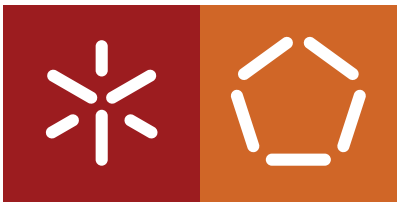

Universidade do Minho
Departamento de Informática

Master Course in Informatics Engineering

Distributed Aggregation Algorithms in Smart Meters

Master Thesis

Telmo Rafael Rodriges Remondes

Supervised by:

Prof. Dr. Carlos Baquero Moreno

Braga, January 2015

"My brother has his sword, King Robert has his warhammer and I have my mind…and a mind needs books as a sword needs a whetstone if it is to keep its edge. That's why I read so much Jon Snow. "

George R. R. Martin in *"A Song of Ice and Fire"*

Acknowledgements

To my supervisor, Professor Carlos Baquero Moreno, for accepting me as my supervisor, for the expertise, careful guidance and counsel provided throughout this work, a special thanks for everything.
To Professor Marco Aiello, for all the fruitful discussions, kindness, availability, counsel and for providing me all the conditions to do my work throughout my time in Netherlands.
To all the members of the Distributed System in the University of Groningen for welcoming so well and for the good working environment provided.
To all the friends i've made in Albertine Student house, for being like my family even though we knew each together for 2 months. To Daniel, Max, Ryan, Moo Jin and Danielle, thank you for understanding all the time i said that I needed to work and for never giving up inviting me to go around.
To my friends in Braga, for understanding the time i didn't show up and for the amazing experiences and stories we shared together. A special thanks to Nuno, Milton and Pimenta for all the support, company and the encouraging words. You made this work a lot easier.
To Sandra, I am extremely gratefully for being with me all the time, for all the support and understanding when i was outside, even when it was most difficult thing to do. Words cannot describe all the support and love you gave me.
To my sisters ,João and Lena a huge thanks for worrying so much and for all the help writing this document. To my parents, Telmo and Jacinta, for providing me everything throughout my live, for making me what I am today and for being my role models.

Abstract

Abstract

Smart Grid is a new concept of power grid, it consists of a connected grid with the introduction of ICT transforming the grid into a group of connect components that communicate with each other in a more efficient an responsive way.
Previous work in Smart Grid research, in most part, consists of proposed solutions to implement it or in pilot projects. Distributed Aggregation Algorithms are algorithms that compute aggregation function in a distributed way in order to reduce the need of resources, they are often used in Wireless Sensor Networks where the devices, sensors, have low computational power.
The aim of this work is understand and answer the question: how these Distributed Aggregation Algorithms perform in a Smart grid context, specially when we consider a network of Smart Meters that are a part of a bigger Smart Grid? We will evaluate the performance of a set of algorithms in a Smart Grid graph representing an example of a dutch Power Grid topology and in another set of Smart Grid graphs by evolving the first by following evolution policies that are expected to happen since the grid is likely to change. Moreover, we will evaluate an test the algorithms considering a real time pricing scenario and measure both speed and accuracy of the algorithms.
Furthermore, we will use Principal Components Analysis to study the proprieties of the graphs that affect the algorithms, in order to know, in the future, what possible parameters should be improved to obtain more accurate results in a faster way.
Our tests suggest that part of the algorithms, *Sketch* based, can be used with good results in accuracy and speed in a normal Power Grid network and that all the transformation/evolution policies improve the algorithm results. Our work also show that as the graphs become more connected and the nodes more close to each other, the algorithms show better performance.

Resumo

Smart Grid é um novo conceito de rede eléctrica, consiste numa rede conectada com a introdução das TIC transformando a rede num grupo de componentes conectados que comunicam uns com os outros em tempo real e de modo mais eficiente.
Trabalho anterior no estudo das Smart Grids, na maior parte, consiste em soluções propostas para a implementar ou em projectos piloto. Algoritmos de Agregação Distribuída são algoritmos que processam funções de agregação numa maneira distribuída de maneira a reduzir a necessidade de recursos, são normalmente usada em redes *wireless* de sensores onde os dispositivos, sensores, têm pouco poder computacional.
O objectivo deste trabalho é compreender e responder a questão: como estes algoritmos de agregação distribuída se comportam num context de Smart Grid, especialmente se considerar uma rede de Smart Meters que são parte de uma maior Smart Grid? Nós vamos calcular o desempenho de um conjunto de algoritmos num grafo Smart Grid que representa um exemplo de uma topologia da rede eléctrica holandesa e em outro conjunto de grafos Smart Grid ao evoluir a primeira rede seguindo um conjunto de políticas de evolução que são esperadas de acontecer já que a rede é propensa a mudar. Mais ainda, será calculado e testado os algoritmos considerando o cenário de preço em tempo real e medindo tanto a rapidez como a precisão dos algoritmos.
Mais ainda, vamos usar analise dos componentes principais para estudar as propriedades dos grafos que afectam os algoritmos, de maneira a saber, no futuro, que possíveis parâmetros podem ser melhorados para no futuro melhorar também o desempenho dos algoritmos.
Os teste sugerem que parte desses algoritmos, baseados em *Sketches*, podem ser usados com bons resultados em precisão e tempo numa rede eléctrica normal, e que todas as estratégias de evolução, melhoram os resultados dos algoritmos. O nosso trabalho também mostrou que a medida que os grafos ficam mais conectados e os nós mais próximos, os algoritmos mostram melhores resultados.

Contents

List of Figures

Abbreviations

AMI Automated Metering Infrastructure

AMR Automatic Meter Reading

LV Low-Voltage Grid

MV Medium-Voltage Grid

ORE Observed Relative Error

PLC PowerLine Communication

SG Smart Grid

SM Smart Meters

WSN Wireless Sensor Network

Chapter 1

Introduction

The power grid is a very important infrastructure in the modern world. The energy it provides is considered of main importance and a basic condition to guarantee minimum life quality. Due to its large size, the power grid consumes a enormous amount of natural resources, making it unsustainable in long term leading to the implementation and usage of new renewable energy sources. This situation leads to a need to modernize the grid since it's mandatory to interconnect them. This evolution, requires the grid to be more sophisticated, eco sustainable and that it properly integrates all the energy sources to enable efficient electrical power distribution. This new concept of grid is called Smart Grid (SG.

SG is a modern power grid that uses computation, information and communication. In an automatic way, SG improves the energy efficiency, sustainability both in power distribution and in electricity production. It enables the grid to become more sustainable because it makes a more efficient management of natural resources. The SG is composed by 'Islands of Automation' interconnected with a communication infrastructure [49].

Smart Meters (SM) are one of the main components of the Smart Grid. They are devices located in the consumers/costumers houses or in industrial facilities that sense the energy consumption. They read data periodically, in short intervals that range from minutes to milliseconds. It can be used for performing statistical analyses that leads to effective consumption forecasting and profiling. This fine grained readings will assist users in achieving a more efficient energy use and adapting to the network status and supply by choosing an appropriate and advantageous tariff [36].

In the next years, the amount of user data collected by the SG is expected to dramatically increase with respect to the current electrical power grid. The amount of *Big Data* collected is important because it leads to a great number of comercial advantages and better energy consumption predictions[39]. Several pilot projects have proposed solutions to deal with these large data size, proposing

and promising the client detailed statistical information regarding their consumption. Yet, these solution can be costly since they require central devices to process and store the data. Other projects, show that it is possible to build a community to collect, store and analyze the consumption by simply using a distributed way to compute it.

Also, more than deploy new grids from zero, it is expected that the current power grid will be improved to become more connected, robust and efficient. In this work, we look at the information collected within the SG, more specifically, the information collected by Smart Meters in the households. This data is very important, not only for billing purposes but also to improve the energy management, enabling it to become more *Smart*, and implement distributed aggregation algorithms, measuring its performance.

1.1 Objectives

There are two types of architectures[36] regarding the SM data aggregation: *decentralized* and *centralized*. In a *centralized* architecture, the meters only sense the energy consumption every specific time and send it to a central data aggregator center. In a *decentralized* architecture the meters sense the consumers consumption and they also perform a partial data aggregation themselves. It's called in-network aggregation[36].
In this work, we will focus on the second type of architecture considering it provides more interesting challenges. The main goal of this work is, considering a *decentralized* architecture, evaluate the performance of a set of data aggregation algorithms that provides relevant information to the consumer and to the electricity producer. Moreover, we will compare the performances of the algorithms, in order to understand if it is possible to use them in a SG network. In addition, as stated before, as the grid will evolve, so, the algorithms will be tested consider these improvements on the grid
In order to achieve these goals, it's important to first understand the various possible *decentralized* architectures and the the role of each component. As we saw in [5] there are some sensors that work as aggregation nodes an others that work as simple nodes. Moreover, it is also important to know in which context and in which scenario it is relevant to use our algorithms, there are many applications for consumption data.
At first, it is important to know how the SG works, how all components interact together and the status of deployed models . Furthermore, it is important to construct a suitable topology for this study, considering a set of meters collecting information about the consumers consumption/production and data aggregators/collectors that aggregate the data in a distributed way. This topology may be constructed

considering real samples of the current power grids. We will study the current distributed aggregation algorithms to know which one we will use to perform distributed aggregation The study of distributed aggregation algorithms embraces the awareness of their functionalities, advantages and disadvantages. It also requires a implementation of then in familiar topologies to understand in a better way how the algorithms behave and also to adquire insight about them.
When we have both the topology and also the algorithms, the next step will be implement them. We are interested in knowing which algorithm provides the best results in speed and accuracy. It is also important to understand which aggregation functions are important to compute in this specific context. Function such as *AVERAGE* or *SUM* may be important, so it is mandatory to choose algorithm that enable these functions. Also, we will improve this topology, following evolution policies that add physical cables or edges to improve the grid in resilience, robustness and efficiency [41]. After, we will test the algorithms in the evolved grids.

In addition, we'll look at all the topologies and the results of our tests, analyze them and understand what parameters of a network should be improved to obtain better results with our algorithms. We will do that by using a statistical methods called principal components analysis.
In the end, we want a intuition about how the algorithms can perform in a power grid scenario, how the proposed evolutions for the current grids can improve or not the accuracy and speed of our algorithms. Finally we want to evaluate insights about how the grids can be improved to obtain better performance using distributed aggregation algorithms.

1.2 Motivation

As stated before, Smart Grid is a new and important concept of grid that is of main importance towards the world energy sustainability. The new needs and urges for integration of the new renewable energy sources, since the new consumer not only consumes but also produces energy too, make the upgrade of the grid mandatory. This new type of consumer will require a deploy of a new type of meters, new data format and new architectures to deal with the differences between consumption data and production data.
With this new model of collection data regarding the costumer information, the communication and collecting process also changes. Aggregation of consumption/production data takes place by other devices. Aggregation summarizes the overall collected data, reducing the computational power required to process the information.

In many pilot projects and solutions, it is chosen to the use of a central data center with high computational power. These solutions often consider a cloud-based service to aggregate the data. However, the vulnerability failure of this single point is a downside. Moreover, the costs associated with implementing this solution, make distributed solution more appealing. SG also leaves the opportunity to create communities to exchange energy, creating small markets in neighborhoods, the costs of a central device to collect the information make it less appealing than distributed solutions for aggregation.The distributed aggregation withdraws the need of a central aggregator with a high processing power. It also enables the aggregation to be more resilient, reliable and fault tolerant since it is distributed and it is cheaper in terms of resources.

Yet, even considering a distributed aggregation and that there are algorithms designed to perform it, there is no work comparing the various solutions. More than compare the current solutions in current power grid topology, we know that Smart Grid is a still evolving concept, therefore, it is important to know what new solutions are valid for using distributed aggregation algorithms and give leads about how the grid should evolve in the future to improve the distributed aggregation performance.

1.3 Document structure

In this document, a state of the art regarding the overall work thematic is presented. In Chapter 2 it is presented the various definitions of the new grid and the point they converge. It is detailed also the infrastructure and model, how the Smart grid is organized and how the diferente components interact. The communication structure and the technologies used on it are also presented, with the various alternatives to realize communication in the modern grid. The important part for this work, smart meters and smart meter network, is detailed. Also, the pilot projects studied are also presented. In Chapter 3 it is referenced the concept of distributed aggregation, some aggregation function and its proprieties. The various aggregation algorithms are referenced with its description. The distributed aggregation within the *smart meters* and WSN is mentioned as well. The definition of *Wireless Sensor Networks*(WSN) is also presented. Smart Metering System could be considered as a specific implementation of WSN so it is important to understand how WSN work and, more important, how in-network data collection takes place. Awareness of this aspects is important considering it's helpful to understand aggregation in Smart Metering Systems. WSN are a concept widely study with similarities with Smart Metering, a bridge between the two concepts are also presented. Although very similar, the two networks have their differences that are presented in the same chapter.

In Chapter 4 it is presented the definition of important concepts we assume in our scenario. Real time

pricing, demand response management, data collector and power line communication are technologies and concepts we used to build our network. Furthermore, our model, scenario and network are explained in this chapter.

In Chapter 5 it is presented the results of our tests in several charts. It is explained in detail what tools were used and explained the structure of our tests and how it was measured the speed and accuracy of the algorithms. In addiction, is explained the evolution policies we consider to add more edges to our first network regarding the evolution of the current grids to adapt themselves to the Smart Grid requirements. Test results in this new generated graph are also presented.

Finally, Chapter 6 contains the explanation of the statistical method used for evaluating the relations between the type of graph and the results of our tests. It is also explained the proprieties that were calculated for our analysis. For each algorithm, is presented a chart in order to comprehend the correlation between computation time and accuracy error and the characteristics of the various graphs.

In the last chapter, conclusions regarding this work and future work are presented.

Chapter 2

Smart Grid

The Smart Grid is a new concept of grid that introduces new technologies into the traditional power system. They enable power grids to become more efficient, integrate other sources of energy rather than traditional ones, and they increase the overall management performance by using modern information technologies. The SG is capable of delivering power in more efficient way and respond to a wide variety of condition and events [32]. Although there are no SGs fully implemented, there are several SG pilot projects showing that the new generation grid pose new opportunities and challenges to both consumers and producers.

There are several definitions for the SG among the literature. For example [32] states that "*SG can be regarded as an electric system that uses information, two-way and cyber-secure communication technologies and computational intelligence in an integrated fashion to achieve a clean, safe, secure, reliable, resilient, efficient and sustainable system*".[37] considers the SG as "*a platform that embraces several multidisciplinary concepts towards computerization of electrical power grids*". The common concept over the literature is that SG main goal is to integrate several components, traditional and new, to achieve better performance, interoperability, energy management and sustainability in long term.

SG creates an environment that introduces a convergence between the infrastructure of generation, transmission, distribution, energy, information technology and digital communication infrastructure that enables the exchange of information and control action among the various segments of the power grid.

As it is possible to notice, this integration means that the SG itself is a very complicated system. Achieving the mentioned goals is a complex task. Due to the variety of problems and challenges, most of the proposed solution and studies regarding the SG focus in some specific aspects. An interesting table that presents a comparison between the traditional grid and the SG is presented in [32]:

Existing Grid	Smart Grid
Electromechanical	Digital
One-way communication	Two-way communication
Centralized generation	Distributed generation
Few sensors	Sensors throughout
Manual monitoring	Self-monitoring
Manual restoration	Self-healing
Failures and blackouts	Adaptive and islanding
Limited control	Pervasive control
Few customer choices	Many customer choices

FIGURE 2.1: Brief Comparison Between the Existing Grid and the Smart Grid

2.1 Smart Grid Model

The SG proposes a new model to the power grid where consumers are no longer passive actors, but *prosumers*[41] that can both consume and produce energy through renewable energy sources. With the introduction of ICT, new actors are now present in the grid that enable new features, otherwise not availabe. Typically, the components in a power grid go one way. In the in SG all the flows of electricity and information go two-ways. These new features enable the operations to become faster and more accurate and the interactions between them are increased resulting that, in the future, everything that happens in the grid can be monitored almost in real time. In a ideal scenario, the SG's new vision, states that a specific component of the grid, such as a household, can both receive energy from the global grid and in the next moment can disconnect from it and become self-sustainable.

There are several visions and models proposed to the SG. One of the most general and accepted model, is based on a vision of actors and their interactions, it is presented the NIST report [25] which proposes a conceptual model providing the main actors towards the SG. Costumers, the end users of electricity,

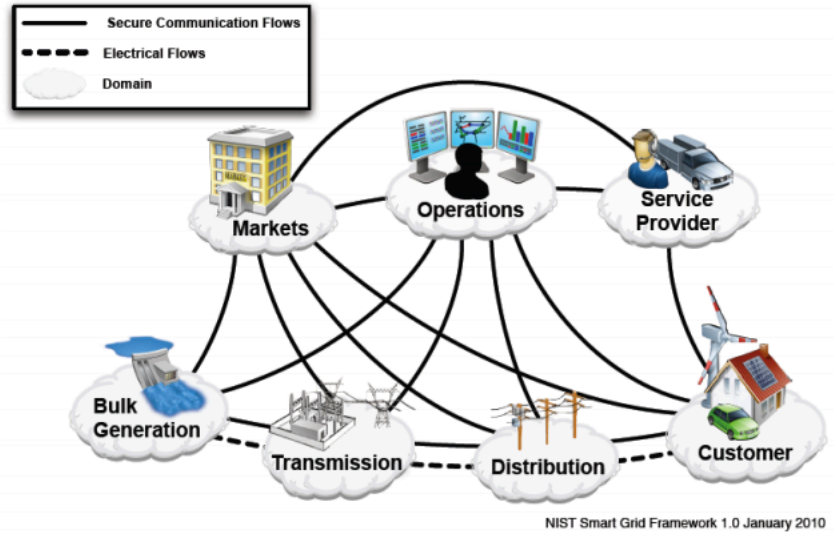

FIGURE 2.2: NIST Conceptual Model for SG

Markets, Service Providers, Electricity Companies, Operations, Managers of the Movement of electricity, Bulk Generation, Generation Centers, Transmission and Distribution of energy. In [32] it is provided a more technical approach where the SG is separated into three major subsystems:

- *Smart infrastructure system* embraces the energy subsystem, information subsystem and communication infrastructure subsystem. The energy subsystem is responsible for advanced electricity generation, delivery and consumption. The information subsystem is responsible for information metering, monitoring and management in the context of the SG. Finally, the communication subsystem is responsible for the communication among the various components and also its connectivity.

- *Smart management system* Provides advanced management and control services and functionalities. [32] considers this system the key reason why SG can revolutionize the grid. Most of the new grid goals are related to energy efficiency improvement, supply and demand balance, emission control etc. These goal are covered by the management systems.

- *Smart protection system* Provides advanced grid reliability analysis, failure protection, security and privacy protection services.

Smart Grids are about improving the current power grid in therms of reliability, energy efficiency and costs while providing a better and more flexibly service to the costumers. These improvements are made possible with the integration of ICT into the power grid, leading to a opportunity for new software applications. In [41] it is stated that Service Oriented Architectures represent the type of software architecture that satisfies the characteristics needed for a SG software: capable of sustaining a set of systems and applications that are diverse, highly distributed and with constrains for security and timing. In [41] it is provided an overall picture that shows the interaction between these type of software and the physical infrastructure.

2.2 Smart Grid Communication

The most important question regarding the communication is "*what network and communication should be used* " [32]? Since there is no standard communication system in SG, several solutions were proposed that are divided into wired and wireless.
Wired solutions are normally more costly to implement than Wireless, mainly because of the need, in some cases, to install or deploy from zero a physical infrastructure like electrical cables. Wireless communication can be a better option in terms of costs, time to deploy and it is normally more suitable

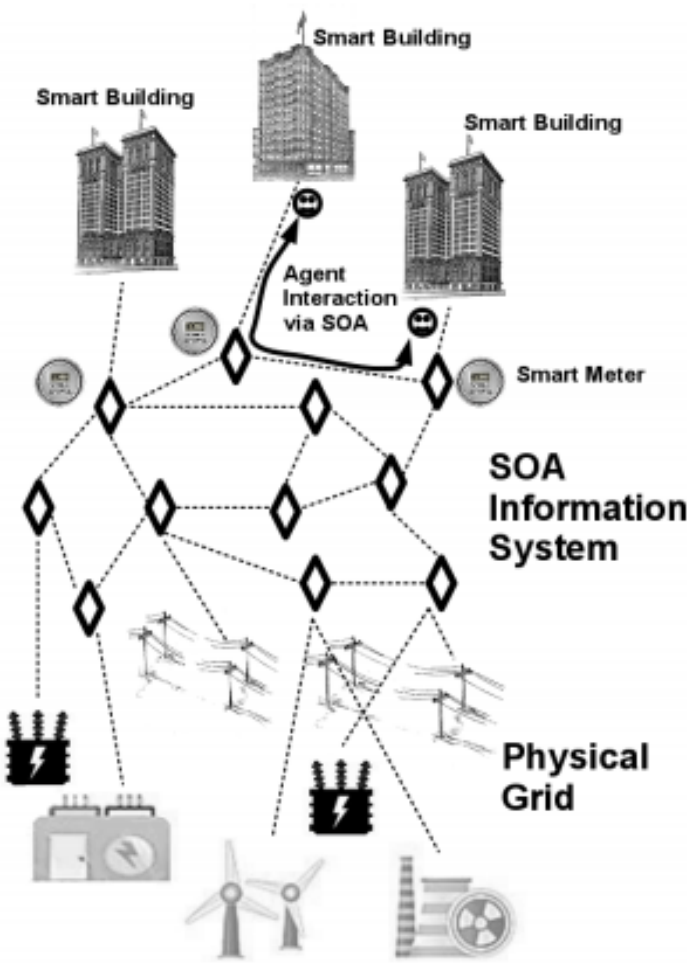

FIGURE 2.3: Smart grid physical and information infrastructure

for remote end applications [19]. However, they lack some performance compared to wired solutions, specially in speed. Also, the costs of deploying an wired communication infrastructure can be reduced if they are implemented in a existing infrastructure. It is the case of power line communication, it uses the power cables.

