Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Master

CO_2 emissions related to electricity, an architecture implemented in Python

Praveenkumar Beedanal.

Course of Study:	Master's in Computer Science
Examiner:	Prof. Dr. Marco Aiello
Supervisor:	Laura Fiorini, M.Sc.
Commenced:	July 1, 2018
Completed:	December 21, 2018

Abstract

In recent years, the environmental impact of energy consumption has gained attention. In this thesis, we create an android application that shows CO_2 emission intensity value of the Germany electricity grid using ENTSOE Transparency Platform API. The application shows the latest available CO_2 information, total electricity production, and electricity produced by each resource type for every fifteen-minutes time intervals. In addition to this, we provided the last 24hours generation mix and CO_2 emission information. Moreover, we formulate the optimization problem for scheduling of hybrid household appliances using mixed integer linear programming technique. The objective of our approach is to minimize the CO_2 emission from household while considering the user preference. We scheduled the appliance based on both load shift in time and combination of multiple energy carriers. We consider electricity, hot water, and natural gas energy carriers. We use a gas boiler to produce hot water, that can be stored in a hot water storage tank. Electricity and natural gas are supplied by distribution grids.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Problem Statement

In recent decades, increased electrification and modernization of power grids are strong trends in today's market[SBN14]. The household appliances responsible for 29% of total electricity demand in Europe [HBF14]. Thereby participation of customer to reduce electricity consumption during peak hours are being highly discussed in today's market[SBN14]. Because the electricity generated during peak hours is dirtier than generated during off-peak hours subjected to weather constraint(e.g., sunny days produce more solar). But most of the situation reducing the consumption during peak hours will make a positive environmental impact. But so far, the participation of the residential sector has been delayed due to lack of hourly information and lack of high-speed communication between retailer and household[SBN14]. "Demand response (DR) is defined as changes in the electricity consumption patterns of end consumers to reduce the instantaneous demand in times of high electricity prices"[LMDB12]. The most common demand response program is price-based, but it is still lower customer acceptance. Therefore to change the consumption habits require stronger encouragement rather than a financial incentive. If price and CO_2 emissions are positively co-related, then load shift or reduce from high price hours resulted in a reduction of carbon emission. Therefore it is important to provide CO_2 emission information to the consumer that uses electricity generation mix data as input to calculate the emission value. It would enable co-optimization of electricity consumption cost and CO_2 emission reduction[SBN14].

There has been a huge transformation from fossil energy generation and centralized power plants, eg., nuclear and coal-fired power plants, to renewable energy and distributed generation, e.g., wind power and combined heat and power(CHP), is one of the significant challenges in current society. It includes multiple energy carriers like electricity, gas and hot water. The flexibilization of energy consumption is important, due to fact that the renewable energies used for the electricity generation are fluctuating and electricity storage like a battery is expensive or pumped-storage hydroelectricity faces heavy opposition by population[MSS15]. But until now, the focus is mainly on the flexibilization of electricity consumption. This excludes the possibility to shift energy consumption from one energy carrier(electricity, hot water, and natural gas) to other. Scheduling of hybrid appliance(uses more than one energy carrier) along with optimization of energy carriers are important features to consider while building future Energy Management System(EMS)[MSS15].

1.2 Research Questions

• How to provide CO_2 emission intensity value quarterly to the consumer as part of a demand response program?.
• To what extent can CO_2 based DR reduce the environmental impact of households?.

1.3 Methodology

We have used an android application to shown CO_2 emission intensity value. Because according to survey done by[Sta18], the number of smartphone users worldwide growing exponentially as shown in fig 1.1. Hence it is an easy way to establish communication between electricity producers and consumers as part of a demand response program.

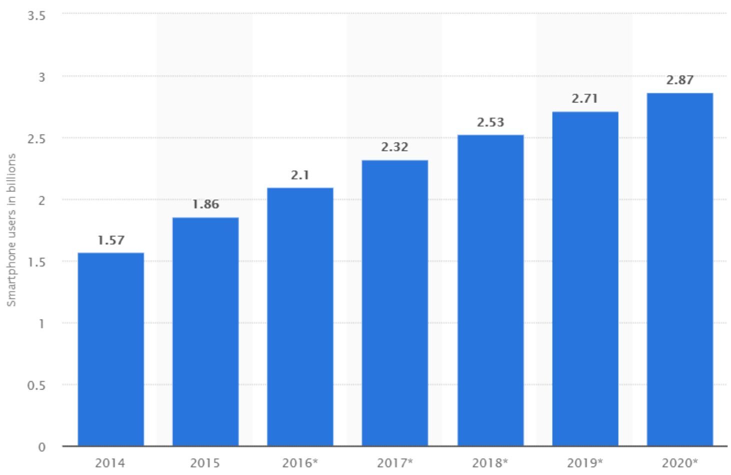

Figure 1.1: Smart Phone Users worldwide from 2014 to 2020(in billions) [Sta18]

The objective of our model is to minimize the CO_2 emission from household by satisfying the constraints. In this thesis, We have scheduled the hybrid household appliances that satisfy the constraint related to the appliance(like energy and time) and user (time preference). Since our problem concerned with minimization, clearly shows possibilities to get benefits from the optimization problem. We have to consider the status of the appliances like, whether it is processed by a particular energy source at a particular time slot or not. These are called decisions in the optimization problem, which is restricted to take only binary variables either 1 or 0.

1.4 Thesis Organization

The thesis is organized as follow. In Chapter 2, we have discussed the papers related to CO_2 emission intensity and scheduling of appliances with their advantages and disadvantages by comparing our proposed system. In subsequent Chapter 3, we have given some background information to readers that help to understand our proposed model effectively. The readers will find the Chapter 4, where we actually discussed our proposed CO_2 emission android application and scheduling of hybrid appliances. The Chapter 5, describes the hardware and software requirements. In Chapter 6, we discussed how we actually implement python client application, an android application that displays Germany CO_2 emission intensity value and mathematical model for our hybrid appliance scheduling problem. In Chapter 7 and Chapter 8, we presented some of the use cases related to our scheduling problem and displaying their results, respectively. At the end Chapter 9 and Chapter 10, we have summarized our research and provide ideas to enhance the research in the future.

2 Related work

In this chapter, we discuss the related works relevant to this thesis and their advantages and disadvantages. This chapter has two sections, in the first one, we are going to discuss the dynamic CO_2 emission from the electricity generation plants and the next section we provide an explanation about the scheduling of household appliances related papers.

2.1 CO_2 Emission Intensity

In recent years, CO_2 emission intensity information plays an import role in demand response incentive-based program. It encourages the electricity producing companies to produce electricity using renewable energy. Along with that the household heating and cooling are responsible for the consumption of approximately 21% of overall appliance consumption in US[CWF12] and household is responsible for 29% of total electricity demand in Europe [HBF14]. Hence it encourages the consumers to save electricity from their side during the peak load hours. Indirectly helps in the reduction of CO_2 emission from household appliances.

In paper [RP18], GridCarbon is the smartphone application, that calculates the carbon intensity value of the United Kingdom(UK) electricity grid. Since the launch of this application renewable energy generation within UK grid has grown significantly and now it represents the major portion of electricity production. The application makes use of two data sources both are available through ELEXON portal[1]. The first data source provides the electricity generation per fuel type, which is updated for every 5 minutes[2]. The second data source provides the wind and solar power generation data, which is updated for every 30 minutes[3]. The application provides both metered and un-metered wind and solar electricity information and includes losses occurred during transmission are the main advantages. The disadvantage is, sometimes data missing will occur, but it is very short. The important difference between our application and grid carbon is, it provides CO_2 emission information of United Kingdom(UK) for every five minutes, but our application provides emission information of Germany grids for every fifteen minutes time interval. Both gridcarbon application and our android application provides almost similar information like generation mix and CO_2 emission intensity value.

The paper [SBN14] describes the discussion between policymakers, researchers and executive of future electric systems, who are trying to reduce or shift the electricity load during peak demand hours through demand-response tariff. Since demand response rates have less acceptance among

[1] www.elexonportal.co.uk

[2] api.bmreports.com/BMRS/FUELINST/v1?APIKey=<APIKey>&ServiceType=xml

[3] api.bmreports.com/BMRS/B1630/v1?APIKey=<APIKey>&SettlementDate=<SettlementDate>&Period=<Period>&ServiceType=xml

customer because they need a stronger reason to change their habit rather than just any financial incentive. Hence authors decided to provide dynamic CO_2 intensity signal that could give motivation to the customer to shift or reduce from peak hours. Eventually, it would optimize both electricity consumption costs and carbon emission reduction from the customer side. The authors of paper[SBN14] calculated the dynamic CO_2 signal using electricity generation data, instead of the capacity of the plant. We have followed a similar approach to calculate the CO_2 signal. In this paper, they didn't introduce about providing the CO_2 information through an android application, but we implemented an android application that provides CO_2 emission value in our research. The disadvantage of the paper [SBN14] is, they have used historical data to study the correlation between dynamic CO_2 intensity and electricity market price.

2.2 Scheduling of Household Appliances

In recent years, due to increased electrification in markets, the power demand reached a new peak level and create an imbalance between consumption and generation. In European countries, highest power demand occurs when outdoor temperature falls[SWSJ11]. Load balancing can be accomplished by scheduling the household to times when renewable energy generation is high. But it is unrealistic to expect consumer has to identify the peak demand and schedule the operation of appliances. Hence it requires automatic decision support that takes control of appliances or provides schedule information to consumer[SWSJ11].

In paper [SWSJ11], the authors modeled the appliance scheduling problem using mixed integer linear programming(MILP). They used an optimal power signal to minimize the cost of the electricity bill, which has to satisfy the operational constraints and user preference. Although model provides a good quality solution in a reasonable amount of computation time, it is not scalable in terms of the appliance. The model failed to find the first feasible solution with 20 appliances. Since the model proposed by the paper[SWSJ11] focus on the appliance scheduling in terms of shift load in time. But we considered, scheduling household appliances in terms of both load shift in time and energy carrier.

The authors in paper [QNK+15], also modeled the appliance scheduling problem using MILP technique. But in addition to power consuming appliances, they considered the power producing appliance like Photo-voltaic panel. That helps in reducing the electricity bill and supply excess electricity to the national grid after satisfying the home demand. They created two separate objectives for this scheduling problem. One deals with minimizing the cost of electricity and other deals with minimizing maximum peak load. Even though, they considered distributed renewable energy production system, didn't explore the hybrid nature of the appliances is one of the downsides of this approach. The "Hybrid appliances are those use at least two energy carrier alternatively". For instance, a washing machine can use electricity or electricity and hot water for their operation. In our approach, we have considered the hybrid nature of the appliances and their scheduling with respect to load shift in time and energy carrier.

In paper [BHH+12], authors proposed the multi-objective optimization model using MILP technique, while properly considered customer preference and comfort level. They formulate the mathematical model of major household appliances like fridge, freezer, dishwasher, washer, dryer, stove, water heater, hot tub, and pool pumps. Along with those, the mathematical model of other household components like lighting, heating, air conditioning and PV panel, and storage device are proposed.

The authors gave a choice to residential user select following combination of objective functions, minimization of the cost of electricity and gas, total energy consumption, peak load, and CO_2 emission. In this scenario, the model considered the energy carrier like electricity and gas, along with customer comfort and customer preference. But they considered only traditional appliances which use only one energy carrier like electricity or gas, but not both. Moreover, in our approach, we choose energy carriers like electricity, gas and hot water. Also, considered the hybrid appliances and scheduling the appliance with respect to load shift according to CO_2 emission value and energy carrier with respect to emission.

In paper [SXZ14], the author proposed a mixed integer nonlinear program(MINLP) optimization model for scheduling typical household appliances in order to minimize the electricity cost and earn an incentive. The model has given flexibility to consumer choose their scheduling preference(in terms of their convenience). By choosing different values of the weighting factor that is associated with varying cost. Therefore consumer has to ready make trade-off between their convenience and cost and made the final decision about participation. They considered only one household for scheduling, whereas in our approach we considered thirty households along with the user preference. In addition, they considered single energy carrier and traditional appliances in their model. As far our approach concern, we considered multiple energy carriers and scheduling of hybrid appliances.

The paper [MSS15] proposed energy management system(EMS) that optimize the energy generation and consumption with respect all different energy carriers and electricity supply to distribution grids. Introduced the hybrid household appliances,“ refers to appliances that use at least two energy carriers alternatively“. For instance, washing machine can use either electricity to heat up the water themselves or the hot water from the storage tank present in the building, which has been stored by a gas boiler or insert heating element(IHE).

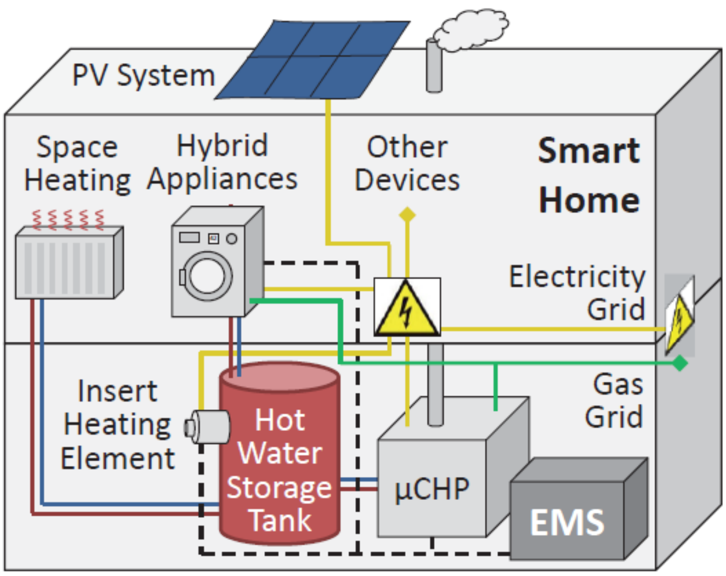

Figure 2.1: Smart Home Scenario [MSS15]

They have considered the smart home as shown in figure 2.1 with a photovoltaic system, micro combined heat and power plant(μCHP), an electric insert heating element(IHE), hot water storage tank and hybrid appliances. Hot water is either produced by μCHP or the IHE and that can be stored in the hot-water storage tank, it makes the system loosely coupled between the generation and consumption of electricity and hot water. Therefore, the hybrid appliances can use either hot-water from a storage tank or heat the water themselves. They used the energy management system(EMS) based on the Organic Smart Home [4](OSH) to solve the optimization problem. The OSH has been designed following the principles of Organic Computing[All13]. The main advantage of the EMS is, it's flexible to take any goal, such as minimization of cost or maximization of self-consumption. Also, considers the complex smart home scenario, compare to our scenario. But OSH uses a Genetic algorithm(GA) to solve the optimization problem. Typically, we use the GA algorithm when the objective function is stochastic, highly nonlinear, discontinuous and non-differentiable. In our approach, we solved our optimization problem by using standard optimization algorithms. Therefore, our model tends to accurate and efficient. Also, they simulate the appliance with a goal of minimizing the electricity cost, whereas we considered the minimization of CO_2 emission.

The authors in paper [SBP+08] proposed the strategies that investigate how smart home appliances can contribute to the load management in the future energy system. The project analyses option of load-shifting for the variety of home appliances and it expects that appliances would be integrated with renewable energy in the future. The renewable energy is partly intermittent hence require smarter generation management, storage capacities, and demand. In one of the chapter, the authors discussed different technical possibilities for flexible operations of the 10 appliances which are a dishwasher, washing machine, tumble dryer, refrigerator, freezer, oven and stove, heating circulation pump, air conditioner, electric water heater, and electric heating. Each device analysis is structured as follows: an overview of technical functionality, energy and water consumption as well as power demand of the appliance are investigated. The subsequent chapters concentrate on the potential of each appliance under smart energy conditions. They presented the technical elements needed to enable the smart operation of the appliance, and their additional cost and energy requirement. They investigate the possibilities of a device for use of renewable energy and CHP in connection to other technologies and storage capacity. Even though the paper points out the specific opportunities and restriction for each appliance, they did not propose a proper solution to the problem. They did not mention above the features of hybrid appliances like in the paper [MSS15]. But in our approach, we have taken the advantage of strategies mentioned in the paper [SBP+08] and used the load demand data presented for various technical possibilities of appliance operation. We developed the optimization model to hybrid appliances scheduling problem with the objective to minimize the CO_2 emission.

In paper [BHJ15], the author presents the primary motivation to introduce the "energy hub" in residential building is save from energy crisis along with environmental concern. The energy hub includes energy production, conversion, and storage technologies such as combined cooling, heating and power systems(CCHPs), renewable energy resources(RESs), batteries and thermal energy storage(TES).

