MASTER'S DEGREE IN INDUSTRIAL ENGINEERING

MASTER'S THESIS

BATTERY NETWORK DESIGN USING QUANTUM COMPUTING

Author: Cristina Barragán Castro

Industrial Advisor: Miguel Rodríguez Asensio

Academic Supervisor: Carlos Mateo Domingo

Madrid

August 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

BATTERY NETWORK DESIGN USING QUANTUM COMPUTING

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2022/2023 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Cristina Barragán Castro Fecha: 29 / 08 / 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Miguel Rodríguez Asensio Fecha: 29 / 08 / 2023

Fdo.: Carlos Mateo Domingo Fecha: 29 / 08 / 2023

MASTER'S DEGREE IN INDUSTRIAL ENGINEERING

MASTER'S THESIS

BATTERY NETWORK DESIGN USING QUANTUM COMPUTING

Author: Cristina Barragán Castro

Industrial Advisor: Miguel Rodríguez Asensio

Academic Supervisor: Carlos Mateo Domingo

Madrid

August 2023

DISEÑO DE UNA RED DE BATERÍAS USANDO COMPUTACIÓN CUÁNTICA

Autor: Barragán Castro, Cristina.
Director: Rodríguez Asensio, Miguel y Mateo Domingo, Carlos.
Entidad Colaboradora: i-DE

RESUMEN DEL PROYECTO

Partiendo del código diseñado para el proyecto Flexener T.3.1.2, que emplea algoritmos genéticos para diseñar una red de baterías que maximice el control de tensiones y la fiabilidad a la vez que se minimice la inversión, se estudia la posibilidad de pasarlo a computación cuántica. Tras un análisis de escalabilidad del algoritmo se determina que la optimización es la parte más compleja y, por tanto, la que más puede beneficiarse del uso de la computación cuántica. Se decide emplear un Quantum Annealer, por lo que es necesario reformular el problema para utilizar formulación QUBO. De esta manera, se espera poder trabajar con redes de mayor tamaño y obtener una posible aceleración en el tiempo de computación necesario para resolver el problema.

Palabras Clave: Optimización, Baterías, Algoritmo Genético, Computación Cuántica, Quantum Annealer, QUBO

1. Introducción

La creciente incorporación de generación distribuida en la red de distribución, particularmente de energías renovables, y el crecimiento de la demanda debido a la electrificación de la economía suponen un gran reto para la operación y planificación de la red. La instalación de sistemas de almacenamiento, y en concreto de baterías, en la red trae consigo una mejora en la flexibilidad de la red, indispensable para equilibrar generación y demanda en escenarios como el actual con una gran incertidumbre. De esta manera, permiten lidiar con la variabilidad de la generación renovable, cargándose durante picos de generación y valles de demanda, e inyectando energía en la red en la situación contraria, lo que facilita la integración en el sistema de energías renovables. Además, las baterías traen consigo una cantidad de ventajas, como la posibilidad de actuar como generadores de reserva en caso de falta mejorando así la fiabilidad del sistema gracias a su rápida capacidad de respuesta. Esto es especialmente importante en sistemas radiales con poca capacidad de reconfiguración, ya que en situaciones de falta resulta más complicado mantener el suministro. Para este proyecto, esta mejora en fiabilidad, junto con su capacidad para ayudar al control de tensiones, son las características más relevantes de las baterías. El control de tensiones está directamente relacionado con su capacidad de control sobre el intercambio de potencia activa y reactiva con la red, mientras que la mejora en fiabilidad está relacionada con su capacidad de proporcionar alimentación en zonas aisladas. Todo esto, junto con el descenso experimentado en el precio de las baterías en los últimos años, hacen a estos sistemas de almacenamiento una opción muy atractiva para el desarrollo de la red eléctrica. Es por ello, que sería extremadamente útil poder diseñar de manera rápida y efectiva redes de baterías para la red de distribución. No obstante, debido al gran número de variables y posibles localizaciones para las baterías, esto es un problema muy complejo computacionalmente. De esta manera, el presente proyecto estudia su implementación empleando computación cuántica.

2. Descripción del Proyecto Flexener

Este proyecto parte del proyecto Flexener, concretamente de la tarea T.3.1.2, desarrollada por la Universidad Pontifica Comillas en colaboración con i-DE. El objetivo de la misma es determinar la capacidad y ubicación óptima de baterías en la red que cumpla con tres objetivos fundamentales: maximizar el control de tensiones en condiciones normales de operación, maximizar la fiabilidad y minimizar la inversión. Esta optimización de resuelve mediante algoritmos genéticos. Para ello, se analiza una red radial de 20kV con generación distribuida que tiene 263 nudos. De esta manera, se desarrolla un algoritmo en MATLAB que puede dividirse en tres partes fundamentales: el pre-procesado de los datos, la optimización y el post-procesado.

Comenzando por el pre-procesado, este a su vez puede dividirse en 3 actividades principales. La primera es cargar los datos en el sistema y prepararlos para su posterior utilización. Esto consiste en transformarlos a formato interno y reordenarlos para evitar errores a la hora de utilizar matpower y facilitar los cálculos posteriores. En segundo lugar, la red se divide en zonas de acuerdo a la posición de los interruptores telemandados. Para cada una de estas zonas, se calculan los parámetros fundamentales que serán necesarios posteriormente para la optimización. Por otra parte, en caso de falta, la zona en que ha ocurrido el fallo se aislará y el resto se reconfigurará, en la medida de lo posible, formando microgrids para mantener el suministro. Dichos microgrids también se identifican durante la etapa de pre-procesado. Por último, dado que el control de tensiones depende del nudo concreto en que se instalan las baterías, se determina el nudo de cada zona en que el impacto de colocar las baterías sería mayor con respecto al equilibrio de tensiones.

En cuanto a la optimización, como se ha comentado anteriormente se realiza empleando algoritmos genéticos y a nivel de zona para facilitar su convergencia. La variable de optimización empleada es continua y representa la capacidad de batería instalada en cada una de las zonas. Cuenta con tres funciones objetivo: la maximización del control de tensiones; la maximización de la fiabilidad, que se modela como la minimización del impacto económico de las faltas; y la minimización de la inversión, que se modela teniendo en cuenta las economías de escala.

Por último, con respecto al post-procesado, los algoritmos genéticos son un mecanismo metaheurístico para resolver problemas de optimización, de manera que no garantizan la optimalidad de la solución, pero si devuelven soluciones razonables. De esta manera, el algoritmo no devuelve una única solución sino multitud de ellas que se presentan de dos maneras: mediante un gráfico que representa el impacto en las tres métricas de incrementar la capacidad de baterías instaladas, y una matriz que contiene los valores de X para cada solución. De esta manera, atendiendo a las tendencias en ambos gráficos, el usuario puede tomar una decisión acerca de la opción final de diseño.

3. Análisis del Tiempo de Computación y Escalabilidad

Se analiza el tiempo de computación y la escalabilidad del algoritmo desarrollado en el proyecto Flexener para determinar que partes del algoritmo son más computacionalmente complejas y, por tanto, podrían beneficiare en mayor medida de la introducción de la computación cuántica.

En el análisis del tiempo de computación se ve que, para un modelo que incluye el cálculo del flujo de cargas, el 56% del tiempo total de computación se emplea para la optimización, mientras que el 43% se emplea para el pre-procesado. Por otra parte, se observa que hay dos funciones concretas con una complejidad mayor que el resto: el flujo de cargas y el algoritmo genético. Dentro de este último, una gran cantidad del tiempo de computación se dedica a funciones internas. Por otra parte, se observa que dentro las distintas funciones objetivo, la más compleja es la fiabilidad. Esto se comprueba mediante el análisis de escalabilidad a nivel de zona, cuyos resultados se muestran en la

Figura 1. Debido a la falta de datos, no ha sido posible realizar un análisis de la escalabilidad a nivel de nudo, por lo que no se conoce el ritmo de crecimiento de la función que calcula el flujo de cargas, aunque se estima que es menor que el de la optimización.

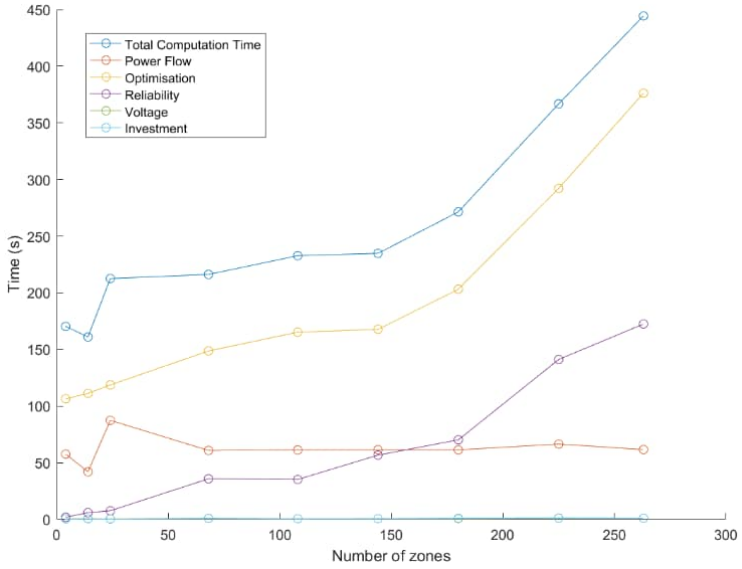


Figura 1: Análisis de Escalabilidad a Nivel de Zona para el Algoritmo del Proyecto Flexener

Con este estudio, se comprueba que el crecimiento en el tiempo de computación está muy ligado al crecimiento de la optimización, y en concreto de la fiabilidad. Dado que el número de puntos no es muy elevado y que los algoritmos genéticos son inherentemente caóticos, resulta complicado determinar con certeza el ritmo de crecimiento del tiempo de computación. No obstante, se concluye que es, como mínimo, cúbico, ya que es el ritmo de crecimiento de la función de fiabilidad.

Con todo lo anterior se concluye que hay dos funciones que podría resultar interesante pasar a computación cuántica: la optimización y el flujo de cargas. No obstante, las tecnologías cuánticas actuales no permiten calcular flujos de cargas para redes grandes, pero muestran resultados prometedores para un futuro en que la tecnología esté más madura. Es por ello que el proyecto se centra en la optimización.

4. Elección de Tecnología Cuántica y de Formulación

Existen diversas tecnologías cuánticas que permiten resolver problemas de optimización.

En primer lugar, se encuentran los ordenadores cuánticos de puertas, en los que se emplearía un algoritmo QAOA. Esta opción es descartada porque este tipo de ordenadores son muy sensibles al ruido, lo que compromete la optimalidad de la solución. Por otra parte, su número de qubits es muy reducido, de manera que no permitirían trabajar con redes de gran tamaño.

