

# BROKOA: Cloud-based Brokerage System for Smart Grids

By

Howard Andrés Martínez Meza

A project report submitted in partial fulfillment of the requirements for the degree

of

MASTER OF ENGINEERING

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO

MAYAGÜEZ CAMPUS

2018

Approved by:

---

Manuel Rodríguez Martínez, Ph.D.  
President, Graduate Committee

---

Date

---

Wilson Rivera Gallego, Ph.D.  
Member, Graduate Committee

---

Date

---

Agustin Irizarry Rivera, Ph.D.  
Member, Graduate Committee

---

Date

---

Pedro I. Rivera Vega, Ph.D.  
Member, Graduate Committee

---

Date

---

Juan G. Araya Martin, Ph.D.  
Graduate School Representative

---

Date

---

José Colom Ustariz, Ph.D.  
Department Chairperson

---

Date

Abstract of Project Presented to the Graduate School  
of the University of Puerto Rico in Partial Fulfillment of the  
Requirements for the Degree of Master of Engineering

## **BROKOA: Cloud-based Brokerage System for Smart Grids**

Electric energy networks provide the necessary energy to carry out daily operations in education, health care, commerce, entertainment, defense, and government. Smart grids (SG) have been proposed as a mechanism to modernize and facilitate the operation of energy grids in the presence of multiple third-parties vying to sell electricity and related services. Recently, a new concept has emerged in relation with SG: transactive energy, a framework to integrate and coordinate multiple independent suppliers of electric services in an economic marketplace that enables purchases in a way that obeys the physical constraints of the underlying energy grid. Researchers at the University of Puerto Rico at Mayagüez in the field of Social Sciences, Computer Engineering and Electrical Engineering are developing a transactive energy framework called Open Access Smart Grids Services (OASIS). OASIS features a distribution-level marketplace where the underlying electric resources and services are open to access and provisioned by third parties. In this research project, we present BROKOA, a cloud-based brokerage system for smart grids, which is a critical component of OASIS. In this brokerage system, consumer energy requests, and energy bids issued by producers, are collected and matched to find a list of energy purchase orders to be dispatched to the winning producers. A central piece in the system is an energy broker software agent, which drives the bidding process.

Resumen de proyecto presentado a la Escuela Graduada  
de la Universidad de Puerto Rico como requisito parcial de los  
requerimientos para el grado de Maestría en Ingeniería

## **BROKOA: Sistema de bróker basado en la nube para redes eléctricas inteligentes**

Las redes de energía eléctrica proporcionan la energía necesaria para llevar a cabo las operaciones diarias en educación, salud, comercio, entretenimiento, defensa y gobierno. Se han propuesto las redes inteligentes como un mecanismo para modernizar y facilitar el funcionamiento de las redes eléctricas en presencia de múltiples terceros que compiten por vender electricidad y servicios relacionados. Recientemente surgió un nuevo concepto en relación con las redes inteligentes: la energía transactiva, este concepto hace referencia a un marco de trabajo para integrar y coordinar múltiples proveedores independientes de servicios eléctricos en un mercado económico que permite realizar compras de forma que obedezcan las restricciones físicas de la red eléctrica subyacente. Investigadores de la Universidad de Puerto Rico en Mayagüez en el campo de las Ciencias Sociales, Ingeniería de Computadoras e Ingeniería Eléctrica están desarrollando un marco de energía transactiva llamado Servicios de Redes Inteligentes de Acceso Abierto (OASIS por sus siglas en inglés). OASIS cuenta con un mercado de distribución donde los recursos y servicios eléctricos subyacentes están abiertos al acceso y aprovisionados por terceros. En este proyecto de investigación, presentamos BROKOA, un sistema de bróker basado en la nube para redes inteligentes, el cual es un componente crítico de OASIS. En este sistema de bróker, las solicitudes de energía de los consumidores y las ofertas de energía emitidas por los productores se recogen y se comparan para encontrar una lista de órdenes de compra de energía que se enviarán a los productores ganadores. El bróker es una pieza central en el sistema el cual impulsa el proceso de licitación.

Copyright ©2018

*by*

*Howard Andrés Martínez Meza*

*I dedicate this to my family, thanks for helping me get to this point and to every one  
that supported me.*

## Acknowledgment

I would like to thank the entire OASIS-CRISP project team to make possible this work. I would also like to thank the assistance of all persons and volunteers whose participation was essential in the successful completion of the work, thanks also to guidance of my advisor Dr. Manuel Rodríguez. Last but not less, i would like to thank the University of Puerto Rico for providing the environment for my professional training.

This project is funded in part by NSF, under grant # ACI-1541106. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

# Contents

Abstract	ii
Spanish Abstract	iii
Acknowledgment	vi
List of Figures	xii
List of Tables	xv
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives . . . . .	4
1.3 Contributions . . . . .	5
1.4 Outline . . . . .	6
<b>2 LITERATURE REVIEW</b>	<b>7</b>
2.1 Electric Networks . . . . .	7
2.2 Energy Markets . . . . .	9
2.3 Transactive Energy . . . . .	11
2.3.1 Transactive Energy Projects . . . . .	12
2.4 Energy Resources and Emissions . . . . .	14

<b>3</b>	<b>METHODOLOGY</b>	<b>16</b>
3.1	OASIS Energy Market Overview . . . . .	18
3.2	OASIS Consumer . . . . .	19
3.3	OASIS Producer . . . . .	20
3.4	OASIS Broker . . . . .	22
3.4.1	Fifo Algorithm . . . . .	22
3.4.2	Knapsack Algorithm . . . . .	23
3.5	OASIS Energy Market Implementation . . . . .	26
3.5.1	System Architecture . . . . .	26
3.5.2	Web Interface . . . . .	27
3.5.3	Application Data Model . . . . .	36
3.6	Energy Market Communication with Hardware-in-the-loop . . . . .	38
<b>4</b>	<b>BROKER MODULE &amp; ALGORITHMS</b>	<b>42</b>
4.1	System Overview . . . . .	42
4.1.1	Job . . . . .	42
4.1.2	Trigger . . . . .	43
4.1.3	Scheduler . . . . .	43
4.2	Algorithms . . . . .	43
4.2.1	FIFO Algorithm . . . . .	44
4.2.2	Knapsack Algorithm . . . . .	48
<b>5</b>	<b>RESULTS</b>	<b>52</b>
5.1	Efficiency . . . . .	52
5.2	Outcomes . . . . .	53
5.2.1	Demand > Offer . . . . .	54
5.2.2	Demand = Offer . . . . .	57
5.2.3	Demand < Offer . . . . .	60



*CONTENTS*

*CONTENTS*

<b>6 CONCLUSIONS AND FUTURE WORK</b>	<b>65</b>
<b>Appendices</b>	<b>68</b>
<b>A Broker - Components</b>	<b>69</b>
<b>B Broker - Algorithms</b>	<b>73</b>

**List of Abbreviations**

**AES** Auction Energy System

**CO<sub>2</sub>** Carbon dioxide

**DERs** Distributed Energy Resources

**DG** Distributed Generation

**FIFO** First In First Out

**GHG** Greenhouse Gase

**GWAC** GridWise Architecture Council

**HEMS** Home Energy Management System

**HTTP** Hypertext Transfer Protocol

**IaaS** Infrastructure as a Service

**IEEE** Institute of Electrical and Electronics Engineers

**JSON** JavaScript Object Notation

**LPG** Liquefied petroleum gas

**NIST** National Institute of Standards

**OASIS** Open Access Smart Grids Services

**PNNL** Pacific Northwest National Laboratory

**PVS** Photovoltaic Systems

**SaaS** Software as a Service

*CONTENTS*

*CONTENTS*

**SG** Smart Grids

**TE** Transactive Energy

**UUID** Universally Unique Identifier

# List of Figures

- 1.1 OASIS System Architecture [5, 7] . . . . . 3
- 2.1 Traditional power system structure. . . . . 8
- 2.2 Spot transaction. . . . . 10
- 2.3 Forward contracts. . . . . 10
- 2.4 Future trading / Future contracts. . . . . 11
- 3.1 Electric services mapped to cloud services. [5, 7] . . . . . 17
- 3.2 OASIS Energy Market Overview. . . . . 18
- 3.3 Consumer’s Profiles . . . . . 20
- 3.4 Bid’s states . . . . . 21
- 3.5 Broker’s bid assignment scenario . . . . . 22
- 3.6 Knapsack Algorithm - Density Field . . . . . 25
- 3.7 Brokerage System Architecture . . . . . 26
- 3.8 Auction Energy System - Login Page . . . . . 28
- 3.9 Auction Energy System - Register . . . . . 28
- 3.10 Auction Energy System - Dashboard of user with producer role . . . . . 29
- 3.11 Producer Dashboard - Graph of bids submitted to the system . . . . . 30
- 3.12 Producer Dashboard - Active Bids . . . . . 30
- 3.13 Producer Dashboard - Add a new bid . . . . . 31

3.14 Producer Dashboard - Bids Canceled . . . . . 31

3.15 Producer Dashboard - Bids Adjudicated . . . . . 32

3.16 Producer Dashboard - Energy Profiles . . . . . 32

3.17 Producer Dashboard - Add an Energy Profile . . . . . 33

3.18 Producer Dashboard - Settings . . . . . 33

3.19 Producer Dashboard - Change Password . . . . . 34

3.20 Consumer Dashboard - Survey . . . . . 35

3.21 Consumer Dashboard - Demand Profile . . . . . 35

3.22 Consumer Dashboard - Bids Available for Consumption . . . . . 36

3.23 Consumer Dashboard - Bids Adjudicated for Consumption . . . . . 36

3.24 Data Model . . . . . 37

3.25 Simulation model running in ControlDesk Software . . . . . 40

3.26 Communication between the broker system and dSPACE device . . . . . 41

4.1 Broker - Algorithm selection . . . . . 44

4.2 Algorithm - Generic FIFO . . . . . 45

4.3 Algorithm - Knapsack Problem Illustration . . . . . 48

4.4 Algorithm - Knapsack CO2 emission constraint . . . . . 49

5.1 Demand > Offer - Algorithms Time Running . . . . . 55

5.2 Demand > Offer - Energy Assignments . . . . . 56

5.3 Demand > Offer - Energy Assigned Vs. Energy Requested . . . . . 57

5.4 Demand > Offer - CO2 Emissions . . . . . 57

5.5 Demand = Offer - Algorithms Time Running . . . . . 58

5.6 Demand = Offer - Energy Assignments . . . . . 59

5.7 Demand = Offer - Energy Assigned Vs. Energy Requested . . . . . 60

5.8 Demand = Offer - CO2 Emissions . . . . . 60

5.9 Demand < Offer - Algorithms Time Running . . . . . 62

*LIST OF FIGURES*

*LIST OF FIGURES*

5.10 Demand < Offer - Energy Assignments . . . . . 62

5.11 Demand < Offer - Energy Assigned Vs. Energy Requested . . . . . 63

5.12 Demand < Offer - CO2 Emissions . . . . . 63

6.1 OASIS Future Work . . . . . 67

# List of Tables

- 2.1 CO2 Emissions of Energy Sources . . . . . 15
  
- 4.1 Bids submitted - FIFO . . . . . 46
- 4.2 Consumers - Daily demand . . . . . 46
- 4.3 FIFO Assignments . . . . . 47
- 4.4 Bids submitted - Knapsack . . . . . 50
- 4.5 Knapsack Assignments . . . . . 51
  
- 5.1 Big O - Types of Order . . . . . 53

# Listings

A.1	Broker - Execute method code in Job Element . . . . .	69
A.2	Broker - Trigger code . . . . .	71
A.3	Broker - Scheduler code . . . . .	72
B.1	Broker - Implementation of FIFO on the broker side . . . . .	73
B.2	Density Calculation . . . . .	75
B.3	Knapsack implementation code . . . . .	76



# Chapter 1

## INTRODUCTION

### 1.1 Motivation

Smart grids (SG) [1] have been proposed as a mechanism to modernize and facilitate the operation of energy grids in the presence of multiple third-parties vying to sell electricity and related services. In a smart grid, sensors, computers and communication networks are integrated into the equipment found at the power generation, transmission, distribution, and load levels. This enables a mechanism to gather information, update operational set-points, control generation of energy, control demand, diagnose problems, and forecast consumption.

With the development of the state-of-the-art of smart grid systems, new challenges have been opened for the management of information, among these challenges is the establishment of an energy market within the smart grid, establishing energy allocations optimally according to the present restrictions. For example, energy generation must always be matched to energy demand, since it is hard to store electricity.

Recently, a new concept has emerged in relation with SG and the establishment of energy markets, which is the concept of Transactive Energy (TE). TE makes reference to a framework to integrate and coordinate multiple independent suppliers of electric services in an economic marketplace that enables purchases in a way that obeys the physical

constraints of the underlying energy grids [2, 3, 4]. In other words, TE exhibits the definition of the smart grid evolved, with increased use of renewable energy generation and distributed energy management technologies.

Energy management is a major concern in smart grid environments. In the past several years, researchers addressed this issue by incorporating the implementation of different components such as Home Energy Management System (HEMS) [1], building energy management system, dynamic pricing, and load shifting. Currently, there are software tools that address different needs in smart grids in the areas of energy management, information management and security. There are only a few software that address the issue of establishing energy markets, these few have at least one of the two fundamental problems. The first is that there is no bidirectional communication between the software and the devices of the electric network, there is a need to integrate a common platform with the smart grid. On the other hand, some of the software are not designed to work cloud-based infrastructure.

The aim of this research project is to define a cloud-based brokerage system for SG, which will enable dynamic interaction between energy consumers and producers. Consumers establish a demand profile through a survey from which the daily supply of energy will be estimated. Producers submit production profiles, each production profile refers to a source of energy that is physically available and from which they can sell energy. The broker performs the energy allocations according to two algorithms. The first algorithm only takes into account the price minimization and the order of arrival of the clients. The second algorithm seeks to minimize the price, but subject to certain conditions on CO<sub>2</sub> emissions to the environment. Researchers at the University of Puerto Rico, Mayagüez in the field of social sciences, computer engineering and electrical engineering are working on developing a transactive energy framework called Open Access Smart Grids Services (OASIS) [5, 6, 7]. In OASIS, we modeled the smart grid as a col-

lection of interdependent electric and cloud services. These services can be dynamically purchased and combined to establish service dependencies between energy producers and consumers, just like people do with computational services (e.g., disk, virtual machines) in a cloud computing platform. Moreover, OASIS features distribution-level marketplace where the underlying electric resources and services are open to access and provisioned by third parties, maximizing the benefits of SG. This can ensure a set of redundant and independent providers of electric and cloud services, some of which might be common citizens whose home has a solar or wind turbine system. The latter aspect makes sustainable energy sources be first-class elements in the system. In addition, it has important societal dimension as it empowers common citizens, especially those living in vulnerable or underserved communities, to become key actors in a sustainable energy market and sway prices and infrastructure development in their favor [8].

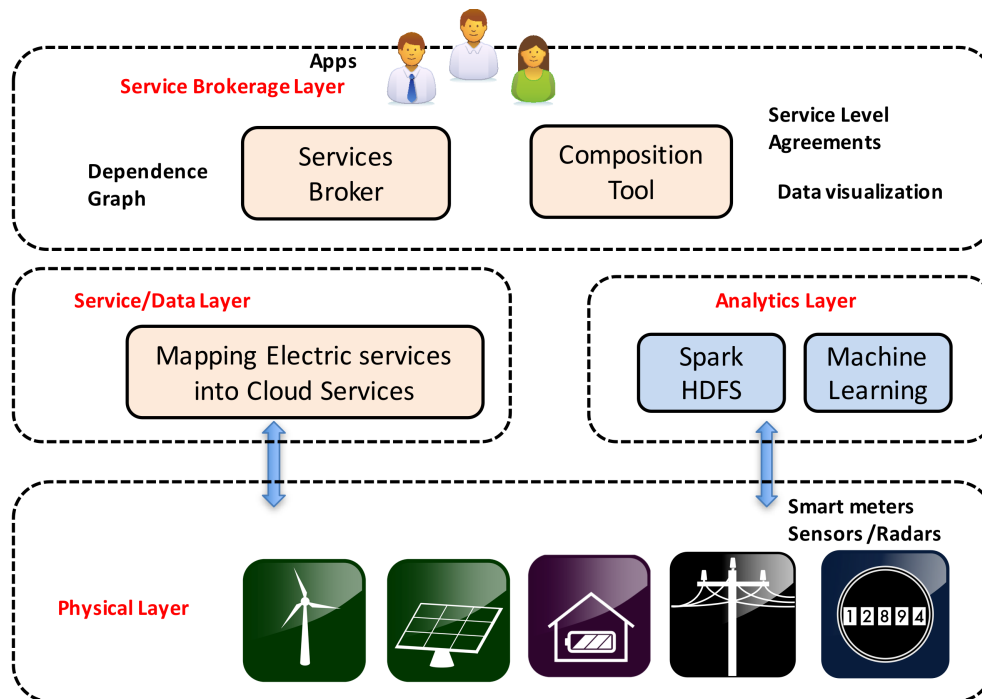


Fig 1.1: OASIS System Architecture [5, 7]

Our current effort to realize OASIS is based on Infrastructure as a Service (IaaS) clouds that provide the building blocks to implement a) an analytic layer to monitor and

characterize the system, and **b)** implement a Software as a Service (SaaS) that enables the exposure of electric services as REST-based cloud services. This is illustrated here in figure 1.1. As we can see from the image, the Physical Layer represents the electrical assets that must be exposed. These are wrapped around a set of RESTful web services.

The Analytics Layer takes care of collecting and monitoring information from sensors in the equipment and other data available to understand the configuration, current state, and operational set points in the system. In addition, this layer provides the machine learning functionality necessary for predictions and other types of detailed analysis. This layer is being implemented with Apache Spark. The stream of sensor readings will be managed with Apache Kafka and fed for processing into the Spark Streaming engine.

The Service Brokerage Layer implements the functionality of a transactive energy marketplace and represents the main subject of the rest of this document.

## 1.2 Objectives

- **Describe the architecture of the energy marketplace and brokerage system in OASIS:** we propose a software approach for a marketplace that combines a web application, stand-alone procedures, communication and storage of information. In this way, it is necessary to describe the design of those components and how they can interact through an architecture.
- **Present the implementation of the marketplace based on web services:** after identifying and describing the elements required in the system, it is necessary to use appropriate technologies that can translate the design and logic of the architecture into source code, this objective refers to the functionality in the server side.
- **Showcase the web applications used to interact with the marketplace and brokerage system:** through this objective we exhibit the tool that is visible for

the users (consumers and producers), so they can manage their profiles.

- **Design the algorithms used to establish the energy assignments into the broker system:** once consumers and producers submit their requirements and their energy offers respectively, the broker component is responsible for making the respective energy allocations through algorithmic logic based on criteria of price minimization and CO2 emission constraints.
- **Explain how the market interacts with a hardware-in-the-loop simulation of an energy grid:** the energy allocations made by the broker are at a high level, to relate sellers to buyers but at the level of the physical layer (low level), the energy exchanges happen in a different way according to physical restrictions. Through this objective we evaluate the assignments made by the broker (high level) in a simulation of the energy grid running at a hardware-in-the-loop (low level) to corroborate that those assignments are really viable.

### 1.3 Contributions

- **A novel implementation in the transactive energy area:** the implementation of the market platform helps to give validity and support to the TE concept. In this platform you can generate the complete workflow of buying and selling energy among consumers, producers and the broker. There are other projects related to TE but in this we focus on market at the cloud level.
- **Developed apps to facilitate explicit interactions between each one of the market participants:** in the current environment of the energy market, the relationship that exists between the producer and the consumer is the bill. In this project, we give them a web platform in which they can have a closer follow-up and interaction, know how they are selling and buying energy.

- **Implementation of the broker system with algorithms for energy allocations:** the broker system is the heart of the operations, all the information submitted by consumers and producers is processed to make the energy assignments by it. We provide two algorithms to make those energy arrangements. The first one takes FIFO theory as basis, the second one implement an approach of the Knapsack algorithm.
- **Communication between market and hardware-in-the-loop:** we provide a communication platform for the market place implemented with a hardware-in-the-loop component. We left the communication open to replace this hardware with a real environment, a real Smart Grid.

#### 1.4 Outline

The outline of this document is as follows. Chapter 2 gives a brief description of the electric networks, energy markets and transactive energy framework that are the environment of application of this project. The step by step process for the completion of the objective is presented in Chapter 3. The presentation of the algorithms used in the broker module is contained in Chapter 4. The Chapter 5 shows the comparative results between the two algorithms used, at the level of their structure and performance. Finally, Chapter 6 gives the conclusions and the direction for further development.

# Chapter 2

## LITERATURE REVIEW

This chapter describes important concepts about energy markets and electric networks. We also point some projects in the field of transactive energy. Finally, we exhibit a section that describe briefly the different kind of energy resources and the importance of renewable energy in nowadays.

