Integrated Optimization of Location, Design, and Operation of Renewable Energy Systems for Urban Microgrids

Navid Shirzadi

A Thesis in the Department of

Building Civil and Environmental Engineering (BCEE)

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy in Civil Engineering

at Concordia University

Montreal, Quebec, Canada

March 2023

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared
By: **Navid Shirzadi**

Entitled: **Integrated Optimization of Location, Design, and Operation of Renewable Energy Systems for Urban Microgrids**

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy in Civil Engineering

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. Adam Krzyzak

______________________________ Thesis Supervisor
Dr. Fuzhan Nasiri

______________________________ Thesis Supervisor
Dr. Ursula Eicker

______________________________ Examiner
Dr. Luiz Lopes

______________________________ Examiner
Dr. Hua Ge

______________________________ Examiner
Dr. Andreas Athienitis

______________________________ External Examiner
Dr. Ahmed Mohamed

Approved by ______________________________
Dr. Mazdak Nik-Bakht, Graduate Program Director

«28 March 2023» ______________________________
Dr. Mourad Debbabi, Dean of Gina Cody School of Engineering and Computer Science

Abstract

Integrated Optimization of Location, Design, and Operation of Renewable Energy Systems for Urban Microgrids

Navid Shirzadi, Ph.D.

Concordia University, 2023

The building sector of urban areas plays a crucial role in carbon emissions and climate change. Distributed generation using clean energies could help to reduce emissions. Furthermore, urban microgrids increase the reliability of power supply since most of the power outages are created in the grid distribution system and transmission lines. However, a cost-effective design and operation of an urban microgrid poses challenges, such as limited space for installing the renewable components, especially in populated areas, the uncertainty of renewable resources, and the resiliency of the designed microgrid in case of not having access to the central grid. Therefore, this thesis was initiated with the objective of developing a comprehensive method for the efficient design of an urban microgrid. The developed framework consists of three main modules. The first module aims at designing an energy system for a community microgrid by sizing and finding the optimum configuration of the energy system. To resolve the spatial issue problem in urban areas, regional renewable generation is proposed in this research where clean energy is produced outside of the populated area as a virtual power plant in relation to the microgrid. A mapping model is also developed to select the best location for installing components outside the microgrid. The mapping model is connected with the optimization model to automatically generate the best configuration and location of regional generation based on several aspects of each zone. The second module deals with renewable resources and electrical load demand uncertainties and tries to reduce them by forecasting strategies. Since renewable resources such as solar irradiance and wind speed are not predictable using just historical data, hybridized numeric weather prediction (NWP) and deep learning models are offered to tackle the drawback. The last module proposes a solution to ensure resilience against power supply failures in electricity grids caused by extreme weather conditions, unavailability of generation capacities, and transmission components problems.

The discussed models were applied to one of Concordia University's largest buildings in downtown Montreal, Canada. The results show a significant improvement in the environmental aspect of the regional generation if the existing gas boiler would be substituted by electric boilers and heat pumps (using generated renewable electricity outside of microgrid), preventing emissions of about 4233 tons CO_2 and 5.3 tons NO_X per year. Using a proposed tariff structure beneficial to both the customer and utility, the resulting levelized cost of energy is about 5.3 Cents per kilowatt hour, i.e., lower than the current rate of about 6 Cents per kWh. Using the second module’s proposed hybrid models for renewable resources and electrical load demand prediction of the case study was also helpful by considerably bringing down the error. Finally, operation dispatch scenarios are developed to reinforce the system’s resiliency in severe conditions for the case study in the third module. A mixed-integer linear programming (MILP) approach is employed to identify global optimum dispatch solutions based on the next 48 h plans for different seasons to formulate a whole-year operational model. The results show that the loss of power supply probability (LPSP), as an indicator of resiliency, could be lowered to near zero while minimizing operational cost using the proposed optimal load (derived from critical load).

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Prof. Ursula Eicker and Prof. Fuzhan Nasiri for all their support throughout this adventure. I was so fortunate to use their knowledge and academic skills along with emotional support since day one, and I am ultimately grateful for that.

I also would like to express my deepest appreciation to my examination committee for their useful and constructive feedback, valuable comments, and support.

I would like to thank Mr. Daniel Gauthier (facility management of Concordia University) and Mr. Anton Kaifel (Centre for Solar Energy and Hydrogen Research) for providing some part of the data used in this study.

A highly deserved thank you goes to all my teammates in both CERC and SEISE labs, especially Dr. Ramanunni Menon, who assisted me in difficult situations.

My absolute love and thanks to my mother Zohreh, my father Mohammadreza, my lovely sisters Nadia and Nushin and my wonderful brother Hooman for all their support and encouragement during this journey which helped me to stand again in challenging situations.

Last but not least, an exceptional thanks to my lovely wife Elaheh, who I couldn't have accomplished this thesis without her endless support, love, and encouragement.

I dedicate this thesis to my loved ones: my mother, my father, my sisters, my brother, and my wife, who I cannot breathe without them.

Contents

List of Figures

List of Tables

Abbreviations

AC	Alternating Current
AI	Artificial Intelligence
AIC	Akaike Information Criterion
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BIC	Bayesian information criterion
CAD	Canadian Dollar
CHP	Combined Heat and Power
CNN	Convolutional Neural Network
CO_2	Carbon Dioxide
COP	Coefficient of Performance
CRAC	Computer Room Air Conditioning
CRF	Capital Recovery Factor
DC	Direct Current
DHI	Diffuse Horizontal Irradiance
DNI	Direct Normal Irradiance
DR	Demand Response
ELM	Extreme Learning Machine
ENCS	Engineering and Computer Science
ESS	Energy Storage System
EV	Electric Vehicle
GAMS	General Algebraic Modeling System
GDP	Gross Domestic Products
GHI	Global Horizontal Irradiance
HQ	Hydro Quebec
HQIC	Hannan–Quinn information criterion
HRES	Hybrid Renewable Energy System
HVAC	Heating, Ventilation, and Air Conditioning
IQR	Interquartile
LBQ	Ljung-Box Q-test
LCOE	Levelized Cost of Energy
LPSP	Loss of Power Supply Probability
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MBE	Mean Bias Error
MG	Microgrid
MILP	Mixed Integer Linear Programming

MINLP	Mixed Integer Nonlinear Programming
MLP	Multi-Layered Perceptron
MLR	Multilinear Regression
MSLE	Mean Squared Logarithmic Error
MW	Megawatt
NEA	Niching Evolutionary Algorithm
NO_X	Nitrogen Oxide
NPC	Net Present Cost
NWP	Numeric Weather Prediction
OF	Objective Function
PDF	Probability Distribution Function
PV	Photovoltaic
RES	Renewable Energy System
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMAX	Seasonal Autoregressive Integrated Moving Average
SVM	Support Vector Machine
SVR	Support Vector Regression
USD	US Dollar
VA	Visual Arts
ZSW	Centre for Solar Energy and Hydrogen Research

Chapter 1: Introduction

1.1. Background and Motivation

With increasing energy demand all around the world and depletion of fossil fuels, and the climate change issue caused by using them, renewable energy has attracted significant global awareness. Raising urbanization is causing a concentration of energy demand in high-density urban areas [1]. To increase the power supply's resilience and reliability, the urban energy system transition from a centralized limited capacity conventional grid to a distributed one needs to be accelerated [2].

A small-scale energy system consisting of generators, energy storage, load, and control units, which could work in an isolated or grid-connected mode, ensuring the power supply for a defined region, is called microgrid (MG) [3]. For supplying resilient power at both neighborhood and community scales, MGs can play a vital role [4]. Although controlling MG is a crucial step in increasing the reliability of a MG, designing, and sizing it in the first place could also impact the power supply resiliency.

Renewable power generation by decentralized energy systems in urban areas is limited to the available space for installing renewable components such as PV panels and wind turbines. This could be more challenging for large-scale consumers such as high-rise residential or institutional buildings typically located in populated areas with a great amount of power consumption, leading to low renewable penetration and high dependency on the central grid. Therefore, designing an urban distributed generation could be a crucial step toward the building's lower carbon emissions and improvement of the economy of renewable power generation.

The next important step after designing an urban microgrid is controlling it in an economical manner. Grid power failures are common, especially in urban areas that rely on a conventional mono grid. The most common causes of electrical disturbances in the power grid could be severe weather conditions such as storms or flooding [5] or natural hazards such as earthquakes. Beyond environmental hazards, there are other causes, such as equipment failure, transmission line damage, or cyberattacks. These may impact the operational resilience of the energy system based on their severity. Therefore, the designed energy system requires not only reliability but also resiliency which is the ability of the system to quickly recover from the events that caused the outages [6]. Although microgrids could work in an isolated mode and are a reliable solution, operation management is necessary to mitigate the unbalanced power supply and increase its quality in case of disconnection from the grid. The control strategy that helps the microgrid alleviate the consequences of major contingencies could be considered operational resiliency [7]. To increase the systems operational resiliency, the probability of loss should be lowered as much as it is possible while considering the economic aspect of the system.

Moreover, the electrical consumption and power output of renewable technologies are influenced by varying meteorological conditions that can be impacted by global warming. Using just historical data could not capture the complexity of changing wind patterns or the movement of clouds that change the solar irradiance and PV panels' power output accordingly. All these behaviors could cause significant uncertainties for unit commitment purposes, and few studies considered these uncertainties for the operational management of the energy system [8], [9]. Load and component power output prediction could reduce this uncertainty and make the model more reliable and robust.

1.2. Problem Statement and Research Questions

There are several challenges with urban microgrids that could be summarized below:

- Space Limit and urban renewable integration: One of the major problems preventing MGs usage in urban areas, especially metropolitan and downtown sectors, is the spatial issues. Renewables such as solar PV panels and wind turbines need large areas to be installed, and there are other aspects, such as wind turbines' noise and social acceptance aspects of it, etc. Low space availability and low wind speeds due to high roughness coefficients often limit the available power, thus requiring regional generation concepts.
- Carbon emissions and grid congestion: when converting existing building heating systems based on fossil fuels with electricity-driven systems, the electricity demand rises and might lead to serious grid congestion. Although zero emissions might be achieved with clean electricity, grid congestion issues need to be solved through more efficient conversion systems while minimizing cost.
- Dynamic tariff: Although dynamic electricity tariffs were considered as a tool for demand side management in some urban microgrids [10], lack of assessing it during the design

stage could lead to lower economic efficiency which could be detrimental to the both utilities and prosumers.

- Intermittency of renewable resources and load demand: Due to the intermittent nature of renewable power generation and load demand, the available resources and the demand need to be forecasted to be used in the operation management stage. However, their complicated behavior brings down the prediction's accuracy.
- Microgrid's Resiliency and Reliability: Since an urban MG is designed with considering access to the central grid, any failure could significantly impact the MG's resilience and reliability.

Based on these challenges, the thesis tries to answer several research questions for the design and operation of renewably powered urban energy systems. For the design optimization, the questions which could be derived from the first three challenges are:

- Is it feasible to generate electricity outside of the urban area (regional generation), and where is the optimum location to install the power plant in terms of cost and availability of renewable resources? How does a design of local generation with PV and low-power urban wind turbines compare economically and in terms of reliability with the design of local PV and regional generation using high-power wind turbines?
- What are the optimum configurations of local and regional renewable generation based on information such as meteorological conditions, grid infrastructure, tariff structure?
- How can dynamic tariffs be designed to be beneficial for both customer (urban microgrid user) and utility ?
- What are the best technological choices for heat generation in urban buildings taking into account economic, environmental, and grid congestion challenges?

While for optimum operation, the following questions could be raised (based on the last two challenges):

How can operational schedules be optimized under grid failure conditions using critical load concepts?

- For a given renewable energy system design and load profile, how well can the load and renewable resources be forecasted, and what economic and resiliency benefits can be expected through operational optimization?

1.3. Research Objectives

Accordingly, this thesis aims at making the urban microgrids more feasible considering economic, environmental and technical aspects by presenting a multi-stage framework that includes three main modules: 1) Design an urban microgrid with the ability to generate electricity outside of the microgrid area as a virtual power plant concept, including energy systems configuration, component sizing, land selection, and environmental assessment to address the first five research questions, 2) Resilience-oriented optimal operation module to deal with the challenge raised in the sixth research question and 3) Power consumption and renewable generation forecasting to cover the last research question. This is achieved by means of the following stages proposed:

1) Reviewing the previous research studies in the literature within the context of design and control of the community microgrids.
2) Considering different possible scenarios for designing an economically feasible renewable energy system. As shown in Figure 1, the methodology is explained in two different modules for a local and regional generation.
3) Establishing a mapping model for regional renewable power generation to deal with the limited space challenges in urban areas
4) Developing robust operational scheduling model using hybridized forecasting methods to diminish uncertainties.
5) Structuring a resilience-oriented model to bring down the central grid dependency in case of failure. As depicted in Figure 1.1, a separate methodology will be discussed for the off-grid scenario.

The proposed research framework is summarized in Figure 1.1.

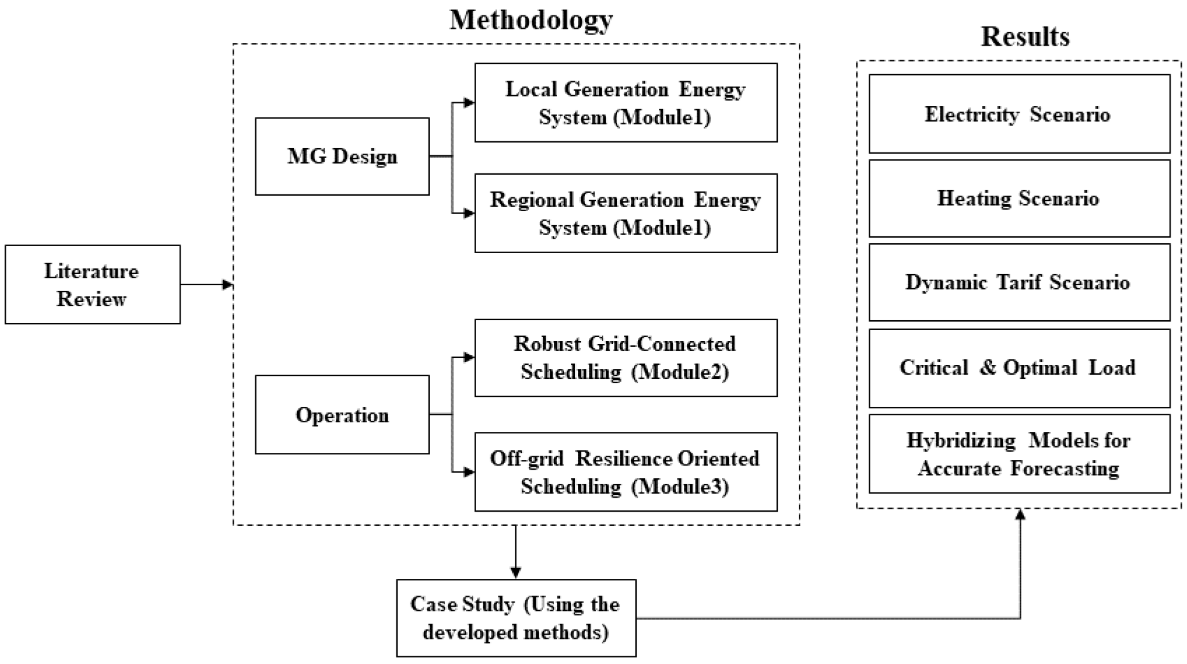

Methodology
Results
Literature Review
MG Design
Local Generation Energy System (Module1)
Regional Generation Energy System (Module1)
Operation
Robust Grid-Connected Scheduling (Module2)
Off-grid Resilience Oriented Scheduling (Module3)
Electricity Scenario
Heating Scenario
Dynamic Tarif Scenario
Critical & Optimal Load
Hybridizing Models for Accurate Forecasting
Case Study (Using the developed methods)

Figure 1.1. Research objectives framework

1.4. Thesis Structure

The remainder of this thesis is structured into five chapters. Chapter 2 represents the previous studies regarding the design and operation of urban microgrids. Chapter 3 illustrates the related methodology of the modules (PV and Wind power output models, local and regional design models, forecast models, and finally, operation model) and submodules. Chapter 4 explores the case study information and data, while Chapter 5 provides the models' results for each module and discusses comparing different scenarios. And finally, a conclusion, including a summary of the research and suggestion for future works, is provided in chapter 6.

Chapter 2: Literature Review

2.1. Optimum Configuration and Sizing

The literature has shown an increasing trend in studies that focus on sizing integrated renewable energy systems (IRES) and MGs considering the different economic, environmental, and even social aspects [11], [12]. In [13], the authors simulated and sized an urban microgrid using 100 percent renewable resources while trying to compare the economic aspects of the designed system in isolated and grid-connected modes. A multi-objective optimization model for designing an IRES, including PV panels, wind turbines, and marine energy, was proposed by [14] for a building in a coastal community. It was shown in this reference that an optimum renewable system mix, with cost, energy utilization, and loss of power objectives, that reaches 100% renewable generation in higher demand hours is a challenging task in practical planning.

Reference [15] has developed a framework with multi-perspective performance evaluation for an optimal design of a flexible hybrid power plant at a community level (Beijing, China). The results of their investigation showed that using 100% of PV rooftops and available biomass could satisfy 73% of the electricity demand with a Levelized cost of energy (LCOE) equal to 0.1030 $/kWh.

To cover the deficiencies caused by using 100% renewable resources, some studies considered the integration of auxiliary fossil fuel generators [16]. Other studies, such as [17], presented hybrid grid-connected energy systems with battery and hydrogen vehicle storage for a typical high rise residential building with 30 stories and 8 two-occupants flat and 8 four-occupants flats in each story). In this reference, the annual net grid exchange is equal to 4.55 MWh.

A large number of studies have also been conducted to develop single and multi-objective algorithms for sizing applications. In [18], they presented an algorithm for planning a residential and campus microgrid to maximize renewable generation while finding the most economical configuration. Meta-heuristic approaches were used by [19] to find the optimum size and topology of a microgrid in an off-grid mode using a super-capacitator and hydrogen fuel cell system in New Zealand. Their results showed that the LCOE of electricity and hydrogen are 0.09 $/kWh and 4.61 $/kg, respectively, which are below the case study's tariff (Between 0.11 $/kWh and 0.16 $/kWh for residential consumers based on their consumption level). Considering the concept of net zero buildings, the authors in [20] carried out research on the optimal design and operation of a solar-hydro energy system with hydrogen generation and fuel cell technology to minimize the investment cost for a 30 kW power demand residential building. Their proposed model could reduce the carbon dioxide emission by about 39546 kg while bringing down the cost by about 50.3%.

2.2. Energy System's Operation

The previous research on economic dispatching can be separated into two main categories: standalone microgrids and grid-connected microgrids. A standalone microgrid is a type of MG with no access to the grid, such as microgrids in remote areas [21]. While in the grid-connected ones, the grid is typically accessible and is part of the system, such as urban microgrids. Augustine et al. [22] employed the reduced gradient method for analyzing the dispatch rate of power for an isolated microgrid consisting of wind turbines and solar panels as main generators. Their results indicate that the system could only be operationally profitable when using governmental subsidies for the installation of solar panels. Conti et al. [23] presented research on optimal dispatching in a medium voltage islanded microgrid using the niching evolutionary algorithm (NEA). The optimization goal was to minimize the overall operational cost and pollutant emissions. Their results show the potential of the optimization process in decreasing operating cost and emissions while boosting the microgrid performance and reliability. However, they have not considered the imprecision caused by using only historical data for future predictions. In addition, their model does not provide a basis for the initial sizing of the components.

In [24], the authors investigated the effect of using a gravitational search algorithm to optimize economic dispatch and boost efficiency and performance in an isolated MG. The results that were validated experimentally indicate that the proposed method can improve MG's performance compared to conventional energy management systems. The study did not provide an approach for predicting the load demand and renewable generation levels. This could contribute to inaccuracies in scheduling. In addition, the impact of storage charging/discharging rates on operational costs were not analyzed. In [25], a day-ahead operational strategy was proposed for an urban microgrid by Kanchev et al. to optimize the CO2 emission and operational cost. They compared the optimal operational planning, considering three different objective functions, namely 1. Emission, 2. Fuel Consumption, 3. Trade-off between emission and fuel consumption with typical operational planning without optimization. Their results highlighted at least a 10% reduction in total operational cost in all optimal planning scenarios. In the other investigation that has been done by KavousiFar et al. [26], although they considered the uncertainty of the wind resources in their optimal dispatch strategy, the effect of wind power curtailment has not been evaluated.

Wen et al. [8] considered the optimal load dispatch of a grid-connected microgrid, including solar photovoltaic (PV), energy storage system (ESS), and electric vehicles (EV). They considered three different scheduling scenarios and employed a particle swarm optimization algorithm to reach the optimal schedule. Their results indicate that using ESS and EVs can decrease daily operational cost by about 9 percent. They reported the final operating cost of each scenario. However, the breakdown of costs was not provided, and as such, the impact of each objective (in the scheduling model) on the final cost was not elaborated. Since the utility and users pursue conflicting goals (utility wants to maximize its profit and users want to minimize the cost), Sun et al. recently published a paper on the day-ahead economic dispatch strategy based on game theory to ensure each one (utility and users) can achieve their optimization goal [27].

Lu et al. [28] developed a robust dispatch optimization model for a community energy hub that includes combined heat and power (CHP), heat storage, gas boiler, wind turbines, PV panels, and EV as means of reducing operational cost and emissions. Their model considered a demand response (DR) program for accommodating electrical and thermal loads. They evaluated three scenarios of EV charging/discharging and DR setting. Their results showed that the DR program with a coordinated charging/discharging mode for EV could further bring down the total cost. Although the dynamism of electricity pricing was taken into account using a robust optimization approach, the impact of possible future variations in load demand and renewable power generation levels were not analyzed.

Yang et al. [29] proposed a two-stage dispatching optimization for a grid-connected home with an integrated PV-Battery system. Their results indicated that the proposed two-stage dispatch strategy could significantly improve the user's benefit. They have also investigated the impact of sunrise time. However, the impact of battery storage system degradation on costs was not incorporated in estimating the user's benefit. The electrical consumption and power output of renewable technologies are influenced by varying meteorological conditions. Using just historical data for unit commitment will cause uncertainties, and few studies considered this uncertainty for the operational management of the energy system. Load and component power output prediction could reduce this uncertainty and make the model more reliable. With the rise of artificial intelligence and machine learning, various research focused on different forecasting types and compared the methods' accuracy.

2.3. Removing Uncertainties

The electrical consumption and power output of renewable technologies are influenced by varying meteorological conditions that can be impacted by global warming. Using just historical data for unit commitment will cause uncertainties, and few studies considered this uncertainty for the operational management of energy systems [8], [9]. Load and component power output prediction could reduce this uncertainty and make the model more reliable [8]. With the rise of artificial intelligence and machine learning, various research focused on different forecasting types and comparing the methods' accuracy.

2.3.1. Load Forecasting

Kialashaki et al. [30] developed and compared two energy demand forecasting models for the United States' residential sector based on ANN and multilinear regression (MLR) techniques. They considered using different indicators on consumption behavior and made three different models based on various features such as Gross Domestic Products (GDP), median household incomes, cost of residential electricity, and cost of residential heating oil and used ANN and MLR techniques for each model. They used the coefficient of determination (R Squared) to evaluate and compare the predictions' accuracy. Their results indicate that the coefficient of determination of the ANN models is slightly more than MLR (about 1-3 percent). In 2017, Nageem et al. [31] published research on forecasting power output of photovoltaic modules in grid-connected mode using a support vector machine (SVR) method. They compared the results of the SVR model with an analytical model. Their results indicate that although both methods' mean absolute percentage error is high, SVR can predict with about 4% less error than the analytical method, while the computational time and complexity of the analytical model are lower.

While many studies have considered artificial neural network for sequential data, few researchers focus on using a recurrent neural network (RNN) and especially the long-term short memory (LSTM) method that is basically designed for time series applications, especially the sequential data with long-term dependencies.

Rahman et al. [32] in 2018 presented an RNN - LSTM model for energy consumption medium-term prediction of a Public Safety Building in Salt Lake City, Utah (US). They also developed a simple deep neural network model to compare the RNN model results with it. They separate the load power into heating, ventilation, and air conditioning (HVAC), Computer room air conditioning (CRAC), convenience power, and elevator power. Their results demonstrate that although the RNN model has significantly less error than a 3-layer multi-layered perceptron (MLP) model in the case of predicting the HVAC load profile over an 83-day time horizon, the MLP model is more accurate in terms of CRAC load prediction. The impact of long-term dependencies is expressed as the reason in this study. However, the comparison of the two models' ability to forecast the total load prediction of the building is not reported.

2.3.2. Wind Speed Forecasting

Wang et al. [33] proposed a hybrid model including an autoregressive moving average (ARMA) and a bivariate fuzzy time series model to forecast daily wind speed in Hainan province in China. Their results show that while the mean absolute percentage error (MAPE) of conventional (ARMA and ARIMA) models for four different sites ranges between 18.15-22.08%, the hybrid model reduces the error to the range of 16.64 – 18.29% for day-ahead wind speed forecasting.

In 2017, Yatiyana et al. [34] presented a statistical model using an autoregressive integrated moving average (ARIMA) to predict wind speed and direction in Western Australia. The shorter response time was mentioned as the reason for the method selection. Their results show that the MAPE for wind speed prediction of 6 hours is 4.9% and 15.6% for wind direction forecasting of 7 days lead time. However, integrating these two models into a single model to increase the overall accuracy was not reported, and capturing the fluctuations of the wind speed with the ARIMA method was not explained. A day ahead and two days ahead forecasting using fractional-ARIMA is reported in [35], and it showed a significant reduction in error and improvement in accuracy compared with persistence methods. To consider the seasonality of the training data, seasonal autoregressive integrated moving average (SARIMA) has been used in several studies. Wang et al. [36] used SARIMA for forecasting daily and monthly wind speed in four sites in northwestern China, and because of the nonlinearity and non-stationary inherent of wind speed, they hybridized it with extreme learning machine (ELM) and Ljung-Box Q-test (LBQ) to improve the accuracy.

For instance, in one of the sites, the results of the mean daily forecast showed about 34 percent MAPE for a single SARIMA method, while their proposed hybrid model brought down the error to about 14 percent.

Other methods, such as employing fuzzy theory [37], [38] and also machine learning approaches, such as using support vector machine (SVM) [39], have also been used by researchers for day-ahead wind forecasting. However, artificial neural network (ANN) gained the most attention and is broadly used for wind speed forecasting, either on its own or as a hybrid model in combination with other models (e.g., statistical models). In [40], the authors reported a comparison between ANN, ARIMA, and a hybrid model, which is a combination of ARIMA and ANN for wind speed forecasting in three regions in India. Their results indicate that the hybrid model could predict the wind speed with less error regardless of the linear and non-linear behavior of wind speed. Although the hybrid model had a significantly lower error when compared to the ANN-only model, the MAPE of the hybrid model forecast (18-25%) for different lead times (1 hour, 3 hours, 8 hours, and 24 hours) still seems high.

In recent years, deep learning methods such as recurrent neural network (RNN), Elman neural network, and convolutional neural network (CNN) are gaining attention for time series forecasting given their inherent ability to deal with sequential data [41]–[44]. Liu et al. [45] presented a hybrid model that consists of an Elman neural network and a long short-term memory (LSTM) for wind speed forecasting. Their results indicate that the LSTM could be suitable for non-stationary wind speed forecasting, and also their proposed hybrid model could forecast with reasonable accuracy. In other research that has been presented by Wang et al. [43], wind power was forecasted using a CNN model, and their results show that the proposed model was sufficient.

While AI and statistical methods provide proper results in most of the forecasting horizons (short-term, medium-term and long-term), using physical approaches becomes necessary in short and very short-term horizons since the impact of atmospheric dynamics becomes more important [46].

Numerical weather prediction (NWP) models are mathematical models describing the current and future status of the atmosphere and surface (ocean and land) with a typical forecast horizon of one to two weeks. In [47], the authors proposed a model based on numerical weather prediction and historical measurements that combines multiple sources of past physical model outputs. Their developed model was then applied to forecast the wind speed in a region near the US. Great Lakes. Their results show improvement in the root mean squared error of the proposed model.

Since the NWP is influenced by initial conditions, it could result in a considerable error for short-term wind forecasting as it is slowly updated and lags behind the actual changes [48].

The above literature shows that AI-based methods, statistical methods, and hybrid models have been considerably used for day-ahead wind speed forecasting. However, due to the unexpected behavior of the wind and its direct relation with physical indicators, the proposed models could not be practically used where higher accuracy is required, such as operational control of a microgrid. Moreover, the methods such as the NWP have also been employed to predict wind speed. However, still, it only considers current physical conditions, and it cannot learn from the past wind speed values and unexpected changes.

This thesis aims to advance the knowledge for more accurate wind speed forecasting that is used in the operation planning of an urban microgrid. The novelty of the proposed approach is in developing a hybrid model consisting of Weibull distribution, LSTM, and NWP models to reduce the error involved with wind speed prediction using a single LSTM model by considering the distribution probability of the historical wind speed data and also the physical description of the area. The main contributions of this section, with respect to the prior literature, are as follows:

- A hybrid model is proposed to circumvent the inaccuracy of the single statistical approaches. In the proposed model, the LSTM method is used, which has several advantages over the conventional feed-forward neural network.
- Creating a Weibull distribution of the wind speed, predicting the wind speed based on a stochastic approach, and combining the probability distribution of the wind speed with the LSTM model creates an integrated model with less error compared to a single LSTM and SARIMA model with exogenous variables.

- Proposing a hybrid model that includes the NWP model's result and AI models with minimum error for short-term forecasting applications (Just for clarity, every time we refer to short-term in this paper, it means 24-72 hours forecasting).

2.3.3. Solar Irradiance Forecasting

Various methods have been used to forecast solar irradiance in previous studies. These methods could be classified into two main categories: numeric weather prediction (NWP), statistical, and learning times series methods. Persistence predictions as the simplest time series models considers the forecast of the irradiance model in the next prediction horizon equal to the current irradiance at the surface [49]. Persistence models are typically used as a naïve predictor, and the forecasting performance of the other robust methods is compared with them [50]. While statistical models use historical data to predict future solar irradiance, NWP models forecast the current and future states of the atmosphere and surface based on duplicating the physical phenomenon. Moreover, different machine learning techniques have recently been widely used for solar irradiance forecasting based on learning the pattern of historical data.

A comparison of different time series solar radiation forecasts has been done by Reikard [51] for different time resolutions (5, 15, 30, and 60 minutes). The result of his work shows that the autoregressive integrated moving average (ARIMA) model can obtain better accuracy in most of the forecast horizons (1, 2, 3, and 4 hours) compared with other statistical and learning models such as neural network (NN). In another study, Mellit et al. [52] used a multilayer perceptron (MLP) technique to predict day-ahead solar irradiance by using temperature as an independent variable. Their forecasting results showed a better correlation coefficient for sunny days (98%) and less for cloudy days (95%). In [53], the authors use a multistage neural network instead of a single-stage neural network for daily global horizontal irradiance forecasting. Their result shows that their proposed model could reduce the mean bias error (MBE) from about 30% (single-stage NN) to about 20%. However, in all mentioned statistical and learning models, the impact of unexpected cloud movement could not be captured from the historical data. Especially on cloudy days, the inaccuracy of these models will rise.

As advanced machine learning methods, deep learning algorithms are also employed to better deal with the dependencies in historical data. Sun et al. [54] used a convolutional neural network (CNN) to forecast PV output using sky images. In another study, Qing et al. [55] investigated the ability of the LSTM method (using meteorological features) for the day ahead prediction with hourly resolutions. Moreover, the hybridized models consist of CNN, and LSTM models also developed in recent years for solar radiation forecasting [56], [57]. Other types of hybrid machine learning models were also developed for solar irradiance forecasting to improve efficiency, and a complete study has been done in [58].

On the other hand, the NWP models that integrate hydrodynamic equations with numeric methods are mainly used with an extended forecasting horizon (up to 15 days) [59]. Although the NWP model could be a good choice for longer-term forecasts, it could result in lower accuracy when it comes to short and very short-term forecasting. Moreover, it is considerably impacted by its initial conditions.

2.4. Operation and Resiliency

A microgrid is a small-scale energy system including distributed generators, energy storage, load, and control units, which could work in a grid-connected or off-grid mode, ensuring the power supply for a defined region [60]. Microgrids can play a significant role in supplying resilience at the neighborhood or even community level [61]. Although microgrids could work in an isolated mode and being a reliable solution, operation management is necessary to mitigate the unbalanced power supply and increase its quality in case of disconnection from the grid. The control-based strategy that helps the microgrid mend and alleviates the consequences of major contingencies could be considered operational resiliency [62]. To increase the systems operational resiliency, the probability of loss should be lowered as much as it is possible while considering the economic aspect of the system.

During the last decade, several types of research have been accomplished on optimally controlling a microgrid's operation. In [63], the authors studied the different optimal dispatching procedures of a grid-connected microgrid. They compared the ability of different optimization methods to minimize the operation cost. In [64] and [13], scheduling problems were solved considering several uncertainties to bring the operating cost to the minimum level.

Augustine et al. [22] investigated the dispatch rate of power for a standalone microgrid consisting of wind turbines and solar panels as main generators using the reduced gradient method. Their results show that using governmental subsidies for the installation of solar panels could cause the system to be operationally profitable. Lu et al. [28] proposed a robust optimal dispatch model for the energy management purpose of a community energy hub, considering the uncertainties of electric vehicles (EVs) and electricity prices. Their results indicate that adopting a charging/discharging mode instead of a single charging mode could bring down the total operational cost of the energy hub.

Various other studies also investigated the different ways of mitigating the operating cost of the microgrid, considering the reliability along with economic aspects. In [65], the authors investigated the operation cost of a microgrid in both grid-connected and off-grid mode with simulation results while considering the reliability using the scenario-based approach to evaluate the reliability index of the system. In the other study that was done by Costa et al. [66], they evaluated the economic benefits while raising the system's reliability.

In recent years, several studies have been carried out on controlling the microgrid using resilience-oriented approaches. In [67], the authors presented a resilience-based technique by taking into account both the survivability of critical loads under emergency conditions and the feasibility of islanding mode under normal operating conditions. In 2021 another study [68], investigated the various disturbances inside the microgrid and the authors proposed a methodology for identifying the vulnerable components and ensuring their operational resilience. A real-time control strategy based on model predictive control techniques is suggested in [15], in accordance with the microgrid's schedule. In 2022, Javier et al. [69] proposed a stochastic model predictive technique to consider feasible transitions from grid-connected to off-grid modes for different scheduling horizons.

Most of the above studies focused on how to increase the reliability aspects based on a trade-off between reliability and the system's operating cost. Nevertheless, they have not investigated the resilience of the microgrid during its operation by generating 100% renewable energy and how to improve the microgrid's reliability and minimize the power supply's deficiency concerning economic factors without using fossil fuels as auxiliary powers. Furthermore, previous literature has not explored enhancing the system's resiliency and reducing the costs associated with severe power outages.

This thesis focuses on the optimum operation design of an urban microgrid while exploring ways of improving its resiliency. The main contributions of this section of the research could be summarized as considering a penalty for loss of power supply probability to improve the energy system's reliability along with optimum loss coefficient selection considering both reliability and economic aspects. Moreover, the possibility of maximizing reliability by diminishing the loss of power supply probability while covering the critical load of an education building in an urban area is also investigated in this research. This thesis also proposes the term "Optimal Load" which satisfies the surplus power constraint while maximizing the microgrid's reliability.

Chapter 3: Methodology

3.1. Introduction

Based on the discussed research objectives, the energy system considered in this thesis includes PV panels, wind turbines, converters and inverters, battery storage system, load, and central grid, which need to be designed in the first step for supplying electricity demand. To also cover the heat demand, heat pump and electric boilers could be used in different scenarios and compared with gas boilers. The overall flowchart of the energy system is shown in Figure 3.1.

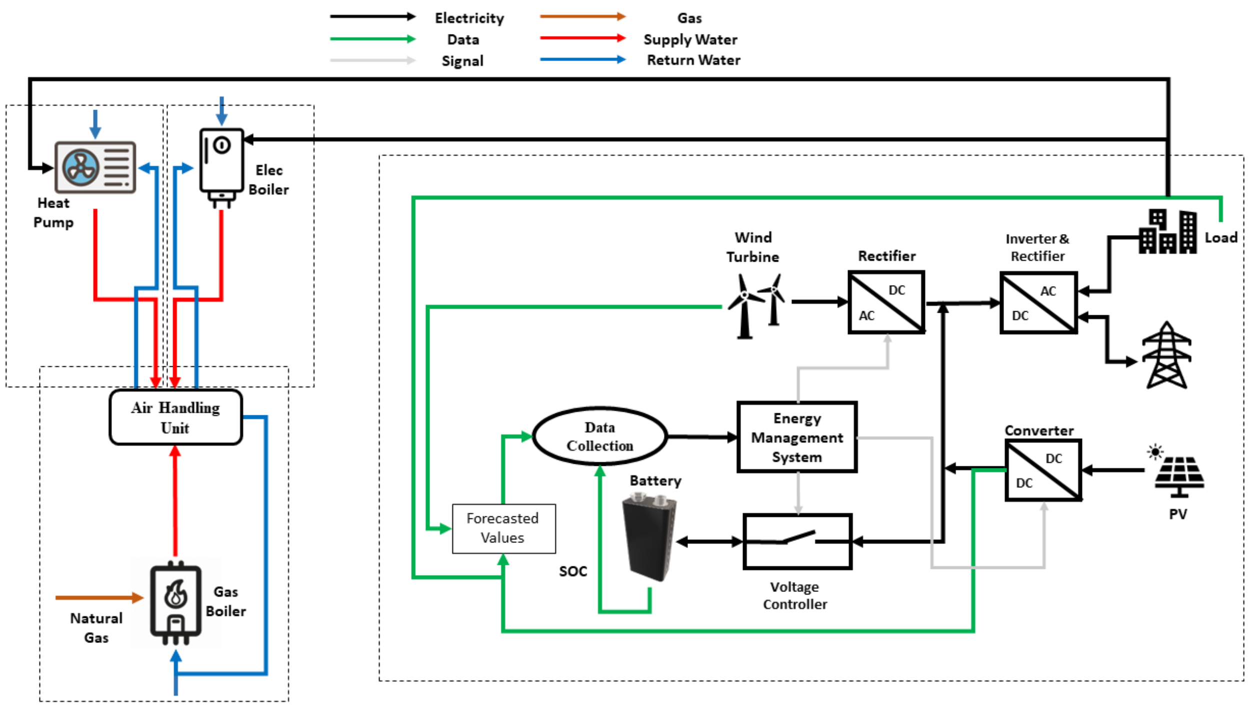


Figure 3.1. Energy system's flowchart

In the following, the methods used in the main components of the system and the overall design model will be explained. At the end, the methodology of the optimal operation model will also be discussed.

3.2. PV Model

To calculate the power output of a PV module, the below parameters and variables are considered:

1. Temperature model: To design a PV system, a thermal model is required to predict the operating temperature of the module. To calculate the module's temperature, first, the module's back surface temperature (T_m) should be determined with the below equation [70]:

$$T_m = I \times \exp(a + b \times W) + T_a \tag{3.1}$$

Where I is solar irradiance on the module surface (W/m2), W is the wind speed at 10m height, T_a is ambient temperature, a is a coefficient that shows the upper limit for the temperature at low wind speed and high solar irradiance, while b is a coefficient determining the rate at which PV module temperature reduces with rising the wind speed (a and b should be calculated from temperature measurements recorded over several different days).

After calculating T_m, then the temperature of the cell (T_c) could be calculated via the below equation [70]:

$$T_c = T_m + I/I_0 \times \Delta T \tag{3.2}$$

Where I_0 is reference solar irradiance (1000 W/m2), and ΔT is the temperature difference between the cell and module back surface at the reference irradiance level (1000 W/m2). ΔT depends on the insulation of the back surface, if insulated, it will be 0, and if not, it is usually 2 or 3 ℃ [70].

Coefficients a and b, and ΔT could vary based on module type and mount method (Table 3.1 [70]). In open rack mount, the arrays are installed on ground level with a tilt angle that allows the air to flow around the panels, while in the roof mount, PV arrays are close to the sloped roof with fixed tilt and limited possible airflow [71].

Table 3.1. PV module’s information

Type	**Mount**	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{\Delta T}$
Glass/cell/glass	Open rack	-3.47	-.0594	3
Glass/cell/glass	Roof mount	-2.98	-.0471	1
Glass/cell/Polymer Sheet	Open rack	-3.56	-.0750	3
Glass/cell/Polymer Sheet	Insulated Back	-2.81	-.0455	0
Polymer/thin-film/steel	Open rack	-3.58	-0.113	3

2. Pressure: evaluating the site pressure based on the altitude of the location. Table 3.2, assumed based on [79], shows the atmospheric parameters.

Table 3.2. Assumed atmospheric parameters

Parameter	**Value**	**Unit**
Pressure at zero altitudes (P_0)	101325	Pa
Temperature at zero altitudes (T_0)	288.15	K
Acceleration due to gravity (g)	9.80665	m/s^2
Lapse rate (L)	-6.5×10^{-3}	K/m
Gas constants for air (R)	287.053	J/(kgK)
Relative humidity (Rh)	0%	dimensionless

3. Air mass (AM): the length at which sunlight goes through the atmosphere normalized to the shortest possible path length (when the sun is directly overhead). As the light passes through the atmosphere, its power reduces and is absorbed by air and dust, and the AM could quantify this. The zenith angle of the sun is used to calculate the relative AM. The absolute AM then could be evaluated based on relative AM and estimated pressure.
4. The angle of incidence (aoi): The angle between the solar vector on the surface and the surface's normal vector. The input parameters to calculate "aoi" are as follows:
   - Surface azimuth
   - Surface tilt
   - Solar zenith
   - Solar azimuth
5. To calculate GHI (Global horizontal irradiance), DNI (Direct normal irradiance), and DHI (Diffuse horizontal irradiance), Ineichen/Prez model is employed. Apparent zenith, absolute airmass, altitude, Linke-turbidity, and dni extra are the parameters used to calculate the irradiances.
6. The effective irradiance then is calculated based on the direct and diffuse irradiance incident over the module (evaluated in step 5), angle of incident (calculated in step 4), absolute airmass (calculated in step 3), and the module type (should be selected from the module libraries).

7. Based on the effective irradiance, cell temperature, and module type, the DC power output of the PV module will be calculated.
8. In the last step, based on the selected inverter type, DC voltage, and DC power output, the AC power output is determined using Sandia's grid-connected PV inverter model [72].

3.3. Wind Farm Model

For designing the wind farm, several parameters are considered and listed below:

1. **Wake effect:** The wind speed and, consequently, the power output of the wind turbines in the downwind direction in the wind farm reduce since the turbines in the upwind direction convert wind energy into wind power. This is called the wake effect [73]. In this research, the wake losses are calculated by considering wind farm efficiency.
2. **Smoothing:** Since wind speed has stochastic behavior, it causes frequency fluctuation in the grid. Therefore, smoothing strategies are required to control this issue. In individual wind turbines, smoothing strategies could be categorized into two main classes. Storage devices and without energy storage devices [73]. Batteries, capacitors, and flywheels could be considered popular storage devices; however, they have fixed and maintenance costs. The other category is inertia, pitch angel control, and DC-link voltage control which are considered as smoothing without energy storage. Furthermore, in the wind farm, since the power generated from individual units is not fully correlated, smoothing out the short-term fluctuations will be done when aggregating the generated power [74]. In this research, the smoothing is set to True in the wind farm simulation model to smooth the power curve before aggregating wind turbines' power curves to one united power curve that represent the wind cluster. The smoothed power will be calculated via the below equation [74]:

$$P_{smoothed} = \sum_{v_i} \Delta v_i \times P(v_i) \times \frac{1}{\sigma\sqrt{2\pi}} exp\left[\frac{(v_{std} - v_i - \mu)^2}{2 \times \sigma^2}\right] \quad (3.3)$$

Where P is power (W), v is wind speed (m/s), σ standard deviation (Gauss Distribution), μ is average (Gauss Distribution), v_{std} is the standard wind speed in the power curve and Δv_i is the interval length between v_i and v_{i+1}. In the simulation model Δv_i is given to the model as a block width parameter that is considered default (0.5).

3. **Standard deviation method:** The method that should be used for calculating the standard deviation for Gauss distribution. This parameter is only required to be set if the smoothing is set to True. Available methods are: "turbulence intensity" and "Staffell Pfenninger". In this research, turbulence intensity is selected as the standard deviation method.
4. **Wind speed model: "**Hellmann", "logarithmic", and "interpolation_extrapolation" are the available methods. In this research, the Hellmann method is selected as the wind speed model, and it is explained in the following:

Correlating the wind speed at two different heights is how Hellmann exponential law works, and it will be evaluated via below formula:

$$\frac{v}{v_0} = \left(\frac{H}{H_0}\right)^{\alpha} \tag{3.4}$$

Where v is the wind speed in height H and v_0 is the speed in the height H_0 that typically considered 10 meters. Also, α is the Hellmann exponent or friction coefficient that typically, for open lands, it is considered equal to 1/7 [75]. However, various indicators could impact this coefficient, such as height, land features, and temperature [76]. Table 3.3 shows the coefficient for different land types [77].

Table 3.3. Different Land's friction coefficient

Land Type	$\boldsymbol{\alpha}$
Smooth hard ground, calm water	0.1
Tall grass on level ground	0.15
High crops, hedges and shrubs	0.2
Wooded countryside, many trees	0.25
Small town with trees and shrubs	0.30
Large city with tall buildings	0.40

In this research, different wind farm power outputs are considered for different locations based on the land type shown in Table 3.3.

5. **Density model:** "Barometric" (dropping the air pressure by 11.3 pascals per meter in the first 1000 meters above the sea, "ideal gas" ($PV = nRT$) and "interpolation_extrapolation" are the available models.

   Below formula (3.5) [78] is used for the barometric model:

$$\rho_{hub} = \left(\frac{P}{100} - \left(h_{hub} - h_{p,data}\right) \times \frac{1}{8}\right) \times \frac{\rho_0 T_0 \times 100}{P_0 T_{hub}} \tag{3.5}$$

Where P is pressure (Pa), T is temperature (K), h is height and ρ is density (kg/m3). Also, $h_{p,data}$ is height of the measurement or model data for pressure P_0, T_0 and ρ_0 are ambient pressure, temperature, and density, respectively. T_{hub} is the temperature at hub height h_{hub}.

While the ideal gas model uses the below equation (3.6) to calculate the pressure at hub height:

$$\rho_{hub} = {}^{P_{hub}}\!/_{R_s T_{hub}} \tag{3.6}$$

Where R_s is specific gas constant (287.058 J/kg*K).

6. **Temperature model:** To calculate the temperature of the air at hub height, two options are available. The first uses linear interpolation-extrapolation, and the other uses the linear gradient method. The linear gradient method is selected in this study, and the following equation (3.7) is used to calculate the temperature at the hub height based on this method:

$$T_{hub} = T_{air} - 0.0065 \times (h_{hub} - h_{T,data}) \tag{3.7}$$

   Where $h_{T,data}$ is the height that T_{air} is measured.

7. **Density correction:** In case the parameter “power-output-model” is set to "power-curve" instead of "power coefficients curve, the density correction has to be selected as True to correct the power curve based on the air density at hub height.
8. **Obstacle height:** is given to the model when there are obstacles with a certain height around the turbine. In case there is no obstacle, this parameter could be set as 0.

The input data to the wind farm simulation model is hourly resolution, and it has 6 features as below:

- Air pressure in 0 height
- Air temperature in 2m height
- Wind speed in 10m height
- Wind speed in 50m height
- Roughness length on 0m
- Temperature in 10m

3.4. Local and Regional Generation Design

3.4.1. Local Design optimization model

Because of the nonlinearity and integrality that exist in design optimization equations (it will be explained in the following sections), an MINLP model needs to be developed, and a proper solver needs to be selected to deal with it. The flowchart of the proposed methodology for the local generation design model is shown in Figure 3.2.

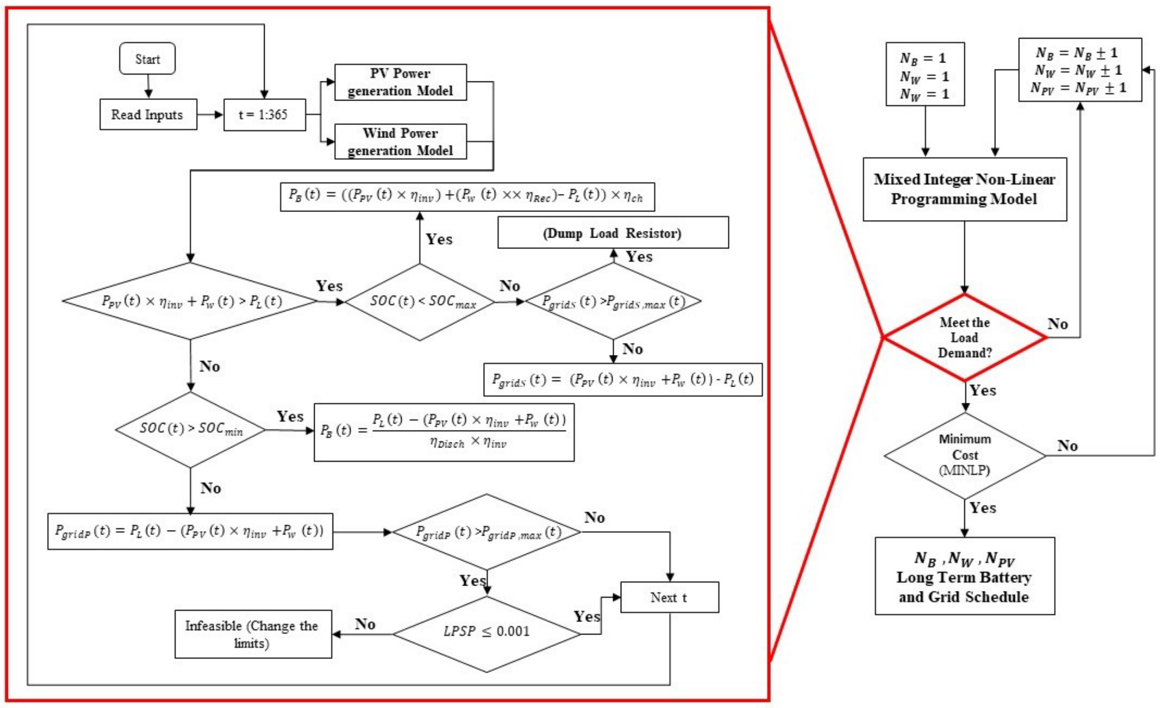

Start
Read Inputs
t = 1:365
PV Power generation Model
Wind Power generation Model
$P_B(t) = ((P_{PV}(t) \times \eta_{inv}) + (P_w(t) \times \times \eta_{Rec}) - P_L(t)) \times \eta_{ch}$
(Dump Load Resistor)
Yes
$P_{PV}(t) \times \eta_{inv} + P_w(t) > P_L(t)$
Yes
$SOC(t) < SOC_{max}$
No
$P_{gridS}(t) > P_{gridS,max}(t)$
Yes
No
$P_{gridS}(t) = (P_{PV}(t) \times \eta_{inv} + P_w(t)) - P_L(t)$
No
$SOC(t) > SOC_{min}$
Yes
$P_B(t) = \frac{P_L(t) - (P_{PV}(t) \times \eta_{inv} + P_w(t))}{\eta_{Disch} \times \eta_{inv}}$
No
$P_{gridP}(t) = P_L(t) - (P_{PV}(t) \times \eta_{inv} + P_w(t))$
$P_{gridP}(t) > P_{gridP,max}(t)$
No
Yes
Next t
Infeasible (Change the limits)
No
$LPSP \leq 0.001$
Yes
$N_B = 1$
$N_W = 1$
$N_W = 1$
$N_B = N_B \pm 1$
$N_W = N_W \pm 1$
$N_{PV} = N_{PV} \pm 1$
Mixed Integer Non-Linear Programming Model
Meet the Load Demand?
No
Yes
Minimum Cost (MINLP)
No
Yes
N_B, N_W, N_{PV}
Long Term Battery and Grid Schedule

Figure 3.2. Proposed design flowchart

The objective function and the constraint of the problem for both local and regional generation are as follows:

Objective function

The objective function (OF) includes several terms, and it is explicated below:

- Equipment costs: This consists of the fixed purchasing cost, installation cost, and operation and maintenance (O&M) cost of the equipment employed in the energy system. In this thesis, fixed and installation costs are considered as a unique cost.

$$EC = N_{PV} \times (C_{PV} + C_{Conv}) \times \left[1 + \frac{C_{O\&M}}{CRF}\right] + N_{WT} \times C_{WT} \times \left[1 + \frac{C_{O\&M}}{CRF}\right] + N_{WT} \times C_{Rec} + N_{Bat} \times (C_{Bat} + C_I) \tag{3.8}$$

Where N_{PV}, N_{WT} are the number of PV panels and wind turbines, respectively while N_{Bat} is the capacity of the battery storage system. C_{PV}, C_{Conv}, C_{WT}, C_{Rec}, C_{Bat} and C_I are PV, converter, wind turbine, rectifier, battery storage, and inverter initial costs, respectively. Moreover, $C_{O\&M}$ is operation and maintenance cost, which is the percentage of the initial cost of the component, and CRF is the capital recovery factor that should be calculated using the below equation:

$$CRF = \frac{i \times (1+i)^n}{(1+i)^n - 1} \tag{3.9}$$

Where i is the real discount rate, and n is the project's lifetime. The real discount rate could be expressed as below:

$$i = \frac{\propto - f}{1 + f} \tag{3.10}$$

Where $\propto$ is the nominal discount rate and f is the inflation rate. To calculate the annual capital recovery, the below formula could be used and multiplied to the related cost:

$$CRFA = n + \frac{1}{(1+i)^n} \tag{3.11}$$

The number of PV panels and wind turbines are limited based on the available area for components installation.

