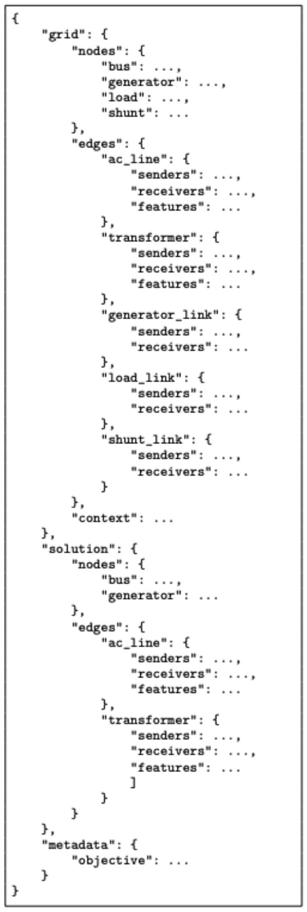
MILESTONE 1

Ewan Long, Donna Nguyen, Anne Tumlin

The Overarching Task

This project aims to leverage **large-scale OPFData** to predict the **total power generation cost** for **AC Optimal Power Flow (AC-OPF)** problems. Traditional OPF solvers rely on computationally intensive optimization techniques, making them intractable for large-scale power grids. By utilizing **machine learning (ML) models**, we aim to develop a more efficient approach that approximates OPF costs without solving the full optimization problem. To ensure our model generalizes well, we will exclusively use **grid features**, which define the physical and operational constraints of the power system. These features, detailed below, will serve as inputs to predict the total generation cost, enabling a data-driven alternative to conventional OPF solvers.

1. Data Explanation (Task 1)



```
{
    "grid": {
        "nodes": {
            "bus": ...,
            "generator": ...,
            "load": ...,
            "shunt": ...
        },
        "edges": {
            "ac_line": {
                "senders": ...,
                "receivers": ...,
                "features": ...
            },
            "transformer": {
                "senders": ...,
                "receivers": ...,
                "features": ...
            },
            "generator_link": {
                "senders": ...,
                "receivers": ...
            },
            "load_link": {
                "senders": ...,
                "receivers": ...
            },
            "shunt_link": {
                "senders": ...,
                "receivers": ...
            }
        },
        "context": ...
    },
    "solution": {
        "nodes": {
            "bus": ...,
            "generator": ...
        },
        "edges": {
            "ac_line": {
                "senders": ...,
                "receivers": ...,
                "features": ...
            },
            "transformer": {
                "senders": ...,
                "receivers": ...,
                "features": ...
                ]
            }
        }
    },
    "metadata": {
        "objective": ...
    }
}
```

Overview of Data

The accompanying figure provides a high-level overview of the **OPFData** dataset [1]. The dataset is structured at the root into three main components: **grid, solution, and metadata**. The **grid** represents the system's state before optimization, serving as the **input to the OPF solver**. This represents a snapshot of the grid state. So, the grid contains various generators, loads, and transmission lines with various physical measurements as features. The initial state of the grid is stored within the grid**.** The **solution** contains the results of the OPF computation, representing the optimized grid state by taking the input grid features, solving multiple complex optimization equations, and outputting the optimal power generation levels.

Finally, the **metadata** includes the **objective**, corresponding to the **total generation cost ($/h),** which is based on the optimal power generation level that is needed to optimize the grid (the OPF task).
We will focus on the **grid data and metadata for this project**, as these contain the necessary inputs and cost information without revealing the OPF solution itself. Excluding the **solution data** ensures that our machine learning model learns to predict the **objective cost** solely based on system constraints rather than indirectly relying on solved OPF values. This approach prevents data leakage and maintains the integrity of our predictive modeling.

Detailed Analysis

Below is the included data dictionary for the OPFData we will utilize in this project. We also define the key components to provide a better understanding of what the data is intuitively representing.

Bus (Electrical Node)
A bus represents a connection point in the power system where power is injected (from a generator) or withdrawn (by a load).