There are several wireless possibilities for communication.

- *Wireless Mesh Network* (WMN) is a communication network made up for radio nodes organized in a mesh topology [32]. It increases reliability and automatic network connectivity, has a large coverage and high data rate.
- *Cellular Communication Systems* GSM and 3G. Useful in case of low computation power devices such as the meters. It is quick and low-cost to obtain data communications coverage over a large geographic area [16]. There are several solutions that use a Short Message Service communication to send the meters data.

- *Wireless Communication based on 802.15.4* ZigBee is a wireless communication that is recommended to be used in SG considering the IEEE 802.15.4 protocol stack[19]. ZigBee is designed for radio-frequency applications that require low data rate, long battery life, and secure networking. Selected as the communication technology for the smart metering devices [18] because it provides a standardized platform for exchanging data between smart metering devices and appliances located on costumer premises[32]. WiMax, WirelessHART and ISA100.11a are other examples of wireless communications based on the IEEE 802.15.4 protocol.

Other examples of wireless communication are satellite, cognitive radio and microwave communications. Fiber-optic communications and Power-line Communications are some of the wired communication possibilities. Power-line communication has the advantage of being already installed so the cost of deployment is less expensive than other wired solutions. Fiber-Optic has also the advantage of being faster than wireless technologies but its downside is that it can be more expensive to deploy. In this case, there is the need to implement from zero in an infrastructure that lacks cables with that sort of technology.

2.2.1 PowerLine Communication

PowerLine Communication (PLC) is a communication technology which uses the installed electrical cables to transmit data. It has been the first choice for communication with the electricity meter due to the direct connectionand successful implementation of AMI in urban areas where other solutions struggle to meet the need of utilities[26].

The typical application of the PLC is to connect the smart meter to the data concentrator. The smart meter communicates the consumption or production data to the collecting device through the power line. After the collection from the meters, normally, the data aggregator sends the stored data to a central facility, it is owned in most cases by the electrical company. The communication goes through a more fast technology like optic fiber. In Europe, the majority of the transformers serves 200 customers or more, so the data collectors are located in these transformers in the Low-Voltage (LV) side. In the USA, the concentrator or aggregator is often in the Medium-Voltage(MV) side of the grid due to the low number of end points per transformer. On cpntrary, the large number of end-points per transformer in Europe avoid the need to locate the concentrator up in the substation, in the MV side, it can be conveniently located on the LV section of the grid. REMPLI project tells us that unlike solutions based on ZigBee or Wifi, PLC-based AMI have a proven track record of being able to avoid network congestion when cooperative schemes are employed[24]

As it was stated before, one of the main advantage of using a PLC based communication infrastructure is the cost to deploy. The fact that the cables are already installed makes PLC a more appealing technology compared to other wired solutions. Considering only the costs, PLC can even be compared to the wireless solutions[26].
The standardization efforts on PLC networks, the cost effective, ubiquitous nature and widely available infrastructure can be the reason for its strength and popularity. It is also well suited for urban areas and for the smart grid applications besides AMI. PLC can also be used to monitor and control, the involved areas are already covered.
Although its popularity, PLC still has some disadvantages due to the nature of the channel and some technical problems. The communication channel is a harsh and noisy environment, making it difficult to model. Furthermore, the network topology, the number and type of the devices connected to the power lines, wiring distance between transmitter and receiver, all of these factors affect the quality of the signal that is transmitted over the power lines. However, recent modulation techniques and technologies are surpassing the mentioned difficulties, mitigating the noisy environment by reducing the data rates and the bandwidth.

2.3 Smart Information SubSystem

This part of the SG refers to the whole information that is collected by sensing the consumers consumption and its management . The data collected is often used for billing, grid status monitoring and user appliance control [32]. It is aggregated and collected, afterwards, *smart management* is ideally performed on the data.
An important concept in the information subsystem is the *Smart Metering* and the Smart Metering System or Automatic Meter Reading AMR. This system is responsible for collecting the data from the measurements that are performed by the SMs.
Other part of the Smart Information SubSystem is the *Smart Monitoring and Measurement* that can be approached by either *sensors* or *phasor measurement units*(PMU) which are a specific type of sensors. General *Sensors* are used for detecting failures, tower collapses, hotspots and extreme mechanical conditions. They can also provide real-time diagnose of the grid status. PMU's are sensors that measure the electrical waves on a electrical grid to determinate the health of the system. These systems collect information regarding the status of the grid in order to monitor it and detect failures and outages.
The Smart Metering Systems only collects data from SM's and it only embraces the management of

that data.The management refers to the whole information analysis and modeling, integration and optimization.In this specific part of SG, there are several areas of research that represent a new set of opportunities.

2.3.1 Smart Meters

Smart meters are devices that sense the energy consumption. They are installed in the costumer side, households or in industrial facilities, depending on the costumer nature. Playing a major role in the information subsystem, smart meters present several number of challenges in sensing and analyzing[36]. The Smart Mettering System, or AMR(Automatic Meter Reading) is referred in [28] as the technology whose goal is to help collect the meter measurement automatically and possibly send commands to the meters.

As referred in the previous section, the main function of a smart meter, and all meters, is sense the consumption in the costumer side. The feature of sending, allocate and aggregate the information that comes from many meters allows a company to remotely read the consumers' consumption at each household, without the need of actually going to the premises and notifying the costumers[49]. Jorge Vasconcelos [13] enlightens in his work the potential benefits of the smart meters, for example, the potential benefits for customers are customer awareness and energy saving, more accurate meter reading, billing, better service quality, greater tariff variety and flexibility, improved conditions for vulnerable customers, easier comparability of offers and it is easier to change supplier. [28] states some benefits of the smart metering system: Real time pricing, power quality measurement, automated billing, load management, remote connect/disconnect, outage notification and bundling with water and gas.

Privacy and security are important concerns when dealing with the sensed information. There are many privacy issues considering that external parties access the consumer energy consumption. Some are authorized parties, but there is a risk of an unauthorized access of this data, leaving to some security and privacy dangers. For example, by analyzing the data, one could determinate which devices are plugged in at some specific time, giving for example information about if there is someone in home or not. Many pieces of work propose solution to securely store this sensible information. Although privacy and security are out of the scope of this work, it is important to mention it.

2.3.2 Demand Side Management and Real Time Pricing

Demand Side Management refers to a series of strategies and policies to load balance the demand efficiently in order to prevent very high demand peak hours, the overload of the production and the

occurrence of blackouts or faults. Although applied in the SG and AMI scenario, it is not a recent trend, demand side management has been considered since the early 80's [20].

One of the applications of Demand Side Management in the Low-Voltage Grid is Demand Response(DR). It refers to the ability to make demand able to respond to the varying supply of generation that cannot be scheduled deterministically. Demand response is a mean to alleviate peak demand and bring more awareness on energy usage to the consumer. It is believed that DR will allow a better control of peak power conditions, maximize the use of available power and increase the power system efficiency.

A more specific example of how DR can be applied is with real time pricing which is one of the strategies to achieve and realize Demand Side Management. Real time pricing is a pricing policy that basically consists of upgrading the prices accordingly to the demand and the supply. In other words, the prices change as the demand increases or decreases along periods of time. This policy tend to persuade the costumer to reduce their energy consumption during the peak hours, since the price is higher, and increase slightly outside the peak hour, when the price is lower.

In [21] is realized a research regarding the aforementioned policies in UAE. The experience states that costumers react better to peak time adaptive tariff. Also, it is stated that with these pricing policies, costumers are more aware of their consumption and, for consequence, more willing to reduce their consumption during peak hours. The authors also point that consumption demand is reduced during the peak hour while balancing it.

[43] proposes an optimal pricing policy for aggregators. The aggregator "buys" the electrical energy from the supplier, and regarding the necessary demand, defines the price. The time is spited in K time intervals, in the k interval the aggregator must make decisions based on the $k-1$ interval information about demand and energy available from the electrical company. The goal is to maximize the profit, but it can also be applied to reduce the demand during peak hours if the decisions are made for reducing the demand when the supply is low and the demand high. The behavior is similar to the PowerMatching city, but the aggregator has instead information about bids to buy or sell energy.

2.3.3 Data Collector/Aggregator

Data collectors or data Aggregators are devices installed in the Low Voltage side of the Grid, in the European case, or in the Medium Voltage side of the grid in the American case due to the SM per transformer differences between the two locations. They normally serve as data storage where a set of SMs are directly connected to it sending the consumption or production of a household.

Data Collectors normally serve as a storage but this is not a rule that apply to all. There are cases

where the data aggregators perform more operations. In the PowerMatching City project case, the data collectors receive data in bid format, to buy or to sell and then calculate the price of the energy to others buy and sell. They work as a market, not only collecting information, but also making decisions about prices. In the case of InovGrid, the data collectors just collect data. Some other projects lack the existence of these devices, using instead a cloud based service, while the remaining projects studied use the collector inside the house to aggregate data from different electrical devices to know which ones are consuming the most. SG projects which don't require the use of any data collector propose a architecture that enable the smart meters to directly send their data to a data center through a cloud based service. These different architectures exist because there are different goals in the different pilot projects. The ones which use big data to perform detailed statistical report to raise awareness in the costumers require the maximum amount of data as possible. In the cases where the goal is to achieve a ideal strategy to buy/sell energy, provide a reasonable Demand Side Management and useful information to the operation side doesn't require a big amount of data, so they use a mid level set of devices to aggregate the data.

2.3.4 AMR and AMI

As referred in this document, the smart metering system is composed by smart meters that sense the energy consumption and send their data to a Gateway or a Data Collector. It can also be defined as AMR or AMI. In [36], the AMR is described in more detailed as an "*technology of automatically collecting diagnostic, consumption and status data from energy metering devices and transferring that data to a database for billing troubleshooting and analyzing*".

The Automated Metering Infrastructure is a more sophisticated version of the traditional AMR, it provides two-way communication, enabling more control of a smart meter behavior. Therefore, all of the meter information is available in real time, allowing improved system operations and costumer power demand management[36]. AMI has also the ability of reconfiguring to adapt to communication failures, perform outage management and reporting, service connect and disconnect and it also enables time stamping of meter data [12]. AMI is built upon AMR.

Current SM enable two-way communication, an important part of the benefits that come from the usage of this new meters, comes from this two-way communication, also it is not only important for behavior control and outage detection, it enables the realization and implementation of in-network algorithms.Normally, the SG contains an AMR which has an AMI built upon it, enabling two-way communication.

In the pilot projects studied, all metering systems were composed by the smart meters. The common behaviour is that in a pre-defined period of time, the devices send the consumption data. As it was previously described, in part of the projects, a cloud based service is used for the SM to send the data . In other cases, substations that work as data collectors are used to concentrate the consumption information from the SMs connected to it, usually a whole neighborhood, afterwards, the data from the substations is sent to a data center. In small SGs, there is no central data center, the substations communicate with each other.

2.4 Wireless Sensor Network

Wireless Sensor Networks(WSN) are *ad-hoc* networks composed by tiny devices with limited computation and energy capacities. These tiny devices, sensors, are called tiny because of their low capability of computation, communication and storage. The WSN low-cost sensors monitor physical properties on environmental conditions, such as temperature, sound, vibration, pressure, monitor pollutants and cooperatively pass their data through the network to a main location(sink node) via multi-hop wireless links[35] or to their peers.

WSNs act under severe technological constraints: individual sensors have severely limited computation, communication and power(battery) resources and need to operate in settings with great spatial and temporal variability.The ad-hoc nature of a WSN implies that sensors are also used in the network infrastructure, i.e., not just sending their own data and receiving direct instructions but also forwarding data for other sensors. Modern networks are bi-directional, enabling control of sensor activity but some WSN could not have bi-direccional communicatin due to low computation power of the sensors. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance.

Today, WSN networks are used in many industrial and consumer applications like industrial process monitoring and control, machine health monitoring and so on. Some of WSNs requirements are: large number of nodes, low energy use, network self organization, collaborative signal processing and querying ability.

WSNs are becoming increasingly popular in many spheres of life [8], they also have the capability of forming the sensor web services which can be considered as an extension of the future internet towards smart devices, Internet of Things(IoT)[35].

2.4.1 WSN and Smart Grids

Considering the application of WSNs, they can also be applied in the SG. Furthermore, the AMR could be considered as a specific example regarding the usage of a WSN. One can implement WSN solutions for data aggregation in AMR .

WSN has been widely recognized as a vital component of the electric power system[33]. WSN contains a large number of low cost and multifuncional sensor nodes which "*can be of benefit to electric system automation application, especially in urban areas*"[22]. The collaborative and context-awareness nature of WSN brings several advantages over traditional sensing including great fault tolerance, improved accuracy, larger coverage area and extraction of localized features [33]. Sensor nodes can monitor the overall network.

WSN could apply to several features in the SG: basic measurement, smart voltage sensors, smart capacitor control, smart sensors for outage detections and weather condition sensors, distributed generation, smart gird storage and, referenced before and more importantly for this work, WSN for AMI(Advanced Metering Infrastructure) or AMR. A specific example is in [33] where a WSN could apply perfectly to a household or House Area Network(HAN) . ZigBee is a communication technology often choosed in Smart Grids due to its reliable wide area coverage and predictable latencies. It is also a suitable choice for a Local Area Network such as a household or a neighborhood. As an example in [33], a WAMR(Wireless Automatic Meter Reading) can determinate real-time energy consumption of the customers by sensing each device that has a wireless sensor on it. The smart meter within the household implements an interface that translates, summarizes and aggregates data of power usage and presents it to the power utility.

Other examples of WSN appliances in SG are found in [33]. WSN could apply in Power Delivery and in Power Generation as well since the sensors can monitor the deliver systems, in the first case, and monitor the energy generated in the second case.

Although very similar, there are some differences between WSN and Automatic Meter Reading. Such diferences are stated in [28]. For example, individual consumption measurments must preserve its information. In WSN, sink doesn't care about indivudal data but in AMR, aggregation nodes must preserve the unique measurments, plus, the meters must have a unique indentifier that links the smart meter to a household/costumer/producer. Also, Smart meters have fixed positions opposite to some WSN, base stations may need to disconnect/connect to a specific costumer. Even in security, there are some differences. The main security concern in WSN is to preserve the privacy of data, in SM, altough privacy is an important issue, integrety of data is the main concern.

WSN, even considering the diferences to AMR, provide a variety of solutions and gives some insight

to understand and comprehend the problem of distributed aggregation in AMR since WSN is a well studied subject.The topology we can find in some WSN can apply to the ones in the AMR. So, even with differents communication infrastrutures or different computation powers, from the topological view, both networks are very similar.

2.4.2 Distributed Data Aggregation in WSN

Distributed Data Aggregation in WSN is a widely studied subject, with several works and proposed solutions. Distributed aggregation adquires a special importance in WSN, since the sensors are low resource devices, the effort to implent distributed solutions is quite mandatory. The aggregation techniques reduce the amount of data communicated within a WSN and thus conserves battery power [8].Periodically, as measurements are recorded by individual sensors, they are collected and processed to produce data representative of the entire WSN. A natural approach is consider that the sensor send the measured data to special sensor nodes, i.e., aggregator nodes [8]. In *in-network* aggregation nodes forward the aggregated data to a sink that store it.

An example of in-network aggregation in WSN is in [8]. In this model, it is assumed that all nodes are

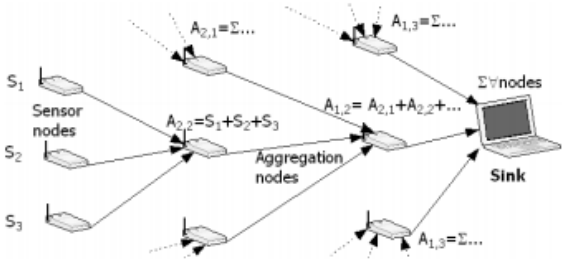

FIGURE 2.4: Principle of in-network aggregation

potential aggregators and that data gets aggregated as they propagate it towards the sink. The aggregation proccess is simple, it doesn't involve any expensive or complex computation. The aggregation requires all sensors to send their data to the sink within the same sampling period so there is a need for a global clock so that all node can synchronize. Another study is in [5], where a special kind of distributed aggregation is proposed, *Concealed Data Aggregation.* This type of aggregation is defined as an approach than promises the combination of end-to-end security and *in-network* aggregation. In [10] it is assumed a general multi-hop network with a set $S = s_1....s_n$ of n sensor nodes and a single base station R. The aggregation is performed over an *aggregation tree* which is the directed tree formed by the union of all the paths from the sensors nodes to the base station. Another WSN distributed aggregation scenario is presented in [15]. The network model consist of a n sensor nodes and one base

station that is also called a sink. Each sensor node can send or receive data to or from all directions. It is assumed that all nodes have the same transmission range for simplicity. A node can either receive or send data at the same time. It can only receive a data packet correctly when it hears this packet at that moment.

2.4.3 Smart Metering Aggregation Model

The two main architectures for smart metering considering data aggregation are *centralized* and *distributed* or *decentralized*[36]. In *centralized* fashion, the meters just sense the data, afterwards, it is sent to a central aggregator with higher computation power that holds a central database. In a *de-centralized* way, the aggregation role is distributed among several meters, not all of then. This type of aggregation is called *in-network* aggregation [5][7]. The aggregation node in this scheme communicates the calculated energy consumed to an appropriate party such as a energy producer. Typically, this communication occurs once per billable period [36]. As introduced before, the architecture chosen for this work is *de-centralized* due to the nature of the aggregation algorithms.
Usually, the *centralized* approach is composed by a cloud service provided to the costumers to store their consumption data. These approaches make use of a data center and uses this architecture so they are capable of storing big quantities of data and perfome Big Data techniches on it, providing useful information for both energy companies and costumers.
One example of the *de-centralized* architecture for aggregation is in the PowerMatching City of Hoogkrek. The goals are different than providing statistycall information regarding the costumers consumption, in the Hoogkrek case, the goal is to form a market in a microgrid that enables the household to be self-sufficient in therms of energy. They use the *de-centralized* archictecture to form several points of this market where households sell and buy their energy. The nodes that concentrates the offers communicate with each creating the city energy market. Another example is in Rottondi *et al*[31], the overall scheme is presented in 2.5, where a set of meters are connected to gateway sharing information, the central station only works to set the aggregation rules.

2.5 Smart Grid Projects

So far there aren't standards to realize the Smart Grid, as it was aforementioned, not even a complete and specific definition about what is a Smart Grid. Even without a specific definition regarding the standard model and communication, there are common concepts that are well accepted and visions that are transversal. The introduction of communication and information technology into the grid, the idea of a consumer that is not only a consumer but also a small producer that can supply the grid and

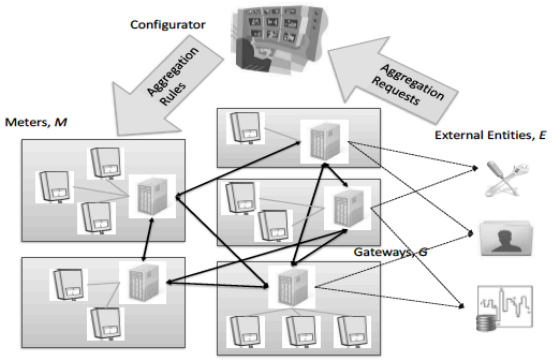

FIGURE 2.5: The functional nodes of the architecture

itself, with electrical energy, the remote control of the components like electrical cables, stations and more, these are ideas that are likely to be features that all the future SG will have. In order to understand how is the status of the deployed SGs and the directions they will take in the future, we analyzed several pilot projects and companies that are currently trying to implement the new grid.

2.5.1 Opower 4

Opower[47] is a company that promises to help costumers to reduce their energy consumption. They provide a cloud based service to gather data regarding the costumers information about their energy consumption. Using big data and behavioral science they provide reports to the costumer with their consumption history in the time period that report is about. Also, the reports give tips and advices where the consumer can reduce the energy consumption, and therefore, reduce the energy bill.
One of the version of the promised platform, one of the most recent, is called Opower 4. Opower 4 works as a service platform, is a *Software as a Service* platform. The model is like the general model for the SG Information subsystem described in Section 2.3. Households using this services have a smart meter installed, every 15 minutes, the device sends the data to a cloud through a cloud based service. The collection of the data is only made in one point, in a Data Center that concentrates all data. There is no reference regarding the number of data centers that the company uses. Big Data analysis is performed in the data. Mainly, as it was stated, the platform exists for billing proposes and to raise awareness in the costumers so that they reduce their consumption with reports that have statistics of each household. For example, one costumer comparison with the neighborhood and what devices are consuming more or less.

2.5.2 DEHEMS Project

The DEHEMS project [48] is an infrastructure that reasons about the household's energy behavior and tests the effectiveness of various persuasive techniques . The system receives energy information from several sensing devices, including the ones that sense electrical consumption. The special feature about the DEHEMS project is that it uses Informix TimeSeries [45] that is a builtin feature of Informix, a database type from IBM, that adds support for managing time series (timestamped) data, this feature is specially important in the management of data regarding the reading of the meter.
This project operates in the distribution grid. The model also includes de household, where several sensors are installed, not only for electricity, but also for gas and water. Each sensor, with a 433Mhz Radio, takes reading every 6 seconds and sends it to a DEHEMS Gateway that aggregates all the information about the house. The data of all the Gateways is concentrated in a Informix database so that big data operations can be performed on it.
The goal of this project is the same as the aforementioned project, raise awareness in the costumers by generating statistics abou each household consumption(CO2 emissions, cost of the energy, history of consumption and comparison with the other consumers) and send it to the consumer.