The energy hub consists of generation and storage devices like CCHP, PV panels, Plug-in Hybrid Electric Vehicle(PHEV), TES as shown in figure 2.2. The proposed hub receives the electricity, natural gas and solar radiation in its input port to supply the electrical, heating and cooling demands at the output port. An optimization problem has been formulated based on the proposed energy hub to optimally schedule the household appliances, production, and storage components (i.e. CCHP, PHEV, TES) by managing the thermal and electrical loads. The problem is solved with the objective

[4] https://sourceforge.net/projects/osmarthome/

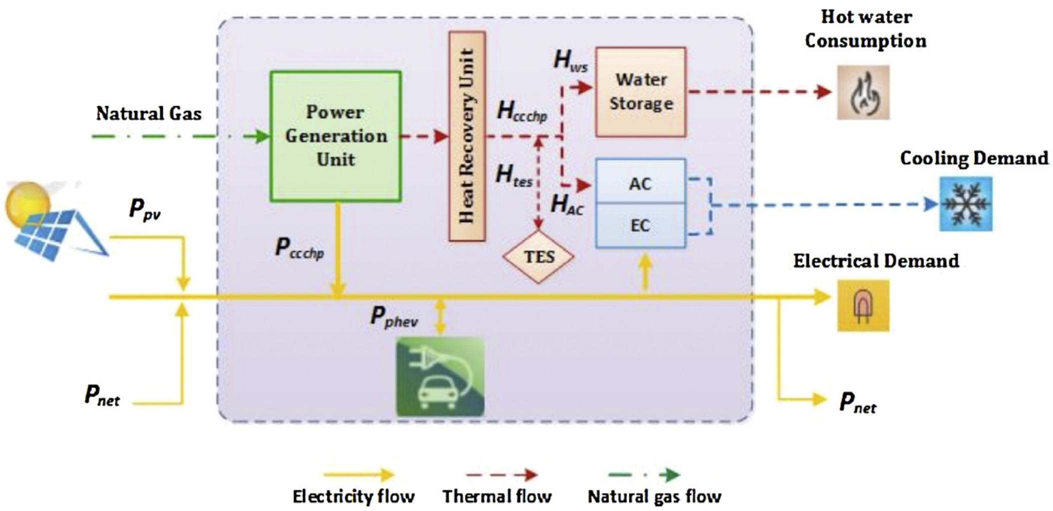

Figure 2.2: The proposed residential energy hub model [BHJ15]

is to minimize the electricity cost while considering the customer preference in terms of operation time and hot water and indoor temperature. Furthermore, a multi-objective optimization problem is considered to find the consumer's contribution to CO_2, NO_x and SO_x emission. The proposed hub operationally flexible to supply the required demands like electrical and thermal[BHJ15]. Even though, they considered the multiple energy carriers in the energy hub, but solve the scheduling problem with respect to only load shift in time and considered the traditional appliances. But in our approach, we scheduled the hybrid home appliances with respect to load shift in time and energy carrier CO_2 emission.

In paper [ZGSS11], the authors used a mixed integer linear programming technique(MILP) to formulate the scheduling problem. It schedules both operations of the household appliances and distributed energy generation across multiple homes based on the users' requirement, with the objective to reduce the energy cost and peak demand. They considered microgrid into account which can work as a local energy provider to domestic buildings and reduce the electricity cost. They have a taken a case study of thirty homes connected to single micro-grid that includes wind generator, CHP generator, boiler, thermal storage, and electrical storage to provide the basic electricity and it also connected to the national grid to obtain the electricity during the peak hours or sell the excess generated electricity back to the grid. The advantage of the proposed model is, they included the operation and maintenance cost of the CHP generator, wind generator, electrical storage, and thermal storage, electricity cost purchased from the grid and total electricity sold to the grid in the objective function of the model. They schedule the appliances and distributed energy generator based on the tariff of electricity and user preference. But in our approach, we schedule the hybrid appliance considering load shift in time and energy carrier while considering user preference.

3 Background

3.1 Introduction to Optimization

According to [Gui18b], “**optimization** is an important tool in making decisions and in analyzing physical system“. In mathematical terms, finding the best solution among a set of feasible solution is called as **optimization problem**. The important step in the optimization process is classifying the optimization model because algorithms for solving problems are fitted to a particular type of problem.

3.1.1 Model Construction

In [Gui18b] explained, the optimization process starts with constructing an appropriate model. **Modeling** is the process of identifying and expressing the **Objective**, the **Variables** and**Constraints** of the problem.

• We are going to measure the performance of the system by setting a *objective* to minimize or maximize. For instance, in the manufacturing industry we may want to maximize the profit or minimize the cost of production.

• *Unknowns* components of systems for that we want to find out the values are called as *variables*. For instance, in manufacturing industry variables may be an amount of each resource consumed or time spent on each activity.

• To describe the relationships between the variables and set the allowable values for the variables are done by the functions called as *constraints*. For instance, in the manufacturing industry, the quantity of resource consumed should not exceed the available resource amoun

3.1.2 Discrete Optimization

According to [Gui18c], models that allow variables take values from the discrete set, often subset of integers. Such models with discrete variables are called discrete optimization problems. Even if some or all of the variables in a model takes values from a discrete set, the model belongs to discrete optimization. Integer linear programming is one of the branches of discrete optimization whose discrete set is always a subset of integers.

3.1.3 Integer Linear Programming

According to [Gui18a], in a general integer linear programming problem aim is to minimize the overall linear cost function, by imposing a set of linear equality and inequality constraints as well as integral restrictions on some or all of the variables in n-dimensional x.

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in Z^n \end{aligned}$$

• If some of the variables x_i ∈x are allowed to take integer values and some are allowed take real values then it is called **mixed integer linear programming(MILP)** problem. If anyone of the objective function and/or the constraints functions are nonlinear functions, then the problem is called **mixed integer nonlinear programming(MINLP)** problem.

• The problem is called as **pure integer programming** if all of the variables x_i ∈x are restricted to take on integer values.

• If all of the variables x_i ∈x are allowed to take only binary values(0 or 1), then problem is called as **binary optimization** problem.

3.1.4 Selecting Software

According to [Gui18b], the third step in the optimization process is to select the software appropriate for the optimization problem that we are solving. Optimization software available in two different kinds of package.

• *Solver software* helps to find a solution for an optimization model of a specific instance. The solver takes an instance of the model as input, applies the particular algorithm on a model and returns a result.

• *Modeling software* helps to formulate the optimization model and provide the solution. Modeling system takes the input as a description of the optimization problem in symbolic form and provides the solution in similar forms required by the algorithm(s) is done internally. To represent the problem in a symbolic form used the modeling language which is specific to the system or adapted from existing programming or scripting language. Most systems support a variety of solvers.

3.1.4.1 Gurobi Solver

According to [Opt18b], Gurobi is a type of solver software that helps to find the solution for optimization model. It supports all major programming languages for the problem modeling. By using Python API productivity can be earned quickly even if the user is currently familiar with another programming language.

3.1.4.1.1 Exploring modeling options in Gurobi - In [Opt18c], one can choose between two alternatives when building the optimization model: either using Gurobi with modeling language such as **A Mathematical Programming Language(AMPL)** or **General Algebraic Modeling Systems(GAMS)** or using Gurobi with programming languages such as C, C++, C#, Java, Python, VB, MATLAB or R. Modeling language will be good choice for non-programmers, with which model can be formulated easily. But programming language can be much more powerful and flexible choice if user interested in formulating the model and deploying that model and/or integrating into larger applications for others to use.

3.1.4.1.2 Python environment in Gurobi - They mentioned in [Opt18c], we can get benefits of modeling language and strength of programming language by using Gurobi Python environment. Their high-level optimization modeling constructs available in Python programming language, hence they eliminated the ambiguity to choose just modeling language or just programming language. The Gurobi Python environment provides a mature, full-featured and easy to use Python ecosystem language. We can save development time by taking the advantages of pre-written and tested packages of Python when creating new features for the program.

Benefits of using Gurobi Python Environment [Opt18c]

Flexible and powerful - We can use it for just prototyping or creating a full-featured optimization application.

Easy to use - With a basic understanding of language we can create simple models. With bit more knowledge we can create a complex application that is efficient.

Robust - Create the full application development by taking the advantages Python's pre-built packages.

Gurobi distribution comes with Python interpreter and the basic set of Python packages. For building and running simple optimization models these are sufficient. We can increase the development experience with Anaconda Python distribution that includes an integrated development environment(Spyder), a notebook-style interface(Jupyter) and 270 useful Python packages. These tools drastically increase the interactivity and productivity of python model and application development experience[Opt18c].

Here are the few things you can do with Python packages included with Anaconda distribution[Opt18c].

- Data extraction from database.
- Data analysis using powerful data analysis tools.
- Develop the Graphical User Interface(GUI) to capture the input and display the result.

3.1.4.2 Mathematical model

Create a mathematical model for a business problem. An LP or MIP model consists of three primary ingredients:

According to authors [Opt18a], the first step is to find a set of decision variables that capture the possible decisions(e.g, should i turn off the appliances or not?, should i send the truck from Stuttgart to Frankfurt, etc). Variables can be continuous(means take any value between specified lower and upper bound), or integral(means take only integral values). The model contains just continuous variables is called linear programming and model contains both integer and continuous variables is called mixed integer programming. The second step is to write the set of constraints, which put the restriction on values that decision variables can take(e.g., 0 for off, 1 for on status of an appliance, the truck should be sent to a single destination in a day, etc). Finally, write an objective function trying to capture value that can be minimized or maximized(e.g., Minimize CO_2 emission, minimize the distance, etc). objective function should be the linear expression for both LP or MILP.

We have explained the benefits of formulating the problem as an LP or MIP. The next step is to discuss how that is done. The fundamental building block of the MIP model is binary decision variables. It basically captures yes/no decision, where 1 means Yes and 0 means No, used to represent business decisions.

3.1.4.3 Model implementation

As explained in [Opt18a], after building the mathematical model, the next step is to implement the model. Gurobi provides interfaces for most of all programming languages like C, C++, Java, .NET, Python, R, and MATLAB.

3.1.4.4 Solving the model

According to authors [Opt18a], we have to declare our problem(the decision variables, constraints, and objective) and also find the optimal solution. The solver applies a sophisticated algorithm on our problem based on problem type[Opt18a]. For instance, Gurobi uses a branch-and-bound algorithm[Wik18a] to solve the MILP problem. But generally, we don't need to understand how the algorithm works to obtain a solution to our problem.

3.1.5 Appliance Scheduling Problem

Job shop scheduling is an optimization problem in computer science and operation research in which jobs are assigned to the machine at a particular time. The basic version is as follows: given n jobs $J_1, J_2, J_3, ..., J_n$ of varying processing times, which needs to schedule on m machines with varying processing power, the aim is to minimize the total time required by the machine to complete all the jobs [Wik18b]. Similarly, hybrid appliance scheduling is a problem in operation research in which appliances are scheduled to start working at a particular time. Let N be the number of appliances with varying electricity requirement and let T be the time slots in a day with varying

CO_2 emission factor. The aim is to start the appliance at a time slot where CO_2 emission is less and also choose the energy carrier whose CO_2 emission is less, based on these two factors we can minimize the total CO_2 emission from the appliances.

3.2 Android Application

Android is a Linux based operating system for mobile devices like tablet computers, smartphones and also a software package. Java is the mainly used programming language to write the code even though other languages can be used[jav18b].

3.2.1 History of Standard

• **Andy Rubin** founded Android Incorporation in the United States, 2003.
• Google acquired Android Incorporation on 17th August 2005.
• Google announces the development of Android OS in 2007.
• HTC launched the first Android mobile on 2008[jav18a].

The table3.1 gives information about android versions and their API level

Version	Code Name	API Level
1.5	cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 and 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15
4.1, 4.2, 4.3	Jelly Bean	16, 17, 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27

Table 3.1: Different versions of Android and API Level

Activates, views, intents, services, content providers and broadcast receivers are the fundamental components of android [Jav18].

3.2.2 Android Installation

Android supports for C++, C#, Java and Kotlin programming languages for development.
Required software to develop android application:
1. By ADT bundle.
2. By set up Eclipse IDE manually.
3. Android Studio IDE.

How to install Android Studio IDE: [Stu18]
1. Download .exe file, double click to launch.
2. Follow setup wizard and install android SDK.

3.3 Service Oriented Architecture(SOA)

According to [Bar18b], an SOA is a collection of services. These services can communicate with each other. Communication can involve simple data passing or two or more service coordination. Therefore, to connect services each other some means are needed.

3.3.1 Services

In [Bar18b], Service is nothing but a function that is well-defined, self-contained and does not depend on other services. We use web services to connect together the services and moreover, service is the endpoint of a connection.

3.3.2 Connections

In [Bar18b], web services technology is the connection technology of service-oriented architecture. The following figure**??** demonstrates the basic service-oriented architecture. Where service consumer sends the service request to the service provider. Service provider accepts the request and returns a response message to the service consumer. The request and response connections should be defined in the same way that is understandable to both service provider and consumer.

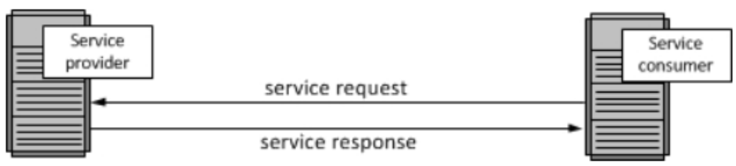

Figure 3.1: Basic Service-Oriented Architecture [Bar18b]

3.3.3 Web service

The term “web service“ is confusing. The term services have a different meaning than the term “web services“. Web service is a technology that allows for making the connection between services. Service act as an endpoint of the connection. In the following section, we will discuss the web service Representational State Transfer(REST) specification.

3.3.3.1 Representational State Transfer(REST)

In [Bar18a], REST is an architectural style provides a set of principles that tells how networked resources are defined and addressed. These principles were invented in 2000 by Roy Fielding as part of his doctoral dissertation.[1] REST is a style of software architecture, as a result, such applications or architecture are sometimes referred to as REST*ful* or REST-*style* applications or architectures. REST is one of the most popular choices for implementing web services.

An application or architecture is considered REST*ful* or REST-*style* if it possess the following characteristics:[Agg17]

• It has to be a stateless client and server model.

It means server should not store any application data that is required for the subsequent response.

• It has to be stateless client and sever model.

It means server should not store any application data that is required for the subsequent response.

• It has to be layered client cache stateless server model.

REST architecture will have a multi-level cache at the client and server side. This will help to improve network efficiency.

• Every REST architecture should have the following elements.

1. **Resource** - Every resource must have unique resource identifier.
2. **Representations** - Representation is the response of resource based on the media-type.
3. **Self Descriptive Messages** - Server should tell the client about resource representation by using content-type header whether it is JavaScriptObjectNotation(JSON) or Extensible Markup Language(XML) or some other format.
4. **Hypermedia as the engine** - It means, once client receives initial response, the response contains hyper links which allows client to move to next application state.

[1] https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

3.3.4 ENTSOE Transparency Platform Restful API

Before talking about components of the application, let us explain about the Transparency API. The data published on the ENTSOE central transparency platform in accordance with transparency regulation.

3.3.4.1 Request for information published on transparency platform

According to [ENT16], the central transparency platform allows data consumer to perform machine-to-machine queries for all structured data. The following figure3.2 shows the use case with involved actors in it.

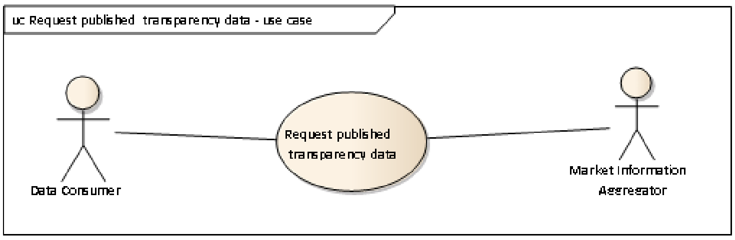

Figure 3.2: Request Use case[ENT16]

The following fig3.3 shows the sequence diagram of the exchange between data consumer and market information aggregator.

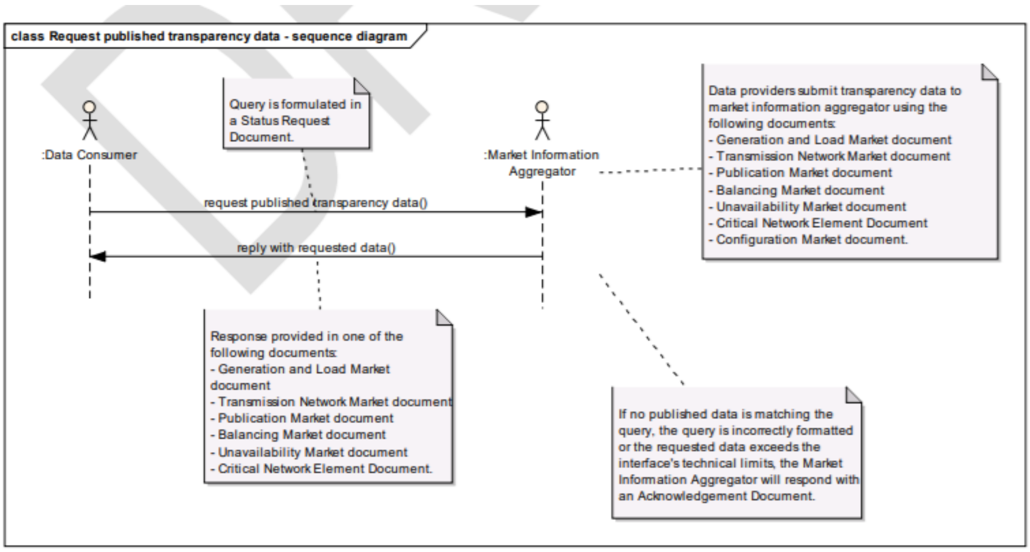

Figure 3.3: Request Sequence Diagram[ENT16]

•Data consumer sends the request query in status request document form that includes some selection criteria. Market Information Aggregator(MIA) responds to request by providing the document containing requested data. MIA is a type of communication interface between data consumer and data providers. That Data providers submit all necessary information to MIA, data consumer sends the required request to MIA and MIA respond with necessary information.