- Each bus is assigned a bus type, which defines its role in power flow analysis:
    - PQ Bus (Load Bus, Type 1): A bus where real power (P) and reactive power (Q) are specified, while the voltage magnitude (V) and phase angle (theta) are unknown and solved.
    - PV Bus (Generator Bus, Type 2): A bus where real power (P) and voltage magnitude (V) are given, but reactive power (Q) and phase angle (theta) must be determined.
    - Slack Bus (Reference Bus, Type 3): A special bus that serves as a reference point for the system, typically a large generator that adjusts power to balance supply and demand.

Generator
A generator is a power-producing unit and is represented as a PV bus.

- The generator can include attributes such as:
    - Real power generation (P_g)
    - Reactive power generation (Q_g)
    - Generation limits
    - Voltage magnitude (V)
    - Generation cost coefficients (used to model the cost of power generation).

Load
A load is a consumer of electricity and is represented as a PQ bus.

- The load can include values such as:
    - Real power demand (P_d)
    - Reactive power demand (Q_d)

Transmission Line (AC Line)

A transmission line connects two buses and allows power to flow between them.

- The transmission line can have features such as:
    - Resistance (r) and reactance (x) (impedance components).
    - Thermal limits (maximum power flow capacity).
    - Phase shift and tap ratio for transformers.

Shunt Element

Shunt elements are used to control voltage and reactive power in the system.

- The shunt elements can have features such as:
    - Shunt susceptance (B_s)
    - Shunt conductance (G_s)

Data Dictionary

First, we will pull out the features of nodes (buses) in the dataset, create dataframes of the data, and analyze the features.

grid.nodes.bus

Feature Name	Description	Data Type	Range/Domain	Mean	Units	Missing (%)
base_kv	Base voltage level of the bus.	float64	[11,500]	196.81	kV	0
bus_type	Type of bus (1: PQ, 2:PV, 3: Reference, 4: Inactive)	float64 (but categorical)	{1,2,3}	1.13	N/A	0
vmin	Minimum voltage magnitude at the bus.	float64	[0.9]	0.9	p.u.	0
vmax	Maximum voltage magnitude at the bus.	float64	[1.1]	1.1	p.u.	0

grid.nodes.generator

Feature Name	Description	Data Type	Range/Domain	Mean	Units	Missing (%)
mbase	Total MVA base (reference value) used for calculations.	float64	[0.12,616.84]	124.009	MVA	0

pg	Initial real power generation.	float64	[-0.1050,16.6629]	1.360	MW	0
pmin	Minimum real power generation limit.	float64	[-2.0000,13.6581]	0.487	MW	0
pmax	Maximum real power generation limit.	float64	[0.1200,19.6677]	2.232	MW	0
qg	Initial reactive power generation.	float64	[-0.8500,1.5800]	0.164	MVar	0
qmin	Minimum reactive power generation limit.	float64	[-5.6160,-0.0055]	-1.211	MVar	0
qmax	Maximum reactive power generation limit	float64	[0.1000, 5.6160]	1.54	MVar	0
vg	Initial voltage magnitude.	float64	[1]	1.00	p.u.	0
cost_squared	Coefficient of pg^2 in cost term.	float64	[0]	0.00	$/MW^2	0
cost_linear	Coefficient of pg in cost term.	float64	[72.2241,13523.8410]	3920.172	$/MW	0
cost_offset	Fixed cost constant term in the cost function.	float64	[0]	0.00	$	0

grid.nodes.load

Feature Name	**Description**	**Data Type**	**Range/Domain**	**Mean**	**Units**	**Missing (%)**
pd	Real power demand.	float64	[0.000080 14.020955]	0.328692	MW	0
qd	Reactive power demand.	float64	[-1.862718 2.762364]	0.100372	MVar	0

grid.nodes.shunt

Feature Name	**Description**	**Data Type**	**Range/Domain**	**Mean**	**Units**	**Missing (%)**

bs	Shunt susceptance.	float64	[-2.9032,2.1484]	0.10523	S	0
gs	Shunt conductance.	float64	[0]	0.0	S	0

Next, we will extract the features of the edges (transmission lines) from the dataset, create dataframes, and analyze these features. Our focus will be on a specific subsection of the provided data. We will concentrate solely on the features of the edges without including the encoded connectivity. The connectivity information is relevant when using graph neural networks, but we will not be employing those methods in this project. Instead, we are prioritizing simpler machine learning models, which means we will extract feature data without incorporating spatial data.

