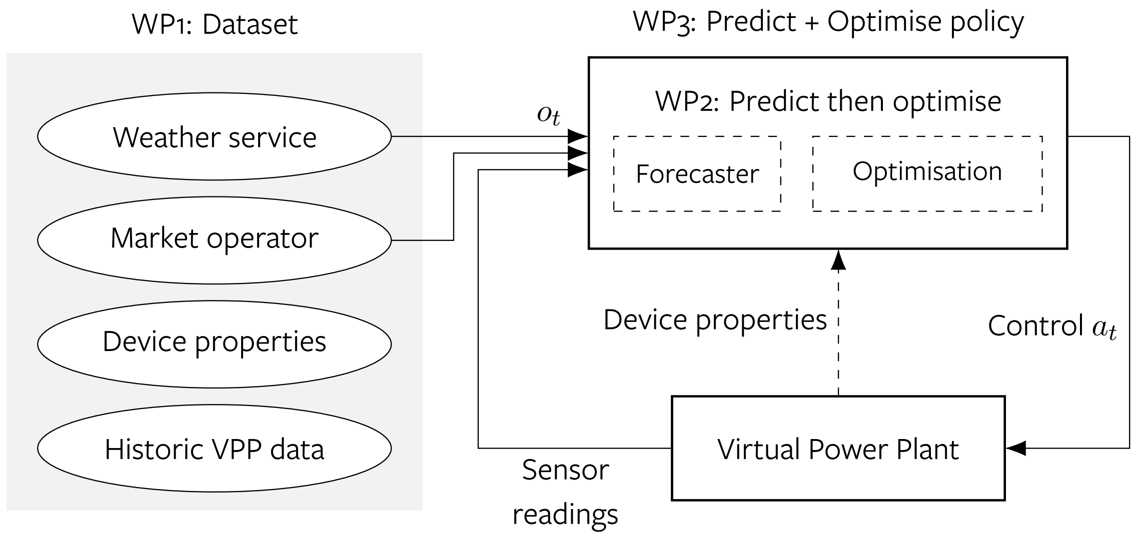


**Figure 1:** Overview of the work packages.

horizon, each able to forecast load or PV values for all consumers individually. As a baseline solution, we then develop a *predict-then-optimise* optimisation model that takes the forecast and optimises the VPP battery schedule for it. The optimisation model is built in the high-level constraint programming language MiniZinc (Nethercote et al., 2007), making it solver agnostic and easy to understand and maintain. Finally, we investigate two *predict-plus-optimise* strategies, an efficient replanning schedule based on column generation and model-predictive control to make use of the improved near-term forecasts, and a structured prediction algorithm based on computing the loss function for a machine learning algorithm directly through the optimisation, resulting in a forecast objective that directly aligns with the optimisation objective.

## Results and discussion

After evaluating several modelling options and forecast designs, we arrived at a strong baseline forecasting system that is capable of making significantly more accurate forecasts than simple baselines and Autoregressive Integrated Moving Average (ARIMA) on the individual load and PV signals. We employ these forecasts in a lin-ear programming optimisation to obtain battery schedules that maximise the VPP's performance in terms of financial outcomes for consumers and peak shaving for the collective. The resulting schedule minimises cost by charging under the cheaper night-time tariff structure, while curtailing evening peak loads (Figure 2).

Nevertheless, we can use a simple forecast-biasing shift to demonstrate that a worse forecast error may in fact lead to better optimisation outcomes. By adding 0.2 kW to all units during off-peak time steps and thereby artificially biasing the forecasts, a forecast is obtained with worse mean-absolute error forecasting score, but better schedule outcomes in terms of overall cost and peak reduction (Table 1). This motivates the investigation of predict-plus-optimise strategies, which aim to solve both the forecast and optimisation objective together, by directly optimising the forecast for the optimisation goal. We develop a prototype of such a custom loss function and demonstrate that it can significantly improve the optimisation objective over a forecast model trained for prediction error. The loss is especially effective when the input features are relatively uninformative, such that the traditional forecast errors like MAE become more difficult to reduce.

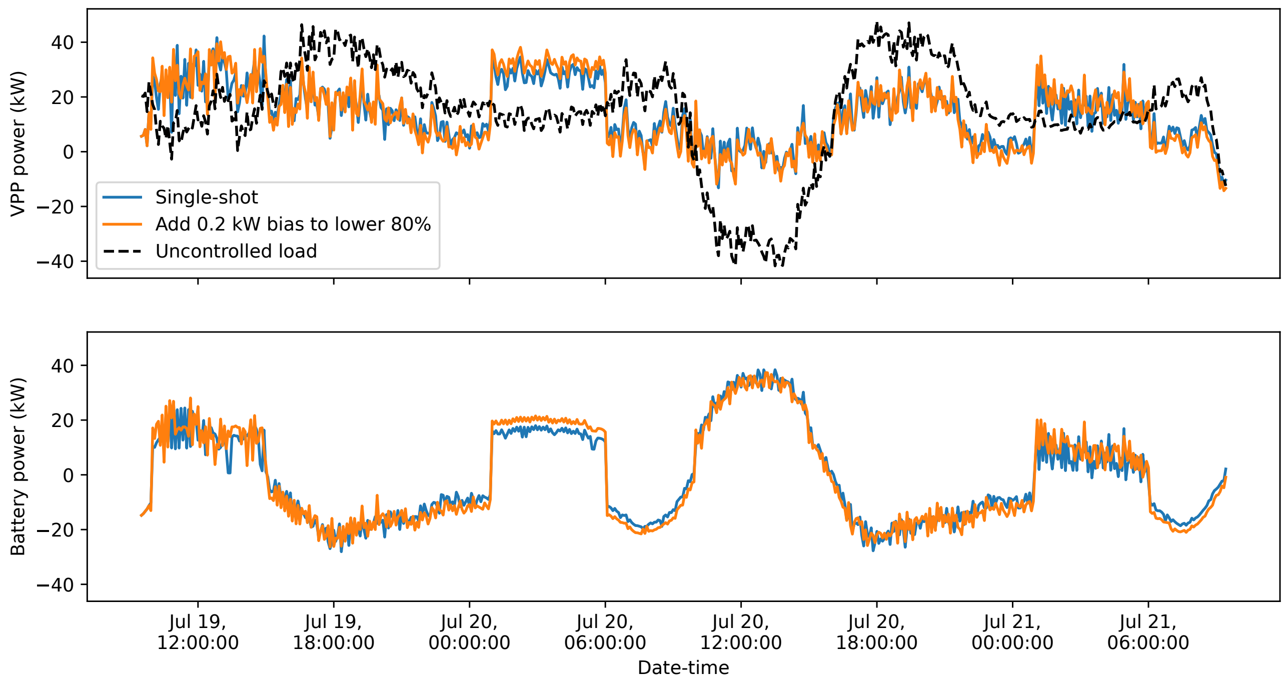


**Figure 2:** Comparison of optimal battery control schedule under a best-fit forecast (blue), versus one where the forecast has been artificially biased to over-predict load (orange). Despite lower mean absolute error, the unbiased forecast has worse gap in the optimisation objectives.

**Table 1:** Performance of the forecasts in terms of forecast error (MAE), compared with the quality of the battery schedule in terms of regret gap against the best-possible hindsight solution.

| | Forecast | |
|---|---|---|
| Error metric | Normal | Biased |
| MAE | **0.874** | 0.934 |
| Tariff gap ($) | 39.013 | **36.258** |
| Peak gap (kW) | 11.659 | **9.559** |

## Findings and recommendations

Our findings suggest that a predict-plus-optimise strategy applied to individual units is in principle feasible for the virtual power plant control problem. Further research is required to assess its practical suitability. This includes constructing a feature set that could be obtained in real time in reality, which requires access to a set of historic weather forecasts. Furthermore, additional evaluation on real edge computing hardware is required to determine scalability limits in practice.

# Contents

# 1 Introduction

As Australia's electricity grid transitions into a more sustainable form, increasingly consumers are taking up part of the charge by contributing self-managed energy resources like solar panels, electric vehicles and even static home energy storage solutions. In extreme cases, these resources may be used to completely disconnect from the main grid; however, it is much more likely that these resources will be used to support the consumer's load profile and reduce the impact it has on the external grid. Unfortunately, because the grid was not designed with this decentralised structure of energy generation in mind, our low- and medium-voltage grids are quickly running into hosting capacity limitations. These prevent subsequent customers from installing and taking advantage of such self-managed resources, *and* reduce the full export potential of installed units. There is not one silver bullet to this problem; resolving this challenge requires a multi-pronged approach of network infrastructure investment, technical solutions on the operation of distributed energy resources, and energy awareness education. This report tackles part of the problem by engaging with the technical challenge of operating a fleet of energy resources as efficiently as possible, by improving the integration between forecast and optimisation.

The fundamental challenge of residential solar generation is the mismatch between time of production and time of use, due to intermittency of solar power. Even during the stay-at-home orders implemented to mitigate 2020-2022 COVID-19 pandemic, household consumption was unlikely to exceed daytime solar production for long, because many consumers opt to install arrays based on their total daily load (net-zero), while from a grid operation perspective a nearly flat (absolute-zero) demand profile would be ideal. Residential energy storage offers a potential solution to this challenge, by allowing users to store energy from solar panels (or the neighbours' through the local grid) for later use, when local production is not available. At a wider grid scale, having access to multiple such storages allows a grid coordinator to make significant swings in load, for example to support the more traditional central power plants during periods of significant generation ramping. Therefore, aggregations of small energy resources are sometimes referred to as Virtual Power Plants (VPP).

Successful VPP operation requires the centralised coordinator to achieve two goals: 1) accurate forecasts of both load and solar generation at the individual asset level, and 2) an efficient optimal scheduling algorithm to dispatch control decisions to devices. Traditionally, these two tasks have long been seen as separable, implicitly assuming that improving the quality of forecasts in terms of accuracy measured in power signal mismatch will improve the performance of optimisation measured in dollar savings. However, for some combinatorial problems, an inverse relationship has been observed, where improving the forecast beyond a baseline point could end up reducing the quality of decisions found by optimisation (Mandi et al., 2020). For example, it might be better for the consumer to discharge the battery more than strictly necessary according to the forecast, because that allows the battery to buffer more of an unpredicted load swing from a new appliance starting up. In this case, improving the forecast by reducing the overestimation bias will end up hurting the overall objective. In such cases, `predict-plus-optimise' approaches can achieve better performance, by including the optimisation loss explicitly in the prediction objective.

We hypothesise that a `predict-plus-optimise' strategy is also beneficial for VPP operation. In this project, we investigate this hypothesis by developing benchmark forecast, optimisation, and combined predict-plus-optimise solutions, and suggest promising research directions to further improve on them. We enable the wider academic community to easily start tackling this challenge by open-sourcing a dataset of high-resolution historic load and PV signals, as well as our reference implementations of forecast and optimisation.

We have implemented a traditional forecast system based on state-of-the-art forecasting algorithm LightGBM (Ke et al., 2017) as a benchmark. Forecasts from this system are taken as input of a battery scheduling implementation in similarly state-of-the-art constraint programming language MiniZinc developed at Monash University (Nethercote et al., 2007). Solving the model end-to-end, we can capture 67.5% of the available value from battery scheduling and flatten peak loads by about 31.8% if we implement the schedule across two days in July. Nevertheless, we observe that better battery utilisation may be obtained for artificially biased (i.e. worse in error) forecasts, motivating a predict-plus-optimise strategy to reduce the mismatch between forecast error and optimisation objective. We show how to address this mismatch by making use of a model-predictive control strategy, as well as a custom loss function for the forecast system.

## 1.1 Report structure

The remainder of this report is structured as follows. First we discuss the data cleaning and exploration in Chapter 2, including the process of feature selection and the construction of the actual forecasts. Subsequently, Chapter 3 describes the optimisation problem from the perspective of finding an optimal battery schedule for a given forecast. Chapter 4 uses the models from the previous two sections to implement a predict-plus-optimise strategy. Chapter 5 discusses the practical aspect of running the software package; finally Chapter 6 summarises our findings and presents pathways for future work.

# 2 Forecasting models

## 2.1 Background

Energy optimisation can provide significant financial benefit for customers owning renewable energy assets such as solar panels and batteries. By forecasting both household solar power generation and load power, the household battery can be optimally charged and discharged to minimise the customer's energy costs. Forecasts of solar power generation and load power at the household level were developed using data provided by SwitchDin and external weather data obtained from Solcast (Solcast, 2023). An in-depth exploratory data analysis (EDA) was performed to build a foundational understanding the data. The EDA helped identify data issues that were the result of on-boarding and off-boarding of assets including duplicate assets with different recorded values of power, inverted load power, and long sequences of zeroes.

Following the in-depth exploratory data analysis (EDA), models were developed for both solar power generation and load power. We demonstrated the importance of a direct global modelling framework when forecasting for each household across a large number of households at low-frequency time intervals over a multi-day horizon. These models were developed in the research environment with which to compare predict-and-optimise methods. They are quality baselines and can be improved with further modelling iterations and refinements. In addition to accuracy, a primary focus is ensuring the reliability of these models, aligning with the objectives of the industry partners. Such reliability is important for deploying the models in a production setting.

The models were evaluated using the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) and the results compared with standard benchmarks outlined in the *Project Plan*. In future work, the model can be improved by incorporating additional inputs like public holidays and events, through further feature engineering, and with hyperparameter tuning. However, this sequence of model development actions should be considered only after a pilot deployment in a limited production environment, given the data quality issues observed through the EDA and the inherent uncertainties of relying on external weather data.

## 2.2 Model Development Process

We developed the solar power generation and load power model using the modelling lifecycle illustrated in Figure 2.1. Residential household solar power generation and load power were collected by SwitchDin and preprocessing by Monash researchers. The datasets were cleaned and an EDA performed to understand the structure, patterns, and anomalies in the datasets. Particular attention was paid to identifying potential data issues which provided insights that guided the pre-processing steps before model development.

The direct global modelling architecture used the LightGBM gradient-boosted decision tree algorithm (Ke et al., 2017) for model training due to its high efficiency, scalability and capability when handling large feature dimensions and massive datasets. The forecasts were evaluated over the full two-day horizon and across smaller intervals within that period.

In production environments, real-time forecasts are crucial for timely integration into optimisation processes. Any latency in forecasting can compromise these processes. The models developed in this research are prototypes, designed with a dual emphasis on fast inference and accuracy. One advantage of the direct global modelling architecture is that each model forecasts one time-step independently, allowing for parallelised inference. Additionally, direct models are trained to forecast for a specific horizon, enabling them to capture patterns specific to that horizon, which non-direct recursive forecasting models may overlook.

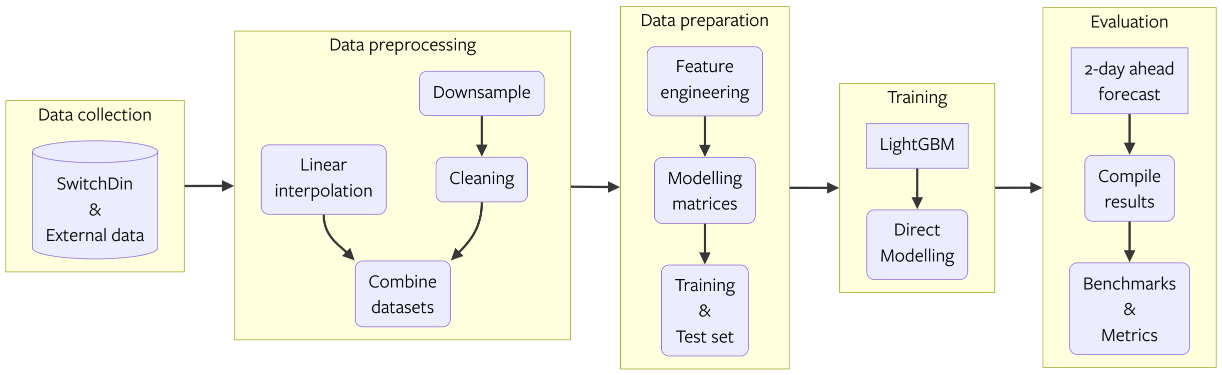


**Figure 2.1:** The modelling lifecycle is an iterative process that includes data collection and preprocessing, model preparation, training, and evaluation. These stages are interconnected and aim to build accurate and reliable models that can be monitored and updated over time.