•If the request contains errors(for example, not all mandatory attributes included) or if no matching data available for selection criteria then Market Information Aggregator respond with negative acknowledgment document.

•In the same way, market information aggregator will respond with a negative acknowledgment document, if the returned data volume exceeds the technical limits imposed on the channel for data exchange.

3.3.4.2 Content of the request

As explained in [ENT16], a status request document contains a collection of key-value pairs. Each key-value pair captures the selection criteria for published transparency data. For instance, requests the data published under article 16.1.b&c(Aggregated Generation per Type) of Transparency regulation. The document contains the attributes like “DocumentType“ with the value “A75“, “ProcessType“ with value “A16“, and also In_Domain and TimeInterval.

The following example3.4 shows how a Status Request Document is used to extract data published under articles 15.1.a&b(planned unavailabilities and changes in actual availability of generation unit) for the National Grid control area during the month of April 2016. One of the General rules is attribute should not be used more than one time in the status request document.


```xml
<StatusRequest_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-5:statusrequestdocument:4:0">
  <mRID>SampleRequest</mRID>
  <type>A59</type>
  <sender_MarketParticipant.mRID codingScheme="A01">10V000000000008F</sender_MarketParticipant.mRID>
  <sender_MarketParticipant.marketRole.type>A07</sender_MarketParticipant.marketRole.type>
  <receiver_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</receiver_MarketParticipant.mRID>
  <receiver_MarketParticipant.marketRole.type>A32</receiver_MarketParticipant.marketRole.type>
  <createdDateTime>2016-05-27T13:00:00Z</createdDateTime>
  <AttributeInstanceComponent>
    <attribute>DocumentType</attribute>
    <attributeValue>A80</attributeValue>
  </AttributeInstanceComponent>
  <AttributeInstanceComponent>
    <attribute>BiddingZone_Domain</attribute>
    <attributeValue>10YGB----------A</attributeValue>
  </AttributeInstanceComponent>
  <AttributeInstanceComponent>
    <attribute>TimeInterval</attribute>
    <attributeValue>2016-03-31T23:00Z/2016-04-30T23:00Z</attributeValue>
  </AttributeInstanceComponent>
</StatusRequest_MarketDocument>
```


Figure 3.4: Example of Status Request Document[ENT16]

Disclaimer: Illustration purpose only.

For instance, if the consumer wants to query for article 6.1. an Actual Total Load data for both Belgian and French zones, two separate status document has to be submitted. In general, OutBiddingZone_Domain, In_Domain, Out_Domain, BiddingZone, Acquiring_Domain, Connecting_Domain and ControlArea are used to specify an area. The attributes contain Energy Identification Codes(EICs) of the area. Data consumer are allowed to query any area or border for which the central transparency platform publishes data. The order of attributes is not necessary inside the Status Request Document. Here we are using aggregated generation per type of article to get the necessary data from API. The table 3.2 gives information about mandatory and optional attributes required to send the request.

3.3.4.2.1 Aggregated Generation per Type Dependency table According to [ENT16], Dependency table provides information about which attributes must or may be used within Status Request Document in order to request the published data. The section below table3.2 explains what type of document the market information aggregator will return.

attribute	Art. 16.1.b&c Aggregated generation per type
DocumentType	A75:actual generation per type A74:wind and solar generation
ProcessType	A16:realised
In_Domain	Used
PsrType	May be used
TimeInterval	Used

Table 3.2: Dependency table for Aggregated Generation per Type

The market information aggregator will respond with Generation and Load Market document includes requested data.

3.3.4.3 Fundamentals for sending the request

3.3.4.3.1 Request Methods Here we have used the Http Get methods to query for desired data. The parameters are part of the URI string.
• Format of the URL: `https://transparency.entsoe.eu/api?`followed by Parameter Name = Parameter value, parameters are separated by '&'.

3.3.4.3.2 Request endpoints The Entso-e Transparency platform provides two endpoint: •Production Purpose - `https://transparency.entsoe.eu/api`. •Testing Purpose - `https://iop-transparency.entsoe.eu/api`.

We have used the Production purpose URL in our Application.

3.3.4.3.3 Parameters

As we already seen the dependency table3.2 for " Aggregated Generation per Type " that provides the applicable parameters to query for desired data. Order of the parameter is not significant but it is recommend to place parameters that identify the data first, followed by additional criteria and end with data range. Also, the name of the parameters is case sensitive.

Time should be always expressed in UTC.

There are two alternative ways to specify desired Time interval:

• Use parameter **TimeInterval**(in Get and Post methods).
- ISO format e.g. 2016-01-01T00:00Z/2016-01-02T00:00Z

• Use **periodStart** and **PeriodEnd** parameters(in Get methods only).
- Pattern yyyyMMddHHmm e.g. 201601010000

3.3.4.3.4 Rejection of Request

Request is rejected when one of the following conditions are met:

• If request does not contain all mandatory parameters.
• If request contains duplicate parameters.
• If parameters contain forbidden characters.
• If no data found.

The system will return the following document as shown in fig3.5 with the reason of rejection.


```xml
<?xml version="1.0" encoding="UTF-8"?>
<Acknowledgement_MarketDocument
    xmlns="urn:iec62325.351:tc57wg16:451-1:acknowledgementdocument:7:0">
    <mRID>be3917c7-0704-4</mRID>
    <createdDateTime>2016-03-30T10:45:36Z</createdDateTime>

    <sender_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</sender_MarketParticipant.mRID>
    <sender_MarketParticipant.marketRole.type>A32</sender_MarketParticipant.marketRole.type>

    <receiver_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</receiver_MarketParticipant.mRID>
    <receiver_MarketParticipant.marketRole.type>A39</receiver_MarketParticipant.marketRole.type>

    <received_MarketDocument.createdDateTime>2016-03-30T10:45:35Z</received_MarketDocument.createdDateTime>

    <Reason>
        <code>999</code>
        <text>No matching data found</text>
    </Reason>
</Acknowledgement_MarketDocument>
```


Figure 3.5: Acknowledgment Market Document with rejection reason [ENT14]

3.3.4.4 Authentication and Authorization

To access the Restful API, we have to register on the Transparency Platform [2]. Once access has been granted, users find their tokens on Transparency platform.

Token has to be sent along with request URL as a value to the **SecurityToken** Parameter key.

[2] https://transparency.entsoe.eu/usrm/user/createPublicUser

3.4 Firebase Database

Firebase is an easy way to store and retrieve data in a database. Firebase provides application developers an API that allows application data to be synchronized across clients and stored in firebase cloud. We can store all type of data in the firebase cloud. And it is free of cost up to some extent.

How to connect an android application to read from and write data to Firebase cloud.

We can connect an android application to firebase in two ways:
The First procedure as follows:
1. Open the android studio.
2. Go to Tools menu bar.
3. Select Firebase option.
4. It opens Assistant window, Click on Connect to Firebase button option.
5. Add the necessary dependencies and code to the application.

The Second procedure as follows:
1. Open Firebase console.
2. Add your project with the project name and package name.
3. Add your project SHA-1 key(Secure hash algorithm(SHA)). Example of SHA1 -> SHA1: 28:71:B2:CC:60:49:68:96:A0:4F:A7:71:A4:63:92:DB:47:A2:66:7F 4. Add the dependencies of Firebase.
5. Download the json file and add it in your application.
6. Ready to use the Firebase.

Following is the configuration in Android studio IDE [Fir18a]

```
buildscript {
  dependencies {
    classpath 'com.google.gms:google-services:4.2.0' // google-services plugin
  }
}

allprojects {

  repositories {
    google() // Google's Maven repository
  }
}

dependencies {
  implementation 'com.google.firebase:firebase-core:16.0.6'
}
```

Fetching data from Firebase Realtime database[Fir18b]:

```
private DatabaseReference mDatabase;
mDatabase = FirebaseDatabase.getInstance().getReference();

ValueEventListener postListener = new ValueEventListener() {
  @Override
  public void onDataChange(DataSnapshot dataSnapshot) {
    Post post = dataSnapshot.getValue(Post.class);
  }

  @Override
  public void onCancelled(DatabaseError databaseError) {
    Log.w(TAG, "loadPost:onCancelled", databaseError.toException());
  }
};
```

4 Proposal

4.1 Germany CO_2 Emission Android Application

Since CO_2 emission intensity information plays an import role in demand response incentive-based program. It encourages the electricity producing companies to produce electricity using renewable energy. Along with that the household heating and cooling are responsible for the consumption of approximately 21% of overall appliance consumption in US[CWF12] and household responsible for 29% of total electricity demand in Europe [HBF14]. Hence it encourages the consumers to save electricity from their side during the peak load hours.

Our Android application achieved the following functionality that helps the end users to reduce the CO_2 emission from household appliances:

1. Total CO_2 emission from Germany electricity grid for every 15 minutes time interval.
2. Total electricity generated per resource type in Germany for every 15 minutes time interval.
3. It shows the past 24 hours CO_2 emission information.
4. It shows the past 24 hours electricity generation mix.

To achieve the above functionality, we use the data available on the ENTSOE Transparency Platform restful API.[1] Here end users are people who own the household appliances and electricity producer in Germany. We will soon discuss how we actually implemented our application to achieve the desired functionality. Also, what are advantages and disadvantages in our application?

4.1.1 Architecture of Application

The abstract architecture of our application is as shown below,

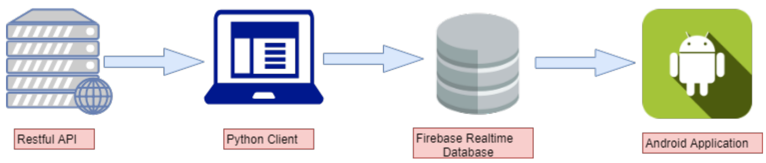

Figure 4.1: Abstract Application Architecture

[1] https://transparency.entsoe.eu/content/static_content/Static%20content/web%20api/Guide.html

Components of architecture:
1. **Restful API** - It represents the ENTSOE Transparency Platform restful API server.
2. **Python client** - It represents the Python client application.
3. **Firebase Realtime database** - It represents the Firebase cloud database.
4. **Android application** - It represents the CO_2 emission android application.

In chapter Implementation, we will understand in details about architecture components.

4.2 Scheduling of Hybrid Household Appliances

Our main objective of appliance scheduling problem is to minimize the CO_2 from households. We assumed each home has both traditional and hybrid appliances.

Traditional appliances - Traditional appliances are those that use a single energy carrier for their operation. It means, they use either electricity or gas or hot water for their working.

Hybrid appliances - It means, the appliance that can use at least two energy carriers alternatively for their operation. For instance, a washing machine can use both electricity and hot water for its operation.

In our model, we considered that appliances can be shifted in time and flexible enough to use energy carrier. It means that starting time and/or the energy carrier(s) are chosen according to CO_2- signals, with the aim of minimizing the emission.

4.2.1 Our proposed model

The following figure4.2 shows the proposed model.

The smart home has the following set of appliances and their energy carrier:

Traditional appliances
The smart home has the following set of appliances and their energy carrier:

Traditional appliances
1. Gas hobs - uses just gas energy carrier.
2. Refrigerator - uses just electricity energy carrier.

Hybrid appliances
1. Washing machine - uses just electricity or both electricity and hot water.
2. Dryer - uses just electricity or both electricity and hot water energy carrier.
3. Dishwasher - uses just electricity or both electricity and hot water energy carrier.

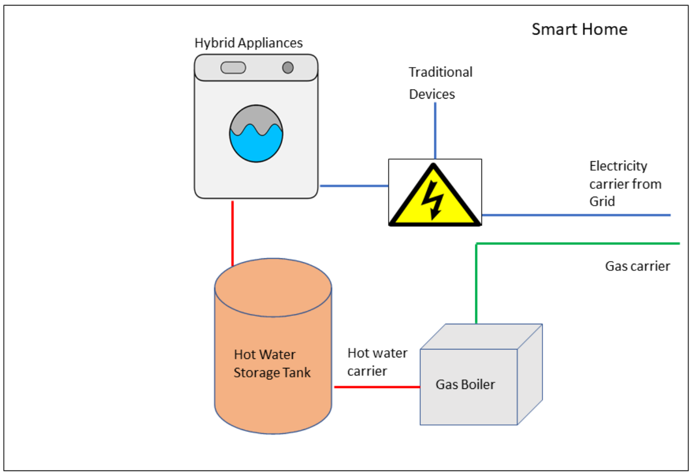

Figure 4.2: Proposed smart home

Others

1. Gas boiler - uses just gas for hot water production.
2. Hot water storage tank - stores the hot water produced by a gas boiler.

In the subsequent chapter, we will discuss in detail the characteristics of appliances, a scheduling problem, mathematical model, and implementation. In section 8 describe briefly about different use cases.

5 Requirement Specification

5.1 CO_2 Emission Android application

5.1.1 Hardware Requirements

The hardware requirements of our application are the following.
1. 2Ghz Quad core Intel i3 processor.
2. Microsoft Windows 7/8/10(32 or 64 bit).
3. 3GB RAM minimum, 8GB recommended.
4. 2GB of disk space minimum, 4 GB recommended.
5. 1280 * 800 minimum screen resolution.
6. Android mobile.
7. Android Emulator.

5.1.2 Software Requirements

1. Python 3.7.1 for Python client application.
2. Pycharm Integrated development environment.
3. Java Development Kit(JDK) 8 for developing Android application.
4. Android Studio Integrated development environment.
5. Android SDK(Software Development Kit).

5.2 Hybrid Appliance Scheduling

5.2.1 Hardware Requirements

The hardware requirements of our model implementation are the following.

1. 2Ghz Quad core Intel i3 processor.
2. Microsoft Windows 7/8/10(32 or 64 bit).
3. 3GB RAM minimum, 8GB recommended.
4. 2GB of disk space minimum, 4 GB recommended.

5.2.2 Software Requirements

1. Python 3.7.1.
2. Anaconda distribution version 4.5.11.
3. Gurobi solver version 8.0.1.
4. Jupyter notebook.

6 Implementation

6.1 CO_2 Emission Android Application

Our android application has three modules:

-Python client application
-Firebase database
-Android application

6.1.1 Python client application

The main objective is to calculate the Germany electricity grid CO_2 emission signal for every 15 minutes. Here we have used the ENTSO-E Transparency Platform restful API to get the required electricity generation information.

Endpoint : https://transparency.entsoe.eu/api

• Mandatory Parameters:
- DocumentType = A75
- processType = A16
- in_Domain = 10Y1001A1001A83F(Germany EIC code)
- PeriodStart = 201812040630 and PriodEnd = 201812040645

• Optinal Parameters:
- psrType:
1. B01 - Biomass
2. B02 - Fossil Brown coal
3. B03 - Fossil Coal Gas
4. B04 - Fossil Gas
5. B05 - Fossil Hard coal
6. B06 - Fossil oil
7. B09 - Geothermal
8. B10 - Hydro Storage
9. B11 - Hydro Poundage
10.B12 - Hydro Reserviour
11.B14 - Nuclear

12.B15 - Other Renewable
13.B16 - Solar
14.B17 - waste
15.B18 - Wind offshore
16.B19 - Wind Onshore
17.B20 - Other

Our Python application includes 3 modules:
•app.py module

This is the starting point of our application, where we have written the the logic to run the application for every 15 minutes.

• config.py

In this module, configure the request URL especially. Set the **PeriodStart** parameter to a difference of current UTC and one and half hours and **PeriodEnd** is equal to current UTC time. **PeriodStart& PeriodEnd** parameter value ceil to the nearest quarter hour. Also, set up the firebase database configuration like database URL.

• service.py

In this module, we sent separate Http Get request for each psrType to a specified endpoint including the mandatory parameters. Parse the response XML such that, query for the latest available data. Set the **PeriodStart** is equal to latest available time from response. Now, We have got the information about the quantity of electricity produced by each psrType, used the CO_2 emission intensity value for each psrTypes referenced from paper [SBN14] and calculate the CO_2 emission value.