### 2.1 Electric Networks

Electric energy networks provide the energy necessary to carry out daily operations in education, health care, commerce, entertainment, defense, and government. Typically, these energy networks are organized along four major components, those components are illustrated in the figure 2.1:

- **Generation:** the set of facilities where the actual generation of electricity occurs.
- **Transmission:** high-voltage wires, substations, and other facilities that work together to move electricity across long distances (e.g., across state lines).
- **Distribution:** the wires, substations, and other equipment that moves electricity from the transmission system into the cities, factories, buildings, residential areas, etc.

- **Loads:** the set of homes, buildings, offices, factories, and other sites that consume electricity.

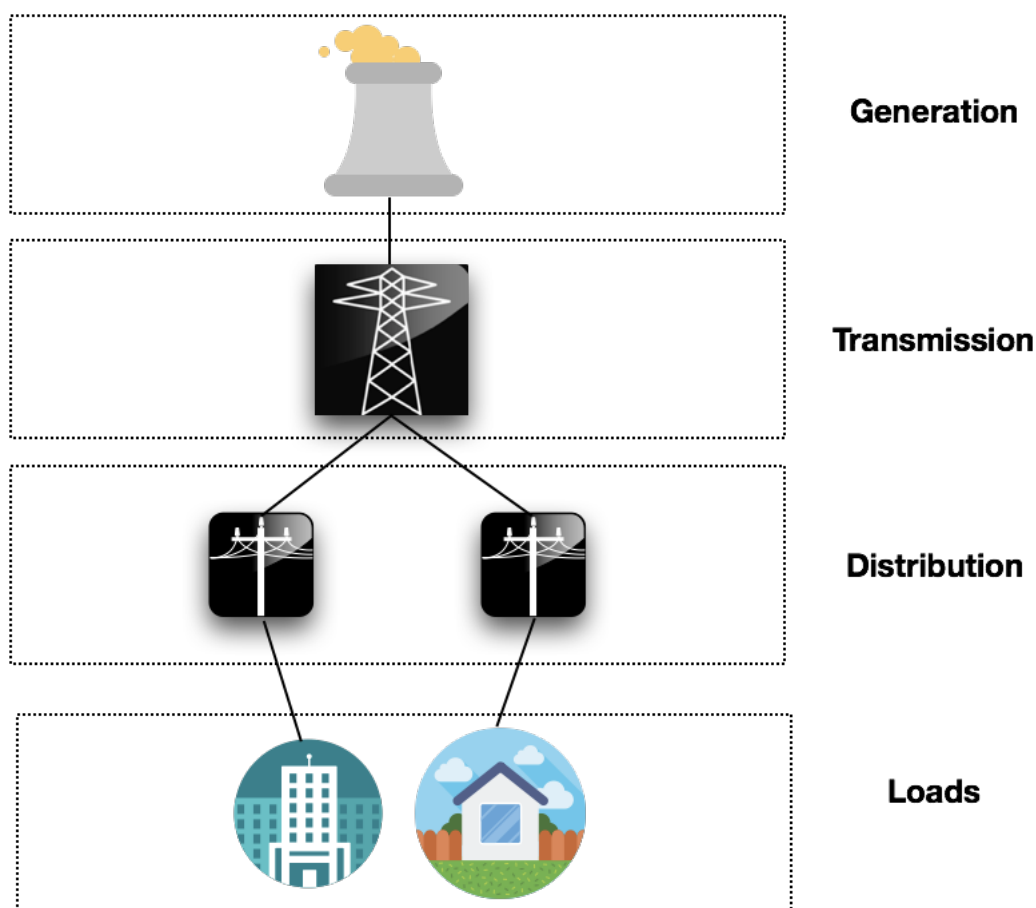


Fig 2.1: Traditional power system structure.

In most countries, a monopoly has been the corporate entity that owns and operates the energy grid. However, many jurisdictions in North America and Europe have deregulated their energy markets, allowing multiple independent companies to sell electricity and other ancillary services at the generation and transmission levels. Thus, a given region might receive energy from independent generators who do not own the transmission grid, but instead get paid to inject electricity into the transmission grid owned by some other company. The main goal of this scheme is to reduce energy prices by promoting competition between energy companies.



The scheme explained before is related to Distributed Generation (DG). The Institute of Electrical and Electronics Engineers (IEEE) defines distributed generation as the generation of electricity by facilities that are sufficiently smaller than central generating plants in such a way to allow interconnection at nearly any point in a power system [9]. From the middle of the 1980s onwards, the major development in energy supply and consumption has been the splintering of central power grids and the simultaneous emergence of regional decentralized configurations. This trend opens the gates to the use of renewable energies at each point of distributed generation. Applying renewables, in particular in the electricity supply, has become a pressing issue and most renewable energy units must be considered forms of DG. Together with improvements that serve efficiency and reliability, a system with a large amount of DG is considered an environmentally friendly alternative to the traditional power supply system [8].

## **2.2 Energy Markets**

Energy markets are commodity markets that deal specifically with the trade and supply of energy. The transactions of the power market can be divided into three types: the spot transaction, forward contracts transaction and the future tradings [10]. Electricity spot transaction generally refers to electricity trading one day before (or even 1 hour) of actual usage. Its main features are real-time or short-time quotation, real-time transaction, frequent price fluctuation, and sometimes greater price volatility, its scenario is illustrated in figure 2.2. On the other hand, forward contracts transactions are used to complete the electricity trading at some time in the future by signing long-term contracts with specific prices or price ranges as illustrated in figure 2.3. The electricity futures trading is one type of the forward contracts trading. The transaction objects to negotiate are the electricity futures contracts themselves, this is depicted in figure 2.4.

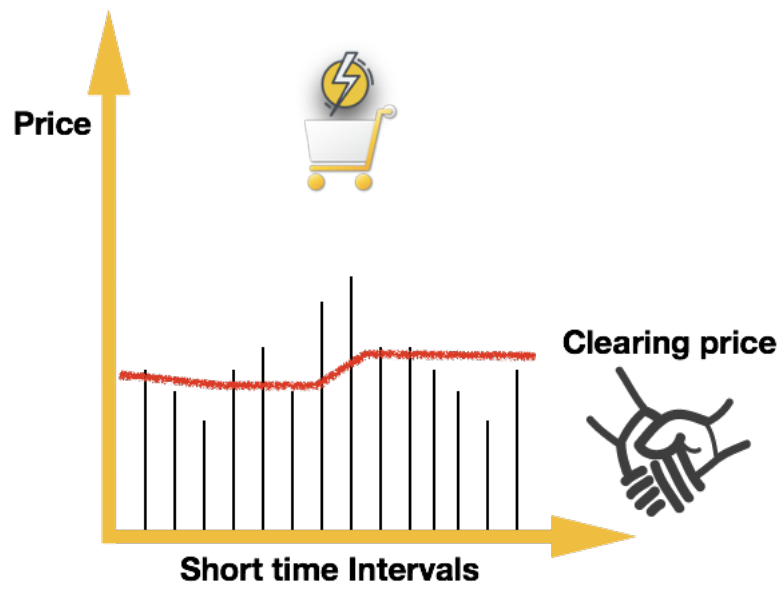


Fig 2.2: Spot transaction.

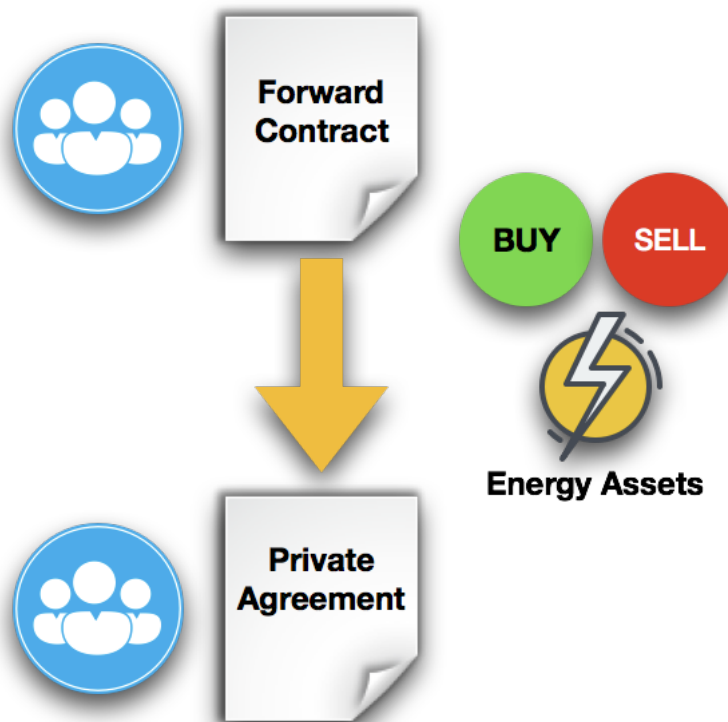


Fig 2.3: Forward contracts.

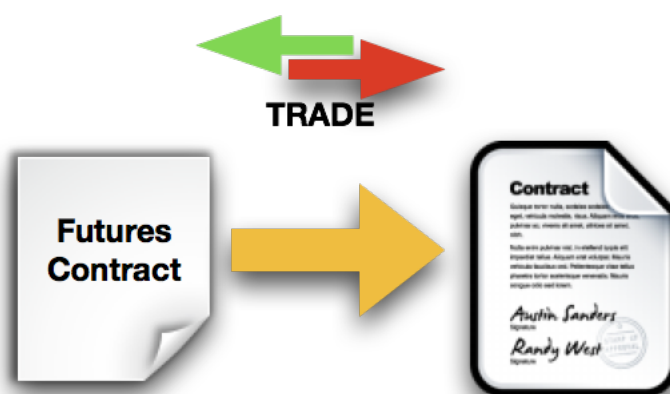


Fig 2.4: Future trading / Future contracts.

Fundamentally, forward and futures contracts have the same function: both types of contracts allow people to buy or sell a specific type of asset, in this case energy at a specific time and at a given price.

### 2.3 Transactive Energy

Transactive Energy is a method to perform transactions involving purchases of energy between consumers and producers. The GridWise Architecture Council (GWAC) has defined Transactive Energy (TE) as “A system of economic and control mechanisms that allows the dynamic balance of supply and demand across the entire electrical infrastructure using value as a key operational parameter” [11, 12]. In other words, the TE approach offers a way for producers and consumers to more closely match and balance energy supply and energy demand. If energy providers and users who are requesting energy can agree on the value of electricity at a certain point in time and place, then the producer and consumer can each make a decision if they want to proceed with the transaction at that given price.

The topic of TE has received more and more attention since 2015. For example, it has been a part of the New York Reforming the Energy Vision discussions and the topic of activities such as the National Institute of Standards (NIST) TE Challenge. TE

systems are a promising approach for facilitating internet-enabled open markets. In such markets, customers and grid systems can negotiate the proper way to address energy transactions, and settle on the proper price for energy services, all done close to real time of consumption. TE systems could also facilitate the integration of large numbers of Distributed Energy Resources (DERs), many of which will be owned by customers.

TE systems have their roots in concepts proposed in the 1970s and early 1980s by Fred Schweppe and his colleagues at MIT in an area they came to refer to as “homeostatic control”. In a 1978 article titled “Power Systems 2000”, Schweppe discussed hierarchical control strategies for the electric power system. He was anticipating the use of dynamic pricing that is included as an element of most of the TE approaches today. In 1980, the MIT team defined homeostatic control as “an overall concept which tries to maintain an internal equilibrium between supply and demand. Equilibrating forces are obtained over longer time scales (5 min and up) by economic principles through an Energy Marketplace using time-varying spot prices” [13]. Therefore, the TE concept offered by GWAC is very consistent with the concept of homeostatic control. The main drawback, at the time the concept of homeostatic control was incorporated, was the lack of necessary technology to implement it effectively, however, it is possible nowadays given the state of the grid, computational and physical resources.

### **2.3.1 Transactive Energy Projects**

Nowadays, there have been several initiatives that promote landing the concept of transactive energy to implementation. For example, in June of 2016, Con Edison filed their Distributed System Implementation Plan with the Public Service Commission [14]. This plan is their comprehensive roadmap to achieving their future vision for the Distributed System Platform. Other projects in the area are: GridWise Olympic Peninsula [15] Demonstration Project in Washington State, the AEP Ohio gridSMART [16] Demonstra-

tion project in Columbus, Ohio, and the PowerMatcher projects [17] completed by the TNO, the Netherlands Organization for Applied Scientific Research. These three last projects used a double-auction market in which both suppliers and loads submit bids. Consumers who participated in the real-time market submitted demand price bids for the expected power to be used by them. On the other hand, the one generator that was able to run in parallel with the power grid always submitted bids for the maximum nameplate generation capacity it could supply. The local market determines the amount of power available for the coming market time period and closes the market based on clearing the bids to assure that no more than that quantity is consumed. Bids are offered and the market cleared for each market period. This approach does not anticipate the future beyond the next market interval [13], this means each market space is independent of another, the behavior of what happens in this interval can not ensure the next one will be carried out in the same way or similar. The trend in those projects is the use of agents to control electric devices to maintain an effective control supply-demand according to an economic signal.

The Pacific Northwest Smart Grid Demonstration [18] implemented a transactive system based on the transactive control concept formulated at Pacific Northwest National Laboratory (PNNL). Under transactive control, decision-making is distributed across the grid, even to consumers and individual devices. This is accomplished via a seamless, two-way communication method that uses signals containing information about the delivered cost of electricity and the amount of power needed by end users. In other words, it will be a system that will tell the consumers when energy is cheapest, in that way, consumers and power authority can work together to save energy and keep prices low. At the end, they expect to move the region and nation closer to establishing a more efficient and effective electricity infrastructure that is expected to help contain costs, reduce emissions, incorporate more wind power and other types of renewable energy, increase power grid

reliability, and provide greater flexibility for consumers.

## 2.4 Energy Resources and Emissions

Due to all human activity, the planet has increased the concentration of carbon dioxide (CO<sub>2</sub>) in the atmosphere by around 31% in the past two centuries [19]. Energy resources play an important role in the future of the world and it is necessary to find ways to reduce the amount of emissions of Greenhouse Gases (GHG), specially CO<sub>2</sub>.

Energy resources have been divided into three categories: fossil fuels, renewable resources (alternative sources of energy), and nuclear resources [20]. One of the measures that can contribute most to the reduction of GHG in the environment is the change from fossil fuels to renewable resources, it impacts especially the reduction of CO<sub>2</sub> since around 98% of carbon emissions result from the combustion of fossil sources while renewable energy sources have the potential to provide energy with zero or near zero emissions of air pollutants and GHG [21, 22, 23].

Renewable resources produce pollution during their manufacturing processes, and subsequent recycling, but not during their operational time [23]. On the other hand, even when they generate pollution through their life cycle, since the benefits that these alternatives provide outweigh that they are considered to be an attractive environmentally friendly alternative to generate power. For example, Photo-voltaic Systems (PVS) contribute GHG emission during their manufacture phase, specially in the transformation phase from metallic silicone to solar silicone. But once, in operation, PVS produce little pollution.

In the work carried out by Stoppato [24], there is a table containing a summary of CO<sub>2</sub> emissions per country. In this table, the emissions of CO<sub>2</sub> by renewable and nuclear sources have an average of 0 kg / kWh. It is important to mention it to support the values of CO<sub>2</sub> emission of this energy sources.

The table 2.1, obtained from [25] shows the CO<sub>2</sub> emission per kWh for several certain energy sources, we complemented this table with the values of renewable and nuclear sources supported by [24].

Energy Source	kg CO <sub>2</sub> e / kWh
Solar	0
Wind	0
Nuclear	0
Wood pellets	0.039
Natural gas	0.1773
Liquefied petroleum gas (LPG)	0.214
Petrol	0.243
Burning oil	0.247
Diesel	0.253
Fuel oil	0.266
Gas oil	0.277
Industrial coal	0.313

Table 2.1: CO<sub>2</sub> Emissions of Energy Sources

This discussion on CO<sub>2</sub> emission and energy source is crucial since one of the algorithms proposed in Chapter 4 makes use of this information to carry out its operation.

# Chapter 3

## METHODOLOGY

In the OASIS (Open Access Smart Grids Services) Project, we propose an operational scheme where the underlying electric resources and services are open to access and provisioned by third parties, much like telecommunication lines are open to third parties. This scheme is illustrated in figure 3.1. In this model, the power lines are open to any vetted entity capable of producing energy or delivering ancillary electric services. Vetting will be done by the local power utility or some other agency that certifies the quality of the provider's electrical services. Third-party energy production most likely come from homes and buildings housing solar systems, wind systems, battery banks, electric cars, or emergency generators. These services include functions like billing, energy supply forecasting, energy demand forecasting, operating reserve generators, voltage control, frequency control, leasing battery banks, and regional weather forecasting, to name a few. This is the key innovation and transformative concept that we are pursuing.



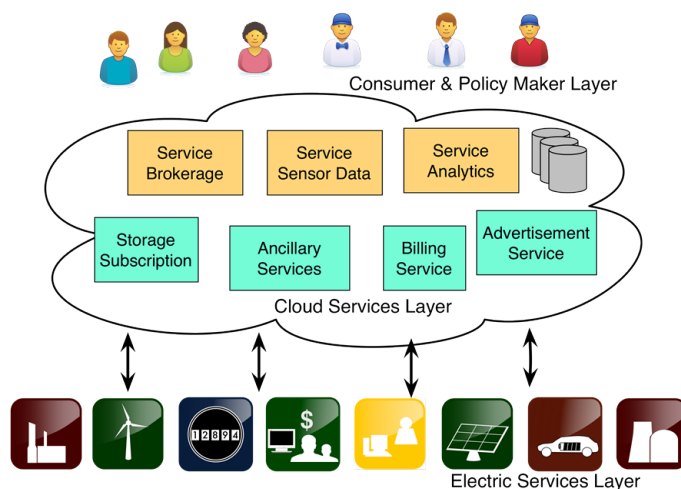


Fig 3.1: Electric services mapped to cloud services. [5, 7]

One analogy to our Smart Grid concept is the storage capability provided by Dropbox, where a person gets a cloud-based storage folder to keep documents readily accessible. Dropbox is mapping disk space on a server (physical computer resources) into a cloud service, seen by the user as a fail-proof folder located somewhere on the Internet. In the same fashion, a battery bank in a home can be mapped to an energy storage service available for lease. The benefit of this approach is that mitigating a generator failure could become as simple as switching from the current electric service provider to a different one, just like users of cloud services change virtual machines or move virtual disks around in response to hardware failures.

The cloud-based brokerage system for Smart Grids is a critical component of OASIS project. In this brokerage system, consumer energy requests, and energy bids issued by producers are collected and matched, to find a list of energy purchase orders to be dispatched to the winning producers. A central piece in the system is an energy broker software agent, which drives the bidding process. The broker uses the metadata in the system and also contacts the cloud services to establish consumer-producer relationships. The broker uses algorithms to link producers and consumers according to some established rules (e.g., minimizing prices, honoring requests in chronological order, or maximizing

renewable energy). After the relationships are established, the broker delivers the energy dispatch list to the producers so they can start injecting energy into the system.

This section describes the most important components of the brokerage system that was developed and defined for the proposed system.

### 3.1 OASIS Energy Market Overview

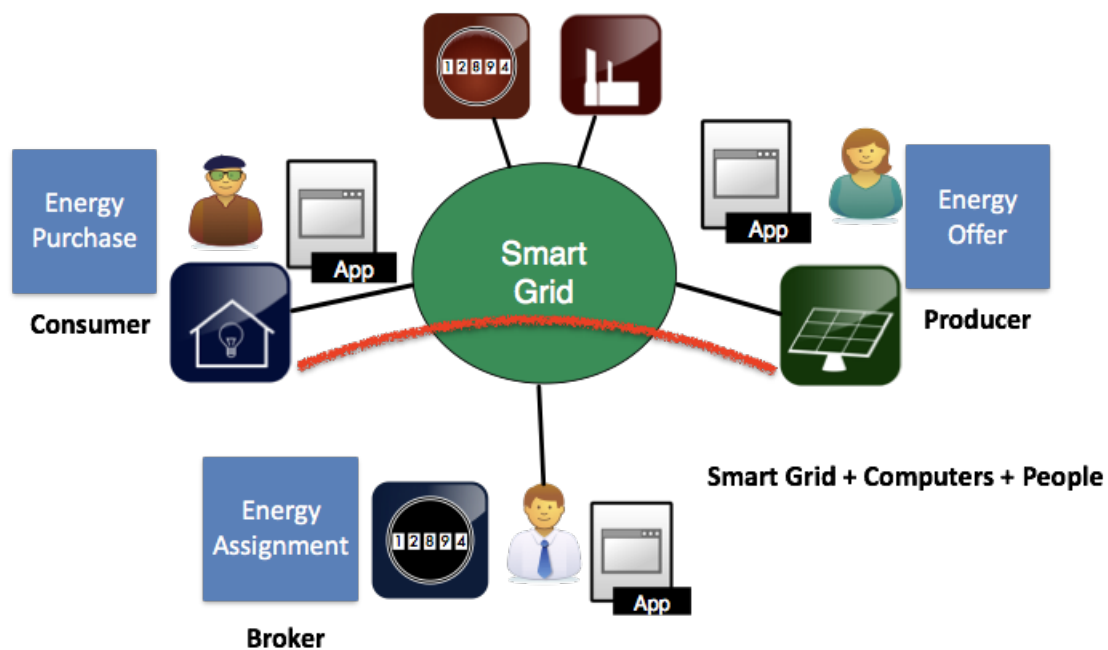


Fig 3.2: OASIS Energy Market Overview.