- Battery operation: which includes charging and discharging costs through the whole horizon.

$$BO = B_{OC} \times \sum_{t=1}^{T} \frac{P_{Ch}(t) + P_{dis}(t)}{CRF} \tag{3.12}$$

Where B_{OC} is the battery operating cost factor (\$/kWh) and $P_{Ch}(t)$ and $P_{dis}(t)$ are charging and discharging power in each time step.

- Battery maintenance and replacement cost: which considers the cost of storage's maintenance and replacement through the project's lifetime using CRF:

$$BM = (M_C \times Max_{CD} + RP_C) \times N_{Bat} \times CRFA \tag{3.13}$$

- Grid trade-off: which is the summation of purchasing power ($P_{pg}(t)$) from and power selling ($P_{sg}(t)$) to the conventional grid:

$$GT = \sum_{t=1}^{T} \frac{CP_g \times P_{pg}(t)}{CRF} - \sum_{t=1}^{T} \frac{CS_g \times P_{sg}(t)}{CRF} \tag{3.14}$$

- Loss of Power Supply Probability (LPSP): To minimize the probability of loss occurrence and the amount of it, the below term needs to be added to the objective function:

$$LS = \sum_{t=1}^{T} Loss(t) \times \theta(t) \tag{3.15}$$

Where Loss(t) is the amount of loss in each time step and $\theta(t)$ is the penalty coefficient that could be different for each time steps.

- Environmental cost: Purchasing from the central grid could lead to a penalty if there is non-clean energy source portion in the central grid electricity. Therefore, a term is added to the objective function to consider this as below equation (3.16):

$$EN = \sum_{t=1}^{T} \frac{P_{pg}(t) \times \varepsilon \times \vartheta}{CRF} \tag{3.16}$$

Where ε is the portion of a central grid's electricity that is non-renewable energy and ϑ is the assumed penalty coefficient for that portion.

Finally, the objective function of the optimization model, which is the net present cost of the energy system (NPC), is defined based on the below equation:

$$\text{Min}\, OF = EC + BO + BM + GT + LS + EN \tag{3.17}$$

Constraints

The NPC of the system is minimized based on the following constraints:

The operating cost of the system is minimized subject to the following constraints:

- Battery Capacity: The capacity of the battery storage system in the first time step (Eb[1]) is equal to the initial state of charge of the battery, while in the next time steps, the capacity of the battery is calculated based on the charge/discharge of it in that time step (equation 3.18):

$$Eb(t) = \begin{cases} SOC_{initial} \times C_T & t = 1 \\ Eb(t-1) + Pch(t) * \eta_c - \dfrac{Pdis(t)}{\eta_d} & t = T \end{cases} \tag{3.18}$$

Where $SOC_{initial}$ is the initial state of charge of the battery (%), C_T is the total capacity of it (kWh) and η_c and η_d are charging/ discharging efficiencies, respectively. Furthermore, the state of charge of the battery in each time step ($SOC(t)$) should follow the below inequality:

$$SOC_{min} \leq SOC(t) \leq SOC_{max} \tag{3.19}$$

- Local renewable generation: To balance the used renewable power in the microgrid and the surplus power, two constraints need to be added as follows for both wind power and PV power generation:

$$\begin{cases} P_W(t) = Ewind(t) + E_sWind(t) \\ P_{PV}(t) = EPV(t) + E_sPV(t) \end{cases} \tag{3.20}$$

Where $Ewind(t)$ and $EPV(t)$ are the used wind and PV power in each time steps, respectively while $E_sWind(t)$ and $E_sPV(t)$ are the surplus electricity of the wind and PV generation. $P_W(t)$ is the total power generated by the wind turbines in time step t and $P_{PV}(t)$ is the total power generated by the PV panels in time step t.

- Reliability: Although a penalty is added to the objective function to minimize the loss, it still needs to be limited to the certain amount that the user should specify. Therefore, the LPSP is calculated with the below equation:

$$LPSP = \frac{\sum_t^T Loss(t)}{\sum_t^T L_d(t)} \tag{3.21}$$

And then, the amount of LPSP should be less than a percentage (φ) that should be defined by the user.

- Grid Trade-off: The energy system is designed to be grid-connected with the ability to access the grid anytime. Therefore, the formulas that should calculate the amount of trade-off between grid and microgrid are the following equations. This equation shows the selling to the grid constraint ($P_{GS}(t)$).

$$P_{sg}(t) \leq (E_s Wind(t) + E_s PV(t)) \times \vartheta \times \varphi \tag{3.22}$$

Where ϑ is the portion of electricity that could be sold to the grid and φ is the binary variable to switch selling to off mode when purchasing from the grid.

While selling to the grid is constrained by the amount of surplus power generated locally by microgrid, purchasing from the grid should be just defined based on the upper limit that should be specified by the grid (the maximum amount of power that could be purchased based on the grid capacity). Therefore, it could be calculated based on the below formula:

$$P_{pg}(t) \leq GC \times \mu \tag{3.23}$$

Where GC is grid purchasing limit and μ is the binary variable to switch purchasing to off mode when selling to the grid.

- Energy Balance: The equation that forces the equity between supply and demand to minimize the loss of power supply is called the balance equation, and it is expressed via equation 3.24.

$$P_W(t) + P_{PV}(t) + Pdis(t) + P_{GP}(t) \geq L_d(t) + Pch(t) + P_{GS}(t) \tag{3.24}$$

Where $L_d(t)$ is the total load demand in time step t.

3.4.2. Mapping model

A crucial step in regional generation is finding an optimum place to install the renewable plant (wind farm) outside of the microgrid's neighborhood. Therefore, a mapping model should be developed. The mapping model consists of several maps: land type [79], grid power lines, and transformer location map. The land type map is used as the base map, and the other two maps are plotted on it. Furthermore, a circle is required to limit the area to certain neighborhoods. A squared mesh is also located in the circle to specify the zones the renewable plant could be installed. A model is developed to find the maximum number of squares that could be placed into the circle. The square mesh function is defined in a way to maximize the number of squares that could be located in the circle. If more than 50 percent of the area of the square is inside of the circle, then it will not be removed.

The user should specify the radius of the circle and the square side of each zone. The information of each zone, such as the land's type and cost, the number of turbines that could be installed (Turbine density), and the distance to transformer stations, could be extracted from the mapping model by positioning the circle with the squared mesh on the maps.

The output of the mapping model could then be used as inputs of the optimization model and also to find the wind power generation in each zone based on the Hellmann exponent selection discussed in Table 3.3.

The proposed framework for the mapping model and extracting the related data from each zone is illustrated in Figure 3.3.

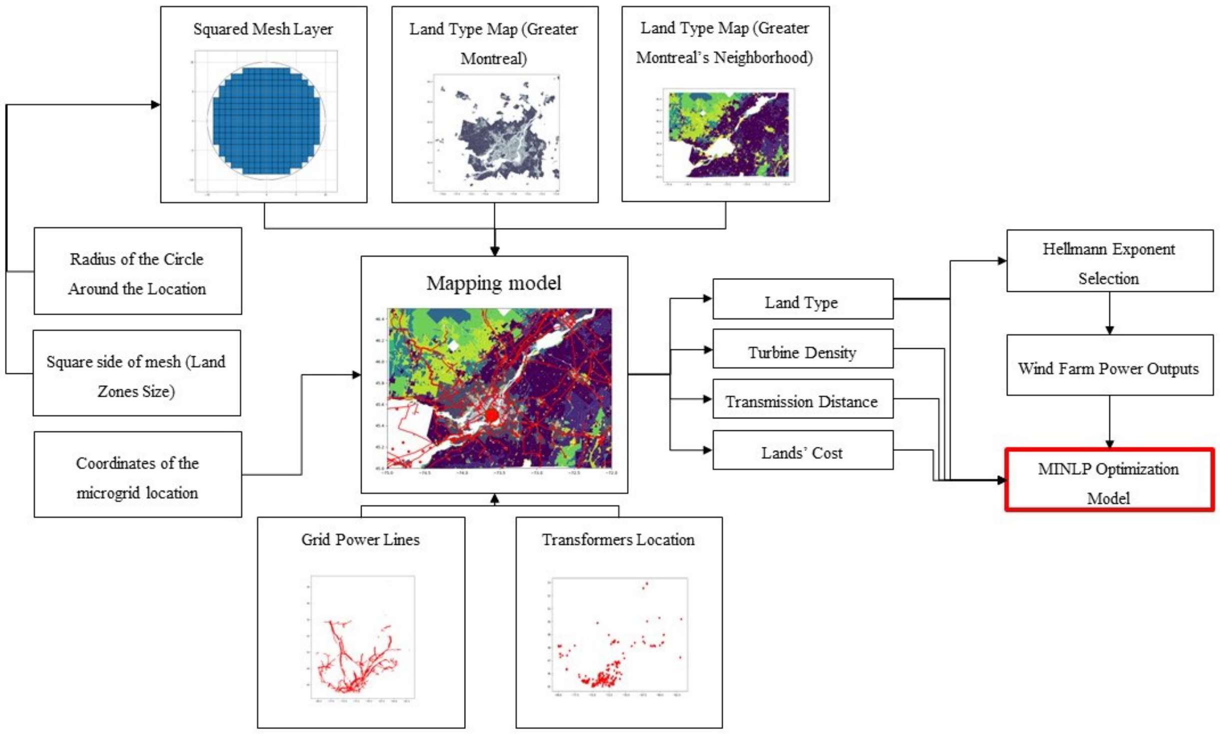

Squared Mesh Layer
Land Type Map (Greater Montreal)
Land Type Map (Greater Montreal's Neighborhood)
Radius of the Circle Around the Location
Square side of mesh (Land Zones Size)
Coordinates of the microgrid location
Mapping model
Land Type
Turbine Density
Transmission Distance
Lands' Cost
Hellmann Exponent Selection
Wind Farm Power Outputs
MINLP Optimization Model
Grid Power Lines
Transformers Location

Figure 3.3. Proposed mapping flowchart

3.4.3. Regional and Local Design Optimization Model

Although the structure of the optimization model when adding the regional generation is same as what was discussed for local design and it includes all the explained equations, there are some other terms which should be added to the developed objective function and constraints of the local design.

The below terms should be added to the discussed objective function:

- Equipment Cost: One of the term which should be added to the objective Function includes the wind turbines fixed cost (C_{RG}) and O&M cost ($C_{RO\&M}$), transmission initial cost (C_{Tr}) with its O&M cost ($C_{TO\&M}$), interconnection cost (C_{IC}) and land cost ($C_L(l)$) and could be calculated by equation (3.25). The transmission cost includes the CAPEX and OPEX of transmitting electricity to the nearest transformer station. The reason which the distance to the nearest transformer station is considered instead of nearest grid power line was decreasing the cost of creating a new transformer station for injecting electricity.

$$RG = N_{RG} \times C_{RG} \times \left(1 + \frac{C_{RO\&M}}{CRF}\right) + \sum_{l=1}^{L} (Dis(l) \times C_{Tr} + C_{IC}) \times \left(1 + \frac{C_{TO\&M}}{CRF}\right) + \sum_{l=1}^{L} C_L(l) \times A \times \beta(l) + HP_S \times C_{HP} \times \left[1 + \frac{C_{O\&MHP}}{CRF}\right] \tag{3.25}$$

Where N_{RG} is the number of wind turbine in the wind farm, L is defined based on number of zones created in the mapping model, A is the area that is required for the wind farm and $\beta(l)$ is the zone selection binary variable. The last term consists of the heat pump's initial capital (C_{HP}) and operation and maintenance cost $(C_{O\&MHP})$ that is the percentage of the capital cost and the selected size of the heat pump (HP_S). This term is only added in the heat pump scenario in the regional generation.

- Gas Consumption: The cost related to gas consumption for heat generation could be calculated by equation (3.26) :

$$GC = \sum_{t=1}^{T} \frac{G(t) \times G_P}{CRF} \tag{3.26}$$

Where $G(t)$ is the amount of gas consumption in each time step and G_P is the gas price per cubic meter.

- Gas Consumption Environmental Penalty: Based on Canada's fuel charge regulation made under the greenhouse gas pollution pricing (FCN10 Regulation) [80], the penalty value for natural gas consumption usage is calculated based on the below formulation:

$$GP = A \times B \times 0.8 \tag{3.27}$$

Where A is the gas consumption in cubic meters and B is equal to 0.0391.

The final objective function which includes both local and regional generation terms, is expressed with below equation:

$$\text{Min}\, OF = EC + BO + BM + GT + LS + RG + EN - GC - GP \tag{3.28}$$

Furthermore, the below constraints should be considered for the regional generation purpose:

- Power generation based on land type: In this thesis, seven land types were considered as the main land types of the city's neighborhood. Based on what was discussed in the Wind model section, regional wind power generation $P_{WR}(t)$ is calculated. Therefore, a conditional statement is added to the constraints to select a proper equation based on the land type selected by the model.

$$\begin{cases} if\ Land\ Type = i \\ P_{WR}(t,i) = ERwind(t,i) + ER_sWind(t,i) \end{cases} \tag{3.29}$$

Where $ERwind(t,i)$ and $ER_sWind(t,i)$ are used and surplus power of regional generation in land type i.

- Land selection: for regional generation purposes, the optimization model should select the optimum place for wind farm installation based on the data that was extracted from the mapping model. Therefore, a constraint is required to emphasize that only one land could be selected in each run.

$$\sum_{l=1}^{L} \beta(l) = 1 \tag{3.30}$$

- Wind turbine density: The number of turbines that could be installed outside of the microgrid (regional) should be limited based on the selected land type. This could also impact the result of the optimum selection of the land type since the zones with the ability of a low number of wind turbine installations (turbine density) could cause less regional generation capacity.

$$N_{RG} \leq \sum_{l=1}^{L} WTD(l) \times \beta(l) \tag{3.31}$$

- Regional design grid trade-off: This equation should be added to regional generation constraints for selling to the grid based on the selected land for regional generation ($P_{GS}(t,i)$).

$$\begin{cases} if\ Land\ Type = i \\ P_{sg}(t,i) \leq (ER_sWind(t,i)\ + E_sWind(t) + E_sPV(t))\ \times \vartheta \times \varphi \end{cases} \tag{3.32}$$

- Regional design energy balance: The equation that forces the equity between supply and demand in the regional generation design is called the balance equation, and it is expressed via equation 3.33 based on the selected land.

$$\begin{cases} if\ Land\ Type = i \\ P_{WR}(t,i) + P_W(t) + P_{PV}(t) + Pdis(t) + P_{GP}(t) = L_d(t) + Pch(t) + P_{GS}(t) \end{cases} \quad (3.33))$$

3.4.4. Economic Evaluation

To calculate the Levelized cost of energy (LCOE), the below equation is employed:

$$LCOE = \frac{NPC \times CRF}{\sum LD + \sum P_{sg}(t)} \tag{3.34}$$

Where LD is the total load demand served in each step.

The renewable penetration shows the renewable fraction of the total load demand determined with equation (3.35).

$$RP = \frac{Total\ Served\ Renewable\ Electricity}{\sum LD} \tag{3.35}$$

The payback period, which is another important economic indicator and shows the time period for recovering the initial investment, is evaluated by dividing the calculated total initial cost by the benefit. The total benefit includes the income from selling electricity to the central grid, removing environmental penalties, and saving from not using natural gas.

3.4.5. Heat generation and gas consumption

The transferred heat by gas boilers could be calculated based on measurement data using the below equation for each time step:

$$Q = mc\Delta T \tag{3.36}$$

Where Q is the total heat transferred in each time step, m is the flow rate, c is the water specific heat (4.187 kj/kg˚C) and ΔT is the difference between water supply and return temperatures.

Based on the calculated heat, the amount of gas consumption (V_G) could be evaluated using below equation (3.37):

$$V_G = \left[\frac{Q}{\text{CV} \times \tau \div \omega}\right] \times \frac{1}{\eta_G} \tag{3.37}$$

Where CV is the natural gas calorific value, τ is the correction factor (1.02264 in this study), ω is the conversion factor for kWh unit (equal to 3.6) and η_G is the gas boiler efficiency.

3.5. Optimum Operation Scheduling

3.5.1. Day ahead forecast methods

To remove the imprecision of the operation model, the first step is to forecast the day ahead load demand and renewable resources in an accurate manner. The proposed framework of the forecasting methods is depicted in Figure 3.4.

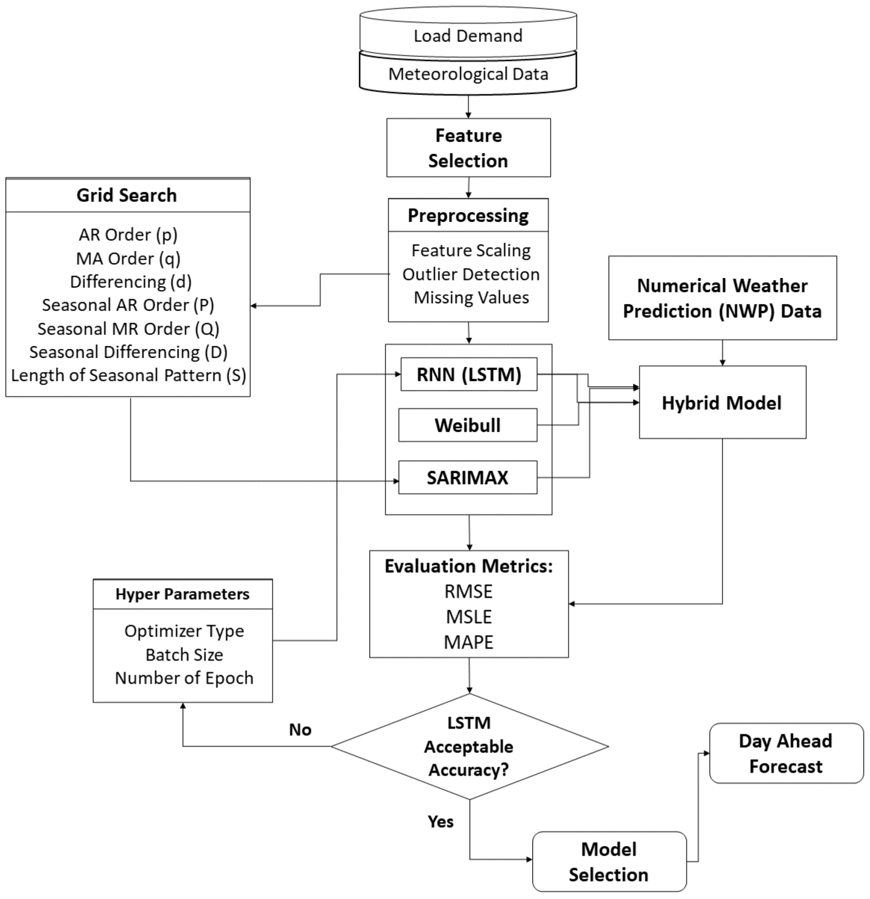


Figure 3.4. Proposed forecasting framework

This flowchart consists of three main sections. The feature selection and preprocessing of the data, including feature scaling, outlier detection, and dealing with missing values, is carried out in the first section. Furthermore, a grid search is employed to determine the optimum parameter for the statistical model in this section. In the second section, the output of the preprocessing stage is used for training the developed models. In this stage, a hybrid model is created using the Weibull distribution output as one of the input features of the LSTM model. Next, the NWP data is extracted from the NWP model and is used as input of the integrated LSTM-Weibull model for wind speed prediction and single LSTM model for solar irradiance prediction to create the final hybrid model. In the last section, the accuracy of each model is evaluated and compared to select the proper model. The hyperparameter optimization of the LSTM model is also conducted in the third section.

In the following sub-sections, each method is explained in detail.

3.5.1.1. SARIMAX

Seasonal autoregressive integrated moving average (SARIMA) with exogenous factors is called SARIMAX and is a statistical model used to predict time series with seasonality. The exogenous factors allow the model to use other features to reduce the forecasting error. With considering y_t as the wind speed in time step t, SARIMAX can be modeled as below [81]–[83]:

$$\varphi_p(B)\emptyset_P(B^s)(1-B)^d(1-B^s)^D y_t = \gamma_q(B)\delta_Q(B^s)\varepsilon_t \tag{3.38}$$

Where B is a lag operator that is responsible for back shifting, $\varphi_p(B)$ and $\emptyset_P(B^s)$ are non-seasonal and seasonal autoregressive operators of order p and P, respectively. $\gamma_q(B)$ and $\delta_Q(B^s)$ are non-seasonal and seasonal moving average function of order q and Q, respectively. $(1-B)^d$ and $(1-B^s)^D$ are non-seasonal and seasonal differencing operators. s is the seasonal length and ε_t is the residual error.

$$\varphi_p(B) = 1 - \varphi_1(B) - \varphi_2(B^2) - \cdots - \varphi_p(B^p) \tag{3.39}$$

$$\emptyset_p(B) = 1 - \varphi_1(B^s) - \varphi_2(B^{2s}) - \cdots - \varphi_P(B^{Ps}) \tag{3.40}$$

$$\gamma_q(B) = 1 - \gamma_1(B) - \gamma_2(B^2) - \cdots - \delta_q(B^q) \tag{3.41}$$

$$\delta_p(B) = 1 - \delta_1(B^s) - \delta_2(B^{2s}) - \cdots - \delta_P(B^{Qs}) \tag{3.42}$$

$p, d,$ and q are integer parameters to show the delay order of non-seasonal autoregressive, differencing, and moving average terms, respectively, while $P, D,$ and Q are integer parameters for indicating the delay order of seasonal autoregressive, differencing and moving average terms, respectively. An optimum set of these parameters could be specified for the model as input using different criteria for parameter selection such as Akaike information criterion (AIC), Bayesian information criterion (BIC), or Hannan–Quinn information criterion (HQIC) methods. Furthermore, the seasonal length of the model should be estimated using the decomposition of the training data. Afterward, the model is applied to forecast the future wind speed. The forecast horizon has a direct impact on the accuracy of prediction. An increase in the length of the horizon results in a reduction in accuracy [36].

3.5.1.2. LSTM

One of the deep neural networks that is suitable for time series data is the recurrent neural network (RNN) [84]. The feature that makes the RNN a proper algorithm for dealing with sequential data is the short-term memory due to the recurrent feedback connections [85]. But practically, RNN is not capable of treating data with long-term dependencies [86]. Therefore, LSTM as a specific type of RNN was developed to consider the issue of long-term dependencies [87], [88].

The LSTM is a type of RNN proposed by Hochreiter and Schmidhuber in 1997 to deal with long-term dependencies by upgrading the remembering capacity of a simple recurrent cell [89].

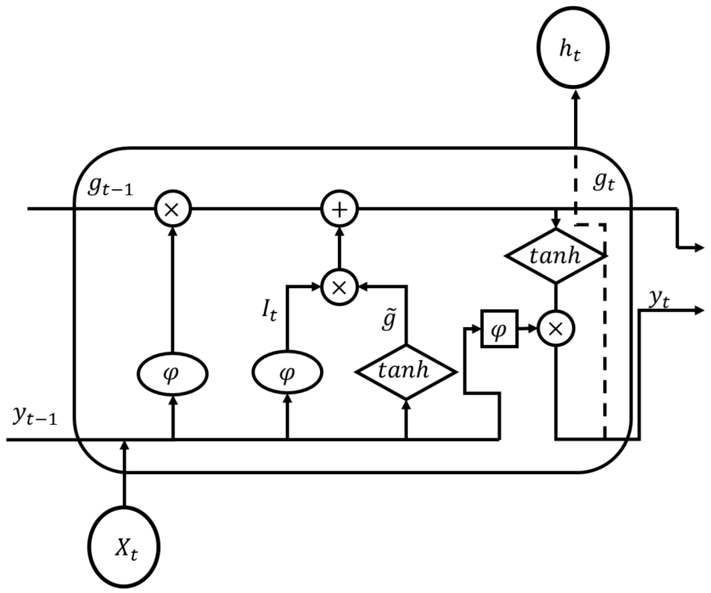


Figure 3.5. Schematic design of an LSTM module

An LSTM cell, unlike a simple RNN cell that includes a single tanh layer [32], is formed with several layers, as depicted in Figure 3.5. The first step is called forget layer, and it is responsible for deciding if the information could pass through the cell or it should be dismissed using an activation function. The activation function generally is a sigmoid function that, based on the input, generates a number between 0 and 1. While 1 shows the input of the cell can be added to the cell state, 0 designates that the input should be forgotten. (f_t) is the output of the forget layer and it is determined by the below equation [8]:

$$f_t = \varphi(w_f . [\, y_{t-1}, x_t] + b_f) \tag{3.43}$$

Where φ is the activation function, y_{t-1} is the output of the previous module, and x_t is input at time t and b_f and w_f are bias and weight, respectively.

In the second step, the update of new values ($\mathrm{I_t}$), using equation (3.44), and a vector of new information ($\tilde{g}$), as shown in equation (3.45), are created to add to the cell state by employing a sigmoid and tanh functions, respectively [2]:

$$I_t = \varphi(w_i . [y_{t-1}, x_t] + b_i) \tag{3.44}$$

$$\tilde{g} = tanh(w_S . [y_{t-1}, x_t] + b_S) \tag{3.45}$$

Subsequently, in the third step, a new cell state (g_t) is expressed as the sum of the previous cell state multiplied with the first step results and the multiplication of the I_t and $\tilde{g}$ shown in notational form as:

$$g_t = f_t \times g_{t-1} + I_t \times \tilde{g} \tag{3.46}$$

In the final step, by employing a sigmoid function, the cell decides what part of the cell state should be the cell's output and input to the next cell, and by using a tanh function, it regenerates the values between -1 and 1.

$$\partial_t = \varphi\, (w_\partial . [y_{t-1}, x_t] + b_\partial) \tag{3.47}$$

$$y_t = \partial_t \times \tanh(g_t) \tag{3.48}$$

Where, ∂_t is the portion of 'cell's state that is transmitted as the output.

The four explained layers in each cell of an LSTM model make it a proper algorithm to be tested for dealing with the unexpected behavior of the load demand and renewable generation.

3.5.1.3. Weibull distribution

Wind speed can be expressed in time series, and the variation of the speed can be described using a probability distribution function (PDF). For many years, the Weibull distribution has been used to fit wind speed data, and it is an explicitly proper fit to average wind speed data [90].

The Weibull PDF can be described with equation (3.49) [91]:

$$F(v) = \left(\frac{k}{c}\right)\left(\frac{v}{c}\right)^{k-1} exp[-\left(\frac{v}{c}\right)^{k-1}] \tag{3.49}$$

Where $F(v)$ is the probability of occurrence of wind speed (v), and k is the Weibull shape parameter that is calculated based on the standard deviation (σ) and the average ($\bar{v}$) of the wind speed data using equation (3.50) [92]:

$$k = \left(\frac{\sigma}{\bar{v}}\right)^{-1.086} \tag{3.50}$$

And c is the Weibull scale parameter that is given as [90]:

$$c = \frac{\bar{v}}{\Gamma\left(1 + \frac{1}{k}\right)} \tag{3.51}$$

Where Γ is the gamma function.

3.5.1.4. NWP Model

In this research, the NWP data was extracted from the NWP model that was developed by the "Centre for Solar Energy and Hydrogen Research (ZSW)" in Stuttgart, Germany.

3.5.1.5. Hybrid Model

In this study, the basic structure of the Hybrid model is the same as the LSTM model. To hybridize model X with the LSTM model, the results of model X will be scaled up along with other predictors and target variables and will be fed to the LSTM model as the input. The integration of Model X with the LSTM model results in an increase of the input dimension of the final neural network by one (in other words, it increases the number of input parameters by one).

3.5.1.6. Hyperparameter Optimization

In all hybrid forecasting models, LSTM is the core model, while other models are being hybridized with it. The tuned values for the LSTM model's hyperparameters, in both single and hybridized ways, are achieved through the grid search method. These hyperparameters could be different for each forecasting scenario.

3.5.1.7. Preprocessing and Evaluation Metrics

Due to the use of several predictor variables, such as humidity, temperature, and air pressure, along with wind speed as the dependent variable, feature scaling is necessary to eliminate the issues associated with dimensionality caused by a dissimilar range of values. The min-max scaler method is used to scale the data into a similar range:

$$X_N = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{3.52}$$

To find the outliers in the dataset based on an extreme outlier detection procedure, the minimum and maximum bounds were calculated based on equations 3.53 and 3.54, respectively.

$$\text{Min Bound} = Q1 - 3(IQR) \tag{3.53}$$

Where Q1 is the lower quartile that shows the number that is more than 25 percent of the data and IQR is the interquartile range.

$$\text{Max Bound} = Q3 + 3(IQR) \tag{3.54}$$

Where Q3 is the upper quartile that shows the number that is more than 75 percent of the data.

To assess the forecasting models' performance, root mean squared error (RMSE), mean absolute error (MAE), and mean squared logarithmic error (MSLE) are employed to determine the goodness of fit. RMSE evaluates the error using equation (3.55):

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\bar{y}_{\text{I}} - y_i)^2}{n}} \tag{3.55}$$

where y is the observed value and $\bar{y}$ is the predicted value.

And MAE is calculated using equation (3.56)

$$MAE = \frac{\sum_{i=1}^{n} |\bar{y}_{\mathrm{I}} - y_i|}{n} \tag{3.56}$$

Due to the wind speed as the target variable is distributed based on Weibull distribution and the difference between the minimum and maximum value in wind speed data is considerable, MSLE could be a proper metric to evaluate the error of a model, and it could be calculated using equation (3.57):

$$MSLE = \frac{1}{n} \sum_{i=1}^{n} (log(y_i + 1) - \log(\bar{y}_i + 1))^2 \tag{3.57}$$

3.5.2. Critical load and resiliency

Resilience is defined by the US department of homeland security as the ability to resist, absorb, recover from, or successfully adapt to adversity or a change in condition [93]. Typically, it is explained based on the performance of a system under specific time horizon and conditions. The system's robustness and the ability of the energy system to provide the critical load during a power outage are the two main aspects of resilience in this thesis.

The minimum load that needs to be supplied to the emergency and uninterruptable functions to ensure their performance is considered is critical load [94]. In this thesis, the critical load could be calculated in 4 main steps listed below:

Step 1: Defining the use types of the building. At a district scale, different buildings might have different use types, for instance, a residential building might have different use types compared to an institutional building. The level of importance of these use types and consequently the loads that should be assigned to them are different.

Step 2: After defining the use types, they are ranked based on their criticality level. Three levels of loads are defined based on [95]: 1) Non-essential loads, which can be disconnected from the power supply for noticeable periods of time, 2) Essential loads, which can suffer short de-energized; and 3) Uninterruptable loads, which need to be supplied without even a short disconnection. Similar to what was mentioned, in an institutional building, criticality levels could be expressed in three categories: low criticality level, such as gym; loads with significant criticality level, such as teaching labs; and loads with high criticality level, for instance, health clinic. In a time of disruption, the demand for this category cannot be disconnected.

In case of unavailability of the load demand for each category, the ratio of use type area to the total area could be considered as an alternative to evaluate the load of each category. This ratio could be calculated based on the below equation:

$$R_i = \frac{A_i}{A} \times 100 \tag{3.58}$$

Where R_i is the ratio of category I to the total area, A_i is the area of the category I (m^2), A is the total area of the building (m^2) and in a building with *n* different use type categories:

$$\sum_{i=1}^{n} R_i = 1 \tag{3.59}$$

Step 3: In this step, coefficients related to each use type to estimate the critical load are defined. The critical load in each building is a portion of the day-to-day demand. Therefore, the coefficients are a number between 0 and 1, and are defined separately for electricity demand and air conditioning systems as follows:

Step 3.1: Air conditioning critical coefficient (CA.)

For a building with a central air conditioning unit, the critical load of the air conditioning unit is considered to be impacted by the working hour and seasons (summer and winter) to simplify the process. Therefore, the critical coefficient is defined based on these two parameters and then multiplied by the actual hourly load of the air conditioning system. As shown in Table 3.4, the value of the coefficients is defined based on the different time ranges during summer and winter. C_{day} shows the critical coefficient of the air conditioning unit during the daytime while C_{Night} refer to the value of the critical coefficients in the nighttime in each season.

Table 3.4. Air conditioning critical coefficients

Coefficient	Time Range
Summer	
C_{day}	08:00 to 18:59
C_{night}	19:00 to 07:59
Winter	
C_{day}	08:00 to 20:59
C_{night}	21:00 to 07:59

Step 3.2: Electricity critical coefficient (CE.)

Electricity demand in use types varies, and a critical coefficient is defined based on each use type. Considering the importance ranking of use types in step 2, the loads with higher levels of importance have coefficients closer to one. Defining coefficients based on the working and closing states of the services is suggested to address the load curve pattern changes during the day and night. Table 3.5 represents the structure of estimating the coefficients where, $C_{Day,i}$ and $C_{Night,i}$ are critical coefficients of use type I during the day and night. The critical coefficient is defined based on professional judgment, which is directly correlated to the range of existing use type categories in the building.

Table 3.5. Electricity demand critical coefficient

	Coefficient	Time Range
Use type i	$C_{day,i}$	08:00 to 17:59
	$C_{night,i}$	18:00 to 07:59

Step 4: In the final step, the total critical load is evaluated. The building's air conditioning critical load (L_{CAir}) is estimated using equation (3.60):

$$L_{CAir}(t) \ = \ C_A \times L_d(t) \tag{3.60}$$

Where, $L_{CAir}(t)$, and $L_d(t)$ are air conditioning critical and actual load in time step t, respectively. C_A is the air conditioning critical coefficient that is divided into seasonally coefficients for day and night (table 3.5).

The hourly critical electricity load for each use type is calculated using equation 3.61, where $L_{CE}(t)$ and $L_d(t)$ are critical and actual electrical load demands in time step t, respectively, R_i is the ratio of category I to the total area (%) and C_E is the Critical coefficient which is further divided into the C_{eday} and C_{enight}, coefficients.

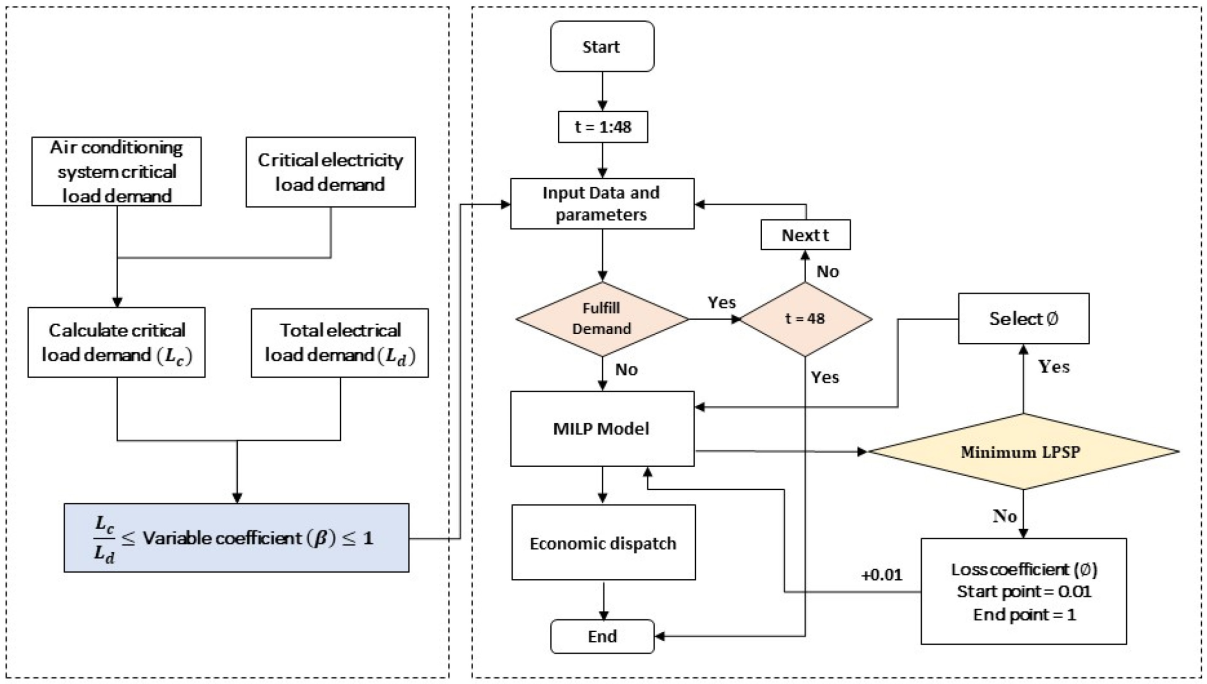
$$L_{CE}(t) = R_i \times C_E \times L_d(t) \tag{3.61}$$

The total critical electricity demand of the building (L_{CT}) is the summation of all the available use types' critical loads and is assessed using equation (3.62):

$$L_{CT} = \sum L_{CE} + L_{CAir} \tag{3.62}$$

3.5.2.1. Optimization model

The optimization model used for the economic dispatch (i.e., operation) of the energy system is explained in this section. Figure 3.6. shows a flowchart for the optimization module. The operation model works with an hourly resolution and a 48 h time horizon.



Air conditioning system critical load demand
Critical electricity load demand
Calculate critical load demand (L_c)
Total electrical load demand (L_d)
$\frac{L_c}{L_d} \leq$ Variable coefficient (β) ≤ 1
Start
t = 1:48
Input Data and parameters
Next t
No
Fulfill Demand
Yes
t = 48
Yes
No
Select Ø
Yes
MILP Model
Minimum LPSP
No
Economic dispatch
+0.01
Loss coefficient (Ø) Start point = 0.01 End point = 1
End

Figure 3.6. A schematic design of the optimization module

Objective function

The economic payoff of the energy system corresponds to its operational cost and its reliability. In this regard, depreciation of the energy storage (batteries), curtailment of renewable energy, and loss of power supply will be key factors in establishing a trade-off between the economy and the reliability of the energy system. These factors are formulated as follows:

- Battery depreciation:

There are several factors that could affect the depreciation of lithium-ion battery storage systems, such as depth of discharge, charge and discharge cycles, and temperature [16]. The impact of charge/discharge cycles is formulated as a cost indicator (equation 3.63) and added to the objective function.

$$DPC = \sum_{t=1}^{T} DPF \times Pch(t) + \sum_{t=1}^{T} DPF \times Pdis(t) \tag{3.63}$$

Where $Pch(t)$ and $Pdis(t)$ are the power of charging and discharging in the time period T, respectively, DPC is the depreciation cost of the battery, and DPF indicates the degradation factor expressed as below:

$$DPF = \frac{RC}{CP_b} \tag{3.64}$$

Where the total charge/discharge power capacity of the battery showed by CP_b while RC is the replacement cost of the battery.

- Renewable Curtailment

The main reason for the renewable curtailment could be a mismatch between the time of peak demand and the peak of the renewable generation [17]. To improve the economy of the energy system, the amount of excess renewable electricity needs to be minimized by operation management. Therefore, one of the main elements of the objective function is a penalty of the renewable curtailment that is calculated via the below equation (3.65):

$$RC = \sum_{t=1}^{T} \partial \times P_{wc}(t) + \sum_{t=1}^{T} \partial \times P_{PVc}(t) \tag{3.65}$$

Where ∂ is the curtailment factor (\$/kW) and $P_{wc}(t)$ is the curtailment of wind power while $P_{PVc}(t)$ shows the excess electricity of PV generation in period *T*.

- Loss of Power Supply

To minimize the loss of power supply probability (LPSP), a term needs to be added to the objective function to penalize the existence of the unmet load in each time step. Therefore, the loss penalty (LP) includes the summation of the amount of unmet load in each time step ($L(t)$) multiplied by the coefficient ($\emptyset$) as indicated in equation (3.66).

$$LP = \sum_{t=1}^{T} \emptyset \times L(t) \tag{3.66}$$

Then, the summation of the discussed terms creates the objective function (OF):

$$OF = DPC + RC + LP \tag{3.67}$$

Constraints

The operating cost of the system is minimized subject to the following constraints:

- Battery Capacity: The capacity of the battery storage system in the first time step (Eb[1]) is equal to the initial state of charge of the battery, while in the next time steps, the capacity of the battery is calculated based on the charge/discharge of it in that time step (equation 3.68):

$$\begin{cases} Eb(t) = SOC_{initial} \times C_T & t = 1 \\ Eb(t) = Eb(t-1) + Pch(t) * \eta_c - \dfrac{Pdis(t)}{\eta_d} & t = T \end{cases} \tag{3.68}$$

Where $SOC_{initial}$ is the initial state of charge of the battery (%), C_T is the total capacity of it (kWh) and η_c and η_d are charging/ discharging efficiencies, respectively. Furthermore, the state of charge of the battery in each time step ($SOC(t)$) should follow the below inequality:

$$SOC_{min} \leq SOC(t) \leq SOC_{max} \tag{3.69}$$

- Renewable generation: To balance the used renewable power in the microgrid and the surplus power, two constraints need to be added as follows for both wind power and PV power generation:

$$\begin{cases} P_W(t) = Ewind(t) + E_sWind(t) \\ P_{PV}(t) = EPV(t) + E_sPV(t) \end{cases} \quad (3.70)$$

Where $Ewind(t)$ and $EPV(t)$ are the used wind and PV power in each time steps, respectively while $E_sWind(t)$ and $E_sPV(t)$ are the surplus electricity of the wind and PV generation. $P_W(t)$ is the total power generated by the wind turbines in time step t and $P_{PV}(t)$ is the total power generated by the PV panels in time step t.

- Energy balance

The equation that shows the balance between the used renewable generation (supply) in the off-grid mode of the system and the demand (including the load demand and energy that requires to charge the battery is called the balance constraint, and it is shown by the equation 3.71:

$$Ewind(t) + EPV(t) + Pdis(t) + L(t) = L_d(t) + Pch(t) \quad (3.71)$$

Where $L_d(t)$ is the total load demand in time step t.

Optimal load

To calculate the load with the upper and lower bound of the actual load demand and critical load demand, the optimal load in each time step needs to be computed. Since the optimization model is responsible for finding the best amount of the load in each time step, a variable has to be defined and added to the balance constraint as below (equation 3.72):

$$Ewind(t) + EPV(t) + Pdis(t) + L(t) = \beta(t) \times L_d(t) + Pch(t) \quad (3.72)$$

Where $\beta(t)$ is the variable coefficients. The Upper bound (ub) of this variable is 1, while the lower bound (lb) could be defined based on the proportion of the critical load to the actual load demand.

$$\begin{cases} lb(t) = \dfrac{L_c(t)}{L_d(t)} \\ ub(t) = 1 \end{cases} \quad (3.73)$$

Where $L_c(t)$ is the critical load in time step t.

LPSP

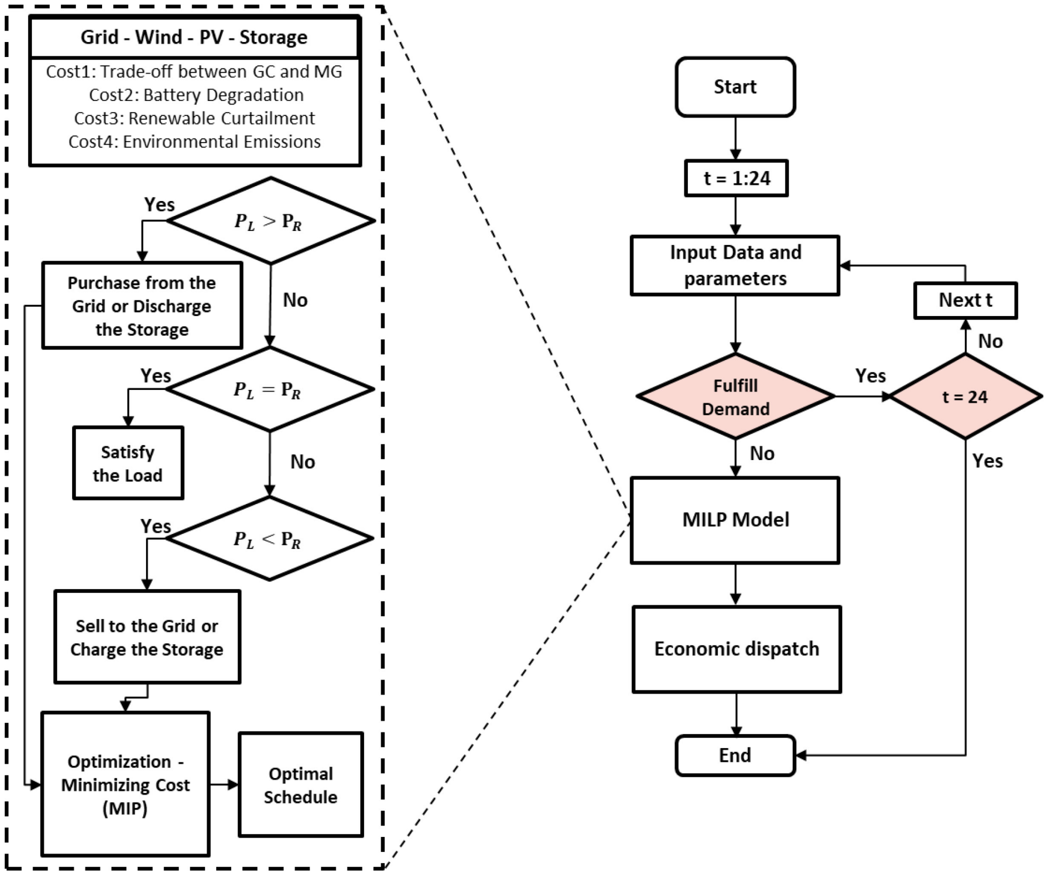
In this study, loss of power supply probability (LPSP) is employed to evaluate and compare the reliability of the energy system in the different scenarios. LPSP could be expressed via the below equation (3.74) [18]:

$$LPSP = \frac{\sum_{t=1}^{T} L(t)}{\sum_{t=1}^{T} L_d(t)} \tag{3.74}$$

Where C_{Rep} is replacement cost, E_{max} is the maximum storage capacity and N_{cycle} lifetime cycles.

3.5.3. Grid-Connected Optimal Operation

In this sub-section, the related methodology for reaching the optimal schedule of the microgrid in grid-connected mode is explained. The overall methodology is shown in Figure 3.7. Based on this figure, the day ahead optimal operation will be predicted by a MILP model to fulfill the load demand in each iteration. The objective function of the optimization model includes 4 terms, 1) the trade-off between grid and microgrid, 2) battery depreciation cost by charging and discharging in each time step, 3) Renewable power curtailment cost, and 4) Environmental emissions in case central grid uses the non-renewable resources.



Grid - Wind - PV - Storage
Cost1: Trade-off between GC and MG
Cost2: Battery Degradation
Cost3: Renewable Curtailment
Cost4: Environmental Emissions
Yes
$P_L > P_R$
No
Purchase from the Grid or Discharge the Storage
Yes
$P_L = P_R$
No
Satisfy the Load
Yes
$P_L < P_R$
Sell to the Grid or Charge the Storage
Optimization - Minimizing Cost (MIP)
Optimal Schedule
Start
t = 1:24
Input Data and parameters
Next t
No
Fulfill Demand
Yes
t = 24
No
Yes
MILP Model
Economic dispatch
End

Figure 3.7. Proposed framework for grid-connected operation

Finally, the optimum operation for the next day will be generated by the model and the battery schedule will be defined based on that. The basic terms of the optimization model is the same as what has been explained in the previous optimization sub-section, except for the grid trade-off that is not evaluated in the off-grid mode. Therefore, the economy of trading between the urban microgrid and the conventional grid is defined as follows:

$$GC = \sum_{t=1}^{T} P_p(t) \times EP - \sum_{t=1}^{T} P_s(t) \times ES \tag{3.75}$$

Where, GC is the amount of trading between CG and MG, $P_p(t)$ and $P_s(t)$ are the amounts of purchased and sold electricity from/to the CG at time *t*, respectively, EP is the purchasing price of electricity from CG and ES is the selling price of electricity to CG.

Chapter 4: Case Study

4.1. Building's General Information

The so-called EV building that houses Concordia University's engineering school is located in the downtown Sir George Williams Campus in Montreal (Quebec) and is considered the case study of this research. Therefore, Concordia university is considered the decision maker and owner of the distributed energy system which is going to be designed in this thesis. However, the proposed framework in the previous chapter is flexible for changing the ownership to utility or cooperation between utility and Concordia University.

EV building is one of the largest buildings in downtown Montreal with high electricity consumption. It consists of four main sectors: the engineering and computer science (ENCS) department, the visual arts department, the 17th floor (that includes labs and the mechanical room and air conditioning unit), and the electric boilers for heating purposes. The annual consumption of this building was about 20 million kWh in 2019, measured with four separate electrical metering devices in 15 minutes resolution. The data is available from 2015 to 2019. Dataset attributes are shown in Table 4.1.