grid.edges.ac_line.features

Feature Name	Description	Data Type	Range/Domain	Mean	Units	Missing (%)
angmin	Minimum angle difference between from and to bus.	float64	[-0.523599 -0.523599]	-0.523	Radians	0
angmax	Maximum angle difference between from and to bus.	float64	[0.523599, 0.523599]	0.523	Radians	0
b_fr	Line charging susceptance at the **from** bus.	float64	[0.000000, 1.343250]	0.023	S	0
b_to	Line charging susceptance at the **to** bus.	float64	[0.000000, 1.343250]	0.023	S	0
br_r	Branch series resistance.	float64	[0.000000 1.104500]	0.012	Ohms	0
br_x	Branch series reactance.	float64	[0.000010, 1.580000]	0.037	Ohms	0
rate_a	Long-term thermal line rating.	float64	[0.089700, 30.969800]	2.606	MW	0
rate_b	Short-term thermal line rating.	float64	[0.100000, 37.163700]	2.796	MW	0
rate_c	Emergency thermal line rating.	float64	[0.100000, 46.454700]	3.157	MW	0

grid.edges.transformer.features

Feature Name	Description	Data Type	Range/Domain	Mean	Units	Missing (%)
angmin	Minimum angle difference between from and to bus.	float64	[-0.523599 -0.523599]	-0.523	Radians	0
angmax	Maximum angle difference between from and to bus.	float64	[0.523599, 0.523599]	0.523	Radians	0
br_r	Branch series resistance.	float64	[-0.004670, 0.010000]	0.004	Ohms	0
br_x	Branch series reactance.	float64	[-0.013520, 0.234690]	0.077	Ohms	0
rate_a	Long-term thermal line rating.	float64	[0.013700, 10.441400]	1.904	MW	0
rate_b	Short-term thermal line rating.	float64	[0.016400, 12.529700]	2.209	MW	0
rate_c	Emergency thermal line rating.	float64	[0.020600, 18.794500]	2.810	MW	0
tap	Branch off-nominal turns ratio.	float64	[0.916670, 1.134690]	1.011	p.u.	0
shift	Branch phase shift angle.	float64	[0]	0	Degrees	0
b_fr	Line charging susceptance at the **from** bus.	float64	[0]	0	S	0
b_to	Line charging susceptance at the **to** bus.	float64	[0]	0	S	0

Finally, we have our objective data, which will serve as the label of our ML task and represent the OPF task's total cost.

metadata.objective

Feature Name	Description	Data Type	Range/Domain	Mean	Units	Missing (%)
total_cost	AC-OPF objective achieved by conventional solver ($/h).	float64	[2.214004e+06, 2.286767e+06]	2.251436e+06	$/Hour	0

2. Definition of the Outcome and Evaluation Metrics (Task 2)

Target Definition

Our project focuses on predicting total generation cost (objective value in the dataset) for AC Optimal Power Flow (AC-OPF) problems, and we will explore how well the Machine Learning (ML) models can approximate OPF costs in the large-scale data (OPFData).

Specifically, each dataset entry contains an objective value, denoted as:

$$y_i = objective_i$$

In the chosen dataset, we can see the value is stored within the JSON field like:

```
metadata: {'objective': numeric_value}
```

An instance from the dataset like:

```
metadata
0  {'objective': 2252269.9743585344}
1   {'objective': 2256933.256383117}
2  {'objective': 2268722.8661677185}
3   {'objective': 2266446.774271797}
4   {'objective': 2232218.578735281}
```

Evaluation Metrics:

We are trying to solve a machine learning regression task, so here are some suitable metrics:

1. Mean Squared Error (MSE)
   a. Measures the average squared error of predictions
   b. Mathematical formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2$$

2. Mean Absolute Error (MAE)
   a. Provides intuitive interpretation of average cost error
   b. Mathematical formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \widehat{y_i}|$$

3. Root Mean Squared Error (RMSE)
   a. Sensitive to large errors, and also provides an error scale comparable to the generation cost unit
   b. Mathematical formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y_i})^2}$$

3. Ingest Data Files (Task 3)

Since the data is stored in a public Google Cloud Platform (GCP) bucket, we set the path to the correct directory and point it to load all the files that end with .json. After inspecting the JSON files' structure by loading a single file and reading through the scientific paper, we understood the format as shown in the first figure of part 1. The exact names of the columns were obtained by reading the paper and hardcoded during extraction (for example: transformer's columns are "angmin", "angmax", "br_r", "br_x", "rate_a", "rate_b", "rate_c", "tap", "shift", "b_fr", "b_to").