Energy markets are commodity markets that deal specifically with the trade and supply of energy. For this project, we highlighted three critical “actors” which take place in this kind of market: the consumer, the producer (energy supplier), and the energy broker. A *consumer* is defined as an entity that consumes the energy in the system; in the context of electric engineering, it is referenced as a load. On the other hand, a *producer* is defined as a market participant who has its own generation portfolio (e.g., solar panels, windmills, etc.) and sells electricity to the consumers. The third actor is the *broker*; it is a “middle-man” that matches the consumer demand with the offers submitted by producers

per some established instructions. The results of those matches are energy transactions that translate into energy dispatch requests to producers. The overview of this model is shown in figure 3.2

### 3.2 OASIS Consumer

Each consumer must declare a demand profile that provides a rough estimate of the daily energy consumption in the consumer's household. These demand profiles are derived through a survey that the consumer answers. We worked with four generic profiles initially:

- **Family with working adults and children:** there is little energy consumption between 8:00 am - 3:00 pm (none is at home). There is a consumption spike between 3:00 pm to 8:00 pm. The monthly average energy consumption for this profile is 830 kWh.
- **Family with stay at home adults and children:** there is consumption between 8:00 am to 3:00 pm, with a spike between 3:00 pm to 8:00 pm. The monthly average energy consumption for this profile is 900 kWh.
- **Family that works and does not have children:** there is little energy consumption between 8:00 am to 3:00 pm (none is at home). Consumption might not show a big spike and most likely starts at 6:00 pm. The monthly average energy consumption for this profile is 600 kWh.
- **Family with stay at home adults and no children:** there is consumption always, but there are no spikes. The monthly average energy consumption for this profile is 625 kWh.

The values in the generic profiles were obtained through simulations that represent

the demand profile of a community in Puerto Rico [26]. A graphic representation of those four profiles are showed in figure 3.3

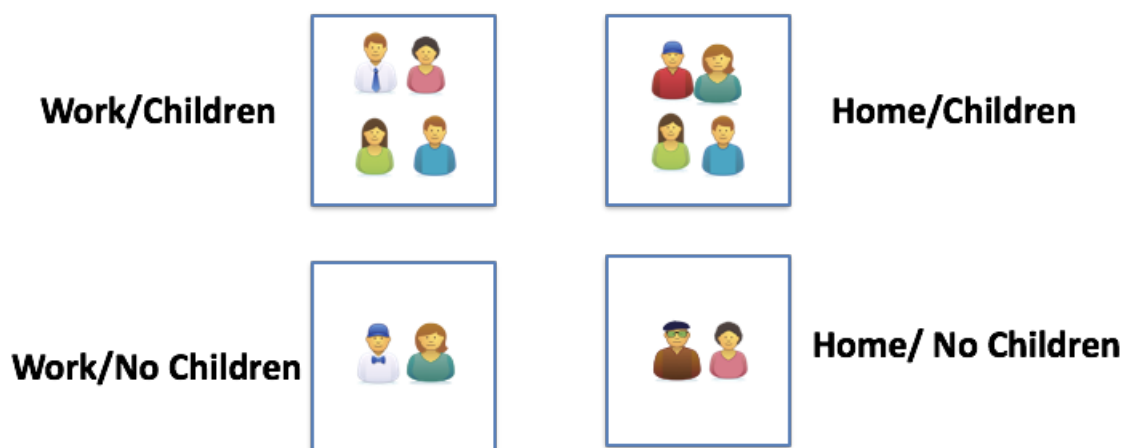


Fig 3.3: Consumer's Profiles

### 3.3 OASIS Producer

The producers can sell the energy that their sources can dispatch. A producer can have multiple sources of production, and the producers must create an energy profile for each one of them. Each energy profile requires to register the maximum capacity of the generation that can be produced and the kind of energy, it means if it is renewable energy or not. The process of selling is through submitting bids to the system, each bid is linked to an energy profile and they are composed of: 1) the size of the energy block, in terms of kWh, 2) the price per kWh, 3) and a range of time according to the availability of the resource. For instance, in a certain scenario a producer, according to the weather information and his/her experience, can submit a bid for tomorrow, linking this bid to his solar panel, and the energy that he/she is going to sell could be available only from 11:00 am to 1:00 pm. A bid may not be 100% accurate because the information is submitted directly by the producers. In a future work on analytics layer of the OASIS project, it is intended to apply machine learning techniques to make predictions and suggestions more accurate for producers.

An energy bid can have one of four states; these states are illustrated in figure 3.4:

- **In play:** it is the initial state of a bid once it is submitted. This state indicates the bid has been submitted successfully and it is available to the next auction process.
- **Cancelled:** this state indicates the bid has been discarded by the producer; this can happen when a producer refuses to sell some amount of energy or when they know it will not be available.
- **Accepted:** this is a state assigned by the broker once the bid and adjudication process has finished. If the bid was assigned to a consumer or group of consumers, this will be the tag that describes the bid.
- **Rejected:** once the energy allocation process is completed, those bids that were not linked to any consumer will be placed in this state. This usually occurs when supply is much higher than demand, or the producer is setting a very high price relative to the market price.

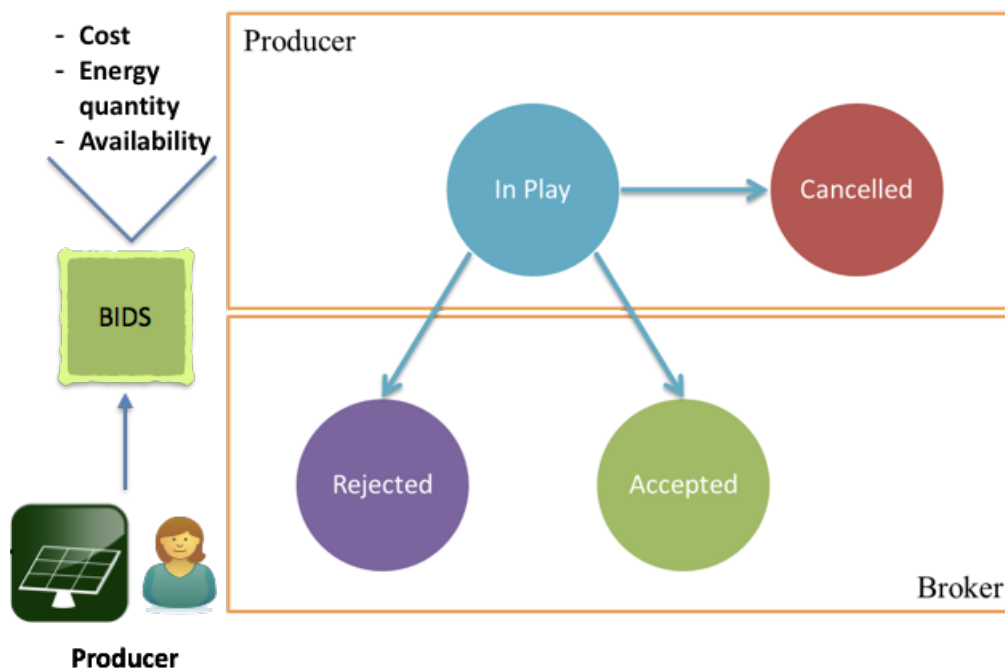


Fig 3.4: Bid's states

### 3.4 OASIS Broker

Energy purchases usually occur in advance to actual consumption, with time-frames ranging from days to a few hours ahead of time. In our case, we have developed the model of one day ahead purchasing. The market closes every day at 3pm. At this time, a scheduler component associated with the broker wakes up, and starts the processing of energy bids and assignments for the next day.

#### 3.4.1 Fifo Algorithm

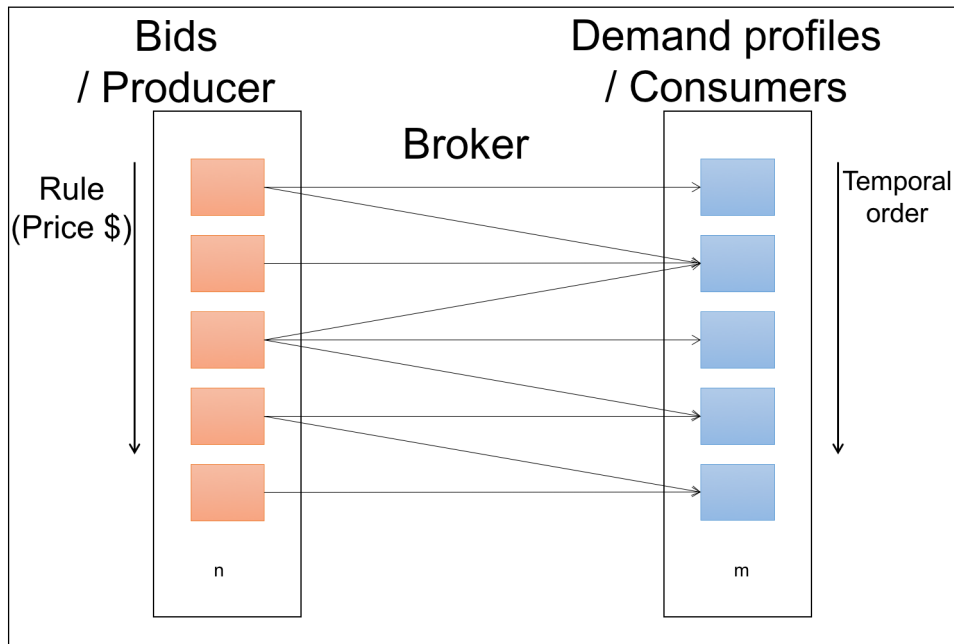


Fig 3.5: Broker's bid assignment scenario

The first algorithm of the broker will sort the bids offered by the producers in an increasing price. Alternatively, another sorting criteria can be used (e.g., energy type, emissions). Next, the broker sorts the client energy purchase requests in temporal order. At the time the market closes, the broker takes all the data about energy production submitted to the system for the next day, takes the consumption quantity and begins the process of

assigning the blocks of energy. The broker looks at each client's request and matches it to best available bid. Notice that some bids must be removed from the system as their energy quanta are consumed. Thus, a client's best bid might not be available because one or more clients already consumed it. In this case, the brokers will make a best effort to honor the best available bids. It might be possible, that none of the offers can be satisfied and the client gets the most expensive bid. This process is illustrated in figure 3.5.

In the scenario, where the energy offered by producers does not cover the entire demand, the missing energy will be requested from the grid. In the opposite case, where the supply is greater than the demand some producers may be left without selling energy.

For each energy order, the best bid available is mapped to the order. This process has some advantages, for example, it seems like a fair scheme from the perspective of the client, and it is simple to implement in software. But, its main disadvantage is that it might lead to sub-optimal allocation of energy blocks.

### 3.4.2 Knapsack Algorithm

The second algorithm takes the Knapsack theory as a basis. Knapsack problem has two variants:

- **0/1 Knapsack:** in this variant, items are indivisible; we can not break an item, we either take an item or not. This variation is solved with dynamic programming approach.
- **Fractional Knapsack:** in this alternative, items are divisible, we can take any fraction of the item. This version is solved with a greedy approach.

The Knapsack algorithm refers to a problem in classical combinatorial optimization: given a set of items, each with a weight and a value, determine the number (or fraction) of each item to include in a collection so that the total weight is less than or equal to a

given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size Knapsack and must fill it with the most valuable items. Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials, selection of investments and portfolios, selection of assets, and others.

We applied this algorithm to our case. Our assets are the bids submitted by the producers as a set of items, the weight of each bid is the amount of CO<sub>2</sub> emissions that will be produced by the energy source, and the value of each bid is the cost of 1 kWh multiplied by the number of kWh that the source can produce. The total weight of the Knapsack is the maximum quantity of CO<sub>2</sub> emissions that electric system operator can admit during the assignment process.

According to market environment, the broker can take a bid and sell only a fraction of it. For instance, a producer has 5kWh available to sell but the market only needs 2 kWh, hence the broker can take only the fractional amount that is necessary and make the arrangement. In this way, we use a greedy approximation algorithm with the fractional approach of Knapsack problem proposed by Dantzig [27]. His version sorts the items in decreasing order of value per unit of weight. In this formulation, value per unit weight is a ratio between the cost of the item and its weight. Then, it proceeds to insert them into the sack, starting with as many copies as possible of the first kind of item until there is no longer space in the sack for more, based on the maximum weight that a sack can resist. We implement a similar approach but items are sorted in increasing order, given that we minimize the cost of energy assignment. The sorting takes into account first, a field that we call density, which is the result of the division of the cost of each kWh per CO<sub>2</sub> emission for each kWh produced. If we let  $C$  denote cost per kWh,  $E$  denote



emission per kWh, then the density  $D$  is defined as:

$$D = \begin{cases} C/E & : E \neq 0 \\ 0 & : E = 0 \end{cases}$$

The second factor that takes into account for the sorting is the cost itself. An example of this sorting is depicted in figure 3.6. In the case that the emission of CO<sub>2</sub> from the source is 0 as the case of solar and wind sources, the density will be taken as 0.

Data Output					Explain	Messages	History
	block_size double precision	cost_per_kwh double precision	emission_per_kwh double precision	density double precision			
1	40	0.86	0	0			
2	50	0.89	0	0			
3	36	0.92	0	0			
4	5	1.1	0	0			
5	25	1.5	0.313	4.792332268370			
6	10	0.85	0.1773	4.794134235758			
7	13	1.3	0.266	4.887218045112			
8	30	2	0.313	6.389776357827			
9	10	10	0.266	37.59398496240			

Fig 3.6: Knapsack Algorithm - Density Field

## 3.5 OASIS Energy Market Implementation

### 3.5.1 System Architecture

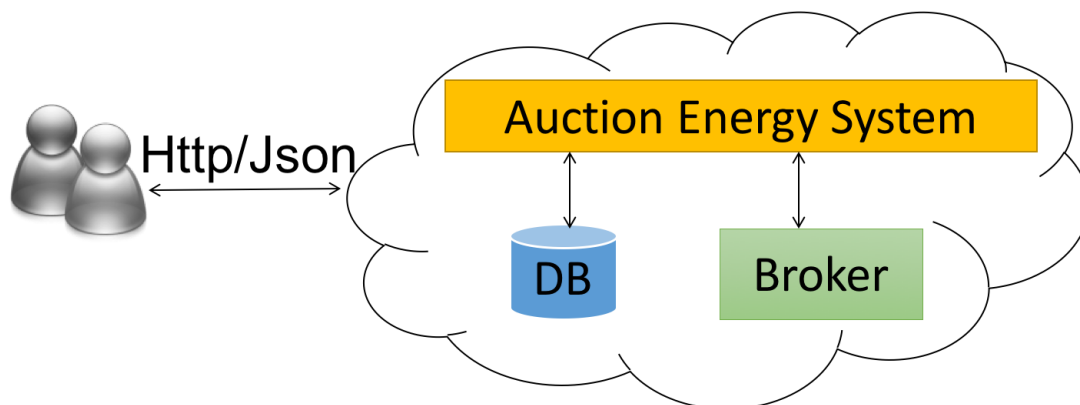


Fig 3.7: Brokerage System Architecture

The architecture of the brokerage system is depicted in figure 3.7. Supported client devices include smart-phones, tablets, laptops and workstations. All these devices communicate with the cloud services of the Auction Energy System (AES) via Hypertext Transfer Protocol (HTTP). This ensures simplicity in communication. The AES establishes communication with the central database and the broker system. All requests to the system are submitted as either POST or GET requests to a web application server. Results arrive encoded in JavaScript Object Notation (JSON).

The web application server, which contains the AES, was built using Java Play Framework v 2.5 [28]. Play is a high-productivity Java and Scala web application framework that integrates the components and APIs for modern web application development. Play is based on a lightweight, stateless, web-friendly architecture and features predictable and minimal resource consumption (CPU, memory, threads).

All the data and metadata related with the users, profiles of energy consumption, and production is persisted with PostgreSQL [29]. It is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven

architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems.

There is another component: the *broker*. This artifact is implemented as a daemon using Java. It contains the algorithms to make the assignment of energy between producers and consumers. The broker also incorporates the use of a scheduler that manages the exact time to close the market, and run the energy bid assignment process. The broker module is responsible to establish the connection to the physical layer mapping the high level energy transactions to the hardware-in-the-loop system. In the near future, the broker will issue the energy bid assignment to the underlying smart grid.

### **3.5.2 Web Interface**

The web interface is implemented with HTML, AngularJS and Bootstrap. We used responsive design to adapt the interface to different sizes of screens according to the devices used to interact with the application.

We use libraries like Angular Charts and ChartJS to exhibit some data in a graphic mode. Each call method is sent over HTTP to a specific Java Play controller for processing and obtaining the result from the server. The main screen is presented in figure 3.8, the form to create a new account is shown in figure 3.9.

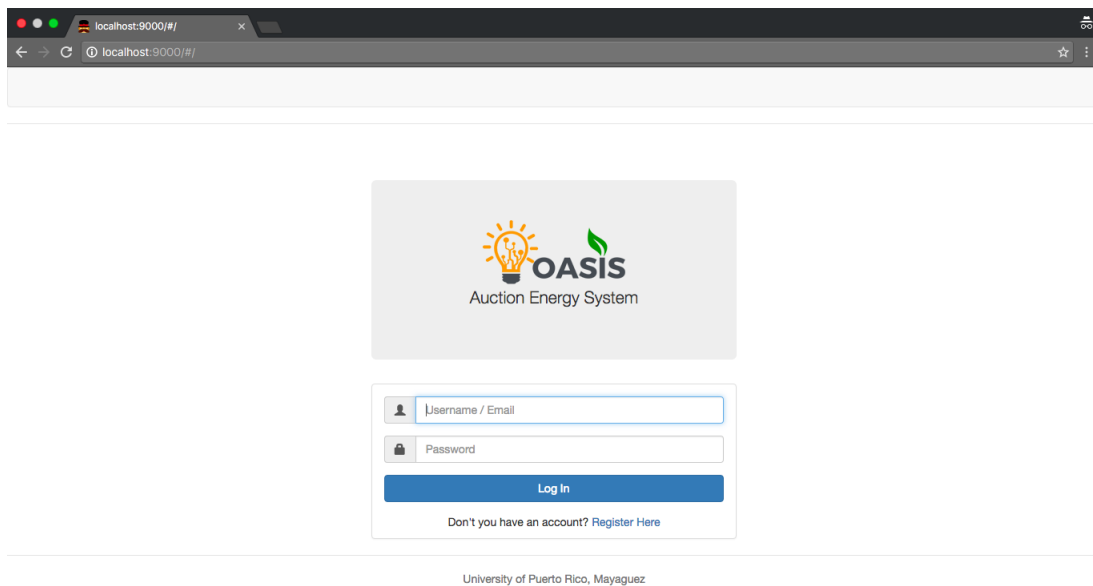


Fig 3.8: Auction Energy System - Login Page

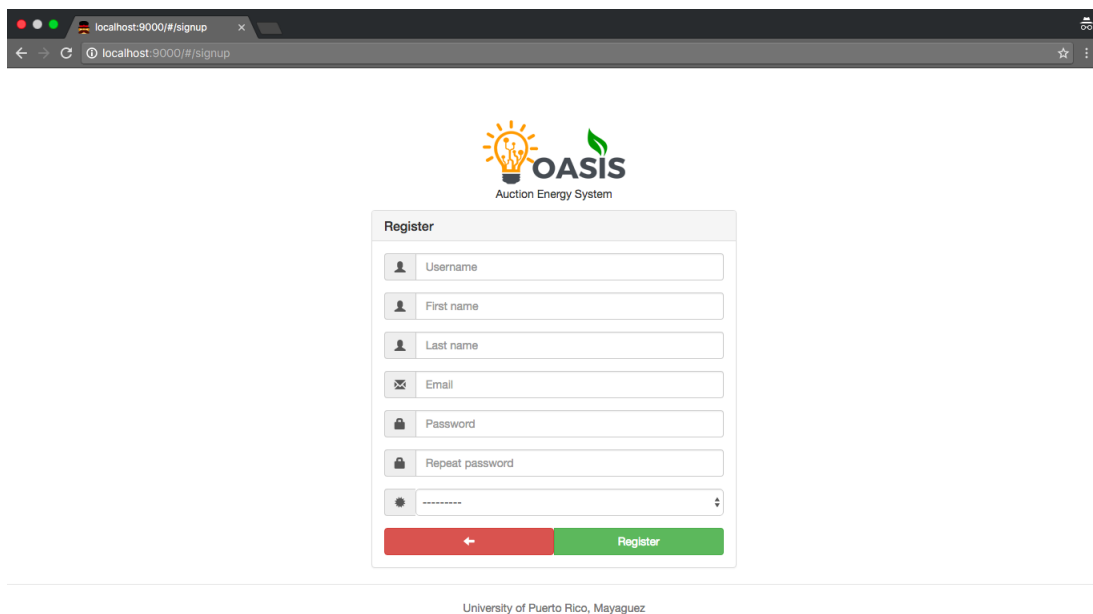


Fig 3.9: Auction Energy System - Register

## Producer

The dashboard for a user with the role of producer has different alternatives. The options in the menu of a producer are: Home, Bids, Energy Profiles and Settings.