Table 4.1. User categories and their floor areas in EV Building

Date	Elec Boiler (kWh)	17th Flr (kWh)	ENCS (kWh)	VA (kWh)	Total EV (kWh)
2015-01-01 00:00:00	662.565	1363.5	387.5	357.25	2770.815
2015-01-01 01:00:00	664.77	1427.25	406.75	357.75	2856.52
2015-01-01 02:00:00	660.63	1416.25	388	339	2803.88
2015-01-01 03:00:00	657.6325	1413.5	416.5	341.25	2828.883
2015-01-01 04:00:00	658.165	1424	389.5	337	2808.665

Moreover, Figure 4.1, shows the last three years of the available data to explore the power consumption yearly trend.

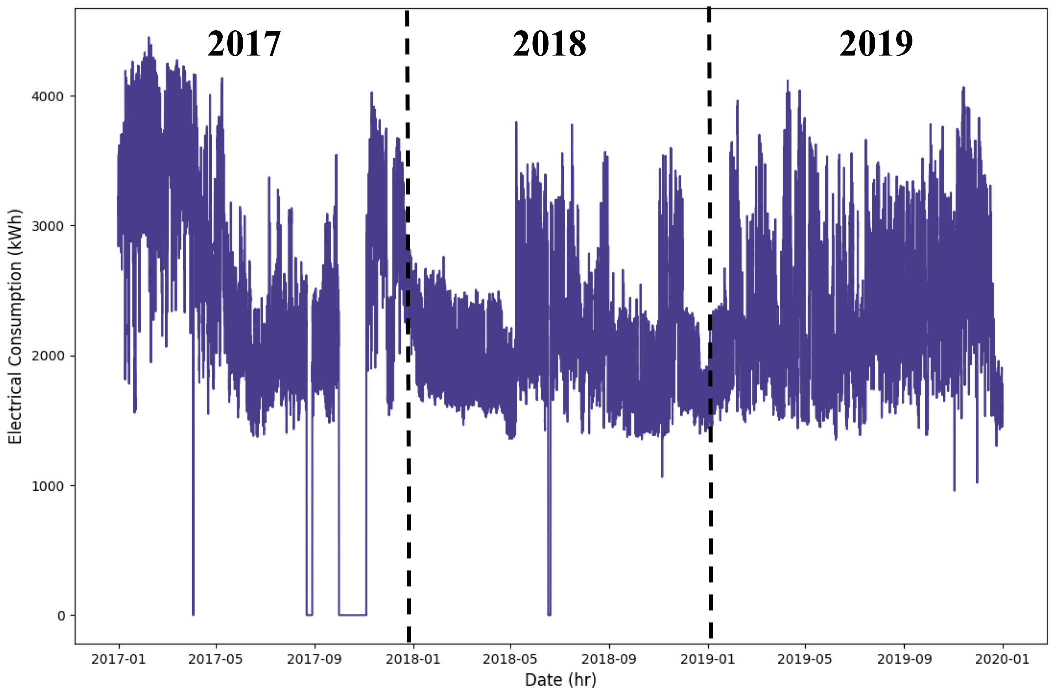


Figure 4.1. Last three years, EV building's power consumption with 15 minutes resolution

As is evident in Figure 4.1, the EV building's electricity consumption does not follow the same pattern as a typical building (lower electricity consumption in summer and higher in winter). This could be justified by two reasons. First, an institutional building could behave differently compared to other typical buildings. Second, the more important cause is that the energy source in the EV building is not only electricity, and the University's facility management is using natural gas also to provide heat, especially in winter.

However, based on Figure 4.2, the zoomed-in daily consumption trend seems reasonable for each day of the week. Evident lower consumption occurs on the weekend for the departments with low consumption at midnight and higher consumption during working hours for all sectors.

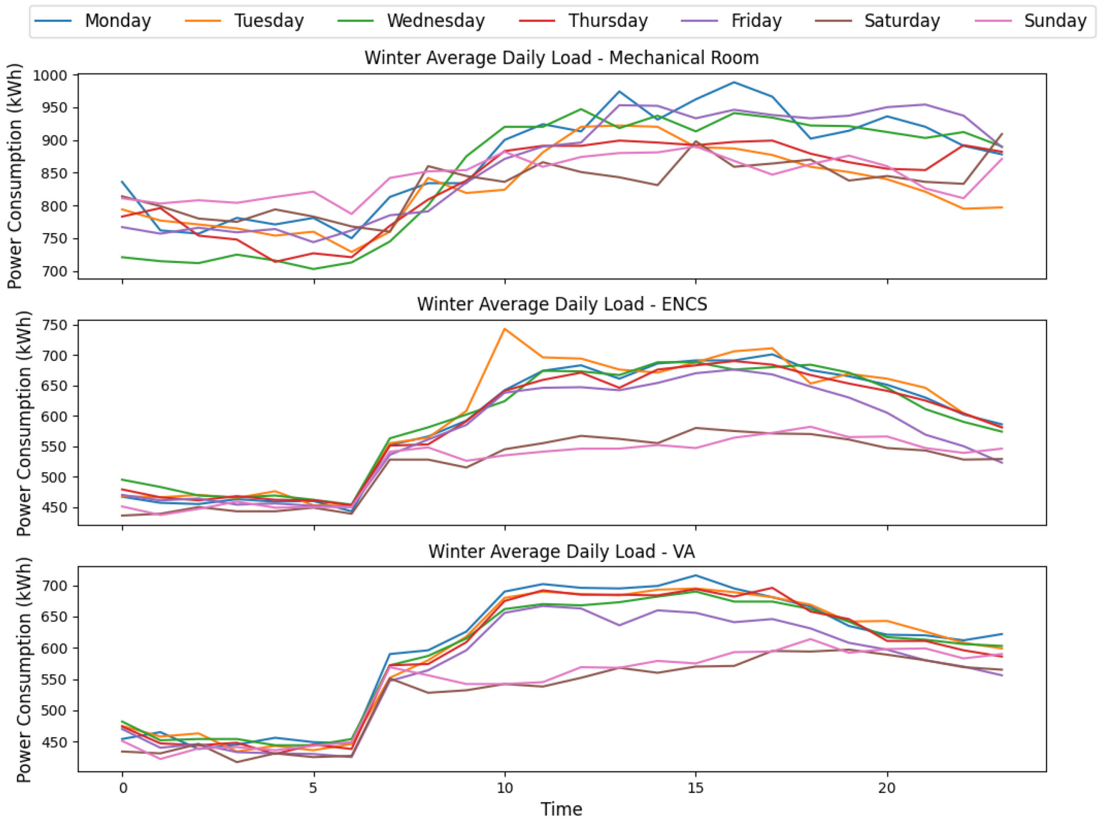


Figure 4.2. Daily consumption of three main sections of EV building for each day of a week

The other important information that needs to be considered in the calculation is the available area of the building. The roof area of a building is the main zone used for installing the components. Concordia University's available roof area for its main buildings on the downtown campus was extracted from the CityGML 3D data model of Montréal, as listed in Table 4.2.

Table 4.2. Concordia University's available roof area

Building Name	Available Roof Area (m^2)
EV	5,790
John Molson	2,457
GM	1,598
Library	5,939
H Building	5,074
FB	3,629
Total	**24,487**

Based on assumptions and the shape of roof areas, 95 percent of Concordia University's available roof area is considered usable for installing renewable energy components. Therefore, 23,262 m^2 is the available area for local generation.

4.2. Location's General Information

Montreal is the second-most populous city in Canada. It is located in the Southern part of the province of Quebec with latitude and longitude coordinates of 45 N and -73 E degrees, respectively [96]. Montreal's hourly resolved data of temperature and humidity is obtained from the NASA prediction of worldwide energy resources website [97]. Giving the exact location of the EV building (45.4955, -73.5782) to NASA portal generates all the required climatic data. Figure 4.3. shows the variation and trend of these independent variables for the year 2020 in hourly resolution. Although the ascending and descending trend from the beginning to the end of the year is noticeable for temperature, no notable trend is detected in relative humidity. However, a lower fluctuation range at the beginning of the year (Winter) compared to the middle of the year (Summer) is perceptible for relative humidity.

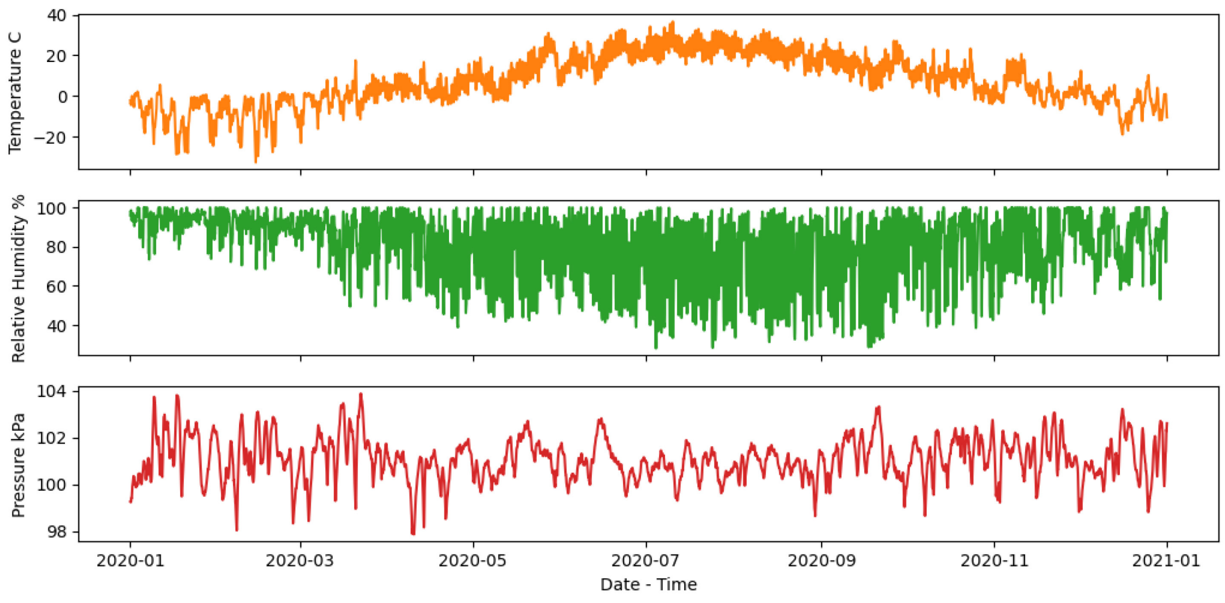


Figure 4.3. Montreal 2020's climatic information

Moreover, the wind speed (m/s) in 50-meter height (based on the EV building height) is also extracted from the NASA data portal for the same year (2020) [97] in hourly resolution. To evaluate the behavior of the target variable further, the additive decomposition of the wind speed data is plotted (Figure 4.4). As it is demonstrated in Figure 4.4, although there is no notable trend, the seasonality graph shows daily fluctuations (ascending and then descending during the day). However, as the residual graph that shows the error of fitting this seasonality on the real wind speed data is noticeable, this seasonality can be seen as not being strong.

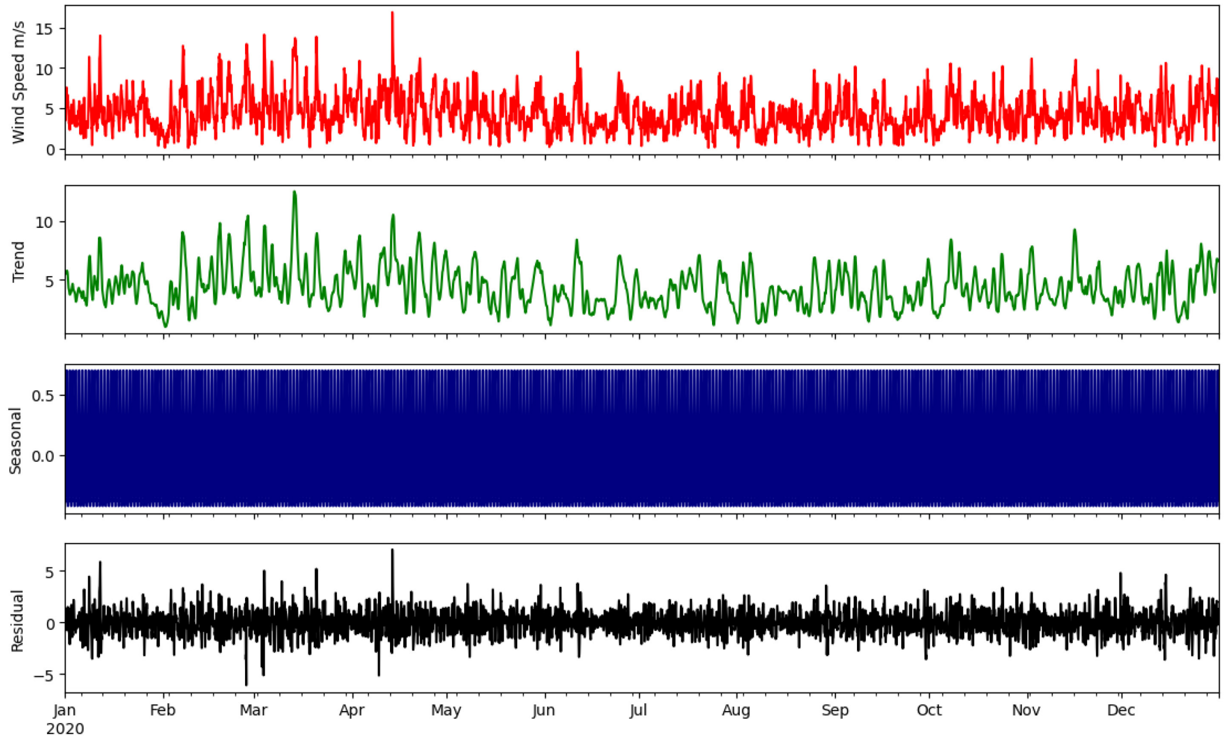


Figure 4.4. Decomposition graph of wind speed data

The same strategy is used after collecting shortwave downward solar irradiance (W/m^2) from the same portal [97]. The decomposition graph shows a clear ascending and descending trend from the beginning of the year till the end, showing the higher available irradiance in summer and lower in winter. This difference is considerable at some points. The seasonality follows an evident pattern while the residual is still significant.

4.3. Critical Load

Based on the explained methodology, different use types in the EV building with corresponding floor areas are listed in Table 4.3 [98] to calculate the critical load. The assumed critical coefficients related to electricity demand are shown in C_{eday} and C_{enight} columns.

Table 4.3. User categories and their floor areas in EV Building

Use Type	**Floor Area** (A_i)	**Ratio of area to total area** (R_i)	C_{eday}	C_{enight}
Health Center	3427.8	4.1%	1	0.2
computer services	134.6	0.2%	1	1
research labs	7957.9	9.5%	1	1
maintenance services	277	0.3%	1	1

lavatories	1298	1.5%	1	0.5
offices	10597.3	12.6%	1	1
food services	568.2	0.7%	1	0
offices	5298.6	6.3%	1	0
indoor parking	1647	2.0%	1	0
teaching labs	7328.3	8.7%	1	0
classrooms	1336.8	1.6%	1	0
community services	5413	6.5%	0.2	0
gym	3427.8	4.1%	0.2	0
common areas	24583.9	29.3%	0.2	0.2
museum	237	0.3%	0.2	0
housekeeping	385	0.5%	0.2	0
others	9896.6	11.8%	0.2	0
Total area (A)	**83814.9**	**100.0%**		

The EV building has central heating and cooling systems, and the critical coefficient regarding air-conditioning (CA) values is assumed, as presented in Table 4.4. The coefficients and the time range for day and night are selected based on the working hours of an educational building. It is assumed that in the case of a power outage caused by natural hazards, the air conditioning will be set to a state that meets minimum needs with respect to ventilation and comfort temperature for critical uses such as health centers or research labs inside the building. These coefficients change based on the season since working hours and space heating and cooling demands differ for summer and winter. These coefficients could also be different from one building to another (even for the same building types) due to the possibility of having different use types with varied characteristics of areas in each building.

Table 4.4. Critical coefficient values for summer and winter

Coefficient	**Value**	**Time Range**
	Summer	
C_{day}	0.4	08:00 to 18:59
C_{night}	0.3	19:00 to 07:59
	Winter	
C_{day}	0.6	08:00 to 20:59
C_{night}	0.2	21:00 to 07:59

4.4. Electricity Tariff

Canada has not a unique electricity tariff, and it is different for each province. In Quebec, Hydro Quebec (HQ) is responsible for defining different electricity rates. These rates are mainly defined based on user types. For example, the rate for a residential building is different from the rate of a commercial one. Since Concordia University is considered a large consumer, HQ has a separate contract with it which is based on the consumption rate. Table 4.5 shows the rate for the year 2020, which is used as the electricity tariff in this research.

Table 4.5. HQ electricity purchase rate for Concordia University – The year 2020

Month	Electricity Rate (CAD/kWh)
January	0.07005544
February	0.06775921
March	0.06770977
April	0.06393077
May	0.06694972
June	0.06953881
July	0.06373102
August	0.06239592
September	0.05969309
October	0.05936981
November	0.061508168
December	0.065685064

Although the mentioned rate in Table 4.5 seems flexible, the price fluctuations each month are not considerable. This is more evident by looking at Figure 4.5.

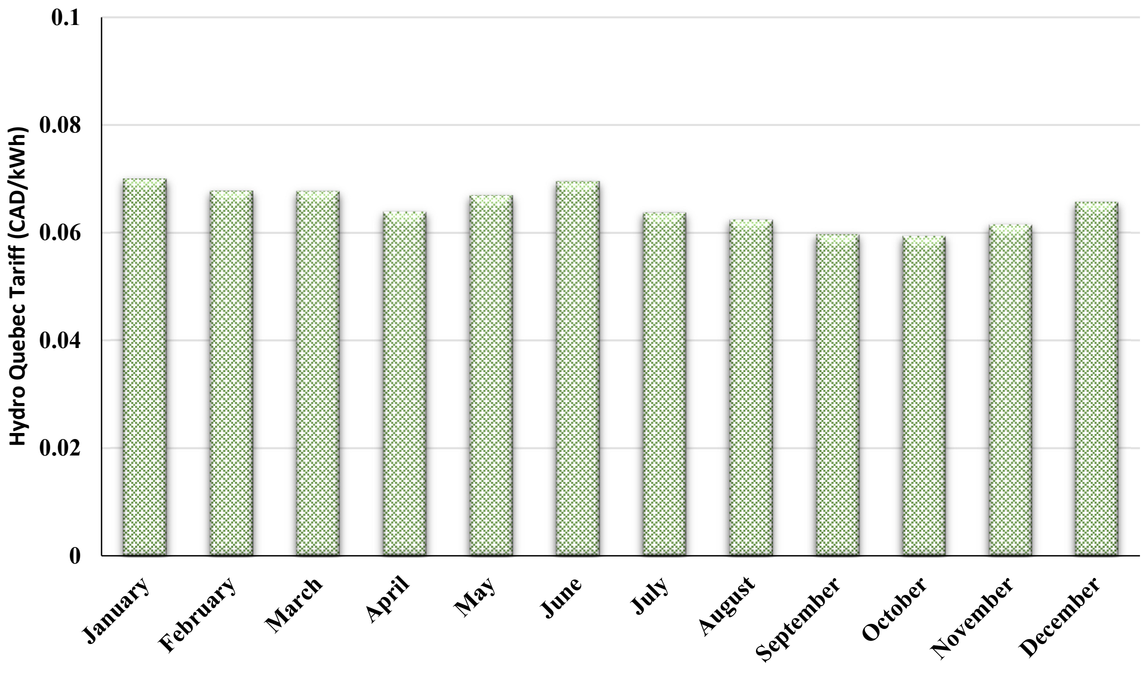


Figure 4.5. Monthly HQ rate for Concordia University

Based on HQ regulations [99], they are not allowed to buy electricity from the users directly and without calling for tenders. However, the self-generators still could inject their excess power into the grid with an exchange of credits in kilowatt-hours. In this research, the credit is considered as a cent/kWh, and Table 4.6 shows the assumed rate for this credit. In this research, the credit is considered as the cent/kWh, and Table 4.6 shows the assumed rate for this credit. This rate is considered flexible since it is based on the time of the year (Winter or Summer), and it is derived from the Rate flex G [100], which is considered a flexible rate.

Table 4.6. Assumed purchase rate

	Summer (April 1 – Nov. 30)	**Winter (Dec.1 – March 31)**
Proposed Flex Daily Rate	0.1 CAD/kWh	0.2 CAD/kWh

Chapter 5: Results and Discussion

5.1. Implementation and Utilized Tools

All modules and sub-modules are written with Python Programming Language using the below libraries for each module:

- PV Model: The “pvlib” library [101] used for creating the PV arrays and power output calculations
- Wind Model: For both a single wind turbine and a cluster of wind turbines (wind farm) power output, the “windpowerlib” library [102] is employed.
- Design Model: The mixed integer nonlinear programming model is developed in the Pyomo platform [103] using the SCIP solver [104].
- Operation Model: developed in the Pyomo platform [103] using CPLEX solver [105]
- Forecasting Model: developed in Python using several libraries. Keras library [106] with Tensorflow backend used for developing LSTM model. The SARIMAX model was developed using “statsmodels” library [107]
- Both the SCIP and CPLEX solvers were used by connecting Python with GAMS [108] to use GMAS options for adding attributes to the solvers.

5.2. PV Power Output

Using solar irradiance data, which was explained in section 3.2, and the input parameters of the selected PV modules, which are shown in Table 5.1. the output of the PV system will be calculated. Based on Table 5.1, the 220W solar panels selected with Glass/Cell/Glass and “Open Rack” module types and mount method, respectively. Moreover, the surface azimuth is set to 180 since Montreal is located in the northern hemisphere; the best direction of the panels should be toward the south.

Table 5.1. PV modules information

Module Parameter	Type/Value
Manufacturer	Canadian Solar
Model	CS5P_220M_2009
Type	Glass-Cell-Glass
Mount Type	Open Rack

Inverter Manufacturer	ABB
Inverter Model	MICRO_0_25_I_OUTD_US_208__208V
Surface Azimuth	180
Surface tilt	45

The Inverter's model is selected in a way to track the maximum power point (MPPT). Therefore, the final AC output of the inverter is shown in Figure 5.1. In each time step, the maximum power output in the form of DC power is calculated. Then based on the inverter efficiency, it will be converted to AC electricity. The PV power output will be fed to the design module for the local generation.

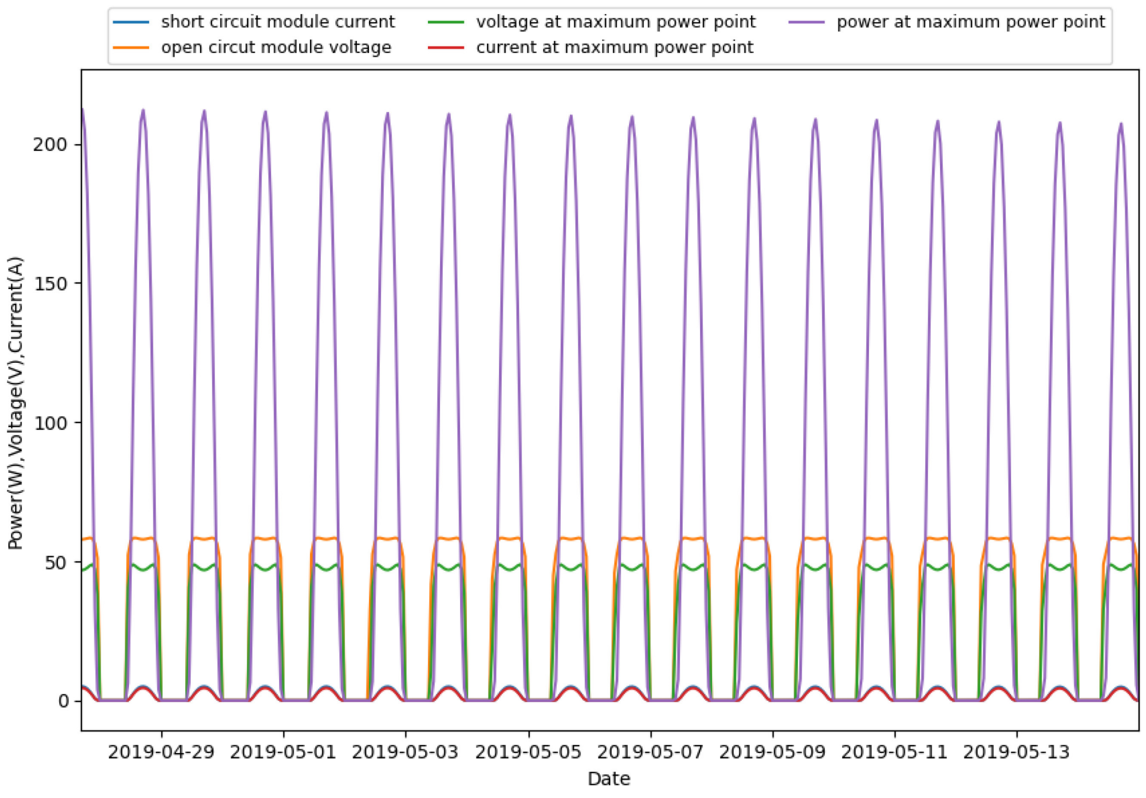


Figure 5.1. DC power output of a PV module

5.3. Wind Power Output

Several parameters and sub-models in the wind power model are defined as input by the user. There are two main models for wind power output: the first model is for single wind turbine power output and the second is for wind farm power output calculation. Table 5.2 shows the sub-models and related parameters of a single wind turbine.

Table 5.2. Single wind turbine sub-models and parameters

Sub-Model/Parameters	**Type/Value**
Wind Speed Model	Hellmann
Density Model	Ideal gas
Temperature Model	Linear gradient
Power Output Model	Power Coefficient Curve
Obstacle Height	0

In Table 5.3. the selected sub-models and parameters of a cluster of turbines (Wind farm) are shown.

Table 5.3. Wind farm sub-models and parameters

Sub-Model/Parameters	**Type/Value**
Wind Speed Model	Hellmann
Density Model	Ideal gas
Temperature Model	Linear gradient
Wake Loss Model	Wind farm efficiency
Obstacle Height	0
Standard Deviation Method	Staffell_Pfenninger
Smoothing Order	Wind farm power curve

Although the obstacle height is equal to zero in both single turbine and wind farm models, using different Hellman friction factors could simulate different geographical situations where the turbines are installed. Since the single turbines should be installed on the roof of the building in Montreal's downtown area, the Hellmann exponent is equal to 0.3. While for the wind farm model, this could be different based on the land type selected for the turbine cluster installation. The assumed Hellmann exponents for the wind farm model are reported in Table 5.4.

Table 5.4. Assumed Hellmann friction factor coefficients

Land Type (Greater Montreal)	**Hellmann Exponent (α)**	**Land Type (Outside of Greater Montreal)**	**Hellmann Exponent (α)**
Waterbody	0.10	Agricultural	0.10
Open Area	0.10	Forest	0.15
Resource and Industrial	0.20	Industrial	0.20
Parks and Recreational	0.25	Public and Recreational	0.25
Commercial	0.30	Residential	0.30
Residential	0.40	Urban Areas	0.40

Since the wind farm could be installed in both greater Montreal (outside of the populated areas) and outside of greater Montreal, the land types are different. Moreover, similar land types inside and outside greater Montreal could have different Hellmann Exponents. For instance, buildings in residential areas are denser, while outside of Greater Montreal could be scattered. However, in the case of using the developed tool for other locations, the Hellmann exponents should be amended accordingly.

In this research, the specification of a single wind turbine is reported in Table 5.5 [109]. The selection of the wind turbine is based on the wind turbine's power curve and the maximum, minimum, and average wind speed of the case study. The EO-25 wind turbine is selected since its cut-out, and cut-in wind speeds are close to the maximum and minimum wind speeds in Montreal.

Table 5.5. Single wind turbine specifications

Wind Turbine	**Type/Value**
Manufacturer	eocycle
Model	EO-25
Rated Power	25 kW
Cut in Wind Speed	3.1 m/s
Cut out Wind Speed	25 m/s
Blade Length	6 m
Number of Blades	3
Area	125 m^2
Hub Height	16m

Based on the above specification, the output of a single wind turbine is shown in Figure 5.2 for year 2019.

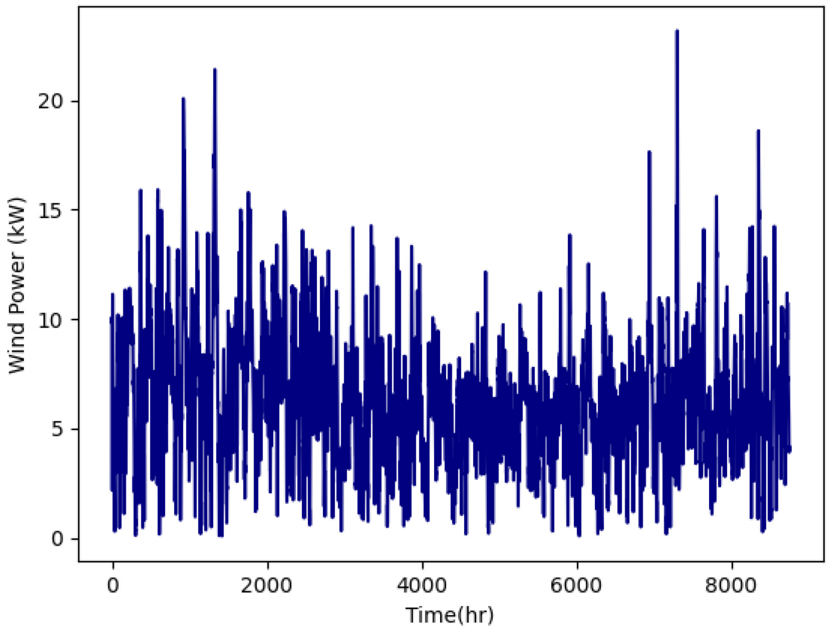


Figure 5.2. Single 25kW wind turbine power output

As mentioned in section 3, the wind speed data for single wind turbine power output is collected for a 50-meter height which considers the height of the EV building and the hub height of the wind turbine.

For wind farm design, a large-scale wind turbine is selected, and the specification is reported in Table 5.6. [110].

Table 5.6. Wind farm wind turbines specifications

Wind Turbine	**Type/Value**
Manufacturer	Enercon
Model	E-53
Rated Power	800 kW
Cut in Wind Speed	3 m/s
Cut out Wind Speed	34 m/s
Rotor Diameter	52 m

Number of Blades	3
Area	2198 m^2
Hub Height	73 m

Since the number of wind turbines in the wind farm depends on several indicators of the land that will be selected by the optimization model, in this stage, the electrical output of one large-scale wind turbine in the middle of the wind farm (considering the wake effect of the front turbine) is displayed in Figure 5.3. Furthermore, the Hellmann friction factor is considered equal to 0.1 (open area). This figure only indicates how the wind farm model works.

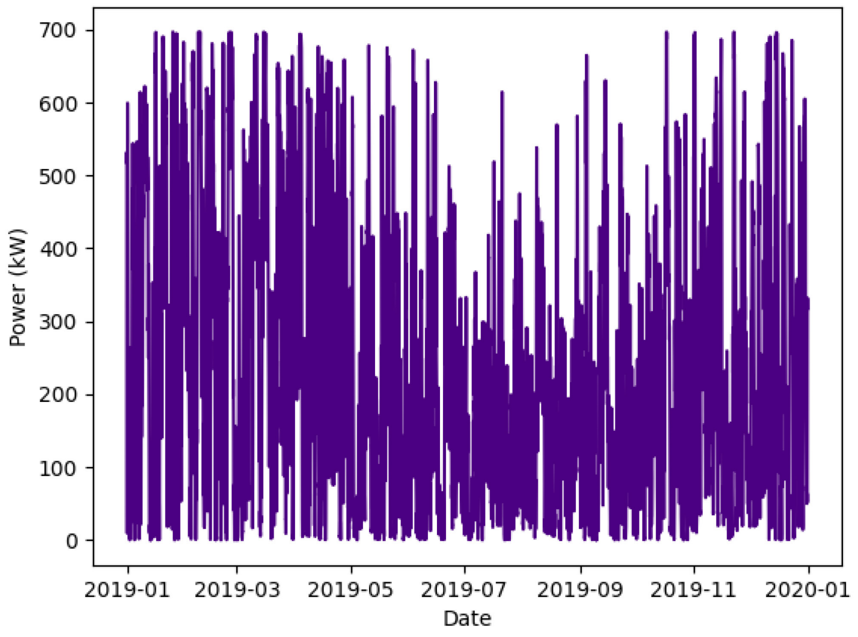


Figure 5.3. The power output of one of the turbines located in the middle of the wind farm

5.4. Local Generation Design

In the local generation, the output of the PV and single wind turbine models are used as input data along with the power consumption. Single wind turbine power output is employed because of the scattered installation of wind turbines on the roof of the building, and since the wind turbines are not supposed to be near each other, they could be considered standalone turbines.

This module of the tool was developed in a way to receive some inputs before designing the energy system. Based on the case study, these inputs could be set as default or defined by the user. In this research, the input parameters are described in Table 5.7 and 5.8.

Table 5.7. Input costs parameters

Component	Value
PV Cost (CAD/kW) [111]	1230
PV O&M Cost (% of Capital Cost) [112]	1
Wind Turbine Cost (CAD/kW) [111]	2046
Wind Turbine O&M Cost (% of Capital Cost) [111]	2
Lithium-Ion Battery Cost (CAD/kWh)	660
Battery Replacement Cost (CAD/kWh)	660
Battery Maintenance Cost (CAD/kWh-year) [112]	12.9
Battery Charge/Discharge Operating Cost (CAD/kWh) [112]	0.00053
Converter/Inverters Cost (CAD/kW)	100

Table 5.7 shows the initial and operational costs of different components of the system, while Table 5.8 reports the battery's input parameters.

Table 5.8. battery's input parameters

Parameter	Value
Battery maximum charge/discharge ratio	35
Minimum SOC of the batteries (%)	20
Portion of generated electricity that could be sold to grid (%)	100
Discharging efficiency (%) [113]	95
Charging efficiency (%) [113]	95

The other important parameter that should be specified based on the case study is the maximum amount of LPSP. In this research, the maximum LPSP should be less than 0.1 percent (0.001). Furthermore, the loss penalty coefficient (discussed in the methodology section) is specified in a flexible manner based on the amount of electricity that is used in each time step. Table 5.9 shows the penalty coefficient based on the consumption value in each time step.

Table 5.9. Flexible loss of power supply penalty coefficient

Scenario	φ **- $/kW**
Power Consumption < 40,000	1
40,000≤Power Consumption ≤50,000	2
50,000≤Power Consumption ≤60,000	3
60,000≤Power Consumption ≤70,000	4
70,000≤Power Consumption ≤80,000	5
80,000 < Power Consumption	6

Using the input parameters, the optimization model results in the optimum configuration and size of each component, considering the economic and reliability aspects. These results are reported in Table 5.10.

Table 5.10. Optimum configuration and economic report of local energy system

Parameter	**Value**
NPC (CAD)	39,634,169
LCOE (CAD/kWh)	0.12144
Initial Capital Cost (CAD)	17,137,929
PV Capacity (kW)	3877

Wind Turbine (kW)	1200
Battery Capacity (kWh)	10763

To evaluate the amount of generation of each components in different time steps, a scheduling graph is shown in Figure 5.4.

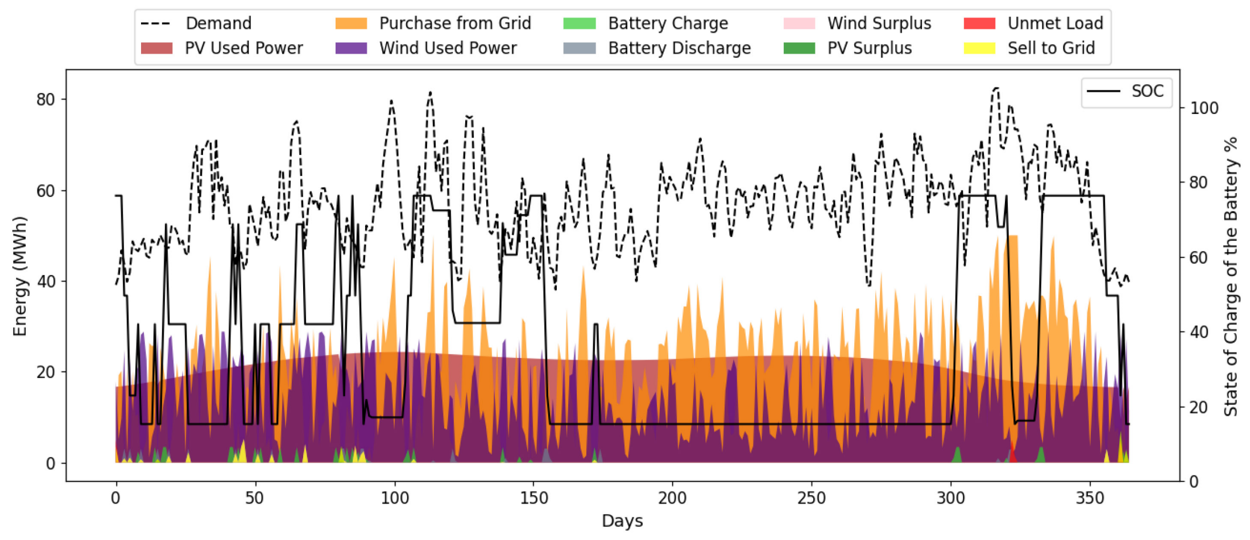


Figure 5.4. The annual schedule of the optimum design for local power generation

Based on this figure, the amount of unmet load (loss of power) is negligible during the year, which means the flex scenario for loss penalty coefficient along with considering a constraint for LPSP, could be functional. However, the amount of purchase from the grid (HQ in this research) is still considered excessive compared to renewable generation. To assess this issue further, the amount of renewable penetration along with some other information about the system, is given in Table 5.11.

Table 5.11. Designed local energy system information

Parameter	**Value**
Renewable Penetration (%)	62
Unmet Load	12 days (15219 kWh)

Purchase from grid (kWh)	7,811,121
Sell to Grid (kWh)	63,764

Based on Table 5.11., the amount of renewable penetration for the design of local microgrid is 62 percent. Since the maximum available area is used for installing renewable components, this is the maximum renewable penetration that could be achieved. The amount of purchase from the grid is about 7.8 MWh, and since the summation of power consumption in the building is about 20.8 MWh, then about 13 MWh could be covered by renewable generation, which could lead to an annual benefit for the University. Since about 95 percent of generated electricity in Quebec comes from renewable sources such as hydro or wind [114], no penalty is considered for purchasing from the grid. However, based on the selected case study, the user can specify the portion of nonrenewable electricity in the central grid and define a penalty for it accordingly.

In Table 5.8, the battery's parameters are specified. Although most of these parameters could be set based on the literature as default (or they could be regulated by the user), the maximum charge/discharge ratio is the one that needs to be studied more to find the optimum value. Therefore, a sensitivity analysis is carried out on this parameter by changing its value and analyzing the impact on the levelized cost of energy. The other important factor that needs to be evaluated is the impact of changing the maximum charge/discharge on the battery's operating cost. The result of the sensitivity is shown in Figure 5.5. It is evident that by raising the charge/discharge rate, the cost of energy reduced; however, this reduction is not sensible after 0.3. On the other hand, the battery operation cost has an ascending and descending order with increasing the maximum charge\Discharge rate. Although the minimum battery operation cost happens when it intersects the LCOE graph, at some point, the operation cost reduction happens because of the decrease in storage capacity. Therefore, the optimum maximum charge\discharge rate could be between 0.3 and 0.4, that LCOE is not changing anymore, and battery operation is not at maximum level. In this research, the average of these two values is considered as the maximum charge\discharge rate.

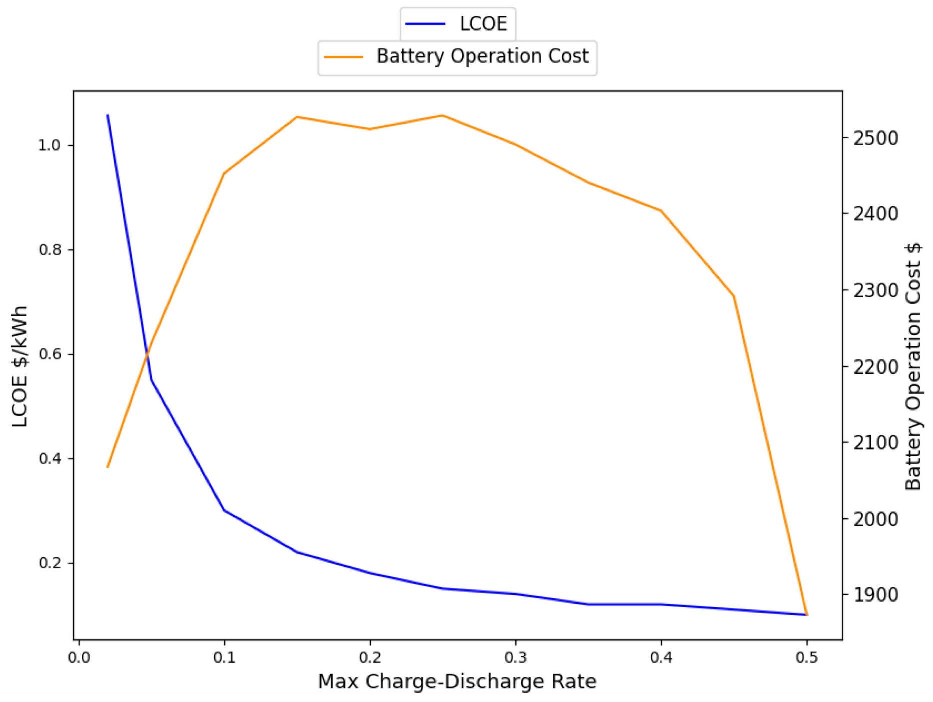


Figure 5.5. The annual schedule of the optimum design

To focus more on the battery operation, Figure 5.6. shows the battery state of charge and charging\discharging amount in each time step.

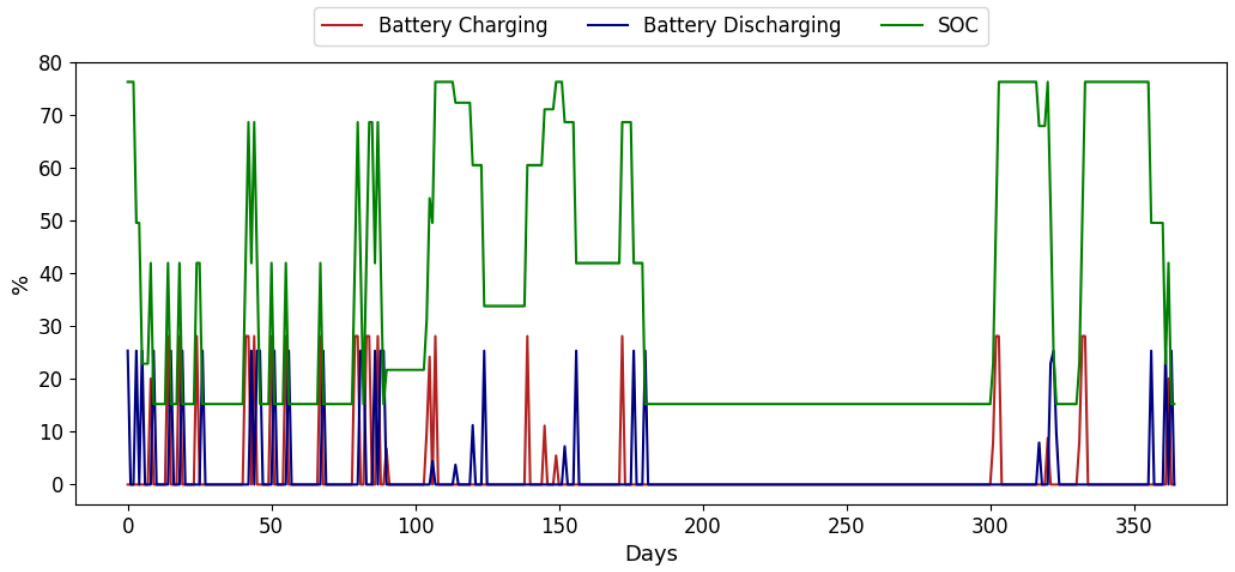


Figure 5.6. Battery State of Charge and Charge\Discharge schedule for the optimum energy system

There is an abnormality between day 200 and day 300, in which the battery is not working at all. This could be because of the lower grid price from August to October. Moreover, accurate analyzing of storage operations requires smaller resolutions, such as hourly. The operation of the battery will be assessed in the "Microgrid Operation" section of this thesis.

The monthly generation plot is displayed in a bar chart (Figure 5.7) to better compare the quantity of renewable generations, unmet load, surplus power and grid purchase.

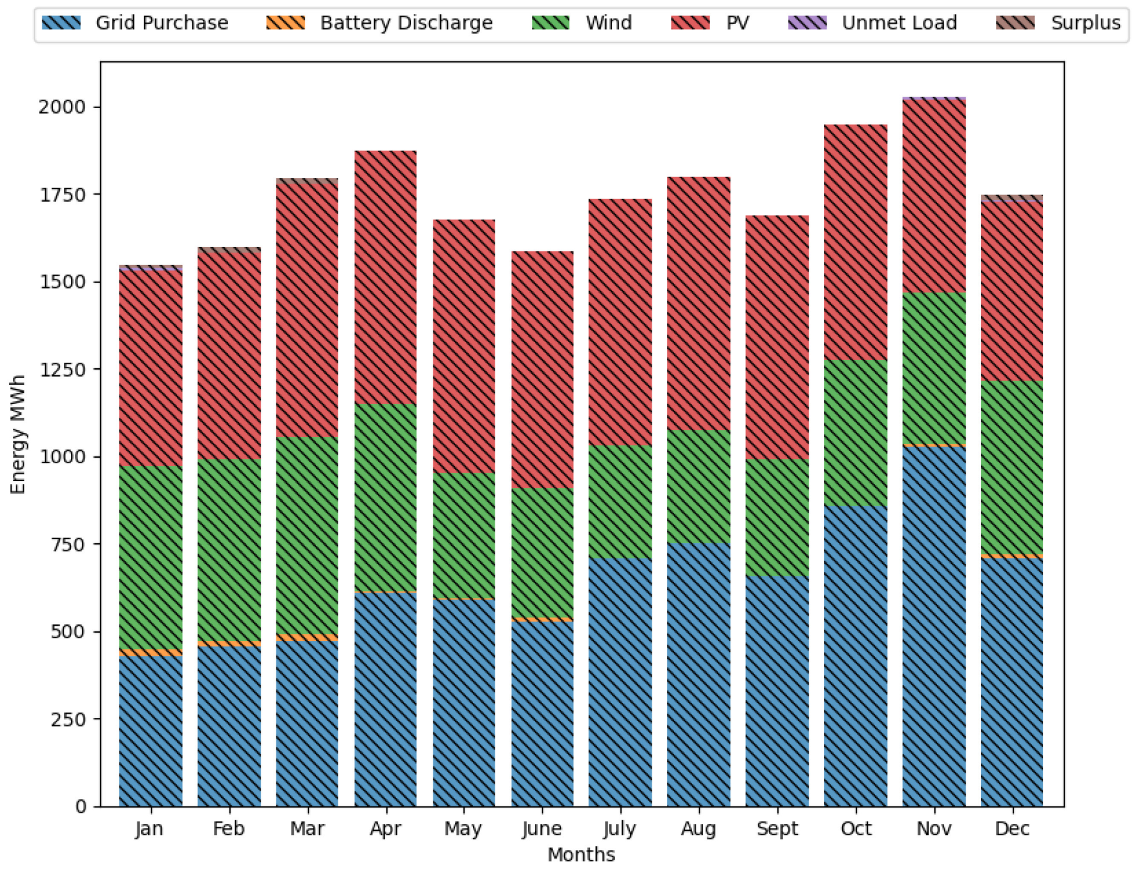


Figure 5.7. Monthly generation, unmet load, surplus power, and grid purchase bar chart

Based on this figure, the grid purchase volume is higher in the second half of the year in a meaningful way because of a lower grid purchase price, especially in November. As expected, the amount of PV generation is higher in warmer months compared to wintertime, while wind generation behaves the opposite. The summation of surplus power and unmet load in each month is negligible.

5.5. Heating System

As discussed in the case study (Section 3), electricity is not the only source of energy at Concordia university. Natural Gas (NG) is the other energy source used for heating purposes.

Although electricity is also engaged in heat generation, the value of the heat that is generated by electric boilers in the building is not considerable. The working schedule of the EV building's electric boilers is illustrated in Figure 5.8 for the year 2019.

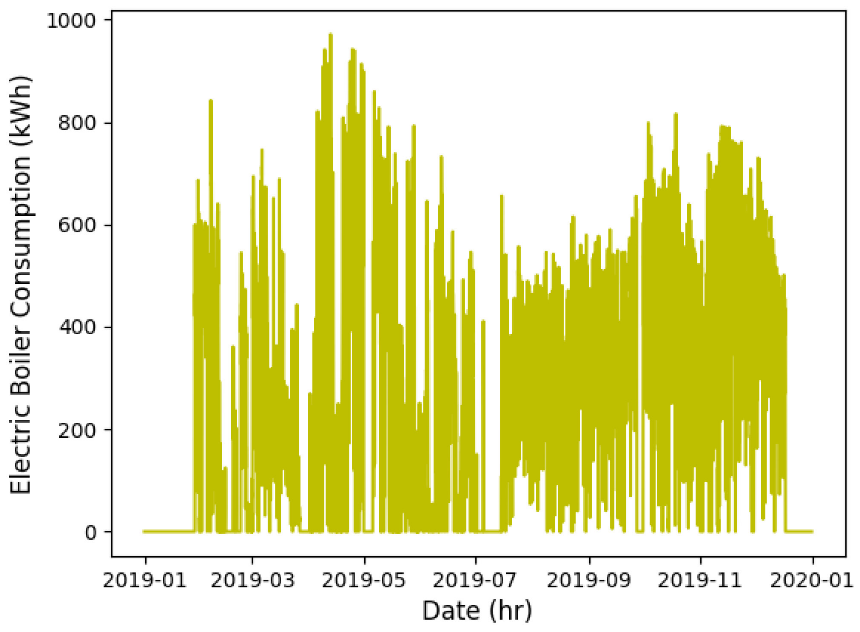


Figure 5.8. Electric boilers' electricity consumption during the year 2019

Assuming the Coefficient of Performance (COP) is equal to 1 for electric boilers, the amount of electric consumption of the electric boiler is equal to the heat generation. Therefore, based on Figure 5.8, the generated heat by electric boilers is not only low but also excessively fluctuates during the year (on-off).

On the other hand, the gas boiler in the building using natural gas is responsible for a high portion of the generated heat in the EV building. The only data that was available about the heat generation by the gas boiler is the supply and return temperature along with the flow rate of the water passing through it in 15 minutes resolution. The available data for the first day of 2019 is shown in Figure 5.9.