## 2.3 Exploratory Data Analysis

Raw data on solar power generation and load power, at a one-minute resolution, were collected from SwitchDin for 273 households in Adelaide, South Australia, spanning from July 2021 to August 2022. The dataset resolution was downsampled from 1-minute to 5-minute intervals to balance the need for granularity with the computational resources available for this project. We developed a data cleaning pipeline to address anomalous readings associated with telemetry communication failures, different times associated with on-boarding and off-boarding of customer assets and even incorrect asset inverter connections by installers. To ensure a reliable dataset for model development, we implemented rigorous cleaning processes. Furthermore, we narrowed down the homes for modelling to those nearest the city center using $k$-means clustering, given that the weather data included in the models originates from a single location in the city center. This refinement reduced the dataset to 136 households.

### 2.3.1 Weather

Hourly weather and solar irradiance data were obtained from Solcast (Solcast, 2023). The data included air temperature, cloud opacity, humidity, precipitable water, diffused horizontal irradiance (DHI), diffused normal irradiance (DNI), direct (beam) horizontal irradiance (EBH), global horizontal irradiance (GHI), global fixed tilted irradiance (GTItilt), and global track irradiance (GTItrack). The data were collected from June 2020 to August 2022 from a central location in Adelaide, South Australia, which is proximate to the households in the solar power generation and load power dataset. University researchers may be eligible to access these data for non-commercial purposes at no cost if they have not previously accessed them.

While the load power model incorporated only weather inputs and excluded solar irradiance, the solar power generation model included various measures of solar irradiance. This inclusion was not only to capture nuances that a single measure might miss but also to enhance model robustness. This consideration is crucial, as the model relies on external weather data sources whose reliability in real-time production environments is unknown. Although measures such as Global Horizontal Irradiance (GHI) and Direct Beam Horizontal Irradiance (EBI) were highly correlated (Figure 2.2), both were included to ensure the model's resilience against potential data feed failures from a single irradiance source. Most residential households use a GTI fixed tilted system, but the tilt angle varies among households. Since the GTI data came from a single source and therefore not tailored to each household, the GTI fixed tilted irradiance was excluded from the model. Tracked systems are uncommon in residential setups and have also been left out of the solar power generation model.

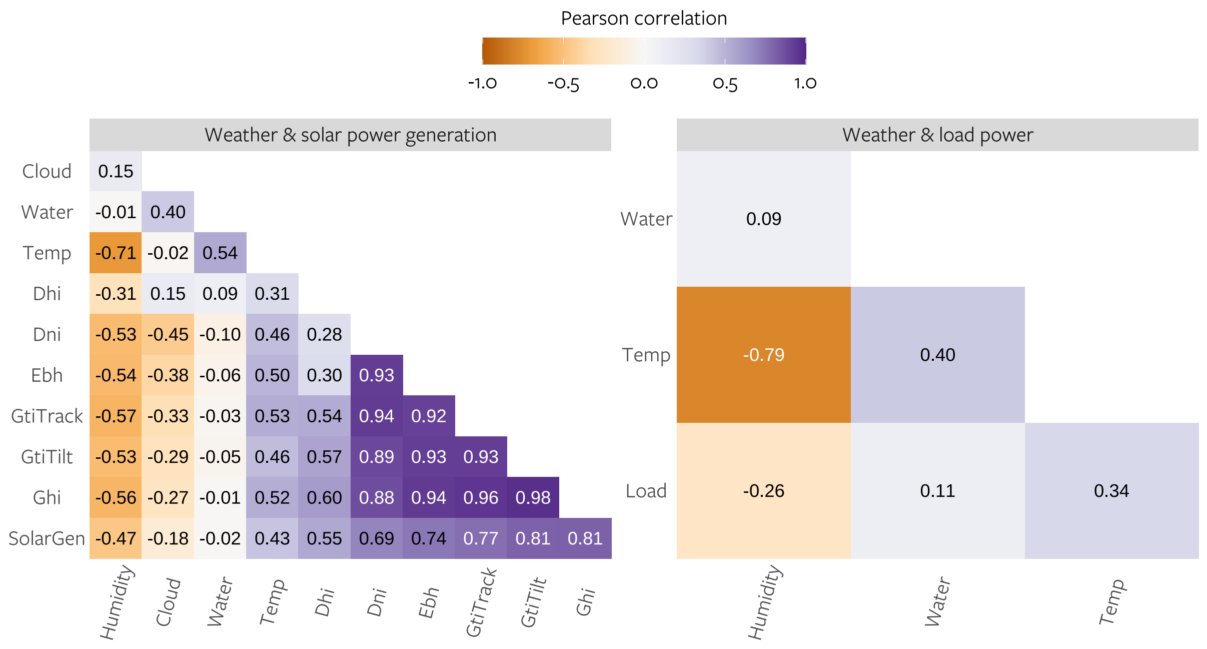


**Figure 2.2:** The correlation coefficient between each weather variable and solar power generation (left) and load power (right) is shown in each box, with the magnitude of the correlation coefficient represented by the colour intensity. Positive correlations are represented by purple, while negative correlations are represented by orange. The right subplot displays the relationship between load power and weather variables for January only due to significant variability across months.

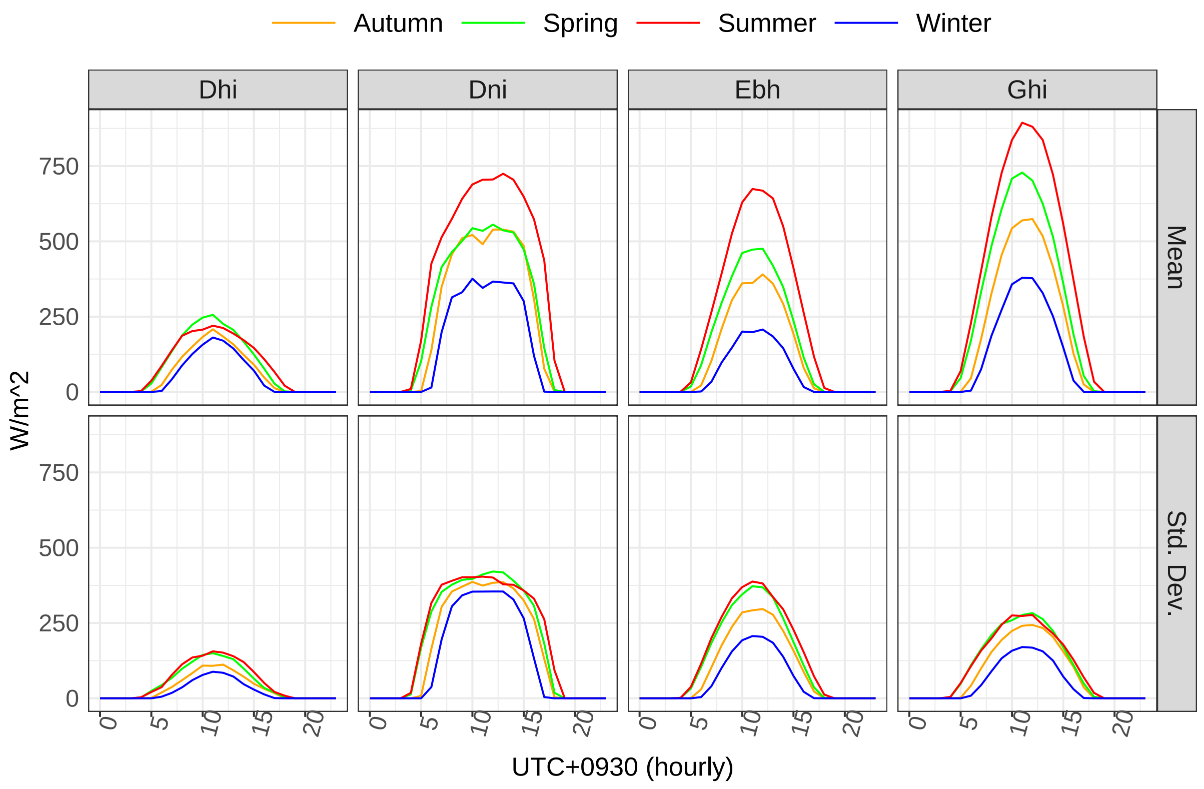


**Figure 2.3:** Hourly mean and standard deviation of four measures of solar irradiance collected from June 2020 to August 2022 at a single location.

Publicly available historical data on weather and solar irradiance forecasts were only available for the most recent month. Therefore, actual historical weather data were used to train the solar power generation and load power model instead of historical weather forecasts. The results in this research are expected to be better than those achieved in a live production environment. In the production environment, live weather forecasts, rather than actual weather, will be directly fed into the model's inputs. This research used actual weather as a proxy due to the unavailability of historical weather forecasts.

The hourly average of the four measures of solar irradiance for each season are shown in Figure 2.3. All measures increase from the 5th hour, peak at midday, and decrease to zero $W/m^2$ by the 17th hour. On average, solar irradiance is highest in summer, but its variability is highest in both spring and summer. This shows that solar power generation is a function of the time of the day and the time of year, so we included date-time information when developing our solar power generation model.

### 2.3.2 Solar and load power

The exploratory data analysis (EDA) indicated that the solar power generation data was relatively clean. In contrast, the load power data needed substantial cleaning and validation to ensure its reliability for modelling. A thorough data cleaning procedure was implemented, which included identifying connection errors, removing duplicate data, replacing erroneous values with NAs, and discarding time series with inadequate data points. Figure 2.4 displays examples of data needing cleaning, such as inverted load power and extended sequences of zeroes. Some homes had more missing data than available data, resulting from telemetry communication failures post-onboarding or due to incorrect asset inverter connections by installers. Consequently, they were excluded from modelling. These anomalies, which are artifacts from the initial data collection and setup processes, posed risks to the model's accuracy if left unaddressed.

Solar power generation follows a consistent pattern across all homes, with generation typically ramping up at the 5th hour and tapering off at the 17th hour of the day, regardless of the season (Figure 2 .5). Between these hours solar power generation can vary significantly depending on the weather c onditions. On clear and sunny days, solar irradiance is consistent and strong, resulting in smooth power generation. Conversely, on cloudy or rainy days, less sunlight reaches solar panels, resulting in fluctuating power g eneration. This predictable pattern of solar power generation renders the development of an accurate forecasting model more feasible compared to a load power model.

Accurately forecasting load power for each household independently is significantly more challenging than forecasting solar power generation. This is because load power depends on many heterogeneous factors, such as household size, individual lifestyle, and appliance usage. The underlying structure in load power is often noisier than solar power generation, making it even more difficult to develop accurate forecasting mo dels. Figure 2.6 shows an example of load power that appears to have a consistent, regularly spaced pattern. If this pattern persists, both the naïve and seasonal naïve forecasting methods will be able to generate accurate forecasts. The naïve forecasting method assumes that the most recent observed value will be the same as the future value. For instance, when forecasting load power 1-day ahead with a naïve method, yesterday's value of load power is used as a forecast. Similarly, the seasonal naïve method assumes that the most recent observed value from the same seasonal interval will be the same as the future value. For example, when forecasting load power 1-day ahead on a Monday, last Monday's value of load power is used as a forecast. Both methods, however, often mistime local peaks, leading to larger errors. When the forecast of these methods were evaluated, the mean method outperformed the naïve and seasonal naïve methods, producing a lower RMSE and MAE[1] by 28.9% and 23.3%

[1] Defined in Section 2.6, Equation (2.1).

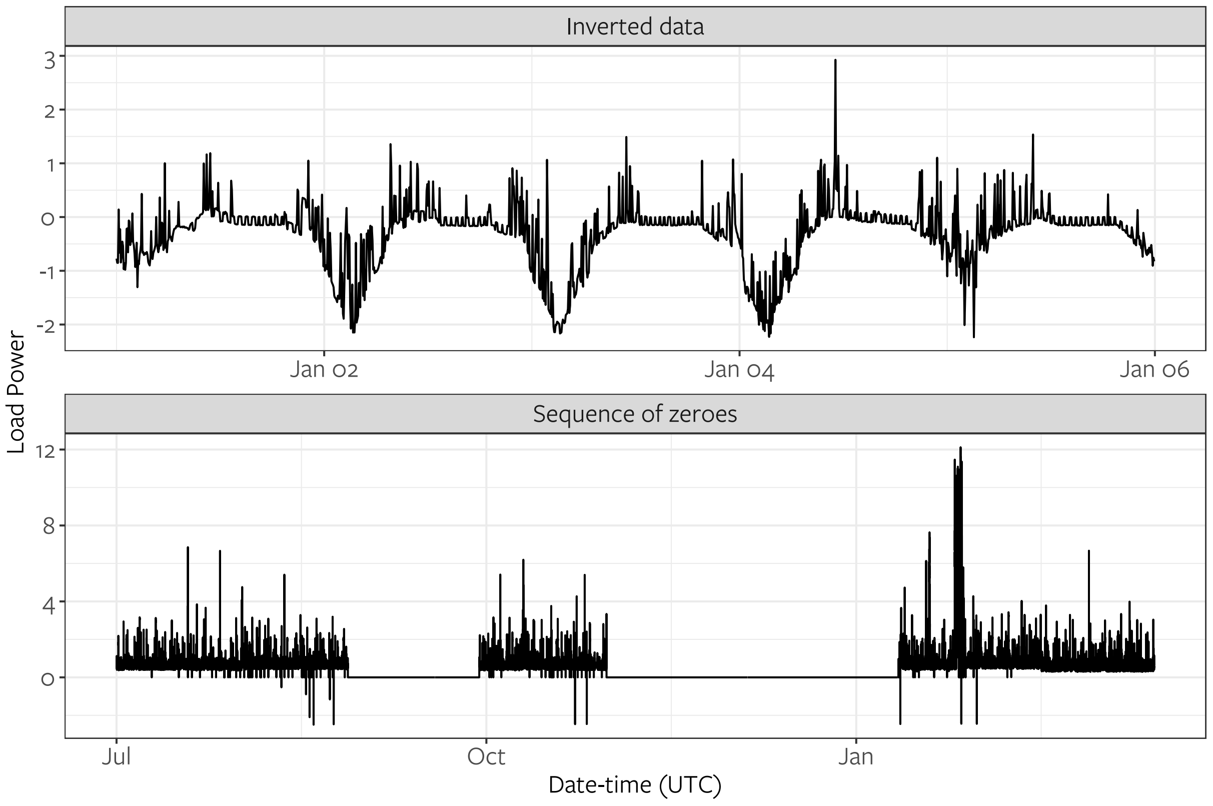


**Figure 2.4:** Data preprocessing is an essentials step in the modelling lifecycle, aimed at preparing the data for analysis and model development. Load power from some households in the data exhibit anomalies as such inverted load power due to connection errors and prolonged sequences of zeroes.

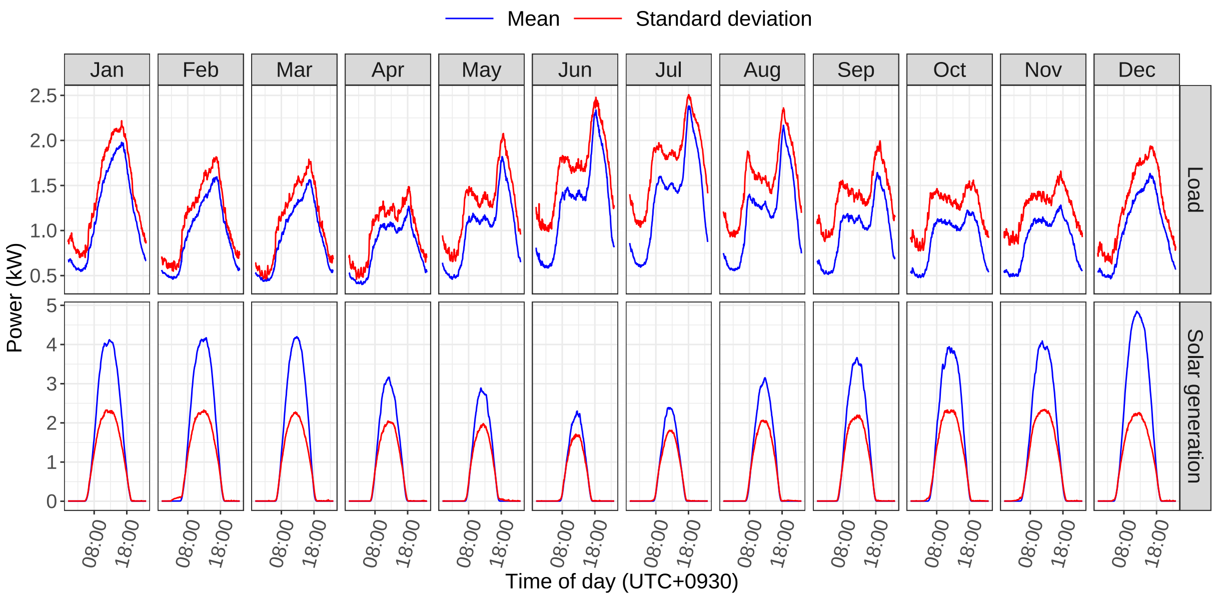


**Figure 2.5:** The mean and standard deviation of solar power generation and load power across different times of the day and months. Load power variability is greater than the average load power, whereas solar power generation variability is typically below its average. In June and July, the variability in load power is at its highest, whereas the variability in solar power generation is at its lowest - typical of southern hemisphere cycles. The shape of average load power varies by month, whereas average solar power generation maintains a consistent shape throughout the year.

when compared with the naïve method and 27.5% and 22.2% when compared with the seasonal naïve method.