**Home:** This option show the profile picture of the user, his/her description and the graph of the bids submitted to the system in the past 90 days, divided in 30-days intervals.

We illustrate this in the figure 3.10 and figure 3.11

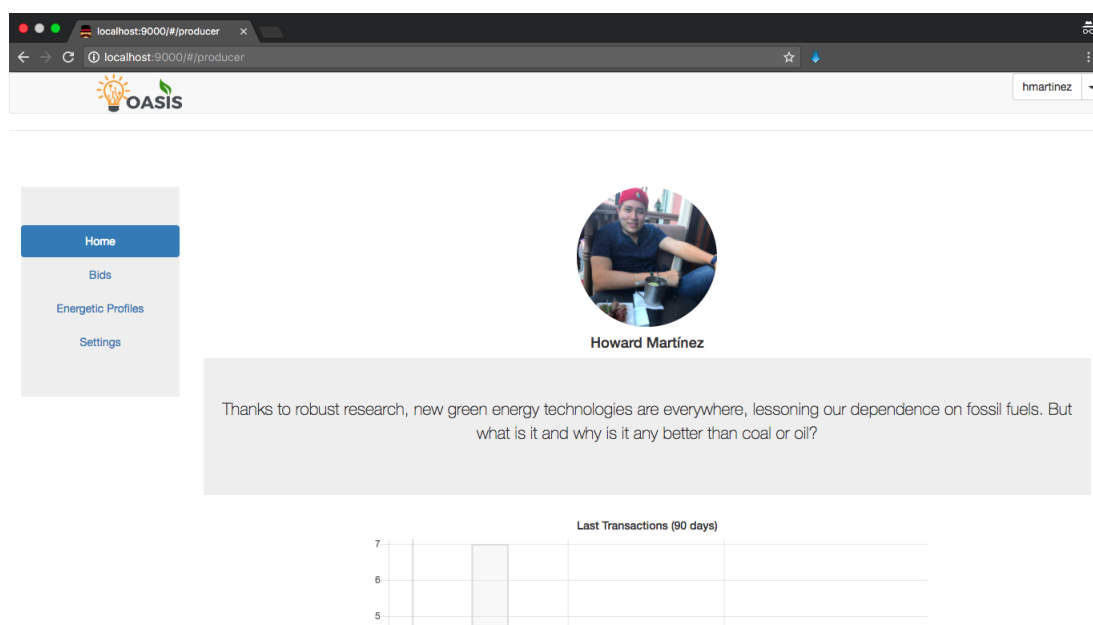


Fig 3.10: Auction Energy System - Dashboard of user with producer role

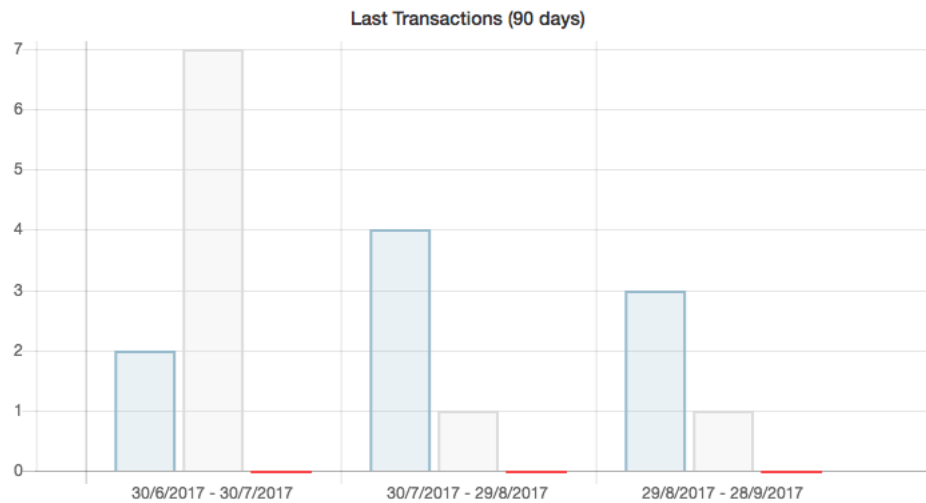


Fig 3.11: Producer Dashboard - Graph of bids submitted to the system

**Bids:** Through this option a producer can: 1) monitor the bids that he/she has submitted to the system, mainly the active bids (figure 3.12), 2) add a new bid for the next auction process (figure 3.13), 3) review the bids cancelled (figure 3.14), 4) and examine the bids that the system has adjudicated and the amount of money that he/she is going to receive from the sell (figure 3.15).

Bid ID	Block Size (kWh)	Start Availability	End Availability	kWh Cost (\$)	Cancel
fb772d2a-c57c-11e7-a884-07e00481c6fa	10	2017-11-10 10:00:00-04	2017-11-10 14:00:00-04	10	Cancel
03a64f88-bcd8-11e7-a987-6f2efdc3a567	13	2017-10-30 09:00:00-04	2017-10-30 13:00:00-04	1.3	Cancel
cc838874-9b29-11e7-bb01-df5ea1a2c683	10	2017-09-17 10:00:00-04	2017-09-17 14:00:00-04	0.85	Cancel
515c80b8-9707-11e7-afa4-77ad7a4daf0b	30	2017-09-12 15:00:00-04	2017-09-12 18:00:00-04	2	Cancel
388af0e2-9707-11e7-afa4-f3602840e68d	25	2017-09-12 11:00:00-04	2017-09-12 15:00:00-04	1.5	Cancel

University of Puerto Rico, Mayaguez

Fig 3.12: Producer Dashboard - Active Bids

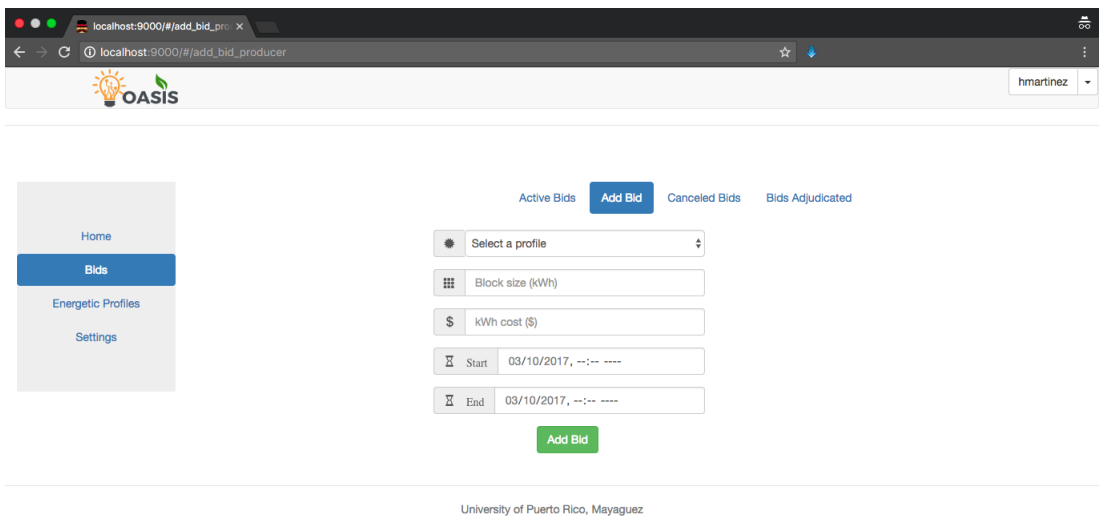


Fig 3.13: Producer Dashboard - Add a new bid

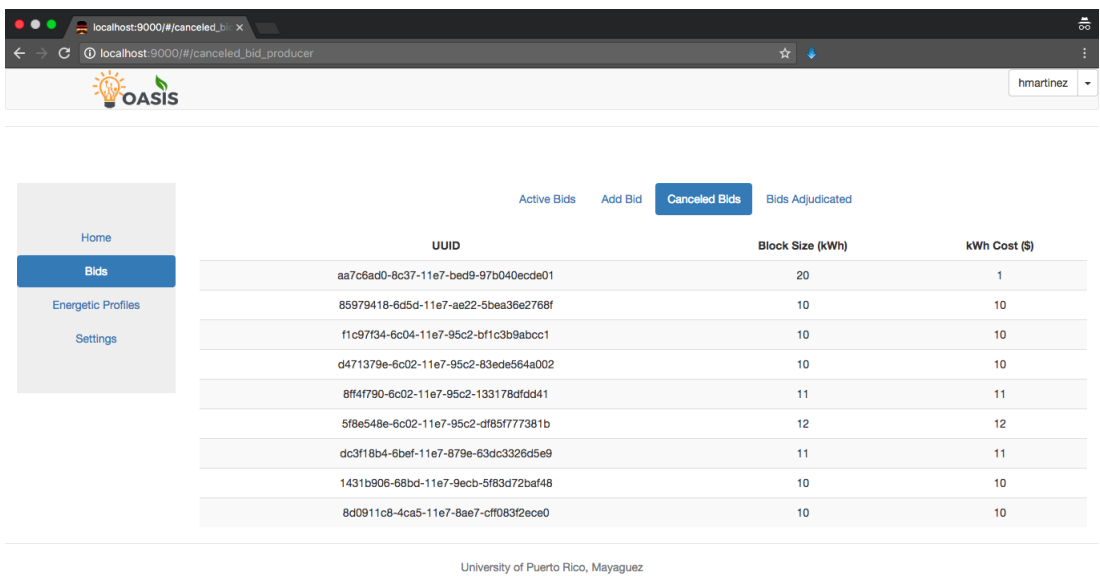
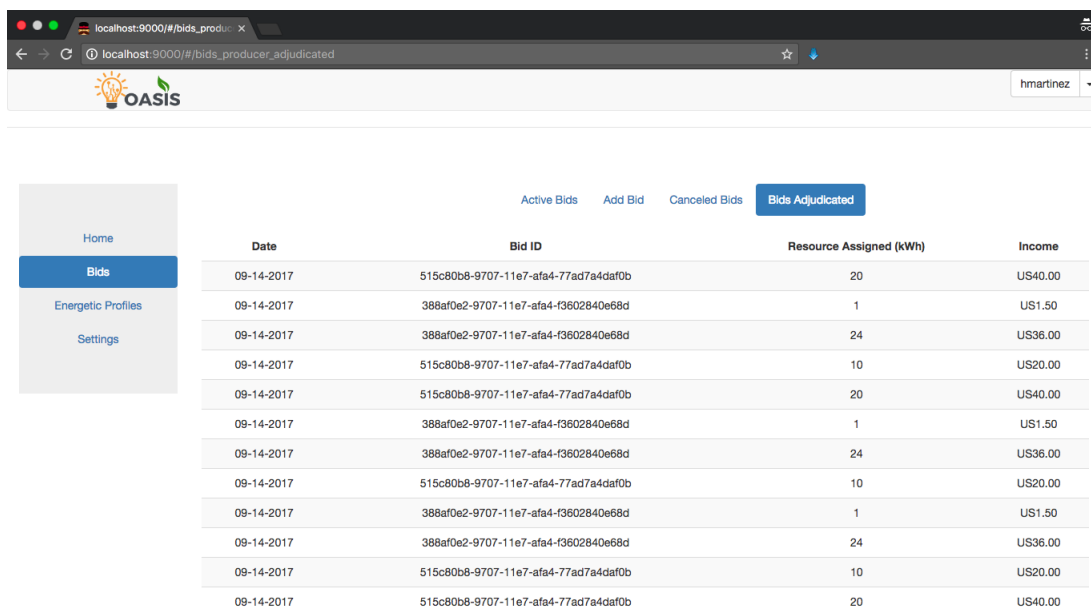


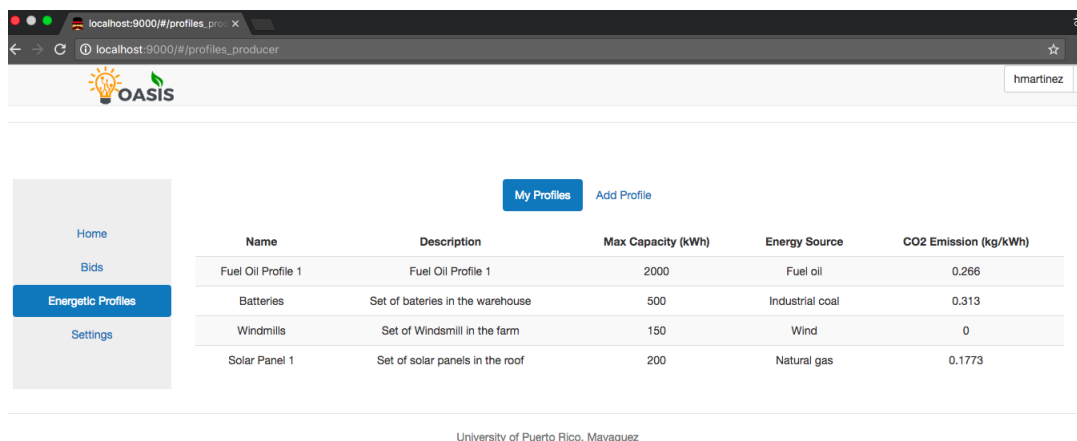
Fig 3.14: Producer Dashboard - Bids Canceled



Date	Bid ID	Resource Assigned (kWh)	Income
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	20	US40.00
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	1	US1.50
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	24	US36.00
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	10	US20.00
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	20	US40.00
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	1	US1.50
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	24	US36.00
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	10	US20.00
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	1	US1.50
09-14-2017	388af0e2-9707-11e7-afa4-f3602840e68d	24	US36.00
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	10	US20.00
09-14-2017	515c80b8-9707-11e7-afa4-77ad7a4daf0b	20	US40.00

Fig 3.15: Producer Dashboard - Bids Adjudicated

**Energy Profiles:** Through this option a producer can set and inspect the different profiles that he/she has created as a result of mapping of their real portfolio of energy sources in the physical layer (figure 3.16). The producer can add a new profile according to his/her available energy sources (figure 3.17).



Name	Description	Max Capacity (kWh)	Energy Source	CO2 Emission (kg/kWh)
Fuel Oil Profile 1	Fuel Oil Profile 1	2000	Fuel oil	0.266
Batteries	Set of bateries in the warehouse	500	Industrial coal	0.313
Windmills	Set of Windmill in the farm	150	Wind	0
Solar Panel 1	Set of solar panels in the roof	200	Natural gas	0.1773

University of Puerto Rico, Mayaguez

Fig 3.16: Producer Dashboard - Energy Profiles



localhost:9000/#/add\_profile\_producer

OASIS hmartinez

Home  
Bids  
Energetic Profiles  
Settings

My Profiles **Add Profile**

Select an energy source:

Name

Description

Max capacity (kWh)

**Add Profile**

University of Puerto Rico, Mayaguez

Fig 3.17: Producer Dashboard - Add an Energy Profile

**Settings:** This option is generic in both cases: producer and consumer. The users can update their personal information that includes their names, their descriptions and their profile pictures (figure 3.18). They can also change their password with the system. (figure 3.19).

localhost:9000/#/settings

OASIS hmartinez

Home  
Bids  
Energetic Profiles  
Settings

**Social Information** Change Password

**Save Changes**

Fig 3.18: Producer Dashboard - Settings

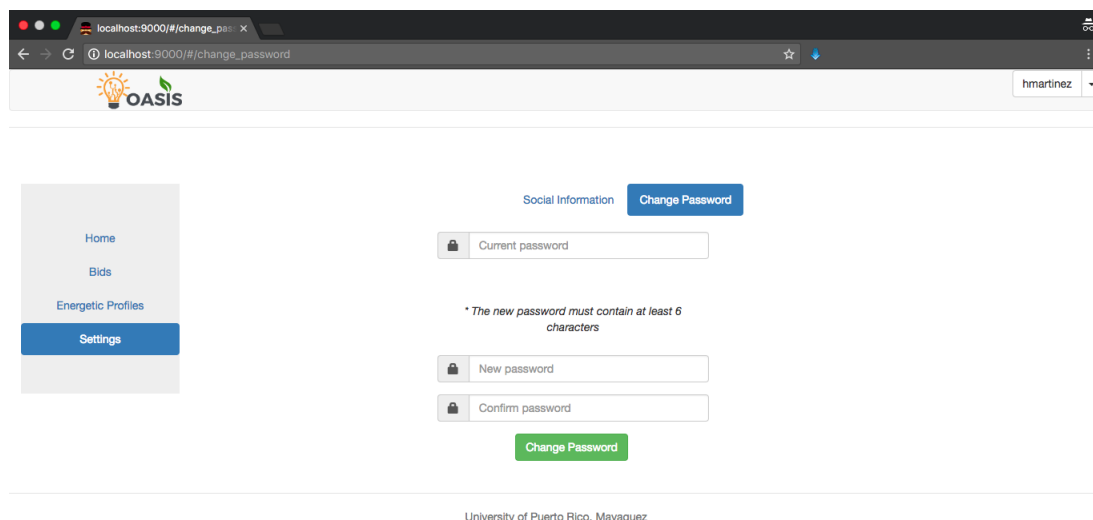


Fig 3.19: Producer Dashboard - Change Password

## Consumer

The consumer shares with the producer the same screens for login, settings, registration, and account management. The dashboard for a user with the role of consumer has different alternatives. The options in the menu of a consumer are: Home, Energy Market, Energy Profiles and Settings.

Each consumer must declare a demand profile that provides a rough estimate of the daily energy consumption in the consumer's household. To declare this demand profile, a consumer needs to answer a survey (figure 3.20). The system assigns the profile according to the answers received from the survey (Discussed in section 3.2) (figure 3.21). The consumers can see the bids available (figure 3.22) and the bids that has been adjudicated to their profile together with their associated cost (figure 3.23)

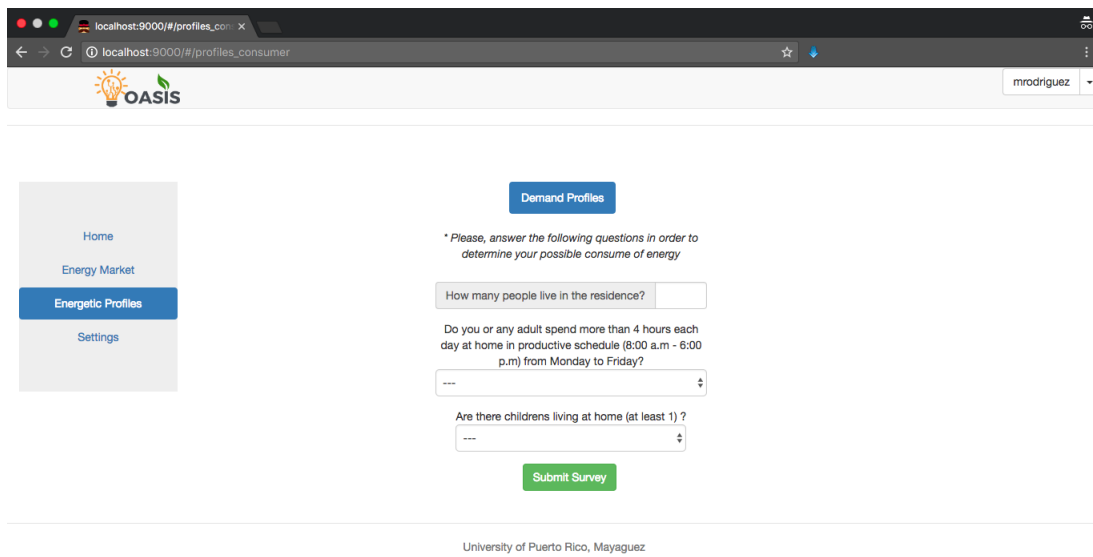


Fig 3.20: Consumer Dashboard - Survey

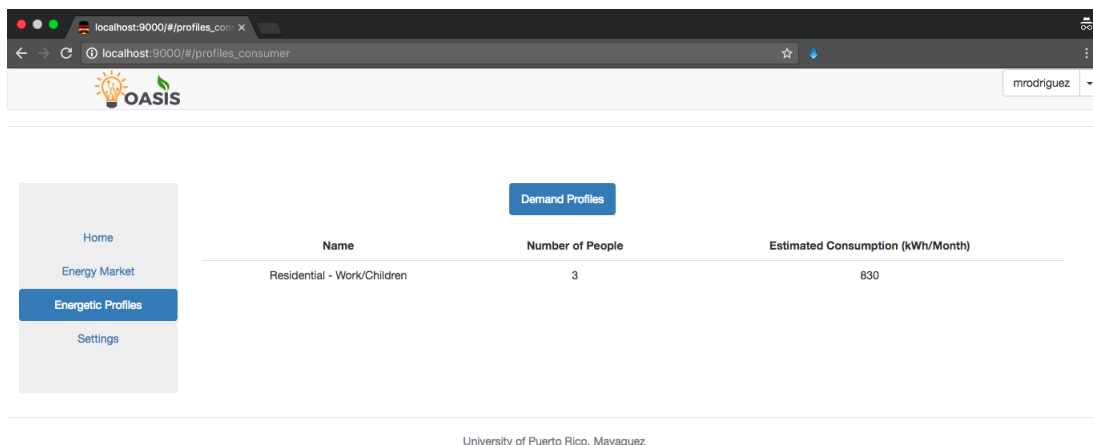


Fig 3.21: Consumer Dashboard - Demand Profile

Bid ID	Block Size (kWh)	kWh Cost (\$)
cc838874-9b29-11e7-bb01-df5ea1a2c583	10	0.85
7b2bb7ab-9715-11e7-9968-67c99c9428	40	0.86
012ca58c-9716-11e7-a399-c1e716696cd8	50	0.89
d05ea0ca-9707-11e7-af54-23306ca7dca8	36	0.92
79664c1e-9716-11e7-a389-a7e421d19301	5	1.1
03a6488b-bcc8-11e7-a987-6f2efdc3a567	13	1.3
388af62-9707-11e7-af54-036284d6e6d	25	1.5
515c80b8-9707-11e7-af54-77ad7a4da0fb	30	2
fb77202a-c57c-11e7-a884-07a00481c6fa	10	10

University of Puerto Rico, Mayaguez

Fig 3.22: Consumer Dashboard - Bids Available for Consumption

Date	Bid ID	Resource Assigned (kWh)	Cost
11-22-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-22-2017	fb77202a-c57c-11e7-a884-07a00481c6fa	10	US\$100.00
11-14-2017	fb77202a-c57c-11e7-a884-07a00481c6fa	10	US\$100.00
11-14-2017	012ca58c-9716-11e7-a399-c1e716696cd8	14	US\$12.46
11-14-2017	d05ea0ca-9707-11e7-af54-23306ca7dca8	14	US\$12.98
11-14-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-12-2017	cc838874-9b29-11e7-bb01-df5ea1a2c583	9	US\$7.65
11-12-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	6	US\$12.00
11-12-2017	03a6488b-bcc8-11e7-a987-6f2efdc3a567	13	US\$16.90
11-12-2017	fb77202a-c57c-11e7-a884-07a00481c6fa	10	US\$100.00
11-12-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-11-2017	7b2bb7ab-9715-11e7-9968-67c99c9428	10	US\$8.60
11-11-2017	7b2bb7ab-9715-11e7-9968-67c99c9428	28	US\$4.08
11-11-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-11-2017	fb77202a-c57c-11e7-a884-07a00481c6fa	10	US\$100.00
11-11-2017	012ca58c-9716-11e7-a399-c1e716696cd8	18	US\$16.02
11-10-2017	012ca58c-9716-11e7-a399-c1e716696cd8	28	US\$4.92
11-08-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-08-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00
11-01-2017	515c80b8-9707-11e7-af54-77ad7a4da0fb	3	US\$6.00

Fig 3.23: Consumer Dashboard - Bids Adjudicated for Consumption

### 3.5.3 Application Data Model

The brokerage system needs to store data for the operations established in the AES, and the energy assignments arranged by the broker artifact. This data include users, energy production, energy consumption, profiles of user and so on. Figure 3.24 depicts the collection of relations that build the data model.