Otra opción sería emplear una solución Quantum-Inspired, que consiste en emplear un ordenador convencional que simula las propiedades de un ordenador cuántico. El algoritmo a emplear sería el QIOA, que es una adaptación del QAOA para este tipo de ordenadores. No obstante, esta opción no está aún muy madura y, además, requiere una gran cantidad de recursos computacionales, por lo que podría generar problemas para trabajar con redes de gran tamaño.

Es por ello que se decide utilizar un Quantum Annealer, que es un tipo de ordenador cuántico cuyo propósito principal es resolver problemas de optimización y sampling. Además, su sensibilidad ante el ruido es mucho menor que para los Ordenadores Cuánticos de Puertas, y tiene un mayor

número de qubits, lo que permite trabajar con grandes redes. Para utilizar este tipo de ordenador, es necesario emplear formulación QUBO (Quadratic Unconstrained Binary Optimisation), de manera que es necesario reformular el problema.

5. Aproximación Cuántica del Problema

Existen dos maneras de enfocar la resolución del problema empleando tecnologías cuánticas:

- Enfoque Cuantitativo: Consiste en reformular el problema original para adaptarlo a la formulación QUBO. De esta manera, en primer lugar, se reformula el problema para una variable continua, pero eliminando los algoritmos genéticos y unificando las tres funciones objetivo, para lo que se emplea formulación clásica. Posteriormente, se discretiza el problema, de manera que la variable de optimización pasa a representar si una determinada capacidad de batería previamente fijada se instala en una zona o no. Por último, se eliminan las restricciones. De esta manera, tenemos un problema en formulación clásica que cumple con las condiciones de un QUBO. Esta formulación posteriormente debe ser adaptada para cumplir con la forma típica de los problemas QUBO, calculando la matriz Q.

- Enfoque Heurístico: Consiste en resolver el problema sin efectuar flujos de cargas y reduciendo el número de cálculos necesarios, empleando el sentido común. De esta manera, se realizan hipótesis como que las baterías deben estar lo más lejos posible de los generadores y entre ellas, y lo más cerca posible de nudos importantes. Así, el problema se transforma en resolver un problema de maximizar y minimizar distancias entre los nudos de un grafo, que se podría resolver empleando computación cuántica. Por otra parte, podría emplearse una variación de un algoritmo de eigencentrality para localizar los nudos más extremos del sistema, que será donde es más probable que se coloque una batería. La detección de comunidades energéticas empleando computación cuántica también podría ayudar a reducir el tamaño del problema para posteriormente resolverlo o con computación cuántica o con algoritmos cuánticos dependiendo del tamaño de la red resultante. Esto permitiría trabajar con redes de gran tamaño.

Se tratará de resolver el problema empleando el enfoque cuantitativo, y se reservará el enfoque heurístico para el caso en que sea imposible encontrar una ventaja cuántica empleando el otro enfoque.

6. Conclusiones

Este proyecto se ha realizado en colaboración con i-DE y Multiverse Computing. De esta manera, forma parte de un proyecto mayor que culminará con la resolución del problema empleando computación cuántica.

Para la solución desarrollada para el proyecto Flexener, el tiempo total de computación parece crecer de forma cúbica. De esta manera, el problema puede ser resuelto fácilmente empleando computación clásica para la red que se estudia. Sin embargo, la computación cuántica y la consiguiente eliminación del algoritmo genético permitiría obtener mejores resultados y reducir el tiempo de computación, a la vez que trabajar con redes de mayor tamaño.

BATTERY NETWORK DESIGN USING QUANTUM COMPUTING

Author: Barragán Castro, Cristina.
Supervisor: Rodríguez Asensio, Miguel and Mateo Domingo, Carlos.
Collaborating Entity: i-DE

ABSTRACT

Starting from the code developed for project Flexener T.3.1.2, which uses genetic algorithms to design a battery network that maximises voltage control and reliability while minimising the needed investment, the possibility of translating it into quantum computing is analysed. After carrying out a scalability analysis, it is concluded that the optimisation is the most complex part of the algorithm, and, therefore, is where quantum computing could be most beneficial. The technology chosen to solve this problem is a Quantum Annealer, which works with a specific formulation: QUBO. Therefore, the problem needs to be reformulated. This is expected to allow for working with bigger networks and for a potential speed-up.

Keywords: Optimisation, Battery Network, Genetic Algorithm, Quantum Computing, Quantum Annealer, QUBO

1. Introduction

The increasing penetration of distributed generation in the distribution grid, particularly of renewable energies, and the growth in the total demand due to the electrification of the economy imply great challenges for grid operation and planning, The introduction of energy storage systems, and in particular batteries, into the power grid brings with it an improvement in flexibility, which is essential to balance generation and demand under uncertainty. This way, they allow for dealing with the variability of renewable generation, charging during peak generation and valley demand periods, and injecting power into the grid in opposite situations. This facilitates the integration of renewable energies into the grid. Moreover, batteries bring many other advantages, such as the possibility of acting as backup generators in case of fault due to their rapid response capabilities, which improves the system's reliability. This is especially important for radial systems, which have a low reconfiguration capacity and, therefore, find it harder to guarantee the power supply. For this project, the improvement in reliability and voltage control capabilities are the most relevant. Voltage control is related to batteries' ability to control their active and reactive power exchange with the system. On the other hand, the improvement in reliability is related to their ability to feed isolated systems.

All these capabilities, combined with the decrease in prices experienced in recent years, make these energy storage systems a very attractive option for investing in the development of the power grid. For this reason, it would be extremely useful to be able to quickly and efficiently design battery networks to be introduced into the distribution grid. Nevertheless, the great number of variables and possible locations for the batteries make this a very computationally complex problem. That is why this project analyses its implementation using quantum computing.

2. Project Flexener Description

This project follows project Flexener, more specifically task T.3.1.2, developed by Universidad Pontificia Comillas in close collaboration with i-DE. The objective of said task is to determine the optimum capacity and location of batteries in the grid in order to meet three objectives: maximising voltage control under normal operation, maximising reliability, and minimising investment. In this case, the optimisation problem is solved by using genetic algorithms. A radial 20kV network with distributed generation is used for the analysis. This way, a MATLAB algorithm is developed, which can be divided into three parts: input data pre-processing, the optimisation, and post-processing.

Starting with the pre-processing, this step could itself be divided into three main activities, the first being loading the data into the system and preparing it for its later use. This means, transforming it to inner format and rearranging it in order to avoid errors when using matpower and facilitate later calculations. Next, the network is divided into zones attending to the placement of tele-controlled switches. For each of these zones, several parameters are calculated as they will be needed later to carry out the optimisation. On another hand, in case of a fault in one of the zones, said zone will be isolated and the rest will reconfigure forming microgrids in order to guarantee the continuity of supply. The aforementioned microgrids are also identified in this step for every possible faulted zone. Lastly, as voltage control directly depends on the specific node in which batteries are placed, the node where the impact of placing the batteries will be greater regarding balancing voltages is identified.

As for the multiobjective optimisation, it is carried out at a zone level to facilitate the convergence of the genetic algorithm. A continuous optimisation variable which represents which battery capacity is installed in each zone is used. There are three objective functions: maximising voltage control during normal operation; maximising reliability which is modelled as the minimisation of the economic impact of the faults; and minimising the investment, which is modelled taking into account economies of scale.

Lastly, regarding post-processing, genetic algorithms are metaheuristic. This means that, when solving the optimisation, they do not guarantee optimality. They produce reasonable solutions instead. This way, the algorithm does not give a unique solution, but rather a number of them, which are presented in two ways: using a graph that represents the impact on all three metrics of increasing the total capacity installed, and a matrix that contains the values of the optimisation variable for each solution. This way, by looking at the tendencies in both graphs, the user can decide on the final design option.

3. Computation Time Analysis and Scalability

Computation time and scalability of the algorithm developed for project Flexener are analysed in order to determine which parts are the most computationally complex, and therefore, would experience greater benefits from the introduction of quantum computing.

With the computation time analysis carried out it is determined that, for a model that includes power flow calculations, 56% of the total computation time is used for the optimisation, while 43% is used for the pre-processing. On another hand, it can be seen that there are two functions which are more complex than the rest. These are the power flow and the genetic algorithm. As for the latter, a big part of the computation time is dedicated to internal functions. Among the different objective functions, the most complex one is the reliability. This is ascertained with the scalability analysis carried out at a zone level, whose results are presented in Figure 1. Due to a lack of data, it has not been possible to perform a scalability analysis at a bus level. Therefore, it was not possible to

determine the growth rate of the power flow. Nevertheless, it is estimated to be lower than the one for the optimisation.

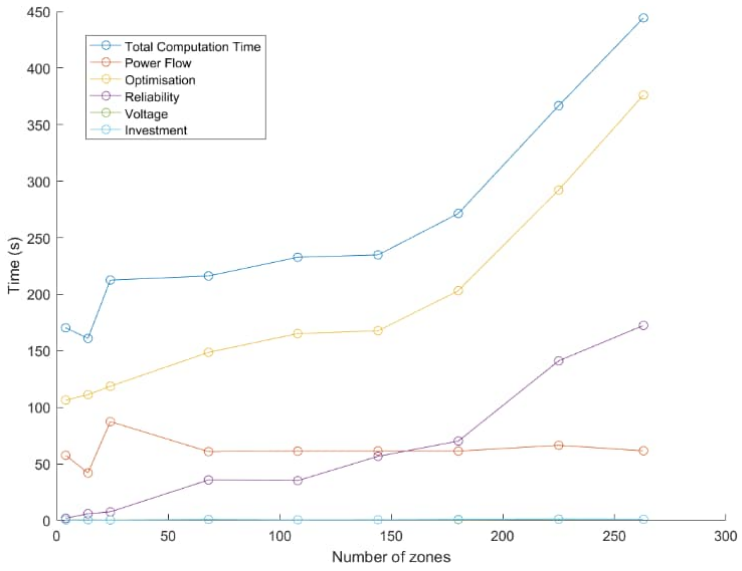


Figure 1: Scalability Analysis of the Algorithm developed for project Flexener at a Zone Level

It is concluded that the growth in the total computation time is directly related to the growth in the time needed for the optimisation and, in particular, for the reliability function. As the number of points used for the analysis is not very high and genetic algorithms are inherently chaotic, it is hard to approximate the growth rate to a particular function with complete certainty. Nevertheless, it is concluded it is at least cubic, as this is the growth rate of the reliability function. This way, there are two functions that can be interesting to translate into quantum computing: the power flow and the optimisation. However, current quantum technologies do not allow for the calculation of power flows for large networks but they show promising results for when the technology is more mature. This way, the project focuses on the optimisation.

4. Choice of Quantum Technology and Formulation

Several quantum technologies allow for solving optimisation problems.

Firstly, there are Quantum Gate Computers, which would use a QAOA algorithm to solve the problem. This option is discarded as this type of computer is very sensitive to noise, which may compromise the optimality of the solution. Moreover, they do not have a great number of qubits, so they cannot work with large networks.