• CO_2 emission intensity value for each psrType measured in gCO_2-eq/kWh:

B01 - Biomass - 18
B02 - Fossil Brown coal -1001
B03 - Fossil Coal Gas - 469
B04 - Fossil Gas - 469
B05 - Fossil Hard coal - 1001
B06 - Fossil oil - 840
B09 - Geothermal - 45
B10 - Hydro Storage - 5
B11 - Hydro Poundage - 5
B12 - Hydro Reserviour - 5
B14 - Nuclear - 16
B15 - Other Renewable - 700
B16 - Solar - 46
B17 - waste - 700
B18 - Wind offshore - 12
B19 - Wind Onshore - 12
B20 - Other - 700

Formula to calculate the CO_2 emission intensity referenced from paper [SXZ14]:

Total CO_2 intensity = $\sum_{i \in psrType}(Electrcity_i * CO_2 \quad emission \quad value_i)/(Total \quad psrType \quad electricity)$

The example URL: https://transparency.entsoe.eu/api?securityToken=17769ff1-d271-458a-b6be-4faed7c304e5&documentType=A75&processType=A16&in_Domain=10Y1001A1001A83F&psrType=B20&periodStart=201812040515&periodEnd=201812040645

The example response shown in below figure6.1:


```xml
<GL_MarketDocument xmlns="urn:iec62325.351:tc57wg16:451-6:generationloaddocument:3:0">
  <mRID>5e5f276378024457b5729e871affc83a</mRID>
  <revisionNumber>1</revisionNumber>
  <type>A75</type>
  <process.processType>A16</process.processType>
  <sender_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</sender_MarketParticipant.mRID>
  <sender_MarketParticipant.marketRole.type>A32</sender_MarketParticipant.marketRole.type>
  <receiver_MarketParticipant.mRID codingScheme="A01">10X1001A1001A450</receiver_MarketParticipant.mRID>
  <receiver_MarketParticipant.marketRole.type>A33</receiver_MarketParticipant.marketRole.type>
  <createdDateTime>2018-12-04T06:36:51Z</createdDateTime>
  <time_Period.timeInterval>
    <start>2018-12-04T05:15Z</start>
    <end>2018-12-04T05:45Z</end>
  </time_Period.timeInterval>
  <TimeSeries>
    <mRID>1</mRID>
    <businessType>A01</businessType>
    <objectAggregation>A08</objectAggregation>
    <inBiddingZone_Domain.mRID codingScheme="A01">10Y1001A1001A83F</inBiddingZone_Domain.mRID>
    <quantity_Measure_Unit.name>MAW</quantity_Measure_Unit.name>
    <curveType>A01</curveType>
    <MktPSRType>
      <psrType>B20</psrType>
    </MktPSRType>
    <Period>
      <timeInterval>
        <start>2018-12-04T05:15Z</start>
        <end>2018-12-04T05:45Z</end>
      </timeInterval>
      <resolution>PT15M</resolution>
      <Point>
        <position>1</position>
        <quantity>418</quantity>
      </Point>
      <Point>
        <position>2</position>
        <quantity>418</quantity>
      </Point>
    </Period>
  </TimeSeries>
</GL_MarketDocument>
```


Figure 6.1: XML response for psrType “Other“

Sample code can be find in the Appendix A

6.1.2 Android Application

Android application is the simplest way to represent data to the user in an easily understandable manner.

• As soon as user open android application, showing the CO_2 emission information in gCO_2/MW and total electricity generated value in MW.
• Also in the front page, we have shown the latest available electricity generation information for each resource in list view.
• We have created a button at the bottom of the front page which navigate to the next page.
• In next page, we have shown 24 hours CO_2 emission information in the line graph and Electricity generated information for each resources types in a stacked bar chart. for example. 05-12-2018 06:00 AM to 06-12-2018 06:00 AM(current time), from current time to past 24 hours.
• Firebase is an easy way of synchronizing and fetch data from a real-time database. It works like API.

• To integrate firebase, Go to File – Project Structure – cloud –check firebase check-box, click ok. This will add the project to firebase.

6.1.2.1 Graphical view Information

• **Electricity generated Information** (Stacked Bar Graph)

X-axis Time stamp.
Y-axis Electricity in MW.

• CO_2 **emission information** (Line Graph)

X-axis Time stamp.
Y-axis – CO_2 emission in $gC0_2/MW$.

6.1.2.2 Screenshots of Android application

The following figures 6.2, 6.3, 6.4, 6.5 and 6.6 gives some idea about our android application.

Figure 6.2: The Initial page gives information about CO_2 emission value, generation mix electricity production and their emission factor at particular time interval

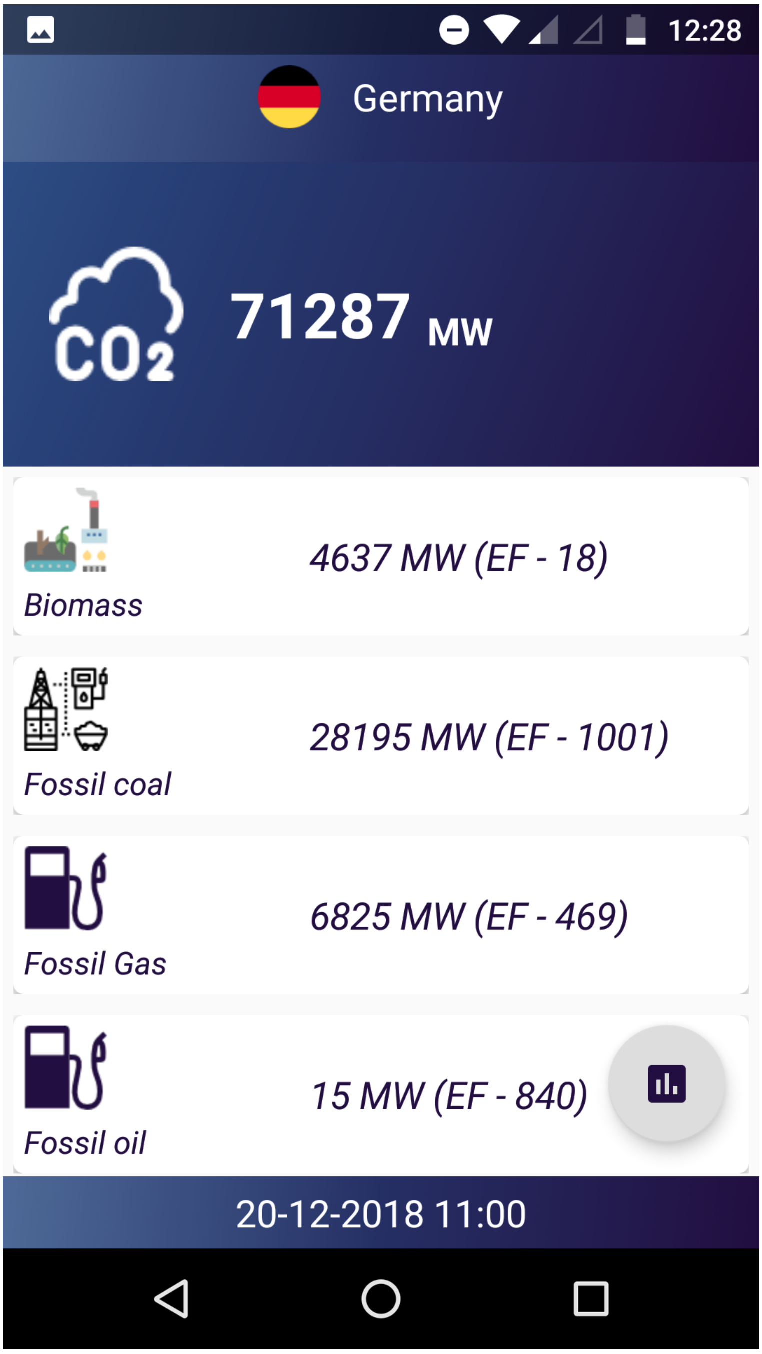

Figure 6.3: The Initial page also gives information about total electricity produced in Germany at particular time interval

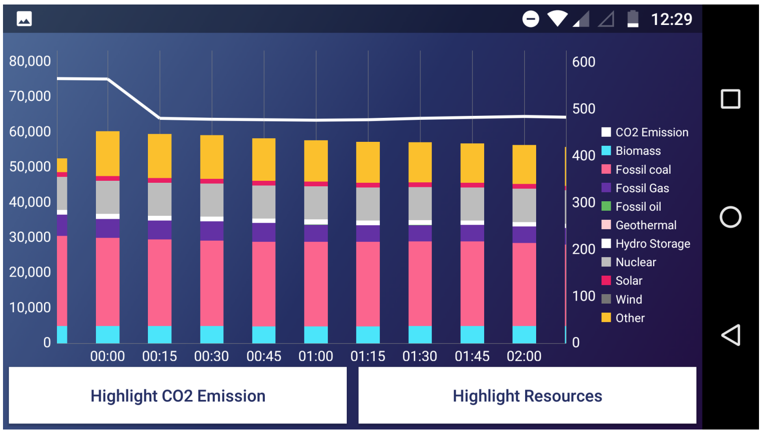

Figure 6.4: The Second page gives 24 hours CO_2 emission information and generation mix production.

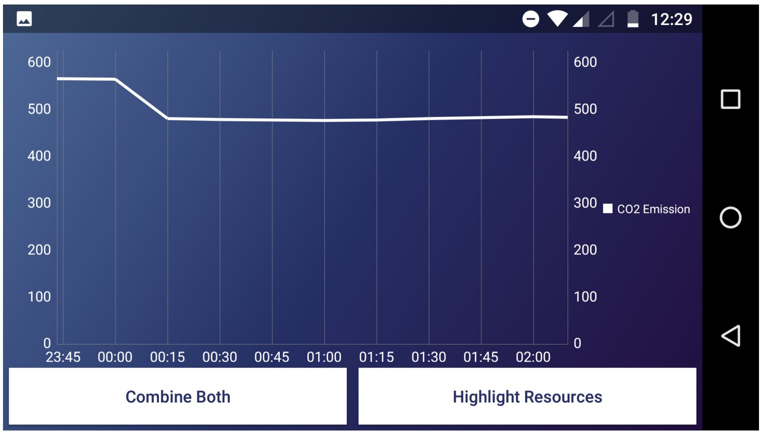

Figure 6.5: In Second page, we can see only 24 hours CO_2 emission information by pressing “Highlight CO_2 Emission“ Button.

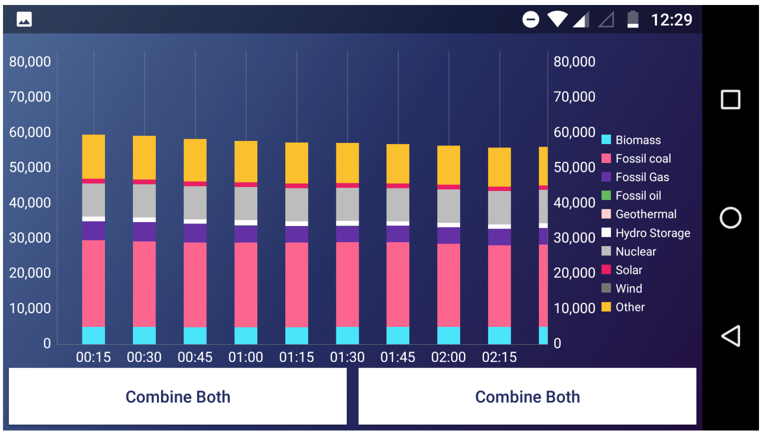

Figure 6.6: In Second page, we can see only 24 hours generation mix information by pressing “Highlight Resource“ Button.

Link to download the android application https://drive.google.com/file/d/19toR9jwOb8lmqzI06AW-w2bOihXgPMmL/view?usp=drivesdk
Sample code can be find in the Appendix B

6.2 Hybrid Appliances Scheduling

Our objective is to schedule the appliances to time slot, such that it reduces the overall household appliance CO_2 emission. Scheduled based on the appliances can be shifted in time based on CO_2 emission value and use the energy carrier according to their emission intensity value at that time slot.

6.2.1 Home Appliances Description

We are considered a household with the following appliances: dryer, dishwasher, washing machine, refrigerator, gas hob1, gas hob2, and heater. Each appliance has definite completion time and has definite power consumption vector. The power consumption measurement is taken from the paper[SBP+08].

1). **Dishwasher**
The dishwasher has 8 energy phases: These are movement, heating1, wash, 1strinse, drain, heating2, 2ndrinse, drain_and_dry and each energy phase takes 15minutes to complete the phase. The dishwasher acts as a hybrid appliance, it means, uses both electricity and hot water. Electricity consumption varies from maximum 500Wh to minimum 20Wh. If it acts as a hybrid, then electricity varies from maximum 250Wh to 20Wh and hot water energy varies from 250Wh to 17.5Wh[SBP+08].

2). **Dryer**
The dryer is divided into 8 energy phases. Each energy phase takes 15 minutes to complete the operation. Dryer act as a hybrid appliance, it means, uses both electricity and hot water. Electricity consumption varies from maximum 500Wh to 235Wh. If it acts as a hybrid, then electricity consumption will 50Wh for all energy phases and hot water consumption will vary from 50Wh to 450Wh for all energy phases[SBP+08].

3). **Washing Machine**
The washing machine has 7 energy phases: These are movement, heating, washing, cooling, 1strinse, 2ndrinse, 3rdrinse, and each energy phase takes 15 minutes to complete the phase. The washing machine acts as a hybrid appliance, it means uses both electricity and hot water. Electricity consumption varies from maximum 500Wh to minimum 12.5Wh. If it acts as a hybrid, then electricity consumption varies from maximum 362.5Wh to 12.5Wh and hot water consumption varies from 200Wh to 137.5Wh[SBP+08].

4). **Refrigerator**
The refrigerator is divided into 96 energy phase and each takes 15 minutes time to complete each phase. It uses just electricity to run the operation and electricity consumption varies from 0.25Wh to 34.5Wh[SBP+08].

5). **Gas Hob**
We divided the gas hob into 2 appliances based on time of operation, it means we assumed gas hob operating in the morning named as “Gas Hob1“ and it runs approximately 1 hour so has 4 energy phases each takes 15 minutes time. The gas hob operating in the evening named as “Gas Hob2“ and runs approximately 1 and a half hour so has 6 energy phases and each takes 15 minutes time to complete. Assumed that gas hob uses all the 4 gas plate. The main energy source will be gas, it’s consumption will always be 225Wh[ene18].

6.2.2 Operation Duration

We divided the 24 hours into 96-time slots and each time slot represents 15 minutes time interval. The starting time slot and ending time slot refers to 6:15AM and 6:00AM, respectively [QNK+15]. User-specified starting time and ending time for each appliance according to their need. For instance, if the user arrived at home evening 5 PM and wants to end dinner within 8.30PM, then gas hob has to schedule within 5:15 PM to 7:45 PM. According to our model, each appliance is executed only once in day. The scheduler is free to switch on any time slot if it is under the specified time range and can switch between energy carrier if the appliance has hybrid operation phase[QNK+15].

6.2.3 Mathematical Modeling

We formulate the appliance scheduling using mixed integer linear programming(MILP) technique for which binary variables are defined and applied. We considered a set of homes denoted by U, each home has a set of appliances I, each appliance has a set of uninterpretable energy phases denoted by J_i for i = 1, 2, 3, . . . , N[QNK+15].

Nomenclature

Sets

K set of time slots in a day $\{1,2,3, \ldots 96\}$

U set of homes considered $\{1, 2,3, 4, \ldots 30\}$

I set of appliances in each home

J_i set of energy levels for each appliances

Indexes

t index of the time slot

i index of the appliances i

j index of the energy level of the appliances j

u index of the home

Variables

H_a^k available hot water in the storage tank at time slot k(Wh)

T_t^{Boiler} Thermal output of the Boiler at time slot k(Wh)

H_k^{Boiler} Hot water produced by the Boiler at time slot k(Wh)

G_k^{Boiler} Amount of gas required by Boiler at time slot k (Wh)

Parameters

$C\_t^{electricity}$ CO_2 emission from electricity at time slot k (gco_2/Wh)

$E_{u,i,j}^{high}$ the task j of appliance i of user u consumes high electricity from the grid(Wh)

$E_{u,i,j}^{low}$ the task j of appliance i of user u consumes low electricity from grid(Wh)

C_t^{gas} CO_2 emission from the natural gas at time slot k(gco_2/Wh)

$G_{u,i,j}^{demand}$ amount of gas consumed by task j of appliance i at time slot k(Wh)

$H_{u,i,j}^{demand}$ amount of hot water consumer by task j of appliance i at time slot k(Wh)

$t_{st,i}$ starting time slot selected by user for appliance i

$t_{end,i}$ ending time slot selected by user for appliance i

r_i run time of the appliance i

C_{ST} Hotwater storage tank capacity(Wh)

$PEAK$ maximum capacity of the power distribution grid(Wh)

T_{max}^{Boiler} maximum thermal output of the Boiler.

η_{th} Boiler efficiency

Decision Variable

$x_{k,u,i,j}$ indicates whether task j of appliance i of user u at time slot k processed by only electricity or not; 1 = task processed; 0 = not processed

$y_{k,u,i,j}$ indicates whether task j of appliance i of user at time slot k processed by both electricity and hotwater or not; 1 = task processed; 0 = not processed

$z_{k,u,i,j}$ indicates whether task j of appliance i of user u at time slot k processed by natural gas or not; 1 = task processed; 0 = not processed

s_k^{Boiler} On and Off status of the Boiler at time slot t; 1 = On; 0 = Off.