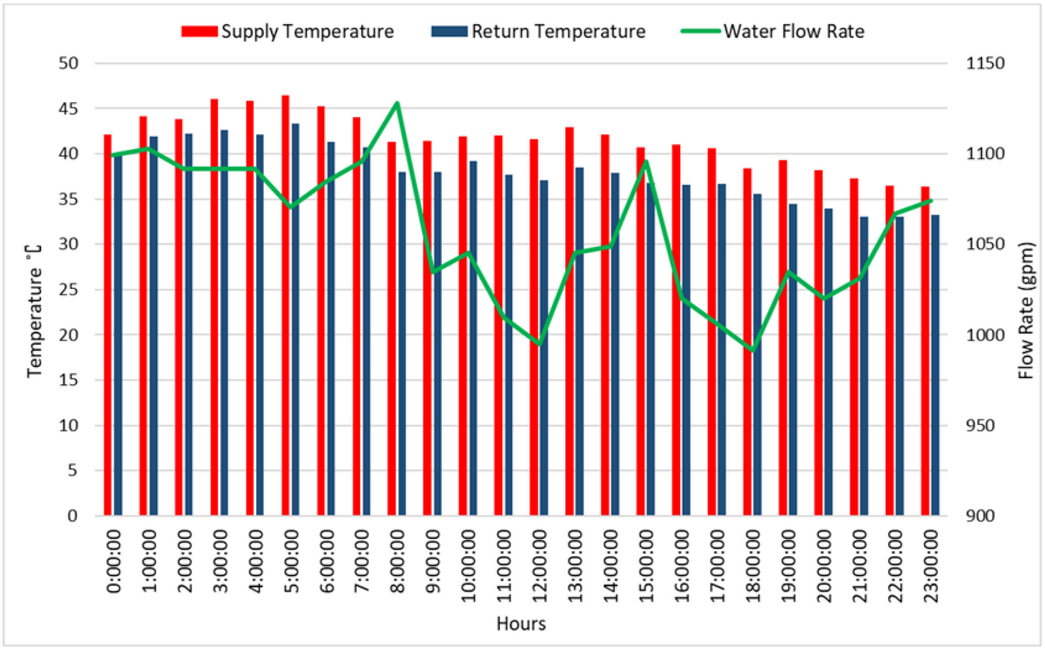


Figure 5.9. Available data for heat transfer calculations

Hence, the final transferred heat, based on the heat transfer formula mentioned in the methodology section, is calculated.

Based on the information provided by the facility management of the University, the electric boilers are oversized 50 percent more than the actual capacity. Therefore, most of the time, there is standby boiler that is not on duty for each electric boiler. Considering the COP equal to 1 for electric boiler, the calculated heat matches the electric boilers' electricity consumption. Figure 5.10 displays the total electrical consumption of the EV building in hourly resolution, including the electric boiler consumption.

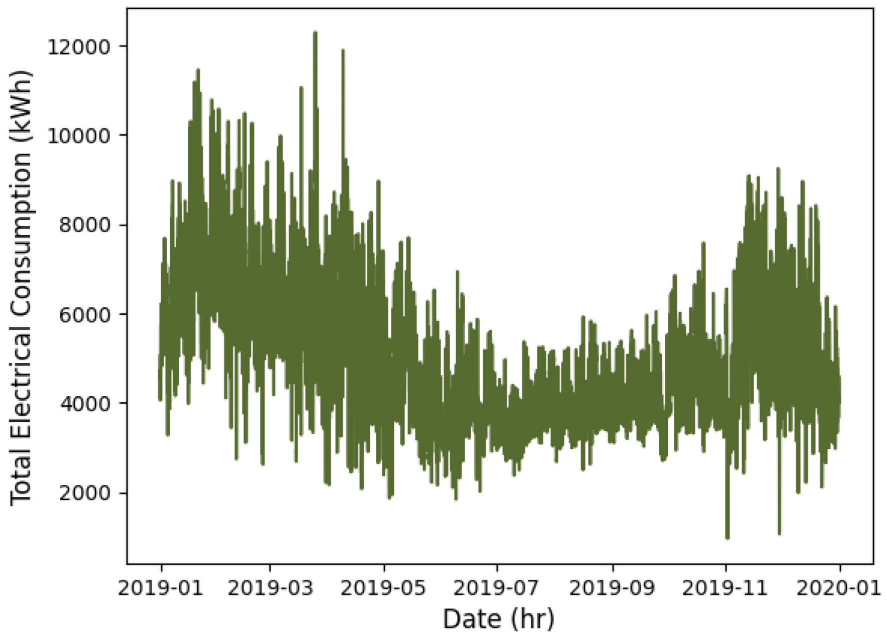


Figure 5.10. EV building's electricity consumption for the scenario with a replacement of the gas boiler with the electric boilers

It is evident from this figure that a considerable amount of electricity is required to replace the gas boilers for heat generation.

After heat calculation, the gas consumption could also be evaluated, as explained in section 2. In this study, the calorific value of Quebec's natural gas is considered equal to 40 MJ/m^3 [115], and the efficiency of the gas boiler is assumed to be 95 percent [116]. The gas consumption for the year 2019 is shown in Figure 5.11 with hourly resolution.

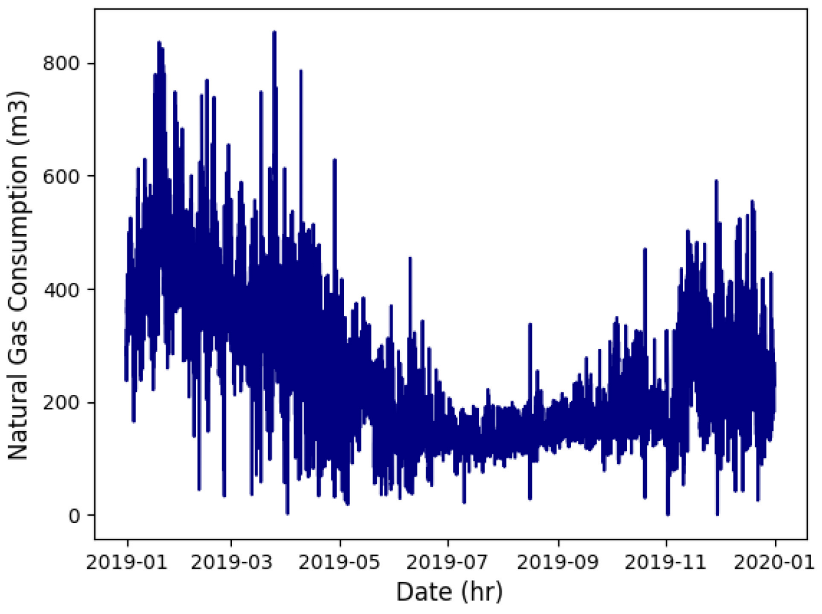


Figure 5.11. Calculated hourly natural gas consumption in the year 2019

5.6. Regional Generation Design

5.6.1. Electric Boiler

As discussed in the previous subsection, an enormous amount of electricity is required to remove the gas boilers and replace them with electric boilers that are already installed in the building. Adding this value to the electrical load demand will cause significant stress on the conventional grid since it is not prepared for this demand. Meanwhile, the renewable energy system that is designed in the local generation design subsection could not be oversized since the maximum available area for both generators (PV and wind turbines) has been reached. Therefore, this research evaluates the possibility of renewable electricity generation and transmitting it to the microgrid. It should be noted that the regional generation only includes wind turbines since installing large-scale wind turbines inside urban areas is more challenging than PV panels.

Based on the discussed methodology for regional generation in section 2, several assumptions need to be made. Since the case study in this research is located in Greater Montreal, the first assumption is specifying the land types inside greater Montreal and outside of it and then assigning the related Hellmann exponent to them. In section 3, the Hellmann exponents for each land type were reported in Table 5.4. The friction factors were assigned to each land based on the literature [117].

It should be noted that the residential meaning in urban areas differs from the meaning of residential outside of the metropolitan regions. Therefore, the Hellmann exponent for residential land type inside Greater Montreal differs from outside of it. This could be justified based on the building's dispersion inside and outside of urban areas.

The other critical parameter is the cost of each land type and the number of wind turbines that could be installed in each zone.

Table 5.12. Land cost and turbine density assumptions

Land Type (Greater Montreal)	**Land Cost ($\$/m^2$)**	**Turbine Density (No.)**
Waterbody	1000	10
Open Area	5	100
Resource and Industrial	25	20
Parks and Recreational	60	40
Commercial	40	10
Residential	80	60

The assumptions are reported in Table 5.12 only for the greater Montreal area since the outside of greater Montreal is not considered in this case study. However, the user can define any radius for the square mesh circle around the case study. Moreover, Waterbody land cost is considered extremely higher than other types (with lower expected turbine density) to exclude it from the model's selection since the selected wind turbines are not designed for offshore applications.

As discussed in section 2, the square mesh circle needs to be created by the user. In this thesis, Concordia's EV building is in the center of the circle (Latitude 45,49, Longitude -73.57), while the circle radius and square side (which define the size of each zone) are 20 and 4 kilometers, respectively. Therefore, the created mesh includes 76 zones, as illustrated in Figure 5.12.

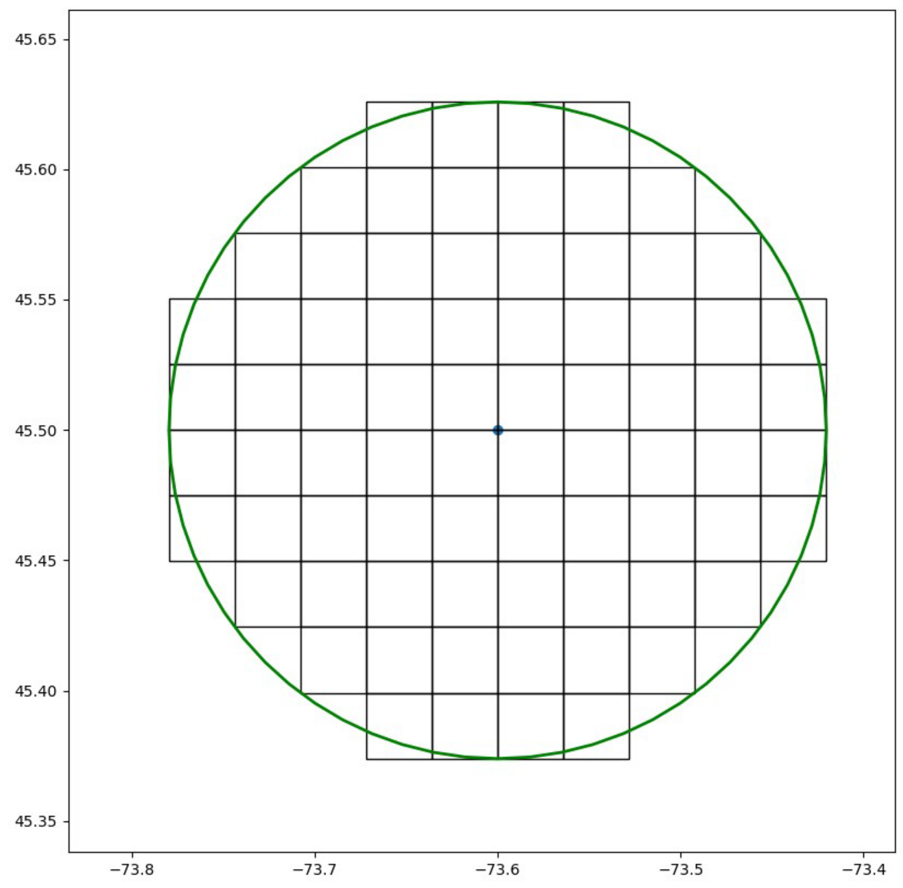


Figure 5.12. Created square mesh circle

Now the created square mesh circle should be located (based on the center's latitude and longitude) on top of the other maps (land types of greater Montreal, land types of outside greater Montreal, Grid power line, and transformer location map). The final created mapping model is displayed in Figure 5.13.

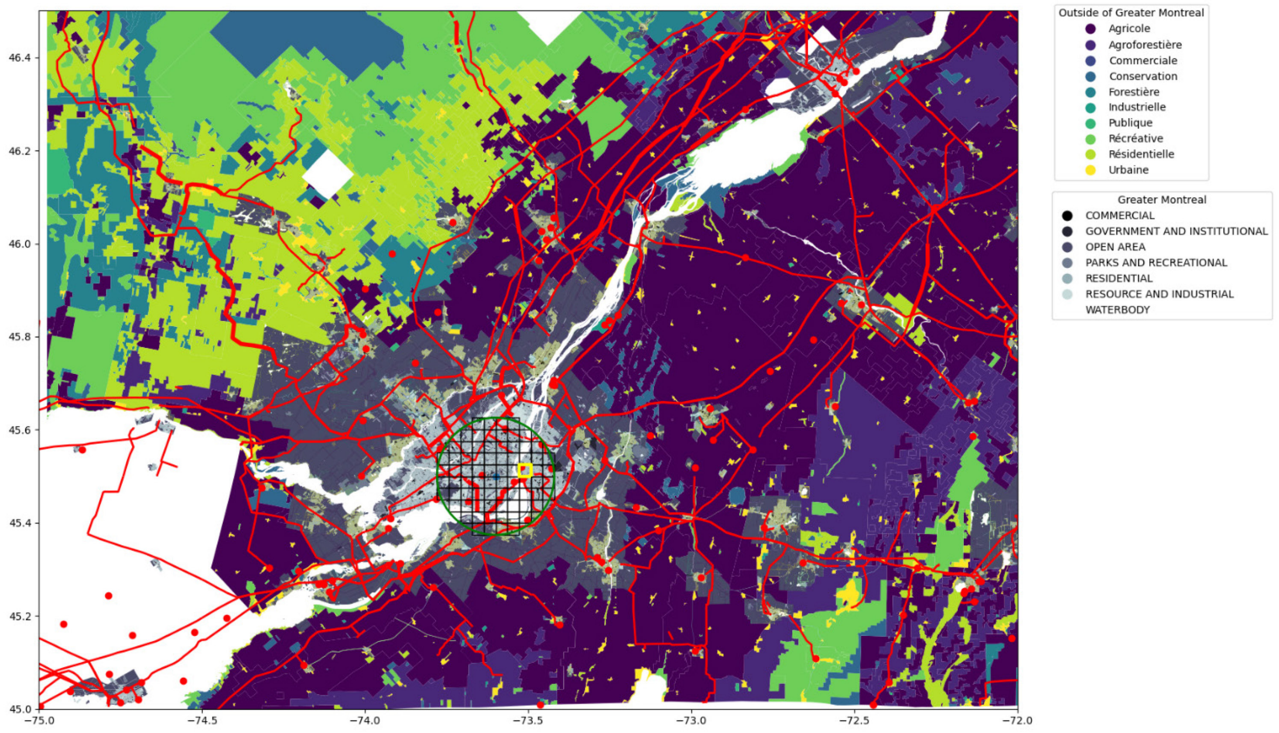


Figure 5.13. Final mapping model output

Afterward, the data for all zones (76 zones) in the square mesh needs to be extracted, and then it could be used as input for wind power and optimization models, as discussed.

The final result for the regional generation model is reported in Table 5.13.

Table 5.13. Regional model optimum configuration and economic report

Output	**Value**
NPC (CAD)	65,598,245
LCOE (CAD/kWh)	0.09974
Initial Capital Cost	57,711,640
PV Capacity (kW)	3877
Local Wind Turbine (kW)	1200
Battery Capacity (kWh)	6000
Regional Wind Turbines (kW)	23200

The result, which is shown in Table 5.13, indicates a sensible improvement in the economic factors such as the LCOE of the system compared to the local generation because of selling higher amount of electricity to the grid. However, the NPC of the system raised because of high initial capital cost which could be caused by purchasing 29 large-scale wind turbines and other related expenses. Since running the regional model provides vast amounts of electricity for the microgrid, therefore, while setting no limit on the battery capacity, it selects no storage in the optimum configuration. However, the designed system without storage could lead to low resiliency and reliability in case of grid failure, and moreover, the lack of storage system could create problems for the interaction with the grid as well because of the intermittent behavior of renewables. Therefore, in this research, about half of the battery storage system in the local generation design is set as the minimum bound of the battery storage capacity in the regional generation design.

The regional generation also improved the other important design indicators such as renewable penetration. The list of these indicators is reported in Table 5.14. Based on this Table, although a considerable amount of load was added to the total electrical demand because of the heat generation, the renewable penetration is increased since clean energy is also generated outside of the microgrid.

Table 5.14. Regional generation optimum design parameters

Parameter	**Value**
Renewable Penetration (%)	81
Unmet Load	0 days (0 kWh)
Purchase from grid (kWh)	14,997,320
Sell to Grid (kWh)	13,314,640

The developed model could also automatically select the optimum zone for installing the wind farm. The designated zone is shown in Figure 5.14. This zone is the optimum zone in terms of wind speed, distance to the transformers, land cost, and the maximum number of wind turbines that could be installed.

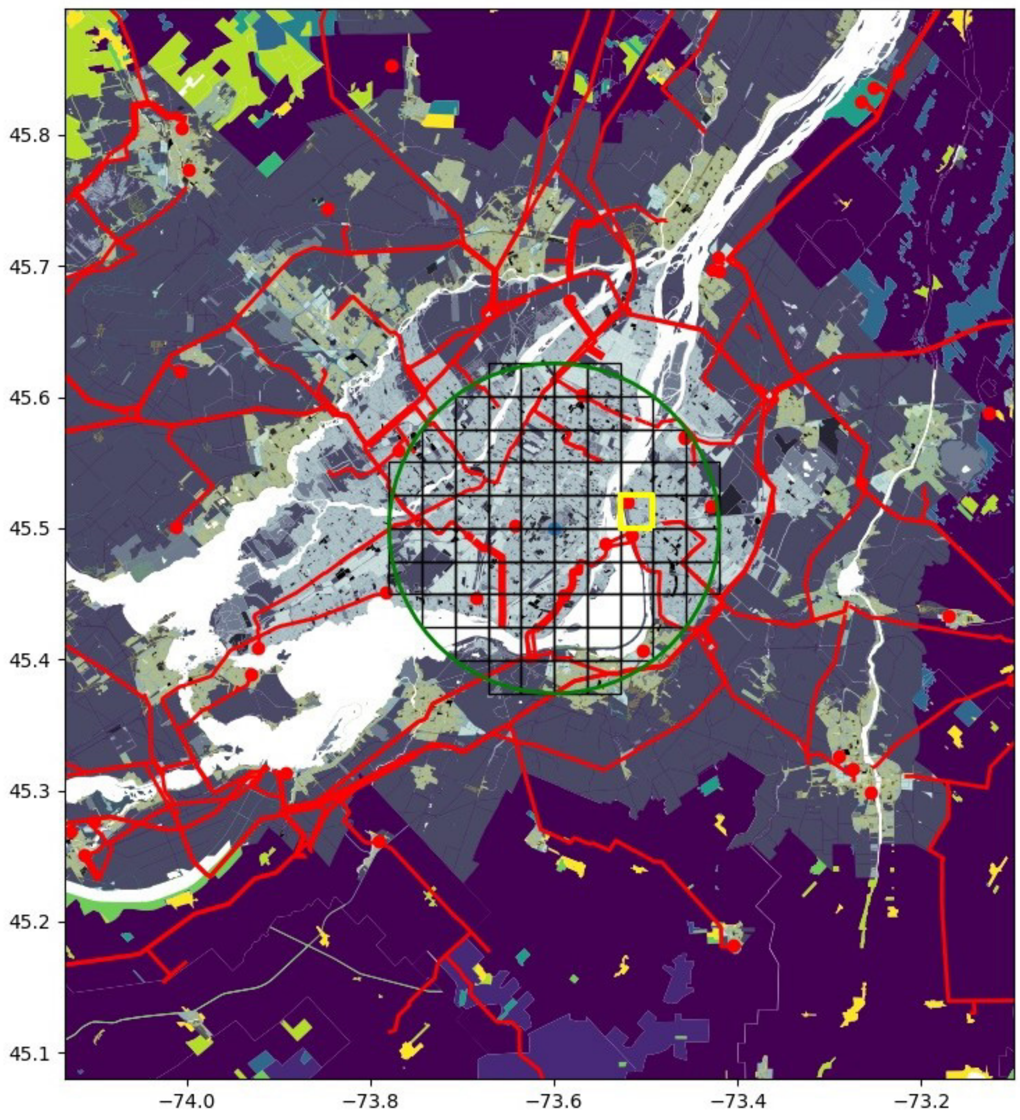


Figure 5.14. Final mapping model output (Zoomed in)

The final yearly schedule of the model is shown in Figure 5.15. Based on this figure, although the renewable penetration increased by about 20 percent, the grid dependency and purchasing electricity from the central grid is still high. Unlike local generation, the battery state of charge fluctuates throughout most of the days. Furthermore, the amount of unmet load is reduced to zero, which means the system never experienced a loss throughout the year.

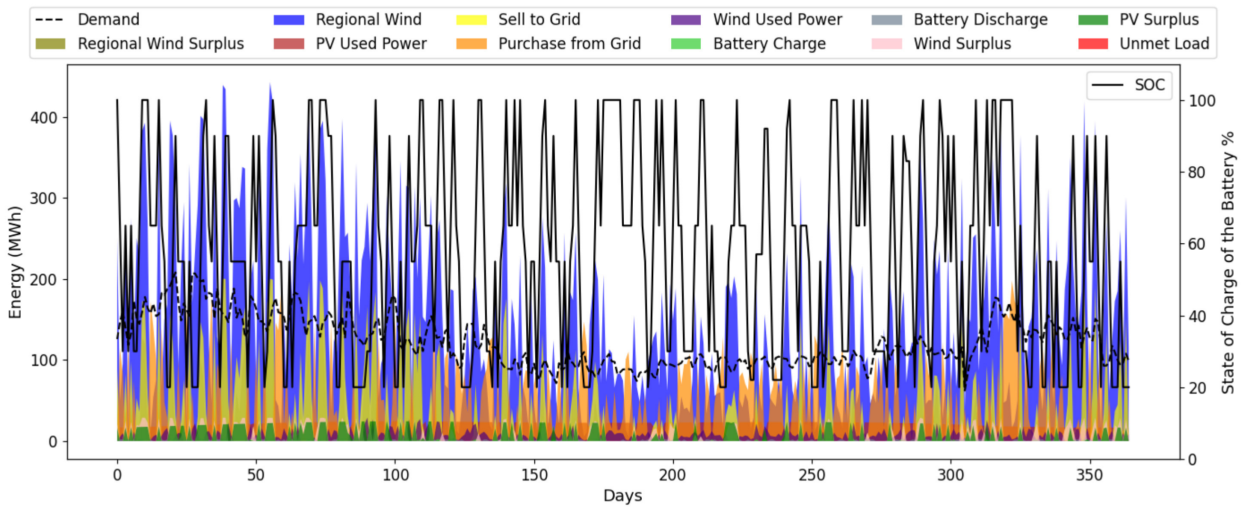


Figure 5.15. The annual schedule of the optimum regional design using the electric boilers

5.6.2. Heat Pump

Although because of the oversized electric boilers in the EV building, the initial capital cost for purchasing them is assumed to be zero, comparing it with a heat pump (even when a heat pump needs to be purchased) could be a viable study. This is mainly because the higher COP of heat pumps compared with electric boilers makes it an economical solution.

Therefore, in this step, a heat pump is added as a new component to the energy system instead of the electric boiler to generate heat using electricity. Since the average COP of an air source heat pump (used just for heat generation) is about 3, hence, after adding the calculated electricity demand of the heat pump to generate the same amount of heat that electric and gas boilers are producing, the final electrical consumption is shown in Figure 5.16.

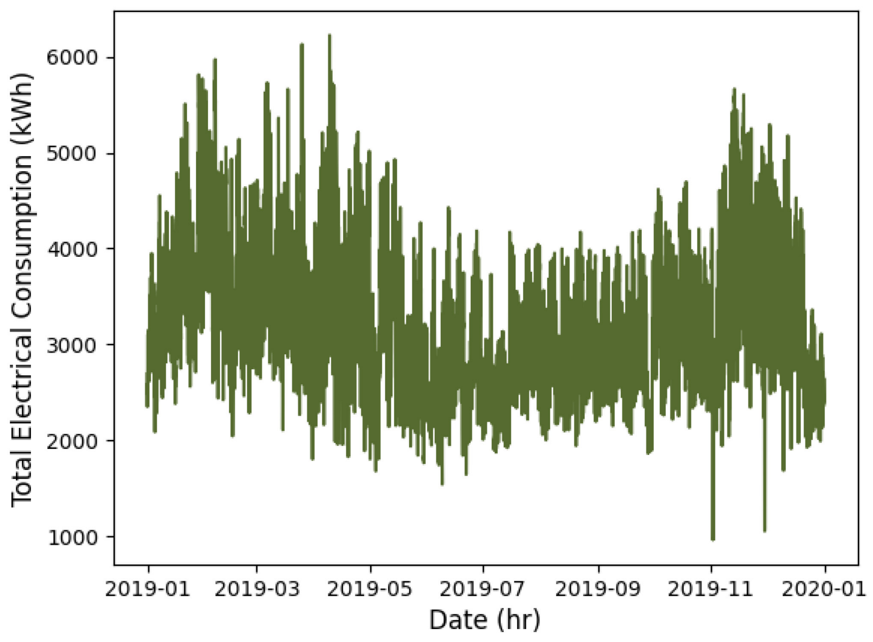


Figure 5.16. EV building's electricity consumption for the scenario with a replacement of the gas boiler with the heat pump

Comparing Figure 5.10 with Figure 5.16 reveals a considerable difference in the electricity consumption of heat pumps and electric boilers. The annual consumption in all the scenarios is reported in Table 5.15 for better estimates.

Table 5.15. Building's annual consumption in different scenarios

Scenario	Annual Consumption (MWh)
Local Generation (Without Heat)	20822
Electric Boiler	65503
Heat Pump	49597

The heat pump specification is reported in Table 5.16. Heat pump capacity is sized based on the peak heat consumption. Therefore, the model limits the lower bound of the heat pump's capacity by the value which is mentioned in Table 16.

Table 5.16. Selected air source heat pump specification

Parameter	Value
Capacity (kW)	5897
Fixed Cost (CAD/kW) [118]	1125

O&M Cost (% of Capital Cost)	0.6

The result of the optimum design of the regional generation energy system using a heat pump is summarized in Table 5.17.

Table 5.17. Regional model using heat pump optimum configuration and economic report

Output	**Value**
NPC (CAD)	56,419,574
LCOE (CAD/kWh)	0.08261
Initial Capital Cost	49,492,415
PV Capacity (kW)	3877
Local Wind Turbine (kW)	1200
Battery Capacity (kWh)	6000
Regional Wind Turbines (kW)	20800
Heat Pump Capacity (kW)	5897

Looking at Table 5.17, it is evident that the economic parameter of the designed system improved by using heat pump compared to the scenario using an electric boiler. The results show that the LCOE of the electricity consumption decreased by about 1.7 cents per kilowatt hour. Moreover, although the heat pump needs to be purchased in the heat pump scenario, and this adds up to the system's fixed cost, the initial capital cost is also reduced since fewer regional wind turbines are selected for this scenario.

To further evaluate the comparison of using the electric boilers and heat pumps, the optimum design parameters are reported in Table 5.18. Based on these outputs, the renewable penetration is also improved by about 5 percent while the amount of unmet load is still zero.

Table 5.18. Regional generation optimum design parameters

Parameter	Value
Renewable Penetration (%)	86
Unmet Load	0 days (0 kWh)
Purchase from grid (kWh)	8,972,505
Sell to Grid (kWh)	14,932,170

The yearly schedule is displayed in Figure 5.17. to evaluate the different generations and consumptions.

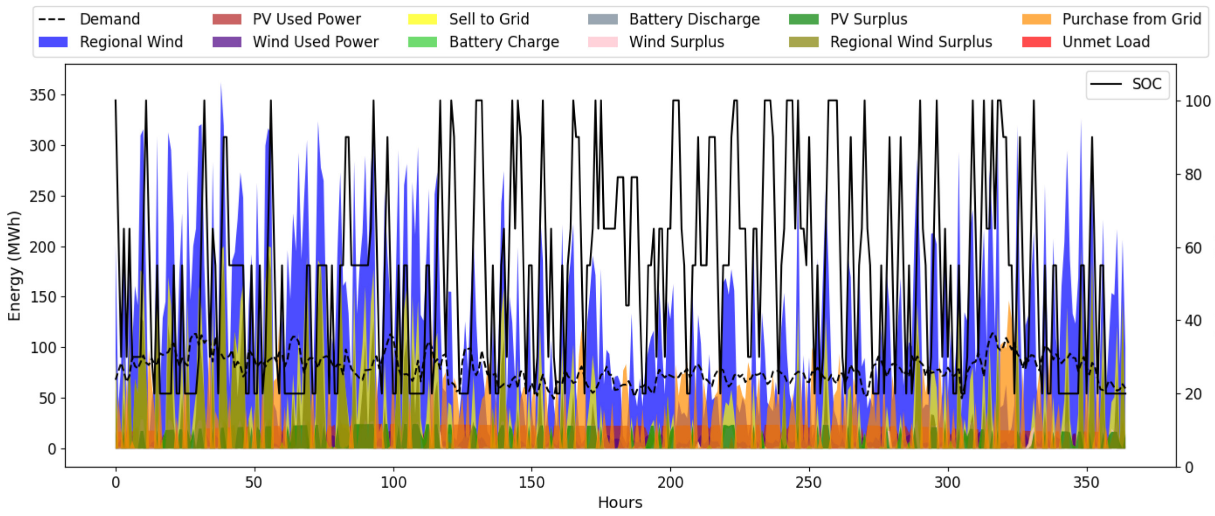


Figure 5.17. The annual schedule of the optimum regional design using a heat pump

While the amount of unmet load is zero during the year, there is considerable surplus power for regional wind generation. This could be solved by increasing the limit (maximum 0.2 MWh per day) which has been set for the grid node that the regional generation needs to be injected. Furthermore, the battery shows considerable fluctuations, especially in wintertime, which could add to the battery operation cost. This is because of the smaller capacity selected for the storage system compared with the local generation scenario.

5.7. Proposed Tariff

Although in the previous subsection, using a heat pump improved the overall system's economic and generation quality, renewable penetration is not still at its highest value since we can generate even more electricity outside of the microgrid area. Moreover, from the central grid perspective, a large prosumer like Concordia University could still add stress to the grid, especially in the wintertime. This could be because of the electricity rate that is being used in the building, and as mentioned earlier, it is based on the contract between the University and Hydro Quebec. This rate is slightly flexible and not dynamic. Therefore, in this thesis, the effect of changing the electricity rate on the designed system is evaluated based on the University and Hydro-Quebec perspectives.

To create a flexible and dynamic pricing scenario, there are two aspects that need to be considered. The first is the amount of electricity used in each time step and the second is the usage time. Taking these two aspects into account could push the design model to purchase less from the grid in the high demand time. Therefore, to categorize each time step into its class, the if-else scenario loops should be created as reported in Table 5.19.

Table 5.19. Dynamic pricing strategy

Electric Boiler Scenario	**Heat Pump Scenario**	**Summer (April 1-Nov. 30) CAD/kWh**	**Winter (Dec. 1 – March 31) CAD/kWh**
If daily demand is less than 140,000	If daily demand is less than 100,000	0.06	0.08
If daily demand is between 140,000 and 160,000	If daily demand is between 100,000 and 120,000	0.08	0.12
If daily demand is between 160,000 and 180,000	If daily demand is between 120,000 and 140,000	0.10	0.18
If daily demand is between 180,000 and 200,000	If daily demand is between 140,000 and 160,000	0.12	0.26
If daily demand is between 200,000 and 220,000	If daily demand is between 160,000 and 180,000	0.14	0.36
If daily demand is more than 220,000	If daily demand is more than 180,000	0.16	0.48
Sell price for any amount less than the defined limit (200,000 kWh/Yr.)		0.10	0.20

Table 5.19 shows the pricing strategy for a regional generation design using both heat pumps and electric boilers scenarios for purchasing and selling. The result of the explained strategy on EV building's data is shown in Figure 5.18 for the regional generation design electric boiler scenario.

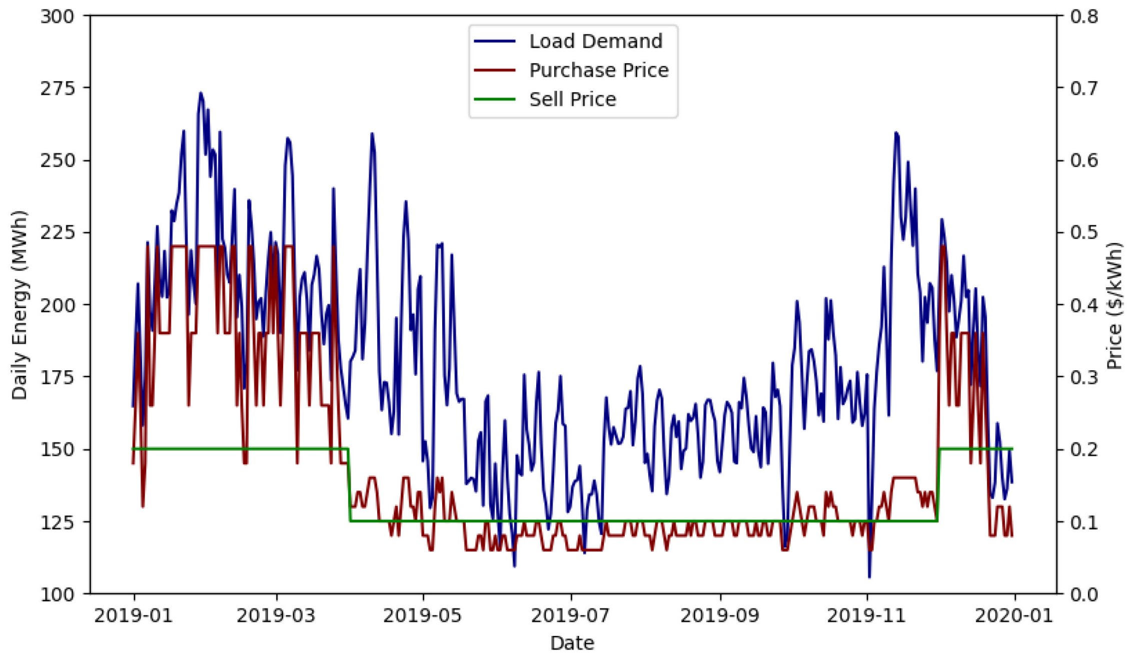


Figure 5.18. The proposed dynamic pricing strategy for regional generation design using an electric boiler

As shown in Figure 5.18, the purchasing price fluctuates in a daily manner based on the time of use (winter or summer) and the amount of electrical consumption each day. For instance, with this pricing strategy, a jump in daily consumption in winter causes a considerable rise in the purchasing price, while the same growth in summer will cause way less increment in price. Unlike purchasing price, the selling price only changes with the season (winter to summer), and it is assumed to be constant during the season.

The same pricing strategy could be used for regional generation using a heat pump scenario, which is shown in Figure 5.19.

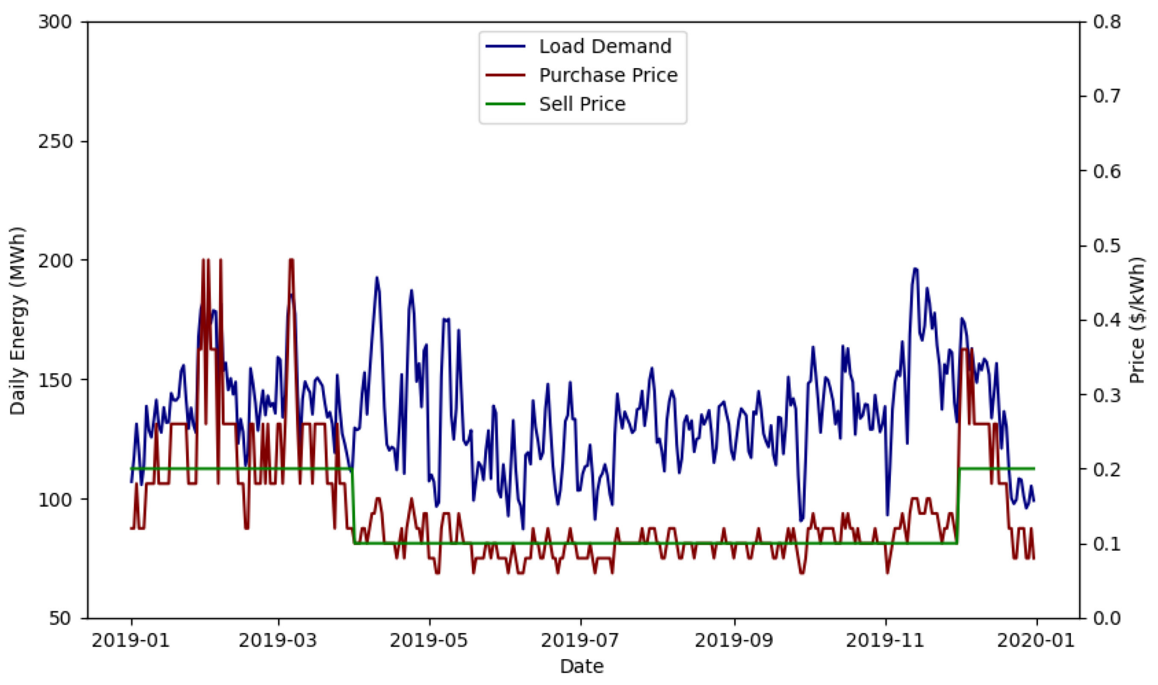


Figure 5.19. The proposed dynamic pricing strategy for regional generation design using a heat pump

To evaluate the impact of the explained pricing strategy on the final design, the created pricing policy function is given to the regional generation models (for both the electric boiler and heat pump scenarios).

Table 5.20. The final optimum configurations and economics using the proposed Tariff

Output	**Heat Pump Scenario**	**Electric Boiler Scenario**
NPC (CAD)	45,311,789	70,661,136
LCOE (CAD/kWh)	0.05282	0.07999
Initial Capital Cost	69,332,415	95,358,040
PV Capacity (kW)	3877	3877
Local Wind Turbine (kW)	1200	1200
Battery Capacity (kWh)	6000	6000
Regional Wind Turbines (kW)	33600	41600

Based on the results reported in Table 5.20, all the economic parameters show improvement in both scenarios compared to the results of using HQ tariff. This means that using dynamic pricing could be in favor of Concordia university in terms of economic aspects.

Furthermore, the proposed pricing strategy could also be beneficial from HQ's perspective as well. By looking at Table 5.21, it is evident how the stress on the grid is removed by reducing the grid purchase in both scenarios when using the proposed Tariff. This reduction is more sensible in Winter when the central grid is experiencing higher load demand.

Renewable penetration is also at the maximum level and considerably higher, 11 and 14 percent rise for heat pump and electric boilers scenarios, respectively, compared to using HQ pricing strategy for both scenarios.

Table 5.21. Regional generation optimum design parameters using proposed tariff

Parameter	**Heat Pump Scenario**	**Electric Boiler Scenario**
Renewable Penetration (%)	94	92
Unmet Load	1 day (49579 kWh)	2 day (65503 kWh)
Purchase from grid (kWh)	4,938,031	7,764,784
Sell to Grid (kWh)	26,121,760	27,754,480

Although changing from the local to regional scenario lowered the payback period, using the proposed tariff could also fairly bring down the amount of time that it takes to recover the initial investment. This could also make the proposed tariff interesting for the prosumer (Concordia University) and the investors.

5.8. Environmental assessment

One of the advantages of regional generation scenarios was the environmental benefit and decarbonization along with economic improvements. As discussed in section 4.4, EV building alone consumed 2,204,762 m^3 of natural gas in one year (2019). Considering 1.92 kg CO2 emissions equivalent per cubic meter of natural gas [119], it generates about 4233 tons of CO_2 per year. Since the gross floor area of the EV building is 69204 m^2, the CO2 emission per area of the

EV building is about 61 kg/m^2 only for space heating which is higher than the CO2 emissions average values for large-scale buildings in different locations [120]. A complete report of the annual emissions of Concordia's EV building is shown in Table 5.22 [119]. The space heating emissions could be reduced to zero by using the regional generation scenarios and replacing the gas boiler with a heat pump or electric boiler.

Table 5.22. EV building's natural gas consumption by-products

NG Consumption By-Product	**Value (kg)**
CO_2	4,233,143
NO_X	5290
CO	2962
N_2O	77
CH_4	81

Based on the formulation mentioned in section 2 for penalty calculation, the total amount to which the user should be penalized is about 68,964 CAD per year. Moreover, the University not only can save by removing the emissions but the annual cost of the natural gas consumption could also be diminished. Final savings are summarized in Table 5.23.

Table 5.23. Savings by ceasing natural gas consumption

	Saving (CAD/Year)
Removing NG Consumption	661,428
Removing Emissions	68,964
Total Save	730,392

5.9. Renewables and Power Consumption Uncertainties

Before starting the operation module, the uncertainties caused by the fluctuations of the renewable resources and the user behavior that lead to unforeseen power demand must be reduced. In this thesis, several methods have been analyzed, as explained in the methodology section. The results of each method will be discussed in the following sections. Since the models are used for the short-term forecast (one or a maximum of two days), they have been used at different times of the year (winter and summer) to generalize their results.

To form the LSTM layers, the Keras library with TensorFlow backend is used. Furthermore, grid search optimization is applied to find the optimum hyperparameters (Table 5.24).

Table 5.24. Hyper Parameters of the LSTM model and Parameter selection results

Parameter	Values/Types	Hyper Parameter Optimization - Load	Hyper Parameter Optimization - Wind	Hyper Parameter Optimization - Solar
Hidden layers (No.)	3	-	-	-
Number of neurons (No.)	60	-	-	-
Activation function	Sigmoid	-	-	-
Optimizer types	{Adam, RMSprop}	Adam	Adam	Adam
Batch size	{1,32,64}	64	32	32
No. of epochs	{50,80,100,120,135,150}	150	120	100

5.9.1. Power Consumption Forecasting

To forecast two days ahead of electrical consumption of the EV building, the hourly historical data for 2019 and 2020 is used for training purposes. Afterward, two randomly selected days in each season, the 12th and 13th of July as representative of summer and the 12th and 13th of March as representative of Winter, are selected for test purposes.

The hyperparameter optimization shows that the combination of the Adam optimizer and a batch size of 32 with 150 epochs results in fits with only minor loss in the training stage. Further increase in the number of epochs results in a slight reduction of error with the training data, as shown in Figure 5.20; moreover, above 150 epochs, the overfitting tends to cause a reduction of accuracy of the model in forecasting the test data set.

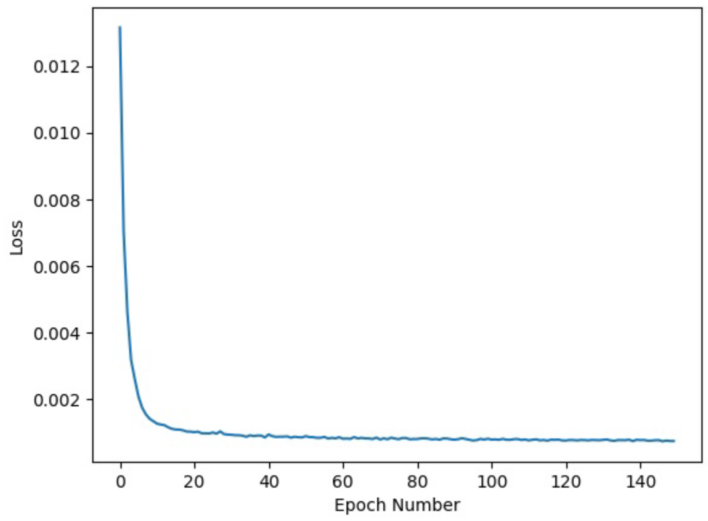


Figure 5.20. Model convergence plot for a single LSTM model

As explained, the trained model was then tested on the 12th and 13th of July and March 2020. The training and test dataset are of the same resolution. The results of all three employed models to forecast the next 48 hours are depicted in Figure 5.21 for both summer and winter and are compared with the observed values.

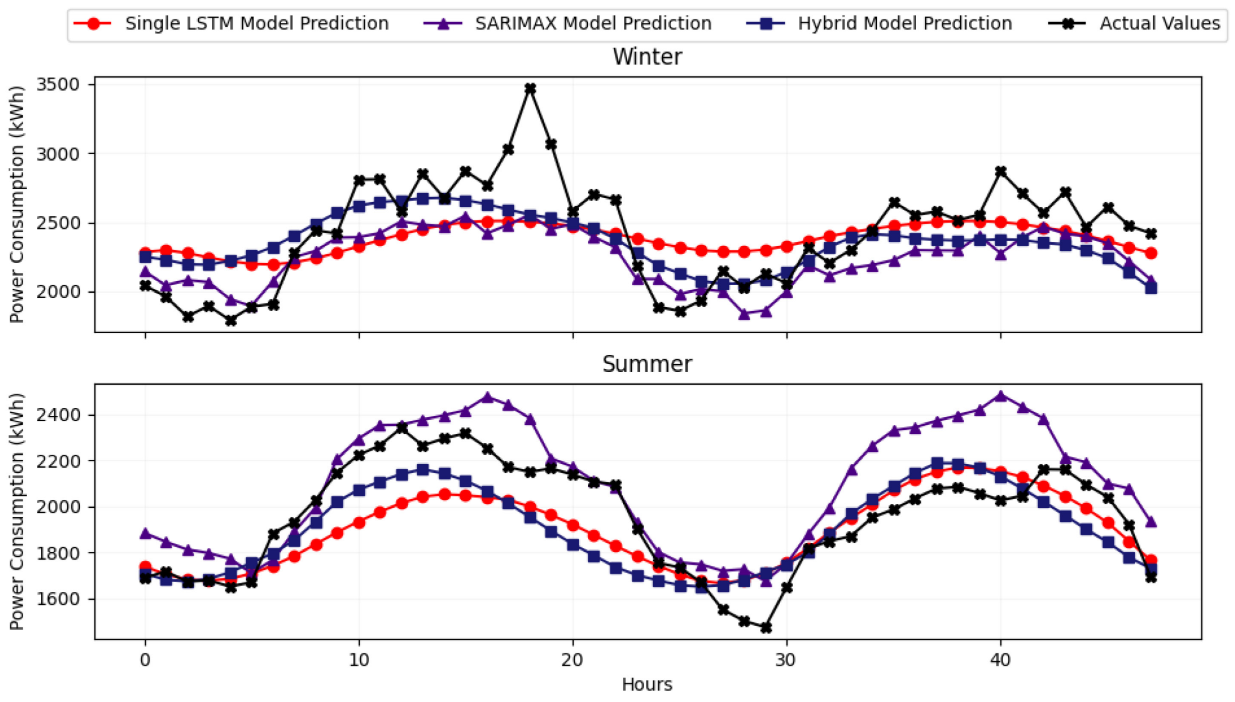


Figure 5.21. Power consumption forecasting results for 48 hours

In Figure 5.21, despite the fact that the SARIMAX model better predicted the seasonality and the noise in the first 24 hours of data, it is not capable of forecasting the next 48 hours as it only repeats the same seasonality trend and noise every 24 hours. This problem could be resolved by employing an LSTM model. As it is shown, the predicted values will be in better accordance with the observed values by using LSTM. However, the small fluctuations, especially in peak hours, are not properly predicted. Using the proposed hybrid model, as it is demonstrated in Figure 5.21, the peaks will be forecasted in a superior way in comparison with the outcomes from each individual model.

The performance of each model is summarized in Table 5.25. The evaluation metrics calculated show less error and higher accuracy for the proposed hybrid model in both seasons. For instance, in summer, he RMSE of the LSTM and SARIMAX models are 155.15 and 189.96, respectively, while the RMSE of the Hybrid model is slightly lower (152.59), showing a lower error for load forecasting. The accuracy improvement is even more considerable in winter, where the Hybrid model reduced the mean absolute percentage error by about 4 and 6 percent compared with single LSTM and SARIMAX models, respectively.

Table 5.25. Power consumption forecasting results

Model	RMSE	MAE	MAPE
Summer – 12 & 13 July			
LSTM	155.15	127.17	6.27
SARIMAX	189.96	151.55	16.74
Hybrid LSTM-SARIMAX	152.49	128.34	6.21
Winter – 12&13 March			
LSTM	314.12	259.57	13.78
SARIMAX	295.28	238.39	15.21
Hybrid LSTM-SARIMAX	285.75	232.51	9.62

5.9.2. Wind Speed Forecasting

LSTM, SARIMAX, and the proposed hybrid methods were developed in Python programming language. To generalize the results of the testing of the developed models for the whole year, three different test sets from summer (the last 2 days of July 2020), Fall (the last two days of October 2020), and Winter (the last two days of December 2020) were selected as the representative of different seasons. For summer, the model trained with the data from Jan 2020 until 29 July 2020, while for Fall and Winter, the training set includes data from Jan 2020 until 29 October 2020 and 29 December 2020, respectively. A few missing values were found in the training sets, and they all have been replaced by the average of the previous and next values. Also, the outlier detection procedure is implemented using boxplot visualization and calculating quartiles based on the formula explained in the methodology section. The results show no outlier in the training sets.

To scale up the predictors and target variable into a unique scale, the MinMaxScaler method from preprocessing sub-package of the Sklearn library is used. All features are scaled into the range between 0 and 1 before feeding to the neural network model.

To form the LSTM layers, the Keras library with TensorFlow backend is used. Furthermore, grid search optimization is applied to find the optimum hyperparameters (Table 5.24). The hyperparameter optimization shows that the combination of the Adam optimizer and a batch size of 32 with 150 epochs results in fits with only minor loss in the training stage.

The Weibull model was developed by creating a function to generate the wind speed distribution. The Weibull distribution of the wind speed for the year 2020 is calculated in the

Python environment by creating a Weibull function using c, k, and Γ parameters that have been explained in the methodology section. The result is shown in Figure 5.22. The histogram graph in Figure 5.22 displays the data distribution in the range of 0-20 m/s. Also, the Weibull probability feature is created using the Stats package of the Scipy library.