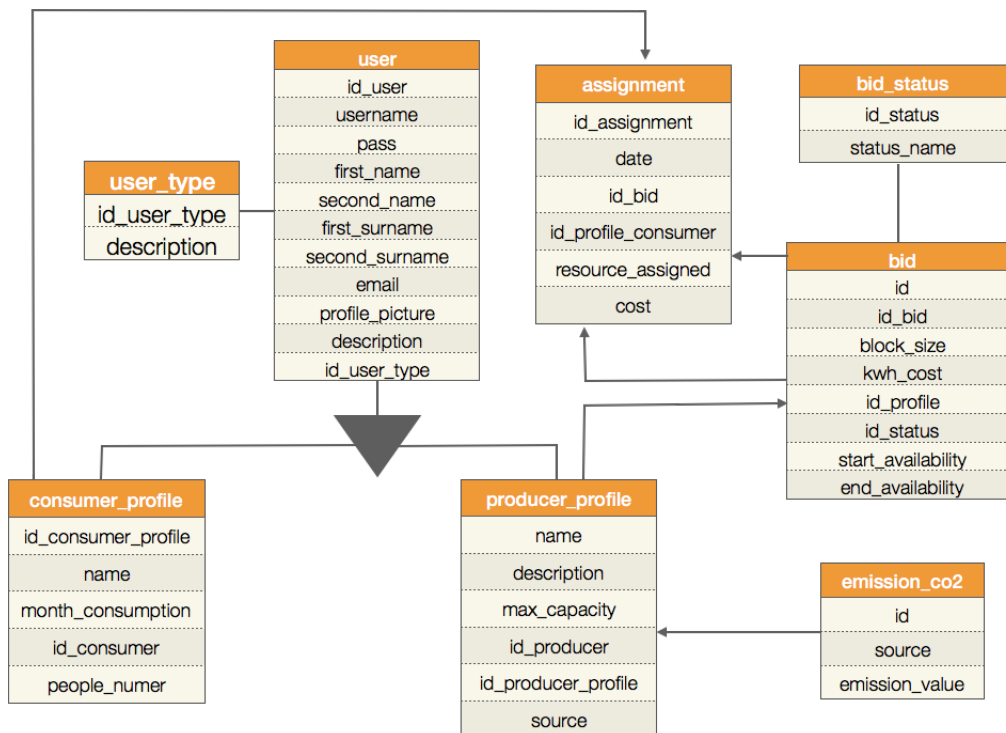


Fig 3.24: Data Model

The descriptions of these tables are as follows:

- **user:** this table has eleven columns; it stores information related to the user. For instance - username, password (encrypted), email and profile picture.
- **user\_type:** this table is to pre-configure the application, we insert directly to the database the kind of user and its description; we have two types of user: producer and consumer. A user with both roles is a prosumer.
- **consumer\_profile:** we establish consumer profiles according to a survey build with house demand profiles (Discussed in section 3.2); this table stores the name of the profile, month consumption, and the number of people that live in the residence.
- **producer\_profile:** a producer can have multiple sources of production according to his/her portfolio; each producer has to create an energy profile for each one.

Each producer profile (energy profile) requires the maximum capacity of the generator, and the energy source. The CO2 emissions for each source are paired with `emission_co2` table.

- **bid:** this table stores all the information related to the energy bids submitted by all the producers. Each bid has a unique identifier, called the Universally Unique Identifier (UUID). We also store the energy block size in terms of kWh, the price of each kWh and, the range of time that indicates the availability of the energy resource.
- **bid\_status:** each bid has a status. In our system there are four states for bids, which were illustrated in figure 3.4. This relation table stores the information about the status of the bid.
- **assignment:** this table stores the result of the energy transactions, linking the demand profile with the winning bids submitted in the system, and related energy cost and the date of selling.
- **emission\_co2:** each energy source has a certain amount of CO2 emission per kWh. This table stores different kinds of sources of energy with their respective CO2 values. We store the values presented in table 2.1 initially, this table can be continuously updated according to the values of emissions of different sources needed.

### 3.6 Energy Market Communication with Hardware-in-the-loop

Once the broker performs power allocations in the system between producers and consumers, the transaction data is communicated to a hardware-in-the-loop to carry out a simulation of the approved energy exchanges in test environment. In other words, the dSPACE provides a platform to simulate a smart grid, carry out the energy exchanges,

and observe the results in terms of load serviced, power flows, and power fluctuations. This hardware is called Modular Laboratory System from dSPACE company [30]. It is a modular, scalable, high-performance real-time system for rapid control prototyping, test, and data acquisition applications. dSPACE's modular laboratory systems provide high processing power, a wide range of I/O functionality and bus interfaces for various applications.

For effective communication between the broker system and the dSPACE device, both software communication and physical communication must be guaranteed. The physical communication is done between the dSPACE and a computer through a fiber optic cable, the computer to which the dSPACE is connected runs the ControlDesk software on which the model of the simulation is executed. This model in the ControlDesk program was previously created by our team of electrical engineers from the University of Puerto Rico. The simulation model represent a power grid interconnected with a number of producers and consumers that interact according to their generation capacities and scheduled consumption needs. The simulation receives the values of the transactions made in the broker, we establish the control of generation and consumption in the physical layer. The ControlDesk program enables the dSPACE to receive information in time intervals through data sequences in python. A view of the simulation model is depicted in figure 3.25. We worked with the one-day ahead model so we divided the day in 24 hours intervals, each interval is a hour. This model simulates 10 generators and 10 loads, the rectangles in the image labeled since DG01 until DG10 represent the generators and the numbers below them indicate the amount of kW that they need to generate in that specific interval. For example, we can say from the image that generator DG01 need to generate 1050 kW for the interval 5 and so on.

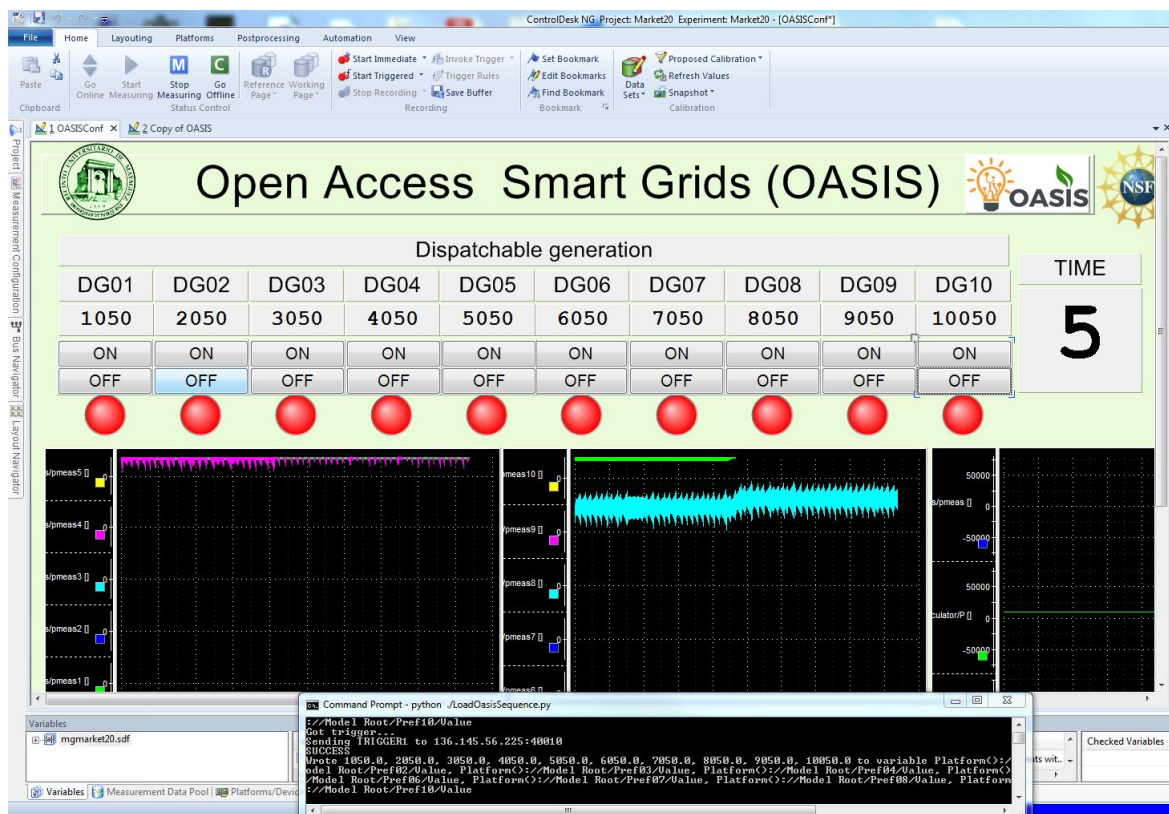


Fig 3.25: Simulation model running in ControlDesk Software

The complete interaction between the broker system and the hardware-in-the-loop can be seen in figure 3.26. The format by which the broker and the hardware-in-the-loop are communicated is JSON, the dispatch and feedback have the same structure, with varying shipping addresses. The dispatch sends the broker to the hardware-in-the-loop, which contains the agreed energy transactions in the auction system. The dispatch tells the hardware-in-the-loop the amount of energy to be generated by each generator in specific hours, at the same time, the dispatch informs how much energy is going to consume each load during the day. In other words, dispatch sends the agreements set in an ideal scenario.



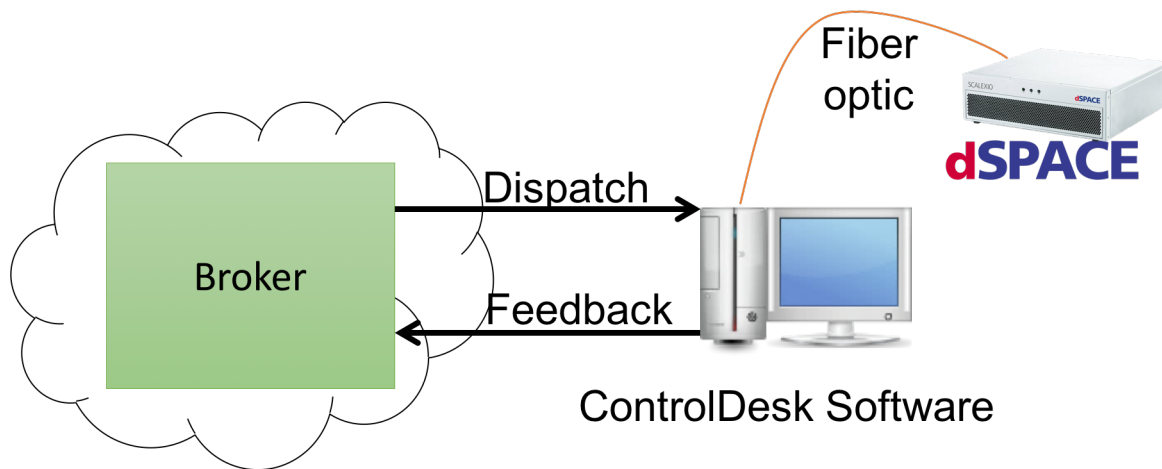


Fig 3.26: Communication between the broker system and dSPACE device

The feedback is the response that sends the hardware-in-the-loop to the broker; it communicates what really happened at a physical level, taking into account how the simulation adds a rate of variability. For example, generator one should generate 100 kWh between 10:00 am and 11:00 am according to what was established in the system but it actually generated 90 kWh. This is sent back to the system to establish the reliability of each generator and some penalties if necessary. It could also happen that a consumption profile has been set at 30 kWh during the day but at the end consumed 40 kWh, this exchange of information is vital to improve the system and evaluate different situations that may arise. In our simulation, we have the traditional energy grid to provide energy when some of the generators cannot produce what they promise, or when the consumers exceed expected demand.

# Chapter 4

## BROKER MODULE & ALGORITHMS

### 4.1 System Overview

The broker module is a job scheduler. In order to carry out the implementation of this module it was necessary to use the Quartz library [31] taking advantage of three main components: **Job, Trigger and Scheduler**. Quartz is a richly featured, open source job scheduling library that can be integrated within virtually any Java application - from the smallest stand-alone application to the largest e-commerce system. We now describe these three components:

#### 4.1.1 Job

The Job is the task that is going to be done. Inside this element we write all the source code that we want to run during the lifetime of the program. In this case, the Broker class implements the Job interface and rewrites the contents of the execute method, this code can be seen in the listing A.1 in appendix A.

The code in the execute method performs three fundamental tasks. In the first case,

it reads the demand and the market offers from the auction web system, and establishes the date of the next day as the one-day ahead model for the energy market. Secondly, the method with the algorithm that is used to perform the energy assignment is run, using as parameters the demand and the supply that it has previously obtained. Finally, the broker establishes the communication with the dSPACE machine to send the assignments so the dSPACE can perform the simulations of the energy system.

#### **4.1.2 Trigger**

The Trigger is the component that indicates the moment at which the Job must be executed. The market closes every day at three in the afternoon, at which time the energy assignments must be made. We use CronMaker [32] to indicate the frequency to accomplish a job. CronMaker requires a regular expression to specify the time and date when a given job is to be run. The specific code that results from the implementation of the component can be seen in the listing A.2 in appendix A.

#### **4.1.3 Scheduler**

The Scheduler ties together the Job and the Trigger, it establishes the interconnection of what is going to be done and when it is going to be realized. Listing A.3 in appendix A shows the code of the class 'Main' along with its main method, here the three components are displayed and the last three lines belong to the Scheduler component.

### **4.2 Algorithms**

We implemented two algorithms to generate the energy allocations on the broker side. The first algorithm takes the FIFO theory as a basis, the second one uses the knapsack problem theory. Once the energy market closes, the broker asks the operator which algorithm to use in order to perform the energy allocations, as can be seen in figure 4.1.

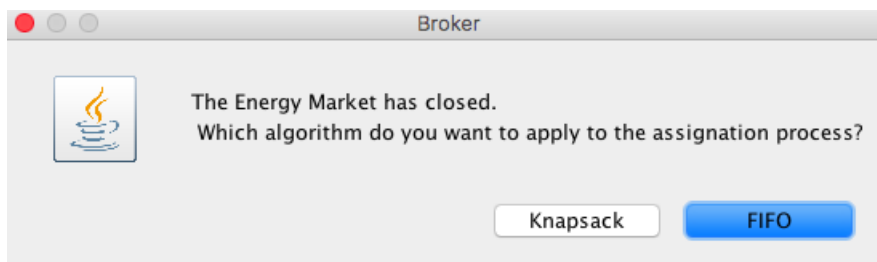


Fig 4.1: Broker - Algorithm selection

#### 4.2.1 FIFO Algorithm

The first algorithm is related to the theory of First In First Out (FIFO) queuing theory, and the graphical representation for this case was presented in figure 3.5. FIFO is a method for organizing and manipulating a data buffer, where the oldest (first) entry, or ‘head’ of the queue, is processed first. The entries in our system refers directly to the consumers. Every time a new consumer arrives to the platform to request energy, we enqueue this user request. At a later time, when the energy allocation are ready to happen we dequeue this request. A generic representation of this process is depicted in figure 4.2. According to the figure, consumers are represented by rectangles and labeled with numbers, we create a queue according to the order they register in the Auction Energy System (AES), for each period after the market close, we start to give the requested energy beginning from consumer 1, if the consumer has been satisfied, we dequeue it and move to the next.

The implementation of the FIFO algorithm is bound to honor consumers according to their timing for requests with the most economical energy price. The source code for the implementation of this algorithm can be seen in listing B.1 in appendix B. Once the customer queue has been created and the energy prices have been organized on the producers side, we will have to make the allocations. The algorithm loops through the list of energy demands requests, and the list of energy supply offers. At each step the algorithm performs the following actions:

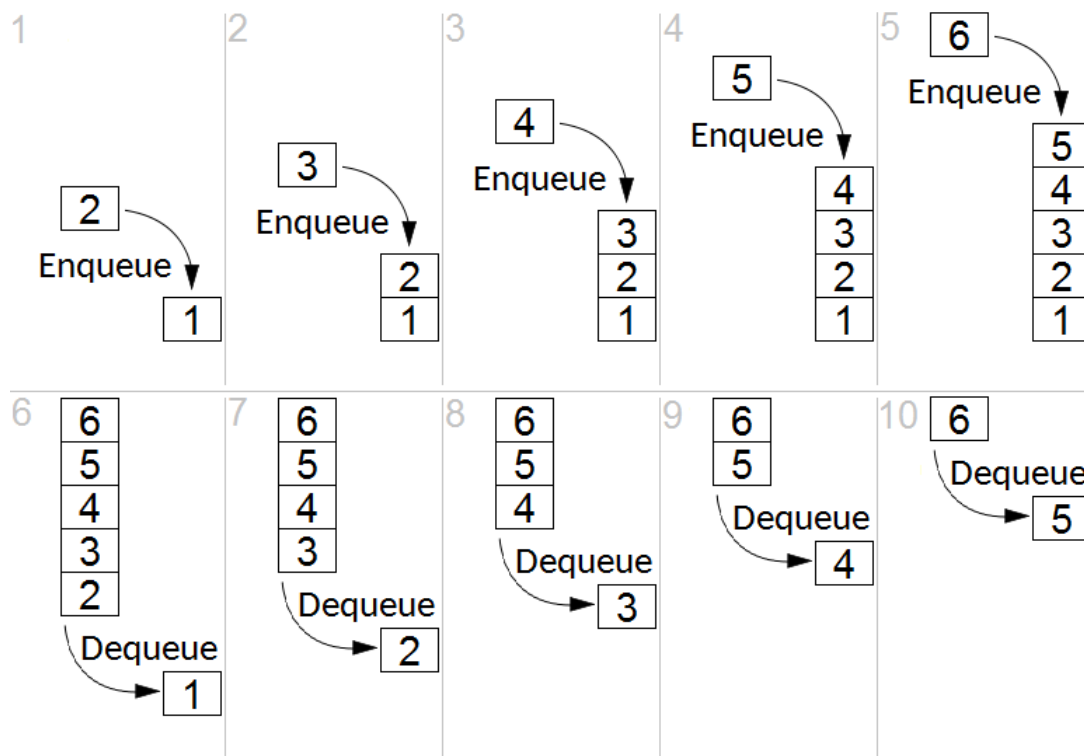


Fig 4.2: Algorithm - Generic FIFO

- Verify if there is demand and energy supply available. If not, finish.
- Pick a client  $C$  for which demand is not yet met.
- Pick an energy offer  $F$  which has not being assigned or is partially assigned.
- Assign energy to  $C$  from offer  $F$ .
- If client  $C$  has been satisfied, then move to next client in the next iteration. Otherwise stay with client  $C$ .
- If the energy offer  $F$  has been totally assigned, move to the next offer. Otherwise, stay with offer  $F$ .