Another option would be using a Quantum-Inspired solution, which consists of using a classical computer that simulates some properties of a quantum computer. The algorithm that would be used is QIOA, which is an analogue of QAOA for this kind of computer. However, this algorithm is not very mature yet and, furthermore, it requires a large number of resources. This way, it may have problems working with large networks.

For this reason, a Quantum Annealer is the chosen technology. This is a type of quantum computer which is specifically designed for solving sampling and optimisation problems. Moreover, it has a lower sensitivity to noise than Quantum Gate Computers, and a higher number of qubits, which allows for working with large networks. To use this kind of computer, it is necessary to use a specific formulation: QUBO (Quadratic Unconstrained Binary Optimisation). Therefore, the problem must be reformulated.

5. Quantum Approach

There are two ways in which the problem can be addressed using quantum technologies:

- Quantitative Approach: This approach consists of reformulating the original problem in order to adapt it to QUBO formulation. This way, the problem is first reformulated by using classical formulation for a continuous variable but getting rid of genetic algorithms and unifying the three objective functions into one. Next, the problem is discretised. In this sense, the optimisation variable now represents whether a certain battery capacity previously specified is installed in a zone or not. Lastly, the restrictions are let go. This way, we are left with a problem which meets the QUBO requirements but is expressed in classical formulation. This must later be adapted in order to have the typical QUBO format, by calculating matrix Q.
- Heuristic Approach: This means solving the problem without computing power flows, reducing the number of calculations and, essentially, using common sense. This way, several hypotheses are made, such as that the batteries should be placed as far as possible from the generators, as far as possible from each other, and as near as possible to important nodes. Thus, the problem is transformed into a matter of maximising and minimising distances between nodes in a graph. This could be solved using quantum computing. On another note, a variation of an eigencentrality algorithm could be used to identify pendant nodes, which are expected to be some of the most interesting for placing the batteries. Moreover, energy community detection using quantum computing could also help reduce the size of the problem, to later be solved using either classical or quantum computing depending on the size of the resulting problem. This would allow for working with larger networks.

It will be attempted to solve the problem using the quantitative approach first and the heuristic approach will be left as a backup in case it is not possible to find a quantum advantage using the quantitative approach.

6. Conclusions

This project has been developed in collaboration with i-DE and Multiverse Computing. This way, it belongs to a larger project that will finish with the implementation of the problem into quantum computing.

In the solution developed for project Flexener, the total computation time increases following a cubic function. This way, the problem can be easily solved using classical computing for the analysed network. Nevertheless, quantum computing and the consequent elimination of the genetic algorithm will allow for obtaining better results and reducing the total computation time, while working with larger networks.

Index

List of Figures

List of Tables

List of Equations

Abbreviations

SMES – Superconductor Magnetic Energy Storage
DER – Distributed Energy Resource
BESS – Battery Energy Storage System
DSO – Distribution System Operator
ENS – Energy Not Supplied
VOLL – Value of Lost Load
OPF – Optimal Power Flow
IPSO – Improved Particle Swarm Optimisation
IWPSO – Improve Weight Particle Swarm Optimisation
ABC – Artificial Bee Colony
MINLP – Mixed Integer Non-Linear Programming
ELF – Equivalent Lateral Forces
GA – Genetic Algorithm
QAOA – Quantum Approximate Optimisation Algorithm
QIOA – Quantum Inspired Optimisation Algorithm
HHL – Harrow-Hassidim-Lloyd Algorithm
WLS – Weighted Least Square Algorithm

1

Introduction

The electrical grid needs to be in permanent evolution to keep up with changes in generation and demand patterns. In this sense, the increase in variability due, among others, to the development of renewable energies and the rise in demand because of electrification, implies a great challenge for electricity networks. They bring with them several technical problems [1] such as under and over-voltages, reaching thermal limits of power lines, reverse power flows, phase imbalances, large short-circuit currents, and an increment of harmonic components. In recent years, information technologies and advanced power electronics have provided grid agents with some necessary tools to improve grid monitoring and control and, assure power quality. Nevertheless, it is essential to keep working towards improving grid flexibility as it is becoming increasingly valuable in this context of uncertainty.

Grid flexibility refers to the capability of a power system to match generation and demand under uncertainty by modifying their generation and consumption patterns to ensure a balanced system. This is especially important now that renewable energies are being introduced into the system, as their generation is intermittent and variable. This way, several flexibility mechanisms can be used to increase flexibility in different parts of the grid (generation, transmission, or distribution), each requiring a different level of investment depending on the resources needed. These include, among others, from lowest to highest investment[2]:

- Improved operations: Like shortening dispatch schedules, improving weather forecasting, consolidating balancing areas…
- Demand Response: Consists of shifting electricity demand to help balance the grid, using more energy during generation surplus periods and less during peak hours and scarcity periods. This may include network tariffs and connection agreements.
- Changes in Grid Infrastructure: Like increasing transmission capacity
- Fast Ramping Supply: Including flexible generation, which is designed to be turned up and down regularly and, therefore, can respond rapidly to changes in demand.
- Energy Storage: Using batteries, pumped hydro, compressed air, SMES…

This project focuses on the latter: Battery Energy Storage Systems or BESS. This technology is especially useful for systems with renewable generation. Both renewable energies and BESS can be classified as distributed energy resources (DERs). Due to the falling costs of renewables, especially PV panels, and different favourable environmental policies, the penetration of renewable energies is rapidly increasing. Moreover, since recently, European regulation dictates DSOs must take into account both traditional passive network elements and DERs when planning to guarantee the quality and continuity of supply [26].

BESSs are connected to the distribution network and allow for dealing with the variability in renewable energy production by charging the batteries with excess energy during periods of low demand and injecting power into the system during peak periods. This way, they allow for a robust and resilient integration of energy resources without continuous generation [9], and they are a suitable solution for substituting power curtailment [4]. On the other hand, they can also reduce peak power demand on a specific feeder transformer, which can extend its useful life. In this sense, they can also help avoid unnecessary investments through deferring grid reinforcement, which is done by reducing the power through some of the lines, especially in LV [8]. Moreover, BESSs can also act as a backup in case of a contingency or a sudden increase in demand due to their rapid response [3]. This is extremely important in distribution grids due to their radial nature, as a contingency can imply the loss of supply for every customer connected downstream. Additionally, with a sufficient installed capacity, energy storage systems could allow operating power plants at their optimal efficiency, and cover peaks in the demand with the BESS capacity [5].

For this project, some of BESSs' most important capabilities are voltage control and reliability improvement, which are done by controlling their active and reactive power exchange with the network. Reliability is related to the ability to provide a sustainable power supply to customers. This way, many elements such as switching devices and maintenance crews are important for improving reliability, by minimising the impact of grid faults. When possible, reliability tends to be improved by introducing new lines into the system that allow for reconfiguration. Nevertheless, this may be hard in some areas, especially in rural networks. Therefore, in these cases, reliability can be improved with DERs, where generation is connected near the customers. This way distributed generation would allow for the formation of microgrids, feeding areas that become isolated because of a fault located upstream. Reliability is often optimised by taking into account high-probability and low-impact events that occur in a specific location of the network [14].

On the other hand, BESSs are implemented into the grid using grid-forming or grid-supporting inverters, allowing them to effectively control both frequency and voltage [7]. This is thanks to the ability of batteries to rapidly deliver large amounts of both active and reactive power [9]. This way, BESSs positively affect the quality of supply, being able to also reduce the harmonic content. They may also help with power factor correction. Voltage control is extremely important for the protection of assets, as they are all designed to work within a certain voltage range.

There are plenty of different types of batteries: Li-ion, Na-ion, Zn-based batteries, and lead-acid, among others. The following figure shows the types of batteries that are currently installed in the distribution network (a) and the ones that are planned for the near future (b):

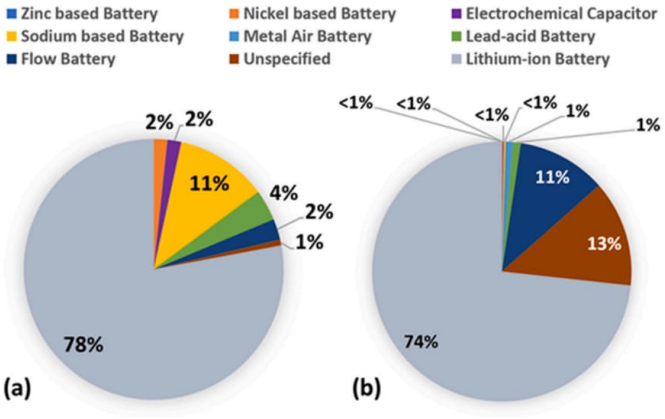


Figure 1: Types of BESSs that are in operation nowadays (a) and planned (b) [9]

As can be seen, currently the most used type of BESS is Li-ion batteries. These are also the type of batteries used in this project, as they are the ones used by i-DE in their distribution grid. It is important to note that, regarding energy storage, pumped hydro is still the most used technology. Nevertheless, Li-ion batteries can provide a wider range of grid services due to the existing variety regarding rated power and duration [9]. Moreover, one of the main benefits of batteries against hydro power is their flexibility of allocation. Hydropower is much more restrictive in terms of where new power plants can be installed.

In recent years, the price of Li-ion batteries has drastically decreased, as can be seen in the following graph.

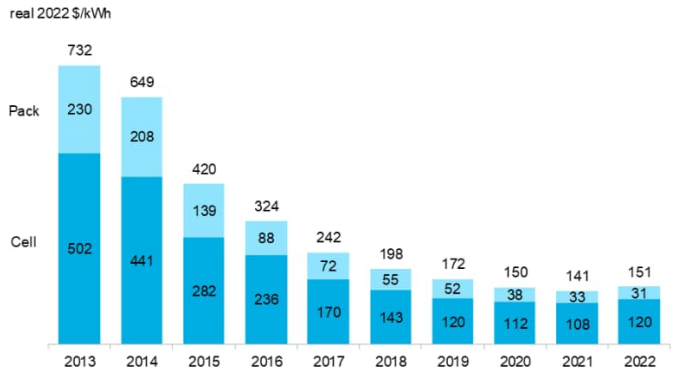


Figure 2: Volume-weighted average Lithium-Ion battery (pack and cell) price evolution in $ [12]

In 2022, cells, which represent 83% of the total battery price [12], have slightly increased their cost due to the price of materials and manufacturing costs, which are directly related to the price of electricity. Nevertheless, improvements in battery packs have helped mitigate the increase in cell price. This will also happen in the coming years, where pack costs are expected to keep dropping by 20% by the year 2025, while cell costs will only increase by 10% relative to the year 2021[13].

Due to all the previously described capabilities and the performance improvement, joined with the decrease in prices experienced during the last decade and expected for the coming years, BESSs are becoming an increasingly attractive investment option. Energy storage, particularly batteries, will play an important part in the development of the energy grid. Therefore, being able to quickly and

effectively design battery networks would be extremely useful. Nevertheless, the enormous number of variables and possible locations make this a very computationally complex problem.