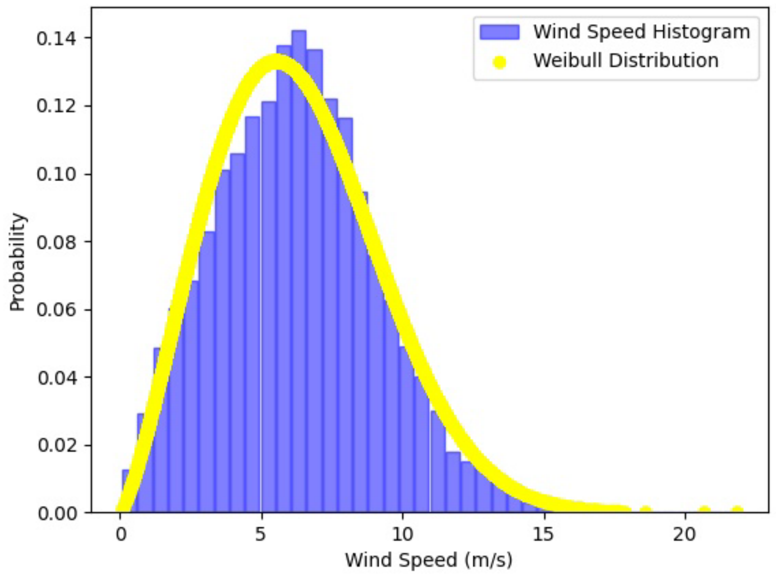


Figure 5.22. Weibull distribution of the historical wind speed

The parameter selection of the SARIMAX model is made by applying the Autoarima package from the Pmdarima library and a grid search through 42 different combinations of the $(p, d, q)(P, D, Q, s)$ parameters. The last two months of wind speed historical data of the first half of the year 2020 is used for training the Autoarima for parameter selection. The results (Table 5.26) show that the combination of (1,0,1)(2,1,0,24) yields the minimum AIC and is selected as the optimum set of parameters for the SARIMAX model. All the other combinations that are not mentioned in Table 37, led to AIC equal to infinity.

Table 5.26. Report of the Grid Search for SARIMAX parameter selection

Combination	**AIC**	**Combination**	**AIC**
(0,0,0)(0,1,0,24)	11271	(1,0,1)(2,1,0,24)	4044

(1,0,0)(1,1,0,24)	4955	(2,0,1)(2,1,0,24)	3959
(1,0,0)(0,1,0,24)	5548	(3,0,1)(2,1,0,24)	3961
(1,0,0)(2,1,0,24)	4709	(2,0,2)(2,1,0,24)	4221
(0,0,0)(2,1,0,24)	10553	(1,0,0)(2,1,0,24)	4707
(2,0,0)(2,1,0,24)	3998	(1,0,2)(2,1,0,24)	3972
(2,0,0)(1,1,0,24)	4211	(3,0,0)(2,1,0,24)	3962
(3,0,0)(2,1,0,24)	3964	(3,0,2)(2,1,0,24)	3963
(3,0,0)(1,1,0,24)	4175	(4,0,1)(2,1,0,24)	3963
(5,0,0)(2,1,0,24)	3965	(3,0,1)(1,1,0,24)	4176

The results consist of details of the selected combination, including the AIC, BIC and HQIC are reported in Table 5.27.

Table 5.27. Selected combination information

Parameter/Metric	**Value/Type**
Optimum non-seasonal orders	(2,0,1)
Optimum seasonal orders	(2,1,0,24)
No. of observations	2160
Log likelihood	-4270.623
AIC	3952.516
BIC	4004.849
HQIC	3976.106
Covariance type	Outer Product of Gradients (OPG)

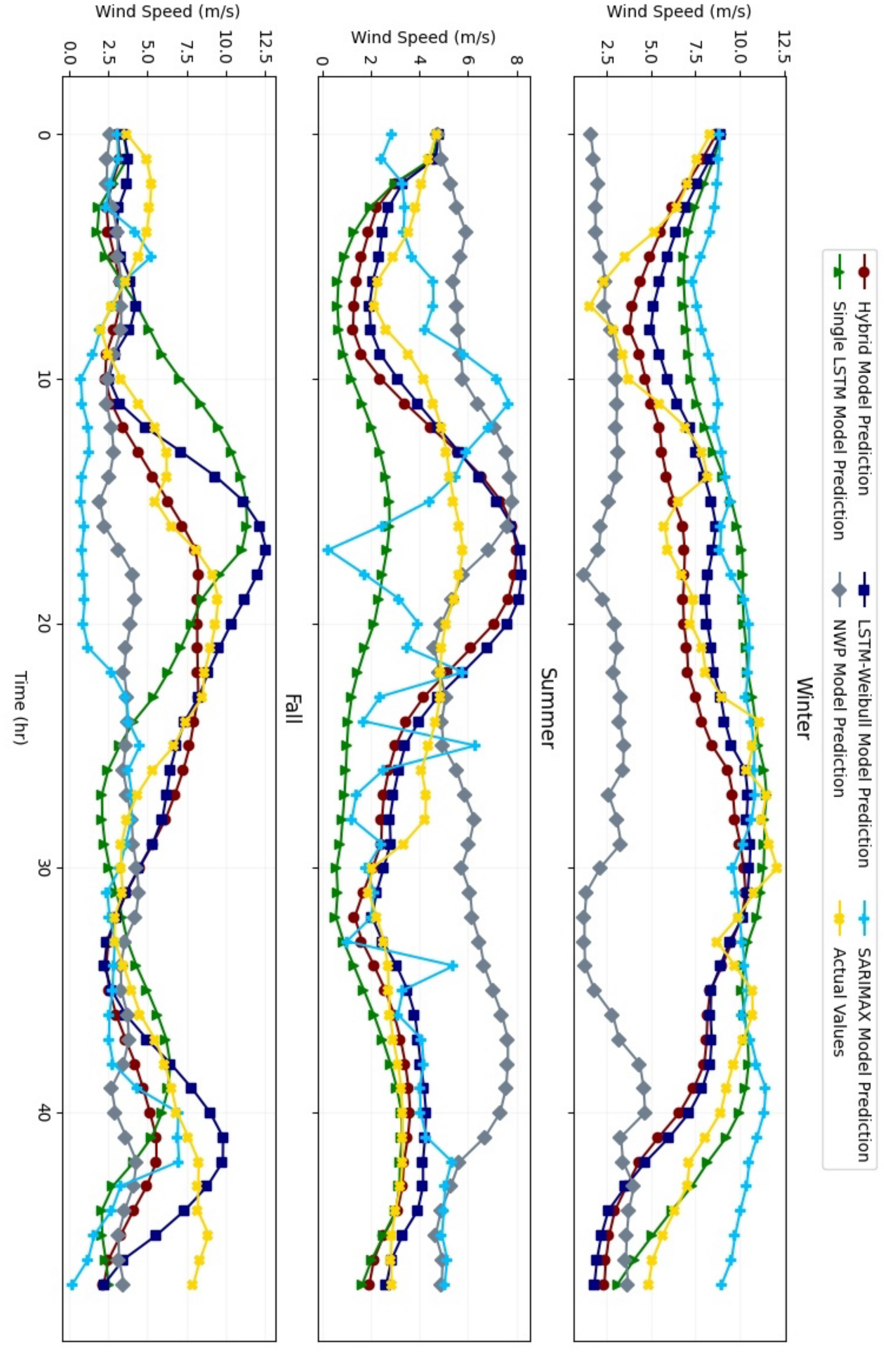


Figure 5.23. 48 hours forecasting results in different seasons

Figure 5.23 shows the forecasting results of all three models and the hybrid models' results for the last two days of July, October, and December 2020. At a glance, the results using a single LSTM model do not show proper wind speed forecasting, especially in peak hours that are way over or under actual values. By applying the SARIMAX model, although the mean value of the forecasted wind speeds is nearer to the mean value of the actual wind speed compared with the single LSTM model, it has not captured the fluctuations, peaks, and trends decently. While using the NWP model resulted in a significant error, especially in winter, applying the proposed integrated model can considerably reduce this error. Furthermore, the seasonality and trend issues seem to be fixed for the whole prediction horizon in different seasons. However, the accuracy is not high in the two major peaks. A quick comparison between the result of the proposed hybrid model and the result of the other models reveals the hybrid model's ability to better integrate the fluctuations and trend.

To evaluate and compare the models precisely, the RMSE, MAE, and MSLE of each model's results are calculated based on what was explained in the methodology section. The results are reported in Table 5.28. The LSTM model results show a 2.21-3.16 root mean squared error in different seasons that depict the LSTM model's inability to accurately predict using the three meteorological historical data (temperature and humidity) as the independent variables. Although with the SARIMAX model, the RMSE and MSLE are improved in the fall, the error is still high. As explained in the methodology section, the LSTM model with different layers in its cells can deal with unexpected behavior of data. Therefore, a proper feature should be added to the LSTM model for better training. Integrating the probability distribution of the wind speed with the LSTM model and using it as an input feature could be one of the alternatives to boost the LSTM ability. The results in Table 5.28 show that the proposed integrated LSTM-Weibull model can reduce the RMSE of the single LSTM model in forecasting winter, summer, and fall representative days by about 13, 39, and 31 percent, respectively. These error reductions show that adding a proper feature, such as the Weibull probability of the wind speed, can help LSTM accurately forecast the future. However, in case of any unexpected wind behavior that has not happened before (and it is normal in climatic situations), even the integrated model could lead to a considerable error. Therefore, to solve this challenge, hybridizing the results of the NWP model predictions with the proposed integrated model could be helpful in capturing the unexpected behavior of the wind that was not recorded in the historical data. The single NWP prediction results also show high RMSE and MAE and even higher MSLE compared with other models, especially in Winter and Fall. However, by hybridizing the NWP model with the integrated model, the RMSE of the proposed model decreased 47%, 17%, and 32%, respectively, in summer, winter and fall compared with the single LSTM model.

Table 5.28. Evaluation metrics of all the models for Wind Speed Forecasting

Model	**RMSE**	**MAE**	**MSLE**
July (Summer)			
LSTM	2.21	1.86	0.395
SARIMAX	2.64	2.24	0.234
NWP	2.02	1.70	0.227
Integrated LSTM-Weibull	1.35	1.12	0.071
Hybrid LSTM-Weibull-NWP	1.18	0.95	0.066
December (Winter)			
LSTM	2.14	1.63	0.106
SARIMAX	2.81	2.3	0.155
NWP	5.58	4.90	0.849
Integrated LSTM-Weibull	1.87	1.55	0.110
Hybrid LSTM-Weibull-NWP	1.78	1.50	0.078
October (Fall)			
LSTM	3.16	2.58	0.272
SARIMAX	2.25	2.73	0.294
NWP	4.14	3.11	0.799
Integrated LSTM-Weibull	2.18	1.60	0.100
Hybrid LSTM-Weibull-NWP	2.16	1.67	0.139

To consider the effect of the prediction horizon on the final accuracy, the prediction periods were extended to 168 hours (one week) for all seasons instead of 48 hours (two days). Since the lookback period is 48 hours, it means that after forecasting the first 48 hours into the future, the next hours will be predicted based on the prior predictions. Therefore, the accuracy of the model could be lower by increasing the prediction horizon. The result of the prediction horizon extension is shown in Figure 5.24 for the hybrid model. It is evident that the hybrid model acts less and less accurately with increasing the prediction period except for fall, which still can predict the third day (until 72 hours) correctly, and that could be because of the fewer fluctuations on the third day.

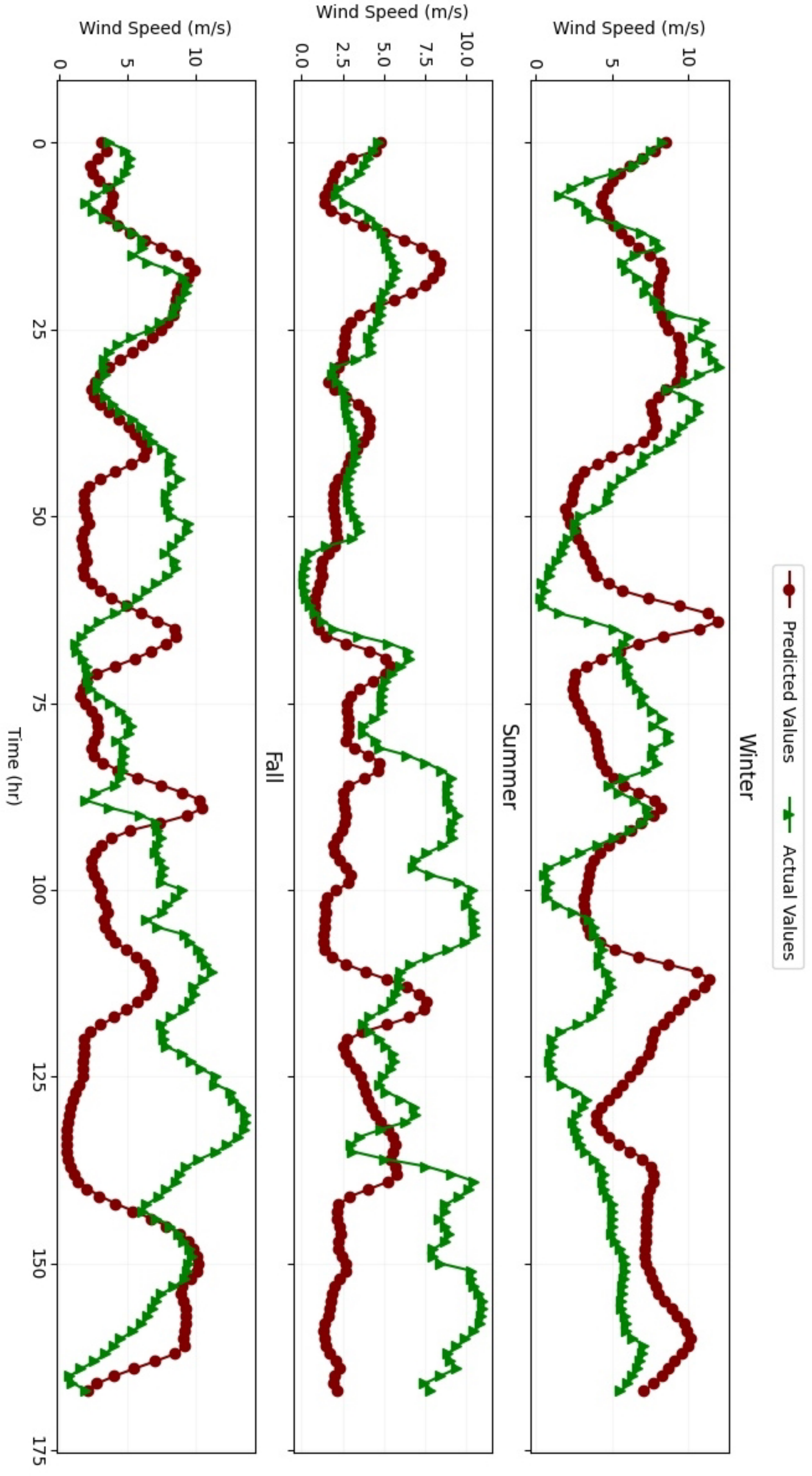


Figure 5.24. One week (168 hours) forecasting results of the hybrid model in different seasons

5.9.3. Solar Irradiance Forecasting

The confusion matrix for predictors and target variables is shown by a heatmap in Figure 5.25. The heatmap shows a strong correlation between downward shortwave solar irradiance (NWP-Forecast-SW) and the observed irradiance. Although, as Figure 5.25 shows, there is a poor correlation between the observed irradiance and the downward longwave irradiance (NWP-Forecast-LW), the predictor is still considered as an input since it could positively impact the results.

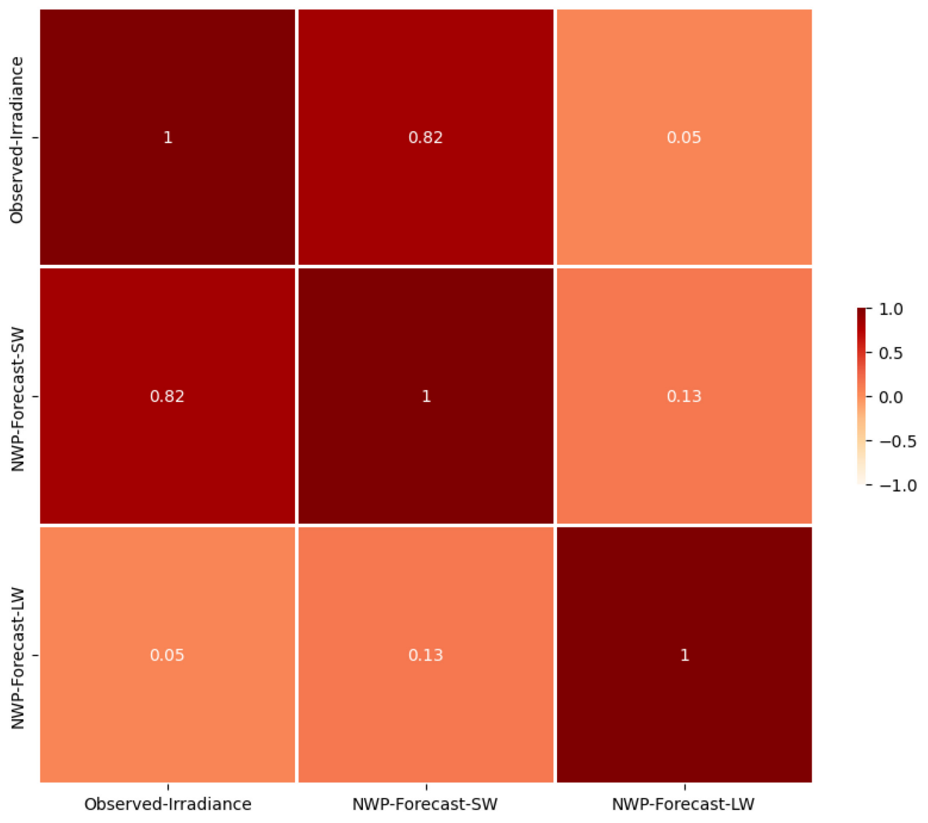


Figure 5.25. Correlation heatmap between independent and target variables

The forecasting results for all models: single LSTM, NWP, and the proposed hybrid model are shown in Figures 5.26 to 5.27. Three different days in May are selected to test the model. The days are chosen based on the clearness index of the sky and the sky cloud coverage. Although there are about six different cloud coverage conditions based on the Oktas unit [121], in this study, three scenarios are considered for sky condition: clear sky, scattered (partly cloudy), and overcast (cloudy). The days are selected in a way to challenge the prediction ability of different models, especially the single LSTM model that only uses historical data. On 7th May, with a clear sky situation, the forecasting result (Figure 5.26) shows an inaccurate forecast for all models; however, the proposed integrated model forecasting result has a better fit with the observed values compared with other single models. Moreover, since the NWP shortwave prediction is shifted to the right and misses the peak, it also caused a slight shift in the hybrid model and decreased its accuracy.

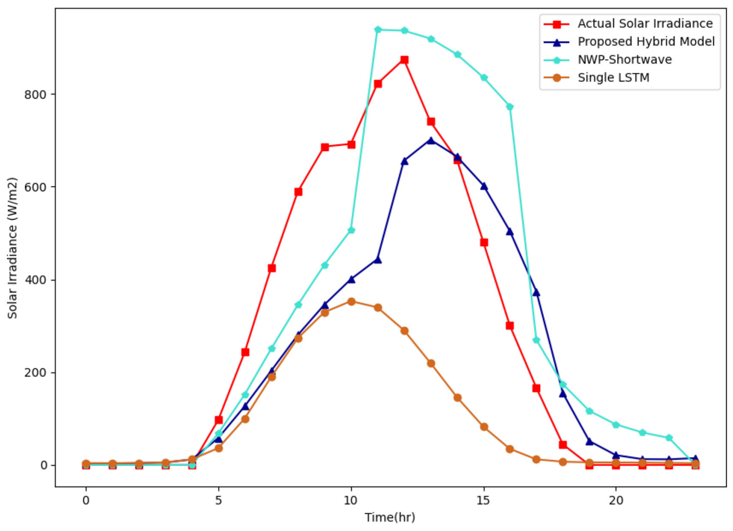


Figure 5.26. Forecasting results for a Clear Sky day (7th May)

On the other hand, on 11th May, the sky is scattered, and the single LSTM model cannot capture the unexpected movement of the clouds from the historical data; it predicts a pattern based on the historical trend while it is far from the actual trend. However, during this time, the NWP prediction could contribute to the hybrid model providing more accurate forecasts. As shown in Figure 5.27, the hybrid model forecasted values have less error than single LSTM and NWP models.

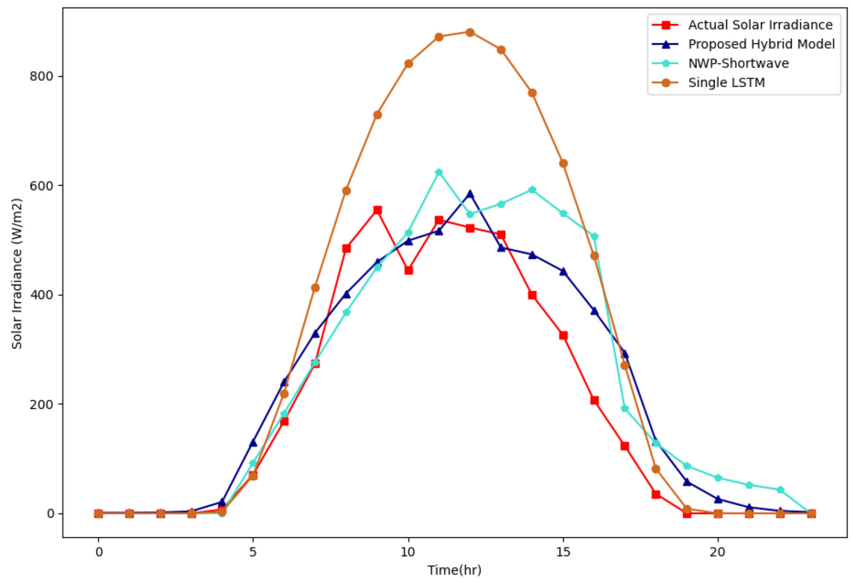


Figure 5.27. Forecasting results for a scattered day (11th May)

The effectiveness of the proposed method is shown even more significantly in the case of the overcast sky on 4th May (in contrast to a clear sky on previous days). The day ahead forecast has shown that the proposed model can deal with the existence of clouds and adequately predict solar irradiance, especially in peak hours (Figure 5.28).

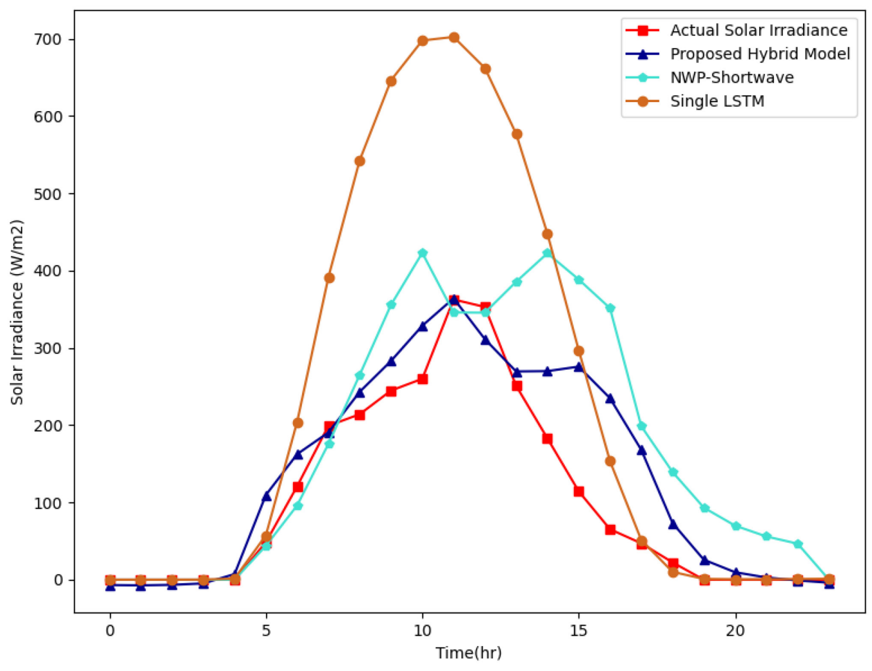


Figure 5.28. Forecasting results for an overcast day (4th May)

Table 5.29. shows the day ahead solar irradiance forecasting results (hourly resolution) of different models for three discussed days. The evaluation metrics show a significant improvement in the results of the single LSTM model when it is hybridized with the NWP model in scattered and overcast scenarios (presents of the clouds in the sky).

Table 5.29. Evaluation metrics for solar radiation forecasting in Cloudy day – Overcast (4-May)

Model	RMSE	MAE
4th May (Overcast)		
Single LSTM	197.19	124.43
NWP-Shortwave	117	77
Integrated Proposed Model	62.96 (±0.10)	40.60
11th May (Scattered)		
Single LSTM	192.81	126.91
NWP-Shortwave	101.48	67.76
Integrated Proposed Model	72.01	53
7th May (Clear-Sky)		
Single LSTM	272.64	185.82
NWP-Shortwave	171.60	123.26
Integrated Proposed Model	167.13	114.43

5.10. Optimum Operation

In the design section, both local and regional generations have been studied. However, since one of the assumptions was transmitting the regional power generation to the nearest transformer of the central grid and then transmitting it to the microgrid, in case of any failure of the grid, the microgrid doesn't have access to the regional generation as well. Therefore, the operation strategies in this research are all developed considering only the local energy system.

Operation management is categorized into two classes as follows. First, assuming the microgrid has access to the grid, and second is the resilience-oriented optimal operation in the time of grid failure.

5.10.1. Grid-Connected Optimal Operation

Rate flex G [122] was selected as the Hydro Quebec Tariff for operation management purposes, and it is reported in Table 5.30.

Table 5.30. Purchasing price based on rate G flex

	Summer (April 1-Nov. 30)	**Winter (Dec. 1-March 31)**
Price of energy outside peak hours	10.29 Cent/kWh	12.815 Cent/kWh
Price of energy during peak hours	N/A	51.967 Cent/kWh

Based on Hydro Quebec rules [123], the peak hours only are defined for winter, and it is between 6 to 9 AM in the morning and between 4 to 8 PM in the evening.

As discussed in the methodology, the main goal of the grid-connected optimal operation is an optimum unit commitment to minimize the operation cost. To generalize the result for a whole year, two randomly selected days from the summer and winter seasons are chosen. The result is displayed in Figure 5.29, which shows the optimal schedule of the grid-connected mode.

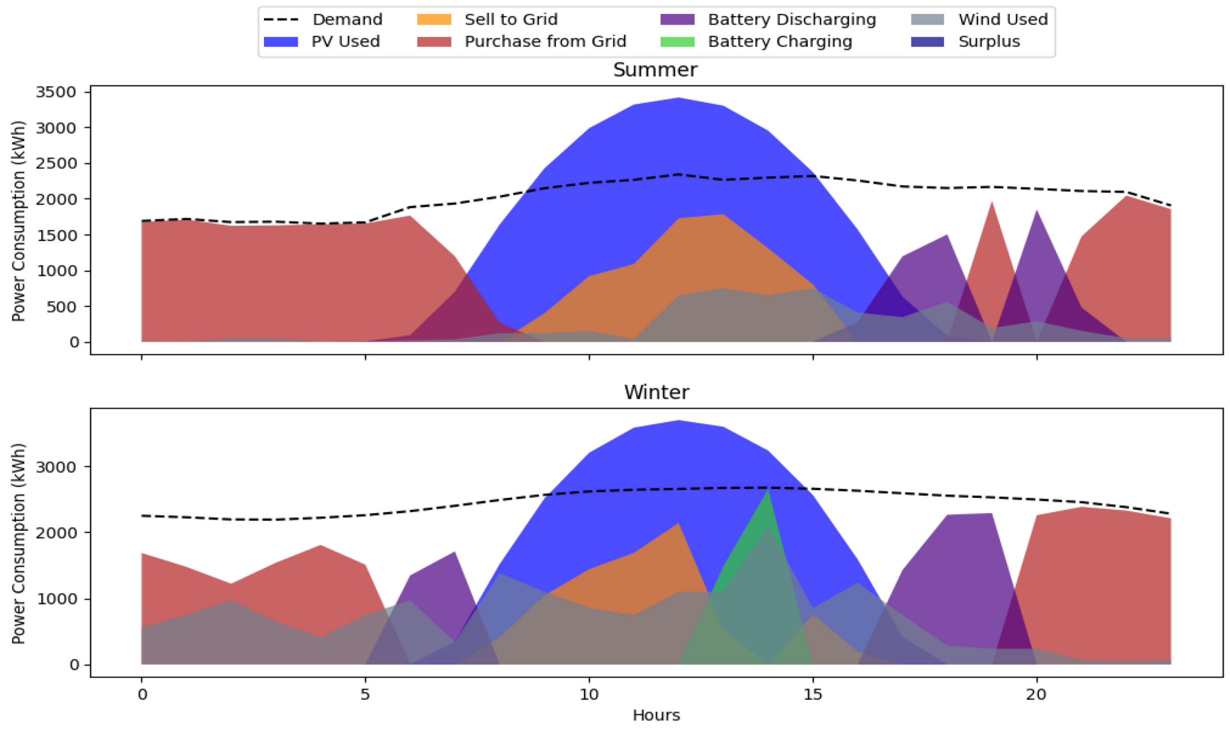


Figure 5.29. Optimal operation schedule in the grid-connected mode

As shown in Figure 5.29, in the optimum schedule, wind power is available through most of the hours of the day in winter, while in summer, due to the lack of wind speed at the beginning of the day, this amount is negligible. On the other hand, solar generation covers the day demand as expected in both seasons. Since the case study is an educational building, the demand is still high at midnight, and while wind power covers some parts of winter, in summer, grid purchase is the only method of supply. The surplus power in both seasons is about zero since one of the terms in objective functions is minimizing the curtailment.

Since the flex tariff is used (Table 5.30) peak and off-peak hours electricity rates, an unintentional peak shaving is also employed in the economic dispatch of the grid connected system which led to the lower operating cost.

The operation cost on the selected winter day is about 1676 CAD, while in summer, it is about 1693 CAD. The cost includes the trade-off between the grid and microgrid and the battery operation.

5.10.2. Resilience-Oriented Off-Grid Operation

The load demand is randomly selected for two days in winter (15th and 16th February 2019) and two days in summer (14th and 15th July 2019) as representatives of the cold and warm seasons when load demand fluctuates more. The resolution of the load demand is hourly, and the time horizon is 48 h.

The number/capacity of the components in the designed energy system is listed in Table 5.31.

Table 5.31. The microgrid components and battery information.

Component	No./Capacity
Wind Turbine 25 kW (No.)	50
PV Panel 220 W (No.)	13,662
Battery (kWh)	9584
Maximum SOC of the batteries (%)	95
Minimum SOC of the batteries (%)	10
Initial State of Charge (kWh)	8000
Discharging Efficiency (%) [24]	90
Charging Efficiency (%) [24]	95
Maximum charge/discharge rate	0.3
Replacement cost (USD/kWh)	156
Total cycles in the lifetime of each unit [25]	300
Curtailment Factor ($/kW) [26]	0.1

Since the microgrid is designed to be grid-connected in urban areas, the renewable penetration is 53%. Therefore, the system requires a sufficient energy management system to control the operation during grid power failure resulting in the loss of power supply.

The other important parameter that needs to be set and one that significantly impacts the final results is the loss coefficient ($\emptyset$), with $/kW unit, which is a constant parameter (not changing through the time horizon). Selecting a proper coefficient needs a trial-and-error process to find an optimal coefficient in the case of both economic and reliability aspects. Therefore, in this research, all possible coefficients in the range of 0.01 to 1 (step = 0.01) are tested. Since the final results of the unmet load only change with certain coefficients (the other coefficients have the same rate and trend of the amount of unmet load (kWh) while having a higher operation cost), the threshold coefficients are selected and are shown in Figure 5.30. The results show that increasing the penalty for having an unmet load could reduce the loss; however, this increment could cause a considerable rise in the operating cost.

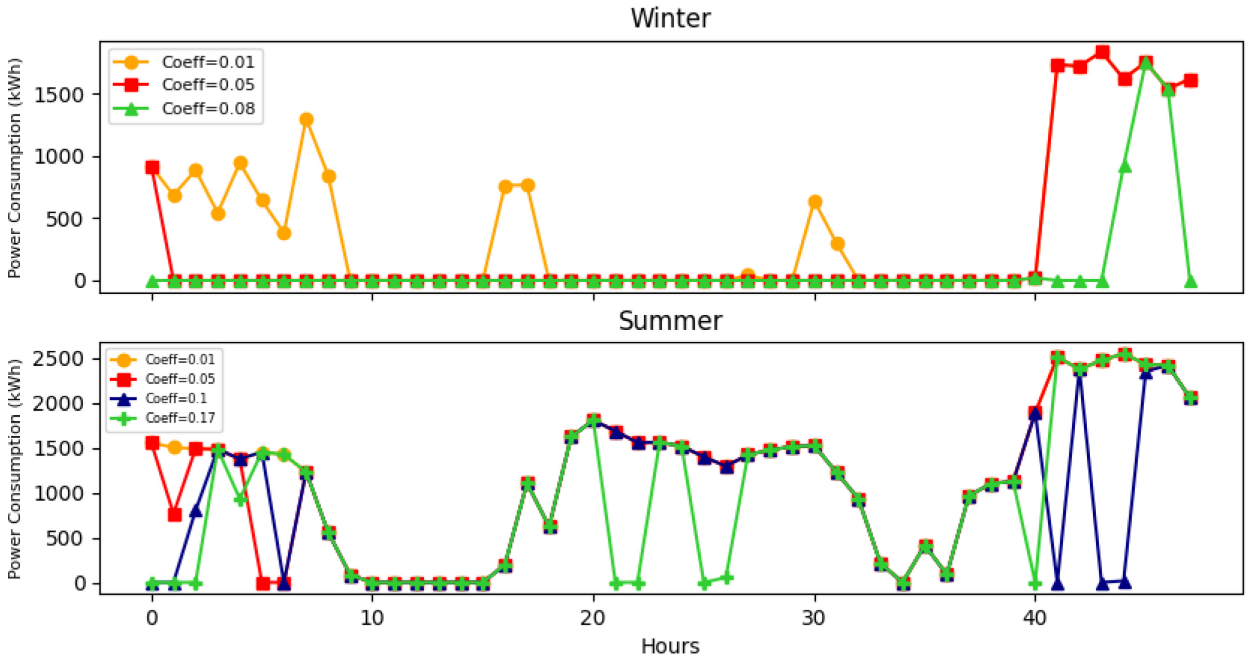


Figure 5.30. Trial and error results for finding the best coefficient

Furthermore, increasing the loss after the maximum thresholds (0.08 and 0.17) will not further affect the loss. Therefore, in this study, the threshold coefficients 0.08 and 0.17 were considered the loss coefficients for winter and summer, respectively. These coefficients have a minimum loss and a minimum cost (compared to the larger coefficients).

The optimization model was coded in Python programming language using the Pyomo platform [103]. Since there is no nonlinearity in the model's equations, and with the presence of binary variables, the mixed-integer linear programming (MILP) method [124] is used to formulate the problem. The CPLEX solver [105] is selected to solve the developed MILP problem.

The results of using the actual load demand to find the optimum schedule of the microgrid in the off-grid mode in summer and winter are illustrated in Figure 5.31. The model's outcome shows that the amount of wind surplus power in the winter season is considerable in some hours. On the other hand, this amount is negligible on the selected days in summer since wind power generation is reduced in summer compared to winter.

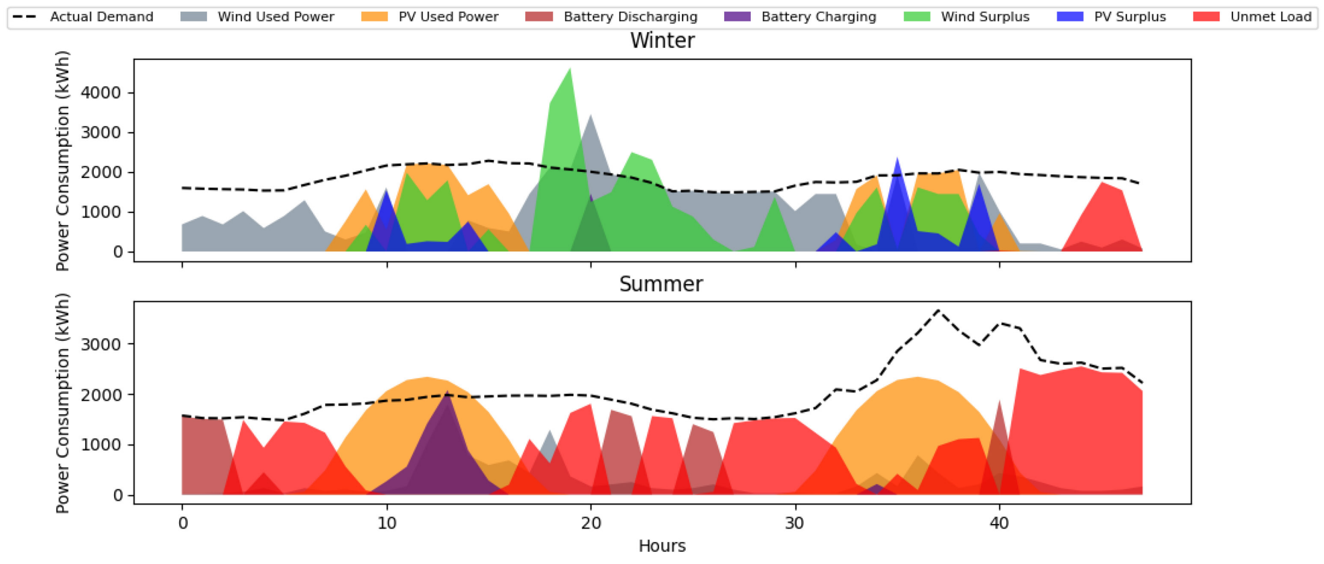


Figure 5.31. Optimal Schedule using actual load demand

The other noticeable trend is the amount of unmet load in summer, which is significant in most hours. The notable volume of the loss of power supply in summer drastically raises the operational cost (Table 5.32). Furthermore, based on the results in Table 5.32, the unmet load in winter is also considerable and needs to be diminished.

Table 5.32. Comparison of the scheduling results for the actual, critical, and optimal loads

Coverage	Loss Occurrence (No.)	Loss (kWh)	Loss Reduction (%)	LPSP	Operating Cost ($)	Cost Reduction (%)
14 and 15 of July (Summer)			**Loss Coefficient = 0.17**			
Actual Load	34	44452.86	-	0.446	8921.71	-
Critical Load	1	551.94	98	0.012	2565.01	71
Optimal Load	2	551.63	98	0.010	1938.22	78
15 and 16 of February (Winter)			**Loss Coefficient = 0.08**			
Actual Load	4	4226.17	-	0.048	5687.43	-
Critical Load	0	0	100	0	7856.48	−27
Optimal Load	0	0	100	0	4725.89	17

One of the alternatives to lessen the unmet load is using the calculated critical load to at least serve the power to vulnerable sections. Figures 5.32 and 5.33 illustrate the number of times that a loss of power could happen and its representing value in an hourly resolution in summer and winter, respectively, using actual, critical, and proposed optimal loads. According to these figures, using both critical load and optimal load can reduce the number of power loss occurrences and their values to zero in winter and to a minimum level in summer. Although using a critical load could decrease the loss of power supply probability to a minimum level, in the winter case, it drastically increases the microgrid's operating cost (Table 5.32). This growth in operational cost is also mildly observed in summer. The curtailed renewable power growth could explain the increase in operation cost in the winter during different hours.

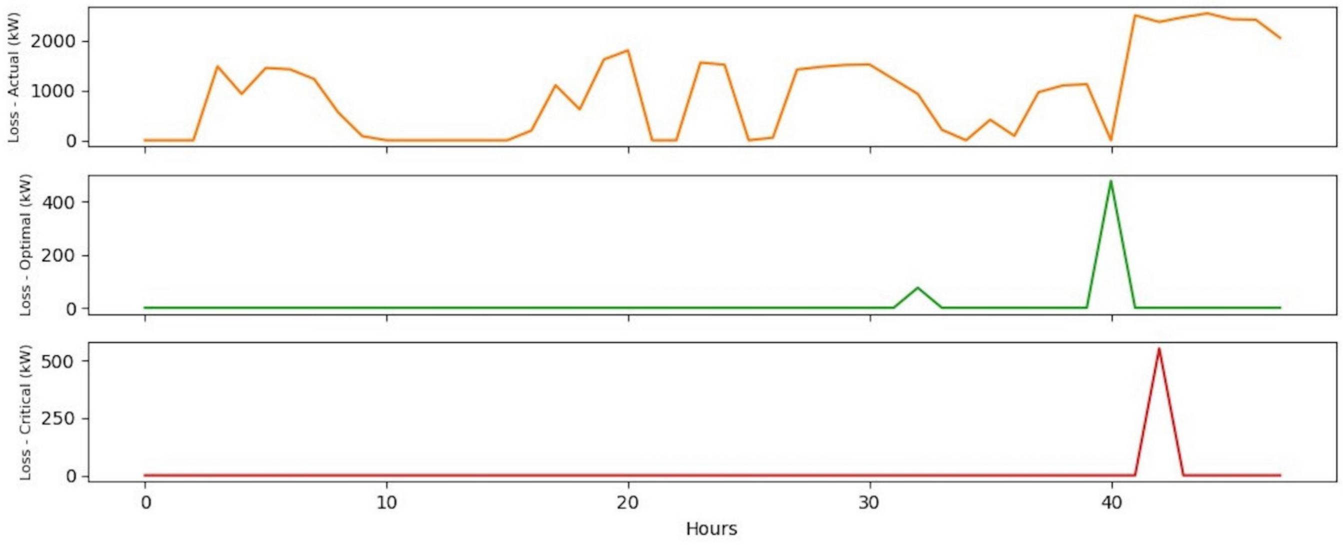


Figure 5.32. Loss occurrence and value in Summer

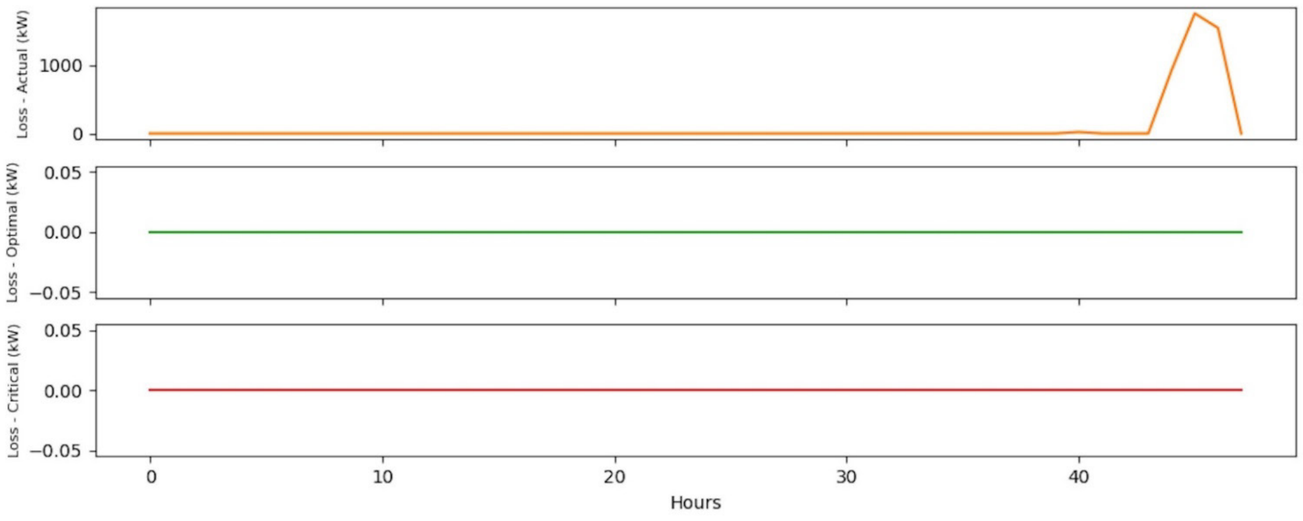


Figure 5.33. Loss occurrence and value in winter

Therefore, to tackle the challenge of having a surplus power penalty caused by using the critical load, the optimal load needs to be calculated by the MILP model to not only bring down the loss of power supply probability but also minimize the operating cost of the system. The optimal loads evaluated by the MILP model for both summer and winter are shown in Figure 5.34.

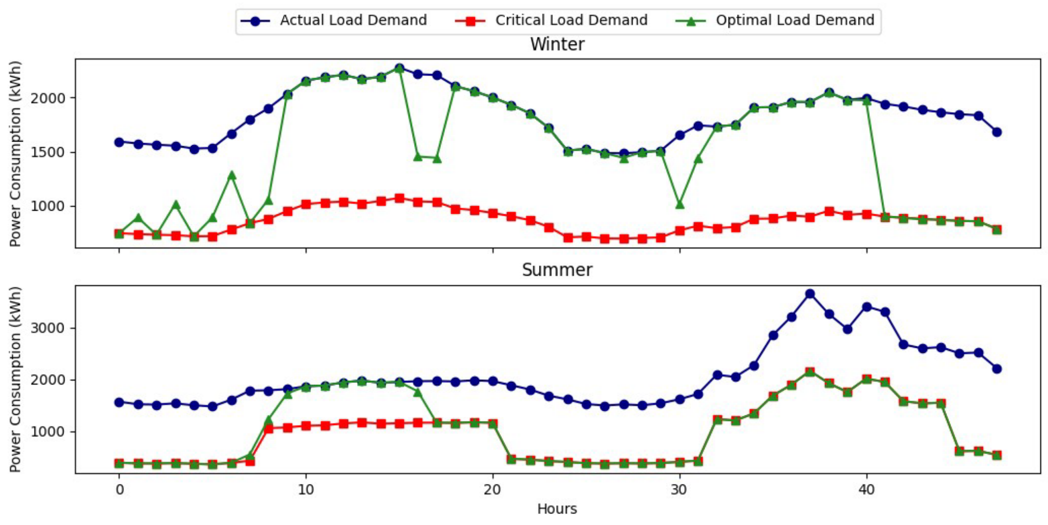


Figure 5.34. Optimal, actual, and critical loads for summer and winter

It is evident from the results that, in winter, the optimal load fluctuates more between the critical and actual loads and tends toward the actual load. In summer, there is just one fluctuation, which leans toward the critical load. This could be justified by the amount of renewable generation on different days and the actual load demand. For example, on the second summer day, the actual load demand rises while the amount of renewable generation is insufficient (Figure 5.31). Therefore, the optimal load tends to be the critical load on this day since this could be the lower limit for the optimal load. The loss value using actual, optimal, and critical loads are depicted in Figures 5.32 and 5.33 for summer and winter, respectively.

The optimum scheduling results of using the optimal load that the optimization model calculates are shown in Figure 5.35. Based on this figure and Table 5.32, the amount of unmet load in winter is zero and in summer is near zero, while the amount of added surplus power in both seasons is not very considerable compared to the actual load schedule. This will cause a meaningful reduction in the operating cost of the microgrid. Moreover, comparing the optimal load schedule (Figure 5.35) with the schedule corresponding to the actual load reveals that the amount of battery charge-discharge increases (not significantly) in the case of optimal load in both summer and winter, slightly raising the operational cost.

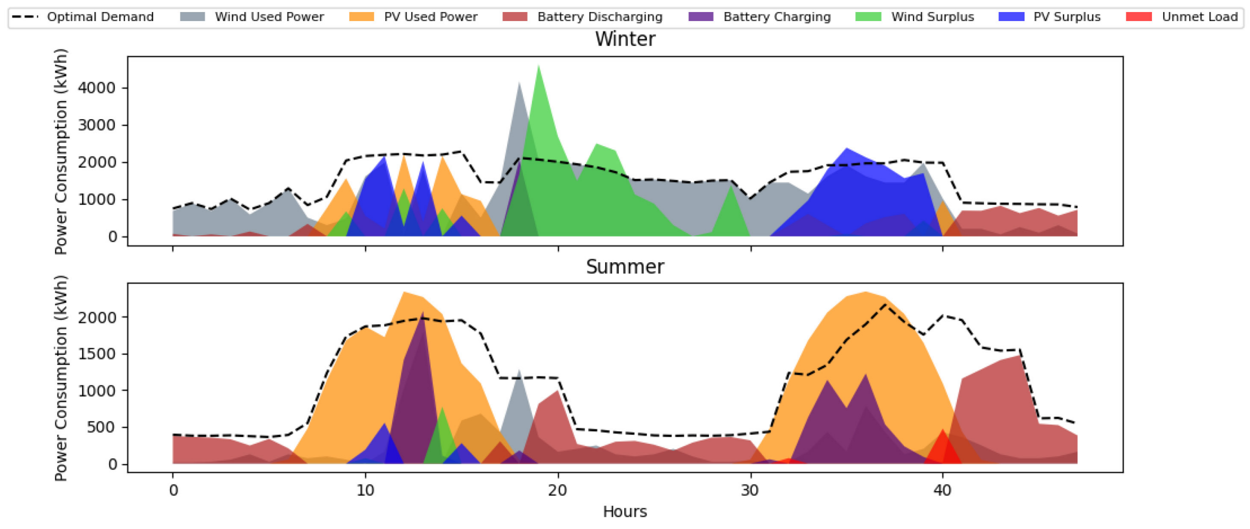


Figure 5.35. Optimal Schedule using optimal load demand

Based on the results demonstrated in Table 5.32, employing the optimal load proposed by the optimization model in summer and winter could bring down the operating cost of the microgrid in the off-grid mode by about 78% and 17%, respectively (loss and cost reductions are reported in Table 5.32 compared to the actual load coverage). Moreover, it could lessen the LPSP to near zero (0.010) in summer and drop it to zero in winter. Although using optimal load has reduced the operating cost of the microgrid in winter to 4725.89$ from 5687.43$, this change is not significant compared to that in summer (reducing operating cost from 8921.71$ to 1938.22$). This is due to variations in renewable generation over those randomly selected days in summer and winter. Since the amount of wind power generation for the chosen day in summer is much less than that of the selected day in winter, the amount of unmet load in summer is considerably higher (44,452.86 kWh) compared to winter (4226.17 kWh).

Chapter 6: Conclusion and Future Suggestions

6.1. Summary

This thesis investigates the possibility of designing an urban microgrid in an optimum way by developing a framework with three modules to address the related barriers and challenges. A regional generation strategy with a virtual power plant concept is proposed in the first module to resolve the low renewable penetration caused by space limitations. A mapping model is integrated with the design model to explore the optimum location of the regional generation plant by considering different aspects, such as the amount of available wind speed and the area which could be used for the installation of components. For robust operation scheduling, the load demand and renewable resources are required to be forecasted accurately. Therefore, the second module presents prediction strategies for removing the uncertainties of the optimal operation by hybridizing different models such as LSTM, NWP, and SARIMAX models. Moreover, the last module explores the resilience-oriented methods for enhancing the reliability of the designed system in case of grid power failure. An optimal load is calculated in this module by considering the critical load of the building to improve the resiliency and minimize the unmet load while decreasing the operation cost of the designed system by reducing the surplus power.

Then the developed methodology was applied to a case study (Concordia University's EV building) in Montreal. In the first module, since the EV building is located in the downtown area and with space limitations, the central grid dependency of the local generation design was relatively high (62% self-sufficiency), and the LCOE was also higher than the average grid purchasing price. The other important issue was the considerable value of gas consumption in the EV building for heating purposes using gas boilers. However, substituting natural gas with clean energy was not practicable since the maximum available area for installing renewable components has already been used. Therefore, a regional generation idea was proposed in this thesis to not only alleviate the grid dependency but also replace natural gas with renewable resources. In the regional generation scenario, a wind farm is designed outside of the microgrid, and the optimum location to install the wind farm was also selected using a developed mapping model. The MINLP optimization model selects the land based on several aspects, such as land cost, the number of turbines that could be installed in that zone, and the distance to the nearest transformer. Considering electric boilers already installed in Concordia for heat generation instead of gas boilers, the regional generation system increased the renewable penetration from 62% in the local generation to 81%, reducing the grid dependency. However, the LCOE of the system (0.099 CAD/kWh) was still higher than the current average grid price (0.064 CAD/kWh). The possibility of installing an air-source heat pump instead of an electric boiler was also investigated, which resulted in economic improvement and a rise in renewable penetration.