2.5.3 Pecan Street Project

The Pecan Street Project[46] is a research project in Smart Grids by Pecan Street Inc., a University of Texas based research organization. Started in Austin an then expanded to other cities and states. The focus of the research is mainly in the information subsystem of Smart Grid. One of the project goals is to understand how to lower the carbon emissions by learning how energy is being used among homes. But understanding the how is only half of the challenge: Pecan Street also seeks to understand what homeowners need in order to manage their energy use.
It operates in the distribution grid and it works in a similar way as the DEHEMS project. In each house, there are several sensors installed to sense the consumption in each device, for example the Air Condition System. The sensors send their data every 2 seconds to the gateway and then to the smart meter every 15s, the meter sends the collected information from the gateway to a data center every 15 minutes. The gateway performs an estimated average of all devices connected to a sensor. The consumption data is in the end used for statistical analysis to produce results about the consumer energy consumption. With this information, the Pecan Street Project staff pretends to lead their costumers to use their energy more efficiently.

2.5.4 Smart Meter Data Stream in the Cloud

This is a solution proposed to handle the SM data for real time pricing in a distribution electrical grid which is in [30]. The simulation considers a set of 1 million meters connect via TCP/IP to a data center/Cloud. Every second, each SM sends a package containing information about the electrical consumption of an household. The cloud model is composed by layers. Since every moment a new package arrives, it is like a stream, so several stream tasks are created in the lowest layer to handle the incoming data. In the upper level, within the cloud, aggregation tasks are created and they work in parallel to handle the information that comes from the lower level, the stream tasks. As the traffic increases or decreases, more aggregation tasks are created or deleted accordingly. In the paper simulation, 2 aggregation task were created. In the highest lawyer there is one real time pricing task that has the role of updating the energy price according to the amount of the energy consumed.
This paper offers a solution to handle smart meters data to provide a realtime pricing policy to balance the demand of energy and also to continuously monitor the meter, mainly to prevent outages and blackouts during peak time.

2.5.5 Inovgrid/InovCity

Inovgrid[40] is a project powered by EDP, Energias de Portugal, that pretends to modernize the portuguese electrical grid, more specifically, the distribution grid. In other words, the project aims to transform the traditional grid into a smart grid by adding information and communication technology. This is still a pilot project, there is no mass scale attempts yet to fully implement. So far, there is only a pilot project called the InovCity in the city of Évora that consists of a smart grid small experiment, with the installation of several smart meters and sensors in some of Évora households.
In further detail, the InovCity model can be explained by dividing the grid into three smaller networks: a Home Area Network(HAN), a network in each house, where each device has a sensor that communicates with the smart meter installed in the house. In the set of devices that composes the HAN, electrical vehicles are also included. Local Area Network, a set of households, a neighborhood connect to a DTC/substation that communicates through the electrical cables, PLC Prime and LMS protocol. Finally, the wide area network that embraces all the other minor networks
In terms of number, in InovCity 300 000 Smart Meters and 300 DTC/Substations were installed. The Smart Meteres communicate the consumption of an household every 15 minutes to a Substation which therefore communicates to other substation and finally to a central facility. Each substation has the capability of performing data analysis and process data function, so, depending on the type of analysis, the substation can perform it locally. Basically, the whole collection of data works in an hierarchical

way, the data is collected in every smart meter regarding the information about every device with a sensor, a substation aggregates information about the houses connected to it, the upper level of substation collects the information of the other substation connected to it in lower levels, and finally the central facility concentrates all, working as a "sink".
There are 3 goals the company claims to achieve with the InovCity architecture. More energy efficiency by raising awareness in the clients with detailed information about their consumption. Increase Operations efficiency and reduce its costs by remotely perform any needed operations from a central station instead of doing it locally. Finally, commercial benefits by having a real time consumption instead of an estimated one and more accurate control by having a real time alarm of a failure in a SM..

2.5.6 PowerMatching City

PowerMatching city[44] is a pilot project of a self sustainable micro smart grid implemented and tested in Hoogkrek, a town in the north of Netherlands. Opposite to the other examples, the goal is to create a self-sustainable city when it comes to energy consumption. The costumers can buy and sell their energy. They buy it from a market that is composed by small local producers that can generate energy through renewable sources. This way, the city becomes independent from major electrical companies.
In PowerMatching, each household has a smart meter that has the information about the consumption of each device and also about the energy produced. The information is sent by the smart meter to a coordinator/data collector through an VPN communication infrastructure. Also, an ADSL communication channel is used between the coordinator and the houses connected to it to prevent the occurrence of faults. The coordinator is responsible for collecting the information about the energy consumed and produced, and generating the prices accordingly, working as a market. This idea can scale adding more coordinators that are connected and communicate among each other to work as a whole market.
In the implementation in Hoogkrek, 25 Households had an SM installed to the PowerMatching city network with, at least, one collector/coordinator. Data is collected in every coordinator station, therefore, the process in the station works in a lawyered scheme, the lower levels receive and collect the information, send it to the upper level in the bid format, to buy or to sell energy that are communicated to every house connected. It is not mentioned what is the interval by which the prices of the energy are changed, but we can admit that it is not consider peak hours of high demand of energy, but in terms of supply and demand as in all markets.
Also, the system contemplates three web portals for data: user Portal where the user can check her/his stats about energy consumption/production, operator Portal which is mainly used for operations(monitoring

and detecting failures, data analysis that generates reports used mainly for research proposers and for the development of the project.
The main goal is to organize a market where the community is independent from global companies. The measured data is used and aggregated mainly for price proposes, i.e., following the market rules, the data is used to give selling and buying prices. Also, in terms of singular house, the data is used also by the system to buy or sell the energy. If a house has low energy supply plus high demand, the systems should buy it, on the other hand, if there is a surplus, ideally it should be sold it to the market.

There are other examples of other smaller SG models that represents more an idea than an deployed project. Keita Suzuki *et al* [42] presents a particular case in a office building in Japan(Heating ventilation and air conditioning facilitie,HVAC) where existis the need to aggregate power curtailments from hundred or thousands of distributed HVAC facilities. Several smart meters where placed, connected to a Gateway that receives the consumption data for daily or monthly billing. The Gateways are connected to a central ADR, Aggregation Cloud, which aggregates all the consumption.
Another work using a *decentralized* way is in Rottondi *et al*[31]. The smart meters generate the energy consumption measurements, the Gateways securely aggregate the metering data and external parties access the aggregation results. Each meter is directly connected to a Gateway, receiving data from a limited number of meters. At regular time intervals, 15 min in this case, the meter generate a measurement and sends it to the Gateway.

Chapter 3

Distributed Data Aggregation

3.1 Definition

Data aggregation is a technique that , on its basis, consists in reducing the amount of data collected and the resources needed to process it. According to [27], data aggregation is considered a subset of information fusion, that aims at reducing the handled data volume. A more precise definition is given in the same report:

Definition 3.1. An aggregation function f takes a multiset of elements from a domain I and produces an output of a domain O.

$$f : \mathbb{N}^I \rightarrow O$$

The order in which the elements are aggregated is irrelevant and a given value may occur several times. The main goal of data aggregation, "*the aggregation function aims to summarize information. The result of an aggregation takes less space than the inputed multiset (element from* $\mathbb{N}^I$*)*".
Distributed data Aggregation or *in-network* aggregation tends to distribute de computation of an aggregation funtion among several nodes in the network. In contrary of a *centralized* architecture, where a central node computes all the data and performs the aggregation function, a *decentralized* aggregation distributes the data computation, hence the effort to compute the aggregation function is reduced.

3.2 Distributed Data Aggregation Algorithms

Distributed Data Aggregation Algorithms are protocols used to compute an aggregation function in a *decentralized* way. They are used and more suitable when the network lacks a node or component that has the computing capacity to process large ammounts of data. This is the case of WSN , were all the nodes are tiny devices with low storage and proccessing capacity.

In [27] is also presented a simple taxonomy of the existing algorithms that perform distributed data aggregation. First is analyzed the algorithms from the communication perspective, i. e., the routing protocols and the intrinsic topologies, afterwards, is analyzed the computation issues, how the aggregation functions are computed by the algorithms.

3.2.1 Communication

3.2.1.1 Hierarchy-based approaches

Traditionally, existing aggregation algorithms operate on a hierarchy- based communication scheme. This is a *structured* communication scheme. It is required to know in advance the topology of the network. A hierarchycal communication tree is constructed, with several levels of nodes. In the root of the tree there is a main repository of all data, denominated as sink. Besides the sink, other special nodes can be defined to compute intermediate aggregates, working as aggregation points that forwards their results to upper level nodes. There are generally two main phases, *request* phase, corresponding to an aggregation request spreading through all the nodes, an the *response* phase where all the nodes respond to the request sending their aggregation results. Some specific examples of these kind of communication are presented.

3.2.1.2 Gossip-based approaches

This type of approach is referred as an *unstructured* approach, contrary of the aforementioned *structed* approaches. In this type of scheme there is no previous knowledge of the topology of the network or any specific structure. The information or messages are commonly disseminated across the network without following any specific topology, the information is passed from a node to one or many nodes, like a infectious disease or a gossip,i.e., an ”infected” node sends a message to a random subset of nodes. This type scheme tends to allow a robust (fault tolerant) and scalable information dissemination all over the network[27]

3.2.1.3 Hybrid approaches

Hybrid approaches propose a solution that merge both hierarchic and gossip-based approaches, using the high accuracy and efficiency of the hierarchic based schemes and the robustness of the gossip approaches. In the disadvantages of one approach, the other one has it as an advantage. Hybrid approaches aim to merge the advantages of both schemes to eliminate both disadvantages.

3.2.2 Computation

3.2.2.1 Hierarchical

The input is separated into groups so it can be computed in a distributed hierarchical way. It depends on the previous formation of a communication structure such as tree or cluster. Some node work as *forwarders*, just forward data to upper levels of the hierarchy, and others work as *aggregators*, apply the aggregation function directly to all received input and then works as a normal *forward* node. This class of algorithms allows any decomposable function with high accuracy without the presence of faults. Algorithms of this class were aforementioned.

3.2.2.2 Averaging

This class of computation scheme is based on an iterative computation of partial aggregates, where all nodes share their results among the network and all of them contribute for the final result. This scheme provides high accuracy, considering that all nodes converge to the same result. However, in order to converge to the correct result, the algorithms must respect an important principle commonly designated as ”mass conservation”. [27] describes ”mass conservation” as an invariant, stating that the sum of the aggregated values of all network nodes must remain constant along time. One example of algorithms of this class, is among the ones with gossip base communication scheme, since the results of the aggregates could be shared randomly with the neighbor nodes. Due to its nature, Averaging algorithms tend to be highly robust, i.e., tolerant to faults on contrary the structured algorithms. Decomposable and duplicate sensitive functions can be computed in this class.

3.2.2.3 Sketches

Sketch based algorithms are proposed in [4] that use small sketches. Based on the probabilistic counting sketches technique that estimates the number of distinct elements in a data collection and it is described in further detail in [11]. It is based on the use of an auxilary data structure with a fixed size that holds a *sketch* of all network values It uses two phases in the communication proccess: the sink propagates the aggregation request across the network and then the results are collected back to the sink. In the first phase, all nodes compute their distances to the root, in the second phase the partial aggregates are computed across the routing structure, using the adapted counting sketch scheme, and send to the upper levels in successive rounds.
Input values are used to create *sketches* that aggregated across the network, using specific operations to update and merge them. The aggregation could be done using multiple paths. This type of algorithms enables operations out of order and enables duplicate insensitive functions. The computational cost of this class depends mainly on the resources used to produce the result by the estimator and the complexity of the operations to produce the *sketches*. This kind of algorithms tend to be very fast, depending on the dissemination protocol used to propagate the sketches, but lack accuracy because they are based on probabilistic methods.

3.2.2.4 Digests

This class of algorithms allowd the computation of more complex functions like median or mode than the normal aggregation function such as SUM or $AVERAGE$. This algorithms produces a *digest*, data structure with a bounded size that holds an approximation of the statistical distribution of input values in the whole network, that summarizes the system data distribution, an histogram. The accuracy of this class of algorithms depends mostly on the quality and size of the obtained *digest*. Usually it requires more resources.

A overall taxonomy table is presented in [27] Considering the table in 3.1, we select the algorithms to use in our tests. Hierarchical and sampling algorithms are not selected because they provide results only at one node, also, sampling algorithms are used mostly for evaluate the size of the network and that is not the kind of function we measure and compare in this work. Also, algorithms based in digests are often used to test a distribution of values in a network or to compute complex aggregation

	Advantage	Disadvantage	Requirements
Hierarchical	- accurate (without faults); - very efficient (messages);	- result at a single node; - not fault-tolerant;	- specific routing structure (e.g. spanning tree);
Sketches	- very fast; - result at all nodes; - fault-tolerant;	- less accurate;	- local knowledge of neighbor IDs, or global UIDs; - source of randomness;
Averaging	- accurate; - result at all nodes; - fault-tolerant; - churn support;	- less efficient (messages);	- local knowledge of neighbor IDs;
Sampling	- efficient (messages);	- not accurate - result at a single node; - not fault-tolerant	- global UIDs; - source of randomness;
Digests	- computation of complex aggregates; - result at all nodes;	- less accurate; - resources needed (e.g. larger messages);	- local knowledge of neighbor IDs;

FIGURE 3.1: Summary of the characteristics of main data aggregation classes

function, these problems are out of the scope of this work. We choose algorithms based on *Sketches* and Averaging, they are fault-tolerant, fast and also they provide results in all nodes. *Sketch* based lack some accuracy, but the tests showed a reasonable error of the value we wanted to evaluate.

3.2.3 Push-Sum

Push-sum protocol is described in [3] and it is a gossip-based protocol. [27] describes the algorithm function : along discrete times t, each node i maintains and propagates information of a pair of values (s_{ti}, w_{ti}) where s represents the sum of the exchanged values and w the weight associated. In each iteration, a neighbor is chosen uniformly at random and half of the actual values are sent to the target node and the other half to the node itself. Upon received, the local values are updated, adding each value from a received pair to its local component. In the Push-Sum algorithm [3], initially, each

Algorithm 1 Push-Sum Algorithm

1: Let $\{(s_1, w_1), \ldots, (s_r, w_r)\}$ be all pairs sent to i in round $t-1$
2: Let $s_{t,i} := \sum_r s_r$, $w_{t,i} := \sum_r w_r$
3: Choose a target $f_t(i)$ uniformly at random
4: Send the pair $\left(\frac{1}{2}s_{t,i}, \frac{1}{2}s_{t,i}\right)$ to $f(i)$ and i (yourself)
5: $\frac{s_{t,i}}{w_{t,i}}$ is the estimate of the average in step t

node generates a pair (s, w) where s is its value to aggregate and w its weight, initiated to one. At

every round t, each node i evaluate s_r and w_r which are the SUM of all the pairs $(s, w$ send to i in the previous round. After evaluating (s_r, w_r), send half the $\frac{1}{2}s_r$ and $\frac{1}{2}w_r$ to a randomly choose neighbor and to itself. The *AVERAGE* in the round t is the *SUM* of all received s in the round i plus s_r divided to the *SUM* of all received w in the same round i plus w_r.

3.2.4 Flow Updating

Algorithm 2 Flow Updating

State Variables:
$f_{ij}, \forall j \in D_i$, flow, initially $f_ij = 0$
$e_{ij}, \forall j \in D_i$, estimates, initially $e_{ij} = 0$
v_i, input value
message-generation function:
msg$(i, j) = (f_{ij}, e_{ij}), \forall j \in D_i$
state-transition function:
for all (f_{ji}, e_{ji}) **received do**
$f_{ij} \leftarrow -f_{ji}$
$e_{ij} \leftarrow e_{ji}$
end for
$e_i \leftarrow \frac{\left(v_i - \sum_{j \in D_i} f_{ij}\right) + \sum_{j \in D_i} e_{ij}}{|D_i|+1}$
for all $j \in D_i$ **do**
$f_{ij} \leftarrow f_{ij} + (e_i - e_{ij})$
$e_{ij} \leftarrow e_i$
end for

In Flow Updating[14], each node i initializes its state variables, a set of pair (f_{ij}, e_{ij}) where $f_{ij} = 0$ and $e_{ij} = 0$, a pair correspondent to each neighbor, contain the flow and an estimate. Also, the node holds an input value v_i, the value to aggregate.

At every round, a node generates and sends a message to each neighbor j, the node i send its correspondent flow and estimate (f_{ji}, eij).

The next step, the state transition function, each node starts by updating the local flows and estimates with the corespondent received one from the corespondent neighbor. Thereafter, each computes a new prediction of the aggregation value e_i by averaging the received estimates and the one locally calculated by the equation bellow, that evaluates the overall estimate $AVERAGE$ of the network. It updates after its state accordingly: the new estimates equals to the one previous estimate calculated and the flow f_{ij} is added the difference between the new estimate e_i and the received estimate from$j$.

$$a_i = v_i - \sum_{j \in D_i} f_{ij}$$

3.2.5 Push-Pull

Similar to the aforementioned *push-sum protocol*, the push-pull gossiping[6] performs an averaging process. This algorithm executes an epidemic protocol to perform a pari-wise exchange of aggregated values among neighbor nodes[27]. In periodic intervals of time, a node send its value to a randomly selected node and waits to receive a result back, the response from the selected node. Afterwards, an average with the new value and the present value its performed in order to calculate and store a new one. When a node receives a value from another node, the same process is performed, send the current value and calculate a new one from the average of the received value and the current one.

In Push-Pull protocol [9], the nodes work with two *threads*. The active *thread* runs once at each

Algorithm 3 Push-Pull Active Thread

$q \leftarrow getneighbour()$
send s_p to q
$s_q \leftarrow receive(q)$
$s_p \leftarrow update(s_p, s_q)$

Algorithm 4 Push-Pull Passive Thread

$s_q \leftarrow receive(*)$
send s_p to $sender(s_q)$
$s_p \leftarrow update(s_p, s_q)$

round. Selects a neighbor q at random and send to it its value to aggregate s_p, afterwards, expects to receive the value s_q from the neighbor q. Update them the value s_p by averaging s_p and s_q.
Each node runs the passive *thread* in background, all the time. This background process basically sends the hold value s_p to every requested neighbor q. After sending it, the node updates it value s_p the same way as the active *thread*.

3.2.6 RIA LC/DC

Algorithm proposed in [11], a multi-path routing aggregation approach. The algorithm consists of two phases. First an aggregation request is sent by the sink throughout the whole network, creating a multi-path routing hierarchy. Second, starting in the lower levels, each node generates a *sketch* correspondent to its current state and sends it to the nodes in the upper level. The node that receives the *sketch*, creates a new one combining its current value and the received *sketch* and sends it to the upper node until the top is reached where the sink computes the aggregation estimate.
Each node initially holds a *sketch*, an array of zeros of size m. Each node initializes the array by

mapping, using a random function that gives the index of the array, the value to aggregate into the sketch, bit by bit. For example, if the value to aggregate is 3, the node should map the values 1, 2 and 3. After initialized the sketch, at each round the nodes share with their neighbor the sketch and merge the received ones with the sketch hold locally by using *XOR*. In order to calculate the estimate $\hat{n}$ of the SUM is calculated by

$$\hat{n} = -m * ln(V_n)$$

Where V_n is equal to the division of the number of zeros in the *sketch* and the size of m

3.2.7 Extrema Propagation

This approach reduces the computation of an aggregation function[27]. A vector x_i of k random number is created at each network node i. Random numbers are generated according to a known random distribution, using the node initial value as an input parameter. The execution of the algorithm *"consists of the computation of the point wise minimum between all exchanged vectors"*[27]. At each node, the obtained vector is used as a sample to produce an approximation of the aggregation result. This algorithm is focused on obtaining a fast estimate, rather than an accurate one. In Extrema Propagation,

Algorithm 5 Extrema Propagation

const K
var $n, x[1..K]$
Require: Init
$n \leftarrow neighbours(self)$
for all $l \in 1..K$ **do**
$x[i] \leftarrow rExp(1)$
end for
Send x to every p $p \in n$
Require: Receive $m_1..m_j$ from all $p \in n$
for all $l \in 1..j$ **do**
$x \leftarrow pointwisemin(x, m_l)$
end for
Send x to every $p \in n$
Require: Query
return $\tilde{N}$

each node holds and shares a *sketch*, an auxiliary data structure to calculate the desired aggregation function. Each node holds an array x with dimension K, and initializes every $x_i \in [1..K]$ equal to a random value calculated by the function $rExp(1)$, which returns a random number with and exponential distribution of rate parameter 1. n is initialized as the set of all neighbors to a node i, thereafter, the each node send the array x to every neighbor from n. At each round, every single node from each message l from $m1..m_j$ updates the array x begin equal to the $pointwisemin(x, m_l)$. After

updating x, send the updated array to each neighbor from n.

To calculate the estimate of the size of the network $\hat{N}$ in each round, each node computes the equation:

$$\hat{N} = \frac{K}{\sum_{i=1}^{K} x[i]}$$

3.2.8 AVERAGE and SUM

The aforementioned are used for evaluating the *AVERAGE*, except for RIA LC/DC. The other algorithms suffer a small modification in order to compute other aggregation function such as *COUNT* and *SUM*.