1.1 Objectives

The main aim of this project is to study the suitability of quantum computing for developing an algorithm to maximize voltage control and grid reliability while minimising the needed investment through the design of a battery network. For this purpose, the project will study the viability and benefits of transferring a pre-existing MATLAB code from project FLEXENER into quantum computing. This way, it will determine whether this new computing paradigm could be useful to solve this kind of optimisation problem, getting rid of computational limitations and allowing for the development of more complex models that can produce useful results. This can be broken down into several objectives:

- Getting to know the background of project FLEXENER and the functioning of the developed code.
- Getting to know genetic algorithms and quantum computing
- Identification of bottlenecks and computational challenges in the code
- Preparing the formulation for quantum computing
- Developing a strategy for the implementation of quantum computing for the set problem

1.2 Methodology

This project has been developed in collaboration with the Spanish DSO i-DE, part of the Iberdrola group, and Multiverse Computing. For its completion the next methodology is used, which follows the logical flow of the objectives previously described:

1. Background Research: To carry out this project it is important to know the distribution network's needs and the impact batteries may have on it. Therefore, different papers relating placing and sizing of DERs are analysed. Moreover, as previously explained, this project comes from T.3.1.2 of project FLEXENER. Therefore, getting to know its background, development, and results is also essential. Finally, as genetic algorithms and quantum computing are key parts of the project, research will be carried out on both topics.
2. Code Analysis: Reverse engineering is done to understand the code developed for project FLEXENER, which is the project's starting point. It is important to get to know the data used as well as the functioning of the code itself.
3. Identification of Weak Points and Computational Challenges: A computation time analysis is carried out for different scenarios. This is a key step to analyse the scalability of the algorithm and identify where quantum computing could be more useful. Different hypotheses are analysed to modify the complexity of the problem as well.

4. Exploring different Classical and Quantum Methodologies for solving these kinds of problems
5. Development of a Strategy for the Implementation of the Problem in Quantum Computing: The most suitable technology and approach for the problem are chosen, and a description of the way this would be applied in the specific case is presented.
6. Problem Reformulation, to adapt the problem for quantum computing and include new hypotheses and changes into the system. The scalability of the new model is also analysed.

1.3 Structure of the Document

This report is organised into different chapters, whose content is explained next.

The project follows Task T.3.1.2 of project FLEXENER, which is explained in Chapter 2, titled "Project FLEXENER Task T.3.1.2 Description". In said project, an algorithm for designing a battery network with the same network objectives was developed using classical computing. Chapter 2 contains relevant information regarding the algorithm developed as part of this task. The information presented was obtained by carrying out reverse engineering on the code., as well as from the different reports generated during the development of the task.

Next is Chapter 3: "Genetic Algorithms and Other Approaches". In project FLEXENER the problem was solved by using genetic algorithms. For this reason, this chapter presents some background research regarding this kind of algorithm and other approaches that could have been taken to solve the problem.

The following chapter presents how the genetic algorithm was implemented in Task T.3.1.2 as well as the results obtained. This is Chapter 4: "Project FLEXENER Task T.3.1.2. Results". It also presents some of the simplifications made to be able to solve the problem with classical computing.

To finish with the state of the art and background research, Chapter 5, titled "Quantum Computing", contains some basic concepts regarding Quantum Computing, as well as current uses of this technology in power systems. Moreover, the different technologies and methodologies for solving optimisation problems available are analysed to allow for choosing the most suitable option for the project.

The next sections of the report are related to code analysis and its preparation for quantum computing. Starting with Chapter 6 "Computation Time Analysis", it analyses the needed time for execution for each part of the code and the impact of the network size on it. Moreover, the most computationally complex parts of the problem are identified, which leads to determining the parts of the code that should be translated into quantum computing. Once identified, Chapter 7, which receives the name "Choice of Quantum Technology and Formulation" selects the most suitable quantum computing methodology for solving the problem and briefly explains the formulation needed for translating it into quantum computing. Then, Chapter 8, titled "Problem Reformulation", prepares the problem for quantum computing. With this aim, the MATLAB algorithm from project FLEXENER is first reformulated using an operational-research-like formulation. This is then transformed into QUBO formulation, which is more appropriate for quantum computing using quantum annealers. For the discrete problem, the algorithm's scalability is analysed, as it resembles more what will be translated into quantum computing. Next, Chapter 9, named "Quantum Approach", contains different quantum methodologies and approaches that could be used to solve the problem.

Lastly, Chapter 10 collects all the conclusions drawn from the project. This is followed by Chapter 11 "Future Work", which contains different lines of work that could be followed to continue the project, as the algorithm has not been implemented in quantum computing yet. Moreover, other considerations that could be taken into account to further improve the existing model and make it more realistic and efficient are also discussed.

2

Project FLEXENER. Task T.3.1.2 Description

This project follows task 3.1.2 of project FLEXENER, developed by i-DE in collaboration with Universidad Pontificia Comillas. Project FLEXENER aims to accelerate the consecution of the energy transition goals. For this, it promotes research into new technologies regarding renewable energies and their system integration, demand management, and energy storage systems. Moreover, it brings to the fore the necessity to increase investments in modern power systems to transition into a cleaner more sustainable model [11].

Specifically, Task 3.1.2 focuses on energy storage systems. It presents a model that allows for determining the optimum size and placement of batteries into a network, maximising reliability and voltage control while minimising the needed investment. In case of fault, the batteries will improve the system's reliability by feeding areas disconnected from the main feeder. On the other hand, when there is no fault, the storage system will help with voltage regulation.

Task 3.1.2 analyses three different networks. However, only the implementation and results obtained for one of them will be analysed in this section: Network 1. The implementation for all of them is practically the same, with only slight modifications regarding specific parameters like the presence of capacitor banks and the maximum battery capacity. Regarding the latter, it is important to set a maximum total capacity for the batteries as the voltage control and reliability are expected to increase with higher capacities. Nevertheless, there is a maximum investment the company is willing to assume, which is modelled as the maximum installed capacity.

In this case, Network 1, corresponds to a 20kV rural network from the east of Spain, which is mainly radial and has both overhead and underground lines. The system counts with renewable generation integrated into the system as DERs, consisting of solar and run-of-the-river hydropower. The selection of a rural network in this case is extremely important, as using an urban area the possible locations of the batteries would be extremely limited by the available space in each zone. Moreover, for this problem, radial networks are more interesting, as they have fewer reconfiguration options and, therefore, lower reliability than meshed networks.

The algorithm developed in T.3.1.2 follows the next basic structure, consisting of four main steps:

Figure 3: Outline of the FLEXENER T.3.1.2 Model

Starting with the input data, the following table includes all external data provided by i-DE and introduced into the model:

Table 1. Project FLEXENER T.3.1.2 - Input Data

Name	Unit	Description
branch_length.mat	km	Branch length
interrupt.mat	-	Array containing the branches in which switches are placed. Branches are specified by the buses at each end.
loop.mat	bool	Array containing which buses are connected to other substations that could provide the necessary power in case of a fault (Mesh)
p_head.mat	MW	Hourly header active power generation profile
p_hydro.mat	kW	Run-of-the-river hourly power production in a year
percentage_oh.mat	pu	Percentage of each branch that is overhead
pot_generadores.mat	kW	Total hourly nominal generation in each bus. It includes both solar power and run-of-the-river hydropower.
potencias_oficiales.mat	kW	Installed active power demand in each bus.
PSSE_V33_4_MT_20210430_03_Estudio bateria CASO 1_NONAMES_reducido.raw	-	PSS-E file of the system without the slack
PSSE_V33_4_MT_20210430_03_Estudio bateria CASO 1_NONAMES_reducido_Complete.raw	-	PSS-E file of the system including the slack
q_head.mat	MW	Hourly header reactive power generation profile
sun.mat	pu	Hourly capacity factor of solar energy production
voll.mat	€/h	Value of Lost Load. Represents the economic impact per hour of the loss of supply in each bus
volt.mat	pu	Hourly voltage in a year and for each bus. It is the result of running function pf_compute_pf_iber.m, which runs the power flow and saves the results into this variable. Therefore, once the function is run and the variable created, it can be used in main.m without calling the function.

Data pre-processing is carried out to make this data suitable for optimisation and reduce computation time. This is done by calculating constant parameters beforehand, so they can simply be used during the optimisation process. There are three main activities:

- Loading and preparing data: All the previously described data is loaded into the system, and additional information such as the fault rate and reparation time is also introduced. A certain portion of the data is chosen as the object of study. In particular, the second-largest island, which has a total of 263 buses and can be seen in Figure 4. The information corresponding to this data set is then transformed into inner format and rearranged to facilitate future calculations.

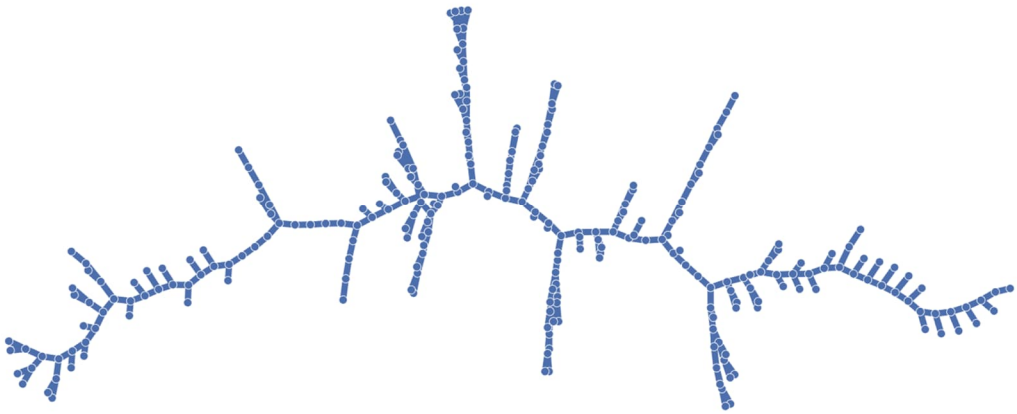


Figure 4: FLEXENER Network 1 Second Largest Island Configuration

- Separating the network into zones and microgrids and calculating their fundamental parameters: The network has many switches, which are the elements that will allow for a certain area of the network to be isolated in case of fault. There is a total of 263 buses in the network, that can be grouped into zones attending at the remote-controlled switch placement. Therefore, there are as many zones as remote-controlled switches plus 1, so 24 zones in total. For each of these zones, a series of fundamental parameters are calculated during data-pre-processing: total demand, total VOLL, the total length of overhead and underground lines, total ENS, and total capacity available in case of fault, among others. On the other hand, as previously explained, in case an event occurs, the faulted zone is isolated by opening the corresponding switches. The rest of the network is reconfigured to reduce the impact and guarantee the quality of supply. This way, different zones are grouped in microgrids, which are also identified during this step.

- Locating the optimal node for placing the batteries: While the reliability of the system is independent of where the batteries are located, voltage control does depend on the specific bus on which they are placed. For this purpose, it is necessary to identify the nodes that have the highest voltage deviations, for which running a power flow is essential. It is also needed to calculate the sensitivity matrix which relates variations in power injections with variations in voltage. It is important to note that deviations in voltage below 3% of the nominal value are considered normal. With this information, the optimal node in each zone is identified, seeing where the impact of placing the batteries will be greater.