The algorithm ends when either all clients are satisfied, or the energy offers are consumed, whichever happens first.

We are going to review the execution of this algorithm given the following data. Suppose we have the following bids in the system (table 4.1), these bids have been submitted by the producers and sorted by price in ascending order. On the other hand, we have consumers registered with some estimated demand for the next day (table 4.2), these consumers were sorted in order of arrival.

Id Bid	Block Size (kWh)	Cost per kWh (\$)
91	10	0.85
88	40	0.86
89	50	0.89
87	36	0.92
90	5	1.1
92	13	1.3
85	25	1.5
86	30	2

Table 4.1: Bids submitted - FIFO

Id Consumer	Demand (kWh)
11	28
12	20
13	28
14	28
15	21
16	30
17	21
19	30
20	28
21	28

Table 4.2: Consumers - Daily demand

If the broker operator selected the FIFO algorithm to make energy allocations, the result will be the assignments presented in table 4.3. The algorithm takes the bids and the consumers demand as inputs, and begins to pair energy blocks to energy demands based on the FIFO procedure presented in listing B.1 in appendix B. For example, consumer 11 requires 28 kWh, and to satisfy this demand, the brokers assigns the energy bids 91 and

88 to it. Notice that energy bid 91 only has 10kWh, and is not enough to satisfy 100% of the demand for consumer 11. Hence, the broker completes the demand with energy from bid 88, which has 40 kWh. Notice also that only 18kWh, out of those 40kWh, are used to complete the demand for consumer 11. The remaining 22kWh are available for the next consumer in line. In this case, that customer is the one with id 12. The broker assigns 20kWh from bid 88 to customer 12, satisfying 100% of this customer's demand. Bid 88 still has 2kWh to be assigned to the next customer. This process continues until all energy bids are consumed or demand is met. Table 4.3 shows the numbers of energy assignments made, the resource assigned in each case and the associated cost. In this scenario, the offers submitted from the renewable sources were less than the client demand. Hence, the power grid will provide the missing 53 kWh to cover the demands from consumer 20 and 21.

Assignment Id	Id Bid	Id Consumer	Resource Assigned (kWh)	Cost (\$)
1	91	11	10	8.5
2	88	11	18	15.48
3	88	12	20	17.2
4	88	13	2	1.72
5	89	13	26	23.14
6	89	14	24	21.36
7	87	14	4	3.68
8	87	15	21	19.32
9	87	16	11	10.12
10	90	16	5	5.5
11	92	16	13	16.9
12	85	16	1	1.5
13	85	17	21	31.5
14	85	19	3	4.5
15	86	19	27	54
16	86	20	3	6

Table 4.3: FIFO Assignments

#### 4.2.2 Knapsack Algorithm



Fig 4.3: Algorithm - Knapsack Problem Illustration

The second algorithm is related to the theory of knapsack problem, we depicted an example in figure 4.3. In this scenario, we have a general constraint (maximum weight that knapsack can resist) that is the maximum amount of CO<sub>2</sub> emissions that system operator will permit. These emissions are measured in terms of kilograms. Each one of the elements that we can put into the knapsack have a weight and a cost, those elements correspond to the the bids that the producers submitted to the AES. The algorithm will find the best allocation for those bids into the Knapsack, minimizing the price. We capture the value of the constraint in figure 4.4 if the Knapsack algorithm is chosen in the broker operation window (figure 4.1).



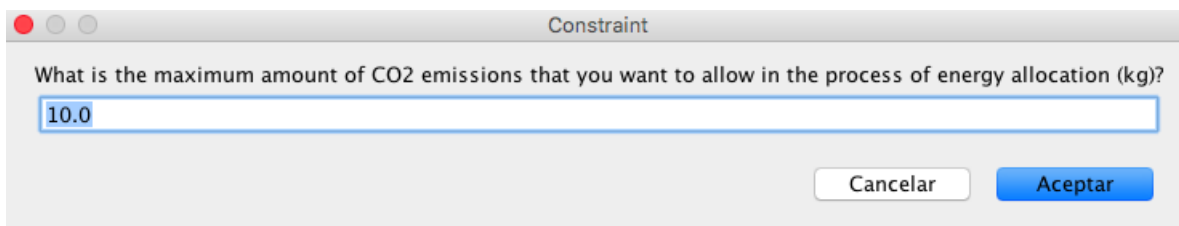


Fig 4.4: Algorithm - Knapsack CO2 emission constraint

We implemented the Fractional Knapsack with a greedy approach, the theory was proposed by Dantzig [27], we chose this variant given the context, each bid submitted by a producer can be splitted. For instance, a producer has 5kWh available to sell but the market only needs 2 kWh, hence the broker can take only the fractional amount that is necessary and make the arrangement. With 0/1 Knapsack variation this would not be possible.

The first key ingredient is the greedy-choice property: we can assemble a globally optimal solution by making locally optimal (greedy) choices [33]. In other words, when we are considering which choice to make, we make the choice that looks best in the current problem, without considering results from subproblems. In a greedy algorithm, we make whatever choice seems best at the moment and then solve the subproblem that remains.

The first step in the implementation is calculate the *density* for each item. The formal definition of *density* and the general overview of the greedy approach implemented was introduced in section 3.4.2. This step was carried out on the server side, taking advantage of the benefits of the database to complete the task. The code is showed in listing B.2 in appendix B. The outcome of this step is a table with information about the block size, the cost, the CO2 emission and, the calculation of the *density*. It was illustrated in figure 3.6.

The second step is sort the items on the basis of the density and value in ascending order giving that we are minimizing the cost, the code presented in listing B.2 also accomplished this task. Note the line of code *“ORDER BY density, value ASC”*.

In last step, we start from the item with the lowest density and value respectively, we put the items into the Knapsack (assign bids to consumers) taking as much items as we can according to the constraint of CO2 emissions. The code is showed in listing B.3 in appendix B.

We are going to discuss the execution of this algorithm given the same data presented before in table 4.1 and table 4.2. The data in table 4.2 that correspond to consumers demand remains the same. We create the table 4.4 from table 4.1 showing extra features that are relevant to the Knapsack algorithm, the sorting of these bids are different given that the criteria is the *density* column in an increasing order, and the second criteria is the cost.

The Knapsack algorithm needs a constraint, in our case this constraint refers to the maximum amount of CO2 emissions that system operator will permit. These emissions are measured in terms of kilograms. For this scenario we entered a restriction of 8 kilograms of CO2.

Id Bid	Block Size (kWh)	Cost per kWh (\$)	Weight	Density
88	40	0.86	0	0
89	50	0.89	0	0
87	36	0.92	0	0
90	5	1.1	0	0
85	25	1.5	0.313	4.79233226837061
91	10	0.85	0.1773	4.7941342357586
92	13	1.3	0.266	4.88721804511278
86	30	2	0.313	6.38977635782748

Table 4.4: Bids submitted - Knapsack

If the broker operator chose the Knapsack algorithm to make energy allocation, the result will be the assignments presented in table 4.5. The algorithm takes the bids, the consumers demand and the general constraint of CO2 emission as inputs to make the arrangements according to the algorithms steps explained before. The table exhibits the numbers of energy assignments made, the resource assigned in each case and the associated

cost. In this scenario, the offers submitted were less than the demand but also we have an extra constraint that reduced the offers in certain way, the grid will provide the missing 106 kWh to cover the consumer 17, 19, 20 and 21. The total of CO2 emission reached was 7.825 kilograms.

Assignment Id	Id Bid	Id Consumer	Resource Assigned (kWh)	Cost (\$)
1	88	11	28	24.08
2	88	12	12	10.32
3	89	12	8	7.12
4	89	13	28	24.92
5	89	14	14	12.46
6	87	14	14	12.88
7	87	15	21	19.32
8	87	16	1	0.92
9	90	16	5	5.5
10	85	16	24	36
11	85	17	1	1.5

Table 4.5: Knapsack Assignments

# Chapter 5

## RESULTS

In this chapter we analyze the two algorithms implemented (FIFO and Knapsack), according to their efficiency, and the outcome of the energy allocation process.

### 5.1 Efficiency

The efficiency can be measured in terms of *running time* and *space* according to the size of the input.

When considering the efficiency of an algorithm we always consider the worst case. We analyze the run-time and space for each algorithm through *Big O* notation [34]. It is a notation to represent classes of functions satisfying a condition of growth, based on the input size.

We establish an *order* to an algorithm to identify the running time for the algorithm and other *order* to identify the amount of memory space that the algorithm needs. In this case,  $n$  is the size of the input and  $f(n)$  is the running time or space needed of the algorithm relative to input size. Suppose that we have two algorithms to solve a problem, we may compare them by comparing their orders. The table 5.1 exhibits a list of common types of orders and their names in ascending order.

In algorithms, one the most expensive steps is sorting. In our case, sorting is executed

Name	Notation
Constant	$O(1)$
Logarithmic	$O(\log(n))$
Linear	$O(n)$
Linearithmic	$O(n \log(n))$
Cuadratic	$O(n^2)$
Cubic	$O(n^3)$
Exponential	$O(b^n), b > 1$
Factorial	$O(n!)$

Table 5.1: Big O - Types of Order

by the database engine itself. Therefore, we do not account for it in our analysis. Both algorithm receive the inputs already sorted.

The FIFO algorithm implementation (listing B.1 in appendix B) has a for loop with a constant quantity of operations that depends on the size of the vector of demand or offer, so this algorithm is order  $O(n)$  where  $n$  is the vector size, it has a linear time and space requirements.

Knapsack algorithm implementation (listing B.3 in appendix B) has a for loop with a constant quantity of operations that depend on the size of the vector of demand or offer, so this algorithm is order  $O(n)$  too ( $n$  is vector size), it has a linear time and space requirements.

Hence, both algorithms have the same efficiency in terms of time and space complexity.

## 5.2 Outcomes

For this section we analyzed the outcomes for three different scenarios in the energy allocation process as follow:

1. **Demand > Offer:** The quantity of energy requested by consumers is more than quantity of energy offered in bids by producers.
2. **Demand = Offer:** The quantity of energy requested by consumers is equal to

the quantity of energy offered in bids by producers.

3. **Demand < Offer:** The quantity of energy requested by consumers is less than quantity of energy offered in bids by producers.

For each scenario, we took in account four factors to examine:

- Wall-clock time for running the algorithm.
- Number of energy assignments made.
- Energy assigned from independent producers Vs. Energy requested from the power grid.
- CO2 emissions

The Knapsack algorithm needs a constraint, in our case this constraint refers to the maximum amount of CO2 emissions that system operator will permit. These emissions are measured in terms of kilograms. In this way, we evaluated Knapsack algorithm for each scenario with three different constraints of CO2 as follows:

$$CO2levels = \begin{cases} Low : \leq 3000kg \\ Medium : \leq 6000kg \\ High : \leq 9000kg \end{cases}$$

### 5.2.1 Demand > Offer

In this scenario, we prepare a total demand of 68772 kWh that comes from 2500 consumers, each individual consumer has a demand from 25 to 30 kWh according to the demand profiles presented in section 3.2.

Offers come from 2000 energy bids submitted to the system, those bids represent a total offer of 51399 kWh. Each submitted bid offers between 1 and 50 kWh and has a

cost between \$1 to \$1.5 per kWh. For this scenario, the price of energy coming from the grid costs more than the energy coming from the independent producers.

Each bid submitted to the system is randomly linked to 1 out of 12 energy sources that include renewable energy (7 of them) and non renewable energy (5 of them). The renewable energy comes from sun and wind sources while not renewable energy comes from natural gas, fuel oil, industrial coal, diesel and liquefied petroleum gas (LPG).

The figure 5.1 depicts the running time of the algorithms. We ran the algorithms three times and we took the average running time. Each running time include the period spent in calculations and storage of information into database. We can see there is no high variation in time between algorithms, the lowest value presented by Knapsack algorithm with a low limit of CO2 is explained by this algorithm did not make as many assignments as the others.

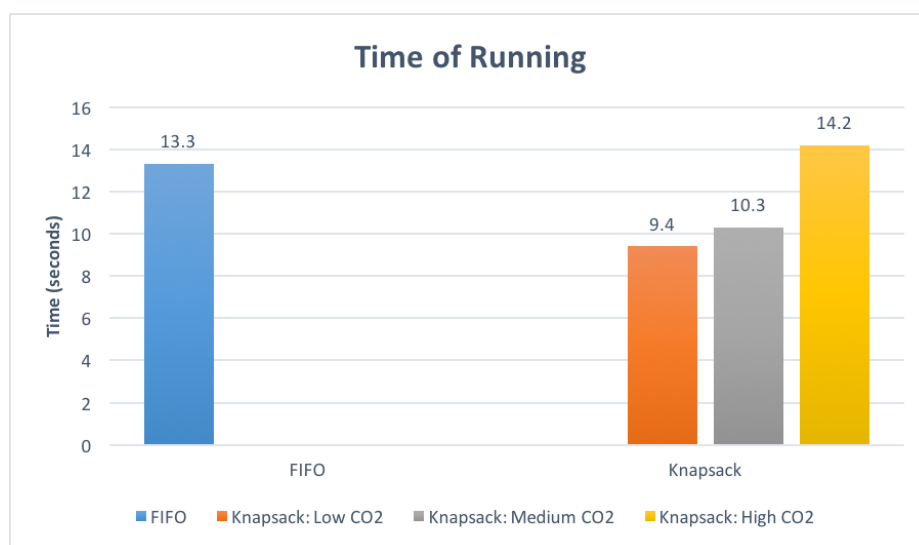


Fig 5.1: Demand > Offer - Algorithms Time Running

The figure 5.2 presents the number of energy assignments that broker made through the algorithms. The behavior is quite similar except in the value presented by Knapsack algorithm with a low limit of CO2. It is because Knapsack algorithm adds an extra constraint that can limit the assignments. The tighter the restriction, the fewer the

assignments that can be made. The FIFO algorithm does not have this restriction.

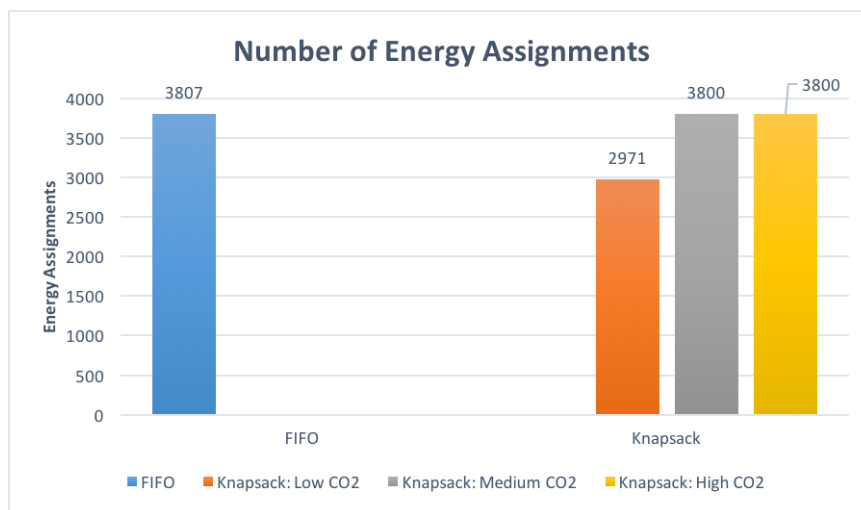


Fig 5.2: Demand > Offer - Energy Assignments

For this scenario the total demand of energy is 68772 kWh. Figure 5.3 depicts the amount of energy assigned from producers Vs. Energy requested from the grid. For FIFO, Knapsack Medium CO2 and, Knapsack High CO2 the quantity assigned by the broker is the same (more than 70%), in consequence, the quantity of energy requested from the grid is the same also (less than 30%). On the other hand, Knapsack Low CO2 has to request more energy from the grid (near 35%) than the other algorithms due to the CO2 restriction.



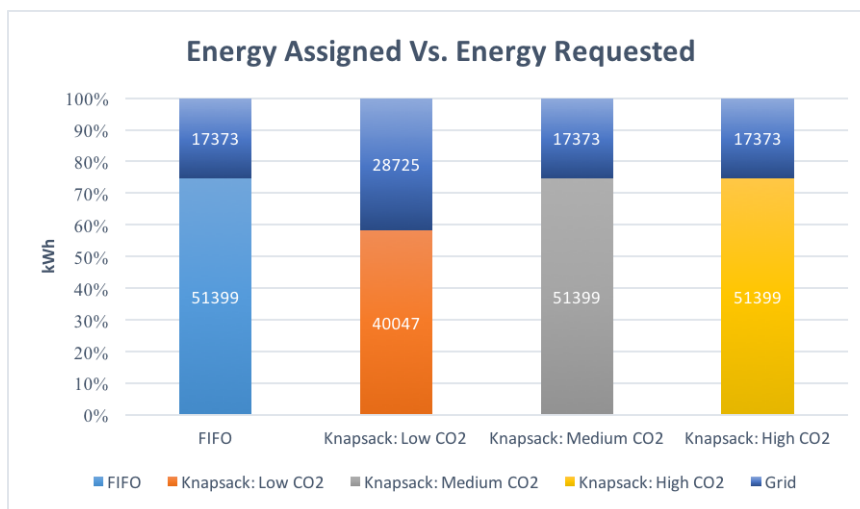


Fig 5.3: Demand > Offer - Energy Assigned Vs. Energy Requested

Figure 5.4 shows the quantity of CO<sub>2</sub> emitted in the energy assignment process. Knapsack Low CO<sub>2</sub> presented the lowest emission rate. The constraint limits this algorithm to a maximum emission of 3000 kilograms of CO<sub>2</sub>.

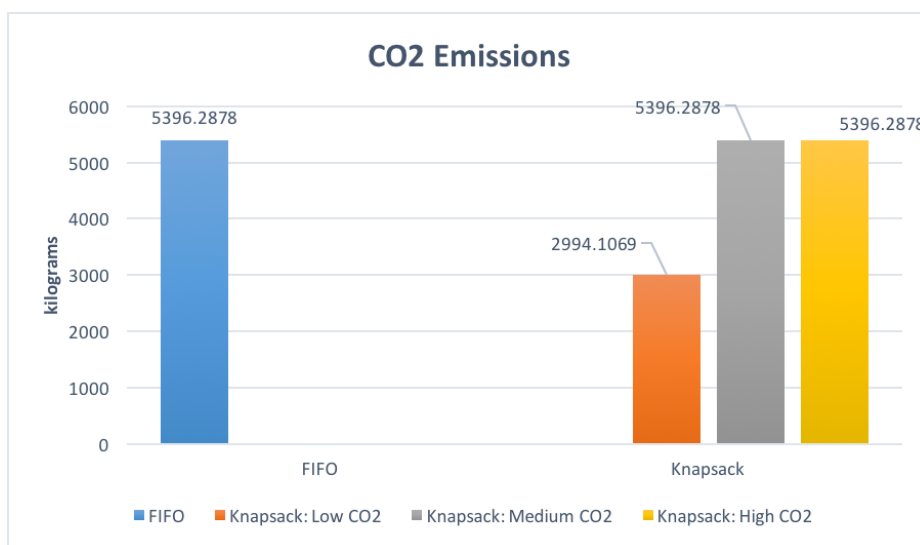


Fig 5.4: Demand > Offer - CO<sub>2</sub> Emissions

### 5.2.2 Demand = Offer

In this scenario, we kept a total demand of 68772 kWh that comes from 2500 consumers, each individual consumer has a demand from 25 to 30 kWh according to the demand

profiles presented in section 3.2.

Offers come from 2500 energy bids submitted to the system, those bids represent a total offer of 68772 kWh. The demand is equal to the offer. Each submitted bid offers between 1 and 50 kWh, and has a cost between \$1 to \$1.5 per kWh. As before, energy from the grid is more expensive than energy from the independent producers.

Each bid submitted to the system is randomly linked to 1 out of 12 energy sources that include renewable energy (7 of them) and non renewable energy (5 of them). The renewable energy comes from sun and wind sources while not renewable energy comes from natural gas, fuel oil, industrial coal, diesel and liquefied petroleum gas (LPG).

The figure 5.5 depicts the time of running of the algorithms. We ran the algorithms three times and we took the average time of running. Each running time include the period spent in calculations and storage of information into database. The figure shows the lowest value was presented by Knapsack algorithm with a low limit of CO<sub>2</sub>, it is explained by this algorithm did not make as much assignments as the others. The other algorithms share a close range in execution time.