Although the regional generation considerably impacted the environmental emissions by removing about 4.2 million kilograms of CO2 per year, the LCOE of the designed system are still high and the grid dependency is still not at its minimum level. Therefore, instead of using the central grid monthly prices, a dynamic pricing strategy was developed and used for regional generation design that caused a considerable reduction in LCOE (for example, in the heat pump scenario, the LCOE reduced to 0.053 CAD/kWh from 0.082 CAD/kWh). Moreover, the renewable penetration increased significantly to 94 and 92 percent using the heat pump and electric boilers scenarios, respectively.

The second module studied the possibility of reducing the forecasting method's error. Since renewable resources and electrical load demand are following unexpected behaviors, even short-term forecasting is not straightforward. Therefore, in this thesis, hybridized models were proposed to solve this problem. For instance, for renewable resources, since they have fluctuating behavior (for example due to cloud movement for solar irradiance), the result of a physical model (Numeric Weather Prediction) was hybridized with a deep learning model to increase the accuracy of the forecasts.

For wind speed forecasting, due to high fluctuations of wind speed and difficulties in finding a genuine daily trend and seasonality in the historical data, this thesis aims to propose a hybrid wind speed forecasting model utilizing deep learning, probability distributions, and numeric weather prediction methods capable of reducing the forecasting error as much as possible. The results showed that although the LSTM model has several layers for remembering and forgetting past values, practically, it is incapable of accurate prediction, especially when there is a sudden peak/unexpected change. The first proposed model that was created by integrating the Weibull distribution probabilities with the single LSTM model reduced the error significantly (The average RMSE for three different prediction horizons during the year decreased by about 28%). To consider unexpected behavior of wind that was not reflected in the historical data, the results of the numerical weather prediction model were also hybridized with the LSTM-Weibull integrated model. The results indicated that the final hybridized model reduces the average RMSE of the single LSTM model prediction by about 32% and can be effective in a more accurate prediction of fluctuations that happen in the central peaks.

Furthermore, for solar irradiance forecasting, three different forecasting scenarios (Clear-sky, scattered, and overcast) were considered. The results show that the proposed hybrid model could significantly improve the forecasting accuracy in the presence of clouds in the sky (in scattered and overcast days). On clear sky days, the inaccuracy is high in all models; however, the proposed model can still make stronger predictions compared with the other two single models.

Finally, in the third module, which used the first and second modules' output, the optimum operation of the microgrid with considering the resiliency aspects of it was studied. The aim was to propose an approach for reducing the risk of power failure in urban microgrids by improving resilience while minimizing operating costs. In particular, employing a two-step process is proposed to reduce the cost while improving reliability. Step 1 considers a penalty for loss and calculates the optimum penalty factor, and Step 2 finds an optimal load demand that can be covered by the microgrid during the off-grid mode. The results indicate that with the proposed method, the LPSP of the system could be significantly reduced to near zero, and the amount of loss could drastically diminish (17% and 78% in winter and summer, respectively). Furthermore, using optimal load, the amount of curtailed renewable power was controlled and remained at the minimum level. The proposed method also minimized the system's operating cost compared to other scenarios.

The developed tool in this thesis could be used by both prosumers with different scales who are experiencing challenges in urban areas, and central grid decision-makers to better design their systems. Furthermore, the output of this thesis could be used by Hydro Quebec to amend its electricity tariff for large consumers that tend to be prosumers in the near future. This tool is designed from the perspective of both parties (microgrid and central grid) to improve their performance. The proposed framework in this thesis is scalable and it could be used for smaller scales (only local generation) and larger scale (with considering regional generation).

6.2. Research Contributions

The research questions raised in the problem statement section could be answered and justified with the research contributions and approaches of this thesis which can be categorized based on each module as below:

- Design Module:

Although in the discussed literature, researchers tried to design an urban microgrid with minimized cost or maximum reliability, some challenges, such as increasing the renewable penetration while having a limited amount of space, especially in metropolitan areas, or increasing the reliability of the designed systems in the higher demand hours using 100% clean energy are still not investigated. Therefore, this study focuses on developing a mathematical model for optimal sizing of an urban microgrid, considering the integration of regional renewable power generation (renewable power generation in a selected area outside of the microgrid) using the virtual power plant concept along with local generation (renewable power generation inside the microgrid).

The main contribution of this section of the thesis could be summarized as 1) a feasibility study and sensitivity analysis on generating electricity outside of the urban area and transmitting it to the nearest grid power line, 2) finding the optimum configuration and size of the urban energy system for both local and regional generation, 3) proposing a mapping model to find the optimum place for installing the regional generation plant based on land type, wind turbine density, power transmission distance and cost of the land, 4) proposing dynamic pricing for electricity rate to improve the economic aspects and renewable penetration, 5) feasibility study on using heat pumps and electric boilers as the substitute for gas boiler.

- Prediction Module:
  Wind Speed Prediction:

The mentioned literature shows that AI-based methods, statistical methods, and hybrid models have been considerably used for day-ahead wind speed forecasting. However, due to the unexpected behavior of the wind and its direct relation with physical indicators, the proposed models could not be practically used where higher accuracy is required, such as operational control of a microgrid. Moreover, the methods such as the NWP have also been employed to predict wind speed. However, still, it only considers current physical conditions, and it cannot learn from past wind speed values and unexpected changes.

This thesis aims for the advancement of knowledge for more accurate wind speed forecasting that can be used in the operation planning of an urban microgrid. The novelty of the proposed approach is in developing a hybrid model consisting of Weibull distribution, LSTM, and NWP models to reduce the error involved with wind speed prediction using a single LSTM model by considering the distribution probability of the historical wind speed data and also the physical description of the area. The main contributions of this section, with respect to the prior literature, are as follows:

- ➢ A hybrid model is proposed to circumvent the inaccuracy of the single statistical approaches. In the proposed model, the LSTM method is used, which has several superiorities over the conventional feed-forward neural network.
- ➢ Creating a Weibull distribution of the wind speed, predicting the wind speed based on a stochastic approach, and combining the probability distribution of the wind speed with the LSTM model creates an integrated model with less error compared to a single LSTM and SARIMA model with exogenous variables.
- ➢ Proposing a hybrid model that includes the NWP model's result and AI models with minimum error for short-term forecasting applications (Just for clarity, every time we refer to short-term in this thesis, it means 24-72 hours forecasting).

Solar Irradiance Prediction:

Although several studies have been carried out for solar irradiance forecasting, there is still a need to development of methods that can more accurately capture solar irradiance behavior. This thesis aims at providing a methodology for precise solar irradiance forecasting in a short-term (day ahead) horizon that could contribute to better smart grid and microgrid energy management. The novelty of the proposed approach rests in the integration of deep learning and the NWP model for better forecasting results. In this sense, the main objectives and originality of this section could be summarized as follows:

- ➢ Development and application of an LSTM model as a time-series learning method for dealing with long-term dependencies

- ➢ Comparison of deep learning forecasting results with NWP solar irradiance forecasting results as single models for 3 days with the different climatic situations.
- ➢ Proposing a hybrid model integrating the results of LSTM and NWP models to better deal with unexpected trends in solar irradiance that could not be captured using historical data.

Electrical Load Prediction:

- ➢ The main contribution of this section compared to the literature is proposing a novel hybrid model for electrical load forecasting to boost the accuracy and lessen the error.

6.3. Directions for Future Research

A few suggestions which could help the enhancement of the developed framework in this thesis could be summarized as follow:

1) A crucial piece of information which is the capacity of each node (transformer) for injecting electricity into power grid lines, should be added to the developed tool. This data was not added to the regional generation model because of the lack of data; however, the tool is developed in a way to add this variable. Adding this data to the model's input layer could make the model's results more trustable and practical.
2) Moreover, a feasibility study could be carried out on adding other components, such as fuel cells (using hydrogen generation on-site and off-site) and biomass gasifier as a regional generation method.
3) Although in this thesis, the impact of generating wind energy outside of the urban areas is studied, PV model regional generation could also be implemented in future studies.
4) Assessing the feasibility of considering fuzzy instead of binary logic in the mapping model and land selection. With using fuzzy approach, the feasibility of selecting more than one location for installing the regional plant could be evaluated.
5) Since the grid power line map is also available, the feasibility of injecting electricity to the nearest grid power line instead of the nearest transformer could be studied.
6) A bilevel programming model could be developed to connect the design and operation model to each other. By considering this connection, design would not be a onetime decision anymore and it could be updated over time based on the feedback of the operation model.

7) To make the location decision (mapping) model more practical, adding other features such as the perspective of the people living in the different zones could be studied. This needs a social study on individual's opinion regarding renewable technologies living in different neighborhoods.

Bibliography

[1] S.-H. Park, Y.-S. Jang, and E.-J. Kim, 'Multi-objective optimization for sizing multi-source renewable energy systems in the community center of a residential apartment complex', *Energy Convers. Manag.*, vol. 244, p. 114446, Sep. 2021, doi: 10.1016/j.enconman.2021.114446.

[2] X. Zhang *et al.*, 'A review of urban energy systems at building cluster level incorporating renewable-energy-source (RES) envelope solutions', *Appl. Energy*, vol. 230, pp. 1034–1056, Nov. 2018, doi: 10.1016/j.apenergy.2018.09.041.

[3] Y. V. P. Kumar and R. Bhimasingu, 'Optimal sizing of microgrid for an urban community building in south India using HOMER', in *2014 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, Mumbai, India, Dec. 2014, pp. 1–6. doi: 10.1109/PEDES.2014.7042059.

[4] A. M. Abdilahi, A. H. Mohd Yatim, M. W. Mustafa, O. T. Khalaf, A. F. Shumran, and F. Mohamed Nor, 'Feasibility study of renewable energy-based microgrid system in Somaliland′s urban centers', *Renew. Sustain. Energy Rev.*, vol. 40, pp. 1048–1059, Dec. 2014, doi: 10.1016/j.rser.2014.07.150.

[5] Z. Li, M. Shahidehpour, F. Aminifar, A. Alabdulwahab, and Y. Al-Turki, 'Networked Microgrids for Enhancing the Power System Resilience', *Proc. IEEE*, vol. 105, no. 7, pp. 1289–1310, Jul. 2017, doi: 10.1109/JPROC.2017.2685558.

[6] M. Panteli, C. Pickering, S. Wilkinson, R. Dawson, and P. Mancarella, 'Power System Resilience to Extreme Weather: Fragility Modeling, Probabilistic Impact Assessment, and Adaptation Measures', *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3747–3757, Sep. 2017, doi: 10.1109/TPWRS.2016.2641463.

[7] M. Borghei and M. Ghassemi, 'Optimal planning of microgrids for resilient distribution networks', *Int. J. Electr. Power Energy Syst.*, vol. 128, p. 106682, Jun. 2021, doi: 10.1016/j.ijepes.2020.106682.

[8] L. Wen, K. Zhou, S. Yang, and X. Lu, 'Optimal load dispatch of community microgrid with deep learning based solar power and load forecasting', *Energy*, vol. 171, pp. 1053–1065, Mar. 2019, doi: 10.1016/j.energy.2019.01.075.

[9] P. Nagapurkar and J. D. Smith, 'Techno-economic optimization and social costs assessment of microgrid-conventional grid integration using genetic algorithm and Artificial Neural Networks: A case study for two US cities', *J. Clean. Prod.*, vol. 229, pp. 552–569, Aug. 2019, doi: 10.1016/j.jclepro.2019.05.005.

[10] J. Freier and V. von Loessl, 'Dynamic electricity tariffs: Designing reasonable pricing schemes for private households', *Energy Econ.*, vol. 112, p. 106146, Aug. 2022, doi: 10.1016/j.eneco.2022.106146.

[11] B. Ugwoke, S. P. Corgnati, P. Leone, R. Borchiellini, and J. M. Pearce, ‘Low emissions analysis platform model for renewable energy: Community-scale case studies in Nigeria’, *Sustain. Cities Soc.*, vol. 67, p. 102750, Apr. 2021, doi: 10.1016/j.scs.2021.102750.

[12] M. Bagheri, N. Shirzadi, E. Bazdar, and C. A. Kennedy, ‘Optimal planning of hybrid renewable energy infrastructure for urban sustainability: Green Vancouver’, *Renew. Sustain. Energy Rev.*, vol. 95, pp. 254–264, Nov. 2018, doi: 10.1016/j.rser.2018.07.037.

[13] N. Shirzadi, F. Nasiri, and U. Eicker, ‘Optimal Configuration and Sizing of an Integrated Renewable Energy System for Isolated and Grid-Connected Microgrids: The Case of an Urban University Campus’, *Energies*, vol. 13, no. 14, p. 3527, Jul. 2020, doi: 10.3390/en13143527.

[14] Y. Q. Ang, A. Polly, A. Kulkarni, G. B. Chambi, M. Hernandez, and M. N. Haji, ‘Multi-objective optimization of hybrid renewable energy systems with urban building energy modeling for a prototypical coastal community’, *Renew. Energy*, p. S0960148122014859, Oct. 2022, doi: 10.1016/j.renene.2022.09.126.

[15] L. Ji, Y. Wu, Y. Liu, L. Sun, Y. Xie, and G. Huang, ‘Optimizing design and performance assessment of a community-scale hybrid power system with distributed renewable energy and flexible demand response’, *Sustain. Cities Soc.*, vol. 84, p. 104042, Sep. 2022, doi: 10.1016/j.scs.2022.104042.

[16] M. R. Cremi, A. M. Pantaleo, K. H. van Dam, and N. Shah, ‘Optimal design and operation of an urban energy system applied to the Fiera Del Levante exhibition centre’, *Appl. Energy*, vol. 275, p. 115359, Oct. 2020, doi: 10.1016/j.apenergy.2020.115359.

[17] J. Liu, S. Cao, X. Chen, H. Yang, and J. Peng, ‘Energy planning of renewable applications in high-rise residential buildings integrating battery and hydrogen vehicle storage’, *Appl. Energy*, vol. 281, p. 116038, Jan. 2021, doi: 10.1016/j.apenergy.2020.116038.

[18] S.-G. Yoon and S.-G. Kang, ‘Economic Microgrid Planning Algorithm with Electric Vehicle Charging Demands’, *Energies*, vol. 10, no. 10, p. 1487, Sep. 2017, doi: 10.3390/en10101487.

[19] S. Mohseni, A. C. Brent, and D. Burmester, ‘A comparison of metaheuristics for the optimal capacity planning of an isolated, battery-less, hydrogen-based micro-grid’, *Appl. Energy*, vol. 259, p. 114224, Feb. 2020, doi: 10.1016/j.apenergy.2019.114224.

[20] H. Mehrjerdi, A. Iqbal, E. Rakhshani, and J. R. Torres, ‘Daily-seasonal operation in net-zero energy building powered by hybrid renewable energies and hydrogen storage systems’, *Energy Convers. Manag.*, vol. 201, p. 112156, Dec. 2019, doi: 10.1016/j.enconman.2019.112156.

[21] A. R. Bhatti, Z. Salam, and R. H. Ashique, ‘Electric Vehicle Charging Using Photovoltaic based Microgrid for Remote Islands’, *Energy Procedia*, vol. 103, pp. 213–218, Dec. 2016, doi: 10.1016/j.egypro.2016.11.275.

[22] N. Augustine, S. Suresh, P. Moghe, and K. Sheikh, ‘Economic dispatch for a microgrid considering renewable energy cost functions’, in *2012 IEEE PES Innovative Smart Grid Technologies (ISGT)*, Washington, DC, USA, Jan. 2012, pp. 1–7. doi: 10.1109/ISGT.2012.6175747.

[23] S. Conti, R. Nicolosi, S. A. Rizzo, and H. H. Zeineldin, ‘Optimal Dispatching of Distributed Generators and Storage Systems for MV Islanded Microgrids’, *IEEE Trans. Power Deliv.*, vol. 27, no. 3, pp. 1243–1251, Jul. 2012, doi: 10.1109/TPWRD.2012.2194514.

[24] M. Marzband, M. Ghadimi, A. Sumper, and J. L. Domínguez-García, ‘Experimental validation of a real-time energy management system using multi-period gravitational search

algorithm for microgrids in islanded mode', *Appl. Energy*, vol. 128, pp. 164–174, Sep. 2014, doi: 10.1016/j.apenergy.2014.04.056.

[25] H. Kanchev, F. Colas, V. Lazarov, and B. Francois, 'Emission Reduction and Economical Optimization of an Urban Microgrid Operation Including Dispatched PV-Based Active Generators', *IEEE Trans. Sustain. Energy*, vol. 5, no. 4, pp. 1397–1405, Oct. 2014, doi: 10.1109/TSTE.2014.2331712.

[26] A. Kavousi-Fard, M.-R. Akbari-Zadeh, F. Kavousi-Fard, and M.-A. Rostami, 'Effect of wind turbine on the economic load dispatch problem considering the wind speed uncertainty', *J. Intell. Fuzzy Syst.*, vol. 28, no. 2, pp. 693–705, 2015, doi: 10.3233/IFS-141350.

[27] L. Sun, Q. Xu, X. Chen, and Y. Fan, 'Day-ahead economic dispatch of microgrid based on game theory', *Energy Rep.*, vol. 6, pp. 633–638, Feb. 2020, doi: 10.1016/j.egyr.2019.11.131.

[28] X. Lu, Z. Liu, L. Ma, L. Wang, K. Zhou, and N. Feng, 'A robust optimization approach for optimal load dispatch of community energy hub', *Appl. Energy*, vol. 259, p. 114195, Feb. 2020, doi: 10.1016/j.apenergy.2019.114195.

[29] H. Yang, Z. Gong, Y. Ma, L. Wang, and B. Dong, 'Optimal two-stage dispatch method of household PV-BESS integrated generation system under time-of-use electricity price', *Int. J. Electr. Power Energy Syst.*, vol. 123, p. 106244, Dec. 2020, doi: 10.1016/j.ijepes.2020.106244.

[30] A. Kialashaki and J. R. Reisel, 'Modeling of the energy demand of the residential sector in the United States using regression models and artificial neural networks', *Appl. Energy*, vol. 108, pp. 271–280, Aug. 2013, doi: 10.1016/j.apenergy.2013.03.034.

[31] R. Nageem and J. R, 'Predicting the Power Output of a Grid-Connected Solar Panel Using Multi-Input Support Vector Regression', *Procedia Comput. Sci.*, vol. 115, pp. 723–730, 2017, doi: 10.1016/j.procs.2017.09.143.

[32] A. Rahman, V. Srikumar, and A. D. Smith, 'Predicting electricity consumption for commercial and residential buildings using deep recurrent neural networks', *Appl. Energy*, vol. 212, pp. 372–385, Feb. 2018, doi: 10.1016/j.apenergy.2017.12.051.

[33] J. Wang and S. Xiong, 'A hybrid forecasting model based on outlier detection and fuzzy time series – A case study on Hainan wind farm of China', *Energy*, vol. 76, pp. 526–541, Nov. 2014, doi: 10.1016/j.energy.2014.08.064.

[34] E. Yatiyana, S. Rajakaruna, and A. Ghosh, 'Wind speed and direction forecasting for wind power generation using ARIMA model', in *2017 Australasian Universities Power Engineering Conference (AUPEC)*, Melbourne, VIC, Nov. 2017, pp. 1–6. doi: 10.1109/AUPEC.2017.8282494.

[35] R. G. Kavasseri and K. Seetharaman, 'Day-ahead wind speed forecasting using f-ARIMA models', *Renew. Energy*, vol. 34, no. 5, pp. 1388–1393, May 2009, doi: 10.1016/j.renene.2008.09.006.

[36] J. Wang, J. Hu, K. Ma, and Y. Zhang, 'A self-adaptive hybrid approach for wind speed forecasting', *Renew. Energy*, vol. 78, pp. 374–385, Jun. 2015, doi: 10.1016/j.renene.2014.12.074.

[37] S. An, H. Shi, Q. Hu, X. Li, and J. Dang, 'Fuzzy rough regression with application to wind speed prediction', *Inf. Sci.*, vol. 282, pp. 388–400, Oct. 2014, doi: 10.1016/j.ins.2014.03.090.

[38] A. Ul Haque and J. Meng, 'Short-Term Wind Speed Forecasting Based On Fuzzy Artmap', *Int. J. Green Energy*, vol. 8, no. 1, pp. 65–80, Feb. 2011, doi: 10.1080/15435075.2010.529784.

[39] J. Zhou, J. Shi, and G. Li, ‘Fine tuning support vector machines for short-term wind speed forecasting’, *Energy Convers. Manag.*, vol. 52, no. 4, pp. 1990–1998, Apr. 2011, doi: 10.1016/j.enconman.2010.11.007.

[40] K. R. Nair, ‘Forecasting of wind speed using ANN, ARIMA and Hybrid Models’, p. 6, 2017.

[41] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, ‘Time series forecasting using a deep belief network with restricted Boltzmann machines’, *Neurocomputing*, vol. 137, pp. 47–56, Aug. 2014, doi: 10.1016/j.neucom.2013.03.047.

[42] H. Z. Wang, G. B. Wang, G. Q. Li, J. C. Peng, and Y. T. Liu, ‘Deep belief network based deterministic and probabilistic wind speed forecasting approach’, *Appl. Energy*, vol. 182, pp. 80–93, Nov. 2016, doi: 10.1016/j.apenergy.2016.08.108.

[43] H. Wang, G. Li, G. Wang, J. Peng, H. Jiang, and Y. Liu, ‘Deep learning based ensemble approach for probabilistic wind power forecasting’, *Appl. Energy*, vol. 188, pp. 56–70, Feb. 2017, doi: 10.1016/j.apenergy.2016.11.111.

[44] S. A. Khadem and A. D. Rey, ‘Nucleation and growth of cholesteric collagen tactoids: A time-series statistical analysis based on integration of direct numerical simulation (DNS) and long short-term memory recurrent neural network (LSTM-RNN)’, *J. Colloid Interface Sci.*, vol. 582, pp. 859–873, Jan. 2021, doi: 10.1016/j.jcis.2020.08.052.

[45] H. Liu, X. Mi, and Y. Li, ‘Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network’, *Energy Convers. Manag.*, vol. 156, pp. 498–514, Jan. 2018, doi: 10.1016/j.enconman.2017.11.053.

[46] J. Jung and R. P. Broadwater, ‘Current status and future advances for wind speed and power forecasting’, *Renew. Sustain. Energy Rev.*, vol. 31, pp. 762–777, Mar. 2014, doi: 10.1016/j.rser.2013.12.054.

[47] J. Bessac, E. Constantinescu, and M. Anitescu, ‘Stochastic simulation of predictive space–time scenarios of wind speed using observations and physical model outputs’, *Ann. Appl. Stat.*, vol. 12, no. 1, Mar. 2018, doi: 10.1214/17-AOAS1099.

[48] S. Hu *et al.*, ‘Hybrid forecasting method for wind power integrating spatial correlation and corrected numerical weather prediction’, *Appl. Energy*, vol. 293, p. 116951, Jul. 2021, doi: 10.1016/j.apenergy.2021.116951.

[49] A. Kumler, Y. Xie, and Y. Zhang, ‘A New Approach for Short-Term Solar Radiation Forecasting Using the Estimation of Cloud Fraction and Cloud Albedo’, NREL/TP--5D00-72290, 1476449, Oct. 2018. doi: 10.2172/1476449.

[50] S. Das, ‘Short term forecasting of solar radiation and power output of 89.6kWp solar PV power plant’, *Mater. Today Proc.*, vol. 39, pp. 1959–1969, 2021, doi: 10.1016/j.matpr.2020.08.449.

[51] G. Reikard, ‘Predicting solar radiation at high resolutions: A comparison of time series forecasts’, *Sol. Energy*, vol. 83, no. 3, pp. 342–349, Mar. 2009, doi: 10.1016/j.solener.2008.08.007.

[52] A. Mellit and A. M. Pavan, ‘A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy’, *Sol. Energy*, vol. 84, no. 5, pp. 807–821, May 2010, doi: 10.1016/j.solener.2010.02.006.

[53] Y. Kemmoku, S. Orita, S. Nakagawa, and T. Sakakibara, ‘DAILY INSOLATION FORECASTING USING A MULTI-STAGE NEURAL NETWORK’, *Sol. Energy*, vol. 66, no. 3, pp. 193–199, Jun. 1999, doi: 10.1016/S0038-092X(99)00017-1.

[54] Y. Sun, G. Szucs, and A. R. Brandt, 'Solar PV output prediction from video streams using convolutional neural networks', *Energy Environ. Sci.*, vol. 11, no. 7, pp. 1811–1818, 2018, doi: 10.1039/c7ee03420b.

[55] X. Qing and Y. Niu, 'Hourly day-ahead solar irradiance prediction using weather forecasts by LSTM', *Energy*, vol. 148, pp. 461–468, Apr. 2018, doi: 10.1016/j.energy.2018.01.177.

[56] S. Ghimire, R. C. Deo, N. Raj, and J. Mi, 'Deep solar radiation forecasting with convolutional neural network and long short-term memory network algorithms', *Appl. Energy*, vol. 253, p. 113541, Nov. 2019, doi: 10.1016/j.apenergy.2019.113541.

[57] K. Yan, H. Shen, L. Wang, H. Zhou, M. Xu, and Y. Mo, 'Short-Term Solar Irradiance Forecasting Based on a Hybrid Deep Learning Methodology', *Information*, vol. 11, no. 1, p. 32, Jan. 2020, doi: 10.3390/info11010032.

[58] M. Guermoui, F. Melgani, K. Gairaa, and M. L. Mekhalfi, 'A comprehensive review of hybrid models for solar radiation forecasting', *J. Clean. Prod.*, vol. 258, p. 120357, Jun. 2020, doi: 10.1016/j.jclepro.2020.120357.

[59] E. Lorenz and D. Heinemann, 'Prediction of Solar Irradiance and Photovoltaic Power', in *Comprehensive Renewable Energy*, Elsevier, 2012, pp. 239–292. doi: 10.1016/B978-0-08-087872-0.00114-1.

[60] M. A. Hossain, H. R. Pota, M. J. Hossain, and F. Blaabjerg, 'Evolution of microgrids with converter-interfaced generations: Challenges and opportunities', *Int. J. Electr. Power Energy Syst.*, vol. 109, pp. 160–186, Jul. 2019, doi: 10.1016/j.ijepes.2019.01.038.

[61] S. Mishra, K. Anderson, B. Miller, K. Boyer, and A. Warren, 'Microgrid resilience: A holistic approach for assessing threats, identifying vulnerabilities, and designing corresponding mitigation strategies', *Appl. Energy*, vol. 264, p. 114726, Apr. 2020, doi: 10.1016/j.apenergy.2020.114726.

[62] M. Borghei and M. Ghassemi, 'Optimal planning of microgrids for resilient distribution networks', *Int. J. Electr. Power Energy Syst.*, vol. 128, p. 106682, Jun. 2021, doi: 10.1016/j.ijepes.2020.106682.

[63] R. Rigo-Mariani, B. Sareni, X. Roboam, and C. Turpin, 'Optimal power dispatching strategies in smart-microgrids with storage', *Renew. Sustain. Energy Rev.*, vol. 40, pp. 649–658, Dec. 2014, doi: 10.1016/j.rser.2014.07.138.

[64] X. Kong, J. Xiao, D. Liu, J. Wu, C. Wang, and Y. Shen, 'Robust stochastic optimal dispatching method of multi-energy virtual power plant considering multiple uncertainties', *Appl. Energy*, vol. 279, p. 115707, Dec. 2020, doi: 10.1016/j.apenergy.2020.115707.

[65] H. Daneshi and H. Khorashadi-Zadeh, 'Microgrid energy management system: A study of reliability and economic issues', in *2012 IEEE Power and Energy Society General Meeting*, San Diego, CA, Jul. 2012, pp. 1–5. doi: 10.1109/PESGM.2012.6344957.

[66] P. M. Costa and M. A. Matos, 'Economic Analysis of Microgrids Including Reliability Aspects', in *2006 International Conference on Probabilistic Methods Applied to Power Systems*, Stockholm, Sweden, Jun. 2006, pp. 1–8. doi: 10.1109/PMAPS.2006.360236.

[67] A. Hussain, V.-H. Bui, and H.-M. Kim, 'Resilience-Oriented Optimal Operation of Networked Hybrid Microgrids', *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 204–215, Jan. 2019, doi: 10.1109/TSG.2017.2737024.

[68] H. Zakernezhad, M. S. Nazar, M. Shafie-khah, and J. P. S. Catalão, 'Optimal resilient operation of multi-carrier energy systems in electricity markets considering distributed energy resource aggregators', *Appl. Energy*, vol. 299, p. 117271, Oct. 2021, doi: 10.1016/j.apenergy.2021.117271.

[69] J. Tobajas, F. Garcia-Torres, P. Roncero-Sánchez, J. Vázquez, L. Bellatreche, and E. Nieto, 'Resilience-oriented schedule of microgrids with hybrid energy storage system using model predictive control', *Appl. Energy*, vol. 306, p. 118092, Jan. 2022, doi: 10.1016/j.apenergy.2021.118092.

[70] J. Kratochvil, W. Boyson, and D. King, 'Photovoltaic array performance model.', SAND2004-3535, 919131, Aug. 2004. doi: 10.2172/919131.

[71] N. M. Kumar, P. R. K. Reddy, and K. Praveen, 'Optimal energy performance and comparison of open rack and roof mount mono c-Si photovoltaic Systems', *Energy Procedia*, vol. 117, pp. 136–144, Jun. 2017, doi: 10.1016/j.egypro.2017.05.116.

[72] W. Boyson, G. Galbraith, D. King, and S. Gonzalez, 'Performance model for grid-connected photovoltaic inverters.', SAND2007-5036, 920449, Sep. 2007. doi: 10.2172/920449.

[73] Y. Shen *et al.*, 'Coordinated optimal control of active power of wind farms considering wake effect', *Energy Rep.*, vol. 8, pp. 84–90, Apr. 2022, doi: 10.1016/j.egyr.2021.11.132.

[74] P. Nørgaard and H. Holttinen, 'A Multi-Turbine Power Curve Approach', p. 6.

[75] R. C. Bansal, T. S. Bhatti, and D. P. Kothari, 'On some of the design aspects of wind energy conversion systems', *Energy Convers. Manag.*, vol. 43, no. 16, pp. 2175–2187, Nov. 2002, doi: 10.1016/S0196-8904(01)00166-2.

[76] S. Rehman and N. M. Al-Abbadi, 'Wind shear coefficients and their effect on energy production', *Energy Convers. Manag.*, vol. 46, no. 15–16, pp. 2578–2591, Sep. 2005, doi: 10.1016/j.enconman.2004.12.005.

[77] G. M. Masters, 'Renewable and Efficient Electric Power Systems', p. 676.

[78] E. Hau, *Wind power systems. Fundamentals, technology, applications, economic aspects. 4.* 2008. [Online]. Available: https://doi.org/10.1007/978-3-540-72151-2

[79] N. R. Canada, 'Maps', May 30, 2019. https://www.nrcan.gc.ca/maps-tools-and-publications/maps/22020 (accessed Nov. 21, 2022).

[80] C. R. Agency, 'FCN10 Regulations Amending the Fuel Charge Regulations Made Under the Greenhouse Gas Pollution Pricing Act', May 04, 2021. https://www.canada.ca/en/revenue-agency/services/forms-publications/publications/fcn10.html (accessed Nov. 28, 2022).

[81] N. S. Arunraj, D. Ahrens, and M. Fernandes, 'Application of SARIMAX Model to Forecast Daily Sales in Food Retail Industry':, *Int. J. Oper. Res. Inf. Syst.*, vol. 7, no. 2, pp. 1–21, Apr. 2016, doi: 10.4018/IJORIS.2016040101.

[82] Y. Chen and S. Tjandra, 'Daily Collision Prediction with SARIMAX and Generalized Linear Models on the Basis of Temporal and Weather Variables', *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2432, no. 1, pp. 26–36, Jan. 2014, doi: 10.3141/2432-04.

[83] D. B. Alencar, C. M. Affonso, R. C. L. Oliveira, and J. C. R. Filho, 'Hybrid Approach Combining SARIMA and Neural Networks for Multi-Step Ahead Wind Speed Forecasting in Brazil', *IEEE Access*, vol. 6, pp. 55986–55994, 2018, doi: 10.1109/ACCESS.2018.2872720.

[84] R. Jozefowicz, W. Zaremba, and I. Sutskever, 'An Empirical Exploration of Recurrent Network Architectures', p. 9.

[85] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber, 'A Clockwork RNN'. arXiv, Feb. 14, 2014. Accessed: Nov. 21, 2022. [Online]. Available: http://arxiv.org/abs/1402.3511

[86] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, 'LSTM: A Search Space Odyssey', *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[87] Y. Bengio, P. Simard, and P. Frasconi, ‘Learning Long-Term Dependencies with Gradient Descent is Difficult’, vol. 2, no. 2, 1994.

[88] J. Hochreiter, ‘Untersuchungen zu dynamischen neuronalen Netzen’. Institut f¨ur Informatik Technische Universit¨at M¨unchen, 1991.

[89] J. Schmidhuber and S. Hochreiter, ‘LONG SHORT-TERM MEMORY’, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[90] C. Ozay and M. S. Celiktas, ‘Statistical analysis of wind speed using two-parameter Weibull distribution in Alaçatı region’, *Energy Convers. Manag.*, vol. 121, pp. 49–54, Aug. 2016, doi: 10.1016/j.enconman.2016.05.026.

[91] A. A. Kadhem, N. Wahab, I. Aris, J. Jasni, and A. Abdalla, ‘Advanced Wind Speed Prediction Model Based on a Combination of Weibull Distribution and an Artificial Neural Network’, *Energies*, vol. 10, no. 11, p. 1744, Oct. 2017, doi: 10.3390/en10111744.

[92] F. C. Odo, S. U. Offiah, and P. E. Ugwuoke, ‘Weibull distribution-based model for prediction of wind potential in Enugu, Nigeria’, p. 7, 2012.

[93] U.S. Department of Homeland Security Risk Lexicon, ‘U.S. Department of Homeland Security Risk Lexicon’. 2008.

[94] V. Chalishazar, S. Poudel, S. Hanif, and P. Thekkumparambath Mana, ‘Power System Resilience Metrics Augmentation for Critical Load Prioritization’, PNNL--30837, 1764623, Jan. 2021. doi: 10.2172/1764623.

[95] UFC 3-540-01, ‘Unified Facilities Criteria (Ufc) Approved for Public Release; Distribution Unlimited Engine-Driven Generator Systems for Prime \1\ and Standby Power Applications /1’. Nov. 2014.

[96] ‘Where is Montreal, Quebec, Canada on Map Lat Long Coordinates’. https://www.latlong.net/place/montreal-quebec-canada-27653.html (accessed Nov. 22, 2022).

[97] ‘NASA POWER | Prediction Of Worldwide Energy Resources’. https://power.larc.nasa.gov/ (accessed Nov. 22, 2022).

[98] ‘Archidata’. https://fmis.concordia.ca/ (accessed Nov. 22, 2022).

[99] ‘Self-generation | Hydro-Québec’. http://www.hydroquebec.com/self-generation/faq.html (accessed Nov. 23, 2022).

[100] ‘Rate Flex G – Business | Hydro-Québec’. https://www.hydroquebec.com/business/customer-space/rates/rate-flex-g-billing.html (accessed Nov. 23, 2022).

[101] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, ‘pvlib python: a python package for modeling solar energy systems’, *J. Open Source Softw.*, vol. 3, no. 29, p. 884, Sep. 2018, doi: 10.21105/joss.00884.

[102] S. Haas, B. Schachler, U. Krien, and S. Bosch, ‘windpowerlib: A python library to model wind power plants (v0.1.0)’. Zenodo, Jan. 17, 2019. doi: 10.5281/zenodo.2542896.

[103] M. L. Bynum *et al.*, *Pyomo — Optimization Modeling in Python*, vol. 67. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-68928-5.

[104] K. Bestuzheva *et al.*, ‘The SCIP Optimization Suite 8.0’, p. 114.

[105] ‘CPLEX User□s Manual’, p. 596.

[106] ‘Keras: the Python deep learning API’. https://keras.io/ (accessed Nov. 22, 2022).

[107] ‘Introduction — statsmodels’. https://www.statsmodels.org/stable/index.html (accessed Nov. 22, 2022).

[108] ‘GAMS - Cutting Edge Modeling’. https://www.gams.com/ (accessed Nov. 22, 2022).

[109] '25 kW Direct-Drive Wind Turbine'. eocycle TECHNOLOGIES. [Online]. Available: eocycle.com

[110] L. Bauer, 'Enercon E-53 - 800,00 kW - Wind turbine'. https://en.wind-turbine-models.com/turbines/530-enercon-e-53 (accessed Nov. 24, 2022).

[111] M. Nasser, T. F. Megahed, S. Ookawara, and H. Hassan, 'Techno-economic assessment of clean hydrogen production and storage using hybrid renewable energy system of PV/Wind under different climatic conditions', *Sustain. Energy Technol. Assess.*, vol. 52, p. 102195, Aug. 2022, doi: 10.1016/j.seta.2022.102195.

[112] A. Malheiro, P. M. Castro, R. M. Lima, and A. Estanqueiro, 'Integrated sizing and scheduling of wind/PV/diesel/battery isolated systems', *Renew. Energy*, vol. 83, pp. 646–657, Nov. 2015, doi: 10.1016/j.renene.2015.04.066.

[113] L. Wen, K. Zhou, S. Yang, and X. Lu, 'Optimal load dispatch of community microgrid with deep learning based solar power and load forecasting', *Energy*, vol. 171, pp. 1053–1065, Mar. 2019, doi: 10.1016/j.energy.2019.01.075.

[114] C. E. R. Government of Canada, 'CER – Provincial and Territorial Energy Profiles – Quebec', Jul. 28, 2022. https://www.cer-rec.gc.ca/en/data-analysis/energy-markets/provincial-territorial-energy-profiles/provincial-territorial-energy-profiles-quebec.html (accessed Nov. 25, 2022).

[115] 'Natural gas properties | Énergir'. https://www.energir.com/en/major-industries/natural-gas-quebec/natural-gas-properties/ (accessed Dec. 01, 2022).

[116] E. K. Vakkilainen, '3 - Boiler Processes', in *Steam Generation from Biomass*, Butterworth-Heinemann, 2017, pp. 57–86. [Online]. Available: https://doi.org/10.1016/B978-0-12-804389-9.00003-4.

[117] G. M. Masters, 'Renewable and Efficient Electric Power Systems', p. 676.

[118] '2022 Heat Pump Cost | Installation & Replacement Prices', *HomeGuide*. https://homeguide.com/costs/heat-pump-cost (accessed Nov. 27, 2022).

[119] Technical Support Division, Office of Air Quality Planning and Standards, U. S. Environmental Protection Agency, 'Emission Factor Documentation for AP-42 Section 1.4—Natural Gas Combustion'. 1997.

[120] I. M. Lisitano, A. Biglia, E. Fabrizio, and M. Filippi, 'Building for a Zero Carbon future: trade-off between carbon dioxide emissions and primary energy approaches', *Energy Procedia*, vol. 148, pp. 1074–1081, Aug. 2018, doi: 10.1016/j.egypro.2018.08.052.

[121] R. Stull, *Practical Meteorology: An Algebra-based Survey of Atmospheric Science*. 2017.

[122] 'Rate Flex G – Business | Hydro-Québec'. https://www.hydroquebec.com/business/customer-space/rates/rate-flex-g.html (accessed Nov. 29, 2022).

[123] 'Dynamic pricing – Business customers | Hydro-Québec'. https://www.hydroquebec.com/business/customer-space/rates/dynamic-pricing.html (accessed Nov. 29, 2022).

[124] L. A. Wolsey, 'Mixed Integer Programming', in *Wiley Encyclopedia of Computer Science and Engineering*, John Wiley & Sons, Ltd, 2008, pp. 1–10. doi: 10.1002/9780470050118.ecse244.

Appendices

Related written python codes and samples from the used datasets are provided in this section.

Appendix I: Python Codes

PV Model

```
import numpy as np

import pandas as pd

import pvlib

import matplotlib.pyplot as plt

import pytz

coordinates = [(45.49, -73.57, 233, 'Etc/GMT+5')]

modules_lib = pvlib.pvsystem.retrieve_sam('SandiaMod')

module = modules_lib['Canadian_Solar_CS5P_220M___2009_']

inverters_lib = pvlib.pvsystem.retrieve_sam('cecinverter')

inverter = inverters_lib['ABB__MICRO_0_25_I_OUTD_US_208__208V_']

temperature_model_parameters=
pvlib.temperature.TEMPERATURE_MODEL_PARAMETERS['sapm']['open_rack_glass_glass'
]

naive_times = pd.date_range(start='2019', end='2020', freq='1h')

temp_air = 5 #input('input the air temperature: ')

wind_speed = 5 #input('input wind speed: ')

system = {'module': module, 'inverter': inverter,

          'surface_azimuth': 180}

#pvlib.clearsky.lookup_linke_turbidity('Etc/GMT+5',     45,     -73,     filepath=None,
interp_turbidity=True)
```

```
for latitude, longitude, altitude, timezone in coordinates:
    times = naive_times.tz_localize(timezone)
    system['surface_tilt'] = 45
    solpos = pvlib.solarposition.get_solarposition(times, latitude, longitude)
    dni_extra = pvlib.irradiance.get_extra_radiation(times)
    airmass = pvlib.atmosphere.get_relative_airmass(solpos['apparent_zenith'])
    pressure = pvlib.atmosphere.alt2pres(altitude)
    am_abs = pvlib.atmosphere.get_absolute_airmass(airmass, pressure)
    tl = pvlib.clearsky.lookup_linke_turbidity(times, latitude, longitude)
    cs = pvlib.clearsky.ineichen(solpos['apparent_zenith'], am_abs, tl,
                                 dni_extra=dni_extra, altitude=altitude)
total_irrad = pvlib.irradiance.get_total_irradiance(system['surface_tilt'],
                                                   system['surface_azimuth'],
                                                   solpos['apparent_zenith'],
                                                   solpos['azimuth'],
                                                   cs['dni'], cs['ghi'], cs['dhi'],
                                                   dni_extra=dni_extra,
                                                   model='haydavies')
    #Angle of Incidence
    aoi = pvlib.irradiance.aoi(system['surface_tilt'], system['surface_azimuth'],
                               solpos['apparent_zenith'], solpos['azimuth'])
```

```
   tcell = pvlib.temperature.sapm_cell(total_irrad['poa_global'],
                                    temp_air, wind_speed,
                                   **temperature_model_parameters)
 effective_irradiance = pvlib.pvsystem.sapm_effective_irradiance(
              total_irrad['poa_direct'], total_irrad['poa_diffuse'],
              am_abs, aoi, module)
 # sapm (Sandia Array Performance Model)
 #DC Power Output (i_sc Short Circut Module Current/ i_mp, v_mp, p_mp are module current, voltage and power at maximum power point
 DC_Power = pvlib.pvsystem.sapm(effective_irradiance, tcell, module)
#AC Power output using Sandia's grid connected model
AC_Power = pvlib.inverter.sandia(DC_Power['v_mp'], DC_Power['p_mp'], inverter)
AC_Power = pd.DataFrame(AC_Power)
AC_Power = AC_Power.rename(columns={0:'PV'})
AC_Power[AC_Power < 0] = 0
AC_Power = AC_Power.iloc[:8760,:]
AC_Power.to_csv('PV.csv')
DC_Power = DC_Power.iloc[:,:5]
plt.plot(DC_Power['i_sc'],label = 'short circuit module current')
plt.plot(DC_Power['v_oc'],label = 'open circut module voltage')
plt.plot(DC_Power['v_mp'],label = 'voltage at maximum power point')
plt.plot(DC_Power['i_mp'],label = 'current at maximum power point')
plt.plot(DC_Power['p_mp'],label = 'power at maximum power point')
plt.xlabel('Date')
plt.ylabel('Power(W),Voltage(V),Current(A)')
plt.legend(ncol=3,bbox_to_anchor=(0.5,1.05),loc='center',fontsize=9)
plt.show()
```

Single Wind Turbine Model

```
import os
import pandas as pd
import requests
from windpowerlib import ModelChain, WindTurbine, create_power_curve
from windpowerlib import data as wt
weather = pd.read_csv(
    'weather_montreal.csv',
    index_col=0,
    header=[0, 1],
    date_parser=lambda idx: pd.to_datetime(idx, utc=True))
df = wt.get_turbine_types(print_out=False)
En_Wind = {
    'turbine_type': 'E-53/800',
    'hub_height': 60
  }
e126 = WindTurbine(**En_Wind)
```

```
modelchain_data = {
   'wind_speed_model': 'hellman',          # 'logarithmic' (default),
                                           # 'hellman' or
                                           # 'interpolation_extrapolation'
   'density_model': 'ideal_gas',           # 'barometric' (default), 'ideal_gas'
                                           #  or 'interpolation_extrapolation'
   'temperature_model': 'linear_gradient', # 'linear_gradient' (def.) or
                                           # 'interpolation_extrapolation'
   'power_output_model':
      'power_coefficient_curve',           # 'power_curve' (default) or
                                           # 'power_coefficient_curve'
   'density_correction': True,             # False (default) or True
   'obstacle_height': 0,                   # default: 0
   'hellman_exp': 0.25}                    # None (default) or None


mc_e126 = ModelChain(e126, **modelchain_data).run_model(weather)
e126power_output = mc_e126.power_output
```

Wind Farm Model

```
from windpowerlib import TurbineClusterModelChain, WindTurbineCluster, WindFarm
import matplotlib.pyplot as plt
wind_turbine_fleet = pd.DataFrame(
       {'wind_turbine': [e126, None],
        'number_of_turbines': [20, None]})
        #'total_capacity': [None, 12.6e6]})
farm_data = {
   'name': 'farm_Data_2',
   'wind_turbine_fleet': [e126.to_group(1)],#,e126.to_group(total_capacity=12.6e6)],
   'efficiency': 0.9}
farm_Data_2 = WindFarm(**farm_data)
modelchain_data = {
        'wake_losses_model': 'wind_farm_efficiency',  #
                                    # 'dena_mean' (default), None,
                                    # 'wind_farm_efficiency' or name
                                    #  of another wind efficiency curve
                #  see :py:func:`~.wake_losses.get_wind_efficiency_curve`
        'smoothing': True,  # False (default) or True
        'block_width': 0.5,  # default: 0.5
        'standard_deviation_method': 'Staffell_Pfenninger',  #
                                     # 'turbulence_intensity' (default)
                                     # or 'Staffell_Pfenninger'
        'smoothing_order': 'wind_farm_power_curves',  #
                                  # 'wind_farm_power_curves' (default) or
                                  # 'turbine_power_curves'
        'wind_speed_model': 'hellman',  # 'logarithmic' (default),
                                     # 'hellman' or
                                     # 'interpolation_extrapolation'
        'density_model': 'ideal_gas',  # 'barometric' (default), 'ideal_gas' or
                                    # 'interpolation_extrapolation'
        'temperature_model': 'linear_gradient',  # 'linear_gradient' (def.) or
                                              # 'interpolation_extrapolation'
        'power_output_model': 'power_curve',  # 'power_curve' (default) or
                                            # 'power_coefficient_curve'
        'density_correction': True,  # False (default) or True
        'obstacle_height': 0,  # default: 0
        'hellman_exp': 0.1}  # None (default) or None
Power_Farm=TurbineClusterModelChain(farm_Data_2,
**modelchain_data).run_model(weather)
power_output = (Power_Farm.power_output)/1000
plt.plot(power_output, color = 'indigo')
plt.xlabel('Date')
plt.ylabel('Power (kW)')
plt.show()
```

Solar Irradiance Forecasting Model (Hybrid)

```
import pandas as pd
import numpy as np
df = pd.read_csv('New_Data.csv')
df.dropna(inplace=True)
import seaborn as sn
chart = sn.heatmap(
square=True,

cbar_kws={'fraction' : 0.01},

cmap='OrRd',

linewidth=1,

data = df.corr(),

vmin = -1,

vmax = 1,

annot = True)

 chart.set_xticklabels(chart.get_xticklabels(), rotation=0, horizontalalignment='center')

chart.set_yticklabels(chart.get_xticklabels(), rotation=90, verticalalignment='center')

training_set = df.iloc[:3432,1:4].values

test_set = df.iloc[3432:,1:4].values

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature_range=(0,1))

training_set_scaled = sc.fit_transform(training_set)

test_set_scaled = sc.fit_transform(test_set)

test_set_scaled = test_set_scaled[:,1:3]

X_train = []

y_train = []

for i in range(24, 3432):

    X_train.append(training_set_scaled[i-24:i, 0:3])

    y_train.append(training_set_scaled[i, 0])
```

```
X_train, y_train = np.array(X_train),np.array(y_train)

X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 3))

from keras import Sequential

from keras.layers import LSTM

from keras.layers import Dense

from keras.layers import Dropout

regressor = Sequential()

regressor.add(LSTM(units = 60, return_sequences = True, input_shape=(X_train.shape[1], 3)))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 60, return_sequences = True))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 60, return_sequences = True))

regressor.add(Dropout(0.2))

regressor.add(LSTM(units = 60))

regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer='adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs=100, batch_size= 32)

regressor.save('Multi_24_32_100_24May')

from keras.models import load_model

regressor=load_model('Multi_24_32_100_24May')

import matplotlib.pyplot as plt

plt.plot(range(len(regressor.history.history['loss'])),regressor.history.history['loss'])
plt.xlabel('Epoch Number')

plt.ylabel('Loss')

plt.show()

prediction_test = []

First_batch = training_set_scaled[-24:]

current_batch= First_batch.reshape((1,24,3))

for i in range (24):

    current_pred = regressor.predict(current_batch)[0]

    prediction_test.append(current_pred)

    New_var = test_set_scaled[i,:]

    New_var = New_var.reshape(1,2)

    New_test = np.insert(New_var,2,[current_pred],axis=1)

    New_test = New_test.reshape(1,1,3)

    current_batch = np.append(current_batch[:,1:,:],New_test,axis=1)

prediction_test = np.array(prediction_test)

SI = MinMaxScaler(feature_range=(0,1))

y_Scale = training_set[:,0:1]

SI.fit_transform(y_Scale)

predictions_H = SI.inverse_transform(prediction_test)

real_values = test_set[:24,0:1]

import matplotlib.pyplot as plt

plt.plot(real_values, color = 'red',label = 'Actual Solar Irradiance')

plt.plot(predictions_H, color = 'blue',label = 'Predicted Solar Irradiance')
```

```
#plt.title('RNN - Radiation Forecasting')

plt.xlabel('Time(hr)')

plt.ylabel('Solar Irradiance (W/m2)')

plt.legend()

plt.show()

import math

from sklearn.metrics import mean_squared_error

rmse = math.sqrt(mean_squared_error(real_values, predictions_H))

from sklearn.metrics import mean_absolute_error

mean_absolute_error(real_values, predictions_H)

from sklearn.metrics import r2_score

Rsqure = r2_score(real_values, predictions_H)

import tensorflow as tf

m = tf.keras.metrics.MeanSquaredLogarithmicError()

m.update_state(real_values, predictions_H)

m.result().numpy()

ShortW = df.iloc[:,2:3].values

LongW = df.iloc[:,3:4].values

plt.plot(ShortW, color = 'navy',label = 'Downward Shortwave Irradiance')

plt.plot(LongW, color = 'red',label = 'Downward Longwave Irradiance')

plt.xlabel('Time(hr)')

plt.ylabel('Solar Irradiance (W/m2)')

plt.legend()
plt.show()

# Prediction results

import matplotlib.pyplot as plt

plt.plot(real_values, color = 'red',label = 'Actual Solar Irradiance',marker = "s")

plt.plot(predictions_H, color = 'darkblue',label = 'Proposed Hybrid Model',marker = "^")

plt.plot(NWP_SW, color = 'turquoise',label = 'NWP-Shortwave',marker = "p")

plt.plot(predictions_S, color = 'chocolate',label = 'Single LSTM',marker = "o")

plt.xlabel('Time(hr)')

plt.ylabel('Solar Irradiance (W/m2)')

plt.legend()

plt.show()

test = df.iloc[:,1:2]

plt.plot(test)
```