Regarding optimisation, this is a multi-objective optimisation problem with 3 objective functions:

- Maximisation of Voltage Control: Voltage variations directly depend on the node in which the batteries are placed. Taking this into consideration, a metric that relates voltage variation in the optimal bus and battery capacity was created:

$$V_{imp-Zona} = abs(sum\left(M_{opt}(:,n)\right) * C_z$$

$$V_{imp-Total} = \sum_{zonas} V_{imp-Zona}$$

Equation 1: Voltage Control Metric [10]

Where C_z represents the battery capacity in kWh, M_{opt} is the variation in voltage due to a modification in the active power injection, and n is the optimal node.

Batteries will be used for this purpose when there is not a fault and voltage limits are not satisfied.

- Maximisation of Reliability: This is modelled as the minimisation of the VOLL. It does not depend on the node in which the batteries are located, but it depends on how much a fault in a set zone affects all the other zones. The equation used is the following:

$$Metric\_Reliab = \sum_{b1=1}^{b1=B} \sum_{b2=1}^{b2=B} fr_{b2} * h\,rep_{b2} * IE_{b1} * ZI_{b1,b2}^{ns}$$

Equation 2: Reliability Metric [10]

Where fr_{b2} represents the failure rate of zone b2, IE_{b1} is the economic impact in b1 of the fault, $hrep_{b2}$ is the repairing time needed for zone b2 after a fault, and ZI_{b1b} is a matrix that indicates how much a fault in b2 affects zone b1. This matrix is calculated using the following expression:

$$ZI_{b1,b2} = 1 - \frac{C + \sum PGmed_Z * hrep}{\sum PDmed_Z * hrep} = 1 - \frac{C}{\sum ENS_Z}$$

Equation 3: Interaction Matrix [10]

Where C is the battery capacity installed in the microgrid b1 belongs to in kWh, hrep is the average repairing time after a fault, $PGmed_z$ is the average generated power in the zones in the microgrid b1 belongs to and $PDmed_z$ is the average demand in the microgrid b1 belongs to (so Z is the zones that belong to the same microgrid as b1 for a fault in b2). As can be seen, the value of ZI depends on the interaction between zones and therefore is directly related to the complexity of the network. Moreover, its value depends on the installed battery capacity and, consequently, needs to be recalculated for every iteration. If the available generation is greater than the demand, the value will be limited to 0. It is important to note the value of ZI will be the same for all the zones that form a microgrid as, being connected after the fault, the power generated in one of them can also be transmitted to the others. Nevertheless, it is important to remember that microgrids change depending on the location of the fault and, therefore, even though two zones may share the same value of ZI for a fault in a certain zone, it does not mean they will always coincide.

- Minimisation of Investments: For evaluating the cost of the batteries, two scenarios are contemplated: one using economies of scale and the other not using them. For the first, the cost of the batteries is represented by a quadratic function, while for the second it is represented by a straight line. This can be seen in the following graph:

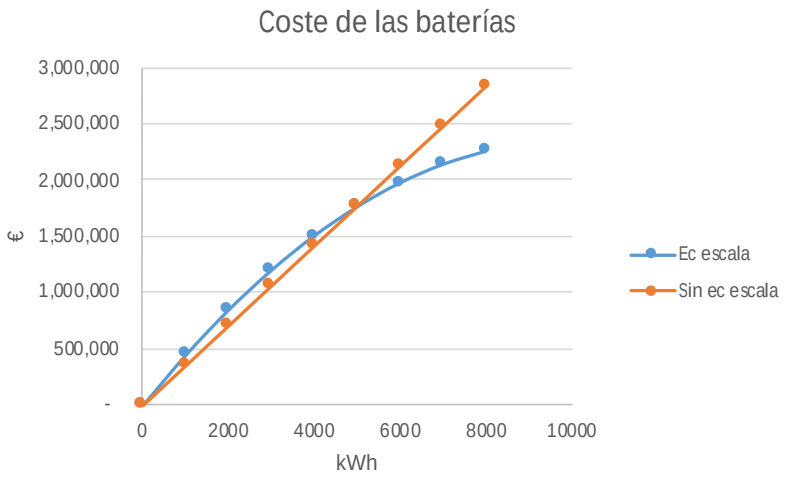


Figure 5: Project FLEXENER T.3.1.2: Cost of Batteries [10]

It is recommended to include economies of scale in the analysis when possible as the results obtained are more accurate. Nevertheless, among big batteries (of above 100kW), economies of scale do not have a significant effect on the price.

Considering this, it is decided to approximate the cost using a quadratic function. Values provided by i-DE regarding the power, capacity and unitary cost of batteries are used.

For calculating the function, which is of the form $f(x)=ax^2+bx+c$, the following steps are followed:

- c: the function starts in the origin, therefore c=0.
- b: The maximum of the quadratic function will coincide with the maximum capacity for the batteries, therefore, deriving the function at that point and knowing the slope of the tangent must be zero at that point: $b = -2 \cdot emax \cdot a$, where emax corresponds to the maximum total capacity for the batteries.
- a: Substituting the previously calculated values in the quadratic function and knowing both functions must cross then:

$$\frac{cost}{cap} \cdot x = a \cdot x^2 + b \cdot x$$

$$\frac{cost}{cap} = a \cdot x - 2 \cdot a \cdot emax$$

$$a = \frac{unitary\ cost}{cap - 2 \cdot emax}$$

Equation 4: Parameters for the Investment Equation

As can be seen, the equation depends on the unitary cost of the batteries. This value is calculated using data provided by i-DE regarding the batteries already installed in their networks. It is concluded the unitary cost of the batteries is 353 €/kWh.

A maximum size for the total installed capacity is set to carry out the optimisation. Once the optimisation is finished, an analysis of the results is needed to determine which is the most suitable solution for the project. This step will be explained in greater depth in Chapter 4.

The following flow diagram shows a more detailed explanation of the sequential model, including all the functions and input data used:

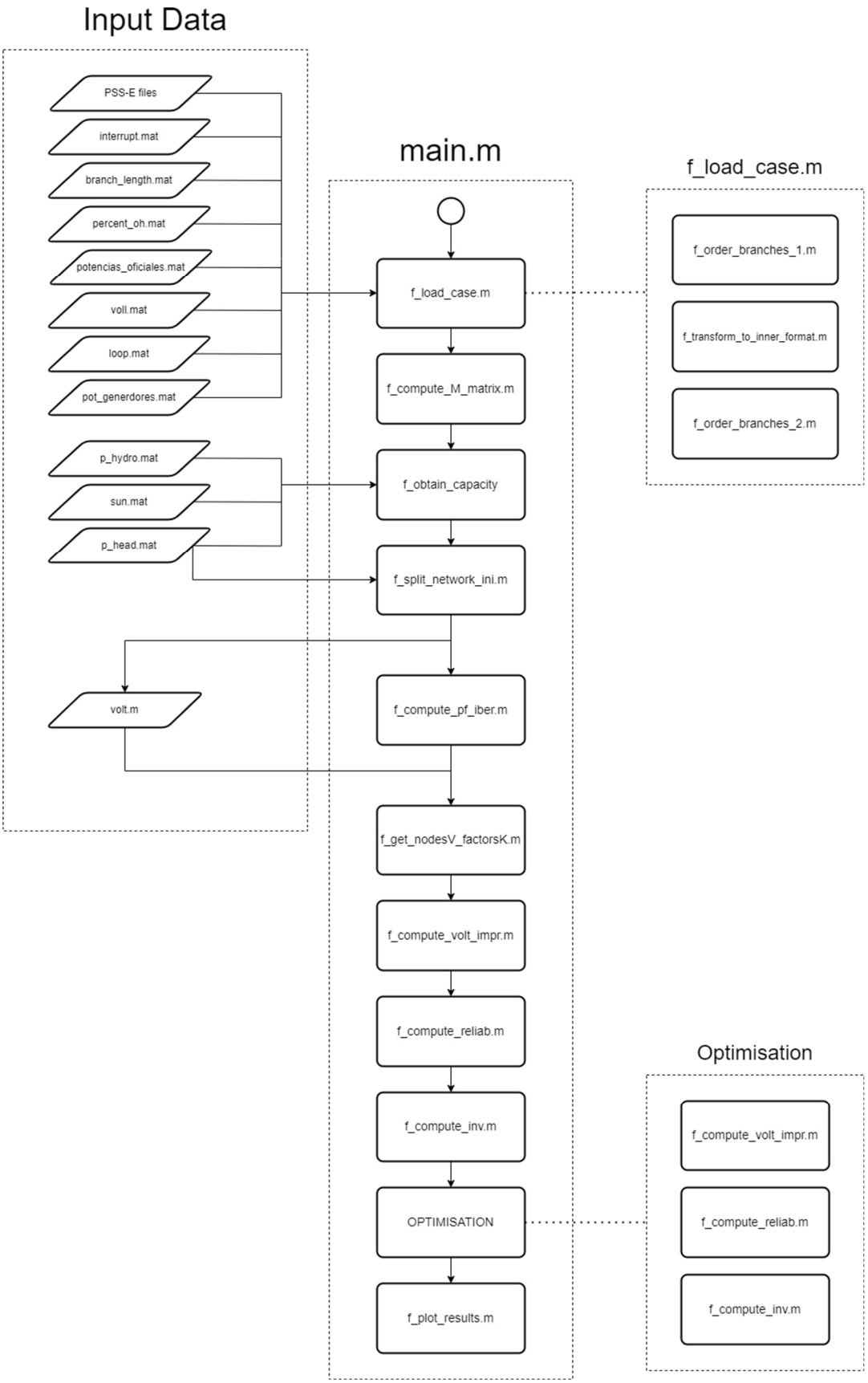


Figure 6: Project FLEXENER T.3.1.2. - Flow Diagram

To better understand the algorithm, the different functions used are presented next, including a brief description of each of them, as well as their inputs and outputs, loaded data and what other functions they call:

Table 2. Project FLEXENER T.3.1.2 - Function Description

	main.m	f_load_case	f_order_branches_1	f_transform_to_inner_format
Description		Loads input data and makes the necessary changes for enabling its processing and use for the optimisation problem. These changes include transforming it to a matpower format and rearranging it based on the direction of the power flow.	Arranges the branches so that each branch's first bus is located upstream. The same is done with the switches.	Transforms the data into matpower format.
Input	-	-	- mpc_v0: Structure. Includes all network data - interrupt: Array. Includes the buses between which each switch is located.	- mpc_formato_ext: Structure. Includes all network data with the branches rearranged. - branches_formato_ext: Array. Includes the branches in which switches are located expressed as the buses located at the line's beginning and end. - buses_formato_ext: Array. Includes the downstream buses of the branches where switches are located.
Output	-	- mpc: Structure. Includes all network data - interrupt_ordenado_int: Bool. For each branch indicates whether there is a switch on it or not. It is expressed using matpower format.	- mpc_v1: Structure. Includes all network data with the branches rearranged. - interrupt_ordenado: Array. Includes the buses between which each switch is located rearranged so that the first bus is the one located upstream	- mpc_formato_int: Structure. Includes all network data with the branches rearranged and in matpower format. - branches_formato_int: Array. Includes the branches in which switches are located expressed as the buses located at the beginning and the end of the line in matpower format.