Considering the averaging algorithms, Flow-Updating, Push-Pull and Push-Sum the principle is the same. The idea is to calculate the number of nodes in a network,*COUNT* function, and aftewards obtain the *SUM* by multiplying the *AVERAGE* result with the number of node in the network. To calculate the number of nodes in a network, each node shares the value 0 instead of its internal value, except for one node that starts with the value 1. From here, the process is the the same. Each node shares the value with its neighbors and after receiving values from its adjacent node, averages all of them with its internal value. The idea is in the end obtain a value close to $1/n$, where n is the number of nodes in a networks. To obtain the *SUM* in every moment, it is multiplied n with the *AVERAGE*.

In the Extrema Propagation case, to calculate the *SUM* instead of the size of the network, the $rExp()$ function used takes as argument the value to aggregate, i.e., each node computes the function with its internal value instead of 1.

Chapter 4

Network Topology

In order to test the aforementioned algorithms, it is necessary to build and evaluate a topology that represents a SG network. The algorithms work with data, their function is to gather and collect information and compute aggregation functions in a distributed way. Therefore, we built our network considering that it is a part of the SG information subsystem.
As it was stated in the Chapter2, the AMI/AMR network is part of the information subsystem. So, our topology must represent an AMI Network, with the smart meter data used to compute the aggregation function. We are interested in the collection of the data and the required process to update the prices according to the overall production/consumption. In the following section, we detail more precisely what scenario we assumed and which communication assumptions we made.

4.1 Case Study

In this work we consider the existence of data collectors. We don't make any assumption regarding the number of households connected per data collector. Instead we assume that a certain number of smart meters are connect to a data collector. Therefore, each device contains information about the connected houses consumption and production.
Furthermore, we assumed a real time pricing scenario, where each aggregator have a behavior similar to [43] and the PowerMatching city, i. e., the aggregator makes decisions regarding the energy price based on the supply from the electrical company, the total consumption from the costumers and from their production by renewal energy sources. The pricing policy followed is outside the scope of this work, we only consider the problem of evaluate the overall consumption or production, so the aggregators can make the pricing decision accordingly.

Definition 4.1. For each data collector D, a set H of households with a smart meter connected to it. H is defined as

$$H = \{h_1, h_2, h_3, \ldots, h_n\}$$

Where n is the number of household in H and $n >= 1$

Definition 4.2. At a given time t, a data collector D contains a consumption C_D which is defined by:

$$C_D = \sum_{i=1}^{n} c_i$$

where n is the size of the set H and c_i is the consumption of the i household connected to the data collector D at the time t.

Also, we assume that each data collector is located not only on the MV/LV transformer, but in every substation of the network that contains electric equipments (e.g., rails, bus bars, electrical switches, etc). Although we don't cover the security aspects of the data collectors: information security, privacy and integrity and physical information, there are in the current projects security measures to avoid the violation/stealing of the data collector and encryption techniques that are applied to provide security information guarantees. Once again, security issues are not covered in this work.
In the InovGrid [40] and Iberdrola projects [34], the data collectors only communicate the data to the electrical company and not among each other. We took the example of the PowerMatching city, and other proposed SG architecture seen in Chapter 2 that consider not a hierarchical network composed by households, data collector substations, and the data center as a sink, but a network of data collectors that communicate with each other. Although they communicate the overall network consumption/production with the electrical company, they share their data with other collectors in order to compute an aggregation function without the need of a central device.
To complete our scenario, we assume one more similar aspect to the PowerMatching city, but instead of considering that the data collector receives the data in a bid format to buy or sell energy, we consider consumption and production values. The price of the energy should be calculate according to this 2 variables, as in a PowerMatching market does with bid. We will not cover the market aspect, the problem we focus on is knowing the consumption or production of a data collector network using Distributed Aggregation Algorithms.

4.1.1 Communication

When building a topology or a network, it is important to know and consider the communication technology used and what were the assumption we made about it. In AMI networks, there are many options, both wired and wireless, for as many needs. A suitable choice for a Smart Meter - data collector communication could not be a suitable choice for a central facility substation flow because there are different needs and requirements. Since we were interested in the data collected by the aggregators and the communication between them. The choice of a communication infrastructure must be among the technologies used in the Low Voltage grid, i.e., where the metering is performed. We assumed a PowerLine Communication Infrastructure, used, for example, in EDP InovGrid[40] and Iberdrola pilot projects [34].

To simulate a PowerLine communication, in each cable that connects two data collectors, we evaluate its channel capacity in *Mbps*. To evaluate it, we use a channel capacity function defined in [17] , assuming that all channel are Gaussian Channels. We also assume that the communication uses the CENELEC A band.

Definition 4.3. Channel Capacity is defined by the function M_{ij}

$$M_{ij} = \int_{B_1}^{B_2} C\left(\frac{S_T|H_{ij}(f)|^2}{N_0\Gamma}\right)\mathrm{d}f$$

Where $H_{ij}(f)$ represent the transfer function between data collectors v_i and v_j, Γ is a gap factor to account for practical coding schemes, $B_1 = 9kHz$ and $B_2 = 95kHz$ are the lower and upper frequency of the CENELEC A band, S_T represent the power spectral density, N_0 is the background noise modeled as white Gaussian noise and $C(\gamma)$ is the capacity at signal-to-noise power ratio (SNR) of γ

We assume the same reference values in [17], $S_T = -60dBV^2/Hz$, $N_0 = -138dBV^2/Hz$ and $\Gamma = 10$. $C(\gamma)$ is defined as:

$$C(\gamma) = \log_2(1+\gamma)$$

We assume that all the cables are of the type NAYY150SE. The cable lengths are the ones provided the samples used in [41], other physical characteristics are taken from [29].

The channel capacity tells us the number of *bits* that a channel can transmit in a second. We use this measure to evaluate the latency of a communication. We assume the latency as the time it takes for a message to be used or processed by a node. We mimic a delay in the communication.

The process to evaluate the latency is explained.

- At each transmission, the data collector divides the message size in *bytes* by the channel capacity in *mbps* to obtain the delay, time it takes for the message to be transmitted.
- The data collector attaches the evaluated delay time plus the current time, in the message that will be sent.
- When receiving the message, the data collector checks the current time and compares it to the time attached in the message.
- If the time attached is greater, the data collector simulation waits until otherwise. After that time, consumes/processes the received message.

We didn't considered a global clock, each node evaluates the time using a function that gives us the current time in miliseconds. Even though we assume the capacity in every channel, this assumption affects little the computation time of our algorithms. In addition, we assume that each algorithm use the same message size which is 32 *bytes*, the same size assumed in [23] for periodic data readings.

4.1.2 Network Architecture

This work focu is on the LV power grid network, with voltage $V_{LV} <= 10kV$, to evaluate a Smart Grid scenario. We start by choosing a Low Voltage power grid network sample to define a Smart Grid graph. For the low voltage power grid sample graph, we choose one from the work of [41] and define a Smart Grid graph.

Definition 4.4. A Smart Grid graph is a graph $G(V, E)$ such that each element $v_i \in V$ is either a substation with electrical equipment or a transformer of a physical power grid with a data collector installed. There is a edge $e_i, j = (v_i, v_j) \in E$ between two nodes if there is physical cable that enables a PowerLine communication connecting directly the element represented by v_i and vj

gode Our Smart Grid graph is a weighted graph, the channel capacity is the weight for the edges. The figure 4.1 illustrates the graph representing the network used for our tests. Moreover, we choose the *Averaging* algorithms and based on *Sketches* because they are designed to evaluate the function *AVERAGE*, and more important, the function *SUM*. Other algorithms are more suitable to understand a distribution of values in a network or to evaluate its size.

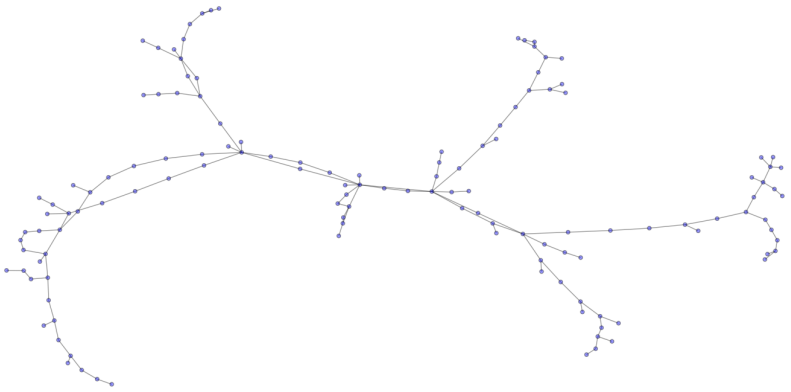

FIGURE 4.1: Smart Grid Graph

Chapter 5

Performance and Results

This chapter presentes the description and results of the tests made in our Smart Grid graph including a description of the implementation and the tools used for the simulation. Moreover, results for each algorithm will be presented with a comparison between all the results.

5.1 Implementation

All the algorithms were implemented using the programming language Python. Other tools and libraries are explained in the next sections.

5.1.1 Tools

5.1.1.1 NetworkX

We used NetworkX to represent our Smart Grid Graph, each node represents a substation/data collector, and each edge represent a physical cable connecting two substation. For our weight we used the channel capacity in *mbps* as explained in the previous chapter.
NetworkX [38] is a Python package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks. It provides a python data structure to represent several different types of graphs. It is also provided and API to perform know algorithms on graphs such as the evaluation of the shortest path between two nodes. NetworkX represents graphs, nodes and edges as classes, providing generators to create standard graphs. An example to generate a simple graph:

```
Import Networkx as nx
```

```
G=nx.Graph()
```

Once the graph it's created, it is possible to add a node, or a set of nodes. The same applies for adding edges.

```
G.add_node('a')

G.add_node('b')

G.add_edge('a','b', weight = 2)
```

NetworkX represents the graph as an *"dictionary of dictionaries of dictionaries"*, according to the website, this allows fast lookup with reasonable storage for large sparse networks. The nodes represent the keys, so, every object that is hashable can be stored as a node.

The method for accessing a node ,`G[node]`, returns an adjacency dictionary. For each edge, NetworkX uses a dictionary to store each property of the edge, in the cases of graphs where the edges have several weights. It is also relevant to note that other important functions were used from the NetworkX. One of the most used in our implementation is the function to get the adjacent nodes for a certain node given as an argument.

```
G.neighbors(node)
```

It is also possible to store data in a node. Each node stores a dictionary that enables data that can be stored, as long as the key to identify it is an hashbale datatype. In our work, we used this feature store data structure such as a list of received messages.

5.1.1.2 Other tools

Other tools were used in our simulations. The network sample was stored in the XML format, it was used the `lxml` package to read the XML files, parse the data and store it as an NetworkX graph. The `lxml` package is a python library used to process XML and HTML files with the auxiliary of the python `Elementtree` API.
Finally, to show the results, we used *Microsoft Excel* to organize the results in tables and design the graphs. To directly store the results of the algorithms simulation we used the Python package `XlsxWriter`

which is a Python module for writing files in the Excel 2007+ XLSX file format.

5.1.2 Implementation Architecture

In this section we detail the architecture of our implementation. For each algorithm, we implemented a independent python project, because each one have different proprieties, they implement different functions and also have different data requirements such as the message format. Also we did that way in order to keep the simplicity of the simulation. Although the implementations are separated from each other, all of them follows the same architecture.

- `Network.py` This python class stores the `NetworkX` graph and provides an API to perform operations in our Smart Grid Graph. The more important are `SendMessage(dest,m)` that stores a message m in a node `dest` *inbox*, and `GetMessage(pos)` that returns the list of messages received from the node `pos` *inbox*. Other methods provided in this class are to get proprieties of the network such as the number of nodes, the channel capacity of a given edge and the list of the neighbors of a given node. Also, for performance measurements, this class also stores the real value of the consumption summation in order to compare with the estimated values from the nodes.

- `Node.py` This python class represents a node from our graph, or, a data collector. Each node stores a position in the graph, obtained from `Network.py`, a list of neighbors in the graph, i.e., adjacent nodes, internal values, in our case the consumption or production, and an list of estimators for each iteration of the algorithm. This class contains the `main()` method that executes the algorithm.

- `GraphUtils.py` This python file provides an API for graph operations, such as generate an `NetworkX` graph from the XML file and apply the transformations that will be detailed in Section 5.3.1. Also it provides proprieties of the Graph related to Complex Network Analysis, explained also in Section 5.3.1.

- `NameOfTheProtocol.py` This class have the same name as the protocol to be tested. We have `PushSum.py`, `PushPull.py`, `Extremapropagation.py`, `FlowUpdating.py` and `RiaLCDC.py`. This class creates an instance of Network.py class and a set of Node.py class instances. Note that this set has the same size as the number of nodes in the graph since each Node object represents one node of the graph. Also it creates an Python *Thread* for each node to execute the algorithm.

At the end of the simulation, it is gathered the results of all nodes and evaluated the results of the simulation that are afterwards stored in an *xlsx* file.

5.2 Performance Tests

In this section, we describe our tests and show the simulations results. As it was stated in Chapter 4, we measured the accuracy of the evaluation of the total consumption and/or production, considering that each node contains, as its initial value, the consumption or production to be aggregated. In our tests, we did not distinguish wether we wanted to evaluate the total consumption or production because both are evaluated in the same way. It is just the context that is different. Therefore, we considered the computation of the aggregation function *SUM*, the summation of all the network values in the tests that were performed.

When executing each algorithm, at every single iteration, a node, or data collector, holds an estimation of the total *SUM* of all nodes. The more iterations we make of the algorithm, ideally, the more close the node estimation is to the actual value. However the price to pay is that is necessary more time to complete the simulation. Since we consider a real time pricing scenario, the execution time is an important matter, so we wanted to know the fastest algorithm in matters of time that can also give us a reasonable estimation. To measure the overall error of the results provided by the algorithms we use a metric called Observed Relative Error (ORE) defined in [27]. We use this measure to calculate both estimation error per node and the average estimation error of the all network.

Definition 5.1. The ORE of a set J with respect to a value to estimate N is defined as the square root to the mean square error.

$$ORE = \frac{\sqrt{\frac{\sum_{i=1}^{J}(\hat{N}_i - N)^2}{J}}}{N}$$

where $\hat{N}_i$ for $i = 1..J$, is a set of observations of the estimate of a given N. Also, at all nodes, we stored an estimation of the actual value at every single iteration. In the end, we compared these estimations so we know how close the nodes are to the average error. The formula used by a single node to evaluate an estimation is equal to the previous stated formula, expect that instead of making a summation, we normalize the difference between the estimation and the value to estimate.

Definition 5.2. The ORE of a node i with respect to a value to estimate N is defined as the square root of the difference between the estimation value $\hat{N}_i$ and the value to estimate N.

$$ORE_i = \frac{\sqrt{(\hat{N}_i - N)^2}}{N}$$

The first metric is defined in terms of the MSE as stated in [27]. The second one is evaluated from the first to obtain an estimation value for each node for each iteration of the algorithm.

5.2.1 Results

In this section, it is presented the results of our first set of tests. In this case we tested each algorithm in the Smart Grid graph under the constrains we stated before. Channel capacity influences the latency, i.e., the time it takes for a message to reach its destination, as it was explained in Chapter 4. In the tests we made, each node executes individually the predefined algorithm, exchanging the required messages with its adjacent nodes, i. e., the neighbors. All nodes execute the algorithms a predefined n number of times. For our tests, we considered that each station executes 100 iterations of each algorithm, so $n = 100$. Each station stores the evaluated estimation according to the formula in 5.2 at each iteration. To simulate the independent behavior of the nodes, each one executes the algorithms as a single thread. The estimations are stored locally by the nodes. In the end of the simulation, all the collected data from the nodes throughout the simulation is aggregated and used afterwards to evaluate the algorithm ORE for each predefined time interval.
In our simulation, we measured how long it takes for each algorithm to execute 100 iterations in all nodes. Moreover we measured the time that is required in average per round to complete an iteration. In the charts, the green dots represent the evaluated ORE according to 5.2 from each node at a given

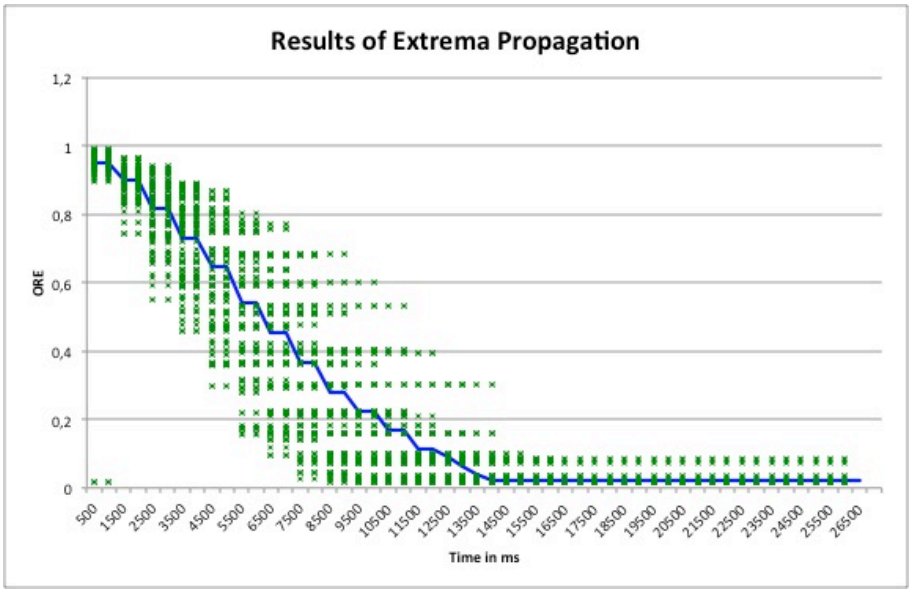

FIGURE 5.1: Test restuls of Extrema Propagation

time in ms and the blue line the average ORE, evaluated according to 5.1 throughout the time in ms.

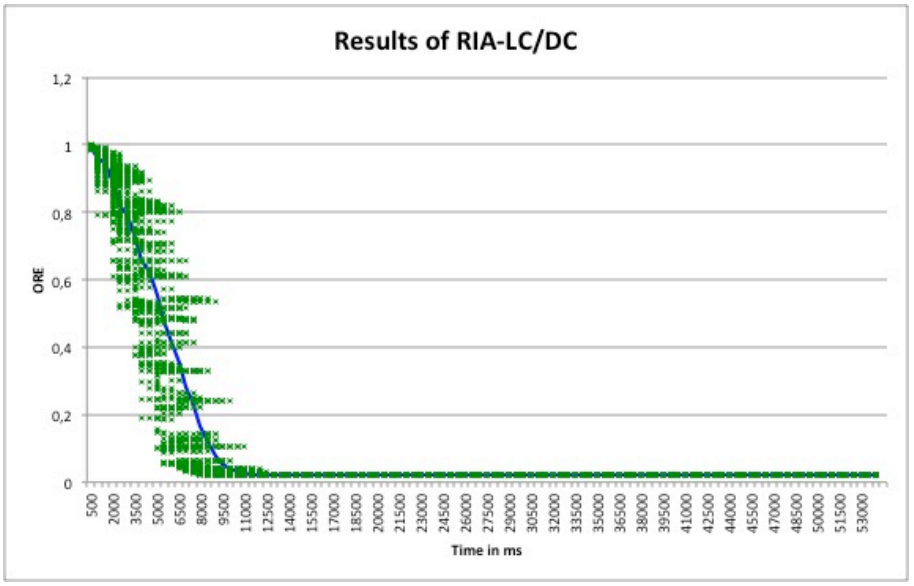

FIGURE 5.2: Test results of RIA LC/DC

Figure 5.1 shows the results of Extrema Propagation and how the algorithm performed. In average, the protocol give us a fair estimation of the total summation, with a relative error of 0.016198 after 26949 ms, or 27s. After the time 13500ms, the algorithm ORE keeps the same value. The estimation at each node follows the same curve as the average ORE, decreasing until a certain estimation number and them it is constant after a certain point. The total time of the algorithm to execute 100 iterations in every node is of 27949ms.

The algorithm RIA LC/DC results are showed in 5.2, they are similar as the ones performed by Extrema Propagation. Since both are *Sketch* based algorithm, the pattern is resembling. The average ORE decreases through time until it reaches an certain value that is constant from that point forward. The ORE at each node behaves the same way as the average ORE along the time, it is possible to see from the green dots in 5.2, all of the nodes have the same estimation error, 0,02188. RIA LC/DC provides an estimation error slightly more inaccurate than Extrema Propagation. Also, Extrema Propagation is faster than RIA LC/DC. RIA LC/DC reach the stationary value in 14000ms, with a total compilation time of 54431ms.