We extracted the input columns accordingly: the grid and solution are split into nodes and edges and nodes are split into several types containing feature matrices where the rows correspond to the entity number, and the columns to features whereas edges are split into AC lines and transformers. The target or objective column (whose name "total_cost" also comes from perusing the paper) was stored separately in metadata. Therefore, we extracted the metadata section in a completely different chunk of code after the features part (and in a similar fashion to how we extracted the grid but objective values are default to None if missing). This early separation could be useful for later when we do our machine learning training.

4. Relevant Data - Structuring the Dataset (Task 4)

The OPFData is a collection of power grid data (**please refer to the JSON file structure overview**). Each JSON file contains **input grid data**, which represents a snapshot of the power grid at a given moment in time (when we say snapshot from here forward we are referring to one JSON file which contains a single temporal accumulation of various grid features). There are 15,000 JSON files within our chosen dataset. Within each snapshot (JSON file), the grid consists of multiple buses, generators, loads, transformers, and transmission lines, each with various physical measurements such as voltage and power flow. Since the dataset is structured as a grid snapshot, our goal is to extract relevant information from these components to **predict the total generation cost** (metadata.objective) for each snapshot of the grid dynamics.

Within the dataset, there are **4661 buses**, but not all of them are relevant for cost prediction. Each bus has a **bus_type** feature that determines its role:

- **If bus_type = 1, it is a LOAD.**
- **If bus_type = 2, it is a GENERATOR.**
- **If bus_type = 3, it is a REFERENCE BUS (Slack Bus).**

Since **slack buses serve as reference points for power balancing** but do not contribute directly to cost, we **prune them** from the dataset. This leaves us with **only generators and loads**, which are the key components that define power generation and consumption in the system.
Each JSON file also contains **a single total generation cost value** in the metadata section. This means that **the dataset is structured such that each JSON file corresponds to one grid**

snapshot and has one associated cost value. Our prediction task is therefore structured as follows:

Input (a grid snapshot, one JSON file): Grid features across loads, generators, transmission lines, and transformers
Output: A single cost prediction (metadata.objective)

Because each snapshot contains **multiple loads, generators, and grid components** (in our dataset there are 4661 buses), we cannot structure our dataset with **one row per bus**—this would create thousands of rows per snapshot, leading to an **incorrect mapping between inputs and cost values**. Instead, we must **aggregate** relevant features **per snapshot**, ensuring that **each row represents a full grid state** with summarized generator, load, and grid constraint features.

Determining Relevant Features

For our cost prediction task and aggregation, we selected features that directly impact power generation cost while removing those that provide minimal or redundant information. **Reactive power features** (qg, qmin, qmax, qd) were dropped since they primarily influence grid stability rather than cost. **Voltage limits (vmin, vmax)** were excluded because they represent categorical constraints rather than numerical predictors of cost. **Angle constraints (angmin, angmax)** were removed as they primarily enforce power flow stability rather than affecting cost directly. Similarly, **shunt elements (bs, gs)** were omitted since they have a negligible effect on cost compared to real power demand and transmission constraints. By eliminating these features, we focus the model on the most relevant inputs, reducing dimensionality and avoiding unnecessary complexity while ensuring that our dataset remains efficient and interpretable.

Joining and Aggregating Data

Since each grid snapshot contains **multiple loads, generators, and grid components**, our **final dataset must summarize this information into a single row per snapshot**. We achieve this by **aggregating features across all relevant elements** within each snapshot.

Step 1: Aggregating Generator Features

Since each snapshot contains **multiple generators**, we **group by bus_type ID** and compute:

- **num_generators** → Total number of generators in the snapshot.
- **total_pg** → Total real power generation (pg sum).
- **avg_vg** → Average generator voltage (vg mean).
- **avg_cost_squared** → Mean of the quadratic cost coefficient (cost_squared).
- **avg_cost_linear** → Mean of the linear cost coefficient (cost_linear).
- **avg_cost_offset** → Mean of the fixed cost term (cost_offset).