				- buses_formato_int: Array. Includes the downstream buses of the branches where switches are located in matpower format.
Loaded Files	volt.mat (This can be used instead of f_compute_pf_iber.m)	• caso_iberdrola.m • PSS-E files • interrupt.mat • branch_length.mat • percent_oh.mat • potencias_oficiales.mat • voll.mat • pot_generadores.mat • loop.mat	-	-
Called Functions	• f_load_case.m • f_compute_M_matrix.m • f_obtain_capacity.m • f_split_network_ini.m • f_compute_pf_iber.m • f_get_nodesV_factorsK.m • f_compute_volt_impr.m • f_compute_reliab.m • f_compute_inv.m • f_plot_results	• f_order_branches_1.m • f_transform_to_inner_format.m • f_order_branches_2.m	-	-

	f_order_branches_2	f_compute_M_matrix	f_obtain_capacity	f_split_network_ini
Description	Arranges the branches so that each branch's first bus is located upstream. The same is done with the switches. It needs to be done again because they may have jumbled when transforming the data into matpower format.	Calculates the sensitivity matrix.	Calculates the total generation capacity available to support the continuity of supply in case of fault. The available generators are solar and run-of-the-river hydro.	Identifies zones and their important parameters. Then identifies microgrids, which are formed by a group of zones, and obtains their parameters.
Input	- mpc_v0: Structure. Includes all network data in matpower format. - interrupt: Array. Includes the branches in which switches are located expressed as the buses located at the beginning and the end of the line in matpower format	- mpc: Structure. Includes all network data in matpower format and rearranged.	- mpc_in: Structure. Includes all network data.	- mpc: Structure. Includes all network data. - interrupt_ordenado_int: Bool. For each branch, indicates whether there is a switch on it or not. It is expressed in Matpower format. - M_opt: Sensitivity matrix which represents the relation between active power injections and voltage variations.
Output	- mpc_v1: Structure. Includes all network data rearranged and in matpower format - interrupt_ordenado: Array. Includes the branches in which switches are located expressed as the buses located at the beginning and the end of the line rearranged so that the first bus is the one located upstream and in matpower format	- M_opt: Array. Sensitivity matrix which represents the relation between active power injections and voltage variations. It will have as many rows as buses-1, as it does not include the slack bus.	- mpc_out: Structure. Includes all network data. Has an extra field in the structure including the available capacity.	- mpc: Structure. Includes all network data. Has six new fields: number of zones, mgs_partition (indicates how all zones are reorganised in case of a fault in one of them), mgs_affect (indicates if a zone is affected by a fault in another zone and how it reacts), the distance Matrix, the zonal table (which includes important parameters for every zone), and the Probability Matrix.
Loaded Files	-	-	• p_hydro.mat • sun.mat	• p_head.mat • p_hydro.mat • sun.mat
Called Functions	-	-	-	-

	f_compute_pf_iber	f_get_nodesV_factorsK	f_compute_volt_impr.m
Description	Calculates hourly power flow	Determines optimal buses for placing the batteries by calculating the improvement in voltage control when placing the battery in each node.	One of the objective functions. Calculates the voltage improvement when installing a certain battery capacity.
Input	- mpc: Structure. Includes all network data.	- mpc: Structure. Includes all network data. - volt: Array. Hourly voltage in a year and for each bus. - M_opt: Sensitivity matrix which represents the relation between active power injections and voltage variations.	- X: Installed battery capacity. Optimisation variable.
Output	- volt: Array. Hourly voltage in a year and for each bus.	- mpc: Structure. Includes all network data. Has some extra information in the zonal table field, regarding the optimal bus in each zone and its voltage KPI.	- metric_volt: Metric representing the total voltage improvement obtained by installing X battery capacity
Loaded Files	• p_head.mat • q_head.mat • p_hydro.mat • sun.mat	-	-
Called Functions	-	-	-

	f_compute_reliab	f_compute_inv	f_plot_results
Description	One of the objective functions. Calculates the reliability of the network expressed as the VOLL.	One of the objective functions. Calculates the needed investment for installing a set battery capacity.	Represents in a 3D graph the voltage and reliability improvement with respect to the investment.
Input	- X: Installed battery capacity. Optimisation variable.	- X: Installed battery capacity. Optimisation variable.	- Fval: Results obtained from the optimisation using gamultiobj. - X: Installed battery capacity. - metric_volt: Metric representing the total voltage improvement obtained by installing X battery capacity - metric_reliab: Metric representing the total VOLL obtained by installing X battery capacity
Output	- metric_ reliab: Metric representing the total VOLL obtained by installing X battery capacity	- metric_ inv: Metric representing the investment needed for installing X battery capacity.	-
Loaded Files	-	-	-
Called Functions	-	-	-

As previously explained, in project FLEXENER the multi-objective optimisation problem is solved by using genetic algorithms, which are described in greater depth in the next chapter, as well as other traditional ways to approach this problem.

3

Genetic Algorithms and Other Approaches

Due to the non-linearity and combinatorial nature of the problem, for large-scale networks, the proposed multi-objective optimisation problem is impossible to solve by exhaustive search using classical computers. Therefore, different approaches can be used to tackle the problem like Optimal Power Flow techniques, analytical techniques, and other artificial intelligence techniques among others.

3.1 Genetic Algorithms

As stated in previous chapters, project FLEXENER uses genetic algorithms, a metaheuristic algorithm, to solve a multi-objective optimisation problem.

Metaheuristic algorithms try to find a quasi-optimal solution to complex problems in a reasonable amount of time. They are a set of generic algorithms that can be used to solve almost any optimisation problem. The success of metaheuristics relies on the integration of exploration and exploitation. This means these algorithms search for feasible solutions both close to the best solutions and in new areas [15]. There are several types of metaheuristic algorithms [16]: swarm-based algorithms, like ant colony optimisation; physics-based algorithms, like simulated annealing; human-based algorithms, like tabu search; and evolutionary algorithms, like genetic algorithms.

Genetic algorithms are global adaptative search methods based on three main principles of natural selection and genetics:

- Selection: The most successful individuals are kept for following generations, while the less successful ones are discarded [18]. However, these individuals can suffer random changes due to mutation or crossover, among others.
- Mutation: Random gen modifications, they can be either flips, swaps, scrambles or inversions [19].
- Crossover: Substring exchange between individuals [18].

A set of possible solutions represents a population, and each of the solutions corresponds to a chromosome. Likewise, chromosomes are divided into genes, representing each parameter that forms the possible solution [17]. The chromosomes are evaluated using one or several fitness functions. Some methods rate the success of the fitness population of solutions, while others only rate a random sample to reduce the needed time. Depending on the results obtained, chromosomes for the next generation are biasedly selected, being more likely to choose those with higher fitness values [18]. These chromosomes randomly mate, and after random mutations and crossovers, a new generation of feasible solutions is created. This new generation will have higher average fitness values and the same population size as the previous generations [17].

Due to their high versatility, they are the most used type of algorithm to solve this kind of problem in the distribution network [10]. In the case of project FLEXENER, the algorithm searches for the best solution by looking at the performance of previous feasible solutions regarding all three metrics (voltage, reliability, and investment) and developing new solutions from the best ones.

3.2 Other Approaches

As previously mentioned, genetic algorithms are the most used method to solve optimisation problems in the distribution grid. Nevertheless, plenty of methods for optimal sizing and placement of batteries and other DERs into the distribution grid can be found in the literature.

The different techniques used to solve this kind of problem can be classified into four groups [21]:

- Analytical Techniques: When the system is small and not very complex mathematical models can be used, providing an accurate solution to the problem. Nevertheless, this is not computationally efficient for complex networks. Moreover, their performance lowers in problems with multiple objectives [45]. The techniques used may include Eigen-Value-Based Analysis, Point Estimation Method and Sensitivity Based Method, among others.
- Classical Optimisation Techniques: This includes Linear Programming, MINLP and OPF, among others. The choice of the method depends on whether the objective function is linear or not, whether the variables are continuous or discrete and the number of objective functions.
- Metaheuristic Techniques: They are often inspired by nature and are able to reduce the search space after every iteration. This way, they can efficiently obtain good solutions for a wide variety of problems in a reasonable time, but they do not guarantee optimality [45]. Some examples are PSO, fuzzy logic, artificial neural networks and ABC, among others.
- Other Techniques: These include other techniques such as Monte Carlo Simulation or Backtracking Search Optimisation Algorithms (BSOA), but also some promising techniques that may be useful for solving this kind of problem in the future. Moreover, in cases where the historical data available is limited and there are a lot of uncertainties, probabilistic optimisation methods can be used [45]. However, these may become very computationally complex with many scenarios.

Some specific examples are presented next, most of them belonging to the groups of optimisation and metaheuristic techniques as they are more suitable for medium and large-sized networks:

- SCOP Optimal Power Flow Algorithm: In [20] this kind of algorithm is used to integrate sizing, placement and other operational strategies. Using an OPF allows for taking into account grid constraints. In this case, an AC multi-temporal OPF with a convex relaxation of power flow equations is used. This assures obtaining an optimal solution.

- IPSO-Monte Carlo: For allocating and sizing DERs in a system with 33 buses, [22] presents a method that mixes the IPSO algorithm with Monte Carlo simulations. The proposed multi-objective optimisation problem minimises the cost associated with power losses, maximises voltage control and reliability and minimises the costs related to energy purchased from the market. Monte Carlo simulation is used for reliability. The results attained are better than the ones obtained using IWPSO and ABC algorithms.

- MINLP Techniques: For a 33-bus and a 69-bus system [23] proposes a two-tier model, using a Siting Planning Model (SPM) and a Capacity Planning Model (CPM). First, SPM is used to identify the set of best possible placements for the DERs. This allows for reducing the search space and, therefore, helps to lower the total computation time. Then, CPM is used to determine the optimal location among the ones previously identified and the size. To do this, the formulation includes Sequential Quadratic Programming (SQP) and Branch and Bound (BAB) algorithms. The article argues the results improve the ones from PSO and ELF.

- Hybrid Ant Colony Optimisation (ACO) and Artificial Bee Colony (ABC) Algorithm: The paper in [24] calculates the optimal location and sizing of DERs for a 33-bus and a 69-bus network. The model presented blends discrete optimisation for the location with continuous structures for the size optimisation, all to minimise power losses, total emissions and electrical costs and maximise voltage stability. This way, it combines global and local search abilities, characteristic of ABC and ACO respectively. ACO is used to determine candidate solutions for the location of the DERs, which is a discrete problem. Then, ABC calculates their size, which is a continuous problem. The paper argues the proposed method is better for multiple DER allocation optimisation problems.