6.2.3.1 Objective Function

The objective of the scheduling of appliances is to minimize the CO_2 emission of the houses based on the 24-hours emission data. The total CO_2 emission function C(k) is given by the following equation:

$$minimize \sum_{t=1}^{96}\sum_{u=1}^{50}\sum_{i=1}^{N}\sum_{j=1}^{n_i} C_t^{electricity} E_{u,i,j}^{high} x_{t,u,i,j} + [C_t^{electricity} E_{u,i,j}^{low} + C_t^{gas} H_{u,i,j}] y_{t,u,i,j} + C_t^{gas}[G_{u,i,j} z_{t,u,i,j} + G_t^{Boiler}] \quad (6.1)$$

6.2.3.2 Constraints

Subject to following constraints:

The appliances can use electricity or both electricity and hot water or Gas energy carrier at a time to process its task.

$$x_{t,u,i,j} + y_{t,u,i,j} + z_{t,u,i,j} \leq 1 \forall t,u,i,j, \{X_{t,u,i,j}, Y_{t,u,i,j}, Z_{t,u,i,j}\} \in \{0,1\} \tag{6.2}$$

The amount of electricity demand at any time slot should not exceed the $PEAK$ electricity threshold of the distribution grid[QNK+15]. Here $PEAK$ is the maximum electricity that a consumer can demand at any slot, it is provided the electricity distributor based on the population of the area.

$$\sum_{u=1}^{50}\sum_{i=1}^{N}\sum_{j=1}^{n_i} E_{u,i,j}^{high} * x_{t,u,i,j} + E_{u,i,j}^{low} * y_{t,u,i,j} \leq PEAK \qquad \forall t, x_{t,u,i,j}, y_{t,u,i,j} \in \{0,1\} \tag{6.3}$$

The amount of hot water demand by all tasks of the appliances at any time slot should not exceed the capacity of the hot water storage tank.

$$\sum_{i=1}^{N}\sum_{j=1}^{n_i} H_{u,i,j} * y_{tu,i,j} \leq C_{ST} \qquad \forall t,u, y_{tu,i,j} \in \{0,1\} \tag{6.4}$$

The amount of gas required by boiler to produce thermal power in Wh [BHJ15].

$$G_t^{Boiler} = \frac{T_t^{Boiler}}{\eta_{th}} \tag{6.5}$$

Convert the thermal power produced by the boiler into the hot water. For instance, the water temperature rises from 0 degrees to 65 degrees. Hence the *Temp* is 65 degree. Where 4 is a 1/4th of an hour and 3412 being a given constant(1kWH = 1 British Thermal Unit) and 2.22 is also constant(1litre = 2.22 BTU). This formula tells the amount of hot water produced by thermal energy of the boiler in 15-minute [ele18].

$$H_t^{Boiler} = \frac{T_t^{Boiler} * 3412}{2.22 * 65 * 4} \tag{6.6}$$

The amount of thermal energy produced by boiler should not exceed its maximum thermal output at time slot t [BHJ15].

$$0 <= T_{t,u}^{Boiler} <= s_t^{Boiler} . T_{max}^{Boiler} \quad \forall t = 1,2,3,4 \tag{6.7}$$

The amount of hot water available in the storage tank at time slot ‘t‘ is equal to the amount of hot water stored at ‘t-1‘ plus the hot water produced by the boiler at time slot ‘t‘ minus hot water demand by the appliances at time slot ‘t‘.

$$H_t^a = H_{t-1}^a + H_t^{Boiler} - \sum_{i=1}^{N}\sum_{j=1}^{n_i} H_{u,i,j} * y_{t,u,i,j} \forall u, t = 2, 3, .. \tag{6.8}$$

Hot water storage tank capacity has to be maintained at any time slot.

$$0 \leq H_t^a \leq C_{ST} \qquad \forall t = \{1, 2, ...96\} \tag{6.9}$$

The task of the appliances should not be performed outside the user-specified time window.

$$\sum_{t \in m-[t_{st,i}, t_{end,i}]} X_{t,u,i,j} + Y_{t,u,i,j} + Z_{t,u,i,j} = 0 \quad \forall i, j, u \tag{6.10}$$

The task of the appliances are not interruptible it means can’t stop until it finishes its task.

$$\begin{aligned} x_{t,u,i,j} &= x_{t-1,u,i,j-1} \\ y_{t,u,i,j} &= y_{t-1,u,i,j-1} \\ z_{t,u,i,j} &= z_{t-1,u,i,j-1} \qquad \forall u, i, j = 2, 3, ..., t = 2, 3, 4...96 \end{aligned} \tag{6.11}$$

The tasks of the refrigerator has to be sequential.

The refrigerator should starts from first task at the first time slot.

$$\begin{aligned} x_{t_0,u,i,j_0} = 1 \quad t_0 &= \text{first time slot of refrigerator} \\ \forall u \quad and \quad i = \text{refrigerator}, j_0 &= \text{1st task of refrigerator} \end{aligned} \tag{6.12}$$

The dryer should start to run after the washing machine completed its task.

$$\begin{aligned} x_{t+1,u,dry,j_0} &<= x_{t,u,wash,j_{last}} \\ y_{t+1,u,dry,j_0} &<= y_{t,u,wash,j_{last}} \\ z_{t+1,u,dry,j_0} &<= z_{t,u,wash,j_{last}} \end{aligned} \tag{6.13}$$

Runtime of the appliance has to be satisfied.

$$\sum_{t}^{T}\sum_{j}^{J} x_{t,u,i,j} + y_{t,u,i,j} + z_{t,u,i,j} == r_i \quad \forall u, i \tag{6.14}$$

All tasks of the Washing machine, dryer and dishwasher has to use either electricity or electricity and hot water but not gas.

$$z_{t,u,i,j} = 0 \quad \forall t, u, i = \text{washingmachine, dryer, dishwasher}, j \tag{6.15}$$

All tasks of the gashob1 and gashob2 can use only gas energy carrier.

$$x_{t,u,gashob1,j} + y_{t,u,i,j} = 0 \quad \forall t, u, i = \text{gashob1, gashob2}, j \tag{6.16}$$

The first task of the appliance has to be before end time of the appliance minus run time of the appliance.

$$\sum_{t_{st}}^{t_{end}-r_i+1} x_{t,u,i,j_0} + y_{t,u,i,j_0} + z_{t,u,i,j_0} = 1 \tag{6.17}$$

Sample code can be find in the Appendix C

7 Simulations

We run our MILP program code to schedule the hybrid appliances for minimizing the CO_2 emission intensity value for the following use cases:

A. A user with fixed time range and energy carrier.-The users who belong to this case does not care about the environment. They want their appliances to run in the fixed time range(= run time of the appliances) without giving room for the scheduler to run the appliance on the other time slot and also impose a constraint on the appliance to operate on the standard mode(acts as traditional appliances).

B. Users with a fully flexible time range and energy carrier - In this case, users are flexible enough to assign the appliances at any time slot range between 1 to 96 and each time slot represents 15min time interval and flexible enough to use alternative energy carrier(appliance acts as a hybrid).

C. A user with a mixed time range and energy carrier.-In this case, users are not fully flexible about time range but provided time range that is greater than or equal to the runtime of the appliances. And allow the appliance to operate on hybrid mode(two energy carrier alternatively).

D. Days with non-renewable and renewable energy generation.-In this case, we considered two days of the year where a day at which renewable energy(e.g., wind, solar) generation is more, whereas another day at which non-renewable energy(e.g., fossil fuels, coal) generation is more.

In this section, we have simulated our model for a group of 30 users and each user has an intelligent hybrid and traditional appliances. For all users, there is a single gas boiler that produce hot water and hot water storage tank that stores hot water. The appliances of each user directly connect to the hot water storage tank. The specification of appliances used in our model are listed in table 7.1.

Number of users	30
Appliances	Washing Machine, Dryer, Dishwasher, Gas Hob, Refrigerator
Hybrid Appliances	Washing Machine, Dryer, Dishwasher
Hot water gas boiler	21.25kW thermal capacity
Hot Water Storage Tank	750 liters, Min.Temp: 57°, Max.Temp: 78°

Table 7.1: Specification of the simulated households

The CO_2 emissions per Wh vary for different times of the day, to reduce the peak load demand and make the horizontal distribution of load. However, maximum peak load demand of the building is controlled by the constraint that is the sum of loads at any time is less than or equal to 20625

Wh(assumed) in our case. This maximum peak load is always assigned by the electricity distributors based on the population of the region. Our main is to reduce the CO_2 emission of the building. Hence, the effect on CO_2 emission will be studied for the appliances specified in table 7.1 [QNK+15].

We discuss each use case in the following pattern. First, we present the scenario, followed by the optimization results. It includes Total Energy Consumption(TEC), Total CO_2 Emission(TCE) and maximum peak load and it's associated time(MPLAT). Where TEC represents the aggregated energy consumption of thirty users in a whole day, it involves both hybrid and traditional appliances consumption. The TCE presents the thirty users total CO_2 emission in a whole day, it involves emission by using electricity, gas and hot water. The MPLAT gives information about the time slot at which maximum load demand occurred. In almost all cases, We got results in less than fifteen seconds using a computer with Intel i5 processor with 8GB RAM. Processing will be faster if we used a computer with higher computational capability [QNK+15].

8 Results and Discussion

8.0.1 User with fixed time range and energy carrier

We have considered the scenario, where the consumer wants to use the appliances in a fixed time and run in standard mode without concerned about the CO_2 emission affecting the environment. Such a situation is described as the worst case. In this scenario, considered the building where appliances operate on the fixed time range that is equal to the runtime of the appliances. Also, appliances operate on the standard mode, it means they act as traditional appliances. The time range and mode of the appliances are given in the table 8.1. Here we imposed the constraint on the appliances to begin from the start time slot, finish at the end time provided and operate in a standard mode. As there is no room for the appliances operating on the time other than the time given by the user. Therefore, starting and ending time of the appliances obtained using the proposed optimization algorithm is the same as the table 8.1.

1	Appliance	Start Time	End Time	Mode
2	Washing Machine	44(5:00PM)	50(6:45PM)	Standard
3	Dryer	51(7:00PM)	58(9:00PM)	Standard
4	Dishwasher	64(10:00PM)	71(12:00AM)	Standard
5	Refrigerator	1(6:15AM)	96(6:00AM)	Standard
6	gashob1(Morning)	4(7:00AM)	7(8:00AM)	Standard
7	gashob2(Evening)	52(7:00AM)	57(8:30AM)	Standard

Table 8.1: Appliance with fixed schedule (Time range = Run time of appliance) and standard mode

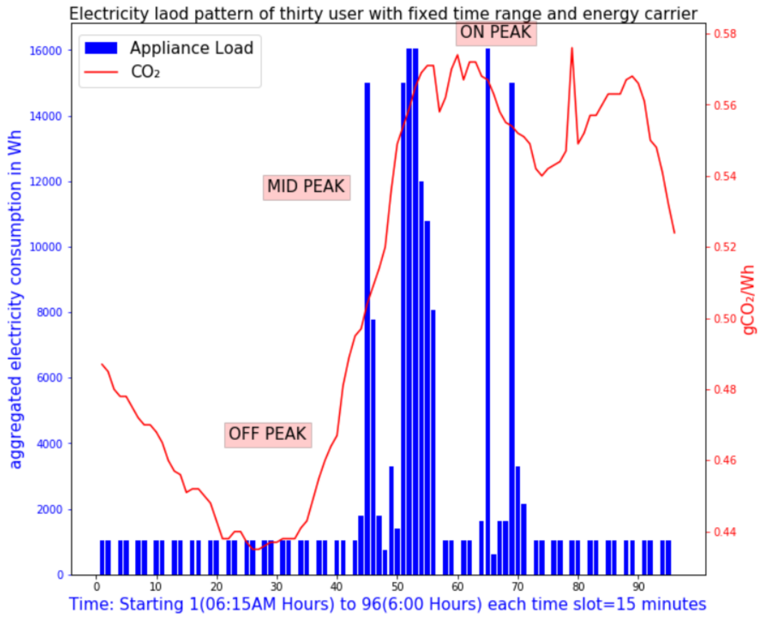

Figure 8.1: Electricity Load pattern of appliances operating in fixed time range and energy carrier

The above figure8.1 shows the demand curve of 30 users appliances. The labels in figure "ON PEAK", "OFF PEAK", and "MID PEAK" referred to maximum, minimum, and medium CO_2 emission of the day respectively. We can see that most of the appliances are scheduled in the region of mid-peak and on-peak CO_2 emission, therefore scheduling is not done with respect to CO_2 emission value. The peak load demand of 12.03 kW occurs at two-time slots 53(7:15 PM) and 65(10:15 PM), those are in the mid-peak and on-peak CO_2 emission respectively. We considered CO_2 emission data between the days 10/07/2018 and 11/7/2018. Other parameters are given in table 8.2 and 8.16.

Parameters	Case A
TEC(kWh/day)	202.68
TCE(gCO_2/day)	121181
MPLAT(kWh@slot)	12.03@53 and 65

Table 8.2: Optimization results for thirty users with fixed time range and energy carrier

Since we considered the users with fixed time range and energy carrier, the scheduler has no chance to run the appliance in other time slot and use alternative energy carrier. The scheduler runs all the appliances using the standard energy carrier e.g., a washing machine uses just electricity, even though it can use both electricity and hot water. Because we imposed the constraint to use just electricity. As we can see in the table8.2, the TEC of all thirty users appliances is 202.68KWh and TCE is 121181 gCO_2 for the 24 hours. We keep the result obtained from this case as reference.

8.0.2 Users with full flexible time range and energy carrier

In this case, users are flexible enough to assign appliance at any time slot range between 1 to 96. Running the optimization algorithm for minimizing the CO_2 emission with appliances time range available from time slot 1(6:15 AM) to time slot 96(6:00 AM) and use alternative energy carrier. From the result, we came to know that some appliances of some users assign to the inconvenient starting time slot. The appliances like gas hob of some users will start operating at time slot 16(10:00 AM), this is too late for the users. Because user wants to run the gas hob from 7:00 AM to 8:30 AM. The dishwasher starts operating at time slot 25(12:15 PM), this is not at all good. Because usually dishwasher scheduled to operate after dinner time. We can see the below figure 8.2, most of the appliances of some users start operating at time slots where the emission of CO_2 is less. Even though, this use case reduces the CO_2 emission, it is inconvenient and not adoptable by all users. Other parameters are given in table 8.16 and 8.3.

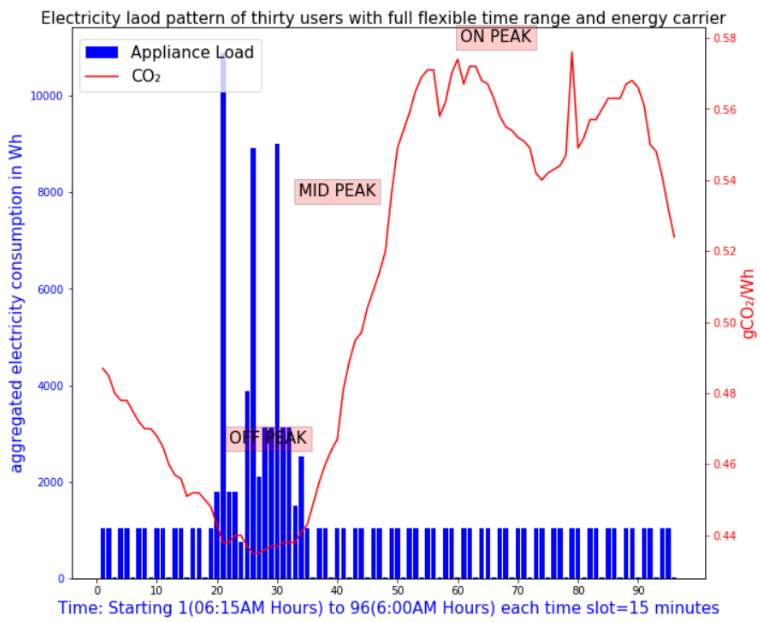

Figure 8.2: Electricity Load pattern of appliances operating in flexible time range and energy carrier

Parameters	Case B
TEC(kWh/day)	113.58
TCE(gCO_2/day)	83722.1
MPLAT(kWh@slot)	10.88@20

Table 8.3: Optimization results for thirty users with full flexible in time range and energy carrier

Both TEC and TCE values are almost 44% and 30% less than the values of Case A as shown in table8.4. Due to the fact that appliance allowed to use alternative energy carrier when the emission from grid electricity more and assign any time slot. For instance., the washing machine can use both electricity and hot water for their operation, instead of using just electricity. Therefore, total electricity consumed by washing machine during hybrid mode is 550Wh, whereas in standard mode 950Wh. Since hot water produced by boiler uses gas energy carrier, whose emission is way less than the emission from grid electricity according to our CO_2 emission of the day 10/07/2018 to11/7/2018.

Parameters	Case A	Case B
TEC(kWh/day)	202.68	113.58
TCE(gCO_2/day)	121181	83722.1
MPLAT(kWh@slot)	12.03@53 and 65	10.88@20

Table 8.4: Compare the results of use case with full flexible time range and energy carrier with use case of fixed time range and energy carrier

8.0.3 User with mixed time range and energy carrier

•In this scenario, we have considered the mixed time range preference as shown in table 8.5 provided by the user as per their convenience. And used the CO_2 emission data between the day 2018-01-05 and 2018-01-06.

•We followed the pattern of increasing the user group by step of 10% every time.

•Initially, we considered just 10% of users with mixed time range, remaining 90% of users have fixed time range and keep 100% users appliance operation in standard mode.

•Then, we considered just 20% of users with mixed time range, remaining 80% of users have fixed time range and keep 100% users appliance operation in standard mode as constant(it does not change). Likewise, we increased the group of mixed time range user 10% every step, simulate our model accordingly and discuss the result.

• Vice versa, first We considered 10% of users appliances allowed to use alternative energy carrier(acts as hybrid appliances), remaining 90% of users operate on standard mode and keep 100% users to fixed time range as constant. Likewise, we increased the group of hybrid mode user 10% every step, simulate our model accordingly and discuss the result.

• In last, we run the simulation by increasing both users who allowed to use mixed time range and alternative energy carrier to 10% in every step. We simulate our model accordingly and discussed the result of each case below.