Load power was modelled using an ARIMA model. The autoregressive and moving average orders were automatically selected using the Hyndman-Khandakar algorithm (Hyndman and Khandakar, 2008). Additionally, Fourier terms were incorporated to account for daily seasonality. The results are illustrated in Figure 2.6. Although the forecasting accuracy of the ARIMA model closely mirrors the mean forecast, the process of fitting the ARIMA model to the load power of just one household took 55 minutes on a standard computer. It is important to note the inherent challenges associated with forecasting due to the data's immense volume and frequency. Given that there are 136 households, each requiring two distinct forecasts (solar power generation and load power) at 5-minute intervals for a one-month test period, traditional statistical models like ARIMA may not be as suitable. Modern machine learning algorithms, built for distribution and efficiency in both training and memory usage, offer a more practical solution.

## 2.4 Modelling Methodology

### 2.4.1 Global vs local models

The decision to train either global or local models for forecasting solar power generation and load power depended on the size of the dataset and the patterns observed across each household. A local model is developed specifically to forecast power for an individual household, relying solely on data from that particular home. For context, a local modelling framework to forecast power for 136 homes requires training 136 distinct local models. Conversely, a global model is designed to forecast power for any household in the dataset, by training across data pooled from all homes without the need to identify each home with indicator variables.

Local models may be the preferred choice in scenarios with fewer homes, especially when the power consumption and generation patterns of individual households differ significantly from one an other. When forecasting power across a multitude of homes, the global modelling approach is preferred. This is because the global model is trained using data pooled from multiple homes, benefiting from a larger dataset and thus increasing accuracy. As such, global models are more robust than local models. For instance, some households may have limited historical data available for model training, potentially compromising the accuracy of local models. In contrast, global models are trained on a larger, pooled dataset of multiple households, leveraging patterns and insights from the collective data. This allows them to provide reliable forecasts even for homes that may not have had sufficient individual historical data for a local model to be as effective. Global models also have the capability to learn unique patterns inherent to each household by incorporating local features.

Figure 2.7 illustrates the differences in solar power generation and load power across a random sample of three residential homes. All three homes exhibit the same daily seasonality in their solar power generation; however, load power patterns differ across each home. The third home shows the clearest pattern of daily seasonality in load power, with increased energy consumption before sunrise when solar power generation begins to ramp up. The second home shows increased energy consumption during sunset when solar power generation begins to taper off.

### 2.4.2 Model architecture

In our investigation of forecasting methods, we explored two global architectures: a global single modelling approach and a global direct multi-modelling approach. Both architectures train univariate models on data from all residential homes, enabling cross-household learning that outperform models trained on isolated time series. This research builds on a body of work at Monash University that has been at the forefront of global forecasting (Bergmeir, 2023).

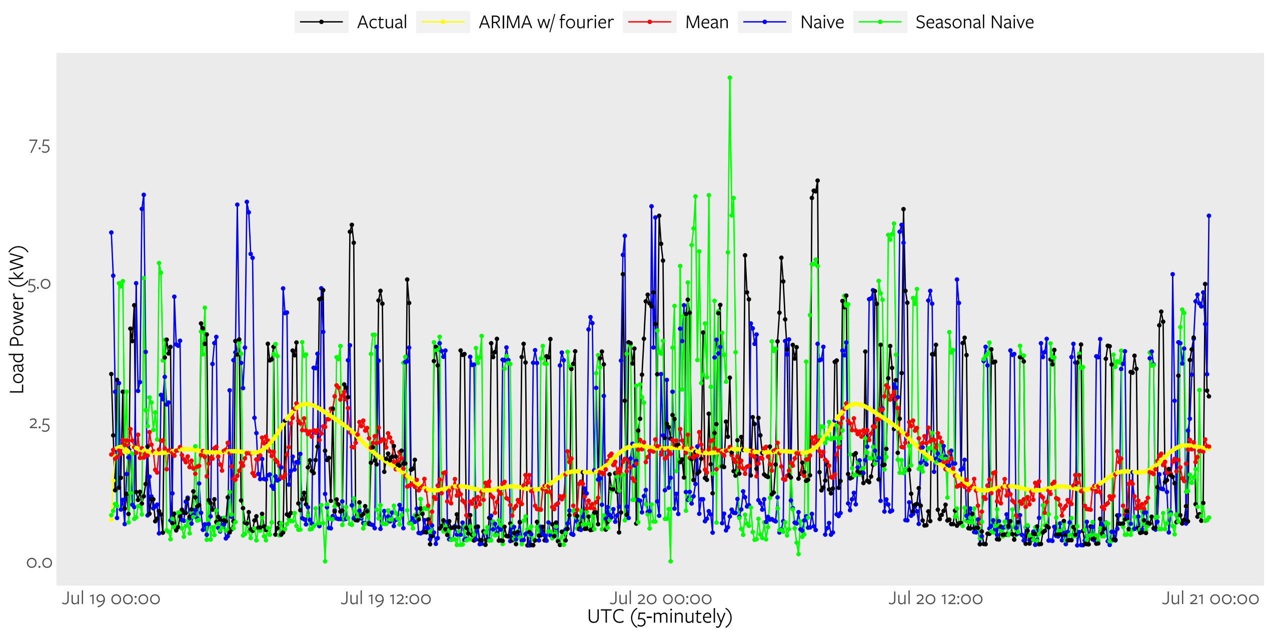


**Figure 2.6:** Load power forecast for a single home using standard benchmark methods (mean, naïve, seasonal naïve) and an ARIMA model. Although the naïve and seasonal naïve methods appear to produce accurate forecasts in the graph, the mean method outperforms them both in terms of RMSE and MAE. The repeating structure from the naïve and seasonal naïve methods, which seem to overlap the actual load, can create a misleading impression of their accuracy. A closer examination reveals many instances of mistimed local peak forecasts, leading to larger errors in the naïve and seasonal naïve methods.

**Table 2.1:** Accuracy of 48-hour ahead forecasts from load power modelling methods for a single unit (shown in Figure 2.6).

| Model | RMSE | MAE |
|---|---|---|
| Naïve Seasonal | 2.1054 | 1.4612 |
| Naïve Mean | 2.0666 | 1.4405 |
| | 1.4958 | 1.1194 |
| ARIMA w/ fourier | 1.4449 | 1.2041 |

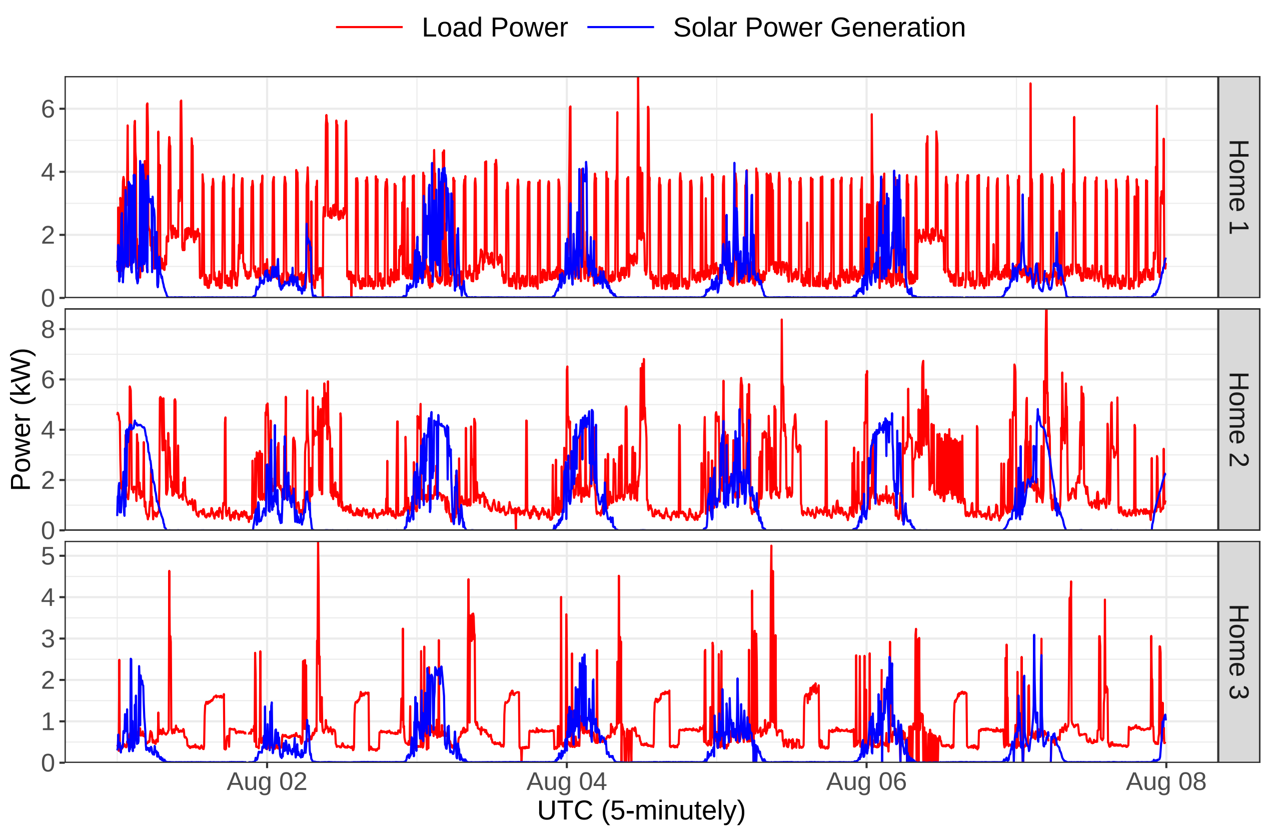


**Figure 2.7:** Three homes with solar power generation (blue) and dissimilar load power patterns (red) from July 1st to 7th, 2021. The dissimilarity across homes presents modelling challenges without localised features.

The total training and inference runtimes differ drastically between single and direct multi-modelling. The single modelling approach requires recursive prediction, also known as ″rolling forecasting″, to produce forecasts that cover the forecast horizon. This involves using predicted values from prior time steps to forecast subsequent time steps. At each timestep, the predicted values from a previous time step are fed back into the model as inputs for forecasting the next time step. When using a rolling forecasting technique to make predictions over extended periods, the single model must be executed repeatedly, leading to a significant increase in computation time when forecasting over large test sets.

The direct multi-modelling approach overcomes this time-consuming hurdle by training a separate model for each forecasting time step. As a result, the total training time increases; however, each model can directly output a prediction for the specific time step that it is trained to forecast. This improvement is important, because reduced computation time permits a wider range of modelling experiments to search for the highest accuracy solar power generation and load power model. Furthermore, direct models are not prone to compounding errors that occur in recursive predictions when small errors from a previous prediction are accumulated and passed towards the next prediction.

The LightGBM algorithm was chosen to train both the solar power generation and load power models in the global direct multi-modelling architecture due to its efficiency, scalability, and ability to handle large feature dimensions and massive datasets. LightGBM is a gradient-boosted decision tree algorithm developed by Microsoft that has been shown to outperform other gradient boosting frameworks in terms of training speed, memory usage, and accuracy in various machine learning tasks. It has demonstrated superior accuracy and computational efficiency in real-world applications, such as sales forecasting (Makridakis, Spiliotis, and Assimakopoulos, 2022), credit risk prediction (Wang, Li, and Zhao, 2022), fraud detection (Ge et al., 2020), and energy consumption prediction (Ullah et al., 2022). LightGBM's excellent performance in the M5 forecasting competition (Makridakis, Spiliotis, and Assimakopoulos, 2022), where the winning solutions used ensembles of models trained with LightGBM, has made it a popular choice in the machine learning community for supervised learning problems.

### 2.4.3 Feature engineering

After preprocessing the solar power generation, load power, weather, and solar irradiance data, the datasets were combined and feature engineering performed to create the features for each model (Figure 2.8). The solar power generation and load power models were trained on a range of features, including date and time information, up to seven days of lagged values, and various statistics computed from the training data. These statistics included the mean, median, maximum, minimum, and standard deviation of each home's solar power generation and load power at the monthly, daily, and hourly aggregates. Rolling statistics were calculated with various window sizes such as 10, 15, and 20 minutes, to capture temporal dependencies in the data. These features were also generated for each weather input such as solar irradiance, air temperature, cloud coverage, humidity, and precipitable water.

The model can be further refined by incorporating inputs including public holidays and events, or applying advanced feature engineering techniques such as automated feature selection, Principal Component Analysis (PCA), and feature scaling techniques, but the impact of potential data quality issues on these derived features remains unclear until tested in limited production.

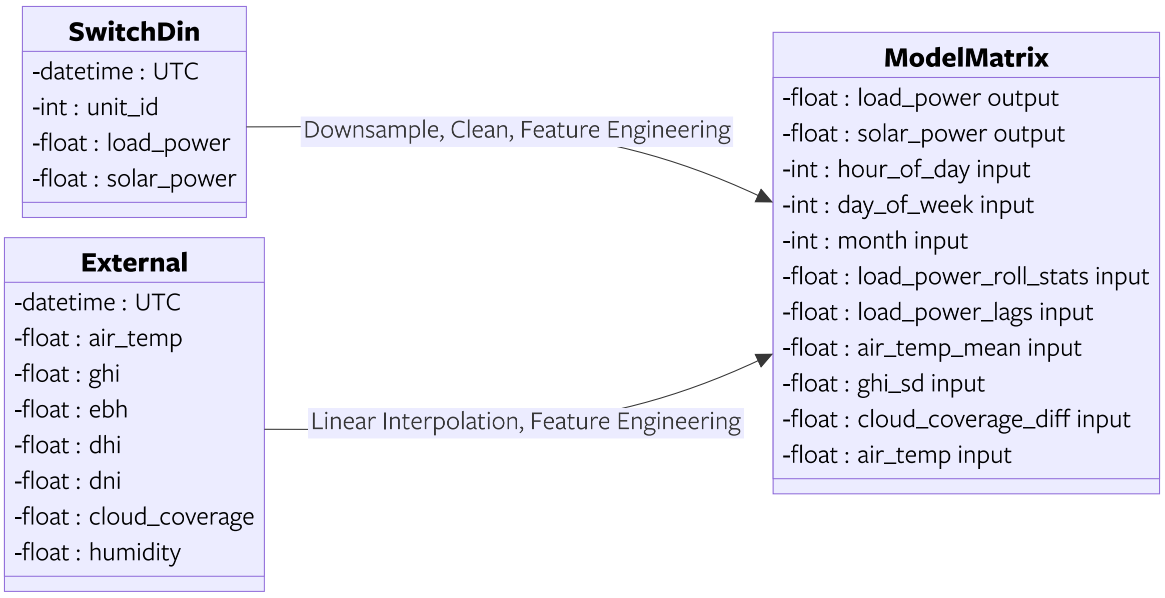


**Figure 2.8:** The structure of solar, load and weather data and processing steps to generate the data required for modelling. The features shown are a sample of the total features incorporated in the final model.

Given the data quality concerns identified from the EDA and the inherent uncertainty in relying on external weather data, the models developed contained standard features to aid in future work diagnosing potential data feed issues when tested in a pilot deployment. This approach minimises challenges for future work transitioning to initial testing and monitoring in a limited production environment. Emphasising robustness involves maintaining simplicity and transparency in the model inputs. This not only aids in diagnosing potential issues in a limited production setting but also streamlines the verification of external data f eeds. As the model is tested and monitored, opportunities for further feature engineering and the integration of additional datasets can be explored.

## 2.5 Forecasting

We developed direct global LightGBM models that produce 5-minute point forecasts of solar power generation and load power up to 48 hours in advance, using historical solar and load data, weather data, and time of day as inputs. These raw inputs are used to generate features required for the models to output forecasts. Initially, a recursive forecasting strategy was used; however, we found that direct forecasting, which involves training many models (one for each step ahead forecast), significantly improved prediction runtime and accuracy. Using the direct modelling approach required training 576 models, each predicting a single 5-minute period, to span a 48-hour horizon.