Gossip Based Algorithms performed poorer than the ones based on *Sketches*, reaching estimation errors in some cases hundreds of times greater than the actual value, although in the end the error is lower. Flow Updating results in figure 5.3 show that the estimation error in average keeps increasing in all nodes until the time of 29000ms. After that point, the estimation error decreases, but without reaching a reasonable error value. In the end of the simulation, the ORE is still greater than the actual value, 2,41 times greater. Even though the algorithm is faster than RIA LC/DC and slightly slower than Extrema Propagation , with a execution time of 46383ms, the results are not as accurate. If we take

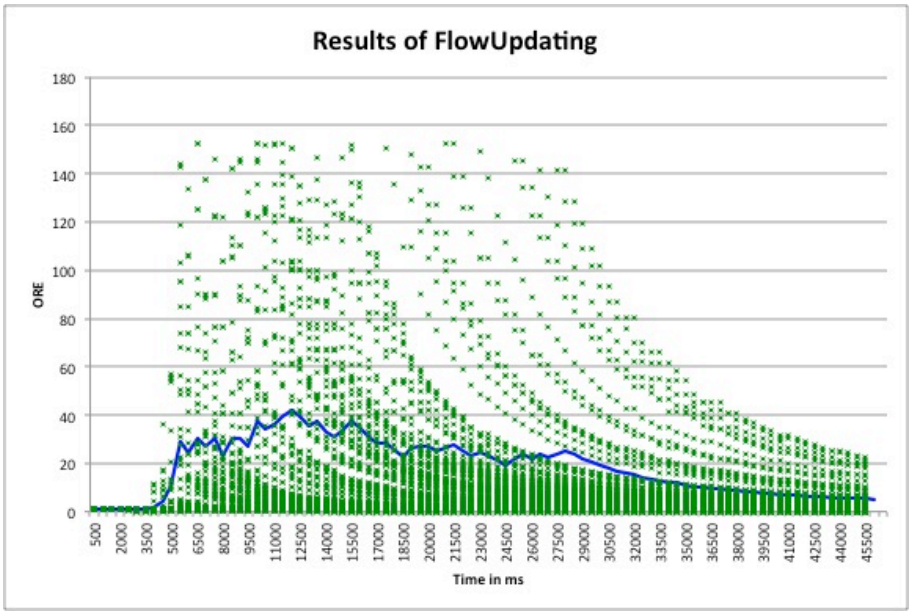

FIGURE 5.3: Test restuls of FlowUpdating

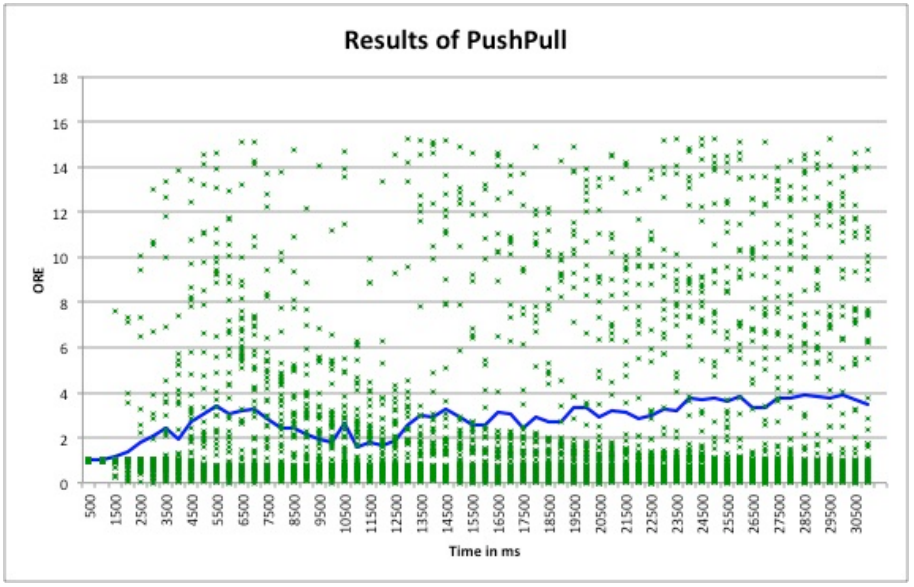

FIGURE 5.4: Test restuls of Push-Pull

a close look at the estimation error in certain nodes, we found more reasonable results. Some of the nodes have estimations lower than 1 but fewer have a lower estimation rate than 0.5. The problem is that other nodes have estimation errors 20 times greater than the actual values. That explains why the average ORE is so high compared to the previous two algorithms.

The same conclusions can be made considering the results of the Push-Pull algorithm in figure 5.4. The average error is in the range of values during all iterations, between 1 and 4 . At the end of the test, it is still higher than the value to estimate, 3.53 times higher. The cause of such high value is the same as FlowUpdating high error value: although there are nodes with estimation error lower than 1 at some

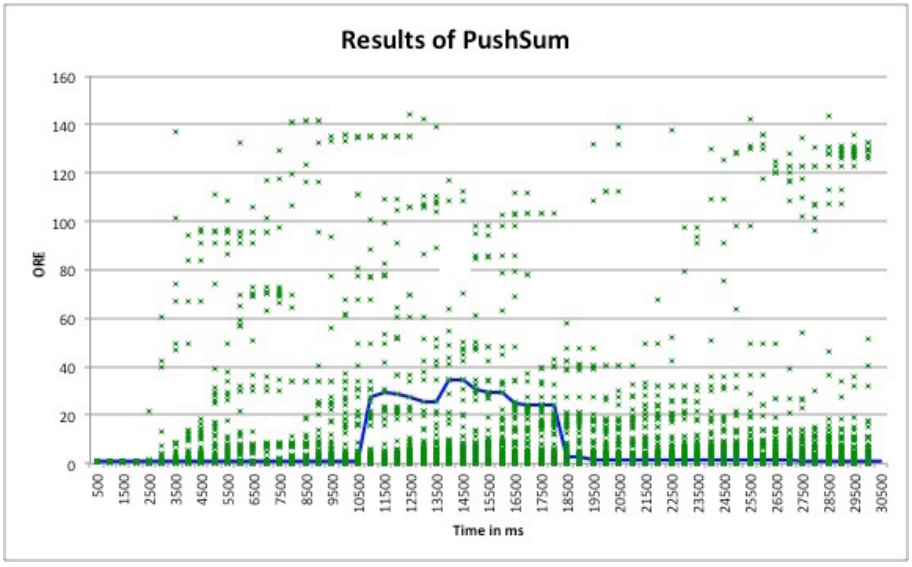

FIGURE 5.5: Test restuls of Push-Sum

of the nodes, others have estimations with errors greater than 5 and 6. Considering the execution time, the algorithm is one of the fastest ones: execution time of 31987ms, but its results lacks in accuracy. Push-Sum results are showed in figure 5.5, they are the poorest ones. The error keeps increasing, reaching 140 times greater than the real value, in some nodes. In the end, the error make the estimation still unusable, it is 1,06 times higher and very few nodes have an estimator with an error less than 1. However, the algorithm is one of the fastest, with an execution time of 30610 ms.

5.2.2 Discussion

Considering the results, the algorithms based on *Sketches* were the only ones that computed estimated values with a low ORE, *Gossip* based performed poorer. Among the *Sketch* based ones, Extrema Propagation provided better results than RIA LC/DC, and therefore better than any of the tested algorithms. Also, Extrema Propagation was not only the most accurate algorithm, it as also faster than RIA LC/DC. Normally, *Gossip* based should be faster than *Sketch* based. That is true for RIA LC/DC, but not for Extrema Propagation. That can be explained because in our *Gossip* based algorithms implementation, we used waiting times at all nodes so that each one can wait long enough to receive all messages from its neighbors. Extrema Propagation, is our tests, used a smaller sized *sketch* than RIA LC/DC, which can be the reason why RIA LC/DC is slower.

The reason because *Gossip* based approaches scored so badly, specially when compared to the ones based on *Sketches*, relies on the topology of our Smart Grid graph. Gossip based approaches tested

consider the exchanging of the nodes partial aggregates with the neighbors. A new estimation is calculates with the received partial aggregates from the adjacent nodes. The higher number of neighbors, i.e., the more connected is the graph, the better these algorithms would perform. As we've seen in Chapte 3, the *averaging* algorithms, multiply the *AVERAGE* with the size of the network to obtain the summation of all the values in a network. The principle to obtain the size of a network, and therefore the summation of all values, is also explained in Chapter 3. The problem that explains the inaccuracy of the results is that the evaluation of the size of the network is also very inaccurate. Our Smart Grid graph is in a radial shape, meaning that most nodes only have 2 neighbors, since the principle to obtain the size is evaluate $1/n$, it requires a high number of iterations to obtain a estimation close to $1/n$.
On the other hand, *Sketch* based approaches hold an array representing a *Sketch* of the value. In every iteration, each node exchanges *Sketches* with its neighbors, but instead of averaging it, it performs other operations. *Pointwisemin* in the Extrema Propagation case and *XOR* in RIA LC/DC. After a certain point, the operations are idempotent, gives always the same value. That's why at after a certain point in time, the error stops decreasing.

5.3 Network Evolutions

So far, the tests presented were in a Smart Grid Graph that is, basically, a power grid topology. Even though this gives us an intuition about how the algorithm performs in a grid context, with data collectors and SM, the power grid is more likely to change in the next years in order to become more connected and more resilient so it can adapt to the demands of the new Smart Grid, especially to support the new consumer/producer approach. The work of Pagani [41] explores in more detail the new demands and challenges the current power grid face when considering the evolution to a fully Smart Grid and it gives us some clues about how the grid can evolve in a topological way, using complex network analysis, and how can new networks can be created.

5.3.1 Network Evolution Strategies

Considering the evolution of the current power grid, Pagani proposes evolution strategies to the current power topologies in order to achieve more connectivity and resilience. The evolutions happen when adding more edges or physical cables, to the power grid topology. The transformations are separated into 4 groups of edge growth: increments of 25%, 50%, 75% and 100%. The evolution strategies are taken from [41]:

- **Assortativity** A network is assortative if nodes having similar characteristics or properties are connected to one another[2].There are two types of transformations, considering the assortativity strategy: *Low Degree*(LD) it considers a set of nodes sorted by node degree and starts connecting the nodes with the lowest equal node degree. *High Dregee*(HD) is equal to LD but instead it connects the nodes with the highest equal node degree.

- **Dissortativity** Opposite of the previous transformation strategy, a network is dissortativity if nodes having different characteristics or properties are connected together. So nodes with highest node degree are linked to nodes with lowest node degree.

- **Triangle Closure** Is is based on the principle of increasing the clustering coefficient of a network. At each step, a node is selected at random. For each pair of its neighbors, an edge is added between then, if not already present.

- **Random** It is based on the random selection of nodes to attach edges. At each step, a pair of distinct nodes are randomly selected and an edge between them is added.

The images 5.7,5.8,5.9 and 5.10 show the evolution following the triangle closure strategy. The figures 5.11, 5.12, 5.13 and figure 5.14 show the fourth step of the other transformation policies.

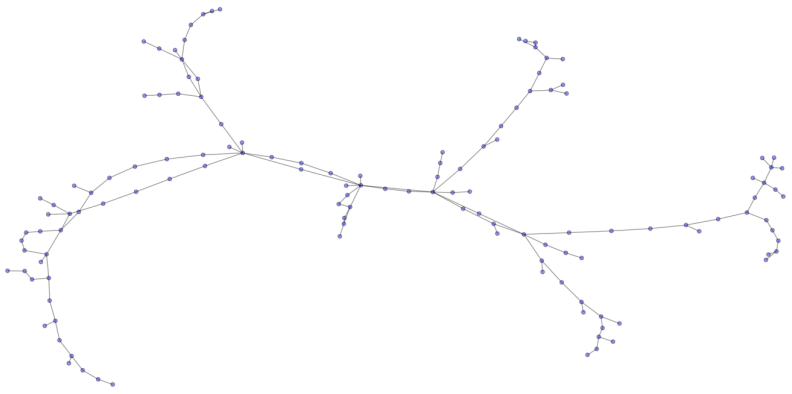

FIGURE 5.6: Smart Grid Graph

5.3.2 Test Results

The algorithms were tested in all graphs resulted from applying the aforementioned transformation policies to our original weighted Smart Grid graph. To add a new edge, or physical cable, we

FIGURE 5.7: First stage of evolution - Triangle Closure(+25% edges)

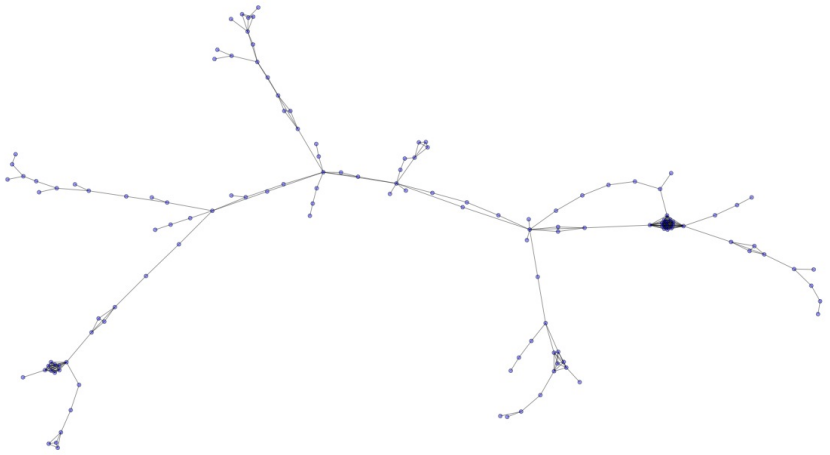

FIGURE 5.8: Second stage of evolution - Triangle Closure(+50% edges)

need to calculate the channel capacity for it. Unfortunately, it was not possible to know the physical distances of the new cable since there is now information of the geographic coordinates of our sub-stations. From our first network, the channel capacity of all edges is between a very short interval, which means that the weights are not very relevante for the performance. Eventhough, a reasonable channel capacity we considered for all the new edges were the middle point of the channel capacity range: 57434 bps.

The goal is the same, to evaluate a total summation off all the values in the network. Since we generate up to 200 new charts, the results are showed in image 5.16 that contains a table with the performance

FIGURE 5.9: Third stage of evolution - Triangle Closure(+75% edges)

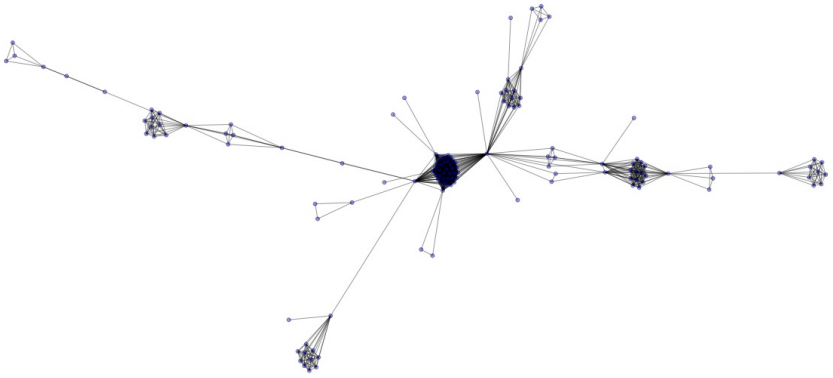

FIGURE 5.10: Fourth stage of evolution - Triangle Closure(+100% edges)

of the algorithms in the new generated Smart Grid graph. In this set of tests, we recorded the final average ORE and how much computational time a algorithm requires. We wanted to know how long it takes for a protocol to reach the lowest ORE, considering that the ORE does not increase from that point forward. As seen in the first set of tests, *Sketch* based algorithm stop improving the ORE after reaching the lowest point, we wanted to know how long it takes to reach that point. As for *Gossip* based algorithms, the ORE keeps improving, so we wanted to know the point where the ORE improvements are residual, or close to zero.

The time is in ms and the the ORE is the average ORE of all nodes is calculated after 100 iterations of

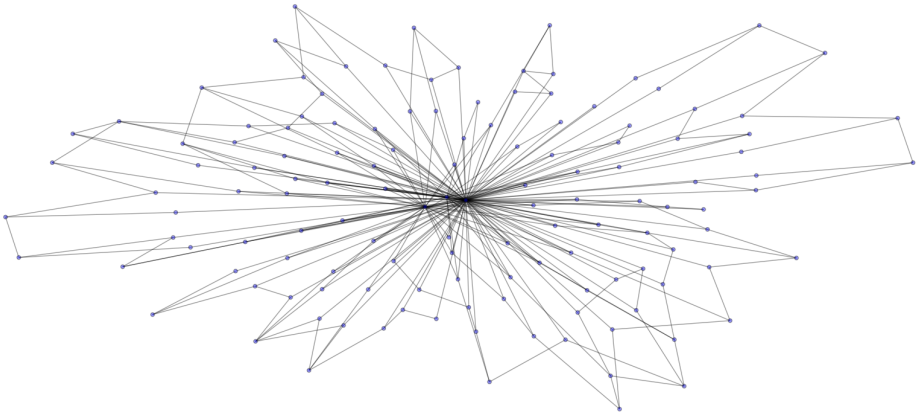

FIGURE 5.11: Fourth stage of evolution - Dissorsativity(+100% edges)

FIGURE 5.12: Fourth stage of evolution - Assorsativity HD(+100% edges)

the algorithms. For measuring the execution time, up to 500 iteration were necessary to know, in the cases of Push Pull and Push Sum, how long it takes to reach the best ORE. The results obtained in the ”normal” Smart Grid graph are also in the table for comparison proposes.

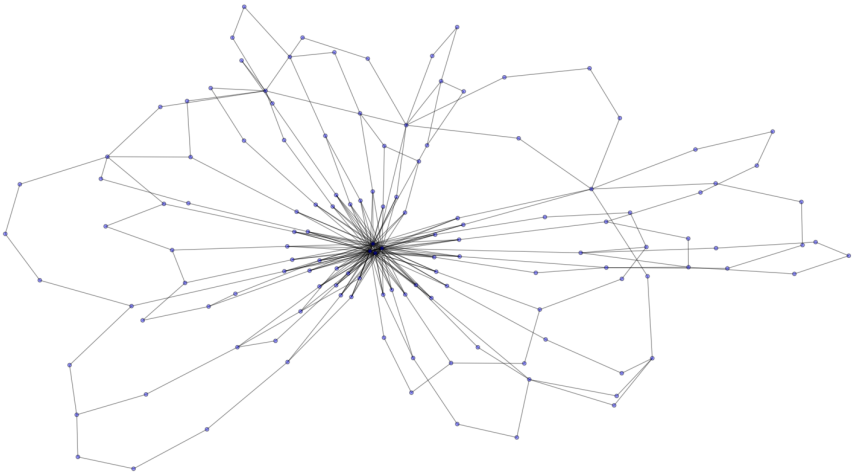

FIGURE 5.13: Fourth stage of evolution - Assorsativity LD(+100% edges)

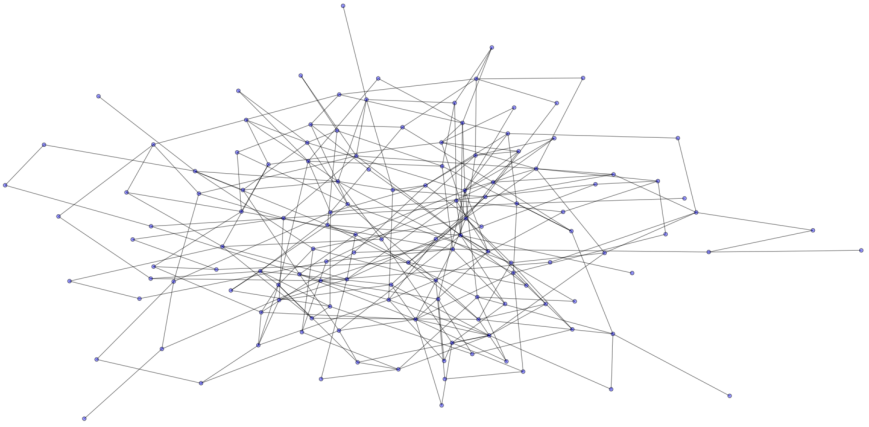

FIGURE 5.14: Fourth stage of evolution - Random(+100% edges)

		Push-Sum		Push-Pull		FlowUpdating		Extrema Propagation		RIA LC/DC	
		Time ms	RMSD	Time ms	RMSD	Time ms	RMSD	Time ms	RMSD	Time ms	RMSD
Normal	0%	30844	1,06406902	31987	3,892558206	49504	2,41624441	26094	0,016198242	54000	0,02188784
Triangle C.	25%	30863	10,7038156	34947	28,71540753	61275	1,82930896	34297	0,048803796	70569	0,10431259
	50%	30973	5,09936059	34648	10,42620299	66618	0,13357251	47010	0,021321652	80096	0,01756662
	75%	39050	3,85025238	35524	3,470965605	79430	0,14925223	52885	0,018010413	80134	0,04333673
	100%	31034	3,92818744	34388	0,830351836	180846	0,57219778	57711	0,042551853	85062	0,06986143
Dissortativity	25%	30547	0,38760007	34751	0,061360895	66570	0,00075673	40786	0,020491376	70142	0,10759254
	50%	31749	0,13706974	34403	0,007629007	101366	0,00017118	39845	0,007686624	74462	0,08140127
	75%	31350	0,12786271	34849	0,48132397	152445	8,5902E-05	49462	0,02150707	80222	0,01851089
	100%	30555	0,04402734	34309	0,078256265	178582	8,7293E-09	49983	0,042687211	81602	0,0813425
AssortativityHD	25%	30586	0,18877706	34985	0,513246373	80138	0,03237479	41778	0,033318846	69443	0,04688726
	50%	30684	0,18396067	35243	0,178792559	93784	0,03054365	40755	0,010324239	73083	0,01934589
	75%	30739	0,14634823	34874	0,123623678	115801	0,01154111	48648	0,049661258	76653	0,06722143
	100%	30743	0,1539824	34771	0,096697942	127741	0,00057366	49511	0,018790733	79814	0,00081847
AssortativityLD	25%	31000	0,20445395	34925	0,109694498	84308	0,00064148	35442	0,001654989	70164	0,14237511
	50%	30963	0,35787717	35221	0,083718482	100706	0,00112988	46876	0,045560258	74830	0,09357414
	75%	31071	0,19384351	35263	0,022034343	126450	1,8829E-05	52648	0,010557559	80423	0,00053121
	100%	30838	0,12586029	35101	0,072350452	142349	0,00029597	52601	0,034969321	83665	0,01677654
Random	25%	30777	0,13500838	36283	0,017499833	65334	0,00415684	36090	0,029894075	70776	0,24297329
	50%	31209	0,01973454	34859	0,105039221	74515	0,0084775	39650	0,009814183	73061	0,11781978
	75%	30816	0,03699709	34732	0,024462929	79708	0,01717719	46335	0,007513989	78402	0,05002644
	100%	33550	0,02986842	35677	0,030040598	77148	0,00614445	48764	0,008068009	83812	0,07216587