- Grasshopper Optimisation Algorithm (GOA) based Fuzzy Multi-objective Approach: Optimum sizing and location of DGs in a 51 and a 69-bus system is done by using a fuzzy GOA approach in [25]. This aims to improve the power factor in substations, minimise power losses and maximise voltage control. The main advantages of this type of algorithm are its fast convergence rate and efficiency at identifying global optimums. The paper argues this method is much faster than GA and PSO.

Many other examples of possible methods for locating and sizing distributed energy resources can be found in the literature like Imperialist Competitive Algorithms, Multi-objective Particle Swarm Optimisation, Improved Harmony search algorithms and Intersect Mutation Differential Evolution, among others [25]. Nevertheless, most of them use small networks as objects of study and are not scalable to bigger systems.

As can be seen, a common denominator among several studies is that the optimisation is carried out in two steps: first, the possible locations for the batteries or other types of DERs are identified, and then, they are sized. Moreover, in all cases, the installation of DERs aims to provide several benefits to the grid; voltage control and stability being the most common.

4

Project FLEXENER

Task T.3.1.2 Results

In MATLAB, the optimisation using genetic algorithms is done by using the function gamultiobj. The problem has three fitness functions, corresponding with the three metrics that are going to be analysed: voltage control, reliability and investment. The gamultiobj function is programmed to solve minimisation problems, which has been taken into account when programming the voltage control metric as it is the only one that is linked to a maximisation problem.

The optimisation variable is the battery capacity that is going to be installed in each zone. Upper and lower bounds are used to limit the maximum and minimum battery capacity.

The problem is solved at a zone level and using a population size of 300 for the options. Regarding its output, there is no one solution for the problem but a set of possible solutions. The number of solutions depends on the population size and the ParetoFraction parameter, which is left at its default value, in this case, 0,35. The gamultiobj function finds a local Pareto set and, later, the results obtained are presented in a graph and a matrix:

- Output Matrix: Each row represents a possible solution, while each column represents a zone. A value for the installed capacity in kWh is presented for each zone. This matrix is very big, 125x24, so to analyse the tendencies only some of the solutions are used. They are chosen through pseudorandom sampling.

Zonas																								Capacidad total
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
2	0	4	1	5	9	7	1	5	12	1	657	3	18	4	0	3	1	3	1	0	6	9	0	751
2	0	14	5	5	21	5	3	10	15	9	1765	6	26	3	2	6	2	19	6	10	8	10	0	1952
5	0	4	1	6	10	1	2	11	6	4	3526	9	3	1	1	3	1	3	1	3	2	6	0	3608
2	0	4	4	4	14	3	2	6	5	14	6558	3	1	2	2	4	1	7	5	2	2	2	1	6647
4	1	6	21	78	37	5	5	16	126	17	9089	97	114	1	3	1	1	2	1	12	4	103	1	9748
1	0	1	0	0	0	0	0	1	2	0	11990	2	0	0	0	0	1	0	0	0	0	1	0	12000

Figure 7: FLEXENER T.3.1.2. Results. Sample from the Output Matrix [10]

- 3D Graph: represents each of the solutions, with a value for each of the three metrics (voltage control, reliability, and needed investment) and the total battery capacity installed.

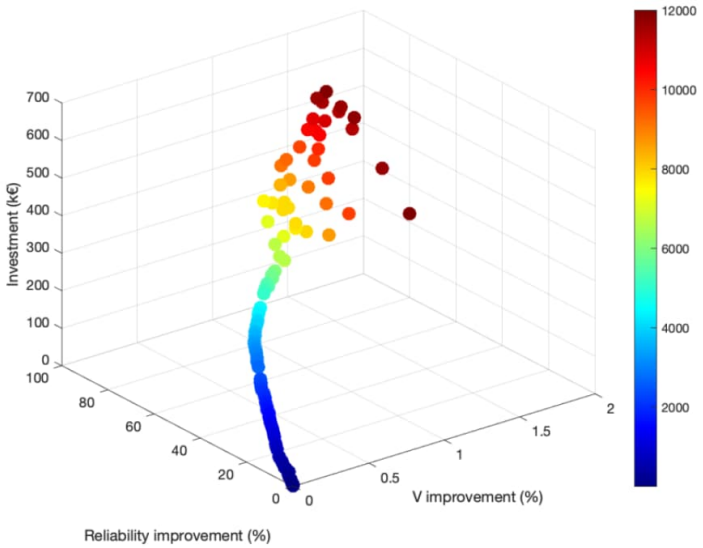


Figure 8: FLEXENER T.3.1.2 Results. 3D Graph [10]

Looking at the output matrix, it can be seen in which zones batteries tend to be placed. On the other hand, the 3D graph represents the impact on all three metrics of increasing the installed capacity. With this information, a suitable solution for the problem can be chosen.

As can be seen, for this specific case batteries tend to be placed on node 12. This is because it is the one furthest from the generation and, therefore, it is the one which is most likely to be disconnected from the generation coming from the main feeder, as can be seen in the following figure:

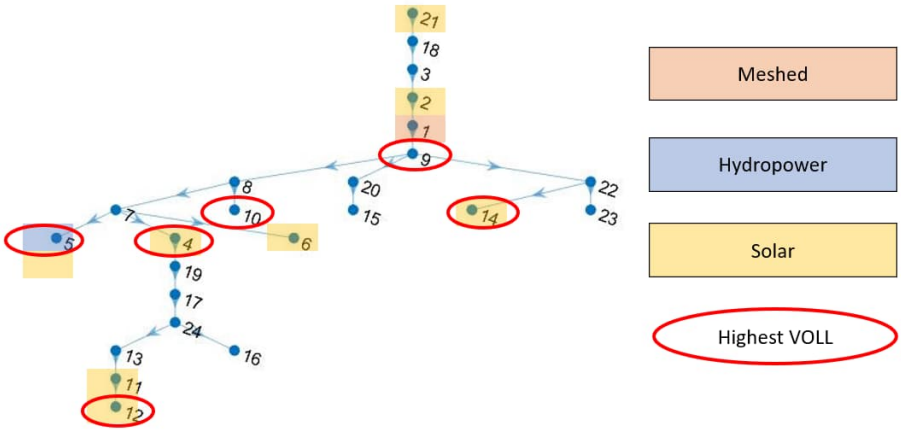


Figure 9: Network Topology after the Zone Division

This way, it is going to be one of the zones which is most likely to not be able to satisfy its demand. Moreover, it has one of the highest VOLLs, and therefore, the loss of supply in this node will have a high impact on the total reliability of the network.

To determine whether a solution is profitable or not, the needed investment needs to be compared to the improvement in reliability. To do this, it is important to take into consideration the TIEPI of the network, which is a reliability index very similar to ASIDI. In those networks where the TIEPI is lower, the profitability of installing batteries is expected to be greater. Moreover, it is also important to look at the graph relating Investment and Reliability. The flatter the curve, the higher the profitability, as it implies a greater improvement in reliability with a low investment.

Regarding the placement of batteries, they should be placed downstream, as it is further from generators. Nevertheless, a model like the one developed in project FLEXENER is needed to determine which is the exact node in which the batteries should be placed to obtain the best results possible.

Finally, project FLEXENER concludes the size of the batteries obtained using the algorithm is extremely related to the ENS.

As can be seen, the genetic algorithm generates a group of feasible solutions among which the best solution is later chosen. However, it does not guarantee optimality. Moreover, the bigger the problem the further the proposed solutions are from the optimal solution. A bigger problem would also affect computation times, as the size of the problem increases combinatorically. Therefore, to solve the problem and obtain the best solutions possible, different simplifications had to be made:

- Network size: Only a portion of the network data provided by i-DE was used for the optimisation problem. This data was chosen during the pre-processing stage.
- Zone Definition: Zones are divided attending to the placing of remotely controlled switches. Nevertheless, smaller clusters could have been used to optimise the placement of the batteries. These could have been done by looking at all locations where remote-control capabilities could be included and taking into consideration the extra cost this would imply. Moreover, other partitioning methods could be analysed.
- Investment Function: It has been considered that 0MWh implies 0€ costs. However, the investment function should not start at the origin. There are other costs apart from the one from the actual batteries in €/kWh that should be taken into consideration, like the cost of the inverters and the installation cost.

In this context, to be able to carry out this analysis in a more complex network, higher computational capacity is needed, which is why this project analyses the possibility of introducing quantum computing.

5

Quantum Computing

Quantum computing is an emerging technology used to solve problems which are too complex for classical computing. This is done by exploiting the laws of quantum mechanics, like superposition, entanglement and interference, and using them to estimate the attributes and behaviour of a quantum system [27].

Qubits are the basic unit of information in quantum computing, instead of using the conventional bit. While the latter can be either zero or one but never both, qubits allow for the superposition of both states. This way, they can be zero and one at the same time but in different proportions [28]. For this reason, superposition allows for the creation of complex multidimensional computational spaces where complex problems can be represented more easily [29]. The value of a qubit cannot be determined until the calculations are finished, as the final measurement ends with the superposition and the qubit collapses one of the two ways [37].

On the other hand, entanglement is the ability of qubits to influence each other, forming a single system. In this case, one qubit's quantum state cannot be described independently of the other's [28]. This way, one qubit is directly affected by changes in another. These relationships are exploited by quantum algorithms to solve complex problems [29], as they allow for the calculation of exponentially more information [28]. Both superposition and entanglement are essential for having a quantum advantage [59].

Finally, interference is related to superposition and refers to the capability of qubits to affect the likelihood of it collapsing in one of the two possible ways. This means qubits lose their quantum state to act like a bit. Quantum computers are designed to reduce interference as much as possible to guarantee good and accurate results [28].

5.1 Differences with Classical Computing

There are many differences between quantum and classical computing, some of which are presented next:

- Basic Unit of Information: As previously mentioned, quantum computers use qubits as their basic unit of information, while classical computers use bits. The ability of qubits to be in both states at the same time allows for testing different possibilities simultaneously, reducing the number of needed operations [30].
- Physical Architecture: Quantum computers consist of three main parts: an area for the qubits, a way of transferring signals to the qubits, and a classical computer to send and program these signals [28]. To reduce interference and stabilise the qubits, the first may need to be kept in a vacuum chamber and at a temperature around absolute zero. To do this, quantum computers use cooled superfluids. On the other hand, for classical computers, a regular fan is enough to maintain the system at a suitable temperature to work [29].
- Functionality: While classical computers are intended for everyday use, quantum computers are too complex to have this approach. In this sense, they are intended for more specific applications [31], such as simulation of complex systems, cryptography, optimisation, quantum machine learning and searches [28]. In particular, four industries are the most likely to experience the earliest benefits from this technology. They are the automotive, chemical, financial services, and life sciences industries [34].
- Programming Languages: In classical computing, a person programs a classical system using linear binary elements that are then processed to obtain the results. On the other hand, for quantum computers, the input is a matrix containing the optimised physical properties of particles, to then obtain the results from qubits [32]. Quantum programming languages are described in more depth in section 5.3.