Figures 2.9 and 2.10 present hourly forecasts for a 48-hour span, starting from 2022-07-20 06:30 ACST and up-dated every hour for the first 9 hours. Figure 2.9 reveals the daily repetition in load power for both residential homes. Specifically, Home 3 displays elevated load power at approximately 07:00 and again near 18:00. Home 4 displayed a gradual increase in load power around midnight, which persisted for several hours before sharply declining around 02:00. Additionally, there were notable load power fluctuations close to 07:00. In Figure 2.10, both homes exhibit a similar pattern of solar power generation, though Home 1 produced more power than Home 2. The global direct load models depicted were successful in capturing these variations in daily repetition for both homes, as evidenced by the quality of the forecasts.

## 2.6 Results and Evaluation

The naïve, seasonal naïve models, and the household historical mean (grouped by month and time of day) are the standard benchmarks outlined in the *Project Plan*. They are used for comparison against the global direct LightGBM models for solar power generation and load power. The test set for evaluating each modelling method starts at midnight UTC on July 19th, 2022, and concludes at 23:00 UTC on August 17th. Forecasts are generated every 5 minutes, covering a 48-hour time horizon, and are assessed over 1-hour, 3-hour, 6-hour, 12-hour, 24-hour, and 48-hour periods.

Data missing in the test set were excluded from evaluation. The mean absolute error (MAE) and root mean squared error (RMSE) serve as metrics to evaluate forecasts from each modelling method. The MAE quantifies the average distance between actual and forecasted values (absolute errors), while the RMSE averages the squared distance between them (squared errors), thereby penalising larger errors more than the MAE. Lower MAE and RMSE values indicate better model accuracy.

$$\begin{aligned}\text{MAE} &= \frac{\sum_{t=1}^{T} |\textit{actual}_t - \textit{forecast}_t|}{T} \\ \text{RMSE} &= \sqrt{\frac{\sum_{t=1}^{T} (\textit{actual}_t - \textit{forecast}_t)^2}{T}}\end{aligned} \tag{2.1}$$

The accuracy of solar power generation and load power forecasting is significantly improved using a global direct modelling architecture trained with the LightGBM algorithm. Our results demonstrate that this approach produces models that outperform benchmark methods across all metrics at each forecast horizon (Table 2.2).

Although our models outperform all standard benchmarks based on the MAE and RMSE, it is important to note that not all forecast errors have equal significance. In applications like solar power generation and load power

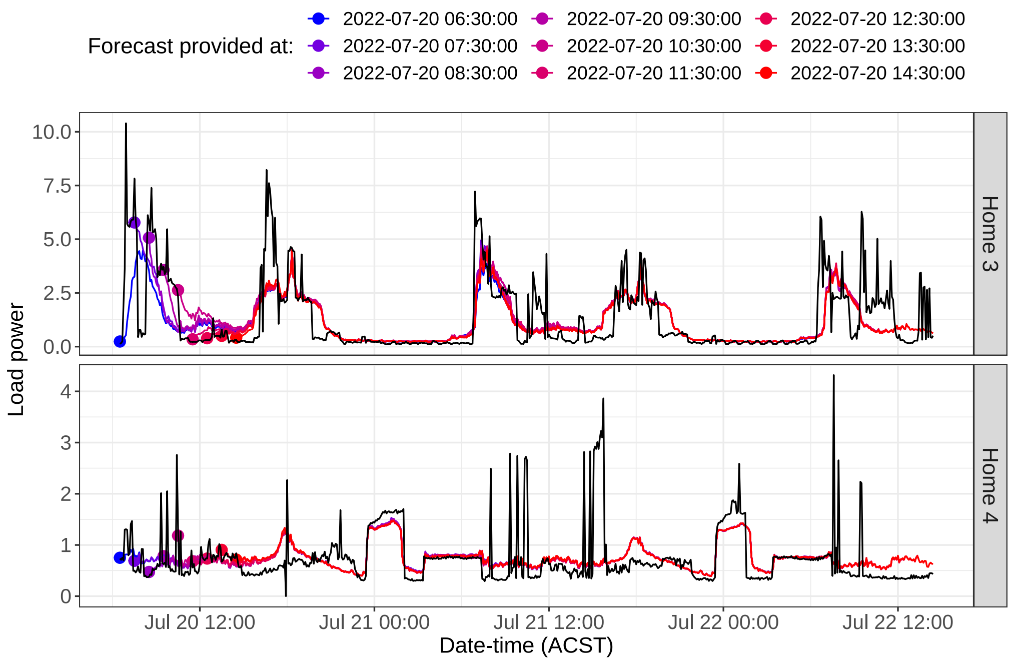


**Figure 2.9:** Our load power forecast (coloured lines) for four households compared to the actual data (black line). The plot displays forecasts for the upcoming 48 hours, updating every hour for a total of 9 times. The dots represent the start of each set of forecasts.

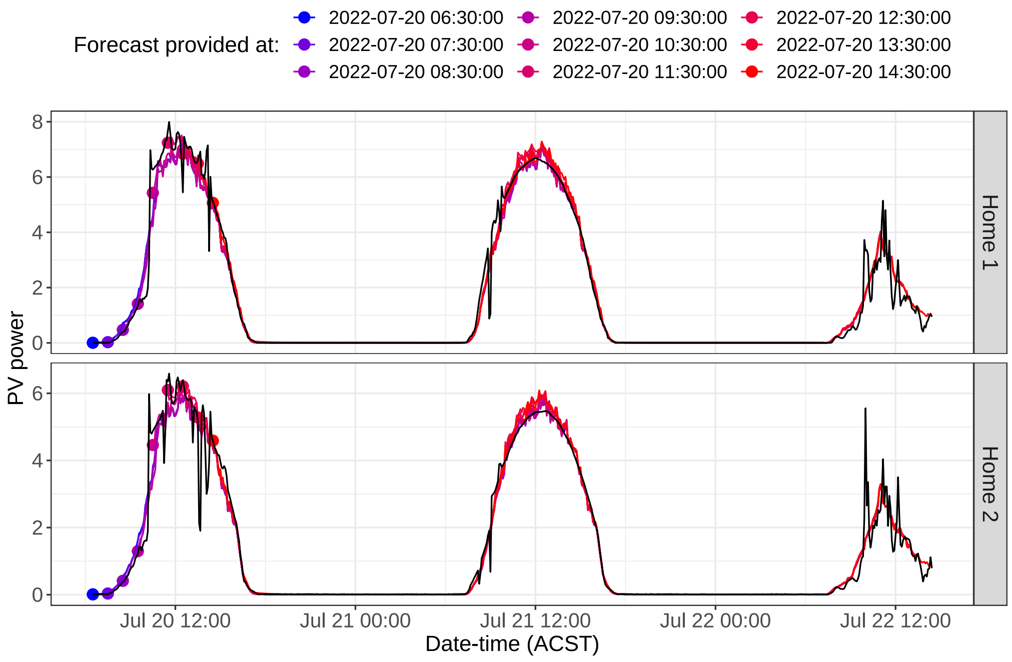


**Figure 2.10:** Our solar power generation forecast (coloured lines) for four households compared to the actual data (black line). The plot displays forecasts for the upcoming 48 hours, updating every hour for a total of 9 times. The dots represent the start of each set of forecasts.

forecasting, it may be more important to accurately predict certain periods such as peak demand than others. Several techniques can be used to complement the MAE and RMSE, such as Mean Absolute Percentage Error (MAPE), Mean Absolute Scaled Error (MASE), Weighted Absolute Percentage Error (WAPE), Symmetric Mean Absolute Percentage Error (SMAPE), and Weighted Symmetric Mean Absolute Percentage Error (WSMAPE). However, further analysis is required to understand the varying importance of forecast errors across solar power generation and load power, as well as different time periods and households, before deciding how to prioritise one evaluation metric over another.

**Table 2.2:** Accuracy of forecasts from solar power generation and load power modelling methods on all households.

| | | | Forecast horizon | | | | | |
|---|---|---|---|---|---|---|---|---|
| Output | Metric | Models | 1-hour | 3-hour | 6-hour | 12-hour | 24-hour | 48-hour |
| Solar | MAE | Naïve | 0.8752 | 0.8461 | 0.9137 | 0.9223 | 0.9166 | 0.9247 |
| | | Mean | 0.7824 | 0.7801 | 0.8561 | 0.8615 | 0.8566 | 0.8672 |
| | | Seasonal Naïve | 0.9013 | 0.8838 | 0.9639 | 0.9737 | 0.9681 | 0.9784 |
| | | LGBM direct models | 0.3773 | 0.4296 | 0.4499 | 0.4862 | 0.5299 | 0.5932 |
| | RMSE | Naïve | 1.6117 | 1.6178 | 1.7428 | 1.7611 | 1.7553 | 1.7713 |
| | | Mean | 1.2340 | 1.3040 | 1.4293 | 1.4405 | 1.4387 | 1.4570 |
| | | Seasonal Naïve | 1.6399 | 1.6736 | 1.8384 | 1.8567 | 1.8519 | 1.8729 |
| | | LGBM direct models | 0.7913 | 0.9647 | 1.0230 | 1.0781 | 1.1418 | 1.2338 |
| Load | MAE | Naïve | 0.4702 | 0.4906 | 0.4779 | 0.4796 | 0.4785 | 0.4777 |
| | | Mean | 0.3465 | 0.3625 | 0.3582 | 0.3595 | 0.3591 | 0.3590 |
| | | Seasonal Naïve | 0.4061 | 0.4243 | 0.4156 | 0.4168 | 0.4164 | 0.4158 |
| | | LGBM direct models | 0.1291 | 0.1372 | 0.1420 | 0.1569 | 0.1715 | 0.1862 |
| | RMSE | Naïve | 1.0188 | 1.1364 | 1.1054 | 1.1156 | 1.1148 | 1.1143 |
| | | Mean | 0.7069 | 0.8050 | 0.7845 | 0.7934 | 0.7938 | 0.7936 |
| | | Seasonal Naïve | 0.9114 | 1.0143 | 0.9913 | 0.9978 | 0.9990 | 0.9988 |
| | | LGBM direct models | 0.3886 | 0.4379 | 0.4410 | 0.4676 | 0.4873 | 0.5082 |

## 2.7 Conclusion and Recommendations

In this research, we demonstrate the crucial role of global direct models in modern residential energy systems by presenting solar power generation and load power models that are robust and outperform standard benchmark methods. The global direct LightGBM solar and load models are effective in identifying and accurately forecasting the variation in daily seasonality across residential homes. However, developing these models posed several challenges and limitations. The absence of publicly available historical weather forecast data required the use of actual historical weather as proxies for weather forecasts. This drawback leads to forecasting accuracies in this research that are expected to surpass those in a production environment as the models would be provided with live weather forecasts that were not available for model training. The difference in accuracies would be greatest for the solar power generation model, as its dependence on solar irradiance is greater than load power's dependence on weather patterns.

The noisy and incompleteness of load power data required careful preprocessing to generate reliable data for modelling. Load power also depends on heterogeneous factors such as household size, individual lifestyle, and appliance usage, so including household level data, such as appliances, smart home sensors, and electric vehicle ownership, could significantly enhance forecasting accuracy.

Our research advances the modelling of solar power generation and load power forecasting in modern residential energy management systems by offering a robust modelling architecture that exceeds standard benchmark accuracies. Moreover, the forecasting accuracy of the ARIMA model for load power closely paralleled that of the mean forecast, one of the standard benchmarks. While an ARIMA model estimation for the load power of a single household required 55 minutes on a standard computer, calculating the historical mean took less than a second. However, there is significant potential for future research to further enhance forecast accuracy, leading to improvements in battery optimisation.

The 1-minute solar power generation and load power data were downsampled to 5-minute intervals to be processed with the available resources. The modelling datasets, containing separate lagged features for each direct model, remained significantly large. Future work should explore cloud-based computing resources to expedite processing tasks during the data preparation stage of the model development cycle.

Hyperparameter tuning implemented in the direct global modelling architecture is highly computationally intensive, as there are 576 direct models which may each require separate tuning. Future work considering hyperparameter tuning with a direct modelling approach should start with a simplified hyperparameter tuning strategy to reduce computational demands. This strategy could involve more refined tuning for short-term direct forecast models and less detailed tuning for longer-horizon direct models. Given the data quality concerns and inherent uncertainty in relying on external weather data, testing in a small production environment should be considered before investing computational resources on hyperparameter tuning. These tests should examine whether the external weather data remains consistent during inference in a production setting.

# 3 Centralised optimisation solutions

## 3.1 Optimisation problem statement

Given a collection of distributed energy resources (DERs) such as solar panels and batteries, that are aggregated as a virtual power plant (VPP). Then, our goal is to calculate a valid forward operating schedule for each of the individual devices such that the load shape of the VPP is as `ideal' as possible. What constitutes an `ideal' load shape depends on the external objectives of the operator, and could include maximising profit in the wholesale energy market, responding to frequency control requests, and minimising the monthly peak load.

To achieve a forward scheduling, the VPP has to optimise the schedule against a forecast of the uncontrolled load, such as the day's expected solar generation. Furthermore, the schedule has to respect DER constraints at all times; constraints include physical limits such as the capacity of a battery and local network voltage bounds, as well as DER owner's requirements such as having sufficient reserve charge in the battery to bridge a power outage.

### 3.1.1 Predict-plus-optimise objective

The goal of the system is therefore to minimise the VPP's *actual* operating costs subject to operating constraints. However, the actual operating costs depend on the actual base load and solar generation characteristics, as well as what events happened during the day (such demand-response requests). This information is revealed gradually over time, which in turn can influence subsequent control decisions. As such, we are optimising for the schedule that minimises the hindsight *regret* of the control decisions.

Let $\pi^*(a_t)$ the optimal hindsight control policy for a given day, giving the optimal control action $a_t$. Then, we are looking to optimise a control policy $\pi(a_t \mid \langle o_0, o_1, \dots o_{t-1}\rangle)$ which is conditional on the past observations $o_0$ through $o_{t-1}$. The observations include sensor readings from the VPP as well as external forecasts over attributes such as temperature and irradiance. We evaluate the control policy over a decision-making horizon $h$ (e.g., a day), with an objective function $f_t(a_t \mid t \in [1, \dots, h])$, meaning that the goal is to minimise the regret,

$$\begin{aligned} &\operatorname*{arg\,min}_{\pi} f(\pi^*_t \mid t \in [1, \dots, h]) - f(\pi(o_0, \dots o_{t-1}) \mid t \in [1, \dots, h]) \\ &\text{subject to } \pi \in \Pi \end{aligned} \tag{3.1}$$

where $\Pi$ indicates the set of safe control policies (e.g., with respect to the battery capacity and voltage bounds).

Figure 3.1 shows a schematic overview of the control problem, where the flows of information are annotated as arrows. One special type of observation in this context is the `device properties' of the VPP, which includes the capacity of the batteries, their charge and discharge rates. These structural properties are unlikely to change often, which means that we assume these to be static for most solutions.

## 3.2 Solution methodology

There are multiple ways to solve this problem, depending on the available processing power to compute a solution (once a day or continuously), and whether or not computation should be distributed across the individual devices of the VPP. Solutions can be classified into closed- / open-loop (continually responsive to new observations or not), and centralised / decentralised.

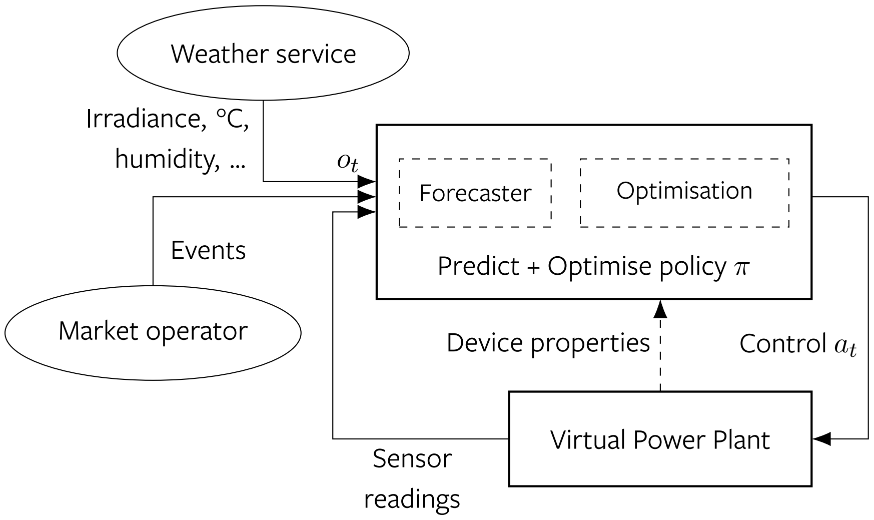


**Figure 3.1:** Conceptual structure of the optimisation problem

### 3.2.1 Optimal hindsight policy $\pi^*$

First, we will define how to compute the ideal goal policy $\pi^*$, if we have access to the ground truth in terms of solar generation and base load. In the process we will define the variables and sets of common constraints that we will reuse throughout this document.