Wind Speed Forecasting Model (Hybrid)

```
import pandas as pd

import numpy as np

df = pd.read_csv('Wind_NWP.csv')

df.dropna(inplace=True)

#import seaborn as sn

#sn.heatmap(df.corr())

training_set = df.iloc[:5064,1:].values

test_set = df.iloc[5064:5232,1:].values
```

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set)
test_set_scaled = sc.fit_transform(test_set)
test_set_scaled = test_set_scaled[:,0:4]
X_train = []
y_train = []
for i in range(24,5064):
    X_train.append(training_set_scaled[i-24:i, 0:5])
    y_train.append(training_set_scaled[i, 4])
X_train, y_train = np.array(X_train),np.array(y_train)
X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 5))
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.layers import Dropout
regressor = Sequential()
regressor.add(LSTM(units = 60, return_sequences = True, input_shape=(X_train.shape[1], 5)))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 60, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 60, return_sequences = True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units = 60))

regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer='adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs=150, batch_size= 32)

regressor.save('Hybrid_Dec')

from keras.models import load_model

regressor=load_model('Hybrid_July')

import matplotlib.pyplot as plt

plt.plot(range(len(regressor.history.history['loss'])),regressor.history.history['loss'])

plt.xlabel('Epoch Number')

plt.ylabel('Loss')

plt.show()

prediction_test = []

First_batch = training_set_scaled[-24:]

current_batch= First_batch.reshape((1,24,5))

for i in range (168):

   current_pred = regressor.predict(current_batch)[0]

   prediction_test.append(current_pred)

   New_var = test_set_scaled[i,:]

   New_var = New_var.reshape(1,4)

   New_test = np.insert(New_var,4,[current_pred],axis=1)
```

```
    New_test = New_test.reshape(1,1,5)
    current_batch = np.append(current_batch[:,1:,:],New_test,axis=1)
prediction_test = np.array(prediction_test)
SI = MinMaxScaler(feature_range=(0,1)
y_Scale = training_set[:,4:5]
SI.fit_transform(y_Scale)
predictions = SI.inverse_transform(prediction_test)
real_values = test_set[:168,4]
import matplotlib.pyplot as plt
plt.plot(real_values, color = 'navy',label = 'Actual Wind Speed')
plt.plot(predictions, color = 'slategrey',label = 'Hybrid Model Prediction',marker = 'o')
#plt.title('RNN - Wind Speed Forecasting')
plt.xlabel('Time(hr)')
plt.ylabel('Wind speed (m/s)')
plt.legend(ncol=2,bbox_to_anchor=(0.5,0.95),loc='center',fontsize=10)
plt.grid(which='minor', linewidth=0.6, alpha=0.1)
plt.show()
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(real_values, predictions))
from sklearn.metrics import mean_absolute_error
mean_absolute_error(real_values, predictions)
```

```
from sklearn.metrics import r2_score
Rsqure = r2_score(real_values, predictions)
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
MAPE = mean_absolute_percentage_error(real_values, predictions)
import tensorflow as tf
mae = tf.keras.losses.MeanAbsoluteError()
mae(real_values, predictions).numpy()
m = tf.keras.metrics.MeanSquaredLogarithmicError()
m.update_state(real_values, predictions)
m.result().numpy()
Prediction_Hybrid = pd.DataFrame(predictions)
Prediction_Hybrid.to_csv('Hybrid_Summer_168.csv')
```

Load Forecasting Model (SARIMAX)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df1 = pd.read_csv('Data-Temp.csv')
df2 = pd.read_csv('Data_EV.csv').iloc[:,5:6]
df = pd.concat([df1,df2],axis=1)
df.dropna(inplace=True)
```

```
Date = pd.date_range('jan 01 2018', periods = 17520, freq = 'H')

df.set_index(Date,drop=True,inplace=True)

df = df.iloc[:,1:]

df.index.freq = 'H'

df.dropna(inplace=True)

train = df.iloc[11640:12384,1:]

test = df.iloc[12384:13416,1:]

exo = df.iloc[:,0:1]

exo_train = exo.iloc[11640:12384]

exo_test = exo.iloc[12384:13416]

from pmdarima import auto_arima

auto_arima(df['Load'], exogenous = exo, m = 24, trace=True, surpress_warnings=True,
D=1).summary()

from statsmodels.tsa.statespace.sarimax import SARIMAX

Model = SARIMAX(train,exog = exo_train, order=(2,0,0),seasonal_order=(2, 1, 0, 24))

Model = Model.fit()

start = len(train)

end = len(train) + len(test) - 1

prediction = Model.predict(start,end,exog = exo_test)

real_values = np.array(test)

prediction = np.array(prediction)

plt.plot(real_values, color = 'red',label = 'Actual Load Demand')

plt.plot(prediction, color = 'blue',label = 'Predicted Load Demand')
plt.xlabel('Time(hr)')

plt.ylabel('Load Demand (kW)')

plt.legend()

plt.show()

import math

from sklearn.metrics import mean_squared_error

rmse = math.sqrt(mean_squared_error(real_values, prediction))

from sklearn.metrics import r2_score

Rsqure = r2_score(real_values, prediction)

def mean_absolute_percentage_error(y_true, y_pred):

    y_true, y_pred = np.array(y_true), np.array(y_pred)

    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

MAPE = mean_absolute_percentage_error(real_values, prediction)

from sklearn.metrics import mean_absolute_error

mean_absolute_error(real_values, prediction)

import tensorflow as tf

mae = tf.keras.losses.MeanAbsoluteError()

mae(real_values, prediction).numpy()

m = tf.keras.metrics.MeanSquaredLogarithmicError()

m.update_state(real_values, prediction)

m.result().numpy()
```

Load Forecasting Model (Hybrid LSTM-SARIMAX)

```
import pandas as pd

import numpy as np

df = pd.read_csv('Data_Summer.csv')

df.dropna(inplace=True)

training_set = df.iloc[:984,1:].values

test_set = df.iloc[984:,1:].values

from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature_range=(0,1))

training_set_scaled = sc.fit_transform(training_set)

test_set_scaled = sc.fit_transform(test_set)

test_set_scaled = test_set_scaled[:,0:2]

X_train = []

y_train = []

for i in range(48, 984):

    X_train.append(training_set_scaled[i-48:i, 0:3])

    y_train.append(training_set_scaled[i, 2])

X_train, y_train = np.array(X_train),np.array(y_train)

X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 3))

from keras.models import Sequential

from keras.layers import LSTM

from keras.layers import Dense

from keras.layers import Dropout
```

Initializing RNN

```
regressor = Sequential()
```

Adding the first LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 60, return_sequences = True, input_shape=(X_train.shape[1], 3)))

regressor.add(Dropout(0.2))
```

Adding the Second LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 60, return_sequences = True))

regressor.add(Dropout(0.2))
```

Adding the Third LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 60, return_sequences = True))

regressor.add(Dropout(0.2))
```

Adding the Forth LSTM layer and some Dropout regularization

```
regressor.add(LSTM(units = 60))

regressor.add(Dropout(0.2))
```

Adding output layer

```
regressor.add(Dense(units = 1))
```

Compiling the RNN

```
regressor.compile(optimizer='adam', loss = 'mean_squared_error')
```

Fitting RNN to the Training Set

```
regressor.fit(X_train, y_train, epochs=150, batch_size= 64)

regressor.save('all_features_100_64-168')

from keras.models import load_model

regressor=load_model('all_features_100_64-168')
```

```
prediction_test = []

First_batch = training_set_scaled[-48:]

current_batch= First_batch.reshape((1,48,3))

for i in range (48):

    current_pred = regressor.predict(current_batch)[0]

    prediction_test.append(current_pred)

    New_var = test_set_scaled[i,:]

    New_var = New_var.reshape(1,2)

    New_test = np.insert(New_var,2,[current_pred],axis=1)

    New_test = New_test.reshape(1,1,3)

    current_batch = np.append(current_batch[:,1:,:],New_test,axis=1)

prediction_test = np.array(prediction_test)

SI = MinMaxScaler(feature_range=(0,1))

y_Scale = test_set[:,2:3]

SI.fit_transform(y_Scale)

predictions = SI.inverse_transform(prediction_test)

real_values = test_set[:48,2:3]

import matplotlib.pyplot as plt

plt.plot(real_values, color = 'red',label = 'Actual Load Demand')

plt.plot(predictions, color = 'blue',label = 'Predicted Load Demand')

plt.xlabel('Time(hr)')

plt.ylabel('Load Demand (kW)')

plt.legend()
plt.show()
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(real_values, predictions))
from sklearn.metrics import r2_score
Rsqure = r2_score(real_values, predictions)
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
MAPE = mean_absolute_percentage_error(real_values, predictions)
from sklearn.metrics import mean_absolute_error
mean_absolute_error(real_values, predictions)
import tensorflow as tf
mae = tf.keras.losses.MeanAbsoluteError()
mae(real_values, predictions).numpy()
m = tf.keras.metrics.MeanSquaredLogarithmicError()
m.update_state(real_values, predictions)
m.result().numpy()
```

Local Optimum Design Model

```
import pyomo.environ as pyo
from pyomo.environ import *
from pyomo.opt import SolverFactory
import pandas as pd
import math

import numpy as np

import matplotlib.pyplot as plt

data = pd.read_excel("Input_Daily_Sum.xlsx")

Grid_P = []

for i in range(len(data['Load'])):

  if (data['time'][i]<=31):

      Grid_P.append(0.07005544*0.74)

  if (31<data['time'][i]<=59):

      Grid_P.append(0.06775921*0.74)

  if (59<data['time'][i]<=90):

      Grid_P.append(0.06770977*0.74)

  if (90<data['time'][i]<=120):

      Grid_P.append(0.06393077*0.74)

  if (120<data['time'][i]<=151):

      Grid_P.append(0.06694972*0.74)

  if (151<data['time'][i]<=181):

      Grid_P.append(0.06953881*0.74)

  if (181<data['time'][i]<=212):

      Grid_P.append(0.063731026*0.74)

  if (212<data['time'][i]<=243):

      Grid_P.append(0.062395918*0.74)

  if (243<data['time'][i]<=273):
      Grid_P.append(0.05969309*0.74)
  if (273<data['time'][i]<=304):
      Grid_P.append(0.059369803*0.74)
  if (304<data['time'][i]<=334):
      Grid_P.append(0.061508168*0.74)
  if (334<data['time'][i]<=365):
      Grid_P.append(0.065685064*0.74)
Grid_P = pd.DataFrame(Grid_P)
Grid_P.columns = ['Grid_Price']
data = pd.concat([data,Grid_P],axis=1)
Grid_S = []
for i in range(len(data['Load'])):
  if data['time'][i]<=90:
      Grid_S.append(0.2*0.74)
  if 90<data['time'][i]<335:
      Grid_S.append(0.1*0.74)
  if data['time'][i]>=335:
      Grid_S.append(0.2*0.74)
Grid_S = pd.DataFrame(Grid_S)
Grid_S.columns = ['Grid_Sell']
data = pd.concat([data,Grid_S],axis=1)
data.loc[data['Load']<40000, 'Loss_Coeff'] = 1
data.loc[(data['Load']>=40000) & (data['Load']<50000), 'Loss_Coeff'] = 2
data.loc[(data['Load']>=50000) & (data['Load']<60000), 'Loss_Coeff'] = 3

data.loc[(data['Load']>=60000) & (data['Load']<70000), 'Loss_Coeff'] = 4

data.loc[(data['Load']>=70000) & (data['Load']<80000), 'Loss_Coeff'] = 5

data.loc[data['Load']>80000, 'Loss_Coeff'] = 6

model = pyo.ConcreteModel()

model.t = pyo.RangeSet(1,365)

model.Load = pyo.Param(model.t, initialize=dict(zip(data.time, data.Load)))

Load = model.Load

model.Wind = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind)))

Wind = model.Wind

model.PV = pyo.Param(model.t, initialize=dict(zip(data.time, data.PV)))

PV = model.PV

model.CGp = pyo.Param(model.t, initialize=dict(zip(data.time, data.Grid_Price)))

CGp = model.CGp

model.CGs = pyo.Param(model.t, initialize=dict(zip(data.time, data.Grid_Sell)))

CGs = model.CGs

model.Coeff = pyo.Param(model.t, initialize=dict(zip(data.time, data.Loss_Coeff)))

Coeff = model.Coeff

model.eta_c = pyo.Param(initialize=0.95)

model.eta_d = pyo.Param(initialize=0.95)

model.maxcd = pyo.Param(initialize=0.35)

model.PE = pyo.Param(initialize=1)

model.cpv = pyo.Param(initialize=205)
model.cOMpv = pyo.Param(initialize=0.001)

model.cc = pyo.Param(initialize=75

model.Gp_up = pyo.Param(initialize=200000)

model.cwt = pyo.Param(initialize=38750)

model.cOM = pyo.Param(initialize=0.001)

model.CRF = pyo.Param(initialize=0.064)

model.CRFB = pyo.Param(initialize=1.7743)

#model.cR = pyo.Param(initialize=75)

model.ci = pyo.Param(initialize=75)

model.cb = pyo.Param(initialize=500)

model.Ob = pyo.Param(initialize=0.0004)

model.Mb = pyo.Param(initialize=9.8)

model.RPC = pyo.Param(initialize=500)

model.Teta = pyo.Param(initialize=1)

model.eps = pyo.Param(initialize=0)

model.socmin = pyo.Param(initialize=0.2)

Ob = model.Ob

Mb = model.Mb

RPC = model.RPC

#CGs = model.CGs

#CGp = model.CGp

#cR = model.cR

ci = model.ci
```

```
cb = model.cb

CRF = model.CRF

cc = model.cc

cOM = model.cOM

cwt = model.cwt

cpv = model.cpv

socmin = model.socmin

eta_c = model.eta_c

maxcd = model.maxcd

eta_d = model.eta_d

Gp_up = model.Gp_up

PE = model.PE

Teta = model.Teta

eps = model.eps

#Coeff = model.Coeff

cOMpv = model.cOMpv

CRFB = model.CRFB

model.Npv = pyo.Var(within=pyo.Integers, bounds=(0, 17623))

Npv = model.Npv

model.Nwt = pyo.Var(within=pyo.Integers, bounds=(0, 48))

Nwt = model.Nwt

model.Nb = pyo.Var(bounds=(0, None))

Nb = model.Nb
```

```
model.Eb = pyo.Var(model.t, bounds=(0, None))

Eb = model.Eb

model.Pc = pyo.Var(model.t, bounds=(0, None))

Pc = model.Pc

model.Pd = pyo.Var(model.t, bounds=(0, None))

Pd = model.Pd

model.Ewind = pyo.Var(model.t, bounds=(0, None))

Ewind = model.Ewind

model.Eswind = pyo.Var(model.t, bounds=(0, None))

Eswind = model.Eswind

model.Epv = pyo.Var(model.t, bounds=(0, None))

Epv = model.Epv

model.Espv = pyo.Var(model.t, bounds=(0, None))

Espv = model.Espv

model.Gp = pyo.Var(model.t, bounds=(0, 50000))

Gp = model.Gp

model.Gs = pyo.Var(model.t, bounds=(0, 200000))

Gs = model.Gs

model.gamma = pyo.Var(model.t, within=pyo.Binary)

gamma = model.gamma

model.teta = pyo.Var(model.t, within=pyo.Binary)

teta = model.teta
```

```
model.lam = pyo.Var(model.t, within=pyo.Binary)
lam = model.lam
model.eta = pyo.Var(model.t, within=pyo.Binary)
eta = model.eta
model.Loss = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, 5000))
Loss = model.Loss
def storage1(model, t):
    if t==1:
        return Eb[t] == Nb
    else:
        return Eb[t] == Eb[t - 1] + Pc[t] * eta_c - Pd[t] / eta_d
model.Const_1 = pyo.Constraint(model.t, rule=storage1)
def storage2(model, t):
    return Eb[t] >= Nb * socmin
model.Const_2 = pyo.Constraint(model.t, rule=storage2)
def storage3(model, t):
    return Eb[t] <= Nb
model.Const_3 = pyo.Constraint(model.t, rule=storage3)
def storage4(model, t):
    return Pc[t] * eta_c + Pd[t] / eta_d <= maxcd * Nb
model.Const_4 = pyo.Constraint(model.t, rule=storage4)
```

```
def storage5(model, t):

    return Pc[t] <= Nb * gamma[t]

model.Const_5 = pyo.Constraint(model.t, rule=storage5)

def storage6(model, t):

    return Pd[t] <= Nb * teta[t]

model.Const_6 = pyo.Constraint(model.t, rule=storage6)

def storage7(model, t):

    return gamma[t] + teta[t] == 1

model.Const_7 = pyo.Constraint(model.t, rule=storage7)

def grid1(model, t):

    return Gs[t] <= (Eswind[t] + Espv[t]) * PE * eta[t]

model.Const_8 = pyo.Constraint(model.t, rule=grid1)

def grid2(model, t):

    return Gp[t] <= Gp_up * lam[t]

model.Const_9 = pyo.Constraint(model.t, rule=grid2)

def grid3(model, t):

    return eta[t] + lam[t] == 1

model.Const_10 = pyo.Constraint(model.t, rule=grid3)

def Wind_surplus(model, t):

    return Nwt * Wind[t] == (Ewind[t] + Eswind[t])

model.Const_11 = pyo.Constraint(model.t, rule=Wind_surplus)
```

```
def PV_surplus(model, t):
   return Npv * PV[t] == Epv[t] + Espv[t]
model.Const_12 = pyo.Constraint(model.t, rule=PV_surplus)
def Balance(model, t):
   return Ewind[t] + Epv[t] + Pd[t] + Gp[t] + Loss[t] >= Load[t] + Pc[t] + Gs[t]
model.Const_13 = pyo.Constraint(model.t, rule=Balance)
def Loss_Constraint(model,t):
   return sum(Loss[t] for t in model.t) <= 0.001*sum(Load[t] for t in model.t)
model.Const_14 = pyo.Constraint(model.t, rule=Loss_Constraint)
def objective_rule(model):
   return (Npv * (cpv + cc)*(1 + cOMpv / CRF)) + (Nwt * cwt * (1 + cOM / CRF)) + Nb * (cb +
ci) + Ob * (
      sum((Pd[t] + Pc[t]) / CRF for t in model.t)) + (Mb * maxcd * Nb)*20.7743 + (RPC * Nb) *
CRFB + sum(
         (CGp[t] * Gp[t]) / CRF for t in model.t) - sum((CGs[t] * Gs[t]) / CRF for t in model.t)+sum(
            Loss[t]*Coeff[t] for t in model.t)+sum(Gp[t]*eps*Teta for t in model.t)
model.objective = pyo.Objective(rule=objective_rule, sense=pyo.minimize)
model.write('Opt_New.lp')
opt = SolverFactory('scipampl')
results = opt.solve(model, tee=True)
#opt.options['limits/time'] = 100
#opt = SolverFactory('gams')
```

```
#results = opt.solve(model, solver= 'dicopt', tee=True, keepfiles=True, add_options=['option reslim=20;'])

#results = opt.solve(model, solver= 'dicopt', tee=True,add_options=['option nlp=ipopt;'])

#model.display()

# **Evaluation of Results**

Data_Load = pd.read_excel('Load.xlsx')

print(results)

print('objec Fu = ' , pyo.value(model.objective)*1.32, 'CAD')

print('Cost of energy (COE) = ', (pyo.value(model.objective)*CRF/Data_Load['Load'].sum())*1.32, 'CAD/kWh')

print('Total Capital Cost = ' , (pyo.value(model.Npv)*(cpv + cc)+pyo.value(model.Nwt)*cwt+pyo.value(model.Nb)*(cb + ci))*1.32, 'CAD')

print('number of PV = ' , pyo.value(model.Npv))

print('PV Capacity = ' , (pyo.value(model.Npv))/4.5455)

print('number of Wt = ' , pyo.value(model.Nwt))

print('Battery Storage Capacity = ' , pyo.value(model.Nb), 'kWh')

Gs_values = pd.DataFrame(list(Gs[:].value),columns=['Gs'])

Gp_values = pd.DataFrame(list(Gp[:].value),columns=['Gp'])

Pd_values = pd.DataFrame(list(Pd[:].value),columns=['Pd'])

Pc_values = pd.DataFrame(list(Pc[:].value),columns=['Pc'])

Eb_values = pd.DataFrame(list(Eb[:].value),columns=['Eb'])

EWind_values = pd.DataFrame(list(Ewind[:].value),columns=['EWind'])

EPV_values = pd.DataFrame(list(Epv[:].value),columns=['EPV'])
```

```
ESWind_values = pd.DataFrame(list(Eswind[:].value),columns=['ESWind'])

ESPV_values = pd.DataFrame(list(Espv[:].value),columns=['ESPV'])

Unmet_Load = pd.DataFrame(list(Loss[:].value),columns=['Loss'])

Indexed_Result                                                                  =
pd.concat([Gs_values,Gp_values,Pd_values,Pc_values,Eb_values,EWind_values,EPV_values,

            ESWind_values,ESPV_values,Unmet_Load],axis=1)

Indexed_Result.to_csv('Results_Local_New.csv')

plt.plot(Unmet_Load)

print('Unmet Load = ',Unmet_Load.sum())

print('Number of days we have unmet load = ', 365-(Unmet_Load==0).sum())

Cycles_C =  365 - (Pc_values==0).sum()

Cycles_D =  365 - (Pd_values==0).sum()

Charging = Pc_values.sum()

Discharging = Pd_values.sum()

print('Charge Cycles = ',Cycles_C)

print('discharge Cycles = ',Cycles_D)

Unmet_sum = Unmet_Load.sum()

SOC = (Eb_values/12094)*100

# Battery Analysis

MaxCH = [0.02,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5]

COE = [1.056,0.55,0.3,0.22,0.18,0.15,0.14,0.12,0.12,0.11,0.1]

Cycles = [95,111,114,121,118,116,122,99,103,126]

Battery = [131547,84659,42329,28219,21164,16931,14109,12094,10582,9406,8465]
```

```
Ch_Dis =
[330780,356686,392332,404214,401678,404585,398483,390494,384498,366652,299703]
B_Operation = [2067,2229,2452,2526,2510,2528,2490,2440,2403,2291,1873]
x = MaxCH
y2=0
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(x,COE, color = 'blue',label = 'LCOE')
ax2.plot(x,B_Operation, color = 'darkorange',label = 'Battery Operation Cost')
ax1.set_xlabel('Max Charge-Discharge Rate',fontsize=13)
ax1.set_ylabel('LCOE $/kWh',fontsize=13)
ax2.set_ylabel('Battery Operation Cost $',fontsize=13)
plt.tick_params(axis='both', which='major', labelsize=12)
plt.tick_params(axis='both', which='major', labelsize=12)
ax1.legend(ncol=1,bbox_to_anchor=(0.5,1.12),loc='center',fontsize=12)
ax2.legend(ncol=1,bbox_to_anchor=(0.5,1.06),loc='center',fontsize=12)
plt.show()
x = range(365)
y2=0
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.fill_between(x, EPV_values.iloc[:,0].values,y2,facecolor='firebrick',label = 'PV Used
Power',alpha=0.7)
```

```
ax1.fill_between(x, Gp_values.iloc[:,0],y2,facecolor='darkorange',label = 'Purchase from Grid',alpha=0.7)
ax1.fill_between(x, EWind_values.iloc[:,0],y2,facecolor='indigo',label = 'Wind Used Power',alpha=0.7)
ax1.fill_between(x, Pc_values.iloc[:,0],y2,facecolor='limegreen',label = 'Battery Charge',alpha=0.7)
ax1.fill_between(x, Pd_values.iloc[:,0],y2,facecolor='slategrey',label = 'Battery Discharge',alpha=0.7)
ax1.fill_between(x, ESWind_values.iloc[:,0],y2,facecolor='pink',label = 'Wind Surplus',alpha=0.7)
ax1.fill_between(x, ESPV_values.iloc[:,0],y2,facecolor='green',label = 'PV Surplus',alpha=0.7)
ax1.fill_between(x, Unmet_Load.iloc[:,0],y2,facecolor='red',label = 'Unmet Load',alpha=0.7)
ax1.fill_between(x, Gs_values.iloc[:,0],y2,facecolor='yellow',label = 'Sell to Grid',alpha=0.7)
ax1.plot(x, data.iloc[:,2], 'black',label = 'Demand',linestyle = 'dashed')
ax2.plot(x, SOC.iloc[:,0], 'black',label = 'SOC')
ax2.set_ylim([0, 110])
ax1.set_xlabel('Days',fontsize=13)
ax1.set_ylabel('Power (kWh)',fontsize=13)
ax2.set_ylabel('State of Charge of the Battery %',fontsize=13)
ax2.tick_params(axis='both', which='major', labelsize=12)
ax1.tick_params(axis='both', which='major', labelsize=12)
ax2.legend(loc='upper right',fontsize=12)
ax1.legend(ncol=5,bbox_to_anchor=(0.5,1.08),loc='center',fontsize=12)
plt.show()
```

```
#Grid Trade-off

plt.plot(Gp_values, color = 'blue',label = 'Purchase from Grid')

plt.plot(Gs_values, color = 'darkorange',label = 'Sell to Grid')

plt.xlabel('Time(day)',fontsize=13)

plt.ylabel('Energy (kWh)',fontsize=13)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.legend(ncol=2,bbox_to_anchor=(0.5,1.06),loc='center',fontsize=12)

plt.show()

#Battery Ch-Disch

SOC = (Eb_values/12094)*100

Pc_Perc = (Pc_values/(12094))*100

Pd_Perc = (Pd_values/(12094))*100


plt.plot(Pc_Perc, color = 'firebrick',label = 'Battery Charging')

plt.plot(Pd_Perc, color = 'navy',label = 'Battery Discharging')

plt.plot(SOC, color = 'green',label = 'SOC')

plt.xlabel('Days',fontsize=13)

plt.ylabel('%',fontsize=13)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.legend(ncol=3,bbox_to_anchor=(0.5,1.08),loc='center',fontsize=12)

plt.show()
```

```
#Monthly Graph

bar_width = 0.15

Month_b = np.arange(12)

plt.bar(1,EPV_values.iloc[:30,0].sum(), bar_width, color = 'firebrick', label = 'PV')

plt.bar(2,EPV_values.iloc[31:60,0].sum(), bar_width, color = 'firebrick')

plt.bar(3,EPV_values.iloc[61:90,0].sum(), bar_width, color = 'firebrick')

plt.bar(4,EPV_values.iloc[91:121,0].sum(), bar_width, color = 'firebrick')

plt.bar(5,EPV_values.iloc[121:151,0].sum(), bar_width, color = 'firebrick')

plt.bar(6,EPV_values.iloc[151:181,0].sum(), bar_width, color = 'firebrick')

plt.bar(7,EPV_values.iloc[181:211,0].sum(), bar_width, color = 'firebrick')

plt.bar(8,EPV_values.iloc[211:241,0].sum(), bar_width, color = 'firebrick')

plt.bar(9,EPV_values.iloc[241:271,0].sum(), bar_width, color = 'firebrick')

plt.bar(10,EPV_values.iloc[271:301,0].sum(), bar_width, color = 'firebrick')

plt.bar(11,EPV_values.iloc[301:331,0].sum(), bar_width, color = 'firebrick')

plt.bar(12,EPV_values.iloc[332:365,0].sum(), bar_width, color = 'firebrick')

plt.bar(1+bar_width,EWind_values.iloc[:30,0].sum(), bar_width, color = 'navy', label = 'Wind')

plt.bar(2+bar_width,EWind_values.iloc[31:60,0].sum(), bar_width, color = 'navy')

plt.bar(3+bar_width,EWind_values.iloc[61:90,0].sum(), bar_width, color = 'navy')

plt.bar(4+bar_width,EWind_values.iloc[91:121,0].sum(), bar_width, color = 'navy')

plt.bar(5+bar_width,EWind_values.iloc[121:151,0].sum(), bar_width, color = 'navy')

plt.bar(6+bar_width,EWind_values.iloc[151:181,0].sum(), bar_width, color = 'navy')

plt.bar(7+bar_width,EWind_values.iloc[181:211,0].sum(), bar_width, color = 'navy')

plt.bar(8+bar_width,EWind_values.iloc[211:241,0].sum(), bar_width, color = 'navy')
plt.bar(9+bar_width,EWind_values.iloc[241:271,0].sum(), bar_width, color = 'navy',)

plt.bar(10+bar_width,EWind_values.iloc[271:301,0].sum(), bar_width, color = 'navy')

plt.bar(11+bar_width,EWind_values.iloc[301:331,0].sum(), bar_width, color = 'navy')

plt.bar(12+bar_width,EWind_values.iloc[332:365,0].sum(), bar_width, color = 'navy')

plt.bar(1-bar_width,Gp_values.iloc[:30,0].sum(), bar_width, color = 'gold', label = 'Purchase from
Grid')

plt.bar(2-bar_width,Gp_values.iloc[31:60,0].sum(), bar_width, color = 'gold')

plt.bar(3-bar_width,Gp_values.iloc[61:90,0].sum(), bar_width, color = 'gold')

plt.bar(4-bar_width,Gp_values.iloc[91:121,0].sum(), bar_width, color = 'gold')

plt.bar(5-bar_width,Gp_values.iloc[121:151,0].sum(), bar_width, color = 'gold')

plt.bar(6-bar_width,Gp_values.iloc[151:181,0].sum(), bar_width, color = 'gold')

plt.bar(7-bar_width,Gp_values.iloc[181:211,0].sum(), bar_width, color = 'gold')

plt.bar(8-bar_width,Gp_values.iloc[211:241,0].sum(), bar_width, color = 'gold')

plt.bar(9-bar_width,Gp_values.iloc[241:271,0].sum(), bar_width, color = 'gold',)

plt.bar(10-bar_width,Gp_values.iloc[271:301,0].sum(), bar_width, color = 'gold')

plt.bar(11-bar_width,Gp_values.iloc[301:331,0].sum(), bar_width, color = 'gold')

plt.bar(12-bar_width,Gp_values.iloc[332:365,0].sum(), bar_width, color = 'gold')

plt.bar(1+2*bar_width,Pd_values.iloc[:30,0].sum(), bar_width, color = 'slategrey', label = 'Battery
Discharge')

plt.bar(2+2*bar_width,Pd_values.iloc[31:60,0].sum(), bar_width, color = 'slategrey')

plt.bar(3+2*bar_width,Pd_values.iloc[61:90,0].sum(), bar_width, color = 'slategrey')

plt.bar(4+2*bar_width,Pd_values.iloc[91:121,0].sum(), bar_width, color = 'slategrey')

plt.bar(5+2*bar_width,Pd_values.iloc[121:151,0].sum(), bar_width, color = 'slategrey')
plt.bar(6+2*bar_width,Pd_values.iloc[151:181,0].sum(), bar_width, color = 'slategrey')
plt.bar(7+2*bar_width,Pd_values.iloc[181:211,0].sum(), bar_width, color = 'slategrey')
plt.bar(8+2*bar_width,Pd_values.iloc[211:241,0].sum(), bar_width, color = 'slategrey')
plt.bar(9+2*bar_width,Pd_values.iloc[241:271,0].sum(), bar_width, color = 'slategrey',)
plt.bar(10+2*bar_width,Pd_values.iloc[271:301,0].sum(), bar_width, color = 'slategrey')
plt.bar(11+2*bar_width,Pd_values.iloc[301:331,0].sum(), bar_width, color = 'slategrey')
plt.bar(12+2*bar_width,Pd_values.iloc[332:365,0].sum(), bar_width, color = 'slategrey')
#Surplus = (ESWind_values.iloc[:744,0]+ESPV_values.iloc[:744,0]).sum()
plt.bar(1-2*bar_width,(ESWind_values.iloc[:30,0]+ESPV_values.iloc[:30,0]).sum(),   bar_width,
color = 'peru', label = 'Surplus Power')
plt.bar(2-2*bar_width,(ESWind_values.iloc[31:60,0]+ESPV_values.iloc[31:60,0]).sum(),
bar_width, color = 'peru')
plt.bar(3-2*bar_width,(ESWind_values.iloc[61:90,0]+ESPV_values.iloc[61:90,0]).sum(),
bar_width, color = 'peru')
plt.bar(4-2*bar_width,(ESWind_values.iloc[91:121,0]+ESPV_values.iloc[91:121,0]).sum(),
bar_width, color = 'peru')
plt.bar(5-2*bar_width,(ESWind_values.iloc[121:151,0]+ESPV_values.iloc[121:151,0]).sum(),
bar_width, color = 'peru')
plt.bar(6-2*bar_width,(ESWind_values.iloc[151:181,0]+ESPV_values.iloc[151:181,0]).sum(),
bar_width, color = 'peru')
plt.bar(7-2*bar_width,(ESWind_values.iloc[181:211,0]+ESPV_values.iloc[181:211,0]).sum(),
bar_width, color = 'peru')
plt.bar(8-2*bar_width,(ESWind_values.iloc[211:241,0]+ESPV_values.iloc[211:241,0]).sum(),
bar_width, color = 'peru')
```

```
plt.bar(9-2*bar_width,(ESWind_values.iloc[241:271,0]+ESPV_values.iloc[241:271,0]).sum(),
bar_width, color = 'peru',)

plt.bar(10-2*bar_width,(ESWind_values.iloc[271:301,0]+ESPV_values.iloc[271:301,0]).sum(),
bar_width, color = 'peru')

plt.bar(11-2*bar_width,(ESWind_values.iloc[301:331,0]+ESPV_values.iloc[301:331,0]).sum(),
bar_width, color = 'peru')

plt.bar(12-2*bar_width,(ESWind_values.iloc[332:365,0]+ESPV_values.iloc[332:365,0]).sum(),
bar_width, color = 'peru')

labels = ('Jan', 'Feb', 'March', 'April', 'May', 'June', 'July', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec')

positions = (1,2,3,4,5,6,7,8,9,10,11,12)

LoadD                                                                                          =
np.array([data.iloc[:30,1].sum(),data.iloc[31:60,1].sum(),data.iloc[61:90,1].sum(),data.iloc[91:12
1,1].sum(),
data.iloc[121:151,1].sum(),data.iloc[151:181,1].sum(),data.iloc[181:211,1].sum(),data.iloc[211:2
41,1].sum(),


data.iloc[241:271,1].sum(),data.iloc[271:301,1].sum(),data.iloc[301:331,1].sum(),data.iloc[332:3
65,1].sum()])

plt.plot(positions,LoadD ,'black',label = 'Demand',linestyle = 'dashed',marker = 'o')

plt.xlabel('Month',fontsize=13)

plt.ylabel('Energy (kWh)',fontsize=13)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.xticks(positions,labels)

plt.legend(ncol=6,bbox_to_anchor=(0.5,1.05),loc='center',fontsize=12)
plt.show()

# Monthly Bar Chart

Date = pd.date_range('jan 01 2019', periods = 365, freq = 'D')

Indexed_Result.set_index(Date, inplace=True)

Indexed_Result_M = Indexed_Result.resample(rule = 'M').sum()

labels = ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'July', 'Aug', 'Sept', 'Oct', 'Nov', 'Dec')

labels = pd.DataFrame(labels)

Indexed_Result_M = Indexed_Result_M.reset_index(drop=True)

Indexed_Result_M = pd.concat([labels,Indexed_Result_M], axis=1)

Indexed_Result_M.set_index(0,inplace=True)

Indexed_Result_M = Indexed_Result_M.rename(columns={'Gs': 'Grid Sell', 'Gp': 'Grid Purchase',
'Pd': 'Battery Discharge', 'Pc': 'Battery Charge'

                                        , 'Eb': 'BState', 'EWind': 'Wind', 'EPV': 'PV','ESPV': 'PV
Surplus','ESWind': 'Wind Surplus','Loss': 'Unmet Load'})

Indexed_Result_M['Surplus'] = Indexed_Result_M['PV Surplus'] + Indexed_Result_M['Wind
Surplus']

Indexed_Result_M = Indexed_Result_M.iloc[:,[1,2,5,6,9,10]]

Indexed_Result_M.plot(kind='bar',stacked=True,hatch='\\\\\\\\',width=0.8,rot=0,alpha=0.75)

plt.xlabel('Months')

plt.ylabel('Power kWh')

plt.legend(ncol=6,bbox_to_anchor=(0.5,1.05),loc='center',fontsize=10)
```

Mix of Regional and Local Optimum Design

```
import pyomo.environ as pyo
```

```
from pyomo.environ import *
from pyomo.opt import SolverFactory
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import os
data = pd.read_excel("Input_Daily_Sum.xlsx")
data2 = pd.read_excel("Map_Params.xlsx")
# Propose Rate
Grid_P = []
for i in range(len(data['Load_HP'])):
  if (data['time'][i]<=90) & (data['Load_HP'][i]+data['Load_E'][i]<100000):
     Grid_P.append(0.08)
  if (90<data['time'][i]<335) & (data['Load_HP'][i]+data['Load_E'][i]<100000):
     Grid_P.append(0.06)
  if (data['time'][i]>=335) & (data['Load_HP'][i]+data['Load_E'][i]<100000):
     Grid_P.append(0.08)
  if (data['time'][i]<=90) & (100000<=data['Load_HP'][i]+data['Load_E'][i]<120000):
     Grid_P.append(0.12)
  if (90<data['time'][i]<335) & (100000<=data['Load_HP'][i]+data['Load_E'][i]<120000):
     Grid_P.append(0.08)
  if (data['time'][i]>=335) & (100000<=data['Load_HP'][i]+data['Load_E'][i]<120000):
    Grid_P.append(0.12)
if (data['time'][i]<=90) & (120000<=data['Load_HP'][i]+data['Load_E'][i]<140000):
    Grid_P.append(0.18)
if (90<data['time'][i]<335) & (120000<=data['Load_HP'][i]+data['Load_E'][i]<140000):
    Grid_P.append(0.1)
if (data['time'][i]>=335) & (120000<=data['Load_HP'][i]+data['Load_E'][i]<140000):
    Grid_P.append(0.18)
if (data['time'][i]<=90) & (140000<=data['Load_HP'][i]+data['Load_E'][i]<160000):
    Grid_P.append(0.26)
if (90<data['time'][i]<335) & (140000<=data['Load_HP'][i]+data['Load_E'][i]<160000):
    Grid_P.append(0.12)
if (data['time'][i]>=335) & (140000<=data['Load_HP'][i]+data['Load_E'][i]<160000):
    Grid_P.append(0.26)
if (data['time'][i]<=90) & (160000<=data['Load_HP'][i]+data['Load_E'][i]<180000):
    Grid_P.append(0.36)
if (90<data['time'][i]<335) & (160000<=data['Load_HP'][i]+data['Load_E'][i]<180000):
    Grid_P.append(0.14)
if (data['time'][i]>=335) & (160000<=data['Load_HP'][i]+data['Load_E'][i]<180000):
    Grid_P.append(0.36)
if (data['time'][i]<=90) & (data['Load_HP'][i]+data['Load_E'][i]>180000):
    Grid_P.append(0.48)
if (90<data['time'][i]<335) & (data['Load_HP'][i]+data['Load_E'][i]>180000):
    Grid_P.append(0.16)
```

```
  if (data['time'][i]>=335) & (data['Load_HP'][i]+data['Load_E'][i]>180000):
      Grid_P.append(0.48)
Grid_P = pd.DataFrame(Grid_P)
Grid_P.columns = ['Grid_Price']
data = pd.concat([data,Grid_P],axis=1)
Grid_S = []
for i in range(len(data['Load_HP'])):
  if data['time'][i]<=90:
      Grid_S.append(0.2)
  if 90<data['time'][i]<335:
      Grid_S.append(0.1)
  if data['time'][i]>=335:
      Grid_S.append(0.2)


Grid_S = pd.DataFrame(Grid_S)
Grid_S.columns = ['Grid_Sell']
data = pd.concat([data,Grid_S],axis=1)
data.loc[data['Load_HP']+data['Load_E'][i]<100000, 'Loss_Coeff'] = 1
data.loc[(data['Load_HP']+data['Load_E'][i]>=100000)                          &
(data['Load_HP']+data['Load_E'][i]<120000), 'Loss_Coeff'] = 2
data.loc[(data['Load_HP']+data['Load_E'][i]>=120000)                          &
(data['Load_HP']+data['Load_E'][i]<140000), 'Loss_Coeff'] = 3
```

```
data.loc[(data['Load_HP']+data['Load_E'][i]>=140000)                                        &
(data['Load_HP']+data['Load_E'][i]<160000), 'Loss_Coeff'] = 4

data.loc[(data['Load_HP']+data['Load_E'][i]>=160000)                                        &
(data['Load_HP']+data['Load_E'][i]<180000), 'Loss_Coeff'] = 5

data.loc[data['Load_HP']+data['Load_E'][i]>180000, 'Loss_Coeff'] = 6

model = pyo.ConcreteModel()

# **Sets and Parameters**

model.t = pyo.RangeSet(1,365)

model.l = pyo.RangeSet(1,76)

model.Load_E = pyo.Param(model.t, initialize=dict(zip(data.time, data.Load_E)))

Load_E = model.Load_E

model.Load_HP = pyo.Param(model.t, initialize=dict(zip(data.time, data.Load_HP)))

Load_HP = model.Load_HP

model.Gas = pyo.Param(model.t, initialize=dict(zip(data.time, data.Gas_Cons)))

Gas = model.Gas


# Wind1 is for local wind generation

model.Wind1 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind1)))

Wind1 = model.Wind1

# Wind2 is for 'OPEN AREA'

model.Wind2 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind2)))

Wind2 = model.Wind2

# Wind3 is for 'RESOURCE AND INDUSTRIAL'
```

```
model.Wind3 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind3)))

Wind3 = model.Wind3

# Wind4 is for 'COMMERCIAL'

model.Wind4 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind4)))

Wind4 = model.Wind4

# Wind5 is for 'PARKS AND RECREATIONAL'

model.Wind5 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind5)))

Wind5 = model.Wind5

# Wind6 is for 'GOVERNMENT AND INSTITUTIONAL'

model.Wind6 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind6)))

Wind6 = model.Wind6

# Wind7 is for 'RESIDENTIAL'

model.Wind7 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind7)))

Wind7 = model.Wind7

# Wind8 is for 'Waterbody'

model.Wind8 = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind8)))

Wind8 = model.Wind8

model.PV = pyo.Param(model.t, initialize=dict(zip(data.time, data.PV)))

PV = model.PV

model.Cl = pyo.Param(model.l, initialize=dict(zip(data2.LandType, data2.Land_Cost)))

Cl = model.Cl      # Land Cost ($/m2)

model.TD = pyo.Param(model.l, initialize=dict(zip(data2.LandType, data2.Turbine_D)))

TD = model.TD
```

```
model.dis = pyo.Param(model.l, initialize=dict(zip(data2.LandType, data2.Distance)))

dis = model.dis

model.LT = pyo.Param(model.l, initialize=dict(zip(data2.LandType, data2.Zone_type)))

LT = model.LT

model.CGp = pyo.Param(model.t, initialize=dict(zip(data.time, data.Grid_Price)))

CGp = model.CGp

model.CGs = pyo.Param(model.t, initialize=dict(zip(data.time, data.Grid_Sell)))

CGs = model.CGs

model.Coeff = pyo.Param(model.t, initialize=dict(zip(data.time, data.Loss_Coeff)))

Coeff = model.Coeff

model.eta_c = pyo.Param(initialize=0.95)

model.eta_d = pyo.Param(initialize=0.95)

model.maxcd = pyo.Param(initialize=0.35)

model.PE = pyo.Param(initialize=1)

model.cpv = pyo.Param(initialize=205)

model.cc = pyo.Param(initialize=75)

model.Gp_up = pyo.Param(initialize=2000000000)

model.cwt = pyo.Param(initialize=38750)

model.cOMwt = pyo.Param(initialize=0.001)

model.cOMpv = pyo.Param(initialize=0.001)

model.CRF = pyo.Param(initialize=0.064)

model.CRFG = pyo.Param(initialize=20.6)

model.CRFB = pyo.Param(initialize=1.7743)
```

```
#model.cR = pyo.Param(initialize=75)

model.ci = pyo.Param(initialize=75)

model.cb = pyo.Param(initialize=500)

model.Ob = pyo.Param(initialize=0.0004)

model.Mb = pyo.Param(initialize=9.8)

model.RPC = pyo.Param(initialize=500)

model.socmin = pyo.Param(initialize=0.2)

model.Bd = pyo.Param(initialize=20000000)

model.Ct = pyo.Param(initialize=2000)

model.A = pyo.Param(initialize=10)

model.crc = pyo.Param(initialize=1240000)

model.Ci = pyo.Param(initialize=20000)

model.omT = pyo.Param(initialize=0.0005

model.eps = pyo.Param(initialize=0)

model.Teta = pyo.Param(initialize=1)

model.hpc = pyo.Param(initialize=853)

model.cOMhp = pyo.Param(initialize=0.00003)

model.maxhp = pyo.Param(initialize=6224)

model.Cg = pyo.Param(initialize=0.1517)

Ob = model.Ob

Mb = model.Mb

RPC = model.RPC

CGp = model.CGp
```

```
CGs = model.CGs

#cR = model.cR

ci = model.ci

cb = model.cb

CRF = model.CRF

cc = model.cc

cOMwt = model.cOMwt

cwt = model.cwt

cpv = model.cpv

socmin = model.socmin

eta_c = model.eta_c

maxcd = model.maxcd

eta_d = model.eta_d

Gp_up = model.Gp_up

PE = model.PE

Bd = model.Bd

Ct = model.Ct

A = model.A

crc = model.crc

Ci = model.Ci

omT = model.omT

Teta = model.Teta

eps = model.eps
```

```
cOMpv = model.cOMpv

hpc = model.hpc

maxhp = model.maxhp

cOMhp = model.cOMhp

Cg = model.Cg

CRFB = model.CRFB

CRFG = model.CRFG

# area = 24487      #This area will be defined by user or automatically by software

#

# def PV_Count(model):

#     return (0,area/1.32

model.Npv = pyo.Var(within=pyo.Integers, bounds=(0, 17623))

Npv = model.Npv

model.Nwt = pyo.Var(within=pyo.Integers, bounds=(0, 48))

Nwt = model.Nwt


model.Nb = pyo.Var(bounds=(6000, None))

Nb = model.Nb

model.Eb = pyo.Var(model.t, bounds=(0, None))

Eb = model.Eb

model.Pc = pyo.Var(model.t, bounds=(0, None))

Pc = model.Pc

model.Pd = pyo.Var(model.t, bounds=(0, None))
```

```
Pd = model.Pd

model.Ewind = pyo.Var(model.t, bounds=(0, None))

Ewind = model.Ewind

model.Eswind = pyo.Var(model.t, bounds=(0, None))

Eswind = model.Eswind

model.Epv = pyo.Var(model.t, bounds=(0, None))

Epv = model.Epv

model.Espv = pyo.Var(model.t, bounds=(0, None))

Espv = model.Espv

model.Gp = pyo.Var(model.t, bounds=(0, 200000))

Gp = model.Gp

model.Gs = pyo.Var(model.t, bounds=(0, 200000))

Gs = model.Gs

model.gamma = pyo.Var(model.t, within=pyo.Binary)

gamma = model.gamma


model.teta = pyo.Var(model.t, within=pyo.Binary)

teta = model.teta

model.lam = pyo.Var(model.t, within=pyo.Binary)

lam = model.lam

model.eta = pyo.Var(model.t, within=pyo.Binary)

eta = model.eta

model.La = pyo.Var(bounds=(0,None))
```

```
La = model.La

model.Nrg = pyo.Var(within=pyo.Integers,bounds=(0,None))

Nrg = model.Nrg

model.Esrg1 = pyo.Var(model.t,bounds=(0,None))

Esrg1 = model.Esrg1

model.Esrg2 = pyo.Var(model.t,bounds=(0,None))

Esrg2 = model.Esrg2

model.Esrg3 = pyo.Var(model.t,bounds=(0,None))

Esrg3 = model.Esrg3

model.Esrg4 = pyo.Var(model.t,bounds=(0,None))

Esrg4 = model.Esrg4

model.Esrg5 = pyo.Var(model.t,bounds=(0,None))

Esrg5 = model.Esrg5

model.Esrg6 = pyo.Var(model.t,bounds=(0,None))

Esrg6 = model.Esrg6


model.Esrg7 = pyo.Var(model.t,bounds=(0,None))

Esrg7 = model.Esrg7

model.Erg1 = pyo.Var(model.t,bounds=(0,None))

Erg1 = model.Erg1

model.Erg2 = pyo.Var(model.t,bounds=(0,None))

Erg2 = model.Erg2

model.Erg3 = pyo.Var(model.t,bounds=(0,None))
```

```
Erg3 = model.Erg3

model.Erg4 = pyo.Var(model.t,bounds=(0,None))

Erg4 = model.Erg4

model.Erg5 = pyo.Var(model.t,bounds=(0,None))

Erg5 = model.Erg5

model.Erg6 = pyo.Var(model.t,bounds=(0,None))

Erg6 = model.Erg6

model.Erg7 = pyo.Var(model.t,bounds=(0,None))

Erg7 = model.Erg7

model.Bin = pyo.Var(model.l,within=pyo.Binary)

Bin = model.Bin

model.Loss = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

Loss = model.Loss

model.Chp = pyo.Var(within=pyo.NonNegativeReals, bounds=(0, None))