FIGURE 5.15: Test results with network transformations measuring the final ORE

		Push-Sum		Push-Pull		FlowUpdating		Extrema Propagation		RIA LC/DC	
		Time ms	RMSD	Time ms	RMSD	Time ms	RMSD	Time ms	RMSD	Time ms	RMSD
Normal	**0%**	152350	0,23806654	171200	2,7115618	68718	1,78047918	21000	0,01619824	13500	0,00709
Triangle C.	**25%**	152350	0,29938499	180200	3,6393572	81000	1,82930896	17500	0,0488038	12000	0,10431259
	50%	152350	0,45029597	176300	10,6853195	156000	0,12934568	22000	0,02132165	13200	0,01756662
	75%	152900	0,36276969	172700	34,0006749	200000	0,11920461	21500	0,01801041	12000	0,04333673
	100%	152350	0,34663573	172700	10,8101729	200846	0,57219778	19000	0,04255185	9900	0,06986143
Dissortativity	**25%**	133100	0,10572265	142700	0,52849098	49500	0,00050713	17500	0,02049138	6000	0,10759254
	50%	101750	0,14004015	86600	0,20461217	99000	9,9401E-09	14500	0,00768662	3900	0,08140127
	75%	20350	0,10420301	19100	0,11825491	94500	0,00061586	10500	0,02150707	2700	0,01851089
	100%	37950	0,05285927	17000	0,0656776	48000	0,00049585	13500	0,04268721	2700	0,0813425
AssortativityHD	**25%**	59400	0,18877706	105200	0,15988161	71500	8,2558E-05	10000	0,03331885	4800	0,04688726
	50%	38500	0,17711351	144500	0,02588199	44000	8,8478E-05	11000	0,01032424	4800	0,01934589
	75%	73700	0,07289273	152600	0,08467044	69000	0,00070142	14500	0,04966126	4500	0,06722143
	100%	44000	0,03448536	174200	0,09866972	102000	0,00052546	15500	0,01879073	4800	0,00081847
AssortativityLD	**25%**	111100	0,1293028	81200	0,04971884	154400	0,00017648	11500	0,00165499	6300	0,14237511
	50%	24200	0,12586029	27500	0,12708978	35500	0,00070533	10500	0,04556026	4200	0,09357414
	75%	50352	0,16109254	23900	0,03184473	118500	0,0017248	13500	0,01055756	5100	0,00053121
	100%	131450	0,15941478	15500	0,02884013	126500	0,00103695	12500	0,03496932	5400	0,01677654
Random	**25%**	40700	0,12888343	40400	0,04087553	28500	0,00867228	8000	0,02989407	5700	0,24297329
	50%	88000	0,02811812	6500	0,44999495	13500	0,00303983	8000	0,00981418	4500	0,11781978
	75%	64900	0,02383201	4500	0,03585616	17500	0,00954026	6000	0,00751399	3600	0,05002644
	100%	119350	0,00297277	6800	0,03340407	17000	0,00243769	5500	0,00806801	3900	0,07216587

FIGURE 5.16: Test results with network transformations measuring the final execution Time

As we can see from table in 5.15 that shows the ORE at the end of each algorithm simulation, the *Gossip-Based* algorithms performed better when we applied any of the transformations in our original Smart Graph. That confirms our first intuition, as the graph becomes more connected, the *Gossip* algorithms tend to give us better results. Also, the estimation error in every node of the graph are lower when we consider to compare the tests made in the original Smart Grid graph in any given time of test. Furthermore, FlowUpdating even get ORE that is almost $0,8,7293 * 10^{-9}$, in the case of Dissorsatativity policie with 100% more edges added. FlowUpdating is actually the algorithm with the best results, even when compared with the *Sketch* based ones. In some cases, *Gossip-Based* algorithms give us better results than *Sketch based.* Even though, FlowUpdating tend to be more slow with more edges, probably because of the need to compute more flows as the number of neighbors per node increases. In the other *Gossip* algorithms, Push-Pull andPush-Sum suffer little or none time penalty when more edges are added in all the strategies.

Sketch based algorithms continued to perform with good results. The time both take to execute 100 iterations is not affected in great scale, expect for some minor cases in Extrema Propagation, RIA LC/DC took up to 15 more seconds to compute. The ORE is not improved, once again, except for one or two cases.

Figure 5.16 show a table containing the necessary time for each protocol to reach a point where the ORE stops improving or makes residual improvements. It gives a more complete understanding, specially about how *Sketch* based algorithms improve when we applied any transformation. The time required for both Extrema and RIA LC/DC is less or equal as we add more edges in almost all cases. Comparing to the the regular network, *Sketch* requires less computation time in every transformation, except for Triangle Closure. This observation makes sense, as we add more edges, the graph diameter is reduced, in some transformations more than others, and if the graph nodes are more close to each, our two *Sketch* protocol improve their time. Push-Sum and Push-Pull also improve their time, as we add more edges, each node have more neighbors in average. With a higher node degree and less distance between nodes, these two *Gossip* protocol improve their results in both time and final average ORE. However, the improvements in time are not seen in all the *Gossip* algorithms since Flow Updating didn't improve in almost any case of our generated graphs.

The propose of this tests is to evaluate how the algorithms perform considering the unavoidable transformations that the current power grids will suffer in the next years. We took policies and strategies that can give us a clue about how the grid can evolve and them tested the algorithms. The goal is not select which transformation is the best one, because the algorithms perform differently in each transformed graph. Even though, a pattern as found that make us conclude that the add of more edges and physical cables, give us better estimation in *Gossip-Based* algorithms without compromising the *Sketch*

based ones, except for the Triangle Closure transformation in the Push-Sum and Push-Pull cases. More important than to understand and choose the better transformation policie is finding what seems to influence the performance of the algorithms. We know by intuition that the connectivity of the graph is an importante parameter in some algorithms, but it is not completely proved. Also, some other aspects of the graphs can influence the algorithms. The next chapter will explore this idea

Chapter 6

Principal Component Analysis

In Chapter 5 it was presented the performance results of the distributed aggregation algorithms in a Smart Grid graph. The first set of tests was performed in the graph defined in Chapter 4. Them another set of tests took place in evolved version of the first graph considering evolution strategies explained previously. When analyzing the results, we state that the *Gossip-based* algorithms improved their results. In some sort of graphs, the results obtained by *Gossip* based surpass the results obtained by *Sketches* based ones, with more accurate results and faster execution time. Considering this results, we considered that more important than understanding what evolution strategy provides the best results for our algorithms, we seek to know what parameters and characteristics in our Smart Grid graphs correlate more or less with the algorithms results.
We used graph theory and complex network analysis definitions, explained in A, to evaluate the characteristics of our Smart Grid Graphs. Also, we used Principal Components Analysis, a statistical procedure, to test if there is any correlation between the graph characteristics and the algorithms results. This definitions are explained in more detail in the next sections.

6.1 Principal Component Analysis Definition

Principal Component Analysis is a statistical procedure that uses an orthogonal transformation to convert a multi-set of values with n dimensions that are possibly correlated into another multi set of uncorrelated values. The number of dimensions of the resulting multi-set is inferior to the original multi-set dimension number.
The following definition and description of PCA is taken from [1], "*PCA provides an approximation of a data table, a data matrix, X, in terms of the product of the two small matrices T and P',. These matrices, T and P', capture the essential data patterns of X*". In a multivariate analysis, the starting point is a

data matrix X. The N rows of this matrix or table are named "objects", each row corresponds often to a data sample, depending on the context. In our case, a sample is a set of characteristics of a graph, plus the correspondent results of the aggregation algorithms in that same graph. The K columns of the table, are named "variables" and comprise the measurement made on the objects. These are defined according to the problem at hand.

When using PCA, usually the goals are related to finding patterns and relationships between objects. In our case, we were interested in knowing if there was any relationship or correlation between the characteristics of the graph and the performance of our algorithms.

PCA can be used to reduce and compress the data to be analyzed, PCA reduces the dimensions of the original multi-set. However, this aspect of PCA were not covered in this work. The usage of PCA also estimates the correlation structure of the variables. The importance of a variable in a PC model is indicated by the size of its residual variance. This is often used for variable selection.

6.1.1 Mathematical Definition

Considering a matrix X that contains N rows, which correspondent to N samples, and P columns that corresponds to the variables collected in each sample. PCA is defined mathematically as an orthogonal transformation. The transformation is defined as a set of P dimensional vectors of *loadings* $w_{(k)(w_1,\cdots,w_p)}$ that map each row vector $X_{(i)}$ of X to a new vector $t_{(i)}$, called the principal component.

$$t_{k(i)} = x_{(i)} w_{(k)}$$

The individual variables of a principal component inherit the maximum possible variance from X. This is an important idea because we analyzed the principal components and its variables to understand how they relate to each other.

The loadings can be obtained using a covariance matrix. Considering the covariance matrix of X, XX^T, each loading w can be obtained by considering that each vector is the eigenvector of the covariance matrix. Hence, for each correspondent eigenvalue of the eigenvector of the covariance matrix XX^T, the principal components are obtained by multiplying the eigenvector with the correspondent eigenvalue. The eigenvector with the highest correspondent eigenvalue is the most relevant principal component, the second eigenvector with the highest correspondent eigenvalue is the second most relevant principal component and so on.

6.2 Case Study

6.2.1 Smart Grid Graph Characteristics

The porpose of using PCA in this work were to understand which graph characteristics are more important when considering the results of the algorithms. In other words, we evaluated the properties of each graph, both the original and resulting graphs from applying the transformations, and we used these characteristics to make a comparison with the average ORE and computation time to know how they affect the algorithms performance.
By intuition, as the graph became more connected with the addition of edges, the results tend to improve. However, we must know how ”connected” should be a graph to obtain better results. To measure this idea of how well connected is a graph we evaluate the following properties:

- *Average Node Degree* is the average number of neighbors per node
- *Cluster Coeficient* Cluster Coefficient(CC) measure the clustering among nodes. It increases as more edges are added to the graph.
- *Average Betweenness* of a node measure how important it is in a graph by calculating the number of shortest path that contains the node. The average betweenness is the average of all nodes betweenness. With the increasing number of edges, the importance of each node considering shortest paths tend to decrease as the graph have more paths available.
- *Average Path Length* Average distance between nodes, more edges mean that are alternative ways to reach a node than may lead to short paths, decreasing the average distance or path length. In a simple way to say, as the APL decreases, the nodes become more close to each other
- *CPL* Similar to APL but instead of the average it is the median point of all distances between nodes.

The table 6.1 shows the value of all properties in each graph. We will use the data in table 6.1 and the results obtained in our tests for each algorithm to calculate the principal components and analyze them.

6.2.2 Applying PCA

As briefly explained in the previous section, we used the proprieties of the graph to compare with each algorithm test results obtained in Chapter 5. The principal components were evaluated for each new table that contained the graph characteristics and the test results for each algorithm. We separated

	CC	CPL	APL	Avg. Node Degree	Avg. Betweenness
Normal	0,011278195	0,646447432	0,660860991	2,105263158	0,080349208
Triangle Closure 25%	0,253627033	0,563451705	0,560875507	2,631578947	0,06704565
Triangle Closure 50%	0,571025466	0,386785333	0,398623684	4,285714286	0,045377149
Triangle Closure 75%	0,755581733	0,246182803	0,251596371	7,894736842	0,025827929
Triangle Closure 100%	0,924153548	0,115756788	0,127982529	34,34586466	0,009372842
Dissortativity 25%	0,011278195	0,23687275	0,246543641	2,631578947	0,025165273
Dissortativity 50%	0,681957852	0,113988136	0,113906978	3,954887218	0,007509231
Dissortativity 75%	0,520180209	0,112675682	0,111813492	6,917293233	0,007233559
Dissortativity 100%	0,87445324	0,111386197	0,108769498	13,83458647	0,006833531
AssortativityHD 25%	0,041887281	0,264778962	0,263127922	2,631578947	0,027422825
AssortativityHD 50%	0,129625534	0,237122962	0,238558188	3,954887218	0,024139113
AssortativityHD 75%	0,346145427	0,154564697	0,168563795	6,917293233	0,014783663
AssortativityHD 100%	0,432972707	0,144039205	0,157269391	13,83458647	0,013289643
AssortativityLD 25%	0,011278195	0,253831758	0,263751851	2,631578947	0,027432391
AssortativityLD 50%	0,227291632	0,144450167	0,16590877	3,954887218	0,014428855
AssortativityLD 75%	0,445239121	0,136194894	0,146034751	6,917293233	0,011785188
AssortativityLD 100%	0,600086305	0,131848447	0,134276722	13,83458647	0,010224207
Random 25%	0,019047619	0,303975856	0,312145546	2,631578947	0,033902417
Random 50%	0,045566297	0,208809303	0,211204213	3,954887218	0,02045885
Random 75%	0,05437524	0,157115568	0,156905648	6,917293233	0,013237466
Random 100%	0,1049978	0,120982144	0,12157124	13,83458647	0,00853887

FIGURE 6.1: Properties of each graph

the analysis of the principal components into two groups: first we compared the graph proprieties with average ORE obtained at the end of the tests, second we performed the same analysis but comparing the computation time instead of the average ORE. One example of a table from each group is presented.

Each table were stored in a xlsx file, each one represents our X matrix in PCA. Each row of a table contains the proprieties of the graphs plus the average ORE or computation time of the algorithm to be analyzed. Each column represents a property, the characteristics of the a graph and the average ORE or the computation time of the algorithm.

To calculate the principal components we used *Stand Alone Chemometrics Software*(SOLO), a software used for data analysis and statistics. Every table, or xlsx file, was imported to SOLO. Afterwards, the principal components were calculated by SOLO for each table. We choose the two most relevant principal components, the ones that cover more covariance, when the PCA were completed, . Them, we visualized the variables of these principal components in a chart, and analyzed its correlations.

	CC	CPL	APL	Avg. Node Degree	Avg. Betweenness	Push-Pull Results
Normal	0,011278195	646447,4318	660860,9913	2	0,080349208	3,892558206
Triangle Closure 25%	0,173049382	560283,6136	582539,4286	2	0,069906724	8,03898384
Triangle Closure 50%	0,465877132	439994,0682	451011,4592	3	0,052386344	19,40299633
Triangle Closure 75%	0,692220643	269911,5076	296624,1734	6	0,031789223	35,00856183
Triangle Closure 100%	0,866768594	185291,7348	186605,6161	13	0,017179486	10,84433998
Dissortativity 25%	0,011278195	236872,75	246543,6408	2	0,025165273	0,289702237
Dissortativity 50%	0,681957852	113988,1364	113906,9784	3	0,007509231	0,145644892
Dissortativity 75%	0,520180209	112675,6818	111813,4918	6	0,007233559	0,341719587
Dissortativity 100%	0,87445324	111386,197	108769,4981	13	0,006833531	0,060205029
Assorsativity HD 25%	0,041887281	264778,9621	263127,9221	2	0,027422825	0,393729482
Assorsativity HD 50%	0,129625534	237122,9621	238558,188	3	0,024139113	0,441297623
Assorsativity HD 75%	0,346145427	154564,697	168563,7953	6	0,014783663	0,082123069
Assorsativity HD 100%	0,432972707	144039,2045	157269,3907	13	0,013289643	0,080689396
Assorsativity LD 25%	0,011278195	253831,7576	263751,8514	2	0,027432391	0,097300822
Assorsativity LD 50%	0,227291632	144450,1667	165908,7703	3	0,014428855	0,07562992
Assorsativity LD 75%	0,445239121	136194,8939	146034,7506	6	0,011785188	0,01654885
Assorsativity LD 100%	0,600086305	131848,447	134276,7223	13	0,010224207	0,009495928
Random 25%	0,017042607	325384,5606	337044,2298	2	0,03720874	0,155579175
Random 50%	0,027622628	209402,3864	211461,1517	3	0,020491026	0,102330877
Random 75%	0,046823352	155006,3636	155733,8102	6	0,013082672	0,279218958
Random 100%	0,1114435	120541,4848	121278,1007	13	0,008498867	0,284242748

FIGURE 6.2: Table containing the graph properties and the ORE from the Push-Pull tests

	CC	CPL	APL	Avg. Node Degree	Avg. Betweenness	Extrema Propagation
Normal	0,0112782	0,64644743	0,66086099	2	0,080349208	13500
Triangle Closure 25%	0,17304938	0,56345171	0,56087551	2	0,069906724	12000
Triangle Closure 50%	0,46587713	0,38678533	0,39862368	3	0,052386344	13200
Triangle Closure 75%	0,69222064	0,2461828	0,25159637	6	0,031789223	12000
Triangle Closure 100%	0,86676859	0,11575679	0,12798253	13	0,017179486	9900
Dissortativity 25%	0,0112782	0,23687275	0,24654364	2	0,025165273	6000
Dissortativity 50%	0,68195785	0,11398814	0,11390698	3	0,007509231	3900
Dissortativity 75%	0,52018021	0,11267568	0,11181349	6	0,007233559	2700
Dissortativity 100%	0,87445324	0,1113862	0,1087695	13	0,006833531	2700
AssortativityHD 25%	0,04188728	0,26477896	0,26312792	2	0,027422825	4800
AssortativityHD 50%	0,12962553	0,23712296	0,23855819	3	0,024139113	4800
AssortativityHD 75%	0,34614543	0,1545647	0,1685638	6	0,014783663	4500
AssortativityHD 100%	0,43297271	0,14403921	0,15726939	13	0,013289643	4800
AssortativityLD 25%	0,0112782	0,25383176	0,26375185	2	0,027432391	6300
AssortativityLD 50%	0,22729163	0,14445017	0,16590877	3	0,014428855	4200
AssortativityLD 75%	0,44523912	0,13619489	0,14603475	6	0,011785188	5100
AssortativityLD 100%	0,60008631	0,13184845	0,13427672	13	0,010224207	5400
Random 25%	0,01704261	0,30397586	0,31214555	2	0,03720874	5700
Random 50%	0,02762263	0,2088093	0,21120421	3	0,020491026	4500
Random 75%	0,04682335	0,15711557	0,15690565	6	0,013082672	3600
Random 100%	0,1114435	0,12098214	0,12157124	13	0,008498867	3900

FIGURE 6.3: Table containing the graph properties and computation time from the Extrema Propagation tests

6.2.2.1 Average ORE

First we compared the average ORE obtained from each algorithm after our tests with the properties of the graph. The correlations are analyzed the same way it is analyzed the covariance of two sample sets. If both variables are positive, or negative, in both principal components, they correlate

to each other. If otherwise, one variable is positive and other is negative in the principal components, they are inversely correlated. If none of these conditions occurs, the variables are uncorrelated.