**Battery constraints** A battery control policy is valid if it respects the capacity constraints of the battery, such that its state-of-charge $s_{i,t} \in [\underline{s}_i, \overline{s}_i]$ (in kWh) is always between the lower and upper bounds. Furthermore, we require that the power control signal $\hat{p}^b_{i,t} \in [\underline{p}_i, \overline{p}_i]$ (in kW) also remains within the battery's flow constraints. Finally, there are three more constants that define the battery control problem: step size $\Delta$ in fractions of an hour, battery efficiency ratio $0 < \eta < 1$ translating internal power draw to an external load, and the current (initial) state of charge $\hat{s}_{i,0}$ given as inputs to the optimisation algorithm.

Given these quantities, we can define the linear constraints of battery operation as follows:

$$\begin{aligned}
&s_{i,t} = s_{i,t-1} + \Delta \hat{p}^b_{i,t} && \forall i,t \\
&s_{i,0} = \hat{s}_{i,0} && \forall i \\
&p^b_{i,t} \geq \eta \cdot \hat{p}^b_{i,t} && \forall i,t \\
&p^b_{i,t} \geq {}^1\!/\!{}_\eta \cdot \hat{p}^b_{i,t} && \forall i,t \\
&\underline{s}_i \leq s_{i,t} \leq \overline{s}_i,\ \underline{p}_i \leq \hat{p}^b_{i,t} \leq \overline{p}_i && \forall i,t
\end{aligned} \tag{3.2}$$

The first two constraints tie the state of charge of the battery to the control decisions and initial state. The next two constraints relate the externally observed power to the internal power through the efficiency factor. We use inequalities here to capture the efficiency curve without requiring boolean variables (which would dramatically increase the complexity of the model). Using inequalities is valid as long as our objective function contains a term to minimise $p^b_{i,t}$. Finally, the range constraints to keep the control variables within bounds.

**Tariff constraints** Subsequently, we map the battery control decisions to the change in $-value objective through the tariff function. We assume a two-sided tariff function, with a feed-in (energy export) tariff $c^{\text{EX}}_t$ that is always equal to, or smaller than the energy import tariff $c^{\text{IN}}_t$ at time $t$. This assumption holds for all tariffs currently on the market. Then, assuming that we are given the ground truth base load time series $b^L_{i,t}$, the cost to each agent is captured by the following constraints:

$$\begin{aligned}
&c_{i,t} \geq c_t^{\text{IN}} \cdot (b_{i,t}^L + p_{i,t}^b) \cdot \Delta && \forall i,t \\
&c_{i,t} \geq c_t^{\text{EX}} \cdot (b_{i,t}^L + p_{i,t}^b) \cdot \Delta && \forall i,t \\
&c_i \geq \textstyle\sum_{t=1}^h (c_{i,t}) && \forall i
\end{aligned} \tag{3.3}$$

Here we again make use of the convexity of the piece-wise linear function to avoid having to implement the linear constraints with binary variables. In the rare case that the energy export tariff exceeds the import tariff, this constraint will have to be translated to a piece-wise linear formulation by introducing a binary indicator variable for the sign of the power $p_{i,t}^b$. This will have significant consequences for the complexity of the model, as it would no longer be scale polynomially in the size of the VPP or the planning horizon.

**VPP peak-shaving optimisation** Thus far, the constraints described have had no interaction (coupling) across the different DER devices (every constraint is for-all $i$). As such, the batteries could be optimised *independently*, were it not for a VPP objective. The simplest such objective is a peak load shaving objective, where the VPP is given an incentive to minimise its peak load. Given a \$/kW peak demand tariff $\hat{c}$, the VPP's optimisation model becomes

$$\begin{aligned}
\text{minimise } & \hat{c}p^P + \textstyle\sum_{i=1}^n (c_i) \\
\text{subject to } & p^P \geq \textstyle\sum_{i=1}^n \left(b_{i,t}^L + p_{i,t}^b\right) \quad \forall t \\
& \text{Battery constraints (3.2)} \\
& \text{Tariff constraints (3.3)}
\end{aligned} \tag{3.4}$$

Solving this model results in an optimal ground-truth VPP schedule; as this optimisation model is a linear program, we know that it can be solved to optimality in time polynomial in the size of the problem (the decision horizon and number of batteries). This property makes it an ideal test bed for evaluating the interaction between forecast accuracy and optimisation, because no approximation is necessary.

**Impact of the linearisation** By converting the model to a linear program, we greatly reduce the total time needed to solve the model. However, it is clear that the assumptions about tariff costs may not hold at all times. In those cases, a mixed-integer formulation will have to be used, with binary variables to activate either the import or the export tariff (depending on the sign of the current load). To demonstrate the impact of this change, we perform a small benchmark test with a MIP variant of the model. Figure 3.2 shows the scaling trends for a 50-unit VPP under increasing decision-making horizon length. We observe that the MIP model runtimes start to diverge from LP scaling around 150 time steps, after which the runtimes rapidly escalate. As such, a MIP model is only suitable for short horizons or large time deltas, making it harder to anticipate future need for state-of-charge.

### 3.2.2 Baseline: Seasonal naïve policy

The simplest solution to the VPP scheduling problem (3.4) when we do not have access to the baseline load signals, is to keep a fixed time-based schedule. For example, we could fix the battery schedule to charge around solar noon to capture solar generation and discharge during the evening peak tariff period. A more realistic approach taking into account the regular shadowing of PV panels and capturing the average usage profile of the consumer, is to use the observations from the previous day(s) to produce a tailored battery schedule.

**Seasonal tariff constraints** In this baseline model, we optimise a battery control decision against the expected value of that decision, where the expectation is taken over the previous several days. By considering multiple days, we are able to average over the typical behaviour of the occupants. Therefore, the battery constraints (3.2) remain unchanged, but the tariff constraints are duplicated across a set of scenarios $j$, resulting in

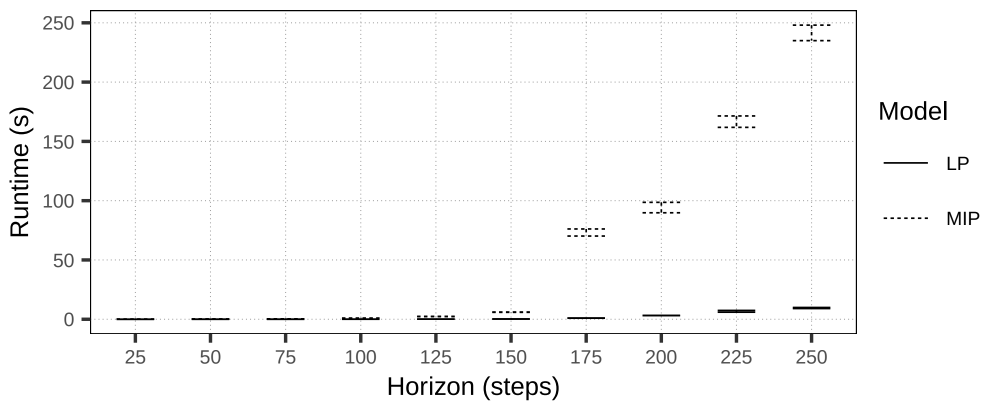


**Figure 3.2:** Performance of linearised model versus a model with integer decisions to allow arbitrary tariffs. Error bars show min/max runtime across five trials.

one cost per previous day $D_j$, where we use the actual observed base load $b_{i,t}^{D_j}$ on that day.

$$\begin{aligned} c_{i,t}^{D_j} &\geq c_t^{\mathsf{IN}} \cdot (b_{i,t}^{D_j} + p_{i,t}^b) \cdot \Delta & \forall i, j, t \\ c_{i,t}^{D_j} &\geq c_t^{\mathsf{EX}} \cdot (b_{i,t}^{D_j} + p_{i,t}^b) \cdot \Delta & \forall i, j, t \\ c_i^{D_j} &\geq \textstyle\sum_{t=1}^h \big(c_{i,t}^{D_j}\big) & \forall i, j \end{aligned} \tag{3.5}$$

**Seasonal naïve optimisation** In optimising for the historic scenarios, we take the expected value (the average) over the total energy costs that the VPP incurs. This ensures that the resulting battery schedule will try to balance for the average day. For the peak load objective, we pick the worst-case scenario, setting the peak load to the maximum observed over all days. This aims to make the policy robust to all periods where peak load could occur, while making the policy insensitive to below-average days (e.g. holidays). The resulting model is:

$$\begin{aligned} \text{minimise } & \hat{c}p^P + \mathbb{E}_j \Big[\textstyle\sum_{i=1}^n \big(c_i^{D_j}\big)\Big] \\ \text{subject to } & p^P \geq \textstyle\sum_{i=1}^n \big(b_{i,t}^{D_j} + p_{i,t}^b\big) \qquad \forall j, t \\ & p^P \geq 0 \\ & \text{Battery constraints (3.2)} \\ & \text{Seasonal tariff constraints (3.5)} \end{aligned} \tag{3.6}$$

Compared to the hindsight model, this model adds another factor of complexity (the number of previous days/scenarios). Thus, while it is still a polynomial-time model, the constant factor will likely make this model significantly more costly to solve. Furthermore, the use of only historic data means that the solution cannot anticipate big weather shifts (e.g. overcast day following a week of sunny weather).

## 3.3 Evaluation of potential savings and scalability

Using the optimal hindsight policy and seasonal naïve baseline, we are able to estimate the best-case energy cost savings. We calculate the regret (Eq. 3.1) of the naïve baseline on a set of 11 test days per month of the year, by evaluating the actual cost that the solution to (3.6) obtains on a given day. Figure 3.3 shows how the month of the year affects the value of better forecasting. We present the results using box-plots, indicating the 25th through 75th quantile of the outcomes with the box, and the 10th through 90th quantile with the line (outliers marked with black points, median the centerline in each box, and mean as the red point). We observe

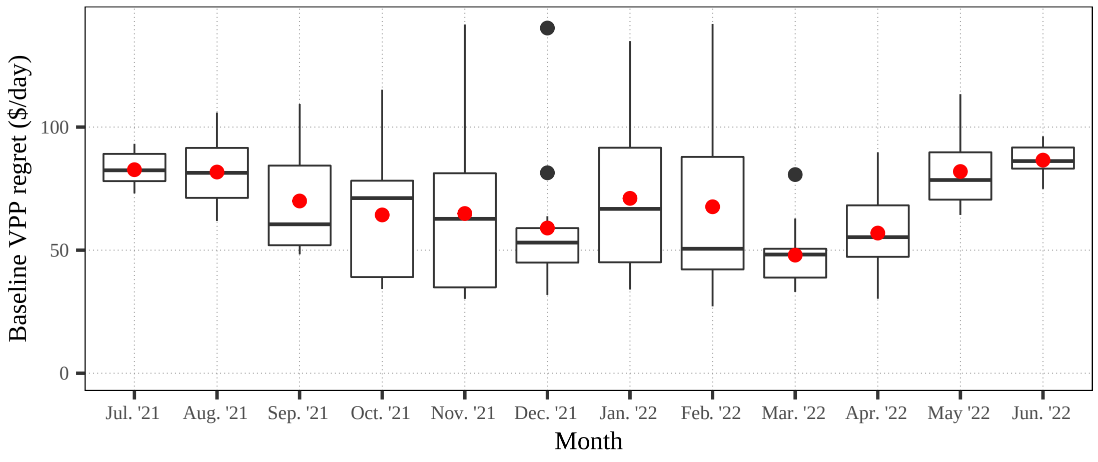


**Figure 3.3:** Regret of the seasonal naïve policy by month of the year.

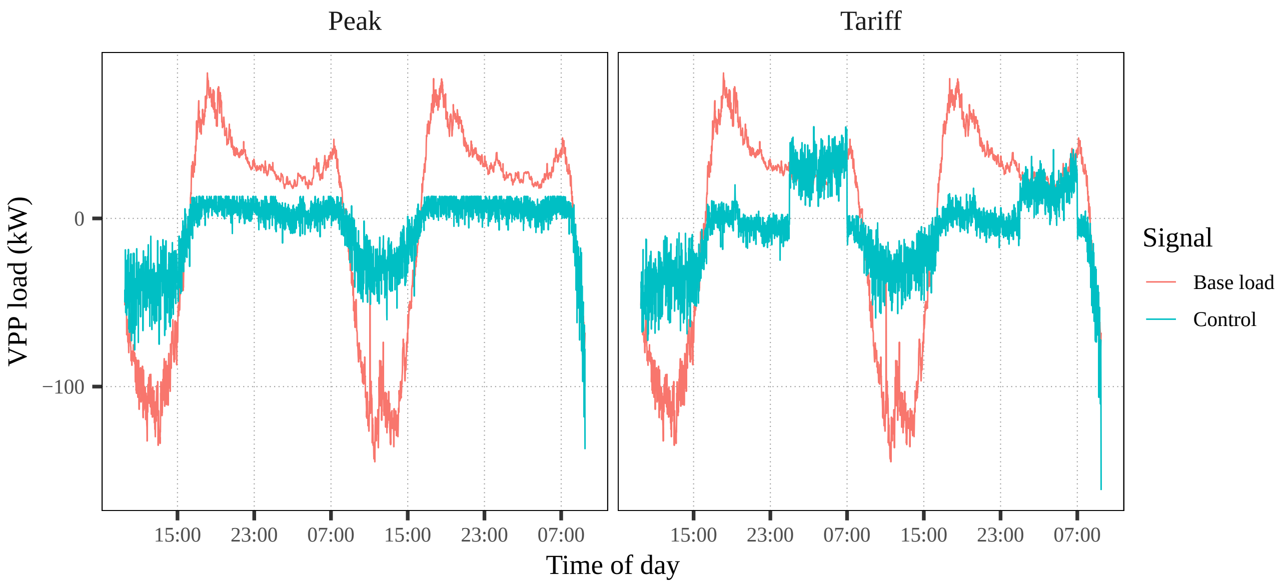


**Figure 3.4:** VPP load trajectory depending on the peak load price.

that the value of ideal forecasts is higher in winter months on average, when the solar potential is smaller. On the other hand, the summer months are more volatile (a rare cloudy day invalidates a solution on the previous sunny days). On average, the regret is $72.6 *per day* for a VPP consisting of 50 devices operating under South Australia tariffs (0.15 $/kWh export, 0.45 $/kWh import with two off-peak periods, a night-time setback of 0.17 $/kWh from 01:00--06:00 and a solar peak setback of 0.20 $/kWh from 10:00--15:00).

Looking more closely at the behaviour of the VPP under different control parameters, Figure 3.4 shows how the price of the peak load affects the control decisions of the VPP. If the demand charge is high, the overnight off-peak tariff will not be used to charge the batteries for the morning peak.