The mixed time range means time range provided by the user is either equal to or greater than the runtime of the appliances, that gives flexibility to appliance scheduling and more time slot options for allocating appliances[QNK+15]. But only for refrigerator time range is equal to the runtime of the appliance. As we know the energy demand of appliances is constant, reduction of CO_2 emission possible by shifting the load away from the peak CO_2 emission value and using alternative energy carrier. We imposed the constraint to keep the peak load less than or equal to 20625Wh at any instant of the time.

we have considered three separate cases as discussed above:

1	Appliance	Start Time	End Time	Mode
2	Washing Machine	36(3:00PM)	64(10:00PM)	Hybrid
3	Dryer	36(3:00PM)	64(10:00PM)	Hybrid
4	Dishwasher	66(10:30PM)	96(06:00AM)	Hybrid
5	Refrigerator	1(6:15AM)	96(6:00AM)	Hybrid
6	gashob1(Morning)	4(7:00AM)	10(8:30AM)	Hybrid
7	gashob2(Evening)	52(7:00PM)	64(10:00PM)	Hybrid

Table 8.5: Mixed time range of users

C1. The user with a mixed time range, but a standard energy carrier. • First, we run our MILP program code by increasing 10% of the user with a mixed time range, keep other 90% of users with a fixed time range and 100% of users appliances operate on the standard mode that means they are not allowed to use alternative energy carrier.

• We simulate our model for 10% user group who allowed mixed time range in every step. But we don't get much efficient optimization result, due to only 10% user operate on mixed time range which is too less. And, 90% of users preferred fixed time range and no user allowed to operate on hybrid mode.

• Even we used 50% of the user with a mixed time range, but there is only 2.57% decrease in emission value compare to Case A(fixed time range and energy carrier) as shown in 8.7.

•If we compare the result in table8.6 where 100% of users allowed to use mixed time range and but 0% of users operate in hybrid mode. With the use case where 100% of users possess fixed time range and energy carrier(standard mode) Case A results shown in table8.7. There is just 5% decrement in CO_2 emission reduction. The total electricity consumed remains same due to using standard mode of appliances, the CO_2 emission value reduced because of 100% of a user operating in a mixed time range.

• Hence, it is clearly stated that, even after allowing the users with a mixed time range and operate appliance in standard mode, there is not much improvement in optimization result. The following figure8.3 shows that, most of the appliances operating on mid peak CO_2 emission.

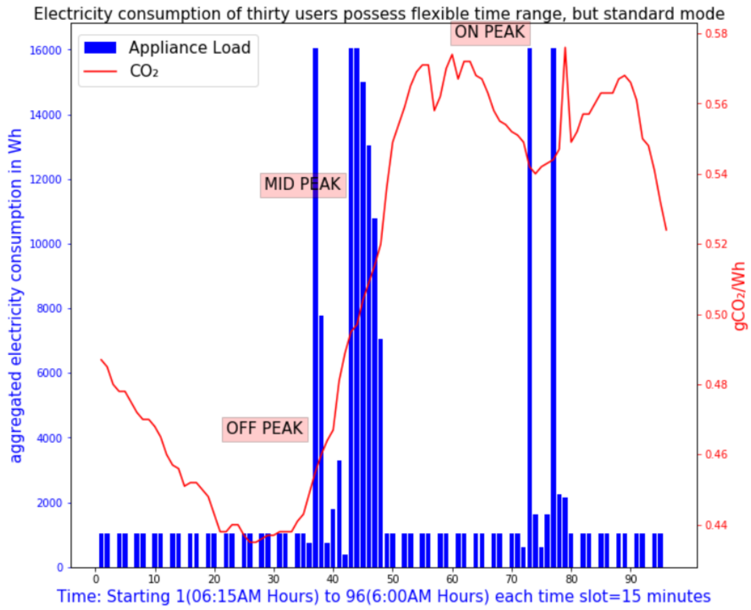

Figure 8.3: Electricity Load pattern of appliances operating in mixed time range and but standard energy carrier and

Parameters	Case C1
TEC(kWh/day)	202.68
TCE(gCO_2/day)	114940
MPLAT(kWh@slot)	16.03@36

Table 8.6: Optimization results for thirty users with mixed time range and standard energy carrier

Parameters	Case A	Case C1(10% mixed time range)	Case C1(50% mixed time range)	Case C1(100% mixed time range)
TEC(kWh/day)	202.68	202.68	202.68	202.68
TCE(gCO_2/day)	121181	120557	118061	114940
MPLAT(kWh@slot)	12.03@53 and 65	15.07@44	15.07@44	16.03@36

Table 8.7: Compare the result of use case with mixed time range, but standard mode with use case of fixed time range and energy carrier

C2. The user with hybrid energy carrier, but fixed time range.

• In this case, simulate the model initially with 10% of users allowed to use alternative energy carrier, keep 90% of users to use standard carrier and all 100% users preferred to run their appliances in fixed time range as shown in table8.1. The optimization result obtained is just 2.68% decrement compared to results of use case A with a fixed time range and energy carrier(standard mode) as shown in table8.9.

• After incrementing the users with 50% who allowed to use alternative energy carrier, 50% users with standard energy carrier and 100% users with a fixed time range. There is almost 13.39% decrement in the CO_2 emission compared to use case A with a fixed time range and energy carrier shown in table8.9.

• If we allowed 100% of users to use alternative energy carrier, then there is significant decrement occurs that is almost 27% compared to result of use case A as shown in table8.9.

• From the simulation, we came to know that hybrid property of the appliances more effective than the users operating in the mixed time range without alternative energy carrier. The following figure8.4 shows appliance load pattern in hybrid mode but fixed time range.

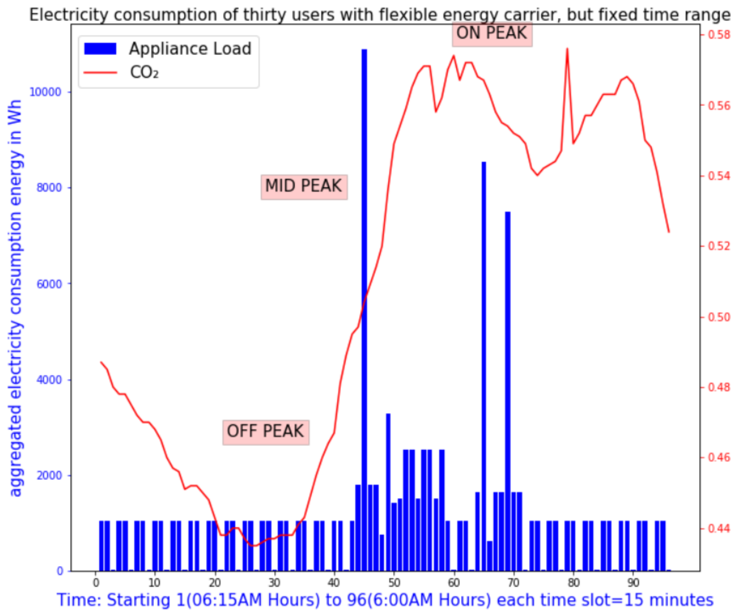

Figure 8.4: Electricity Load pattern of appliances operating in flexible energy carrier(hybrid mode), but fixed time range

Parameters	Case C2
TEC(kWh/day)	113.58
TCE(gCO_2/day)	88731.2
MPLAT(kWh@slot)	10.88@44

Table 8.8: Optimization results for thirty users with hybrid energy carrier, but fixed time range

Parameters	Case A	Case C2(10% hybrid mode)	Case C2(50% hybrid mode)	Case C2(100% hybrid mode)
TEC(kWh/day)	202.68	194.78	158.13	113.58
TCE(gCO_2/day)	121181	117936	104956	88731.2
MPLAT(kWh@slot)	12.03@53 and 65	15.28@64	12.94@44	10.88@44

Table 8.9: Compare the results of use case with hybrid energy carrier, but fixed time range with use case of fixed time range and standard energy carrier

C3. The user with hybrid energy carrier and mixed time range.

• In this case also, we start simulation initially for 10% of users allowed to use hybrid energy carrier, 90% of users with standard mode, 10% of users preferred to operate on mixed time and 90% of user operate on a fixed time range. The optimization result shows 2.82% decrement compared to use case A with fixed energy carrier and time range.

• Since we considered just 10% of users in overall 30 users, hence it is not affecting overall optimization result. Even 50% increment in the user group of mixed time range and hybrid mode does not make effective changes to optimization result. There is still 14% decrement in the optimization result compare to use case A with fixed energy carrier and time range as shown in table 8.11.

• If we allowed all the users to operate on hybrid mode and mixed time range, there is 28.18% significant decrement in the optimization result compared to result shown in table 8.11.

• From this result we came to know that, it would be better if the user planned their convenient mixed time range in advance and allowed the appliance to operate on hybrid mode. It gives flexibility to the scheduler to assign the appliance at a time slot with respect to CO_2 and energy carrier. Which gives a significant positive contribution to the environment. The following figure8.5 shows appliance load pattern thirty users operate in hybrid mode and mixed time range. The optimization result of this case is shown in table8.8

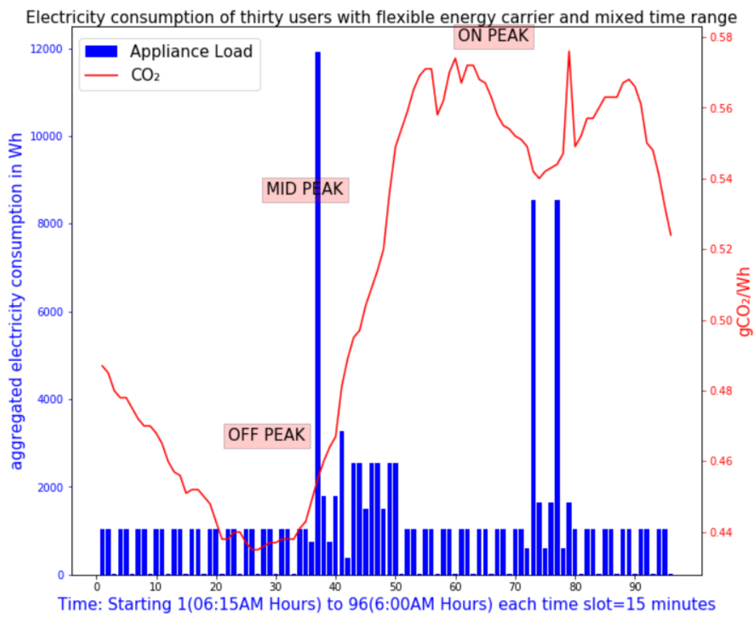

Figure 8.5: Electricity Load pattern of appliances operating in flexible energy carrier(hybrid mode) and mixed time range

Parameters	Case C3
TEC(kWh/day)	113.58
TCE(gCO_2/day)	87032.1
MPLAT(kWh@slot)	11.91@36

Table 8.10: Optimization results for thirty users with hybrid mode and mixed time range

Parameters	Case A	Case C3(10% mixed time range&hybrid mode)	Case C3(50% mixed time range&hybrid mode)	Case C3(100% mixed time range&hybrid mode)
TEC(kWh/day)	202.68	193.77	158.13	113.58
TCE(gCO_2/day)	121181	117766	104107	87032.1
MPLAT(kWh@slot)	12.03@53 and 65	14.53@51	8.53@51	11.91@36

Table 8.11: Compare the results of use case with hybrid energy carrier and mixed time range with use case of fixed time range and standard energy carrier

8.0.4 Days with renewable and nonrenewable energy generation

D1. Days with more renewable energy generation In this scenario, the electricity generators used more renewable energy to generate most of the electricity. Here we have used CO_2 emission data produced between 5th January 2018 6:15 AM to 6th January 2018 6:00 AM. The electricity producer used a resource like wind, solar, hydro, nuclear and biomass to produce most of the electricity. Hence the carbon emission is way less than the electricity produced using nonrenewable resources like fossil fuel, hard coal etc., The below fig8.6 shows the percentage of renewable and non-renewable electricity produced during the days from 5th to 6th of January. We simulate our model by imposing the constraining on the appliances to operate on hybrid mode and users allowed to operate on mixed time range preference as shown in table 8.5.

The following figure 8.7 shows the load pattern of appliances during the day at which renewable energy production is more. We can observe that even though highest load pattern occurs during the on-peak carbon emission, there is almost 34.8% decrement in the CO_2 emission compared to use case A with a fixed time range and energy carrier(standard mode) as shown in table8.13. Since we are simulating the model during less CO_2 emission day along with, users are allowed to their appliances to operate in hybrid mode and mixed time range.

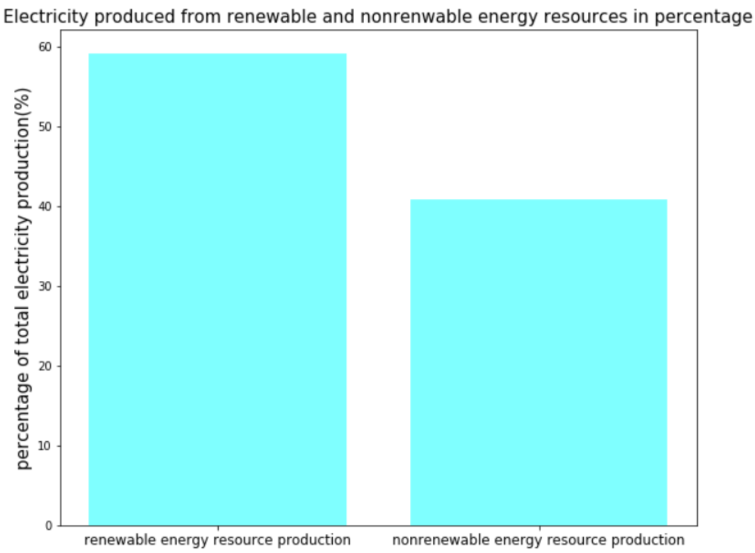

Figure 8.6: Percentage of renewable and nonrenewable electricity generation

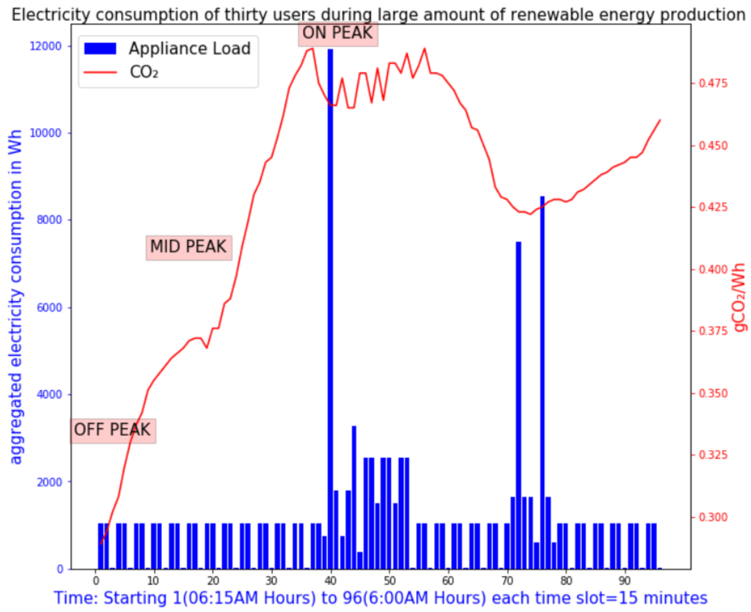

Figure 8.7: Electricity Load pattern of appliances operating in flexible energy carrier(hybrid mode) and mixed time range during the day of more renewable energy production

Parameters	CaseD1
TEC(kWh/day)	113.58
TCE(gCO_2/day)	79004.4
MPLAT(kWh@slot)	11.91@39

Table 8.12: Optimization results for thirty users with hybrid mode and mixed time range during the more renewable energy production day

Parameters	Case A	Case D1
TEC(kWh/day)	202.68	113.58
TCE(gCO_2/day)	121181	79004.4
MPLAT(kWh@slot)	12.03@53 and 65	11.91@39

Table 8.13: Compare the results of use case with full flexible time range and energy carrier during the more renewable day with use case of fixed time range and energy carrier

D2. Days with more non-renewable energy generation

In this case, most of the electricity produced from non-renewable resources like Fossil brown coal and Fossil gas. We have used emission data produced from 5th February 2018 6:15 AM to 6th February 2018 6:00 AM. We simulate our model by imposing the constraining on the appliances to operate on hybrid mode and accept the users mixed time range preference as shown in table 8.5. The following figure 8.8 shows load pattern of the appliances during the day at which nonrenewable energy production is more. The following figure8.8 shows that the highest peak load demand occurs at the mid peak carbon emission. Even though CO_2 emission is high during the day, there is almost

23.74% decrement in the CO_2 emission compared to use case with A fixed time range and energy carrier as shown in table8.15. It states advantage of using mixed time range and hybrid mode of the appliance during nonrenewable energy production.