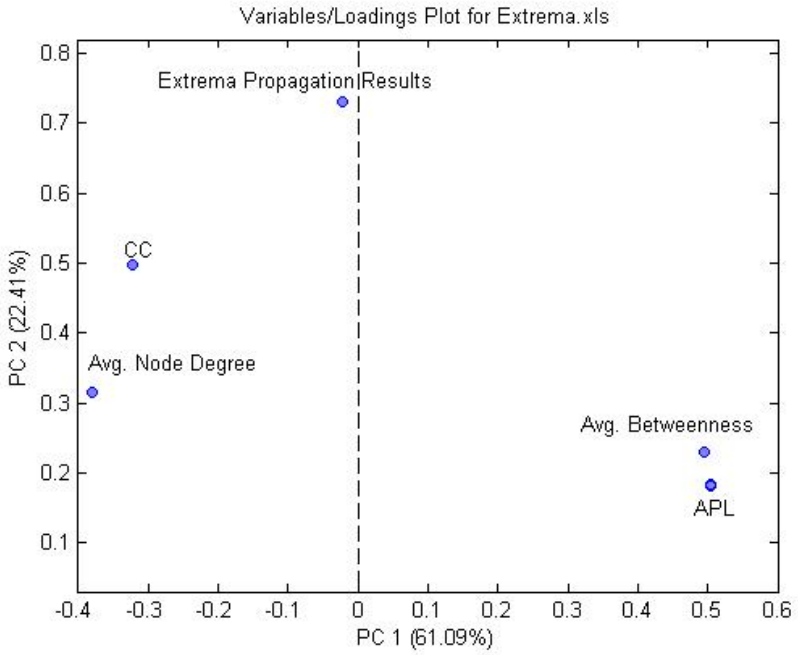

FIGURE 6.4: Chart with the value of the variable in each Principal Component for Extrema Propagation

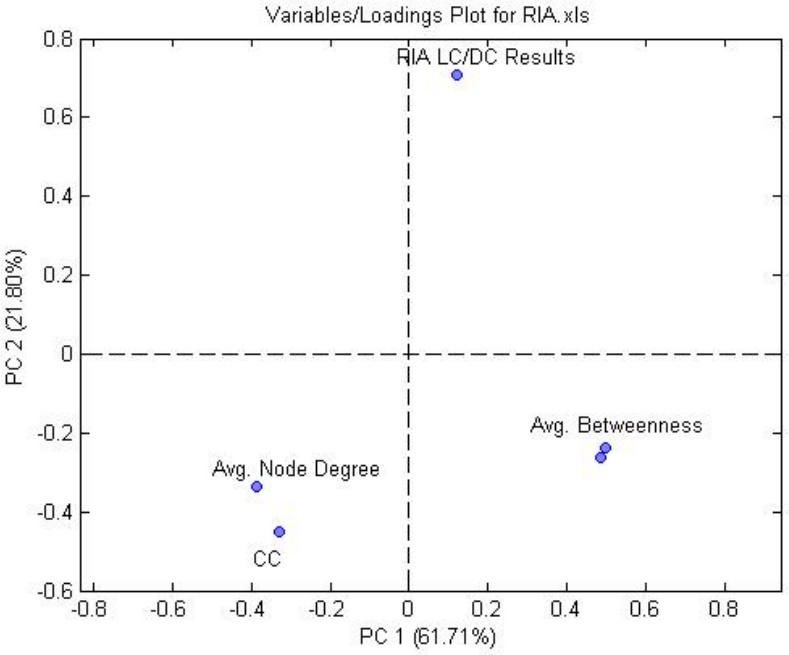

FIGURE 6.5: Chart with the value of the variable in each Principal Component for RIA LC/DC

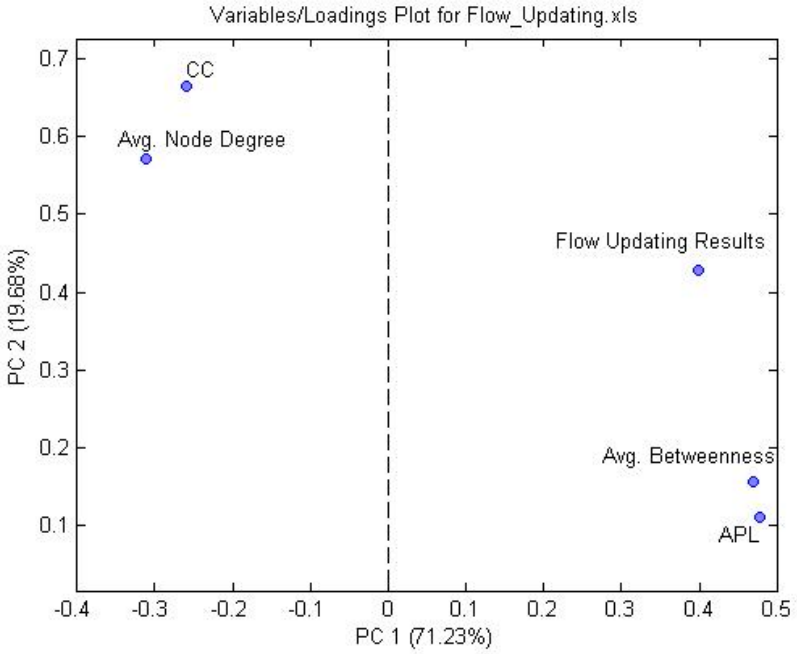

FIGURE 6.6: Chart with the value of the variable in each Principal Component for Flow Updating

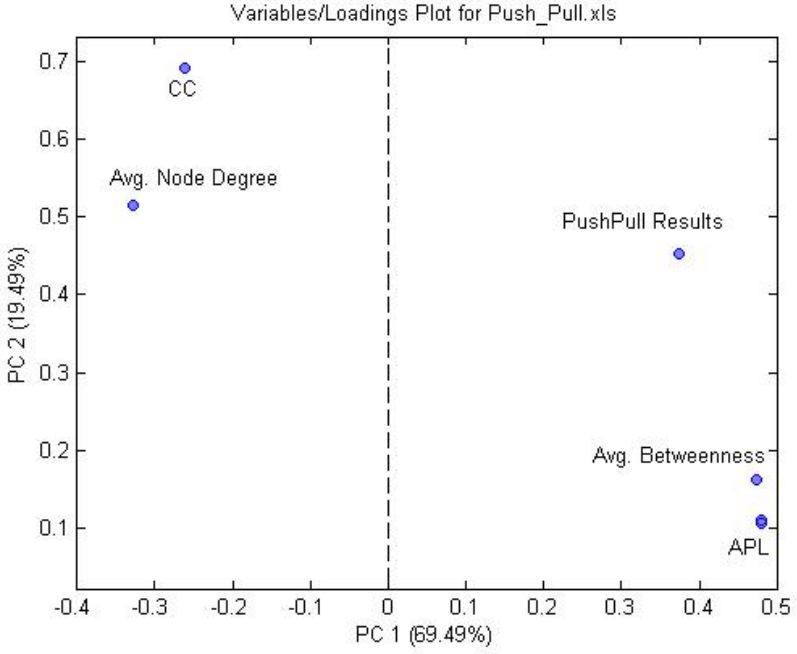

FIGURE 6.7: Chart with the value of the variable in each Principal Component for Push-Pull

In figure 6.4 we see the variables value in each principal component for the Extrema Propagation average ORE. The X axis represents the most relevant principal component and the Y axis represents the second most relevant principal component. We see that Extrema Propagation average ORE, represented with the label *Extrema Propagation Results*, in the Y axis, its value in the first principal component is almost zero. CC and average node degree are in the negative side of the X axis but

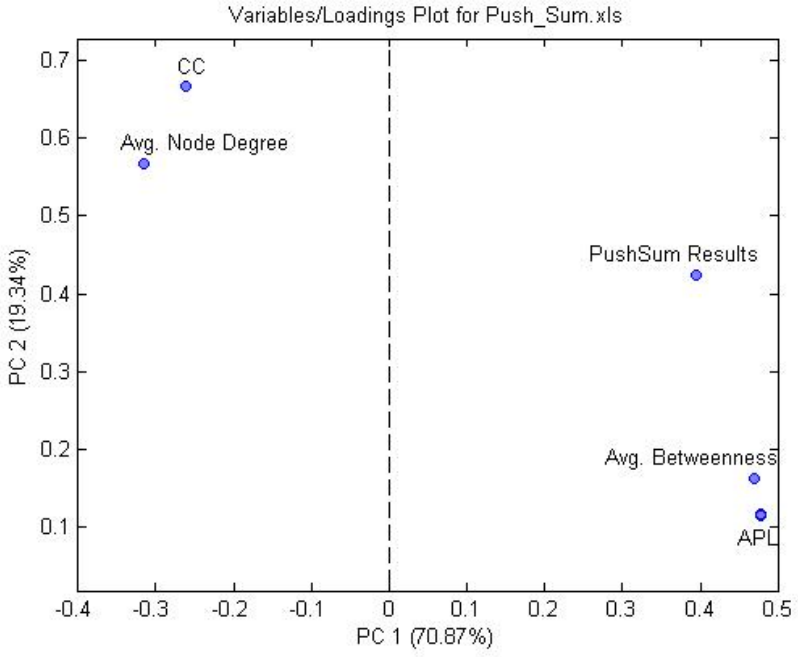

FIGURE 6.8: Table containing the graph properties and the average RMSD from the Push-Pull tests

positive in the Y axis while the other three properties, average betweenness, CPL and APL, are in both positive parts of the X and Y axis. This means that these two groups are inversely correlated. When CC and node degree increases, the other three decreases and vice-versa as well. That is not surprising, the goal of adding more edges is to reduce the distance between nodes and have more shortest paths while increasing the clustering coefficient and he average number of neighbors per node. We see that Extrema Propagation and the two groups represent a triangle in our chart, that means that Extrema Propagation results and these two groups are uncorrelated. The increasing or decreasing of the variable values in this two groups does not affect the accuracy of Extrema Propagation results. Figure 6.5 shows the same kind of analysis for RIA LC/DC. We can make the same conclusions as for Extrema Propagation case although the triangle is not so clear. The two groups that were possible to observe in Figure 6.4 are now in the negative side of the X axis in Figure 6.5. RIA LC/DC results are in positive side of the X axis and more far away from the Y axis. Since we can observe a triangle, the variables into the two groups are not correlated with the RIA LC/DC results. RIA LC/DC is the same type of algorithm as Extrema Propagation, so a similar correlation between its results and the proprieties of the graph is expected.

The PCA performed on the Flow Updating results show a different correlation between the average ORE in each test and the proprieties of the graph 6.6. The Flow Updating results are closer to the Average Betweenness, CPL and the APL, both in the positive side of the X and Y axis, that means that they are correlated. Graphs with higher CPL, APL and Average Betweenness tend to result in a higher ORE,

or higher errors. As the nodes become more far apart from each other, Flow Updating gives worse results. A negative correlation is seen between the average ORE and the average node degree and CC, the higher average number of neighbors per node and higher cluster coefficient, lower ORE is obtained while computing Flow Updating.

Analyzing figure 6.8 and figure 6.7 showing Push Sum and Push Pull PCA, the same conclusions can be made. Both graphs are almost equal to 6.6. Average Node Degree and Cluster coefficient are inversely correlated to both Push Pull and Push Sum results while APL, CPL and Average Betweenness are correlated with the ORE from the two algorithms.

It is interesting to know, considering the average ORE at the end of the tests, how close the algorithms

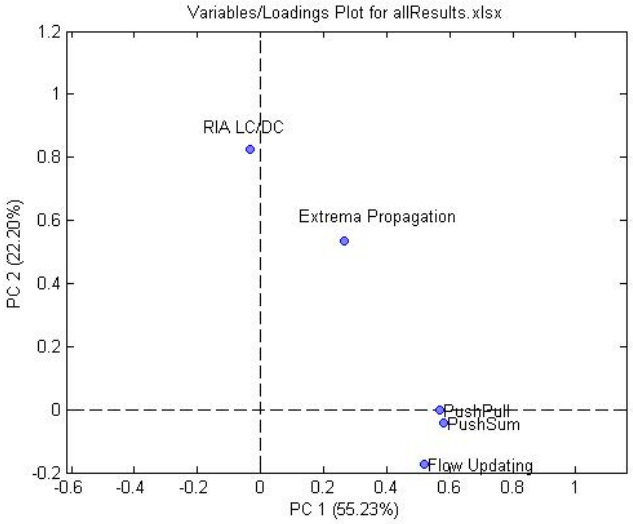

FIGURE 6.9: Chart with the value of the variable in each Principal Component for the RMSD in all algorithms

are to each other. We can conclude that *Gossip* based algorithms are close to each other, they have the same correlations between certain properties of the graph. The same can be concluded for the *Sketch* based ones. Both Extrema Propagation and RIA LC/DC don't correlate with the analyzed proprieties of the graph. To assert this conclusions about relations between the algorithm results, we performed principal components analysis using only columns that contains the average ORE per algorithm and gather them in a table showed in **??**. The results of PCA are showed in figure 6.9. We observe that *Gossip* based algorithms are very close to each other, meaning that their test results as highly correlated, Push-Pull and Push-Sum are more correlated with each other than with Flow-Updating and both three are far apart from the other two *Sketch* algorithms meaning that there is no correlation between the two groups of algorithms. Furthermore, RIA LC/DC and Extrema are in the same positive side of both x and y axis. Even though they are not as much correlated with each other as the *Gossip* ones, both two show a correlation.

6.2.2.2 Computation Time

In this section, we compare the computation time obtained from each algorithm after our tests with the properties of the graph. In Figure 6.10 and Figure 6.11 it is represented the same kind of

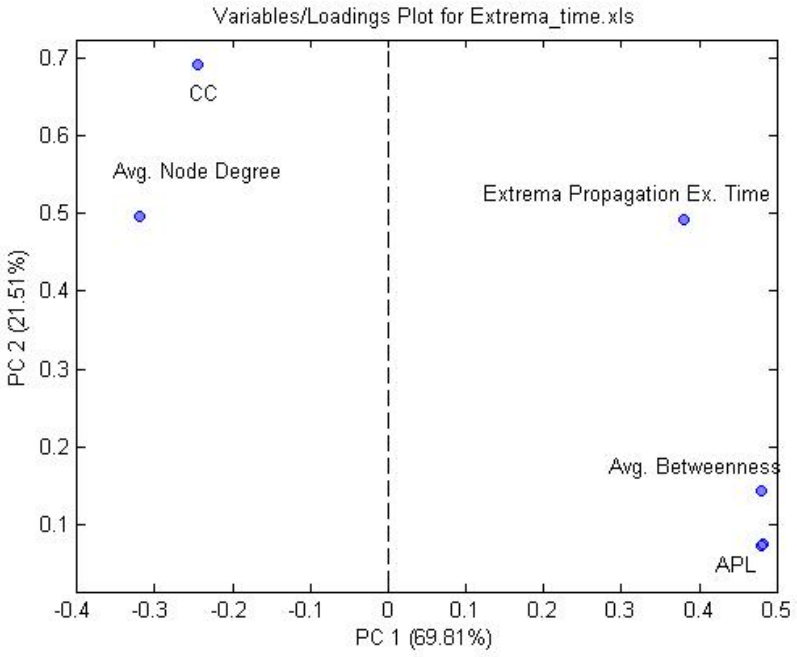

FIGURE 6.10: Table containing the graph properties and the computation time from the Extrema Propagation tests

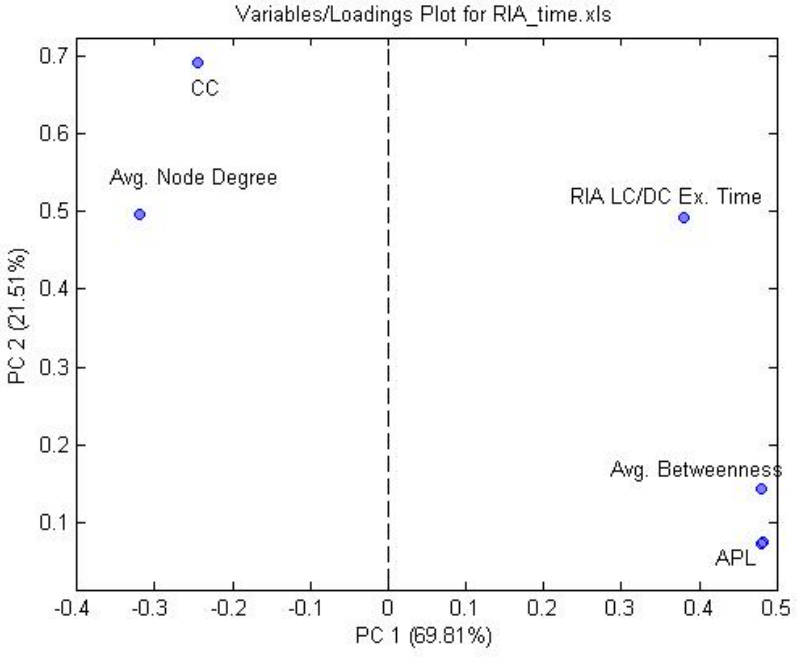

FIGURE 6.11: Table containing the graph properties and the computation time from the RIA LC/DC tests

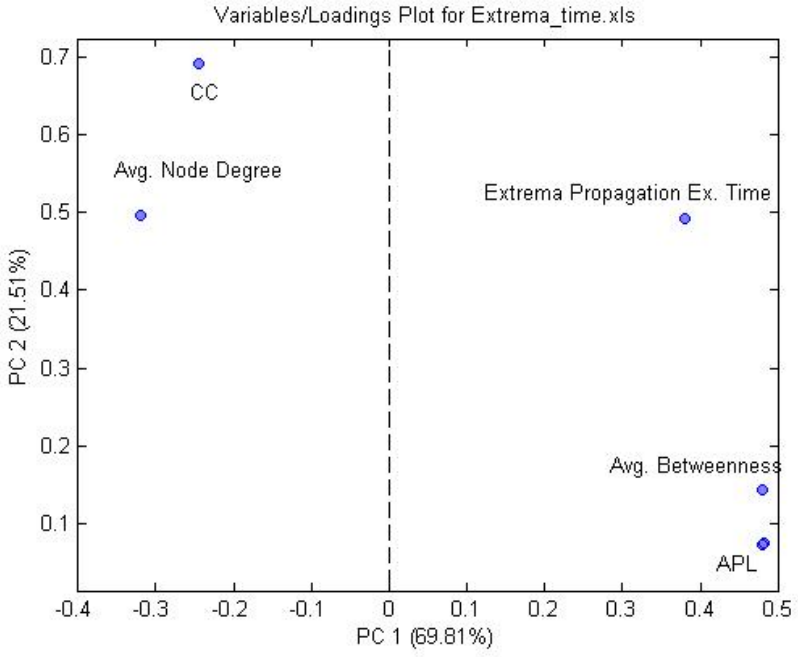

FIGURE 6.12: Table containing the graph properties and the computation time from the Flow-Updating tests

chart for *Skecth* based algorithms as seen in the previous section. Instead of a variable with the average ORE, the chart contains a variable with the computation time. Both algorithms have the same chart, meaning the same kind of correlation. RIA LC/DC and Extrema Propagation computation time are both in the positive side of the X and Y axis, the Average Betweenness, CPL and APL are there as well. That means that both variable are correlated. Graphs with higher APL, CPL and average betweenness have higher computation time for both algorithm. The explanation is simple for APL and CPL, as the node are more close to each other, the graph tend to have a lower diameter leading to lower time for *Sketch* based algorithms to compute. The explanation for the correlation between average betweenness is not so directly seen. Furthermore, the computation time from RIA LC/DC and Extrema Propagation is inversely correlated with both CC and average node degree.

The PCA made on Push-Sum and Push-Pull algorithms seen in 6.13 and 6.14 show similar correlations. Both have computation time correlated to CPL, APL and Average Betweenness but inversely correlated to average node degree and the CC. However they are slightly more far apart from APL, CPL and Average Betweenness which tells us that they are not so directly correlated as the *Sketch* based algorithms. Flow-Updating is a special case, the computation time and the characteristics of the graphs are uncorrelated as we can see in figure 6.12. Flow Updating computation time is not affected by the type of graph. We can see a very slight correlation between the computation time and average Node Degree

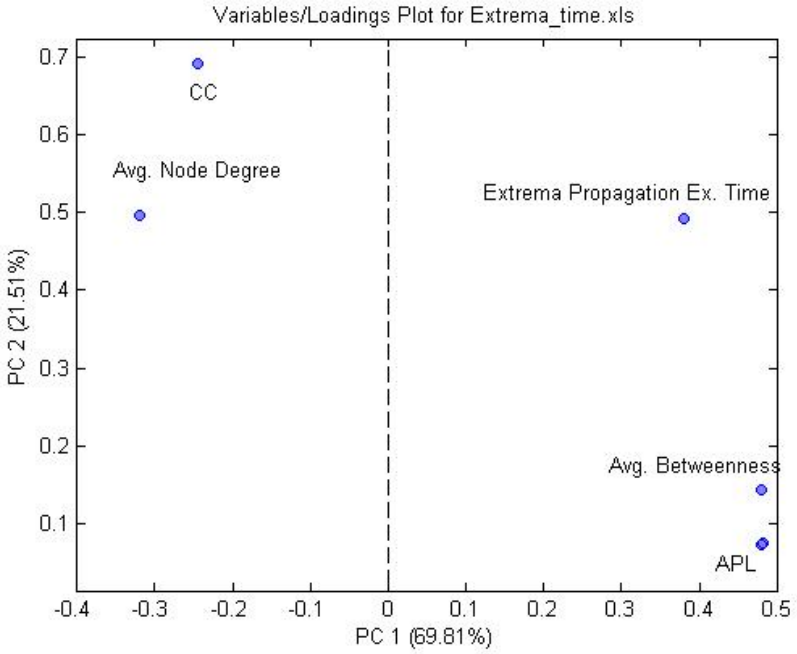

FIGURE 6.13: Table containing the graph properties and the computation time from the Push-Sum tests

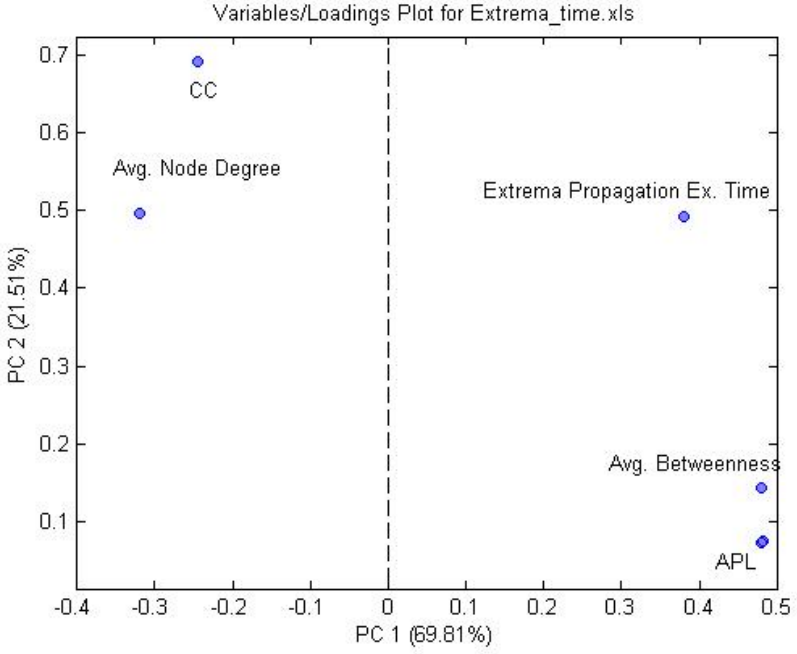

FIGURE 6.14: Table containing the graph properties and the computation time from the Push-Pull tests

and more strong with CC as the increasing of flow with higher neighbors tend to require more computation time.