A key concern is the time to find an optimal solution. In typical operation, we will be replanning the schedules whenever a new, more accurate forecast comes in. Furthermore, the cost of commercial solvers is prohibitive, requiring an evaluation of suitable open-source solvers. We perform an extensive evaluation of three different LP solvers, Gurobi (Gurobi Optimisation, 2022), HiGHS (Huangfu and Hall, 2018) and SoPlex (Gamrath et al.,

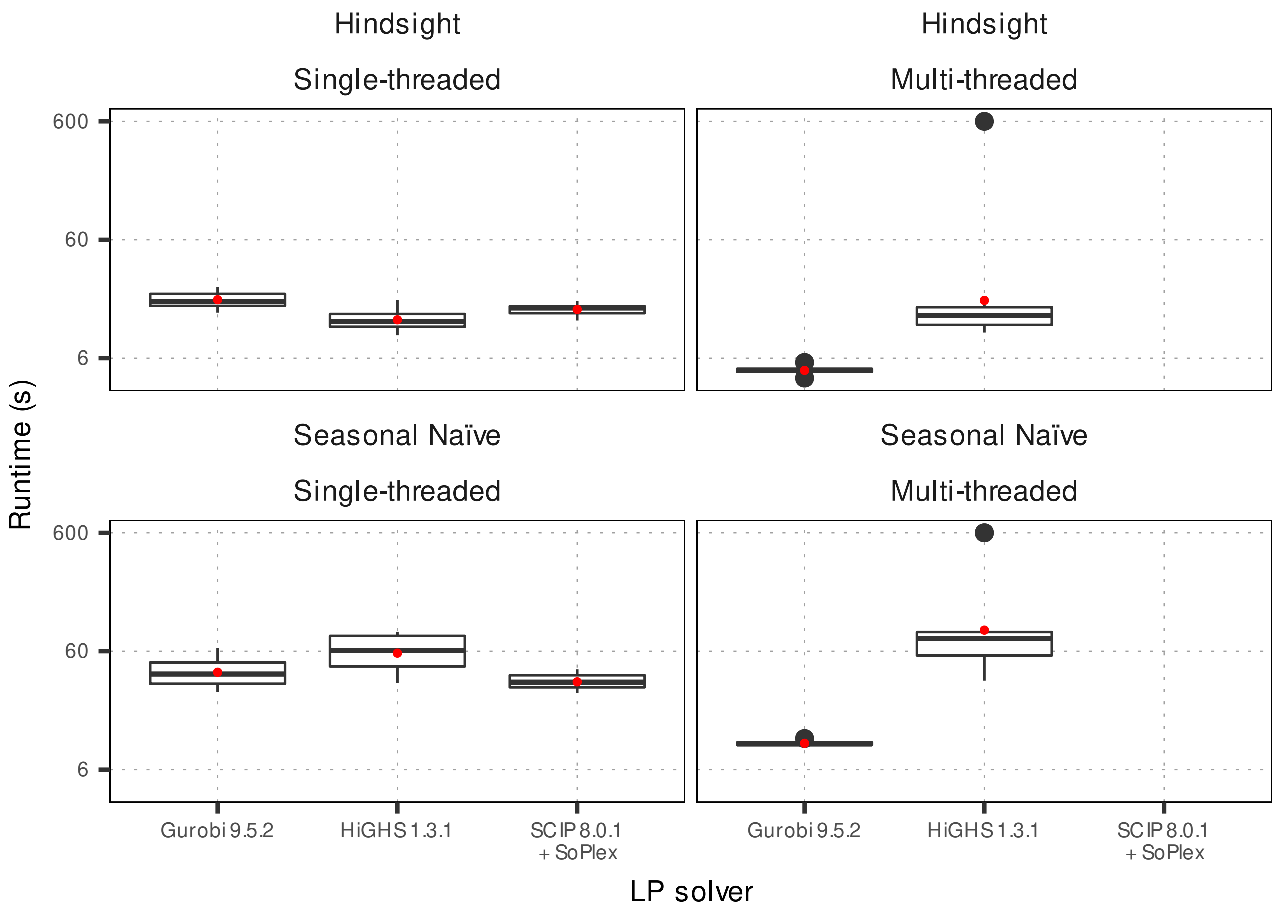


**Figure 3.5:** Runtime comparison of commercial (Gurobi) and open-source (HiGHS, SoPlex) linear programming solvers, across both optimisation models. Note the log-$y$ scale, and that SoPlex has no multi-threading mode; runs timed out after 600 seconds (10 minutes), usually a sign of numerical difficulties).

2020), on the MPS model output by MiniZinc (Nethercote et al., 2007), with the runtime statistics shown in Figure 3.5. All runs were performed on a 16-core consumer-grade Linux desktop machine. Unfortunately, only the commercial solver Gurobi clearly benefits from multi-threading; other solvers either do not support or suffer from timeouts when multi-threading is used. Nevertheless, currently runtime costs of the optimal model are reasonable (around one minute), but this would change if we extend the horizon to multiple days or increase the number of units beyond 50, at which point real-time minute-by-minute optimisation would be out of reach. This suggests an important role for efficiency improvements, especially since we also need to allow for time for calculating forecasts.

# 4 Predict-plus-optimise strategies

Using the forecast models developed in Chapter 2 and the optimisation algorithms presented in Chapter 3, we are now in a position to discuss solutions that combine both aspects in a predict-plus-optimise solution. We have developed strategies to tackle the predict-plus-optimise problem, ranging from fully decoupled strategy (model-predictive control, Section 4.2), to integrated solutions that directly use the optimisation objective to improve a predictive model, namely structured prediction with dynamic programming (Section 4.3).

## 4.1 Background: column generation and dynamic programming

To be able to deploy a predict-plus-optimise strategy on individual units, we need to decouple the centralised optimisation problem (3.4) such that we can manage each unit in isolation. We can make use of the fact that the optimisation problem is loosely coupled (has few constraints binding the battery scheduling problems together) to gain efficiency. In the case of our hindsight model (3.4), only the VPP peak load constraint is shared across the devices. We reformulate this problem as a column generation problem where the peak load constraints are the only constraints, and the `columns' represent individual units' battery schedules.

Formally, let $i$ index over the $n$ units in the VPP, and let $Z_i$ be the set of all possible battery schedules of unit $i$ (where *possible* means technically feasible within the battery constraints). Each individual battery schedule $\pi_k \in Z_i$ has an associated value under the tariff structure and the unit's base load and generation time series, value $V_{\pi_k}$. Furthermore, it has an associated net power consumption $C_{\pi_k,t}$ over each time step. Here we use capital letters for $V$ and $C$ to indicate that these quantities are technically random variables with an unknown true value because they are dependent on the future realisations of the actual base load and solar generation. With these quantities defined, we could equally solve the following schedule mixing problem to solve our hindsight model (3.4):

$$\begin{aligned} \text{minimise } & \hat{c}p^P + \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k} V_{\pi_k} \\ \text{subject to } & p^P \geq \sum_{i=1}^{n} \sum_{\pi_k \in Z_i} x_{i,k}\, C_{\pi_k,t} && \forall t, \\ & \sum_{\pi_k \in Z_i} x_{i,k} = 1, && \forall i, \\ & p^P \geq 0,\ 0 \leq x_{i,k} \leq 1, && \forall i,k. \end{aligned} \tag{4.1}$$

This model assigns a mixing probability $x_{i,k}$ to each possible policy, such that the combination jointly minimises the peak load tariff, in expectation.

The obvious downside to this approach is that the set of possible schedules $Z_i$ is enormous, too large to enumerate directly. However, we can efficiently generate individual battery schedules that are member of $Z_i$ (for example by solving a reduced model containing only the battery constraints of unit $i$), and we can do so in parallel for each unit $i$. This is especially useful in the context of SwitchDin's application, where each unit has its own local computing power, allowing for the distribution of workload.

We apply a Lagrangian relaxation to the shared peak load constraints, to construct an objective for the single-agent decision-making problem that can be implemented directly in an optimisation routine (de Nijs and Stuckey, 2020). We make use of a discretisation of the battery scheduling problem to implement the scheduling problem extremely efficiently.

**Dynamic program** The battery scheduling problem for a single unit can be expressed as a discrete dynamic programming problem. We discretise the state space of the battery by selecting a minimum state-of-charge change delta $\delta_s$ in kWh. The action space is equally discretised into steps of multiples of $\delta_s$.

$$\begin{aligned} S &= \left[\underline{s}_i, \underline{s}_i + \delta_s, \ldots, \overline{s}_i - \delta_s, \overline{s}_i\right] \\ A &= \left[\frac{\Delta \underline{p}_i}{\delta_s}, \frac{\Delta \underline{p}_i}{\delta_s} + 1, \ldots, 0, \ldots, \frac{\Delta \overline{p}_i}{\delta_s}\right] \\ T(s,a) &= \textsf{CLIP}\left(\underline{s}_i,\ s + a,\ \overline{s}_i\right) \end{aligned} \tag{4.2}$$

The transition function $T(s' \mid s, a)$ maps how many $\delta_s$ steps the control action $a$ shifts the state space, clipped against the total capacity constraints of the battery. In a typical Markov Decision Process (Bellman, 1957), the transition function is a stochastic function, but in our case we assume the behaviour for the battery to be deterministic. The reward function $R_t(s, a)$ captures how the control decisions affect the battery tariff, and the peak load through the dual costs $\lambda_t$. In our case, the reward function is given as:

$$\begin{aligned} \delta_t^{s,a} &= \delta_s \cdot (T(s,a) - s), \\ \bar{\delta}_t^{s,a} &= \max\left(\eta\, \delta_t^{s,a}, {}^1\!/\!_\eta\, \delta_t^{s,a}\right) + b_{i,t}^L \Delta, \\ R_t(s,a) &= \max\left(c_t^{\text{IN}} \bar{\delta}_t^{s,a},\ c_t^{\text{EX}} \bar{\delta}_t^{s,a}\right) + \frac{\bar{\delta}_t^{s,a} \lambda_t}{\Delta}, \end{aligned} \tag{4.3}$$

where $\delta_t^{s,a}$ is the actual energy transmitted into or out of the battery, and $\bar{\delta}_t^{s,a}$ the actual energy consumed or generated at the unit (after efficiency losses and including the base load $b_{i,t}^L$). The reward function itself has two terms, the instantaneous energy cost calculated through the tariff, and the contribution to the peak load valued through the Lagrangian dual costs $\lambda_t$.

The dynamic program optimises the value of a decision at every decision point (cross product of time step $t$ with state $s$), through a recursive relationship called the Bellman equation. The value function at a given point is defined as:

$$\begin{aligned} V_t[s] &= \max_{a \in A}\big(R_t(s,a) + V_{t+1}[T(s,a)]\big) \quad && \forall s, t < h \\ V_h[s] &= 0 && \forall s \end{aligned} \tag{4.4}$$

By calculating the value V for all states backwards across time (from $t = h - 1$ down to $t = 0$), we can calculate the optimal value function in a single linear pass, with the complexity of each step depending on the size of the state and action space, $|S|$ and $|A|$.

**Evaluation** The dynamic program is executed every time a new column is required, which is for every new set of dual costs calculated. We expect the dynamic program to do this in comparable or better time to the single-agent linear program, because it is a problem-specific solution. However, the quality of the solution can be traded off with the runtime, depending on the fundamental decision of the discretisation factor $\delta_s$. Figure 4.1 explores how a Python implementation of dynamic programming compares with the commercial LP solver Gurobi on the single-agent battery scheduling problem. We observe that the runtime strongly depends on the discretisation factor (where an increasing factor results in more states and actions); at $4\times$ the dynamic program is about ten times as fast as the linear program, with a solution that is nearly identical down to tie breaking differences, suggesting that the dynamic program is a suitable choice for optimising the single-agent problem.

## 4.2 Model-predictive control

One of the fundamental properties of any forecasting system is that forecasting accuracy tends to be highest in the near term. The forecasting models developed in Section 2.5 show a similar trend, where forecasts for the first hour incur a smaller error than the forecasts for the first three hours, and continuing to hold true up to the forecast horizon of two days. Because of the computational efficiency of the LightGBM forecasting models and the optimisation, we are able to recalculate a battery schedule for an updated forecast in real-time, potentially adjusting the battery control from what it would have been under a fixed schedule. By calculating a battery schedule for the entire two days, but only implementing the first decision we are implicitly implementing a model-predictive control strategy.

Figure 4.2 presents the effect of model-predictive control (MPC) on the battery schedule and its error gap when compared to the optimal schedule. We observe that MPC improves on both tariff and peak metrics, with the largest gain in peak kW reduction. This is a direct consequence of the improved near-term forecast, as peak kW is an instant measurement that is directly controlled by the first decision implemented. By taking this decision based on the most accurate near-term forecast, we are able to reduce the likelihood of missed load spikes influencing the control decision.

## 4.3 Structured prediction with dynamic programming as a loss

The model-predictive control strategy discussed in the previous section uses the forecasts as-is, optimised for the standard error metrics. In this section, we present how the dynamic program of Section 4.1 can be used to inform the gradients of the LightGBM update step directly. We achieve this by applying a convex regularisation to the non-linear max operator of the Bellman equation (4.4).

Recall that the state space $S$ of the battery scheduling problem captures the state-of-charge. Supposing a 13 kWh capacity battery, its charge in kWh is split up into $|S|$ steps, e.g.,

$$S = \{0, 0.1, 0.2, \dots 12.9, 13.0\}. \tag{4.5}$$

Action space $A$ captures the internal power gain of the battery in kWh under a decision to run at $a$ kW. For example, if our battery has a maximum rate of 2 kW, its actions could be split up into $|A|$ steps, e.g.:

$$A = \{-2, -1, 0, 1, 2\}. \tag{4.6}$$

Assuming a time delta of $\Delta = 6$ minutes, the state of charge impact of these decisions is

$$\delta s_a = \frac{\Delta}{60} a = \frac{a}{10} \tag{4.7}$$

Then the transition function is given as $T(s, a) \rightarrow s'$, where the next state depends on the current state of charge plus action,

$$T(s, a) = s + \delta s_a, \tag{4.8}$$

for example,

$$T(0.3, 2) = 0.3 + \frac{2}{10} = 0.5. \tag{4.9}$$

The transition function is undefined for actions that would over/undercharge the battery.

The reward function $R(a) \rightarrow \mathbb{R}$ is given by the tariff cost of the decision implemented, which in turn depends on the total behind-the-meter load of the unit. Assuming given a baseload of $y_t$ kW, and a battery efficiency rating $\mu$, the total load $L_t(a)$ is a function of the action,

$$\begin{aligned} a_\mu &= \max(\mu a, \frac{a}{\mu}) \\ L_t(a) &= y_t + a_\mu \end{aligned} \tag{4.10}$$

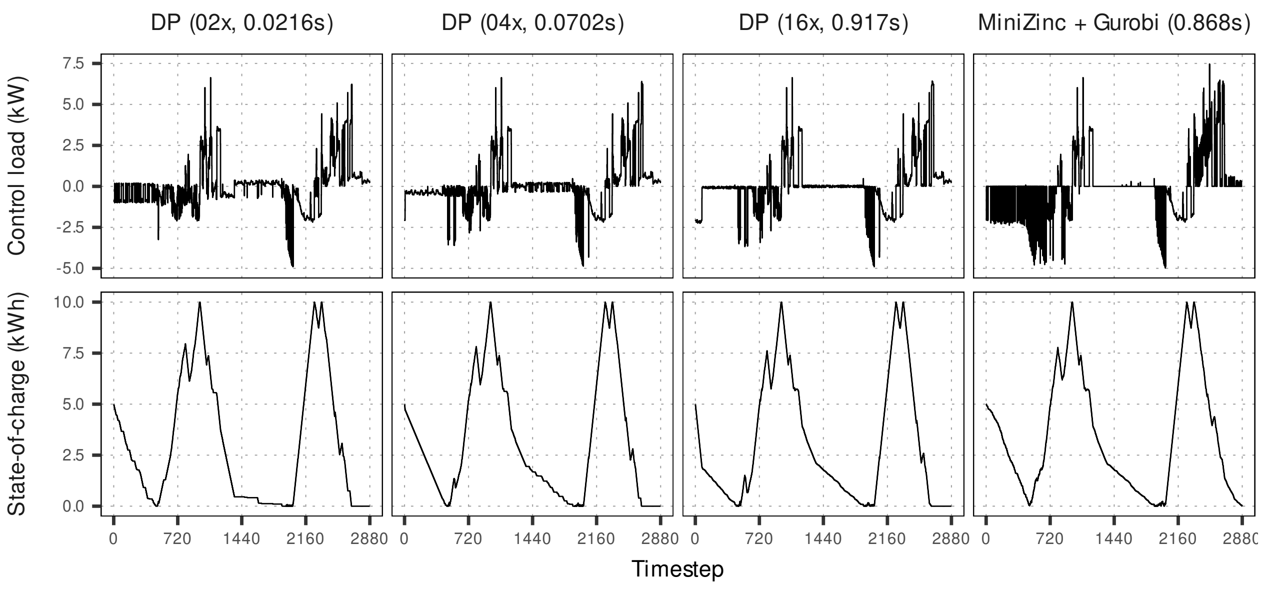


**Figure 4.1:** Effect of discretisation level on battery scheduling policy. Battery schedules are calculated for a period of two days and compared against the continuous optimal schedule (rightmost column).