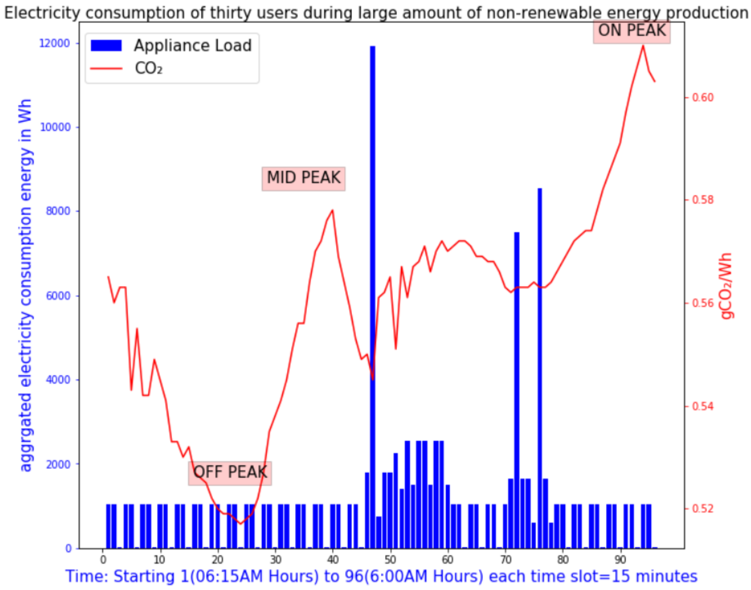

Figure 8.8: Electricity Load pattern of appliances operating in flexible energy carrier(hybrid mode) and mixed time range during the day of more nonrenewable energy production

Parameters	CaseD2
TEC(kWh/day)	113.58
TCE(gCO_2/day)	92739.1
MPLAT(kWh@slot)	11.91@39

Table 8.14: Optimization results for thirty users with hybrid mode and mixed time range during the more non-renewable energy production day

Parameters	Case A	Case D2
TEC(kWh/day)	202.68	113.58
TCE(gCO_2/day)	121181	92739.1
MPLAT(kWh@slot)	12.03@53 and 65	11.91@39

Table 8.15: Compare the results of use case with full flexible time range and energy carrier during the more nonrenewable day with use case of fixed time range and energy carrier

Parameters	Case A	Case B	Case C1	Case C2	Case C3	CaseD1	CaseD2
TEC(kWh/day)	202.68	113.58	202.68	113.58	113.58	113.58	113.58
TCE(gCO_2/day)	121181	83722.1	114940	88731.2	87032.1	79004.4	92739.1
MPLAT(kWh@slot)	12.03@53 and 65	10.88@20	16.03@36	10.88@44	11.91@36	11.91@39	11.91@39

Table 8.16: Results Comparison

9 Conclusion

In this thesis, we have created an android application that shows CO_2 emission intensity value of the Germany electricity grid as part of the Demand Response incentive-based program. We have used ENTSOE Transparency Platform API[1], that provides Germany electricity information for every fifteen-time interval. We have used CO_2 emission value of each resource type to calculate the total CO_2 emission intensity value of Germany using the formula presented in the paper[SBN14]. Based on the proposed architecture shown in fig4.2, we formulate the optimization problem of scheduling of hybrid appliances by using MILP technique. The objective of the optimization problem is to minimize the CO_2 emission value from the household appliances. We have simulated our model for four different case studies, while considering the customer preference in terms of the desired operation time of appliances. Simulation results are shown in table8.16 state that case A produce least total CO_2 emission value, ignoring the users preference constraint. On the other hand, case B produce the highest CO_2 emission value because considered both time and energy carrier fixed. In case C3, the emission reductions to 28.18% compared to case B, while just increased to 3.8% compared to case A. But in the case C3, we have considered the customer desired preference which encourages them to participate in the scheduling. The case D1 and D2 show the emission during more renewable and non-renewable electricity generation respectively. Both electricity and natural gas are supplied by the distribution grid. Since we are using natural gas to produce hot water by gas boiler locally. The CO_2 emission from the natural gas is always constant. All use cases result shown in table8.16. Both case A and case B consumed the same amount of total electricity but the total CO_2 emission is different. The reason is case A considers fixed time range and standard energy carrier whereas case B considers mixed time range and standard energy carrier. Even by giving flexibility in the operation time of appliances may cause a reduction in CO_2 emission which positively contributes to the environment. In case C, we considered three different combinations of cases. In each case increased the user group by step of 10%, it means in case c1 keep 10% of user operate on mixed time range, 90% of users with a fixed time range and all 100% of users appliances in standard energy carrier. In case c2 10% of users allowed the appliance to operate on hybrid mode, 90% of users standard mode and all 100% of users fixed time range. In case c3, considered 10% of user mixed time range and hybrid mode, 90% of users fixed time range and standard mode. The hybrid appliances play an important role in the future market, due to their feature of using multiple energy carriers. It encourages the consumer to aware about environment and installation of a microgrid in the home. Along with environmental concern helps to save electricity bill as well.

[1] https://transparency.entsoe.eu/content/static_content/Static%20content/web%20api/Guide.html

10 Future Work

In the future, we can extend our proposed model by including renewable energy and distributed generation technology like wind, solar and CHP and send the surplus electricity to the national grid once it satisfies house demand. Also, we can add appliances like space heater which operate using both electricity and hot water acts as hybrid appliances, air conditioned and heating element to heat the water. In this thesis, we have used historical data to schedule the appliance. In the future, we can get the day ahead electricity generation information from ENTSOE Transparency platform API, from that we can calculate the CO_2 emission value for next 24 hours and provide that information to the user using our android application. Also, accept the user desired appliance operation time as input through our android application and store it into out firebase database. Provide both CO_2 emission value and user input to our MILP model by connecting to the database. Then run our scheduler and provide each appliances optimal operation time window to the user using our android application.

Bibliography

[Agg17] A. Aggarwal. *REST API—Back to Basics*. 2017. URL: https://restful.io/rest-api-back-to-basics-c64f282d972 (cit. on p. 27).

[All13] F. Allerding. “Organic Smart Home – Energiemanagement für Intelligente Gebäude”. PhD thesis. Karlsruhe Institute of Technology, 2013 (cit. on p. 19).

[Bar18a] I. Barry & Associates. *Representational State Transfer (REST)*. 2018. URL: https://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html (cit. on p. 27).

[Bar18b] I. Barry & Associates. *Service Architecture*. 2018. URL: https://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html (cit. on p. 26).

[BHH+12] M. C. Bozchalui, S. A. Hashmi, H. Hassen, C. A. Canizares, K. Bhattacharya. “Optimal Operation of Residential Energy Hubs in Smart Grids”. In: *IEEE Transactions on Smart Grid* 3.4 (Dec. 2012), pp. 1755–1766. ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2212032 (cit. on p. 17).

[BHJ15] F. Brahman, M. Honarmand, S. Jadid. “Optimal electrical and thermal energy management of a residential energy hub, integrating demand response and energy storage system”. In: *Energy and Buildings* 90 (2015), pp. 65–75. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2014.12.039. URL: http://www.sciencedirect.com/science/article/pii/S0378778814011128 (cit. on pp. 19, 20, 50).

[CWF12] Z. Chen, L. Wu, Y. Fu. “Real-Time Price-Based Demand Response Management for Residential Appliances via Stochastic Optimization and Robust Optimization”. In: *IEEE Transactions on Smart Grid* 3.4 (Dec. 2012), pp. 1822–1831. ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2212729 (cit. on pp. 16, 34).

[ele18] elementsofheating. *How to calculate the kW required to heat a volume of water in a particular time*. 2018. URL: https://elementsofheating.wordpress.com/2012/09/26/how-to-calculate-the-kw-required-to-heat-a-volume-of-water-in-a-particular-time/ (cit. on p. 50).

[ene18] C. about energy. *Cookers*. 2018. URL: https://www.confusedaboutenergy.co.uk/index.php/buying-household-appliances/cookers (cit. on p. 47).

[ENT14] ENTSO-E. *ENTSO-E GENERATION AND LOAD TRANSPARENCY PROCESS IMPLEMENTATION GUIDE*. 2014. URL: https://www.entsoe.eu/fileadmin/user_upload/_library/resources/Transparency/MoP%5C%20Ref05%20-%5C%20EMFIP-1-gl-market-document_V3R0-2014-01-24.pdf (cit. on p. 31).

[ENT16] ENTSO-E. *ENTSO-E TRANSPARENCY PLATFORM DATA EXTRACTION PROCESS IMPLEMENTATION GUIDE*. 2016. URL: https://transparency.entsoe.eu/content/static_content/download?path=/Static%5C%20content/web%5C%20api/RestfulAPI_IG.pdf (cit. on pp. 28–30).

[Fir18a] Firebase. *Add Firebase to Your Android Project*. 2018. URL: https://firebase.google.com/docs/android/setup (cit. on p. 32).

[Fir18b] Firebase. *Read and Write Data on Android*. 2018. URL: https://firebase.google.com/docs/database/android/read-and-write (cit. on p. 32).

[Gui18a] neos Guide. *Integer Linear Programming*. 2018. URL: https://neos-guide.org/content/integer-linear-programming (cit. on p. 22).

[Gui18b] neos Guide. *Introduction to Optimization*. 2018. URL: https://neos-guide.org/content/optimization-introduction (cit. on pp. 21, 22).

[Gui18c] neos Guide. *Types of Optimization Problems*. 2018. URL: https://neos-guide.org/optimization-tree (cit. on p. 21).

[HBF14] M. Hayn, V. Bertsch, W. Fichtner. "Electricity load profiles in Europe: The importance of household segmentation". In: *Energy Research & Social Science* 3 (2014), pp. 30–45. ISSN: 2214-6296. DOI: https://doi.org/10.1016/j.erss.2014.07.002. URL: http://www.sciencedirect.com/science/article/pii/S2214629614000802 (cit. on pp. 13, 16, 34).

[Jav18] Javatpoint. *Android Core Building Blocks*. 2018. URL: https://www.javatpoint.com/android-core-building-blocks (cit. on p. 25).

[jav18a] javatpoint. *History of Android*. 2018. URL: https://www.javatpoint.com/android-history-and-versions (cit. on p. 25).

[jav18b] javatpoint. *what is Android*. 2018. URL: https://www.javatpoint.com/android-what-where-and-why (cit. on p. 25).

[LMDB12] J. M. Lujano-Rojas, C. Monteiro, R. Dufo-López, J. L. Bernal-Agustín. "Optimum residential load management strategy for real time pricing (RTP) demand response programs". In: *Energy Policy* 45 (2012), pp. 671–679. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2012.03.019. URL: http://www.sciencedirect.com/science/article/pii/S0301421512002248 (cit. on p. 13).

[MSS15] I. Mauser, H. Schmeck, U. Schaumann. "Optimization of Hybrid Appliances in Future Households". In: *International ETG Congress 2015; Die Energiewende - Blueprints for the new energy age*. Nov. 2015, pp. 1–6 (cit. on pp. 13, 18, 19).

[Opt18a] G. Optimization. *Math Programming Modeling Basics*. 2018. URL: http://www.gurobi.com/resources/getting-started/modeling-basics (cit. on p. 24).

[Opt18b] G. Optimization. *Starting with Gurobi*. 2018. URL: http://www.gurobi.com/resources/getting-started/starting-with-gurobi (cit. on p. 22).

[Opt18c] G. Optimization. *The Gurobi Python Modeling and Development Environment*. 2018. URL: http://www.gurobi.com/resources/getting-started/gurobi-python-modeling-and-development-environment (cit. on p. 23).

[QNK+15] F. A. Qayyum, M. Naeem, A. S. Khwaja, A. Anpalagan, L. Guan, B. Venkatesh. "Appliance Scheduling Optimization in Smart Home Networks". In: *IEEE Access* 3 (2015), pp. 2176–2190. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2496117 (cit. on pp. 17, 47, 50, 54, 58).

[RP18] P. A. Rogers, D. O. Parson. "GridCarbon: A smartphone app to calculate the carbon intensity of the UK electricity grid". In: (2018) (cit. on p. 16).

[SBN14] P. Stoll, N. Brandt, L. Nordström. "Including dynamic CO2 intensity with demand response". In: *Energy Policy* 65 (2014), pp. 490–500. ISSN: 0301-4215. DOI: https://doi.org/10.1016/j.enpol.2013.10.044. URL: http://www.sciencedirect.com/science/article/pii/S0301421513010677 (cit. on pp. 13, 16, 17, 40, 67).

[SBP+08] R. Stamminger, G. Broil, C. Pakula, H. Jungbecker, M. Braun, I. Rüdenauer, C. Wendker. "Synergy potential of smart appliances". In: *Report of the Smart-A project* (2008), pp. 1949–3053 (cit. on pp. 19, 46).

[Sta18] Statista. *Smartphone users worldwide 2014-2020*. 2018. URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (cit. on p. 14).

[Stu18] A. Studio. *Android Studio Install*. 2018. URL: https://developer.android.com/studio/install (cit. on p. 26).

[SWSJ11] K. C. Sou, J. Weimer, H. Sandberg, K. H. Johansson. "Scheduling smart home appliances using mixed integer linear programming". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. Dec. 2011, pp. 5144–5149. DOI: 10.1109/CDC.2011.6161081 (cit. on p. 17).

[SXZ14] D. Setlhaolo, X. Xia, J. Zhang. "Optimal scheduling of household appliances for demand response". In: *Electric Power Systems Research* 116 (2014), pp. 24–28. ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2014.04.012. URL: http://www.sciencedirect.com/science/article/pii/S0378779614001527 (cit. on pp. 18, 41).

[Wik18a] Wikipedia. *Branch and bound*. 2018. URL: https://en.wikipedia.org/wiki/Branch_and_bound (cit. on p. 24).

[Wik18b] Wikipedia. *Job shop scheduling*. 2018. URL: https://en.wikipedia.org/wiki/Job_shop_scheduling (cit. on p. 24).

[ZGSS11] D. Zhang, L. G Papageorgiou, N. Samsatli, N. Shah. "Optimal Scheduling of Smart Homes Energy Consumption with Microgrid". In: (Jan. 2011) (cit. on p. 20).

A Python client Application sample code

```
"""
Firebase services

"""

import json
from bs4 import BeautifulSoup
import dateutil.parser as dp
from datetime import datetime, timedelta

import requests


class Firebase:
    """class that provides firebase services
    """

    def __init__(self, fb_cfg):
        self.database_url = fb_cfg.database_url

    def __str__(self):
        return "Connection to firebase services"

    def realtime_database(self):
        return RealtimeDatabase(self.database_url)


class RealtimeDatabase:
    """class the perform REST API calls to Realtime database
    """

    def __init__(self, database_url):
        self.database_url = database_url
        self.path = ""

    def _check_token(self):
        return f"{self.database_url}{self.path}.json"

    def post(self, path, data):
        # TODO handle response errors?
        self.path = path
        response = requests.post(self._check_token(), data=data)
        return response.json()
```

```
PSR_TYPES = {
    'B01': ['Biomass', 18],
    'B02': ['Fossil Brown coal', 1001],
    'B03': ['Fossil Coal Gas', 469],
    'B04': ['Fossil Gas', 469],
    'B05': ['Fossil Hard coal', 1001],
    'B06': ['Fossil oil', 840],
    'B09': ['Geothermal', 45],
    'B10': ['Hydro Storage', 5],
    'B11': ['Hydro Poundage', 5],
    'B12': ['Hydro Reserviour', 5],
    # 'B13': ['Marine', 8],
    'B14': ['Nuclear', 16],
    'B15': ['Other Renewable', 700],
    'B16': ['Solar', 46],
    'B17': ['waste', 700],
    'B18': ['Wind offshore', 12],
    'B19': ['Wind Onshore', 12],
    'B20': ['Other', 700]
    }


class Entsoe:

    def __init__(self, entsoe_cfg):
        self.cfg = entsoe_cfg

        self.endpoint = self.cfg.endpoint
        self.params = self.cfg.params
        self.periodStart = '0'
        self.data = {}
        self.sum_ef = 0
        self.sum_quantity = 0
        self.co2 = 0
        self.quantity = 0

    def setup(self):
        # loop over all resources
        flag = True;
        flag1 = False;
        for res in self.cfg:
            # set url psrType
            self.params["psrType"] = res

            try:

                # self.quantity = float(BeautifulSoup(requests.get(self.endpoint, params=self.
params).text,
                #                             "html.parser").find("quantity").text)
                response_url = requests.get(self.endpoint, params=self.params)
                print(response_url.url)
                response = response_url.text
```

```
                soup = BeautifulSoup(response, "html.parser")
                parsed_t = dp.parse(soup.find("timeseries").find("period").find("end").text)
                periodEnd = parsed_t.strftime("%Y%m%d%H%M")
                print("the scraped periodEnd for ", res, "is", periodEnd)
                all_quantity = soup.find("timeseries").find("period").findAll("quantity")


                #lets keep it 7:45 , because utc time(9.15) - 1.5 hours
                copy_periodStart = datetime.strptime(self.params.get("periodStart"),"%Y%m%d%H%
M")

                while flag1:
                    #if periodstart set 15min after actual period start
                    if self.periodStart == copy_periodStart.strftime("%Y%m%d%H%M"):
                        self.quantity = float(all_quantity[0].getText())
                        break

                    elif self.periodStart == (copy_periodStart +timedelta(minutes= 15)).
strftime("%Y%m%d%H%M"):
                        if self.periodStart == periodEnd:
                            self.quantity = float(all_quantity[0].getText())
                            break
                        else:
                            self.quantity = float(all_quantity[1].getText())
                            break

                    # if periodstart set 30min after actual period start
                    elif self.periodStart == (copy_periodStart +timedelta(minutes= 30)).
strftime("%Y%m%d%H%M"):
                        if self.periodStart == periodEnd:
                            self.quantity = float(all_quantity[1].getText())
                            break
                        else:
                            self.quantity = float(all_quantity[2].getText())
                            break

                    # if periodstart set 45min after actual period start
                    elif self.periodStart == (copy_periodStart +timedelta(minutes= 45)).
strftime("%Y%m%d%H%M"):
                        if self.periodStart == periodEnd:
                            self.quantity = float(all_quantity[2].getText())
                            break
                        else:
                            self.quantity = float(all_quantity[3].getText())
                            break

                    # if periodstart set 1hour after actual period start
                    elif self.periodStart == (copy_periodStart + timedelta(hours=1)).strftime(
"%Y%m%d%H%M"):
                        if self.periodStart == periodEnd:
                            self.quantity = float(all_quantity[3].getText())
                            break
                        else:
```

```
                        self.quantity = float(all_quantity[4].getText())
                        break
                # if periodstart set 1 hour 15min after actual period start

                else:
                    if self.periodStart == periodEnd:
                        self.quantity = float(all_quantity[4].getText())
                        break
                    else:
                        self.quantity = float(all_quantity[5].getText())
                        break

            while flag:
                flag = False
                flag1 = True
                #7.45+ 15 = 8:00
                if periodEnd == (copy_periodStart +timedelta(minutes= 15)).strftime("%Y%m%
d%H%M"):
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = copy_periodStart.strftime("%Y%m%d%H%M")

                # 7.45+ 30 = 8:30
                elif periodEnd == (copy_periodStart +timedelta(minutes= 30)).strftime("%Y%
m%d%H%M"):
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = (copy_periodStart + timedelta(minutes=15)).strftime
("%Y%m%d%H%M")

                # 7.45+ 45 = 8:15
                elif periodEnd == (copy_periodStart +timedelta(minutes= 45)).strftime("%Y%
m%d%H%M"):
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = (copy_periodStart + timedelta(minutes=30)).strftime
("%Y%m%d%H%M")

                # 7.45+ 1 hour = 8:45
                elif periodEnd == (copy_periodStart +timedelta(hours=1)).strftime("%Y%m%d%
H%M"):
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = (copy_periodStart + timedelta(minutes=45)).strftime
("%Y%m%d%H%M")

                # 7.45+ 1 hour and 15 min = 9:00
                elif periodEnd == (copy_periodStart +timedelta(hours=1, minutes=15)).
strftime("%Y%m%d%H%M"):
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = (copy_periodStart + timedelta(hours=1)).strftime("%
Y%m%d%H%M")

                else:
                    self.quantity = float(all_quantity[-1].getText())
                    self.periodStart = (copy_periodStart + timedelta(hours=1, minutes=15))
.strftime("%Y%m%d%H%M")
```

```
            except (AttributeError, IndexError) :
                self.quantity = 0
                self.periodStart =self.params.get("periodStart")
                print("Errror occured while parsing the XML")

            res_ef = self.quantity * 0.25 * PSR_TYPES.get(res)[-1]
            self.sum_quantity += self.quantity
            self.sum_ef += res_ef
            self.data[res] = self.quantity

    def resources(self):
        res_data = {PSR_TYPES.get(key)[0]: val for key, val in self.data.items()}
        res_data.update({"periodStart": self.periodStart})
        print(res_data)
        # default UTF-8
        return json.dumps(res_data).encode()

    def calculate_co2(self):
        try:
            return self.sum_ef / (self.sum_quantity * 0.25)
        except ZeroDivisionError:
            print("There is no value for this time")

    def emission_factor(self):
        ef_data = {
            "co2"        : self.calculate_co2(),
            "periodStart": self.periodStart,
            "Electricity": self.sum_quantity
            }
        print(ef_data)
        return json.dumps(ef_data).encode()
```