We also compared the computation time of all algorithms to know how correlated and close to each other they are. With the previous charts, we had the intuition that all algorithms, except Flow Updating, have the same relation with the properties of the graph. Figure 6.15 show the PCA performed

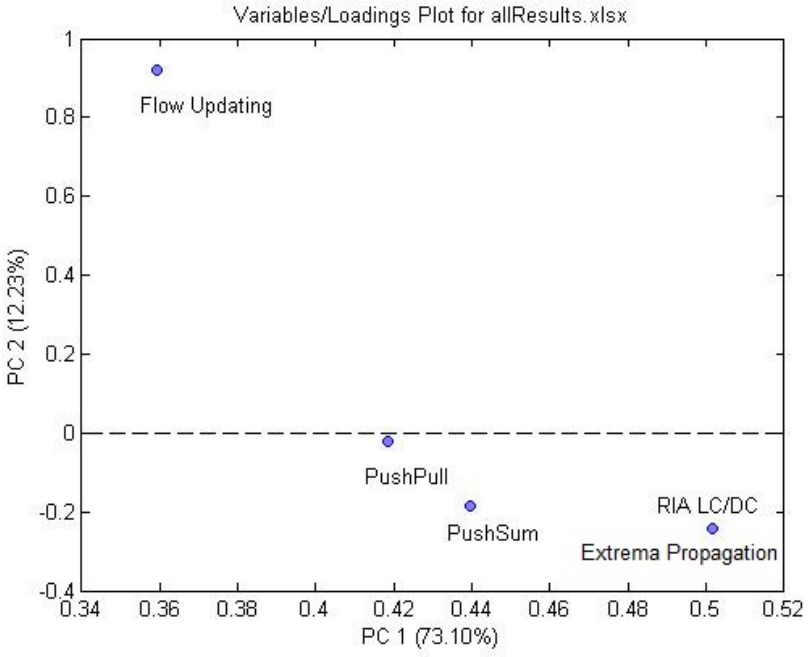

FIGURE 6.15: Chart with the value of the variable in each Principal Component for the computation time in all algorithms

on the algorithms computation time. As we can state, Push Sum and Push Pull are both close to each other meaning a strong correlation. Moreover, RIA LC/DC and Extrema Propagation are right on top of each other. These two algorithms are highly related in computation time and therefore have a strong correlation. The explanation is that both algorithm can be improved in the same way. Considering the computation time, when the algorithm diameter is reduced, both algorithms reach the best result faster.

Flow Updating have no correlation with the other four, because the proprieties of the the graph don't have much impact in the computation time of the algorithms.

6.2.3 Discussion

By analyzing our results from Chapter 5 and the proprieties of each graph was possible to know some insights on the future Smart Grids and AMR/AMI networks may be improved in order to obtain better results when using Distributed Aggregation Algorithms. Moreover, it was possible to know how the algorithms correlate with each other considering the final average error and the computation time in each graph.

As for the average ORE, the PCA performed showed us that when decreasing the APL, the CPL and the average betweenness, it is decreased also the error of the total summation when we use *Gossip* based algorithms. When improving the average number of neighbors and the CC, the error also decreases.

As each node has more neighbors in average to share the partial aggregates, the results also improve. The *Sketch* based didn't improve with the evolution and that lead to close to none correlation between the algorithms results and the final average ORE.

As for the average computation time, all algorithms improved when the APL, CPL and average betweenness is decreased. This idea is stronger in Extrema and RIA LC/DC than in Push-Pull and Push Sum. Flow Updating, however, didn't improve its results in computation time. Theres is no correlation between the Flow-Updating computation time and the graph characteristics. It is a clue that Flow-Updating cannot be improved in computation time, if so, it was not possible to be asserted in this work.

Chapter 7

Conclusion

This work presents a specific scenario for the implementation of distributed aggregation algotihms in a Smart Grid/AMI network and gives an experience and an ideia about how can these protocols adapt in the AMI network.

With this work, it was possible to obtain leads about how the algorithms can perform in a Smart Grid topology. Even though the tests we made were only in one power grid sample, the radial structure is similar to many other Low Voltage grids. The set of tests made possible to know that *Gossip* based algorithms perform poorly and that they are not a suitable choice when implementing a distributed aggregation algorithm in this type of topology. On the other hand, *Sketch* based algorithm produce good results in both accuracy and time and they can be tested in practical scenarios.

When evolving our network by adding more edges according to the transformation policies seen in [41]. We've seen that these transformations improve not only the resilience, efficiency and robustness of the network, but also improve the results of the tested algorithms, especially the *Gossip* based one since they give more accurate results, even more accurate than the *Sketch* based ones.

Furthermore, our statistical analysis using principal components analysis gave us an understanding and leads considering the proprieties and characteristics the future AMI/Smart Grid networks should follow in order to use and obtain better accuracy and speed when using distributed aggregation algorithms.

7.0.4 Future Work

From this point, there is several work directions that can be sougth. In this work we used a theoretical Smart Grid network topology. Experiments in the field may take place from this work in order to validate our conclusions. Using a set of data collectors connected by power line communication, gathering consumption or production information by a set of connected meters

Although we considered a real time pricing scenario, we did not test the efficiency of the algorithms when consider a real time pricing practical solutions. The variations of price the various distributed algorithms can achieve and the accuracy of the decision to increase or decrease price according to consumption and production remains yet to be explored.

Moreover, no tests were made for some constrains of real examples of an real fully implemented AMI, such as dynamic reading and how can they affect the performances of the algorithms, the need or not to individualize the metering data to evaluate the consumption of a specific household and how better or worse the algorithms would perform if different message sizes where used considering the specification of each algorithm.

Other applications for the distributed algorithms in a Smart Meter Network can be explored, the evaluation of the average consumption, the maximum and minimum consumption and it's relevance for the grid is a matter that can pose some challenges. There is also application for Smart home that can use this algorithms for its internal management, evaluate the maximum consumption device, minimum consumption device and the total consumption are problems that can be solved by the usage of a management software that use our distributed aggregation algorithms.

Appendix A

Graph Theory and Complex Network Fundamentals

The approach used in this Thesis to model the Smar grid graph and the evoltuion strategies is based on Graph Theory and Complex Networks Theory. All the definitions for Graph Theory and Complex Networks we use throughout this thesis and presented bellow are taken from [41].
We recall the definition of a Smart Grid Graph.

Definition A.1. DEFINITION(SMART GRID GRAPH). A Smart Grid graph is a graph $G(V, E)$ such that each element $v_i \in V$ is either a substation with electrical equipment or a transformer os a physical power grid with a data collector installed, There is a edge $e_i, j = (v_i, v_j) \in E$ between two nodes if there is physical cable that enables a PowerLine communication connecting directly the element represented by v_i and vj

Definition A.2. DEFINITION(WEIGHTED GRAPH). A Weighted Smart Grid graph is a Smart Grid graph $G_w(V, E)$ with and additional function $f : E \to \mathbb{R}$ associating a real number to an edges representing the channel capacity of the corresponding cable in bps

Definition A.3. DEFINITION(ORDER AND SIZE OF A GRAPH) given the graph G the order is given by $N = |V|$, while the size is given by $M = |E|$. Average node degree is given by $< k >= \frac{2M}{N}$

Definition A.4. DEFINITION(ADJACENY, NEIGHBORHOOD AND DEGREE). If $e_{x,y} \in E$ is an edge in graph G, then x and y are adjacent, or neighboring, vertexes, and the vertexes x and y are incident with the edge $e_{x,y}$. The set of vertexes adjacent to a vertex $x \in V$, called the neighborhood of x, is denoted by Γ_x. The number $d(x) = |\Gamma_x|$ is the degree of x.

Definition A.5. DEFINITION (WEIGHTED DEGREE). Let $x \in V$ be a vertex in a weighted graph G, the weighted degree of $x, d_w(x)$ is:

$$d_w = \sum_{y \in \Gamma(x)} w_{x,y}$$

where $w_{x,y}$ is the wieght of the edge joining vertexes x and y, and $\Gamma(x)$ is the neighborhood of x

Definition A.6. DEFINITION(CLUSTERING COEFICIENT (CC)). The clustering coefficient γ_v of Γ_v is

$$\gamma_v = \frac{E(\Gamma_v)}{\binom{k_v}{2}}$$

where $|E(\Gamma_v)|$ is the number of edges in the neighborhood of v and $\binom{k_v}{2}$ is the total number of possible edges in Γ_v

This local property of a node can be extended to an entire graph by averaging over all nodes.

One important property is how much any two nodes are far apart from each other, in particular the minimal distance between them or shortest path. The concepts of $path$ and $pathlength$ are crucial to understande the way two vertexes are connected

Definition A.7. DEFINITION(PATH AND PATH LENGTH) A path of G is a subgraph P of the form:

$$V(P) = \{x_0, x_1, \cdots, x_l\}$$

$$E(P) = \{(x_0, x_1), (x_1, x_2), \cdots, (x_{l-1}, x_l)\}$$

Definition A.8. such that $V(P) \subseteq V$ and $E(P) \subseteq E$. The vertexes x_0 and x_l are end-vertexes of P and $l = |E(P)|$ is the length of P. A graph is connected if for any two distinct veretxes $v_i, v_j \in V$ there is a finite path from v_i to v_j

Definition A.9. (AVERAGE PATH LENGTH(APL)). Let $v_i \in V$ be a vertex in a graph G. The average path length for GL_{av} is

$$L_{av} = \frac{1}{N(N-1)} \sum_{i \neq j} d(v_i, v_j)$$

where $d(v_i, v_j)$ is the finite distance between v_i and v_j and N is the order of G.

Definition A.10. DEFINITION(CHARACTERISTIC PATH LENGTH(CPL)). Let $v_i \in V$ be a vertex in graph G, the characteristic path length for G, L_{cp} if defined as the median of d_{vi} where:

$$d_{vi} = \frac{1}{(N-1)} \sum_{i \neq j} d(v_i, v_j)$$

is the mean of the distances conencting v_i to any other vertex v_j in G and N is the order of G.

Definition A.11. DEFINITION(WEIGHTED CHARACTERISTIC PATH LENGTH(WCPL)). The weighted characteristic path length for graph G, L_{wcp} is the median for $(v_i, v_j) \in V$ of d_{wi} where.

$$d_{wi} = \frac{1}{(N-1)} \sum_{i \neq j} e_{w_{i,j}}$$

is the the mean of the weighted distances connecting v_i, to any other vertex v_j and $e_{w_{i,j}}$ is an edge in the minimal weighted path between v_i and v_j in G and N is the order of G.

Bibliography

[1] Svante Wold, Kim Esbensen, and Paul Geladi. "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1 (1987), pp. 37–52.

[2] Mark EJ Newman. "Assortative mixing in networks". In: *Physical review letters* 89.20 (2002), p. 208701.

[3] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-based computation of aggregate information". In: *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*. IEEE. 2003, pp. 482–491.

[4] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. "Approximate aggregation techniques for sensor databases". In: *Data Engineering, 2004. Proceedings. 20th International Conference on*. IEEE. 2004, pp. 449–460.

[5] Joao Girao, Markus Schneider, and Dirk Westhoff. "CDA: Concealed Data Aggregation in Wireless Sensor Networks". In: *ACM Workshop on Wireless Security*. Poster presentation. WiSe 2004. Philadelphia, USA, Oct. 2004. URL: `http://www.ece.cmu.edu/~adrian/wise2004/`.

[6] Márk Jelasity and Alberto Montresor. "Epidemic-style proactive aggregation in large overlay networks". In: *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*. IEEE. 2004, pp. 102–109.

[7] Claude Castelluccia. "Efficient aggregation of encrypted data in wireless sensor networks". In: *In MobiQuitous*. IEEE Computer Society, 2005, pp. 109–117.

[8] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. "Efficient aggregation of encrypted data in wireless sensor networks". In: *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*. IEEE. 2005, pp. 109–117.

[9] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. "Gossip-based aggregation in large dynamic networks". In: *ACM Transactions on Computer Systems (TOCS)* 23.3 (2005), pp. 219–252.

[10] Haowen Chan, Adrian Perrig, and Dawn Song. "Secure hierarchical in-network aggregation in sensor networks". In: *Proceedings of the 13th ACM conference on Computer and communications security*. ACM. 2006, pp. 278–287.

[11] Yao-Chung Fan and Arbee LP Chen. "Efficient and robust sensor data aggregation using linear counting sketches". In: *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE. 2008, pp. 1–12.

[12] David G Hart. "Using AMI to realize the Smart Grid". In: *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*. IEEE. 2008, pp. 1–2.

[13] Vasconcelos and Jorge. *Survey of Regulatory and Technological Developments Concerning Smart Metering in the European Union Electricity Market*. EUI-RSCAS Working Papers 1. European University Institute (EUI), Robert Schuman Centre of Advanced Studies (RSCAS), Sept. 2008. URL: `http://ideas.repec.org/p/erp/euirsc/p0193.html`.

[14] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. "Fault-tolerant aggregation by flow updating". In: *Distributed Applications and Interoperable Systems*. Springer. 2009, pp. 73–86.

[15] Bo Yu, Jianzhong Li, and Yingshu Li. "Distributed data aggregation scheduling in wireless sensor networks". In: *INFOCOM 2009, IEEE*. IEEE. 2009, pp. 2159–2167.

[16] BA Akyol, Harold Kirkham, S Clements, and M Hadley. "A survey of wireless communications for the electric power system". In: *Prepared for the US Department of Energy* (2010).

[17] Mauro Biagi and Lutz Lampe. "Location assisted routing techniques for power line communication in smart grids". In: *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE. 2010, pp. 274–278.

[18] Hassan Farhangi. "The path of the smart grid". In: *Power and Energy Magazine, IEEE* 8.1 (2010), pp. 18–28.

[19] Palak P Parikh, Mitalkumar G Kanabar, and Tarlochan S Sidhu. "Opportunities and challenges of wireless communication technologies for smart grid applications". In: *Power and Energy Society General Meeting, 2010 IEEE*. IEEE. 2010, pp. 1–7.

[20] Pedram Samadi, A-H Mohsenian-Rad, Robert Schober, Vincent WS Wong, and Juri Jatskevich. "Optimal real-time pricing algorithm based on utility maximization for smart grid". In: *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE. 2010, pp. 415–420.

[21] Wasim M Taqqali and Nidhal Abdulaziz. "Smart grid and demand response technology". In: *Energy Conference and Exhibition (EnergyCon), 2010 IEEE International*. IEEE. 2010, pp. 710–715.

[22] Soma Shekara Sreenadh Reddy Depuru, Lingfeng Wang, and Vijay Devabhaktuni. "Smart meters for power grid: Challenges, issues, advantages and status". In: *Renewable and Sustainable Energy Reviews* 15.6 (2011), pp. 2736–2742. URL: http://EconPapers.repec.org/RePEc:eee:rensus:v:15:y:2011:i:6:p:2736-2742.

[23] Mostafa M Fouda, Zubair Md Fadlullah, Nei Kato, Rongxing Lu, and Xuemin Shen. "A lightweight message authentication scheme for smart grid communications". In: *Smart Grid, IEEE Transactions on* 2.4 (2011), pp. 675–685.

[24] Stefano Galli, Anna Scaglione, and Zhifang Wang. "For the grid and through the grid: The role of power line communications in the smart grid". In: *Proceedings of the IEEE* 99.6 (2011), pp. 998–1027.

[25] U.S. Government. *Nist Framework and Roadmap for Smart Grid Interoperability Standards, Release 1 0*. General Books, 2011. ISBN: 9781234541682. URL: http://books.google.pt/books?id=-C7UygAACAAJ.

[26] Vehbi C Gungor, Dilan Sahin, Taskin Kocak, Salih Ergut, Concettina Buccella, Carlo Cecati, and Gerhard P Hancke. "Smart grid technologies: communication technologies and standards". In: *Industrial informatics, IEEE transactions on* 7.4 (2011), pp. 529–539.

[27] Paulo Jesus, Carlos Baquero, and Paulo Sérgio Almeida. "A Survey of Distributed Data Aggregation Algorithms". In: *CoRR* abs/1110.0725 (2011). URL: http://dblp.uni-trier.de/db/journals/corr/corr1110.html#abs-1110-0725.

[28] Tarek Khalifa, Kshirasagar Naik, and Amiya Nayak. "A survey of communication protocols for automatic meter reading applications". In: *Communications Surveys & Tutorials, IEEE* 13.2 (2011), pp. 168–182.

[29] Lutz Lampe and AJ Vinck. "On cooperative coding for narrow band PLC networks". In: *AEU-International Journal of Electronics and Communications* 65.8 (2011), pp. 681–687.

[30] Björn Lohrmann and Odej Kao. "Processing smart meter data streams in the cloud". In: *Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on*. IEEE. 2011, pp. 1–8.

[31] Giacomo Verticale Cristina Rottondi and Christoph Krauß. "Implementation of a Protocol for Secure Distributed Aggregation of Smart Metering Data". In: *International Conference on Smart Grid Technology, Economics and Policies (SG-TEP 2012)*. IEEE, Nov. 2012.

[32] Xi Fang, Satyajayant Misra, Guoliang Xue, and Dejun Yang. "Smart Grid - The New and Improved Power Grid: A Survey." In: *IEEE Communications Surveys and Tutorials* 14.4 (2012), pp. 944–980. URL: http://dblp.uni-trier.de/db/journals/comsur/comsur14.html#FangMXY12.

[33] Yide Liu. "Wireless Sensor Network Applications in Smart Grid: Recent Trends and Challenges." In: *IJDSN* 2012 (2012). URL: http://dblp.uni-trier.de/db/journals/ijdsn/ijdsn2012.html#Liu12.

[34] Alberto Sendin, Inigo Berganza, Aitor Arzuaga, Anssi Pulkkinen, and Il Han Kim. "Performance results from 100,000+ PRIME smart meters deployment in Spain". In: *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*. IEEE. 2012, pp. 145–150.

[35] Omar Asad, Melike Erol-Kantarci, and Hussein T Mouftah. "A Survey of Sensor Web Services for the Smart Grid". In: *Journal of Sensor and Actuator Networks* 2.1 (2013), pp. 98–108.

[36] Zekeriya Erkin, Juan Ramón Troncoso-Pastoriza, Reginald L. Lagendijk, and Fernando Pérez-González. "Privacy-Preserving Data Aggregation in Smart Metering Systems: An Overview." In: *IEEE Signal Process. Mag.* 30.2 (2013), pp. 75–86. URL: http://dblp.uni-trier.de/db/journals/spm/spm30.html#ErkinTLP13.

[37] S. Ghosh, Manisa Pipattanasomporn, and Saifur Rahman. "Technology deployment status of U.S. smart Grid projects - Electric distribution systems." In: *ISGT*. IEEE, 2013, pp. 1–8. ISBN: 978-1-4673-4894-2. URL: http://dblp.uni-trier.de/db/conf/isgt/isgt2013.html#GhoshPR13.

[38] A Hagberg, D Schult, and P Swart. "NetworkX". In: *URL http://networkx. github. io/index. html* (2013).

[39] Dejan Ilić, Stamatis Karnouskos, and Martin Wilhelm. "A Comparative Analysis of Smart Metering Data Aggregation Performance". In: *IEEE 11th International Conference on Industrial Informatics (INDIN), Bochum, Germany*. July 2013. URL: http://diktio.dyndns.org/files/2013_INDIN_aggregationPerformance.pdf.

[40] Pedro Godinho Matos, Antonio Aires Messias, Pedro Ricardo Daniel, Manuel Sao Miguel Oliveira, Ana Margarida Veiga, and Paulo Libano Monteiro. "Inovgrid, a smart vision for a next generation distribution system". In: *Electricity Distribution (CIRED 2013), 22nd International Conference and Exhibition on*. IET. 2013, pp. 1–4.

[41] Giuliano Andrea Pagani and Marco Aiello. "From the Grid to the Smart Grid, Topologically". In: *arXiv preprint arXiv:1305.0458* (2013).

[42] Keita Suzuki, Chuzo Ninagawa, Hiroki Yoshida, Seiji Kondo, Junji Morikawa, Taiga Kanbe, and Takao Aoki. "Smart grid ADR aggregation delay model on large-scale distributed building HVAC facilities". In: *ISGT Europe*. 2013, pp. 1–5.

[43] Xue Lin, Yanzhi Wang, and Massoud Pedram. "Designing the Optimal Pricing Policy for Aggregators in the Smart Grid". In: *Green Technologies Conference (GreenTech), 2014 Sixth Annual IEEE*. IEEE. 2014, pp. 75–80.

[44] *Hoogkrek, PowerMatching City*. URL: http://www.powermatchingcity.nl/.

[45] IBM. *Informix TimeSeries*. URL: http://www-01.ibm.com/software/data/informix/timeseries/.

[46] Pecan Street Research Institute. *Pecan Street Project*. URL: http://www.pecanstreet.org/.

[47] OPower. *OPower 4*. URL: http://opower.com/company/news-press/press_releases/67.

[48] *The DEHEMS Project*. URL: http://www.dehems.eu/.

[49] G. N. Ericsson. "Cyber Security and Power System Communication—Essential Parts of a Smart Grid Infrastructure". In: *IEEE Transactions on Power Delivery* 25.3 (July 2010), pp. 1501–1507. ISSN: 0885-8977. DOI: 10.1109/MSP.2010.49. URL: http://dx.doi.org/10.1109/MSP.2010.49.