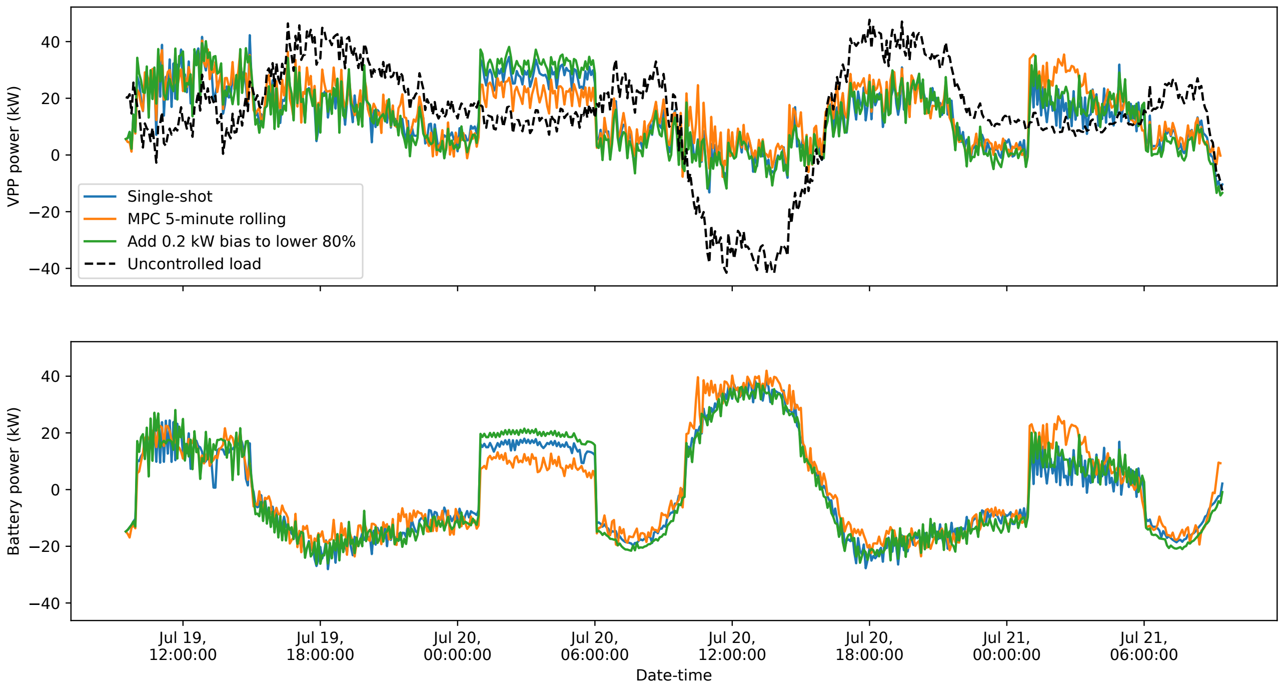


**Figure 4.2:** Model-predictive control solution (orange line) versus a single-shot schedule for two days.

**Table 4.1:** Performance metrics of the different schedules in Figure 4.2

| | Forecast | | |
|---|---|---|---|
| Error metric | Normal | Biased | MPC |
| Tariff gap ($) | 39.013 | 36.258 | 38.982 |
| Peak gap (kW) | 11.659 | 9.559 | 9.781 |

Furthermore, we assume given a two-sided, time-specific export- and import tariff; respectively $c_t^{\text{ex}}$ and $c_t^{\text{im}}$ (in negative dollars, e.g. $c_t^{\text{im}} = -0.40$\$/kW). The reward function then becomes

$$R_t(a) = c_t^{\text{ex}} L_t(a) \mathbf{1}_{x<0}(L_t(a)) + c_t^{\text{im}} L_t(a) \mathbf{1}_{x>0}(L_t(a)) \tag{4.11}$$

using the indicator function $\mathbf{1}(x)$ to choose which tariff applies. In practice, $b_t$ has to be forecast; suppose we have access to load forecasting function $f_t(x_t)$ for input features $x_t$. Then,

$$R_t(a) = \begin{cases} c_t^{\text{ex}} f_t(x_t) + c_t^{\text{ex}} a_\mu & L_t(a) < 0 \\ c_t^{\text{im}} f_t(x_t) + c_t^{\text{im}} a_\mu & L_t(a) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

For a given value of $y_t' = f_t(x_t)$, the action determines which case we are in. Note that *at most* a single action will fall into the $R(a) = 0$ category.

Given these functions, we can calculate the optimal control policy of the battery using the Bellman equation:

$$\begin{aligned} Q_t^*[s,a] &= R_t(a) + \sum_{s'} T(s,a,s') V_{t+1}^*[s'] \\ V_t^*[s] &= \max_a Q_t^*[s,a] \qquad\qquad \forall s, a \\ V_h^*[s] &= 0 \\ \pi_t^*(s) &= \arg\max_a Q_t^*[s,a] \end{aligned} \tag{4.13}$$

Here the transition matrix $T(s,a,s')$ is 1 if $s' = s + \delta s_a$, $-\infty$ if $s + \delta s_a < 0$ or $s + \delta s_a > \max S$, and 0 otherwise. As such, each *valid* action uniquely determines which $V_{t+1}^*[s + \delta s_a]$ to pick. There will always be at least one valid action (i.e., hold charge), meaning that the invalid actions will never be chosen.

## Fitting the forecast function

In a classic predict-then-optimise setting, we first fit the forecast function to minimize the forecast error in the base load, for example absolute $|f_t(x_t) - y_t|$ or squared $(f_t(x_t) - y_t)^2$ error. However, Mensch and Blondel (2018) show how to use a dynamic program such as the one defined in (4.13) as the final layer in a prediction system, allowing us to fit the forecast function against the optimisation objective directly. To achieve this, we smooth out the decision-making dynamic program through use of the soft maximum operator log-sum-exp,

$$\begin{aligned} Q_t[s,a] &= R_t(a) + \sum_{s'} T(s,a,s') V_{t+1}[s'] \\ V_t[s] &= \log \sum_a \exp\big(Q_t^*[s,a]\big) \\ \pi_t(s,a) &= \frac{\exp(Q_t[s,a])}{\sum_a \exp(Q_t[s,a])} \end{aligned} \tag{4.14}$$

As a result of this transformation, the policy is now stochastic, with actions selected with probability $\pi_t(s,a)$. Note that because $\exp(-\infty) = 0$, invalid actions that would over-/undercharge the battery will not be chosen by the soft policy either.

Under this transformation, we have a choice of loss function. We could either target the value function $V$, or the policy $\pi$ directly. If targeting the value function, the error is with respect to the optimal (soft) value function $V_t[s]$

calculated for $y_t$ against the predicted value function $V'_t[s]$ calculated for $y'_t$, i.e.

$$\begin{aligned}\mathcal{L}(y'_t) &= (V'_t[s] - V_t[s])^2 \\ &= (V'_t[s] - v_{t,s})^2 \\ &= \left(\text{log}\sum_a \text{exp}\Big(R_t(a) + \text{log}\sum_{a'} \text{exp}\big(Q'_{t+1}[s + \delta s_a, a']\big)\Big) - v_{t,s}\right)^2\end{aligned} \tag{4.15}$$

Other error distances than the square norm $||.||^2$ could be considered, such as absolute value or Huber loss. Note that the error in prediction at time $t$ is now a function of all predictions $[y_t, y_{t+1}, \dots, y_{h-1}]$ from $t$ up to the horizon $h$, capturing the fact that early prediction errors will have a knock-on effect on the state-of-charge and thus future decisions; something which is not captured by the usual loss.

The other loss function targets the policy directly, through the Kullback-Leibler divergence between the optimal soft policy $\pi_t(s, a)$ and the soft policy for the forecast $\pi'_t(s, a)$, minimising $D_{KL}(\pi_t(s, a) \,||\, \pi'_t(s, a))$,

$$\begin{aligned}\mathcal{L}(y'_t) &= -\sum_{a\in A}\big(\pi_t(s, a)\,\text{log}(\pi'_t(s, a))\big) \\ &= -\sum_{a\in A}\left(\pi_{t,s,a}\,\text{log}\left(\frac{\text{exp}(Q'_t[s, a])}{\sum_a \text{exp}(Q'_t[s, a])}\right)\right) \\ &= -\sum_{a\in A}\left(\pi_{t,s,a}\,\text{log}\left(\frac{R_t(a) + \text{log}\sum_{a'} \text{exp}\big(Q'_{t+1}[s + \delta s_a, a']\big)}{\sum_a \text{exp}(Q'_t[s, a])}\right)\right)\end{aligned} \tag{4.16}$$

### Evaluation

We evaluate the dynamic programming value loss on a small example to demonstrate the best-case effect of using the optimisation as a loss for a forecasting algorithm. To this end, we implemented the dynamic programming loss on two machine learning frameworks, PyTorch neural networks and LIGHTGBM gradient-boosted decision trees.

Our example consists of a single load shape to be forecast, under two different features:

- *Unique* features uniquely identify the actual time step ($h = 24$ classes),
- *Tariff* features uniquely identify the current import tariff (3 classes).

By having a single load shape and unique features, we expect any forecasting model to (eventually) fit the load curve exactly. On the other hand, tariff features strongly correlate with the battery scheduling decision, meaning that it should be possible to fit predictions that result in good policy decisions even if they are mismatched with the actual load.

We evaluate all combinations of algorithm (PyTorch, LIGHTGBM), feature (Unique, Tariff) and loss function: mean absolute error (MAE; Equation 2.1), and optimisation value loss (Equation 4.15) on the small example. We expect the following observations:

1. Tariff features are harder to fit than unique features, due to ambiguity of many time steps sharing the same feature. This implies that final loss on tariff features will be higher than on unique features.
2. The targeted loss will have a lower error than the other metric (e.g., targeting value loss will result in comparatively lower value loss than targeting MAE with the same algorithm), because the algorithm explicitly reduces the targeted loss.

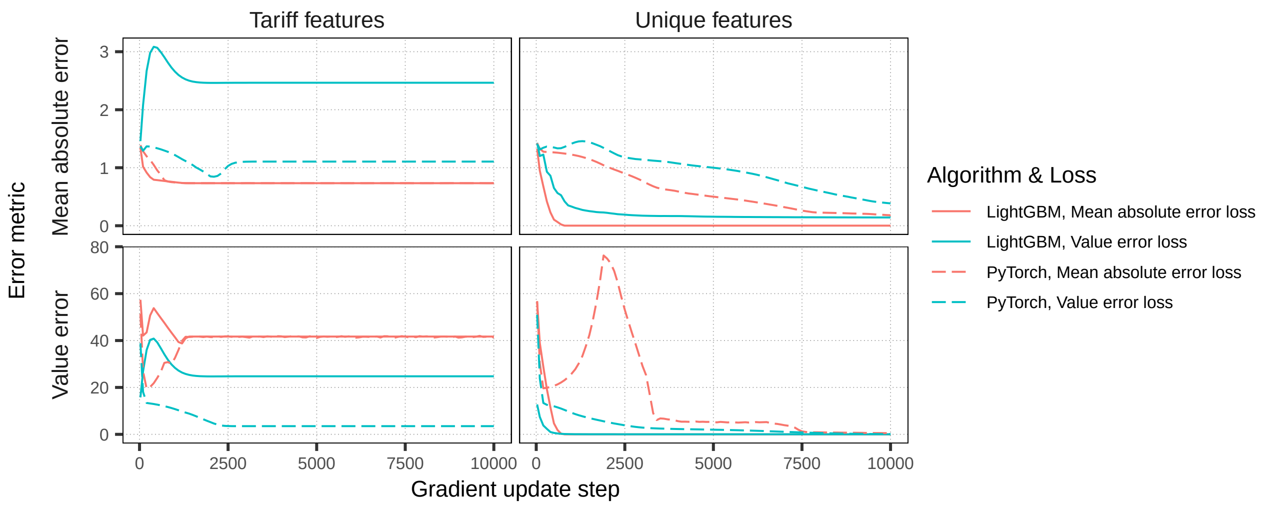


**Figure 4.3:** Training curves measuring how the prediction errors reduce over time as a forecast model is made to fit to the data. We measure two different errors (rows) for every run (a combination of loss function, tariff and algorithm). The mean absolute error is a proxy for the optimisation value error, however, targeting it generally results in higher value error compared to targeting value error as a loss directly.

Figure 4.3 presents the evolution of prediction errors over training time, for all eight combinations. Every combination produces two trend lines, the mean absolute error trend, and the value error trend. Because the errors have a different magnitude, it is not meaningful to compare them numerically; only the relative differences are important. Therefore, we present the error trends in separate panels (the plot rows). Thus, each combination appears in both rows (e.g., both solid red lines in the tariff features column correspond to the same run, using LightGBM to reduce MAE on Tariff features).

Generally, we expect the loss (the targeted error) to trend downwards over time, up to a convergence limit beyond which error cannot be reduced further. Thus, we expect the red lines (optimising for MAE loss) to decrease the MAE error (trends in top row) over time, but not *necessarily* the value error (corresponding red lines in bottom row). A striking example of this is the PyTorch implementation; it continually reduces the MAE loss on unique features, while the value loss is inversely correlated for a time. Value errors increase until around gradient step 2000, beyond which both errors decrease together.

Our first hypothesis (tariff features are harder to learn than unique features) says that for any algorithm and loss function, the final error will be lower for unique features. We observe from the plot that this is indeed the case. Our second hypothesis, that the error will be lower if it is targeted as a loss suggests that red lines will fall under blue lines in the top row (MAE loss results in lower MAE error), while blue lines fall under red lines in the bottom row. This holds for tariff features, and for the PyTorch implementation in general, but does not hold for the LightGBM implementation on unique features, where it is able to reduce both MAE and value error to virtually zero under MAE loss (solid red). We hypothesise that this is an outlier result due to the nature of decision trees, which are (in the limit) able to assign unique predictors to each class (unique feature), as opposed to neural networks which compute non-linear functions of all features as input. By having a unique feature for each time step, the decision tree can perfectly match the data. Figure 4.4 presents the forecast fits, showing that this is indeed the case (top right).

Looking at Figure 4.4 in more detail, we observe that under unique features, the fits for MAE and value loss (points) conform to the ground truth (dashed line) in general, reinforcing that in an ideal case with low forecast

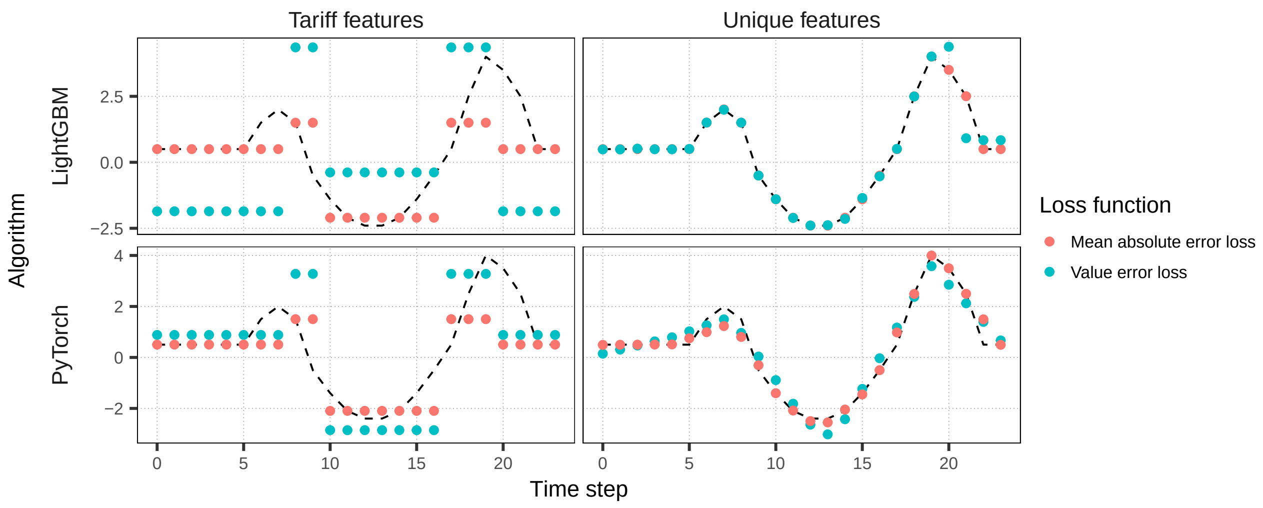


**Figure 4.4:** Fit at the end of 10,000 training steps, for the two feature sets and algorithms, comparing the forecast quality of targeting either loss function. Under `unique' features, both algorithms end up fitting to the ground truth on either loss function, as the features uniquely identify each timestep. However, under `tariff' features, the value error loss exaggerates the predicted load curve to obtain a better policy.

errors, the loss functions agree. However, when features are sparse or uninformative such as the tariff case, the forecast models disagree. The MAE loss results in the predictions fitting to the median of the class, as expected. In the case of the policy loss, the predictions are exaggerated, in some cases even beyond the range of the true values, in order to improve the resulting battery schedule's objective.

## 4.4 Conclusions and future work

One way to achieve better overall VPP performance from forecasts is to apply a rolling horizon optimisation, continually refreshing the forecast and battery schedule and making use of the higher forecast accuracy in the near term. However, doing so requires an efficient optimisation scheme capable of running in real-time for many units. We have developed a dynamic programming solution to the battery scheduling solution capable of running an order of magnitude faster than an equivalent LP solution. Together with a distributable computation column generation scheme, this solution can be used to employ a rolling model-predictive control scheme to control the VPP with the forecasts as prediction.

However, even under a model-predictive control, a simple biasing experiment shows that better control outcomes can be obtained with a less accurate forecast when measured by traditional error measures. To demonstrate a possible approach to obtain more aligned forecasts, we show an application of structured prediction with the dynamic program. By constructing a differentiable loss function from a softened stochastic policy version of the dynamic program, we show we can train forecasts with lower schedule error than an equivalent mean-absolute error policy.