B Android application sample code

```
package com.thesis.carbon;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.*;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.view.*;
import android.support.v7.widget.*;
import android.util.Log;
import android.view.View;
import android.widget.*;

import com.github.mikephil.charting.components.*;
import com.google.firebase.database.*;

import java.text.SimpleDateFormat;
import java.util.*;


import static com.thesis.carbon.Constants.*;

public class HomeActivity extends Activity {

    ViewPager viewPager;

    Integer[] imageId = {R.drawable.co2, R.drawable.co2};
    //String[] imagesName = {"205","22.8"};
    ArrayList<Integer> imagesName = new ArrayList<>();
    String[] unitArray = {"gCO2/kWh","MW"};

    private FloatingActionButton fab;

    int currentPage = 0;
    Timer timer;
    final long DELAY_MS = 5000;//delay in milliseconds before task is to be executed
    final long PERIOD_MS = 10000;
    private static int NUM_PAGES = 0;
    private RecyclerView recycler_view;
    private List<ElectricityModel> electricityModelList = new ArrayList<>();
    private ArrayList<String> nameArray = new ArrayList<>();
    private ArrayList<Integer> valArray = new ArrayList<>();
    private ArrayList<Integer> imageArray = new ArrayList<>();
    private ArrayList<Integer> efValues = new ArrayList<>();
```

```
private ElectricityAdapter electricityAdapter;
DatabaseReference rootRef;
DatabaseReference resorceref;
DatabaseReference co2ref;

private String co2EmissionTime;
private String formattedDate;
private String str_currentDate;
private float co2EmissionValue;
private int electricityValue;

private Button chart1;
private Button chart2;

private TextView text_sync_time;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);

    chart1 = (Button)findViewById(R.id.chart1);
    chart2 = (Button)findViewById(R.id.chart2);

    Calendar c = Calendar.getInstance();
    c.add(Calendar.HOUR, -4);

    SimpleDateFormat df = new SimpleDateFormat("yyyyMMdd");
    SimpleDateFormat current_date = new SimpleDateFormat("dd-MMM-yyyy");
    str_currentDate = current_date.format(c.getTime());
    formattedDate = df.format(c.getTime());

    nameArray.clear();
    valArray.clear();

    imageArray.add(R.drawable.img_fact);
    imageArray.add(R.drawable.img_coal);
    imageArray.add(R.drawable.img_gas);
    imageArray.add(R.drawable.img_gas);
    imageArray.add(R.drawable.img_geoth);
    imageArray.add(R.drawable.img_hydrop);
    imageArray.add(R.drawable.img_nuclear);
    imageArray.add(R.drawable.img_solar_energy);
    imageArray.add(R.drawable.img_wind);
    imageArray.add(R.drawable.img_renewable);

    efValues.add(18);
    efValues.add(1001);
    efValues.add(469);
    efValues.add(840);
    efValues.add(45);
    efValues.add(5);
    efValues.add(16);
```

```
        efValues.add(46);
        efValues.add(12);
        efValues.add(700);

      //  4627*0.25*469

        rootRef = FirebaseDatabase.getInstance().getReference();
        resorceref = rootRef.child("resources");
        co2ref = rootRef.child("emission_factor");

        fetchCo2data();
        firebasedata();
        firebasedata3();

        recycler_view = (RecyclerView)findViewById(R.id.recycler_view);
        text_sync_time = (TextView)findViewById(R.id.text_sync_time);

        text_sync_time.setText(str_currentDate);

        chart1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(), CombinedChartActivity.
class);
                startActivity(intent);
                finish();
            }
        });

        chart2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplicationContext(), BarchartActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }

    private void fetchCo2data() {

        Query lastQuery = co2ref.orderByKey().limitToLast(1);

        lastQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                for (DataSnapshot postSnapshot : snapshot.getChildren()) {

                    String periodStart11 = postSnapshot.child("periodStart").getValue(String.
class);
                    String timep1 = periodStart11.substring(0,8);

                    if(timep1.equals(formattedDate)){
```

```
                        co2EmissionTime = postSnapshot.child("periodStart").getValue(String.
class);
                        co2EmissionValue = postSnapshot.child("co2").getValue(Float.class);
                        electricityValue = postSnapshot.child("Electricity").getValue(Integer.
class);

                        int co2int = Math.round(co2EmissionValue);
                        imagesName.add(co2int);
                        imagesName.add(electricityValue);
                    }

                }
                viewPager = (ViewPager) findViewById(R.id.viewPager);
                final PagerAdapter adapter = new CustomAdapter(HomeActivity.this,imageId,
imagesName,unitArray);
                viewPager.setAdapter(adapter);

                NUM_PAGES = imageId.length;

                final Handler handler = new Handler();
                final Runnable Update = new Runnable() {
                    public void run() {
                        if (currentPage == NUM_PAGES) {
                            currentPage = 0;
                        }
                        viewPager.setCurrentItem(currentPage++, true);
                    }
                };

                timer = new Timer(); // This will create a new Thread
                timer .schedule(new TimerTask() { // task to be scheduled

                    @Override
                    public void run() {
                        handler.post(Update);
                    }
                }, DELAY_MS, PERIOD_MS);
            }
            @Override
            public void onCancelled(DatabaseError firebaseError) {

            }
        });
    }
```

C Python Model sample code

```
from gurobipy import *
get_ipython().run_line_magic('run', 'data.ipynb')

# Create the new model

m = Model("scheduling")

# Create Binary variables for model

vars_tup = [(t, u, app, task) for t in time_slots for u in users for app in appliances for
task in task_appliances[app]]

x = m.addVars(vars_tup, vtype = GRB.BINARY, name = 'x')

y = m.addVars(vars_tup, vtype = GRB.BINARY, name = 'y')

z = m.addVars(vars_tup, vtype = GRB.BINARY, name = 'z')

s = m.addVars(time_slots, vtype = GRB.BINARY, name = 's')

# # Creating the Variables for the Model

available_hotwater_storagetank = m.addVars(time_slots, lb = 0, ub = 31719 , vtype = GRB.
CONTINUOUS, name = 'available_hotwater_storagetank')

boiler_produced_thermal = m.addVars(time_slots, lb = 0, ub = 21250 , vtype = GRB.CONTINUOUS,
name = "boiler_produced_thermal")

boiler_produced_hotwater = m.addVars(time_slots, lb =0, ub = 125615 ,  vtype = GRB.CONTINUOUS,
 name = "boiler_produced_hotwater")

boiler_required_gas = m.addVars(time_slots, lb = 0, ub = 19125, vtype = GRB.CONTINUOUS, name =
 "boiler_required_gas")


# update the model

m.update()

# # Set the Objective Function

obj = quicksum(co2_factor_electricity[t] *  float(electricity_high_req[u, app, task]) * x[t, u
, app, task] +
```

```
               co2_factor_electricity[t] * float(electricity_low_req[u, app, task]) * y[t, u,
app, task]  + co2_factor_gas * float(hotwater_req[u, app, task]) *  y[t, u, app, task] +
               co2_factor_gas * float(gas_req[u, app, task]) * z[t, u , app, task]  +
co2_factor_gas * boiler_required_gas[t]
                 for t in time_slots for u in users for app in appliances for task in
task_appliances[app])

m.setObjective(obj, GRB.MINIMIZE)

# Update the Model.

m.update()


# # Create the Constraints for the model

# Subject to Constraints

energy_carrier = m.addConstrs((x[t, u, app, task] + y[t, u, app, task] + z[t, u, app, task] <=
 1
             for t in time_slots for u in users for app in appliances for task in
task_appliances[app]),
                              name = "energy_carrier")


peak_constraint = m.addConstrs((quicksum(electricity_high_req[u, app, task] * x[t, u, app,
task] + electricity_low_req[u, app, task] * y[t, u, app, task]
                                          for u in users for app in appliances for task in
task_appliances[app]) <= PEAK for t in time_slots),
                              name = "peak_constraint")


hotwater_demand_constarint = m.addConstrs((quicksum(hotwater_req[u, app, task] * y[t, u, app,
task] for app in appliances for task in task_appliances[app]) <= tank_capacity
                                           for t in time_slots for u in users), name = "
hotwater_demand_constarint")


boiler_gas = m.addConstrs((boiler_required_gas[t] == (boiler_produced_thermal[t]/
boiler_efficiecny) for t in time_slots),
            name = "boiler_gas")

boiler_hotwater = m.addConstrs((boiler_produced_hotwater[t] == (boiler_produced_thermal[t] *
3412)/(2.22 * 65 * 4)
                                for t in time_slots), name = "boiler_hotwater")



Boiler_capacity = m.addConstrs((0  <= boiler_produced_thermal[t] <= 21250  for t in time_slots
),
                               name = "Boiler_capacity")
```

```
boiler_hotwater_capacity = m.addConstrs((0 <= boiler_produced_hotwater[t] <=   125615 for t in
 time_slots), name = "Boiler_hotwater_capacity")



available_hotwater = m.addConstrs((available_hotwater_storagetank[time_slots[t_index]] ==
                                   available_hotwater_storagetank[time_slots[t_index-1]] +
boiler_produced_hotwater[time_slots[t_index]] -
                                   quicksum(hotwater_req[u, app, task] * y[time_slots[t_index
], u, app, task] for app in appliances for task in task_appliances[app])
                                  for t_index, t in enumerate(time_slots) for u in users if
t_index != 0))

available_hotwater_capacity = m.addConstrs((0 <= available_hotwater_storagetank[t] <=
tank_capacity
                                            for t in time_slots), name = "
available_hotwater_capacity")

not_perform_task = m.addConstrs((quicksum(x[time_slots[t_index], u, app, task] + y[time_slots[
t_index], u, app, task]+ z[time_slots[t_index], u, app, task]
                                          for t_index, t in enumerate(time_slots) if t_index
not in range(starting_time_slot[app]-1, ending_time_slot[app]) and t_index != len(time_slots))
                                          == 0 for u in users for app in appliances for task
in task_appliances[app]),
                                          name = "not_perform_task")

appliance_task_contionously1 = m.addConstrs((x[time_slots[t_index], u, app, task_appliances[
app][task_index]] ==
                                            x[time_slots[t_index-1], u, app, task_appliances[
app][task_index-1]]
                                            for t_index, t in enumerate(time_slots) for u in
users for app in appliances for task_index, task in enumerate(task_appliances[app])
                                            if t_index != 0 and task_index !=0), name = "
appliance_task_contionously1")

appliance_task_contionously2 = m.addConstrs((y[time_slots[t_index], u, app, task_appliances[
app][task_index]] ==
                                            y[time_slots[t_index-1], u, app, task_appliances[
app][task_index-1]]
                                            for t_index, t in enumerate(time_slots) for u in
users for app in appliances for task_index, task in enumerate(task_appliances[app])
                                            if t_index != 0 and task_index !=0), name = "
appliance_task_contionously2")

appliance_task_contionously3 = m.addConstrs((z[time_slots[t_index], u, app, task_appliances[
app][task_index]] ==
                                            z[time_slots[t_index-1], u, app, task_appliances[
app][task_index-1]]
                                            for t_index, t in enumerate(time_slots) for u in
users for app in appliances for task_index, task in enumerate(task_appliances[app])
                                            if t_index != 0 and task_index !=0), name = "
appliance_task_contionously3")
```

```
starts_with_1st_task = m.addConstrs((x[time_slots[t_index], u, app, task] == 1 for t_index, t
in enumerate(time_slots) for u in users for app in appliances for task in task_appliances[app]
                                    if app == appliances[3] and t_index == 0 and task ==
task_appliances[app][0]),
                                   name = "starts_with_1st_task")

dryer_after_washer1 = m.addConstrs((x[time_slots[t_index + 1], u, 'dryer', 'drying1'] <= x[
time_slots[t_index], u, 'washingmachine', '3rdrinse'] for t_index, t in enumerate(time_slots)
for u in users if t_index != len(time_slots) -1),
                                  name = 'dryer_after_washer1')

dryer_after_washer2 = m.addConstrs((y[time_slots[t_index + 1], u, 'dryer', 'drying1'] <= y[
time_slots[t_index], u, 'washingmachine', '3rdrinse'] for t_index, t in enumerate(time_slots)
for u in users if t_index != len(time_slots) -1),
                                   name = 'dryer_after_washer2')

dryer_after_washer3 = m.addConstrs((z[time_slots[t_index + 1], u, 'dryer', 'drying1'] <= z[
time_slots[t_index], u, 'washingmachine', '3rdrinse'] for t_index, t in enumerate(time_slots)
for u in users if t_index != len(time_slots) -1),
                                  name = 'dryer_after_washer3')

appliance_runtime = m.addConstrs((quicksum(x[t, u, app, task] + y[t, u, app, task] + z[t, u,
app, task] for t in time_slots for task in task_appliances[app])== run_time[app] for u in
users for app in appliances),
                                                          name = "appliances_runtime")

washingmachine_no_gas = m.addConstrs((z[t, u, app, task ] == 0 for t in time_slots for u in
users for app in appliances for task in task_appliances[app] if app == appliances[0]), name =
"washingmachine_no_gas")

dryer_no_gas = m.addConstrs((z[t, u, app, task ] == 0 for t in time_slots for u in users for
app in appliances for task in task_appliances[app]
                                     if app == appliances[1]), name = "dryer_no_gas")

dishwasher_no_gas = m.addConstrs((z[t, u, app, task ] == 0 for t in time_slots for u in users
for app in appliances for task in task_appliances[app]
                                     if app == appliances[2]), name = "dishwasher_no_gas")

gashob1_no_electricity_hotwater = m.addConstrs((x[t, u, app, task ] + y[t, u, app, task] == 0
for t in time_slots for u in users for app in appliances for task in task_appliances[app]
                                     if app == appliances[4]), name = "
gashob1_no_electricity_hotwater")

gashob2_no_electricity_hotwaterr = m.addConstrs((x[t, u, app, task ] + y[t, u, app, task] == 0
 for t in time_slots for u in users for app in appliances for task in task_appliances[app]
                                     if app == appliances[5]), name = "
gashob2_no_electricity_hotwater")

appliance_first_task = m.addConstrs((quicksum(x[time_slots[t_index], u, app, task] + y[
time_slots[t_index], u, app, task] + z[time_slots[t_index], u, app, task]
                                             for t_index in range(starting_time_slot[app]-1,
ending_time_slot[app]-run_time[app])) == 1
```

```
                                               for u in users for app in appliances for task in
 task_appliances[app]
                                     if app != "refrigerator" and task == task_appliances[app
][0]), name = "appliance_first_task")

# Update the Model
m.update()

# # Call the Optimize
# m.optimize()
```

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

____

place, date, signature