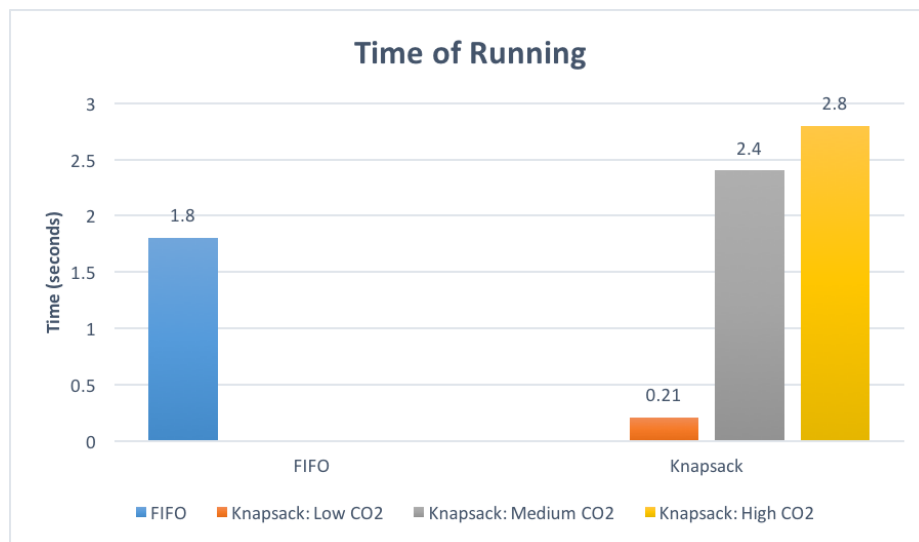


Fig 5.5: Demand = Offer - Algorithms Time Running

Figure 5.6 exhibits the numbers of energy assignments that broker made through the

algorithms. The behavior is quite similar except in the value presented by Knapsack algorithm with a low limit of CO<sub>2</sub>. It is because Knapsack algorithm adds an extra constraint that can limit the assignments. The lower the value of CO<sub>2</sub> emission restriction, Knapsack algorithm trends to make less assignment.

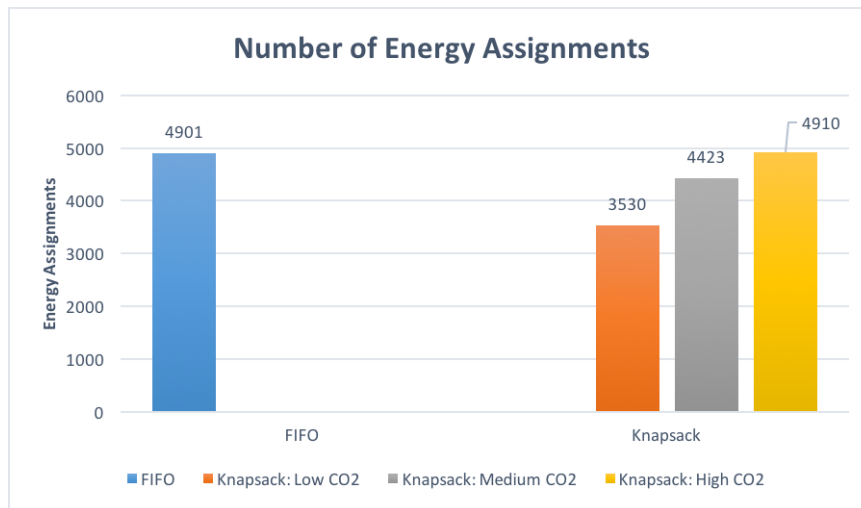


Fig 5.6: Demand = Offer - Energy Assignments

For this scenario, demand and offer are equals (68772 kWh). The figure 5.7 depicts the amount of energy assigned from producers Vs. Energy requested from the grid. FIFO and Knapsack High CO<sub>2</sub> did not require any energy from the grid. FIFO algorithm does not take into account this restriction of CO<sub>2</sub> and Knapsack High CO<sub>2</sub> has a large limit for this restriction so, they almost do not have limit to make assignments in this scenario. On the other hand, Knapsack Low CO<sub>2</sub> and Knapsack Medium CO<sub>2</sub> had to request energy from the grid due to the constraint.

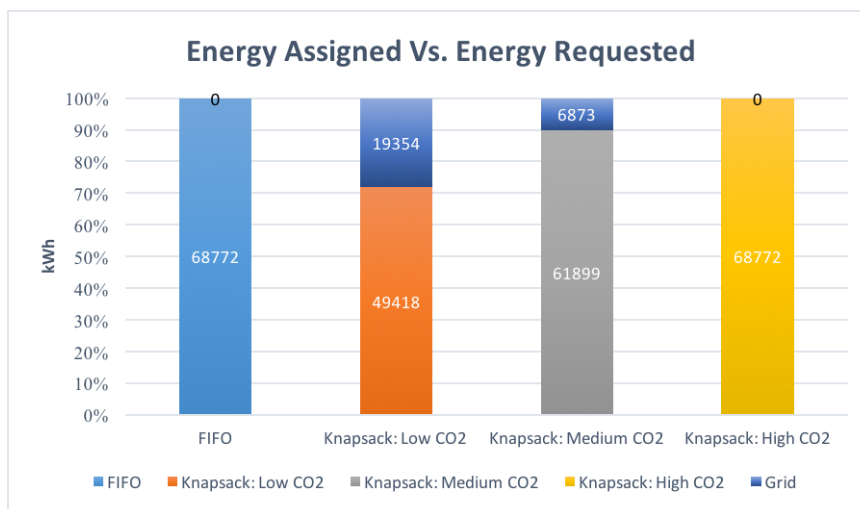


Fig 5.7: Demand = Offer - Energy Assigned Vs. Energy Requested

Figure 5.8 exhibits the quantity of CO<sub>2</sub> emitted in the energy assignment process. Knapsack Low CO<sub>2</sub> presented the lowest emission rate. The constraint limits this algorithm to a maximum emission of 3000 kilograms of CO<sub>2</sub>.

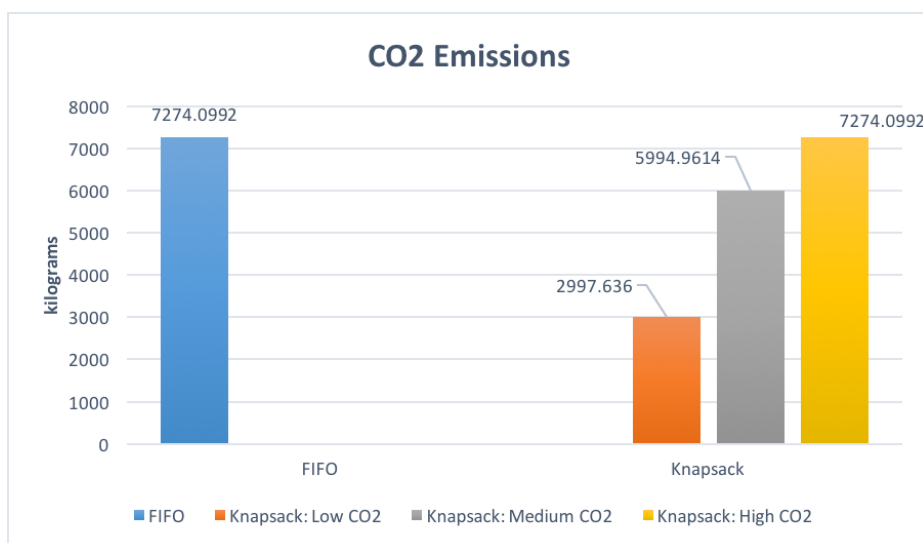


Fig 5.8: Demand = Offer - CO<sub>2</sub> Emissions

### 5.2.3 Demand < Offer

In this scenario, we kept a total demand of 68772 kWh that comes from 2500 consumers, each individual consumer has a demand from 25 to 30 kWh according to the demand

profiles presented in section 3.2.

Offers comes from 3500 energy bids submitted to the system, those bids represent a total offer of 93855 kWh. The demand is less than offer. Each submitted bid offers between 1 and 50 kWh and has a cost between \$1 to \$1.5 per kWh. As before, energy from the grid is more expensive than energy from the independent producers.

Each bid submitted to the system is randomly linked to 1 out of 12 energy sources that include renewable energy (7 of them) and non renewable energy (5 of them). The renewable energy comes from sun and wind sources while not renewable energy comes from natural gas, fuel oil, industrial coal, diesel and liquefied petroleum gas (LPG).

It is important to highlight that some producers will not sell their energy. If we use FIFO algorithm to make energy allocation is more probable that energy with a low price can sell, energy with a high price will stay without selling. On the other hand, if we use Knapsack algorithm to make energy assignments is more probable that energy from renewable sources can sell, energy with a high emission of CO<sub>2</sub> could stay without selling.

Figure 5.9 depicts the time of running of the algorithms. We ran the algorithms three times and we took the average time of running. Each running time include the period spent in calculations and storage of information into database. The figure shows that any variant of the Knapsack runs faster than FIFO, and that among the variants of Knapsack the running is virtually the same.

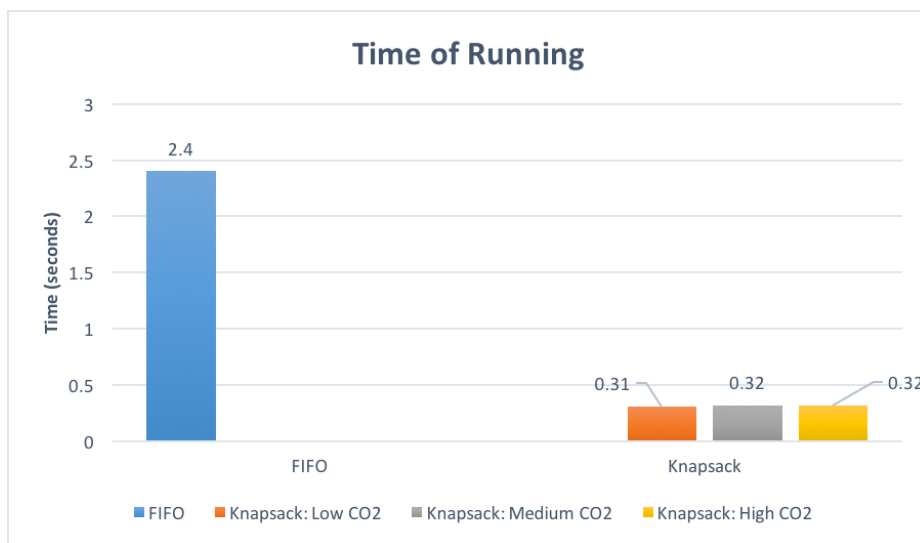


Fig 5.9: Demand < Offer - Algorithms Time Running

Figure 5.10 exhibits the numbers of energy assignments that broker made through the algorithms. The behavior is quite similar except in the value presented by Knapsack algorithm with a low limit of CO<sub>2</sub>. It is because Knapsack algorithm adds an extra constraint that can limit the assignments.

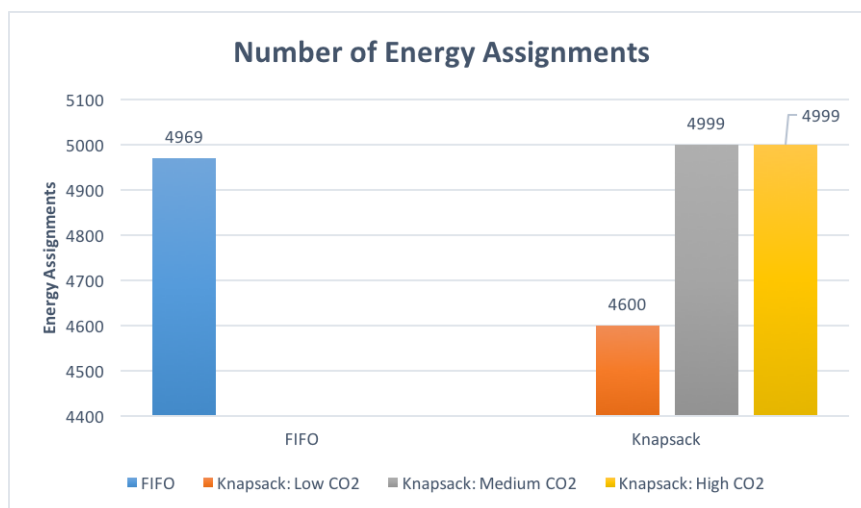


Fig 5.10: Demand < Offer - Energy Assignments

For this scenario, demand is less than offer. It is likely that grid power is not required. The figure 5.11 depicted the amount of energy assigned from producers Vs. Energy

requested from the grid. The only one who requested energy from the grid was Knapsack Low CO<sub>2</sub> due to the constraint. Those result indicate also a high offer from renewable energy.

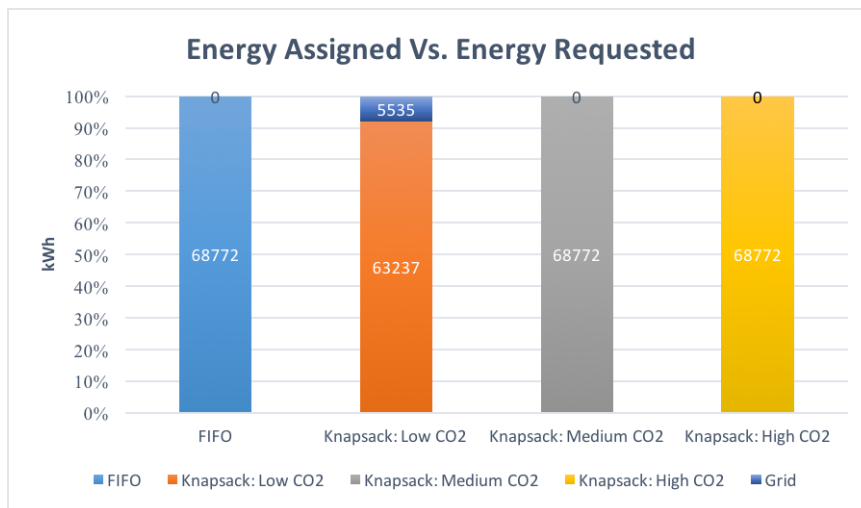


Fig 5.11: Demand < Offer - Energy Assigned Vs. Energy Requested

The figure 5.12 exhibits the quantity of CO<sub>2</sub> emitted in the energy assignment process. Knapsack Low CO<sub>2</sub> presented the lowest emission rate. The constraint limits this algorithm to a maximum emission of 3000 kilograms of CO<sub>2</sub>.

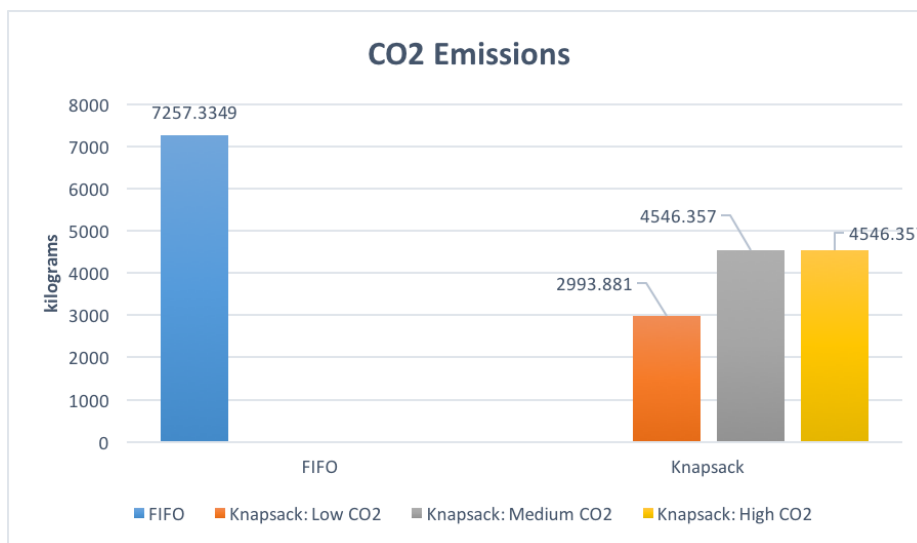


Fig 5.12: Demand < Offer - CO<sub>2</sub> Emissions

For this scenario, the results for Knapsack Medium CO2 and Knapsack High CO2 are the same for the four factors analyzed. We noticed that the CO2 emissions do not reach their limits in both cases, it could be a consequence of high presence of renewable (solar and wind) energy in the offers.



## Chapter 6

# CONCLUSIONS AND FUTURE WORK

This document exhibits the design, implementation, and communication scheme of a brokerage system for Smart Grids in the framework of the OASIS project. SG are constantly evolving and this work introduces a valuable computational contribution as it supports in obtaining rules and important market factors in micro grids. The work performed contributes to generate explicit interactions between each of the market participants, their responsibilities, benefits, penalties, individual and global requirements.

By the time we write this final part of the document, Puerto Rico has only 7% of customers with electricity, due to the passage of hurricane Maria two weeks ago. It is expected that the remaining 93% will be re-established over the next 6 months. In less than a year, Puerto Rico has been hit by events that have left the country without electricity for several days. It is necessary to give continuity and validity to the alternatives of renewable sources that allow the independence of the service that supplies the network of the Electric Energy Authority. This is a call not only to Puerto Rico but to the whole world to offer a sustainable future to the next generations.

Both algorithm, FIFO and Knapsack, have the same efficiency in terms of time and

space complexity. They minimize the price since they assign the energy from the less expensive sources. They also can make arrangements in different ways according to the necessity and parameters chosen by broker system operator.

Our future work will focus on give more options to consumers. For example, they could establish a payment limit and make request about the minimum confidentiality expected from the energy system. On the other hand, we will test the system with a complex model running into the dSPACE device, this scenario is under design by Electrical Engineers in the OASIS team, we expect an IEEE 13 Node Test Feeder model with all the specifications and regulations to complete other simulations, receive the feedback of this system and incorporate this to the calculations to approve or reject some transactions according to the physical layer constraints.

The implemented architecture allows you to add multiple brokers and have millions of producers and consumers. We expect to put ten or more brokers into operation and receive a gigantic amount of transaction data, while it is expected to soon implement the 1 hour ahead model in exchange for the 1 day ahead model, which positions the system as an application of real-time big data services which impacts the confidentiality and availability of the electric power service.

OASIS project in the computational and electrical part expect to replace the hardware-in-the-loop for a real SG environment as we depicted in figure 6.1. In this scenario, we do not have a machine carrying out simulations, instead, we are going to interact directly with real resources in SG.