Chp = model.Chp


def storage1(model, t):

   if t==1:

       return Eb[t] == Nb

   else:

       return Eb[t] == Eb[t - 1] + Pc[t] * eta_c - Pd[t] / eta_d

model.Const_1 = pyo.Constraint(model.t, rule=storage1)

def storage2(model, t):
    return Eb[t] >= Nb * socmin
model.Const_2 = pyo.Constraint(model.t, rule=storage2)
def storage3(model, t):
    return Eb[t] <= Nb
model.Const_3 = pyo.Constraint(model.t, rule=storage3)
def storage4(model, t):
    return Pc[t] * eta_c + Pd[t] / eta_d <= maxcd * Nb
model.Const_4 = pyo.Constraint(model.t, rule=storage4)
def storage5(model, t):
    return Pc[t] <= Nb * gamma[t]
model.Const_5 = pyo.Constraint(model.t, rule=storage5)
def storage6(model, t):
    return Pd[t] <= Nb * teta[t]
model.Const_6 = pyo.Constraint(model.t, rule=storage6)



def storage7(model, t):
    return gamma[t] + teta[t] == 1
model.Const_7 = pyo.Constraint(model.t, rule=storage7)
def gridS1(model, t,l):
    if LT[l] == 'OPEN AREA':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg1[t]) * PE * eta[t]
    if LT[l] == 'RESOURCE AND INDUSTRIAL':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg2[t]) * PE * eta[t]
    if LT[l] == 'COMMERCIAL':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg3[t]) * PE * eta[t]
    if LT[l] == 'PARKS AND RECREATIONAL':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg4[t]) * PE * eta[t]
    if LT[l] == 'GOVERNMENT AND INSTITUTIONAL':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg5[t]) * PE * eta[t]
    if LT[l] == 'RESIDENTIAL':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg6[t]) * PE * eta[t]
    if LT[l] == 'WATERBODY':
        return Gs[t] <= (Eswind[t] + Espv[t] + Esrg7[t]) * PE * eta[t]
model.Const_8 = pyo.Constraint(model.t,model.l, rule=gridS1)
def grid2(model, t):
    return Gp[t] <= Gp_up * lam[t]



model.Const_9 = pyo.Constraint(model.t, rule=grid2)
def grid3(model, t):
    return eta[t] + lam[t] == 1
model.Const_10 = pyo.Constraint(model.t, rule=grid3)
def Wind_surplus(model, t):
    return Nwt * Wind1[t] == (Ewind[t] + Eswind[t])
model.Const_11 = pyo.Constraint(model.t, rule=Wind_surplus)
```

```
def PV_surplus(model, t):
    return Npv * PV[t] == Epv[t] + Espv[t]
model.Const_12 = pyo.Constraint(model.t, rule=PV_surplus)
model.Const_13 = pyo.Constraint(expr = La == Nrg*A)
def LandCost(model, l):
    return dis[l]*Ct + Cl[l]*La <= Bd
model.Const_14 = pyo.Constraint(model.l, rule=LandCost)
def Wind_regional2(model, t,l):
    if LT[l] == 'OPEN AREA':
        return Nrg * Wind2[t] == (Erg1[t] + Esrg1[t])
    if LT[l] == 'RESOURCE AND INDUSTRIAL':
        return Nrg * Wind3[t] == (Erg2[t] + Esrg2[t])
    if LT[l] == 'COMMERCIAL':
        return Nrg * Wind4[t] == (Erg3[t] + Esrg3[t])
    if LT[l] == 'PARKS AND RECREATIONAL':
        return Nrg * Wind5[t] == (Erg4[t] + Esrg4[t])
    if LT[l] == 'GOVERNMENT AND INSTITUTIONAL':
        return Nrg * Wind6[t] == (Erg5[t] + Esrg5[t])
    if LT[l] == 'RESIDENTIAL':
        return Nrg * Wind7[t] == (Erg6[t] + Esrg6[t])
    if LT[l] == 'WATERBODY':
        return Nrg * Wind8[t] == (Erg7[t] + Esrg7[t])
model.Const_15 = pyo.Constraint(model.t,model.l, rule=Wind_regional2)
```

```
def Balance(model, t,l):
    if LT[l] == 'OPEN AREA':
        return Ewind[t] + Epv[t] + Erg1[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'RESOURCE AND INDUSTRIAL':
        return Ewind[t] + Epv[t] + Erg2[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'COMMERCIAL':
        return Ewind[t] + Epv[t] + Erg3[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'PARKS AND RECREATIONAL':
        return Ewind[t] + Epv[t] + Erg4[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'GOVERNMENT AND INSTITUTIONAL':
        return Ewind[t] + Epv[t] + Erg5[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'RESIDENTIAL':
        return Ewind[t] + Epv[t] + Erg6[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
    if LT[l] == 'WATERBODY':
        return Ewind[t] + Epv[t] + Erg7[t] + Pd[t] + Gp[t] + Loss[t] >= Load_E[t] + Load_HP[t] +
Pc[t] + Gs[t]
model.Const_16 = pyo.Constraint(model.t,model.l, rule=Balance)
def Land_Selection(model,l):
    return sum(Bin[l] for l in model.l) == 1
```

```
model.Const_17 = pyo.Constraint(model.l,rule=Land_Selection)

def Turbine_Density(model,l):

   return Nrg <= sum(TD[l]*Bin[l] for l in model.l)

model.Const_18 = pyo.Constraint(model.l,rule=Turbine_Density)

def Loss_Constraint(model,t):

   return sum(Loss[t] for t in model.t) <= 0.001*sum(Load_E[t] + Load_HP[t]   for t in model.t)

model.Const_19 = pyo.Constraint(model.t, rule=Loss_Constraint)

def Heat_Pump(model,t):

   return Chp >= maxhp

model.Const_20 = pyo.Constraint(model.t,rule=Heat_Pump)

# **Objective Function

def objective_rule(model):

   return (Npv * (cpv + cc)*(1 + cOMpv / CRF)) + (Nwt * cwt * (1 + cOMwt / CRF)) + Nb * (cb + ci) + Ob * (

      sum((Pd[t] + Pc[t]) / CRF for t in model.t)) + (Mb * maxcd * Nb)*20.7743 + (RPC * Nb) * CRFB + sum(

         (CGp[t] * Gp[t]) / CRF for t in model.t) - sum((CGs[t] * Gs[t]) / CRF for t in model.t)+(

            Nrg * crc * (1 + cOMwt / CRF)) + sum(((dis[l]*Ct) + Ci)*(1 + omT / CRF)*Bin[l] for l in model.l) + sum(

            Cl[l]*La*Bin[l] for l in model.l)+sum(Loss[t]*Coeff[t] for t in model.t)+sum(Gp[t]*eps*Teta for t in model.t)+(

              Chp*hpc*(1 + cOMhp / CRF)) - sum(Gas[t]*Cg for t in model.t)/CRF - sum(Gas[t]*0.0391*0.8*0.74 for t in model.t)*CRFG

model.objective = pyo.Objective(rule=objective_rule, sense=pyo.minimize)
model.write('Opt_New.lp', io_options={'symbolic_solver_labels': True})

opt = SolverFactory('scipampl')

#opt = SolverFactory('gams')

#results = opt.solve(model, solver= 'dicopt', tee=True, add_options=['option optcr=0.2;','option reslim=3600;'])

#results = opt.solve(model, solver= 'dicopt', tee=True,add_options=['option nlp=ipopt;'])

#results = opt.solve(model, solver= 'scip', tee=True, keepfiles=True, add_options=['option optcr=3.56;','option reslim=200;'])

#results = opt.solve(model, solver= 'scip', tee=True, keepfiles=True, add_options=['option reslim=100;'])

#results = opt.solve(model, solver= 'couenne', tee=True, keepfiles=True)

#results = opt.solve(model, solver= 'scip', tee=True)

results = opt.solve(model,tee=True)

Data_Load = pd.read_excel('Load.xlsx')

print(results)

print('objec Fu = ' , pyo.value(model.objective)*1.32, 'CAD')

print('Cost of energy (COE) = ', pyo.value(model.objective)*CRF/Data_Load['Load'].sum()*1.32, 'CAD/kWh')

print('Total Capital Cost = ' , pyo.value(model.Npv)*(cpv + cc)+pyo.value(model.Nwt)*cwt+pyo.value(model.Nb)*(cb + ci)+
    pyo.value(model.Nrg)*crc + pyo.value(model.Chp)*hpc*1.32, 'CAD')

print('number of pv = ' , pyo.value(model.Npv))

print('number of Wt = ' , pyo.value(model.Nwt))

print('Battery Storage Capacity = ' , pyo.value(model.Nb), 'kWh')
print('number of Nrg = ' , pyo.value(model.Nrg))

print('Heat Pump Capacity = ' , pyo.value(model.Chp), 'kWh')

for l in model.l:

    if Bin[l].value!=0:

        print('Zone',l, 'is the selected for regional generation')

Gs_values = pd.DataFrame(list(Gs[:].value),columns=['Gs'])

Gp_values = pd.DataFrame(list(Gp[:].value),columns=['Gp'])

Pd_values = pd.DataFrame(list(Pd[:].value),columns=['Pd'])

Pc_values = pd.DataFrame(list(Pc[:].value),columns=['Pc'])

Eb_values = pd.DataFrame(list(Eb[:].value),columns=['Eb'])

EWind_values = pd.DataFrame(list(Ewind[:].value),columns=['EWind'])

EPV_values = pd.DataFrame(list(Epv[:].value),columns=['EPV'])

ESWind_values = pd.DataFrame(list(Eswind[:].value),columns=['ESWind'])

ESPV_values = pd.DataFrame(list(Espv[:].value),columns=['ESPV'])

Erg_values = pd.DataFrame(list(Erg1[:].value),columns=['Erg'])

Esrg_values = pd.DataFrame(list(Esrg1[:].value),columns=['Esrg'])

Unmet_Load = pd.DataFrame(list(Loss[:].value),columns=['Loss'])

Indexed_Result                                                                                     =
pd.concat([Gs_values,Gp_values,Pd_values,Pc_values,Eb_values,EWind_values,EPV_values,

                      ESWind_values,ESPV_values,Erg_values,Esrg_values,Unmet_Load],axis=1)

plt.plot(Unmet_Load)
```

```
print('Renewable Penetration = ' ,
((np.array(EPV_values.sum()))+(np.array(Erg_values.sum()))+(np.array(EWind_values.sum())))
/(
(np.array(EPV_values.sum()))+(np.array(Erg_values.sum()))+(np.array(EWind_values.sum()))+
(np.array(Gp_values.sum()))))
print('Unmet Load = ',Unmet_Load.sum())
print('Number of days we have unmet load = ', 365 - (Unmet_Load==0).sum())
Indexed_Result.to_csv('Results_Local_New.csv')
SOC = (Eb_values/6000)*100
Date = pd.date_range('jan 01 2019', periods = 365, freq = 'D')
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(Date, data['Load_HP']+data['Load_E'], 'navy',label = 'Load Demand')
ax2.plot(Date, Grid_P, 'maroon',label = 'Purchase Price')
ax2.plot(Date, Grid_S, 'green',label = 'Sell Price')
ax2.set_ylim([0, 0.8])
ax1.set_xlabel('Date',fontsize=10)
ax1.set_ylabel('Power (kW)',fontsize=10)
ax2.set_ylabel('Price ($/kWh)',fontsize=10)
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper center',fontsize=10)
plt.show()
```

```
x = range(365)

y2=0

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()

ax1.fill_between(x, Erg_values.iloc[:,0],y2,facecolor='blue',label = 'Regional Wind',alpha=0.7)

ax1.fill_between(x, EPV_values.iloc[:,0].values,y2,facecolor='firebrick',label = 'PV Used
Power',alpha=0.7)

ax1.fill_between(x, EWind_values.iloc[:,0],y2,facecolor='indigo',label = 'Wind Used
Power',alpha=0.7)

ax1.fill_between(x, Pc_values.iloc[:,0],y2,facecolor='limegreen',label = 'Battery
Charge',alpha=0.7)

ax1.fill_between(x, Pd_values.iloc[:,0],y2,facecolor='slategrey',label = 'Battery
Discharge',alpha=0.7)

ax1.fill_between(x, ESWind_values.iloc[:,0],y2,facecolor='pink',label = 'Wind
Surplus',alpha=0.7)

ax1.fill_between(x, ESPV_values.iloc[:,0],y2,facecolor='green',label = 'PV Surplus',alpha=0.7)

ax1.fill_between(x, Esrg_values.iloc[:,0],y2,facecolor='olive',label = 'Regional Wind
Surplus',alpha=0.7)

ax1.fill_between(x, Gp_values.iloc[:,0],y2,facecolor='darkorange',label = 'Purchase from
Grid',alpha=0.7)

ax1.fill_between(x, Gs_values.iloc[:,0],y2,facecolor='yellow',label = 'Sell to Grid',alpha=0.7)

ax1.fill_between(x, Unmet_Load.iloc[:,0],y2,facecolor='red',label = 'Unmet Load',alpha=0.7)

ax1.plot(x, data.iloc[:,2], 'black',label = 'Demand',linestyle = 'dashed')

ax2.plot(x, SOC.iloc[:,0], 'black',label = 'SOC')

ax2.set_ylim([0, 110])
```

```
ax1.set_xlabel('Hours',fontsize=13)

ax1.set_ylabel('Power Consumption (kWh)',fontsize=13)

ax2.set_ylabel('State of Charge of the Battery %',fontsize=13)

ax2.tick_params(axis='both', which='major', labelsize=12)

ax1.tick_params(axis='both', which='major', labelsize=12)

ax2.legend(loc='upper right',fontsize=12)

ax1.legend(ncol=6,bbox_to_anchor=(0.5,1.08),loc='center',fontsize=12)

plt.show()

# Grid Trade-off

plt.plot(Gp_values, color = 'blue',label = 'Purchase from Grid')

plt.plot(Gs_values, color = 'darkorange',label = 'Sell to Grid')

plt.xlabel('Time(hr)',fontsize=13)

plt.ylabel('Energy (kWh)',fontsize=13)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.tick_params(axis='both', which='major', labelsize=12)

plt.legend(ncol=2,bbox_to_anchor=(0.5,1.05),loc='center',fontsize=12)

plt.show()

# Battery Charge and Discharge Power

plt.plot(Pc_values, color = 'red',label = 'Battery Charging')

plt.plot(Pd_values, color = 'blue',label = 'Battery Discharging')

plt.plot(Eb_values, color = 'green',label = 'SOC')

plt.xlabel('Time(hr)')

plt.ylabel('Energy (kWh)')
plt.legend()

plt.show()
```

Resilience Oriented Optimal Dispatch Operation

```
import pyomo.environ as pyo

from pyomo.opt import SolverFactory

import pandas as pd

import math

import numpy as np

import matplotlib.pyplot as plt

!pip install cplex -q

data = pd.read_excel("Input_Dispatch_OffGrid.xlsx", sheet_name = 'Summer')

model = pyo.ConcreteModel()

# **Sets and Parameters**

model.t = pyo.RangeSet(48)

model.Load = pyo.Param(model.t, initialize=dict(zip(data.time, data.Load)))

Load = model.Load

model.Wind = pyo.Param(model.t, initialize=dict(zip(data.time, data.Wind)))

Wind = model.Wind

model.PV = pyo.Param(model.t, initialize=dict(zip(data.time, data.PV)))

PV = model.PV

lb={1:0.25022,
2:0.24969,3:0.25000,4:0.25015,5:0.24837,6:0.24447,7:0.24270,8:0.24130,9:0.59211,10:0.59218
,11:0.59205,12:0.59210,13:0.59205,14:0.59211,15:0.59185,16:0.59203,17:0.59197,18:0.59208,1
9:0.59212,20:0.59188,21:0.59118,22:0.24805,23:0.25060,24:0.25185,25:0.25164,
26:0.25149,27:0.25206,28:0.25241,29:0.25252,30:0.25101,31:0.25302,32:0.25225,33:0.59047,3
4:0.59011,35:0.59059,36:0.59070,37:0.59061,
38:0.59053,39:0.59073,40:0.59061,41:0.59060,42:0.59069,43:0.59110,44:0.59121,45:0.59109,4
6:0.24575,47:0.24677,48:0.24519}
ub={1:1,2:1,3:1,4:1,5:1,6:1,7:1,8:1,9:1,10:1,11:1,12:1,13:1,14:1,15:1,16:1,17:1,18:1,19:1,20:1,2
1:1,22:1,23:1,24:1,25:1,26:1,27:1,28:1,
29:1,30:1,31:1,32:1,33:1,34:1,35:1,36:1,37:1,38:1,39:1,40:1,41:1,42:1,43:1,44:1,45:1,46:1,47:1,
48:1}


def fb(model, i):
  return (lb[i], ub[i])
model.eta_c = pyo.Param(initialize=0.95)
model.eta_d = pyo.Param(initialize=0.90)
model.RPC = pyo.Param(initialize=156)
model.Deg = pyo.Param(initialize=0.074)
model.SOC0 = pyo.Param(initialize=8000)
model.CCF = pyo.Param(initialize=0.1)
model.Ebamx = pyo.Param(initialize=9375)
model.maxcd = pyo.Param(initialize=0.3)
model.Coeff = pyo.Param(initialize=0.17)
model.SOCmin = pyo.Param(initialize= 937)
RPC = model.RPC
eta_c = model.eta_c
eta_d = model.eta_d
```

```
Deg = model.Deg

SOC0 = model.SOC0

CCF = model.CCF

Ebmax = model.Ebamx

maxcd = model.maxcd

Coeff = model.Coeff

SOCmin =model.SOCmin

model.Eb = pyo.Var(model.t, domain=pyo.NonNegativeReals, bounds=(0, 9375))

Eb = model.Eb

model.Pc = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

Pc = model.Pc

model.Pd = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

Pd = model.Pd

model.EWind = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

EWind = model.EWind

model.ESWind = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

ESWind = model.ESWind

model.EPV = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

EPV = model.EPV

model.ESPV = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

ESPV = model.ESPV

model.Loss = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=(0, None))

Loss = model.Loss
```

```
model.teta = pyo.Var(model.t, within=pyo.Binary)

teta = model.teta

model.eta = pyo.Var(model.t, within=pyo.Binary)

eta = model.eta

model.C = pyo.Var(model.t,domain=pyo.NonNegativeReals, bounds=fb)

C = model.C

def storage1(model, t):

    if t==1:

        return Eb[t] == SOC0

    else:

        return Eb[t] == Eb[t - 1] + Pc[t] * eta_c - Pd[t] / eta_d

model.Const_1 = pyo.Constraint(model.t, rule=storage1)

def storage2(model, t):

    return Pc[t] * eta_c + Pd[t] / eta_d <= maxcd * Ebmax

model.Const_2 = pyo.Constraint(model.t, rule=storage2)

def storage3(model, t):

    return Pc[t] <= Ebmax * eta[t]

model.Const_3 = pyo.Constraint(model.t, rule=storage3)

def storage4(model, t):

    return Pd[t] <= Ebmax * teta[t]

model.Const_4 = pyo.Constraint(model.t, rule=storage4)

def storage5(model, t):

    return eta[t] + teta[t] <= 1
```

```
model.Const_5 = pyo.Constraint(model.t, rule=storage5)

def storage6(model, t):

   return Eb[t] >= SOCmin

model.Const_6 = pyo.Constraint(model.t, rule=storage6)

def Surplus1(model, t):

   return Wind[t] == EWind[t]+ESWind[t]

model.Const_7 = pyo.Constraint(model.t, rule=Surplus1)

def Surplus2(model, t):

   return PV[t] == EPV[t]+ESPV[t]

model.Const_8 = pyo.Constraint(model.t, rule=Surplus2)

def Balance(model, t):

   return EWind[t] + EPV[t] + Pd[t] + Loss[t] == C[t]*Load[t] + Pc[t]

model.Const_9 = pyo.Constraint(model.t, rule=Balance)

# **Objective Function

def objective_rule(model):

   return sum(Deg*Pc[t] for t in model.t) + sum(Deg*Pd[t] for t in model.t) + sum(CCF*ESPV[t]
for t in model.t) + sum(

      CCF*ESWind[t] for t in model.t) + sum(Loss[t]*Coeff for t in model.t)

# **Solving and Results**

model.objective = pyo.Objective(rule=objective_rule, sense=pyo.minimize)

model.write('Opt_New.lp')

opt = SolverFactory('cplex_direct')

results = opt.solve(model, tee=True)
```

```
print(results)

print('obec Fu = ' , pyo.value(model.objective), '$')

EWind_values = pd.DataFrame(list(EWind[:].value),columns=['Ewind'])

EPV_values = pd.DataFrame(list(EPV[:].value),columns=['EPV'])

Pd_values = pd.DataFrame(list(Pd[:].value),columns=['Pd'])

Pc_values = pd.DataFrame(list(Pc[:].value),columns=['Pc'])

Eb_values = pd.DataFrame(list(Eb[:].value),columns=['Eb'])

ESWind = pd.DataFrame(list(ESWind[:].value),columns=['ESWind'])

ESPV = pd.DataFrame(list(ESPV[:].value),columns=['ESPV'])

Loss = pd.DataFrame(list(Loss[:].value),columns=['Loss'])

C = pd.DataFrame(list(C[:].value),columns=['C'])

Indexed_Result                                                                    =
pd.concat([EWind_values,EPV_values,Pd_values,Pc_values,Eb_values,ESWind,ESPV,Loss],axi
s=1)

#Indexed_Result.to_excel('Summer_Actual.xlsx')

#Indexed_Result.to_excel('Summer_Optimal.xlsx')

SOC = (Eb_values/9375.303)*100

x = range(48)

y2=0

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()

ax1.fill_between(x,     EPV_values.iloc[:,0],y2,facecolor='firebrick',label     =     'PV     Used
Power',alpha=0.7)
```

```
ax1.fill_between(x, Pd_values.iloc[:,0],y2,facecolor='indigo',label = 'Battery Discharging',alpha=0.7)

ax1.fill_between(x, Pc_values.iloc[:,0],y2,facecolor='limegreen',label = 'Battery Charging',alpha=0.7)

ax1.fill_between(x, EWind_values.iloc[:,0],y2,facecolor='darkorange',label = 'Wind Used Power',alpha=0.7)

ax1.fill_between(x, ESWind.iloc[:,0],y2,facecolor='slategrey',label = 'Wind Surplus',alpha=0.7)

ax1.fill_between(x, ESPV.iloc[:,0],y2,facecolor='blue',label = 'PV Surplus',alpha=0.7)

ax1.plot(x, data.iloc[:,1], 'black',label = 'Demand',linestyle = 'dashed')

ax2.plot(x, SOC.iloc[:,:], 'black',label = 'SOC')

ax2.set_ylim([0, 110])

ax1.set_xlabel('Hours')

ax1.set_ylabel('Power Consumption (kWh)')

ax2.set_ylabel('State of Charge of the Battery %')

ax2.legend(loc='upper right')

ax1.legend(ncol=9,bbox_to_anchor=(0.5,1.05),loc='center',fontsize=8)

plt.show()

# Sensetivity on Loss Coeff

Cost1 = np.repeat(675,48)

Cost2 = np.repeat(1242,48)

Cost3 = np.repeat(1569,48)

Cost4 = np.repeat(1659,48)

Cost5 = np.repeat(1938,48)
```

```
Loss_fun = pd.read_excel('Loss.xlsx',sheet_name = 'Summer')

Loss_fun = Loss_fun.iloc[:48,0:5]

x = range(48)

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()

ax1.plot(x, Loss_fun.iloc[:,0], 'maroon',label = 'Loss for Coeff=0.01', marker= 'o')

ax1.plot(x, Loss_fun.iloc[:,1], 'green',label = 'Loss for Coeff=0.05',marker= '^')

ax1.plot(x, Loss_fun.iloc[:,2], 'navy',label = 'Loss for Coeff=0.08',marker= 'v')

ax1.plot(x, Loss_fun.iloc[:,3], 'black',label = 'Loss for Coeff=0.1',marker= 's')

ax1.plot(x, Loss_fun.iloc[:,4], 'red',label = 'Loss for Coeff=0.17',marker= 'P')

ax2.plot(x, Cost1, 'maroon',label = 'Cost for Coeff=0.01')

ax2.plot(x, Cost2, 'green',label = 'Cost for Coeff=0.05')

ax2.plot(x, Cost3, 'navy',label = 'Cost for Coeff=0.08')

ax2.plot(x, Cost4, 'black',label = 'Cost for Coeff=0.1')

ax2.plot(x, Cost5, 'red',label = 'Cost for Coeff=0.17')

ax2.set_ylim([500, 2500])

ax1.set_xlabel('Hours')

ax1.set_ylabel('Loss (kWh)')

ax2.set_ylabel('Operating Cost($)')

#ax2.legend(bbox_to_anchor=(0.9,1.08),loc='center',fontsize=8)

ax2.legend(ncol=6,bbox_to_anchor=(0.5,1.04),loc='center',fontsize=8)

ax1.legend(ncol=6,bbox_to_anchor=(0.5,1.09),loc='center',fontsize=8)

plt.show()
```

```
Load_N = pd.DataFrame(data['Load'])

Load_Optimal = pd.DataFrame(Load_N.values*C.values, columns=Load_N.columns, index=Load_N.index)

C_Load = pd.DataFrame(data['C_Load'])

plt.plot(x, C_Load, 'red', label = 'Critical Load Demand',marker= 's')

plt.plot(x, Load_N, 'navy', label = 'Actual Load Demand', marker= 'o')

plt.plot(x, Load_Optimal, 'forestgreen', label = 'Covered Load Demand',marker= '^')

plt.xlabel('Hours')

plt.ylabel('Load Demand (kW)')

plt.legend(ncol=3,bbox_to_anchor=(0.5,1.07),loc='center',fontsize=8)

plt.show()
```

Mapping Model

```
import numpy as np

import pandas as pd

import geopandas

from shapely import geometry as sg

from itertools import product

import math

def set_grid(gdf, size):

    """

    Args:

        gdf: input geo dataframe (of the circle)

        size: the square size in meters
    Return:
        the grid in 4326
    proj_gdf = gdf.to_crs(3857)
    aoi_bb = sg.box(*proj_gdf.total_bounds)
    min_x, min_y, max_x, max_y = aoi_bb.bounds
    lon = np.concatenate([np.arange(min_x, max_x, size), [max_x]])
    lat = np.concatenate([np.arange(min_y, max_y, size), [max_y]])
    squares = []
    for ix, iy in product(range(len(lon) - 1), range(len(lat) - 1)):
        square = sg.box(lon[ix], lat[iy], lon[ix + 1], lat[iy + 1])
        squares.append(square)
    # create a buffer grid in lat-long
    grid = geopandas.GeoDataFrame({"geometry": squares}, crs="EPSG:3857")
    grid = grid.to_crs(4326)
    return grid, squares
def updated_grid(res, r, x, squares):
    '''
    r and x are in km
    squares is the same as in the previous definition
    '''
    res_complete = [0.0] * int((round(2*r/x)-len(res))/2)+res+[0.0] * int((round(2*r/x)-len(res))/2)
    start_loop = [(round(2*r/x)-k)/2 if k!=0 else 0.0 for k in res_complete]
    start_loop_real = [int(k+n*20) if k!=0 else 0 for n, k in enumerate(start_loop)] #(20-
math.floor(k/2))
    end_loop_real = [int(z+ res_complete[w]) for w, z in enumerate(start_loop_real)]
    select_squares = []
    for ix in range(len(start_loop_real)):
        select_squares.append(squares[start_loop_real[ix]:end_loop_real[ix]])
    #     print(len(squares[s[ix]:e[ix]]))
    select_squares = [tt for r in select_squares for tt in r]
    grid_3857 = geopandas.GeoDataFrame({"geometry": select_squares}, crs="EPSG:3857")
    grid_3857_centroid = grid_3857.centroid
    grid = grid_3857.to_crs(4326)
    grid_centroid = grid_3857_centroid.to_crs(4326)
    return grid, grid_centroid, grid_3857


def grid_land_type(Base_map, square_map):
    land_type = []
    for x in range(len(square_map.geometry.values)):
        ref_gdf = geopandas.GeoDataFrame({"geometry":square_map.iloc[x,:].values},
crs="EPSG:4326")
        ex_data_clip = geopandas.clip(Base_map, ref_gdf)
        ex_data_clip = ex_data_clip.to_crs(3857)
        land_type.append(ex_data_clip[ex_data_clip.area ==
max(ex_data_clip.area)]._category.values[0])
```

```
    return land_type
def clean_up_grid(Circle_map, grid_map):
    clips_df =[]
    square_clip = []
    for x in range(len(grid_map)):
        grid_1            =            geopandas.GeoDataFrame({"geometry":grid_map.iloc[x,:].values},
crs="EPSG:4326")
        df_clip =geopandas.clip(Circle_map, grid_1)
        if df_clip.empty==False:
            if df_clip.is_empty[0]==False:
                clips_df.append(df_clip)
#               print((df_clip.to_crs(3857).area/grid_1.to_crs(3857).area)[0]*100)
                if (100*(df_clip.to_crs(3857).area/grid_1.to_crs(3857).area)[0])>=55:
                    square_clip.append(grid_1)
                else:
                    pass
    return clips_df, square_clip
def distance(df1, df2):
    df_distance=df1.to_crs("EPSG:3857").geometry.apply(lambdag:
df2.to_crs("EPSG:3857").distance(g))
    if len(df2)==1:
        df_distance = df_distance.rename(columns={0: "Distances"})
    return df_distance
```

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import geopandas

import gurobipy as gp

from gurobipy import GRB

from shapely.geometry import Point

from shapely import geometry as sg

from itertools import product

import math

from geopandas import GeoDataFrame

import geopandas as gpd

resf1 = "57331/dmti_landcover_2021_s_poly.shp"

Land_1 = geopandas.read_file(resf1)


res1 = "57332/ppat_affect_terri_2020_s_poly.shp"

Land_2= geopandas.read_file(res1)

fp3 = "canvec_50K_QC_Res_MGT_shp/canvec_50K_QC_Res_MGT/power_line_1.shp"

pwline_df = geopandas.read_file(fp3)

fp1                                                                                    =
"canvec_50K_QC_Res_MGT_shp/canvec_50K_QC_Res_MGT/transformer_station_0.shp"

tstation_df = geopandas.read_file(fp1)
```

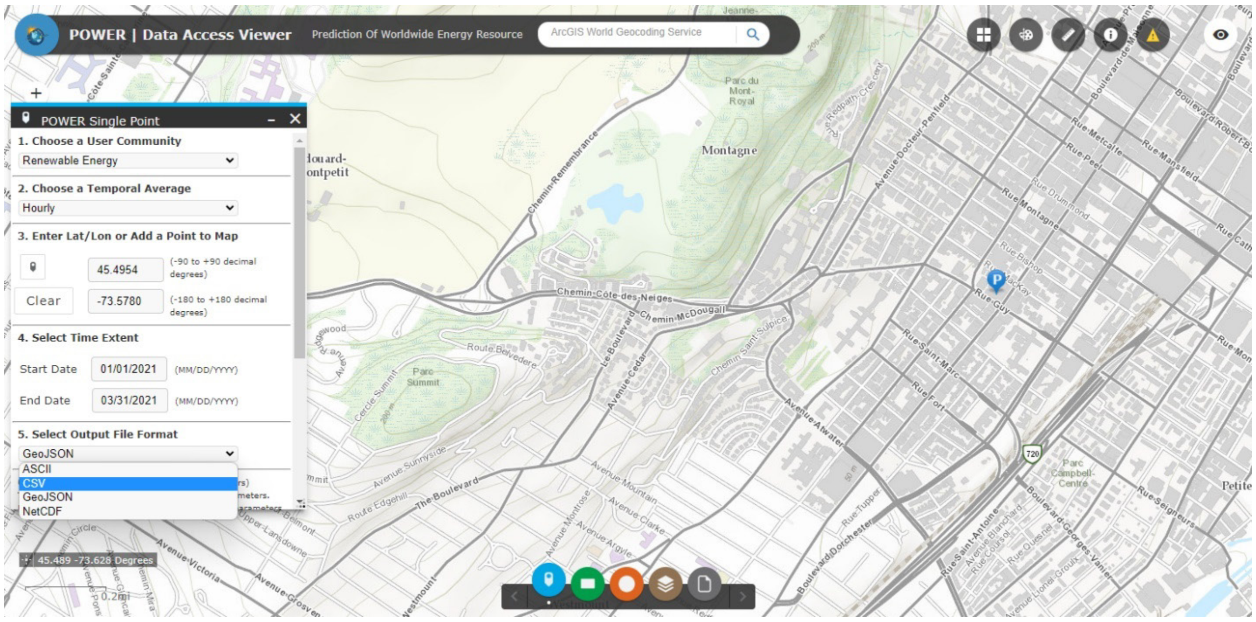
```
pwline_df = pwline_df.to_crs(epsg=4326)

Land_C2 = pd.concat([Land_2.iloc[:319,],Land_2.iloc[323:,]]).reset_index(drop=True) #319-322

Land_C1 = pd.concat([Land_1.iloc[:280,],Land_1.iloc[283:,]]).reset_index(drop=True)

rad =20

side =4

#Circle geo information

df = pd.DataFrame(

    {

        'lat':[45.5],

        'lon':[-73.6],

        'rad':[rad*1000]

    }

)

gdf = geopandas.GeoDataFrame(

    df, geometry=geopandas.points_from_xy(df.lon, df.lat), crs="epsg:4326")

gdf_centre = gdf.copy()

gdf = gdf.to_crs('epsg:3857') #or 3797 or 3347

gdf['geometry'] = gdf.geometry.buffer(df.rad)

gdf=gdf.to_crs('epsg:4326')

from Square_grid_func_Updated import
set_grid,updated_grid,grid_land_type,clean_up_grid,distance

grid, squares =set_grid(gdf, side*1000)

#grid_update, grid_update_centroid, grid_update_3857 =updated_grid(res, rad, side, squares)
```

```
clips_df, square_clip = clean_up_grid(gdf,grid)

land_array=grid_land_type(Land_C1, pd.concat(square_clip, ignore_index=True)) # When change to Montreal change it to C1

griddf = pd.concat(square_clip, ignore_index=True)

fig, ax = plt.subplots(1, 1, figsize=(10,10))

Land_C2.plot(ax=ax, column = "_theme_pro",cmap="viridis", lw=0.7, legend_kwds={'loc': 'lower right','bbox_to_anchor':(1.28, 0.75),'title':"Outside of Greater Montreal"})

leg1 = ax.get_legend()

#leg1.set_title("First graph legend")

Land_C1.plot(ax=ax, column = "_category",cmap="bone", lw=0.7,legend=True, legend_kwds={'loc': 'lower left','bbox_to_anchor':(1.03, 0.55),'title':"Greater Montreal"})

pwline_df.plot(ax = ax, color='red', linewidth=2*pwline_df['numli'])

tstation_df.plot(ax = ax, color='red')

gdf.plot(ax=ax, label = "Circle",edgecolor="green", facecolor="None", lw=2, zorder=12)

#grid_update.plot(ax=ax, edgecolor="red", facecolor="None", label="Square box", zorder=11)

for y in square_clip:

    y.plot(ax=ax, edgecolor="black", facecolor="None",zorder=11)

# grid.plot(ax=ax, edgecolor="black")

# gdf.geom.plot(ax=ax, label = "Circle",edgecolor="green", facecolor="None", lw=2, zorder=11)

# gdf.geometry.plot(ax=ax, label = "Centre", color="red",zorder=11)

gdf_centre.plot(ax=ax)

test_gdf1 = geopandas.GeoDataFrame({"geometry":griddf.iloc[62,:].values}, crs="EPSG:4326")

# Bottom to top, left to right (Start from left)
```

```
test_gdf1.plot(ax=ax, edgecolor="yellow", facecolor="None", lw=3, zorder=63)

ax.set_xlim(-75, -72)

ax.set_ylim(45, 46.5)

ax.add_artist(leg1)

plt.show()

sq_centroid_df = pd.concat(square_clip, ignore_index=True).to_crs(3857).centroid.to_crs(4326)

dist= distance(sq_centroid_df, gdf_centre)

val3 = distance(tstation_df, sq_centroid_df)

# Selecting the nearest transformer to each square

dist2_M = val3.min(axis=0)

# List of Land Cost

List_LC=[]

# List of Turbine Density

List_TD=[]

LC = [5,40,30,60,80,25,800]

TD = [100,10,20,40,60,20,10]

land_array = np.array(land_array)

for x in range (len(land_array)):

    if land_array[x] == 'OPEN AREA':

        List_LC.append(LC[0])

        List_TD.append(TD[0])

    if land_array[x] == 'COMMERCIAL':

        List_LC.append(LC[1])
        List_TD.append(TD[1])
    if land_array[x] == 'GOVERNMENT AND INSTITUTIONAL':
        List_LC.append(LC[2])
        List_TD.append(TD[1])
    if land_array[x] == 'PARKS AND RECREATIONAL':
        List_LC.append(LC[3])
        List_TD.append(TD[1])
    if land_array[x] == 'RESIDENTIAL':
        List_LC.append(LC[4])
        List_TD.append(TD[1])
    if land_array[x] == 'RESOURCE AND INDUSTRIAL':
        List_LC.append(LC[5])
        List_TD.append(TD[1])
    if land_array[x] == 'WATERBODY':
        List_LC.append(LC[6])
        List_TD.append(TD[1])
```

Appendix II: Sample Data

NASA Data Portal


POWER | Data Access Viewer

Prediction Of Worldwide Energy Resource

ArcGIS World Geocoding Service

POWER Single Point

1. Choose a User Community

Renewable Energy

2. Choose a Temporal Average

Hourly

3. Enter Lat/Lon or Add a Point to Map

45.4954 (-90 to +90 decimal degrees)

Clear -73.5780 (-180 to +180 decimal degrees)

4. Select Time Extent

Start Date 01/01/2021 (MM/DD/YYYY)

End Date 03/31/2021 (MM/DD/YYYY)

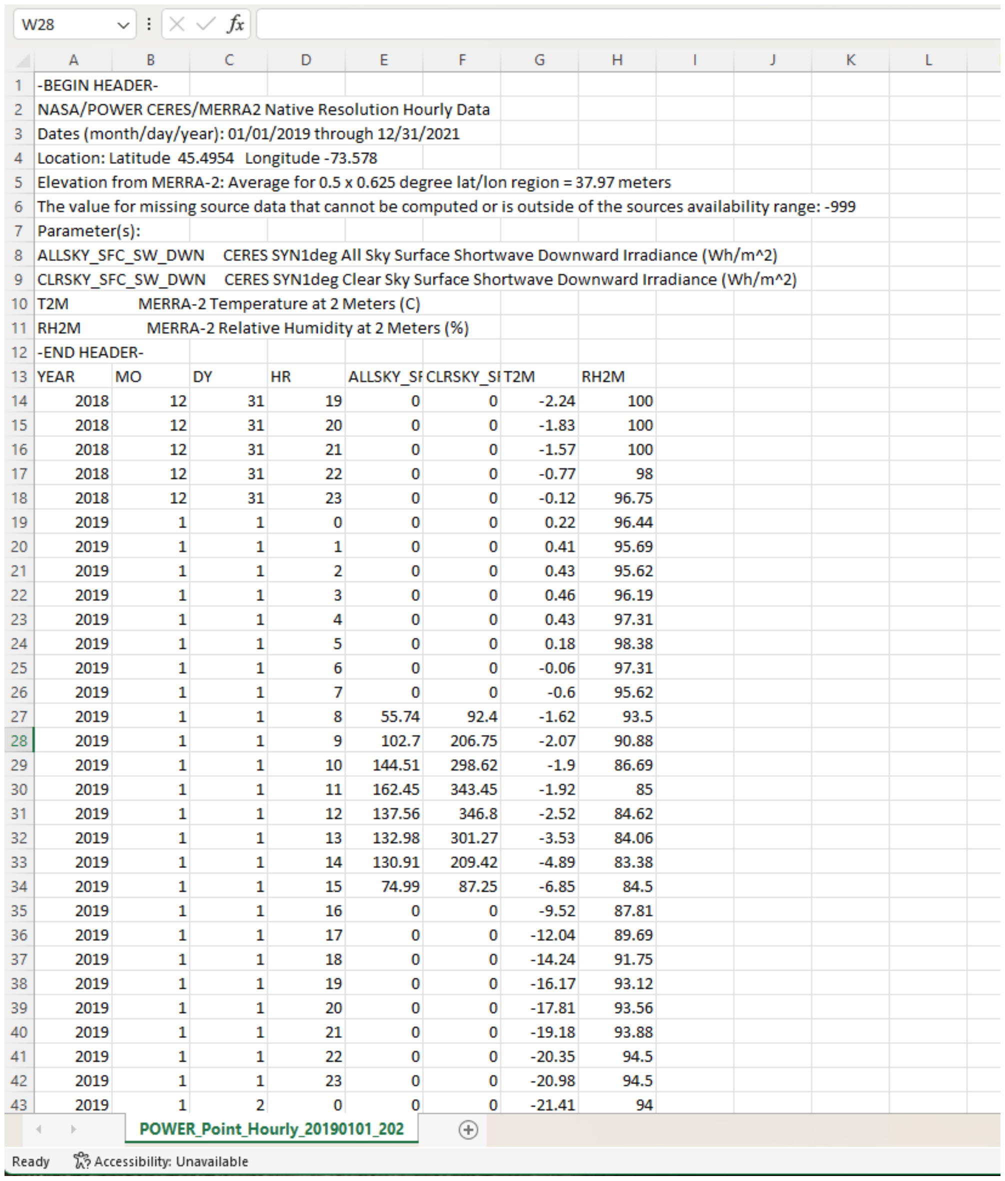
5. Select Output File Format

GeoJSON

- ASCII
- CSV
- GeoJSON
- NetCDF

45.489 -73.628 Degrees

Sample Downloaded Data from NASA Portal



-BEGIN HEADER-

NASA/POWER CERES/MERRA2 Native Resolution Hourly Data

Dates (month/day/year): 01/01/2019 through 12/31/2021

Location: Latitude 45.4954 Longitude -73.578

Elevation from MERRA-2: Average for 0.5 x 0.625 degree lat/lon region = 37.97 meters

The value for missing source data that cannot be computed or is outside of the sources availability range: -999

Parameter(s):

ALLSKY_SFC_SW_DWN CERES SYN1deg All Sky Surface Shortwave Downward Irradiance (Wh/m^2)

CLRSKY_SFC_SW_DWN CERES SYN1deg Clear Sky Surface Shortwave Downward Irradiance (Wh/m^2)

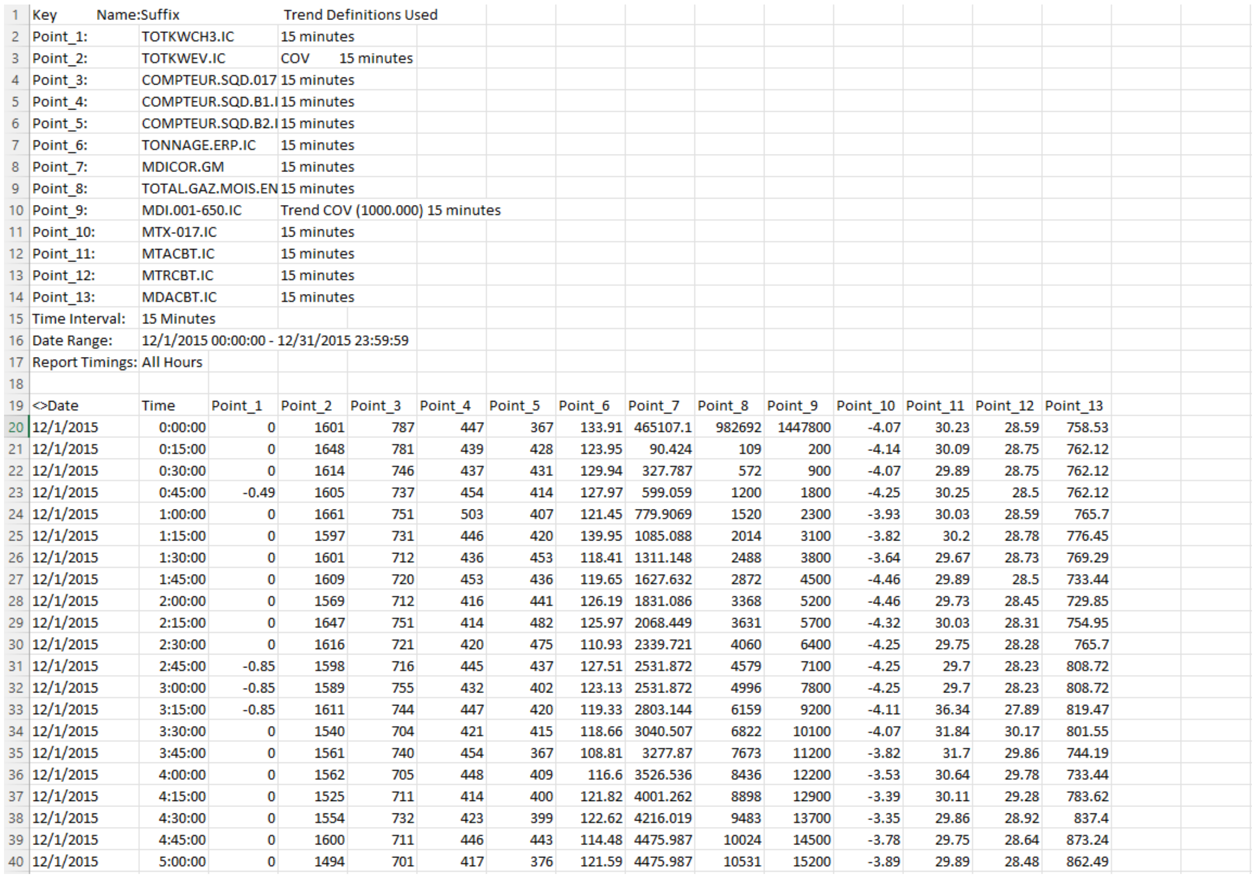
T2M MERRA-2 Temperature at 2 Meters (C)

RH2M MERRA-2 Relative Humidity at 2 Meters (%)

-END HEADER-

YEAR	MO	DY	HR	ALLSKY_SF	CLRSKY_SI	T2M	RH2M
2018	12	31	19	0	0	-2.24	100
2018	12	31	20	0	0	-1.83	100
2018	12	31	21	0	0	-1.57	100
2018	12	31	22	0	0	-0.77	98
2018	12	31	23	0	0	-0.12	96.75
2019	1	1	0	0	0	0.22	96.44
2019	1	1	1	0	0	0.41	95.69
2019	1	1	2	0	0	0.43	95.62
2019	1	1	3	0	0	0.46	96.19
2019	1	1	4	0	0	0.43	97.31
2019	1	1	5	0	0	0.18	98.38
2019	1	1	6	0	0	-0.06	97.31
2019	1	1	7	0	0	-0.6	95.62
2019	1	1	8	55.74	92.4	-1.62	93.5
2019	1	1	9	102.7	206.75	-2.07	90.88
2019	1	1	10	144.51	298.62	-1.9	86.69
2019	1	1	11	162.45	343.45	-1.92	85
2019	1	1	12	137.56	346.8	-2.52	84.62
2019	1	1	13	132.98	301.27	-3.53	84.06
2019	1	1	14	130.91	209.42	-4.89	83.38
2019	1	1	15	74.99	87.25	-6.85	84.5
2019	1	1	16	0	0	-9.52	87.81
2019	1	1	17	0	0	-12.04	89.69
2019	1	1	18	0	0	-14.24	91.75
2019	1	1	19	0	0	-16.17	93.12
2019	1	1	20	0	0	-17.81	93.56
2019	1	1	21	0	0	-19.18	93.88
2019	1	1	22	0	0	-20.35	94.5
2019	1	1	23	0	0	-20.98	94.5
2019	1	2	0	0	0	-21.41	94

Sample Measured Power Consumption Data Received from Concordia's Facility Management



Key	Name:Suffix	Trend Definitions Used
Point_1:	TOTKWCH3.IC	15 minutes
Point_2:	TOTKWEV.IC	COV 15 minutes
Point_3:	COMPTEUR.SQD.017	15 minutes
Point_4:	COMPTEUR.SQD.B1.I	15 minutes
Point_5:	COMPTEUR.SQD.B2.I	15 minutes
Point_6:	TONNAGE.ERP.IC	15 minutes
Point_7:	MDICOR.GM	15 minutes
Point_8:	TOTAL.GAZ.MOIS.EN	15 minutes
Point_9:	MDI.001-650.IC	Trend COV (1000.000) 15 minutes
Point_10:	MTX-017.IC	15 minutes
Point_11:	MTACBT.IC	15 minutes
Point_12:	MTRCBT.IC	15 minutes
Point_13:	MDACBT.IC	15 minutes
Time Interval:	15 Minutes	
Date Range:	12/1/2015 00:00:00 - 12/31/2015 23:59:59	
Report Timings:	All Hours	

<>Date	Time	Point_1	Point_2	Point_3	Point_4	Point_5	Point_6	Point_7	Point_8	Point_9	Point_10	Point_11	Point_12	Point_13
12/1/2015	0:00:00	0	1601	787	447	367	133.91	465107.1	982692	1447800	-4.07	30.23	28.59	758.53
12/1/2015	0:15:00	0	1648	781	439	428	123.95	90.424	109	200	-4.14	30.09	28.75	762.12
12/1/2015	0:30:00	0	1614	746	437	431	129.94	327.787	572	900	-4.07	29.89	28.75	762.12
12/1/2015	0:45:00	-0.49	1605	737	454	414	127.97	599.059	1200	1800	-4.25	30.25	28.5	762.12
12/1/2015	1:00:00	0	1661	751	503	407	121.45	779.9069	1520	2300	-3.93	30.03	28.59	765.7
12/1/2015	1:15:00	0	1597	731	446	420	139.95	1085.088	2014	3100	-3.82	30.2	28.78	776.45
12/1/2015	1:30:00	0	1601	712	436	453	118.41	1311.148	2488	3800	-3.64	29.67	28.73	769.29
12/1/2015	1:45:00	0	1609	720	453	436	119.65	1627.632	2872	4500	-4.46	29.89	28.5	733.44
12/1/2015	2:00:00	0	1569	712	416	441	126.19	1831.086	3368	5200	-4.46	29.73	28.45	729.85
12/1/2015	2:15:00	0	1647	751	414	482	125.97	2068.449	3631	5700	-4.32	30.03	28.31	754.95
12/1/2015	2:30:00	0	1616	721	420	475	110.93	2339.721	4060	6400	-4.25	29.75	28.28	765.7
12/1/2015	2:45:00	-0.85	1598	716	445	437	127.51	2531.872	4579	7100	-4.25	29.7	28.23	808.72
12/1/2015	3:00:00	-0.85	1589	755	432	402	123.13	2531.872	4996	7800	-4.25	29.7	28.23	808.72
12/1/2015	3:15:00	-0.85	1611	744	447	420	119.33	2803.144	6159	9200	-4.11	36.34	27.89	819.47
12/1/2015	3:30:00	0	1540	704	421	415	118.66	3040.507	6822	10100	-4.07	31.84	30.17	801.55
12/1/2015	3:45:00	0	1561	740	454	367	108.81	3277.87	7673	11200	-3.82	31.7	29.86	744.19
12/1/2015	4:00:00	0	1562	705	448	409	116.6	3526.536	8436	12200	-3.53	30.64	29.78	733.44
12/1/2015	4:15:00	0	1525	711	414	400	121.82	4001.262	8898	12900	-3.39	30.11	29.28	783.62
12/1/2015	4:30:00	0	1554	732	423	399	122.62	4216.019	9483	13700	-3.35	29.86	28.92	837.4
12/1/2015	4:45:00	0	1600	711	446	443	114.48	4475.987	10024	14500	-3.78	29.75	28.64	873.24
12/1/2015	5:00:00	0	1494	701	417	376	121.59	4475.987	10531	15200	-3.89	29.89	28.48	862.49