Nevertheless, there is significant scope for future work. Currently the battery scheduling problem only considers the battery capacity limits as hard constraints; it would be straightforward to extend the model to capture additional requirements. Examples include:

- dynamic operating envelopes, which aim to prevent voltage excursions beyond the feasible range by imposing power import and export limits. An open question is if doing so is required during training time in predict+optimise. Ideally, these constraints would not be imposed during training, because the dynamic envelopes depend on the state of the local grid, something which will change over time as upgrades are brought in.

- battery degradation prevention, which are constraints to attempt to reduce early degradation of batteries from excessive cycling. Examples of this include limiting the total cumulative energy in and out to at most one full cycle per day, or introducing a maximum change in power between time steps, ensuring control is smooth instead of min/max.

# 5 User documentation

This chapter discusses the practical considerations around setting up and running the open source implementations provided by our software package. Our software package contains the original dataset, an implementation of the forecasting models and the centralised optimisation solutions (GitHub repository).

## 5.1 Residential power and battery data

### 5.1.1 Overview

The Residential Power and Battery data is an open-source dataset designed to facilitate the advancement of predictive and optimisation algorithms. It features anonymised, minute-by-minute real-world customer data on energy consumption, solar generation, and battery measurements. This dataset was compiled by SwitchDin and made available through Monash University on the https://zenodo.org/.

Open-sourcing the Residential Power and Battery data offers numerous benefits to researchers, developers, and industry stakeholders. By providing access to comprehensive, real-world data on energy consumption, solar generation, and battery usage, the dataset enables the development of more accurate and efficient algorithms for energy management systems. For example, the dataset could facilitate the development of machine learning models that forecast energy consumption patterns, enabling better demand-side management strategies. These improved algorithms contribute to more effective demand response, grid stability, and renewable energy integration, helping to build a more resilient and sustainable energy future.

Furthermore, open-sourcing this dataset fosters collaboration and knowledge sharing among researchers and professionals in the energy sector. By making the data freely available, researchers from various backgrounds and organisations can work together to identify patterns, trends, and innovative solutions to pressing challenges in energy management. This collaborative approach accelerates the pace of innovation, as diverse perspectives can generate novel ideas and methods that might not emerge in isolation. As a result, the open-sourcing of the Residential Power and Battery data has the potential to significantly advance the pursuit of a more efficient, reliable, and environmentally friendly energy landscape.

### 5.1.2 Dataset structure

The dataset is structured as follows:

**anonymous_public_power_data.rds**

- utc - The date-time in UTC, formatted as yyyy-mm-dd hh:mm:ss.
- unit - A categorical label denoting the unique identifier for the unit.
- metric - A categorical label indicating whether the data point corresponds to load or solar power generation.
- max - A numerical variable denoting the peak value of load or solar power generation in kilowatts (kW) within a one-minute interval.
- min - A numerical variable denoting the minimum value of load or solar power generation in kilowatts (kW) within a one-minute interval.

- sum - A numerical variable denoting the sum of values of load or solar power generation in kilowatts (kW) within a one-minute interval.
- count - A numerical variable denoting the number of values of load or solar power generation in kilowatts (kW) within a one-minute interval.

**anonymous_public_battery_data.rds**

- unit - A categorical label denoting the unique identifier for the unit.
- batt_kwh - A numerical variable representing battery kilowatt-hour rating.
- batt_p_ch - A numerical variable representing battery charge power rating.
- batt_p_dch - A numerical variable representing battery discharge power rating.

#### 5.1.3 Usage and licensing

Residential Power and Battery Data is released under the Creative Commons Attribution-NonCommercial 3.0 Unported (CC BY-NC 3.0) license, which allows for free use, distribution, and modification of the dataset, provided appropriate credit is given to the original authors.

## 5.2 Forecast package

For detailed instructions on setting up and using the project refer to the GitHub repository.

It contains step-by-step guidance on the setup and installation process, running the project, input data requirements, modelling, and prediction outputs. By following the instructions provided in the repository, you can navigate and use the project's features for development and testing purposes.

The repository is designed to process and clean raw power, train separate models for load power and PV power forecasting using the LightGBM algorithm, and generate multi-step ahead forecasts for both load power and PV power. The entire workflow is performed by executing the main.R script, which sources necessary functions from other R scripts, covering data preprocessing and cleaning to model training and prediction.

## 5.3 Optimisation package

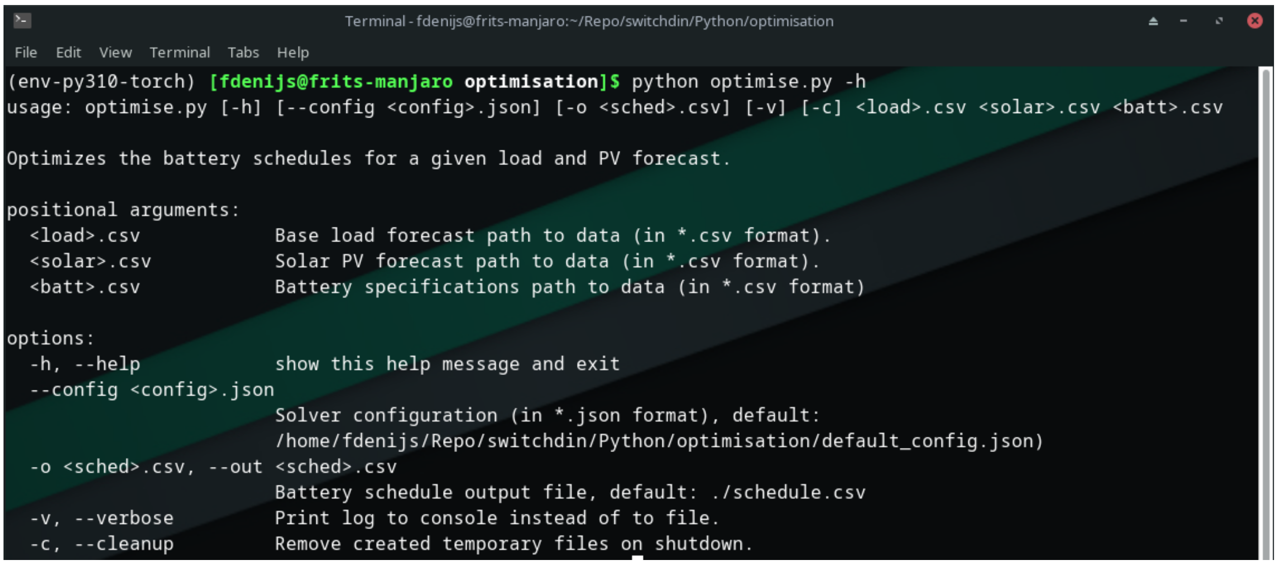
The optimisation code is provided as a Python implementation, with dependencies on MiniZinc and one or more linear programming solvers (such as Gurobi, COIN-BC or HiGHS). The optimisation has a stand-alone command-line interface to be able to run with any forecast, whether produced by our package or externally.

**Installation** First, install the software dependencies from their respective web pages, using the latest versions for MiniZinc and the solver. For Python, we have developed and tested the code with version 3.10. As is common for Python code, some additional Python packages will need to be installed. For this purpose we recommend creating a virtual environment (venv) to host the packages without conflict with other uses of Python on the system.

On Windows:

```
> py -3.10 -m venv predopt
> predopt\Scripts\activate
```

On Linux:



```
(env-py310-torch) [fdenijs@frits-manjaro optimisation]$ python optimise.py -h
usage: optimise.py [-h] [--config <config>.json] [-o <sched>.csv] [-v] [-c] <load>.csv <solar>.csv <batt>.csv

Optimizes the battery schedules for a given load and PV forecast.

positional arguments:
  <load>.csv            Base load forecast path to data (in *.csv format).
  <solar>.csv           Solar PV forecast path to data (in *.csv format).
  <batt>.csv            Battery specifications path to data (in *.csv format)

options:
  -h, --help            show this help message and exit
  --config <config>.json
                        Solver configuration (in *.json format), default:
                        /home/fdenijs/Repo/switchdin/Python/optimisation/default_config.json)
  -o <sched>.csv, --out <sched>.csv
                        Battery schedule output file, default: ./schedule.csv
  -v, --verbose         Print log to console instead of to file.
  -c, --cleanup         Remove created temporary files on shutdown.
```

Terminal - fdenijs@frits-manjaro:~/Repo/switchdin/Python/optimisation
File Edit View Terminal Tabs Help

**Figure 5.1:** Command-line interface of the optimisation package.

```
> python3 -m venv predopt
> source predopt/bin/activate
```

Subsequently, install the required packages listed in `requirements.txt`:

```
> python -m pip install -U pip
> python -m pip install wheel
> python -m pip install -r requirements.txt
```

**Running** Once the dependencies have been installed, the optimisation program can be invoked with `python optimise.py`. This program has a number of required and optional command-line arguments. These are explained through an interactive help text, which is printed when the program is run with the `--help` flag, as shown in Figure 5.1.

The required positional arguments are three comma-separated value (CSV) files; one for each of the load and solar PV forecasts, and a third file containing the battery specifications (capacity, maximum charge and discharge power, efficiency and current state-of-charge).

The optional flags influence how the output is generated. By default, the battery schedule is exported to a new CSV, and the MiniZinc log containing statistics about the optimisation process is printed to a temporary log file. Temporary files such as the model data (dzn) and the log are by default be retained; with the `-c` flag, temporary files will be removed on program termination. The location of the schedule CSV can be influenced with the `-o` flag, and the log can be printed to terminal with the verbose `-v` flag.

# 6 Findings and next steps

In this project, we explored technical solutions to the virtual power plant control problem, targeting the integration of forecasting and optimisation from a unified predict-plus-optimise viewpoint. To this end, we undertook three work packages:

**WP1** **Dataset:** we extracted and cleaned a dataset of individual connections' baseloads and solar generation, to allow future academic research into the virtual power plant scheduling problem to use this as a reference.

**WP2** **Baselines:** we developed 5-minutely point forecasts of solar power generation and load power consumption allowing forecasts up to 48 hours in advance, using LightGBM. These forecasts are used in a global linear program constructed in MiniZinc that calculates a battery schedule for each individual unit, minimising a combination of the overall (peak) load, and individual cost-of-energy of the connection.

**WP3** **Predict+Optimise:** we developed a dynamic programming solution to the individual battery scheduling problems, which enables distributed solution method where each unit optimises their own schedule. We subsequently demonstrated that a relaxed version of the dynamic program can be used as the loss function of a forecasting algorithm, allowing the entire system to be trained end-to-end.

We show that, compared to a seasonal naïve battery schedule that is optimised for the best performance over the previous five days, a perfect forecast is worth $1.45 per unit per day (or $529.98 per unit per year) on average. Our baseline forecasts are capable of cutting the forecast errors (MAE and RMSE) approximately in half compared to a naïve forecast of the previous timestep's value ($0.9247 \rightarrow 0.5932$ for solar, $0.4777 \rightarrow 0.1862$ for load). Combined with baseline optimisation, our system is able to capture 67.5% of the available value of bat-tery scheduling, while reducing peak load to 11 kW over the minimum possible. We demonstrate that a dynamic programming solution is capable of reducing the time required to optimise a battery schedule from 0.868 sec-onds for a commercial linear programming solver, to 0.0216 seconds for a coarsely discretised discrete model. We can also use the dynamic program to train a forecast model, through application of convex regularisation of the maximum operator into a softmax, and applying gradient descent with the resulting loss gradient. If the features of the forecast function are insufficiently informative, prioritising the resulting value loss can greatly improve the quality of the optimisation solution in terms of cost-of-energy. In summary, we have proposed an efficient predict+optimise strategy to fit forecasts and calculate battery schedules at an individual unit level, im-proving individual consumers' return on investment while simultaneously enabling virtual power plant operators to achieve market objectives such as peak load reduction.

## 6.1 Next steps

Below, we list a number of possible extensions of this work to further develop the capabilities of the forecast and optimisation technology developed in this project.

- **Historic weather forecasts.** The absence of publicly available historical weather forecast data required the use of actual historical weather as proxies for weather forecasts. This leads to forecasting accuracies in this research that are expected to surpass those in a live production environment as the models would be provided with live weather forecasts. Future work would therefore benefit from making weather forecasts publicly available for researchers.
- **Additional features and hyper-parameter tuning.** In developing our forecast models we considered all available features. While we obtained good accuracy improvements, we believe that additional features

could lead to further improvements. Load power also depends on heterogeneous factors such as household size, individual lifestyle, and appliance usage, so including household level data, such as appliances, smart home sensors, and electric vehicle ownership, could significantly enhance forecasting accuracy. Furthermore, we have performed relatively little hyper-parameter tuning due to the computational resource requirements. Additional hyper-parameter tuning may bring improved performance, but will need to be traded off with the cost of doing so. Future work should investigate this trade-off.

- **Additional constraints on scheduling problem.** Currently the battery scheduling problem only considers the battery capacity limits as hard constraints; it would be straightforward to extend the model to capture additional requirements. Examples include:
  - dynamic operating envelopes, which aim to prevent voltage excursions beyond the feasible range by imposing power import and export limits. An open question is if doing so is required during training time in predict+optimise. Ideally, these constraints would not be imposed during training, because the dynamic envelopes depend on the state of the local grid, something which will change over time as upgrades are brought in.
  - battery degradation prevention, which are constraints to attempt to reduce early degradation of batteries from excessive cycling. Examples of this include limiting the total cumulative energy in and out to at most one full cycle per day, or introducing a maximum change in power between time steps, ensuring control is smooth instead of min/max.
- **Evaluation in practice.** At the moment, all our evaluation is in simulation, evaluating against historic data. An important next step for validation of the technology is a deployment on real hardware, to evaluate the feasibility of distribution of calculation (runtime of the forecast algorithm and planner on edge devices) and the impacts of real-world communication overhead.

## References

Bellman, Richard E. (1957). ``A Markovian Decision Process''. In: *Journal of Mathematics and Mechanics* 6.5, pp. 679–684.

Bergmeir, Christoph (2023). *Journal papers*. URL: https://cbergmeir.com/publications/ (visited on 2023).

De Nijs, Frits and Peter J. Stuckey (2020). ``Risk-Aware Conditional Replanning for Globally Constrained Multi-Agent Sequential Decision Making''. In: IFAAMAS, pp. 303–311.

Gamrath, Gerald, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig (2020). *The SCIP Optimization Suite 7.0*. eng. Tech. rep. 20-10. Takustr. 7, 14195 Berlin: ZIB.

Ge, Dingling, Jianyang Gu, Shunyu Chang, and JingHui Cai (2020). ``Credit Card Fraud Detection Using Lightgbm Model''. In: *International Conference on E-Commerce and Internet Technology (ECIT)*, pp. 232–236.

Gurobi Optimisation (2022). https://www.gurobi.com/.

Huangfu, Q. and J. A. J. Hall (2018). ``Parallelizing the dual revised simplex method''. In: *Mathematical Programming Computation* 10, pp. 119–142.

Hyndman, Rob J. and Yeasmin Khandakar (2008). ``Automatic Time Series Forecasting: The **forecast** Package for R''. In: *Journal of Statistical Software* 27.3, pp. 1–22.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu (2017). ``LightGBM: A Highly Efficient Gradient Boosting Decision Tree''. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc.

Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos (2022). ``M5 accuracy competition: Results, findings, and conclusions''. In: *International Journal of Forecasting* 38.4. Special Issue: M5 competition, pp. 1346–1364. ISSN: 0169-2070.

Mandi, Jayanta, Emir Demirovi?, Peter J. Stuckey, and Tias Guns (Apr. 2020). ``Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems''. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, pp. 1603–1610.

Mensch, Arthur and Mathieu Blondel (July 2018). ``Differentiable Dynamic Programming for Structured Prediction and Attention''. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 3462–3471.

Nethercote, N., P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack (2007). ``MiniZinc: Towards a standard CP modelling language''. In: *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*. Vol. 4741. LNCS. Springer, pp. 529–543.

Solcast (2023). *Historical Time Series*. URL: https://solcast.com/time-series (visited on 2023).

Ullah, Irfan, Kai Liu, Toshiyuki Yamamoto, Rabia Emhamed Al Mamlook, and Arshad Jamal (2022). ``A comparative performance of machine learning algorithm to predict electric vehicles energy consumption: A path towards sustainability''. In: *Energy & Environment* 33.8, pp. 1583–1612.

Wang, Di-ni, Lang Li, and Da Zhao (2022). "Corporate finance risk prediction based on LightGBM''. In: *Information Sciences* 602, pp. 259–268. ISSN: 0020-0255.

www.racefor2030.com.au

RACE for 2030

Australian Government
Department of Industry, Science and Resources

AusIndustry
Cooperative Research Centres Program