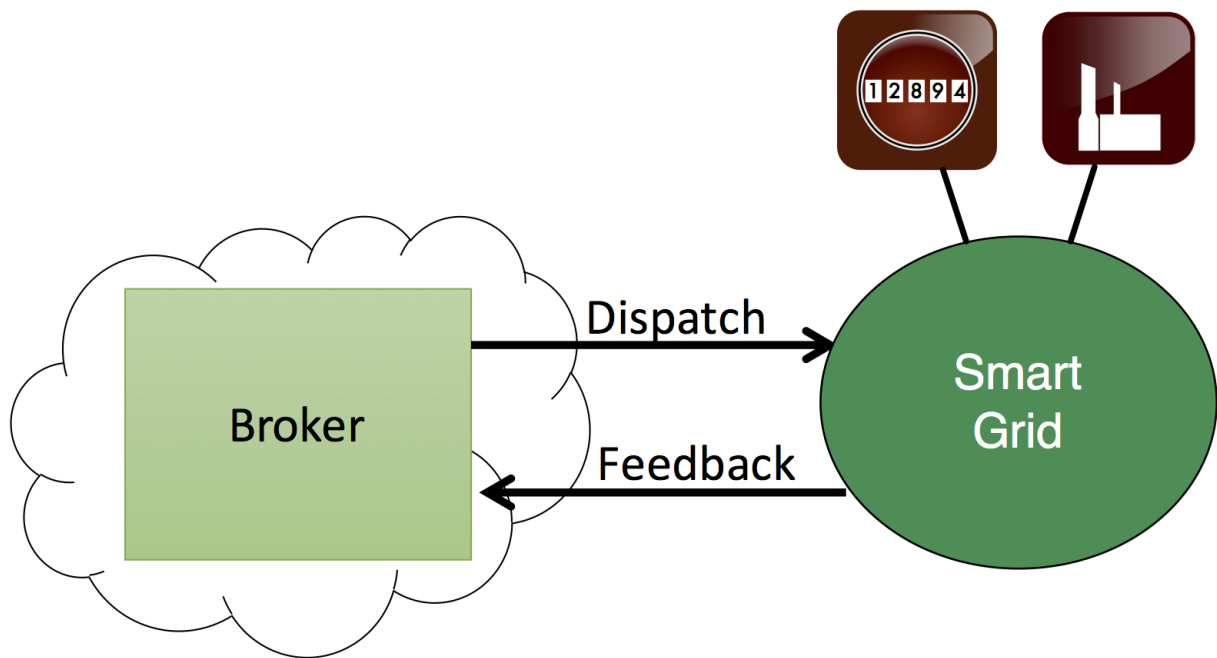


Fig 6.1: OASIS Future Work

# Appendices

# Appendix A

## Broker - Components

```
1 public class Broker implements Job {
2
3     private static String MODELNAME = "OASIS";
4     private static String IP_COMPUTER_DSPACE = "136.145.56.225";
5     private static int COMPUTER_PORT = 40002;
6     private ArrayList<Offer> next_day = new ArrayList<>();
7     private DspacePayload dspacePayload = new DspacePayload();
8
9     @Override
10    public void execute(JobExecutionContext jobExecutionContext) throws
11    JobExecutionException {
12        ArrayList<Demand> fifo_demand = new Demand().fifo_demand();
13        ArrayList<Offer> offers = new Offer().get_offers();
14        ArrayList<Knapsack> offers_knapsack = new Knapsack().
15    get_offers_knapsack();
16        Assignment assignment = new Assignment();
17        Date date = new Date();
18        Calendar calendar = Calendar.getInstance();
19        calendar.setTime(date);
20        calendar.add(Calendar.DAY_OF_YEAR, 1);
```

```

19     date = calendar.getTime();
20     Object [] options = {"FIFO",
21         "Knapsack"};
22     int n = JOptionPane.showOptionDialog(new Frame(),
23         "The Energy Market has closed. \n Which algorithm do you
want to apply to the assignation process?",
24         "Broker",
25         JOptionPane.YES_NO_CANCEL_OPTION,
26         JOptionPane.QUESTION_MESSAGE,
27         null,
28         options,
29         options[0]);
30     if (n == 0){ // FIFO
31         boolean result = assignment.assignment_x_cost(fifo_demand,
offers, date, this);
32         dspacePayload.setConsumptionSchedule(assignment.
consumptionSchedule(fifo_demand));
33         dspacePayload.setProductionSchedule(this.productionSchedule());
34         System.out.println(dspacePayload.getConsumptionSchedule());
35         System.out.println(dspacePayload.getProductionSchedule());
36         if (result)
37             JOptionPane.showMessageDialog(new Frame(), "Operation
Successful", "Broker", JOptionPane.INFORMATION_MESSAGE);
38         else
39             JOptionPane.showMessageDialog(new Frame(), "Error in the
Operation", "Broker", JOptionPane.ERROR_MESSAGE);
40     }else if(n == 1){ // Knapsack (Ask for maximum CO2 emissions -> W)
41         double W = Double.parseDouble((String) JOptionPane.
showInputDialog(new Frame(),
42             "What is the maximum amount of CO2 emissions that you
want to allow in the process of energy allocation (kg)?",
43             "Constraint",

```

```

44         JOptionPane.PLAIN_MESSAGE,
45         null ,
46         null ,
47         "10.0"));
48         boolean result = assignment.assignment_knapsack(W, fifo_demand ,
offers_knapsack , date , this);
49         dspacePayload.setConsumptionSchedule(assignment.
consumptionSchedule(random_demand));
50         dspacePayload.setProductionSchedule(this.productionSchedule());
51         if (result)
52             JOptionPane.showMessageDialog(new Frame() , "Operation
Successful" , "Broker" , JOptionPane.INFORMATION_MESSAGE);
53         else
54             JOptionPane.showMessageDialog(new Frame() , "Error in the
Operation" , "Broker" , JOptionPane.ERROR_MESSAGE);
55
56     }
57     OABroker oasisModel = new OABroker(MODELNAME, IP_COMPUTER_DSPACE,
COMPUTER.PORT);
58     oasisModel.schedule(dspacePayload.getProductionSchedule() ,
dspacePayload.getConsumptionSchedule() , dspacePayload.
getConsumptionFeedback() , oasis.OAFlags.RESET);
59     oasisModel.feedback();
60 }

```

Listing A.1: Broker - Execute method code in Job Element

```

1 Trigger t1 = TriggerBuilder.newTrigger().
2     withIdentity("CronTrigger").
3     withSchedule(CronScheduleBuilder.
4     cronSchedule("0 0 15 1/1 * ? *")).
5     build();

```

Listing A.2: Broker - Trigger code

```
1 public class Main{
2
3     public static void main(String [] args) throws SchedulerException {
4
5         //Job
6         JobDetail job = JobBuilder.newJob(Broker.class).build();
7
8         // Trigger
9         Trigger t1 = TriggerBuilder.newTrigger().
10             withIdentity("CronTrigger").
11             withSchedule(CronScheduleBuilder.
12                 cronSchedule("0 0 15 1/1 * ? *")).
13             build();
14
15         // Scheduler
16         Scheduler scheduler = StdSchedulerFactory.getDefaultScheduler();
17         scheduler.start();
18         scheduler.scheduleJob(job, t1);
19     }
```

Listing A.3: Broker - Scheduler code



# Appendix B

## Broker - Algorithms

```
1 public boolean assignment_x_cost(ArrayList<Demand> demand, ArrayList<Offer>
  offer, Date date, Broker b) {
2     ArrayList<Assignment> assignments = new ArrayList<>();
3     boolean lock_client = true;
4     boolean lock_offer = true;
5     double availability = offer.get(0).getBlock_size();
6     double pending_demand = demand.get(0).getDaily_demand();
7     double unitary_cost = offer.get(0).getKwh_cost();
8     int k = 0;
9     for(int i = 0; i<demand.size()&&k<offer.size();i++) {
10        Assignment assignment = new Assignment();
11        if(!lock_client){
12            pending_demand = demand.get(i).getDaily_demand();
13        }
14        if (!lock_offer){
15            availability = offer.get(k).getBlock_size();
16            unitary_cost = offer.get(k).getKwh_cost();
17        }
18        if (availability <= pending_demand) {
19            if(availability > 0) {
```

```

20         assignment.setResource_assigned(availability);
21         assignment.setId_bid(offer.get(k).getId());
22         assignment.setCost(availability * unitary_cost);
23         assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
24         assignment.setDate(date);
25         pending_demand = pending_demand - availability;
26         availability = 0;
27         assignments.add(assignment);
28     }
29     lock_offer = false;
30     lock_client = true;
31     i = i-1;
32     k = k+1;
33 } else if (availability > pending_demand){
34     if (pending_demand > 0){
35         assignment.setResource_assigned(pending_demand);
36         availability = availability - pending_demand;
37         assignment.setId_bid(offer.get(k).getId());
38         assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
39         assignment.setCost(pending_demand * unitary_cost);
40         assignment.setDate(date);
41         assignments.add(assignment);
42     }
43     lock_client = false;
44     lock_offer = true;
45 }
46 }
47 printAssignments(assignments);
48 boolean flag1 = post_assignments(assignments);
49 boolean flag2 = deliverNextDay(b);

```

```

50     if(flag1 && flag2)
51         return true;
52     else
53         return false;
54 }

```

Listing B.1: Broker - Implementation of FIFO on the broker side

```

1  public List<Knapsack> offers_knapsacks () {
2      Transaction t = Ebean.beginTransaction();
3      List<Knapsack> result = new ArrayList<>();
4      try {
5          String sql = "SELECT " +
6                      "B.id, B.block_size, B.kwh_cost AS value, " +
7                      "E.emission_value AS weight, " +
8                      "(COALESCE((B.kwh_cost/NULLIF(E.emission_value,0)),0))
AS density " +
9                      "FROM bid B " +
10                     "INNER JOIN producer_profile PP " +
11                     "ON B.id_profile = PP.id_producer_profile " +
12                     "INNER JOIN emission_co2 E " +
13                     "ON PP.source = E.id " +
14                     "WHERE B.id_status = :status " +
15                     "ORDER BY density, value ASC";
16          RawSql rawSql = RawSqlBuilder.parse(sql)
17              .columnMapping("B.id", "id")
18              .columnMapping("B.block_size", "quantity")
19              .columnMapping("B.kwh_cost", "value")
20              .columnMapping("E.emission_value", "weight")
21              .columnMapping("(COALESCE((B.kwh_cost/NULLIF(E.
AS density)", "density")
22              .create();
23          Query<Knapsack> query = Ebean.find(Knapsack.class);

```

```

24     query.setRawSql(rawSql)
25         .setParameter("status", 1);
26     result = query.findList();
27     t.commit();
28 }catch (Exception e){
29     System.out.println(e.getMessage());
30     t.rollback();
31 }finally {
32     t.end();
33 }
34 return result;
35 }

```

Listing B.2: Density Calculation

```

1 public boolean assignment_knapsack(double W, ArrayList<Demand> demand,
   ArrayList<Knapsack> offers_knapsack,
2     Date date, Broker broker) {
3     boolean lock_client = true;
4     boolean lock_offer = false;
5     double availability = 0.0;
6     double pending_demand = demand.get(0).getDaily_demand();
7     double unitary_cost = 0.0;
8     double weight = 0.0;
9     double curWeight = 0.0; // Current weight in knapsack
10    double finalvalue = 0.0; // Result (value in Knapsack)
11    ArrayList<Assignment> assignments = new ArrayList<>();
12    int k = 0;
13
14    // Looping through all Items
15    for (int i = 0; i < demand.size() && k < offers_knapsack.size() ; i
16        ++) {
17        Assignment assignment = new Assignment();

```

```

17     if (!lock_client){
18         pending_demand = demand.get(i).getDaily_demand();
19     }
20     // If adding Item won't overflow, add it completely
21     if (curWeight + offers_knapsack.get(k).getWeight()*
offers_knapsack.get(k).getQuantity() <= W) {
22         if (!lock_offer) {
23             availability = offers_knapsack.get(k).getQuantity();
24             unitary_cost = offers_knapsack.get(k).getValue();
25             weight = offers_knapsack.get(k).getWeight();
26         }
27         if (availability <= pending_demand) {
28             if (availability > 0) {
29                 assignment.setResource_assigned(availability);
30                 assignment.setId_bid(offers_knapsack.get(k).getId()
);
31                 assignment.setCost(availability * unitary_cost);
32                 assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
33                 assignment.setDate(date);
34                 pending_demand = pending_demand - availability;
35                 curWeight += availability * weight;
36                 finalvalue += availability * unitary_cost;
37                 availability = 0;
38                 assignments.add(assignment);
39             }
40             lock_offer = false;
41             lock_client = true;
42             i = i - 1;
43             k = k + 1;
44         } else {
45             if (pending_demand > 0) {

```

```

46         assignment.setResource_assigned(pending_demand);
47         availability = availability - pending_demand;
48         assignment.setId_bid(offers_knapsack.get(k).getId()
);
49         assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
50         assignment.setCost(pending_demand * unitary_cost);
51         curWeight += pending_demand * weight;
52         finalvalue += pending_demand * unitary_cost;
53         assignment.setDate(date);
54         assignments.add(assignment);
55     }
56     lock_client = false;
57     lock_offer = true;
58 }
59 }else { // If we can't add current Item, add fractional part of
it
60     double remain = W - curWeight;
61     if(!lock_offer){
62         double last_profit = offers_knapsack.get(k).getValue()
* remain / offers_knapsack.get(k).getWeight();
63         unitary_cost = offers_knapsack.get(k).getValue();
64         weight = offers_knapsack.get(k).getWeight();
65         availability = last_profit / unitary_cost;
66     }
67     if (availability <= pending_demand) {
68         if (availability > 0) {
69             assignment.setResource_assigned(availability);
70             assignment.setId_bid(offers_knapsack.get(k).getId()
);
71             assignment.setCost(availability * unitary_cost);
72             curWeight += availability * weight;

```

```

73         finalvalue += availability * unitary_cost;
74         assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
75         assignment.setDate(date);
76         assignments.add(assignment);
77     }
78     lock_offer = false;
79     lock_client = true;
80     i = i - 1;
81     k = k + 1;
82 } else {
83     if (pending_demand > 0) {
84         assignment.setResource_assigned(pending_demand);
85         assignment.setId_bid(offers_knapsack.get(k).getId()
);
86         assignment.setId_profile_consumer(demand.get(i).
getId_consumer_profile());
87         assignment.setCost(pending_demand * unitary_cost);
88         curWeight += pending_demand * weight;
89         finalvalue += pending_demand * unitary_cost;
90         assignment.setDate(date);
91         assignments.add(assignment);
92     }
93     lock_client = false;
94     lock_offer = true;
95 }
96 break;
97 }
98 }
99 printAssignments(assignments);
100 boolean flag1 = post_assignments(assignments);
101 boolean flag2 = deliverNextDay(broker);

```

```
102     if(flag1 && flag2)
103         return true;
104     else
105         return false;
106 }
```

Listing B.3: Knapsack implementation code



# Bibliography

- [1] Samaresh Bera, Sudip Misra, and Joel J.P.C. Rodrigues. “Cloud Computing Applications for Smart Grid: A Survey”. In: *IEEE Transactions on Parallel and Distributed Systems* 26.5 (2015), pp. 1477–1494. ISSN: 1045-9219. DOI: 10.1109/TPDS.2014.2321378. URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6809180>.
- [2] M. N. Akter, M. A. Mahmud, and Amanullah M. T. Oo. “A hierarchical transactive energy management system for microgrids”. In: *2016 IEEE Power and Energy Society General Meeting (PESGM)* (2016), pp. 1–5. ISSN: 19449933. DOI: 10.1109/PESGM.2016.7741099. URL: <http://ieeexplore.ieee.org/document/7741099/>.
- [3] Ralph Masiello and Julio Romero Agüero. “Sharing the Ride of Power: Understanding Transactive Energy in the Ecosystem of Energy Economics”. In: *IEEE Power and Energy Magazine* 14.3 (2016), pp. 70–78. ISSN: 15407977. DOI: 10.1109/MPE.2016.2524965.
- [4] Sayyid Mohssen Sajjadi et al. “Transactive Energy Market in Distribution Systems : A Case Study of Energy Trading Between Transactive Nodes”. In: *North American Power Symposium (NAPS)*. IEEE, 2016, pp. 1–6. ISBN: 9781509032709.
- [5] Wilson Rivera and Manuel Rodriguez. “Towards Cloud Services in Smart Power Grid”. In: *2016 IEEE Innovative Smart Grid Technologies - Asia (ISGT-Asia)*. 2016, pp. 570–573. ISBN: 9781509043033.

- [6] *OASIS Project*. URL: <http://oasis.uprm.edu>.
- [7] Manuel Rodriguez-Martinez et al. *A Case for Open Access Smart Grids (OASIS)*. 2016.
- [8] Maarten Wolsink. “The research agenda on social acceptance of distributed generation in smart grids: Renewable as common pool resources”. In: *Renewable and Sustainable Energy Reviews* 16.1 (2012), pp. 822–835. ISSN: 13640321. DOI: 10.1016/j.rser.2011.09.006.
- [9] Peter Dondi et al. “Network integration of distributed power generation”. In: *Journal of Power Sources* 106.1-2 (2002), pp. 1–9. ISSN: 03787753. DOI: 10.1016/S0378-7753(01)01031-X.
- [10] Jiayan Yuan et al. “A real-time balancing market clearing model for grid-connected micro-grid including energy storage system”. In: *Proceedings - 2015 International Symposium on Smart Electric Distribution Systems and Technologies, EDST 2015*. 2015, pp. 32–36. ISBN: 9781479977369. DOI: 10.1109/SEDST.2015.7315178.
- [11] GridWise Architecture Council. *GridWise Transactive Energy Framework*. Tech. rep. 2015, pp. 11–15. URL: [http://www.gridwiseac.org/pdfs/te%7B%5C\\_%7Dframework%7B%5C\\_%7Dreport%7B%5C\\_%7Dpnnl-22946.pdf](http://www.gridwiseac.org/pdfs/te%7B%5C_%7Dframework%7B%5C_%7Dreport%7B%5C_%7Dpnnl-22946.pdf).
- [12] David Holmberg et al. *Transactive Energy Application Landscape Scenarios*. 2016.
- [13] By David Forfia, Mark Knight, and Ron Melton. “The View from the Top of the Mountain”. In: *IEEE Power Energy Magazine* 14.3 (2016), pp. 25–33.
- [14] Consolidated Edison. *Distributed System Implementation Plan (DSIP)*. Tech. rep. New York, 2016, pp. 19, 59. URL: <https://www.coned.com/-/media/files/coned/documents/dg/dsp/pdf/cecony-dsip.pdf?la=en>.
- [15] D J Hammerstrom et al. *Pacific Northwest GridWise ffdfffdfffd Testbed Demonstration Projects Part I . Olympic Peninsula Project*. 2007.

- [16] AEP Ohio. *AEP Ohio gridSMARTSM Demonstration Project*. Ohio, 2014. URL: [https://www.smartgrid.gov/files/AEP%7B%5C\\_%7DOhio%7B%5C\\_%7DDE-OE-0000193%7B%5C\\_%7DFinal%7B%5C\\_%7DTechnical%7B%5C\\_%7DReport%7B%5C\\_%7D06-23-2014.pdf](https://www.smartgrid.gov/files/AEP%7B%5C_%7DOhio%7B%5C_%7DDE-OE-0000193%7B%5C_%7DFinal%7B%5C_%7DTechnical%7B%5C_%7DReport%7B%5C_%7D06-23-2014.pdf).
- [17] Koen Kok. “The PowerMatcher: Smart Coordination for the Smart Electricity Grid”. PhD thesis. VRIJE UNIVERSITEIT, 2013.
- [18] Pacific Northwest National Laboratory. *Pacific Northwest Smart Grid Demonstration Project Technology Performance Report Volume 1: Technology Performance*. Tech. rep. 2015. URL: [https://www.smartgrid.gov/document/Pacific%7B%5C\\_%7DNorthwest%7B%5C\\_%7DSmart%7B%5C\\_%7DGrid%7B%5C\\_%7DTechnology%7B%5C\\_%7DPerformance.html](https://www.smartgrid.gov/document/Pacific%7B%5C_%7DNorthwest%7B%5C_%7DSmart%7B%5C_%7DGrid%7B%5C_%7DTechnology%7B%5C_%7DPerformance.html).
- [19] Howard Herzog, Baldur Eliasson, and Olav Kaarstad. “Capturing greenhouse gases”. In: *Scientific American* 282.2 (2000), pp. 72–9. ISSN: 0036-8733. DOI: 10.1038/scientificamerican0200-72.
- [20] Kojo Menyah and Yemane Wolde-Rufael. “CO<sub>2</sub> emissions, nuclear energy, renewable energy and economic growth in the US”. In: *Energy Policy* 38.6 (2010), pp. 2911–2915. ISSN: 03014215. DOI: 10.1016/j.enpol.2010.01.024. URL: <http://dx.doi.org/10.1016/j.enpol.2010.01.024>.
- [21] R. E H Sims, Hans Holger Rogner, and Ken Gregory. “Carbon emission and mitigation cost comparisons between fossil fuel, nuclear and renewable energy resources for electricity generation”. In: *Energy Policy* 31.13 (2003), pp. 1315–1326. ISSN: 03014215. DOI: 10.1016/S0301-4215(02)00192-1.
- [22] Ayhan Demirbas. “Potential applications of renewable energy sources, biomass combustion problems in boiler power systems and combustion related environmental

- issues”. In: *Progress in Energy and Combustion Science* 31.2 (2005), pp. 171–192. ISSN: 03601285. DOI: 10.1016/j.pecs.2005.02.002.
- [23] Jinqing Peng, Lin Lu, and Hongxing Yang. “Review on life cycle assessment of energy payback and greenhouse gas emission of solar photovoltaic systems”. In: *Renewable and Sustainable Energy Reviews* 19 (2013), pp. 255–274. ISSN: 13640321. DOI: 10.1016/j.rser.2012.11.035. URL: <http://dx.doi.org/10.1016/j.rser.2012.11.035>.
- [24] A. Stoppato. “Life cycle assessment of photovoltaic electricity generation”. In: *Energy* 33.2 (2008), pp. 224–232. ISSN: 03605442. DOI: 10.1016/j.energy.2007.11.012.
- [25] FINK Machine INC. *Understanding emissions of energy sources and calorific values of common energy*. URL: <http://www.finkmachine.com/pdf/emissions-calorific-values.pdf>.
- [26] Isaac Jordán. “A Net Zero Energy Community Microgrid”. PhD thesis. University Of Puerto Rico at Mayaguez, 2017.
- [27] George B Dantzig. “Discrete-variable extremum problems”. In: *Operations research* 5.2 (1957), pp. 266–288.
- [28] The Play Team. *Java Play Framework*. URL: <https://www.playframework.com/documentation/2.5.x/JavaHome>.
- [29] PostgreSQL. *PostgreSQL Features*. URL: <https://www.postgresql.org/about/>.
- [30] DSPACE. *dSPACE - Modular Laboratory Systems*. URL: [https://www.dspace.com/en/pub/home/products/hw/rcp-systems/modular%7B%5C\\_%7Dlaboratory%7B%5C\\_%7Dsystems.cfm](https://www.dspace.com/en/pub/home/products/hw/rcp-systems/modular%7B%5C_%7Dlaboratory%7B%5C_%7Dsystems.cfm).
- [31] SOFTWARE AG. *Quartz library*. URL: <http://www.quartz-scheduler.org/>.
- [32] CronMaker. *CronMaker Tool*. URL: <http://www.cronmaker.com/>.

- [33] Thomas Cormen et al. *Introduction to Algorithms*. Third Edition. Massachusetts, 2009. ISBN: 978-0-262-03384-8.
- [34] Paul E Black. “big-O notation”. In: *Dictionary of Algorithms and Data Structures* 2007 (2007).