Step 2: Aggregating Load Features

Each snapshot contains **multiple load buses**, so we compute:

- **num_loads** → Total number of loads in the snapshot.
- **total_pd** → Total real power demand (pd sum).

Step 3: Aggregating Transmission Line Features

Since power flow is influenced by **transmission line constraints**, we summarize these as:

- **br_r_mean** → Mean transmission line resistance (br_r mean).
- **br_x_mean** → Mean transmission line reactance (br_x mean).

- **rate_a_sum** → Sum of long-term thermal line rating (rate_a sum).
- **rate_b_min** → Minimum short-term line rating (rate_b min).
- **rate_c_max** → Maximum emergency line rating (rate_c max).

Step 4: Aggregating Transformer Features
Similar to transmission lines, transformers affect power flow, so we summarize them as:

- **trans_br_r_mean** → Mean transformer resistance (br_r mean).
- **trans_br_x_mean** → Mean transformer reactance (br_x mean).
- **trans_rate_a_sum** → Sum of long-term thermal transformer rating (rate_a sum).
- **tap_mean** → Mean transformer tap setting (tap mean).

Step 5: Extracting the Objective (Total Cost)
The **objective** provides the ground truth labels needed for **supervised learning (MSE etc.)**. By ensuring a structured dataset where each row represents **one full grid snapshot** with a corresponding cost, we maintain **data integrity** and enable meaningful **predictive modeling**, so we compute:

- **total_cost** → Total generation cost of operating the power grid for a given grid.

Step 6: Joining Aggregated Features
Once all relevant features are aggregated, we **join them together**, ensuring that **each row in the final dataset represents a single grid snapshot** with a corresponding **total generation cost**.

5. Creating Checkpoints (Task 5)

We tested the steps outlined in Task 3 and Task 4 on the first 100 files to ensure they are in the correct format. Next, we will begin expanding to the full dataset for Milestone 2. We plan to use PySpark for the machine learning component and may also use it to load the JSON files and gather feature statistics if that method proves to be faster.

6. Splitting the Data and Normalization (Task 6)

Our subset from the original OPFData dataset contains 15000 examples: from example_15000 to example_29999. Similar to the approach in the paper, we decided that the canonical dataset split for OPFData is a train / validate / test split of 0.9/ 0.05 / 0.05, resulting in 13.5k / 750 / 750 examples. The examples are i.i.d. and are taken sequentially, i.e. the train set consists of example_{15000-28499}.json, the validation set of example_{28500-29249}.json, and the test set of example_{29250-29999}.json. Normalization of data may occur depending on how skewed the statistics appear.

6. Credit Assignment Plan (Task 7)

For this project, we're aiming to divide the workload as evenly as possible, assigning tasks after each milestone is given to ensure a fair distribution. Since different milestones may require different types of work, we'll adjust assignments accordingly to keep things balanced. To track contributions, we'll use our shared GitHub repository, where everyone will push their work.

We’ll also monitor GitHub’s built-in contribution statistics to get a sense of how much each person is contributing and make adjustments if necessary to maintain fairness.

To keep everything documented, we’ll include a short write-up with each milestone detailing who did what. This will serve as a record of our contributions so that when we get to the final report, we’re not trying to piece together our individual efforts from memory. This way, we ensure credit is given accurately and fairly throughout the project.

Milestone 1 Contributions:

Anne (Leader): Explained the dataset, its structure, and collected statistics. Proposed a plan for joining and aggregating the data as needed to prepare the final feature set for the model. Wrote the data dictionary and aggregation plan.

Ewan: Defined outcome metrics. Worked on implementing the pipeline for joining and aggregating data tables and debugged the current method in Google Cloud. Identified & Dealed any Missing or corrupt data. Conducted a test run by passing the processed data through the training pipeline to generate outputs.

Donna: Led the initial data ingestion, including processing JSON files. Drafted the credit assignment plan. Contributed to multiple sections of the Milestone 1 report.

References

[1] S. Lovett *et al.*, “OPFData: Large-scale datasets for AC optimal power flow with topological perturbations,” Jun. 18, 2024, *arXiv*: arXiv:2406.07234. doi: 10.48550/arXiv.2406.07234.