5.2 Quantum Computing Technology Types

As several companies are carrying out research in this field of quantum computing and there is not an established technology yet, there are many kinds of quantum computers. Some examples are photonic, neutral, and Rydberg atom processors, among others [33]. Nevertheless, there are two methods whose use is more widespread:

- Gate-Based Quantum Computing: There are different types of gate-based quantum computers, such as ion trap processors and superconducting processors, among others [33]. For all, qubit states are initialised according to the input data [36] and are then modified according to a series of gate operations represented by a quantum circuit [35]. This means quantum circuits can control the evolution of the quantum states [56]. There are many different types of gates. They can change the quantum state of a small number of qubits [36], and put together they form quantum algorithms [41]. This method is more sensitive to errors like decoherence and noise than quantum annealing. However, it can be used to solve a great variety of problems, including both combinatorial and non-combinatorial problems [56].

- Quantum Annealing: This computing method is used for solving sampling and optimisation problems taking advantage of the fact that every system tends to seek a minimum energy state [38]. The initial state of the system is set as the ground state of a trivial Hamiltonian [36], thus it is known, and the system's configuration is modified so that the energy landscape reflects the problem that needs to be solved [33]. It also includes big energy penalties for violating the constraints [56]. This way, the system evolves to the final solution, which is the ground state of the final Hamiltonian [36]. In short, instead of manipulating quantum states like gate computers, quantum annealers use the natural evolution of quantum states to solve certain types of problems, which are probabilistic sampling and combinatorial optimisation [66]. One of the main advantages of quantum annealers is that they allow for a larger number of qubits than gate-based quantum computers [33]. Therefore, nowadays they can solve much larger problems. This can be achieved either by using purpose-built quantum computers or digital simulation in general-purpose quantum computers [56].

Sometimes, it can be useful to have a quantum and a classical computer working together to solve a problem. This is what is called Hybrid Quantum Computing [39] and is especially attractive since, as of today, quantum computers have a small number of qubits [41]. This way, it allows for working with somewhat larger problems [56]. With the evolution of quantum computing, both paradigms are becoming more and more integrated. This method takes advantage of both the flexibility of classical computing and the ability of quantum computers to solve complex problems [40]. In this sense, quantum computing performs the hard tasks, while classical computing deals with the rest, including HMI and high-level applications. This allows users to access quantum computing using classical computers through the cloud [40]. This method also allows researchers to explore the huge potential of quantum computing while it is still in its early stages.

Finally, it is important to mention another type of technology known as Quantum-Inspired Computing. They are able to solve specific problems by using the mathematical formulation of Quantum Mechanics. This can refer to methods like tensor networks and other methods that try to simulate the behaviour of quantum computers using classical ones [44]. In this sense, they are still limited by the computational capacities of classical computers regarding memory and processing speed [43].

5.3 Quantum Programming Languages

To solve problems in quantum computers, specific programs developed for this environment need to be developed. There is one fundamental difference between classical and quantum programming. While in classical computing a system is programmed to process linear binary elements to obtain a solution, in quantum computing a matrix with the optimised physical properties of particles is fed into the system to then determine the solution to the problem [32].

There are two main kinds of programming languages when it comes to quantum computing [32]: Imperative languages, which use statements to modify a program's state, and functional languages, which are based on applying and creating functions. Some of the most popular quantum imperative languages are QCL, QMASM and Silq, while regarding functional languages QML, Quantum Lambda Calculus and QFC and QPL are the most used [70].

In the future, to make quantum programming more efficient the automation of certain tasks that are usually done manually and ad-hoc will be needed. Such is the case of circuit layout and gate synthesis [42].

Another alternative to these languages is Quantum Software Development Kits or SDKs, which are the most popular option. These kits allow for the use of classical programming languages such as Python for developing quantum algorithms. Different open-source SDKs are being developed, among which some of the most popular are QDK by Microsoft, Qiskit by IBM and Ocean by D-Wave [71]. Qiskit is intended for Quantum Gate Computers, while Ocean is for Quantum Annealers. These SDKs provide the necessary tools to implement quantum algorithms through the cloud [71]. This way, they make quantum computing more accessible to the public, allowing for working in different hardware topologies.

5.4 Quantum Technology Benefits and Drawbacks

One of the main advantages of quantum computing is its inherent parallelism deriving from the superposition capabilities of qubits [33]. This allows for computing millions of operations simultaneously, considerably decreasing the total computation time needed to achieve the results. Nevertheless, this speed-up only exists when quantum mechanics offer algorithmic advantages [56]. Moreover, they can come in different levels: polynomial and exponential speed-ups, for example. In this sense, the main benefit of quantum computing lies in its potential to accelerate progress by solving problems that are too complex for classical computing. This is the case, for example, of quantum gravity, in the physics field [27].

Nevertheless, quantum computing still has many drawbacks, including the very specific conditions under which qubit housing needs to be placed. Moreover, the existing technology will need to be improved to control quantum coherence [27].

There are two main concerns regarding quantum computing. The first is building a large quantum computer with a sufficient quantity and quality of qubits, to ensure the obtention of correct results and minimise errors [34]. The second is its relationship with cybersecurity, as they are able to break asymmetric cryptosystems, frequently used for secure internet communication. Therefore, new cryptographic systems will need to be developed to ensure quantum safety, for which several candidates already exist [27].

In conclusion, quantum computing is an emerging technology with enormous potential. However, it still has various issues regarding scalability and quantum decoherence [31].

5.5 Optimisation Using Quantum Computing

There are several ways optimisation problems can be solved using quantum computers. In this section, some of them are presented:

- Quantum Annealers: These were previously explained in section 5.2
- QAOA: QAOA is a heuristic algorithm that belongs to the group of hybrid quantum-classical algorithms [50]. This is done to reduce the amount of quantum resources needed to solve the problem in comparison to a fully quantum algorithm [59]. It is designed to run in Quantum Gate Computers and approach QUBO formulation problems. They are able to solve combinatorial optimisation problems by generating approximate solutions. The quality of this approximation improves by increasing a parameter p [49].
- QIOA: This algorithm is an analogue of QAOA, meaning it is a classical quantum-inspired algorithm coming from the latter [50].

Moreover, other quantum-inspired methods can also be used to solve optimisation problems. This is the case with Digital Annealers. This alternative, which is specifically intended for solving complex large-scale combinatorial optimisation problems, is still conditioned by classical computing limitations. However, they offer a more robust solution than quantum computers as they do not suffer from problems with noise and quantum decoherence. The most popular is Fujitsu's Digital Annealer [72].

5.6 Applications in Energy Systems

Quantum computers can be extremely useful for power grids, where large and complex optimisation problems with numerous input and output variables are very frequent, each time more and more as the number of DERs increases and the power flow stops being unidirectional. In this sense, as quantum computing is still relatively new, literature shows several studies have been made regarding the possible uses of quantum computing for the power grid:

- Facility Location-Allocation in Energy Systems: This problem is essential for energy systems planning. It includes finding the optimal location of generation units such as solar, wind or hydropower taking into consideration electrical constraints, facility opening costs, cost of transportation and demand requirements, among others [51]. Most of these problems can be formulated as quadratic assignment problems, which are some of the most complex problems in the NP-hard class. In [51], a problem with 20 candidate locations and 20 facilities is solved using both classical and quantum computing, comparing both an IBM Q's quantum computer and a Quantum Annealer. Results show that with classical computing time grows exponentially when the problem size increases, which does not happen with quantum. Nevertheless, a reduction in the quality of the solution is observed for larger problems. As another example, in the article [54] a system with 5 nodes is modelled to solve a Max-Cut problem using QAOA, to help identify areas where renewable areas should be placed. This is run on a 5-qubit quantum computer obtaining positive results, as the quantum computer can consistently find the maximum cut. Nevertheless, more powerful quantum computers would be needed to scale this for larger networks.

- Unit Commitment Problem: This problem is critical for power systems operation [51]. It is usually formulated as a MINLP problem, which is an NP-complete problem impossible to solve with an algorithm with polynomial computation time. The problem needs to be discretised to transform it into a QUBO formulation which can be solved with a Quantum Annealer. This increases the number of variables in the problem. As shown in the article [51], as the size of the problem grows, the QUBO cannot be directly embedded into the QPU. This way, it requires to be divided into smaller problems, which reduces the chances of finding a globally optimal solution. However, larger networks also have higher limitations for biases, so a compromise must be found regarding the size of the problem. Other studies show the possibility of solving the unit commitment problem by means of QAOA. This is the case of [59], which uses QAOA to work with the binary variables and a classical optimising algorithm for the continuous variables. This allows for limiting the necessary gates for quantum computing and obtaining a considerable quantum advantage. The correctness of the results obtained shows that when the technology is more mature it will be able to perform effectively in large systems.

- Grid Security: Taking planned preventive and corrective actions is needed to prepare and react to power outages. This may require system operators to perform thousands of power flow scenarios, each of them considering a different grid fault. The potential speed-up in these calculations is explored in [53]. By using the HHL algorithm [52] in a quantum computer, the resulting speedup could be exponential. This way, for N-3 type contingency, a classical computer would take 10 days to perform the study while a quantum computer just 0.7 seconds. Nevertheless, there are not any quantum computers with a number of qubits which is high enough to solve this problem yet.

- Power Flow: In [55], quantum power flows are explored as, using noise-free quantum computers, they have the potential to achieve exponential speedups over classical computers. To do this, an improved version of the HHL algorithm [52] is used. In the study, the results achieved with the quantum power flow coincide with the ones from classical computing, which proves the correctness of the results obtained. Nevertheless, this test was done in a small system. It is still very limited due to quantum computers' coherence times and noise. As the technology matures, it is expected to become an extremely useful tool.

- State Estimation: The most common algorithm for state estimation in power systems is WLS, which implies carrying out several iterations of a set of linear equation systems. Article [57], talks about the implementation of HHL algorithms [52] to solve this kind of problem using quantum computing. However, for problems that are not well-conditioned, the quantum approach may not generate correct results and lose the quantum speedup. For this reason, the problem can be preconditioned, using an iterative optimisation to precompute the power system states, instead of using the HHL algorithm to calculate them. Studies have shown great results for both approaches in small systems. Nevertheless, it needs fault-tolerant quantum computers, so Variational Quantum Linear Solvers could also be explored with this purpose in the future.

These are just some of the possible applications of quantum computing in energy systems. Nevertheless, there are many others such as energy management, energy trading and emergence

control, which can also be formulated as optimisation problems and are essential for power systems operation [57]. Moreover, Quantum Machine Learning could also help carry out transient stability assessment, and Electricity prices and demand forecasting may also benefit from quantum computing.

In conclusion, the application of quantum computing in electrical grid optimisation could bring many benefits, such as the ability to solve complex problems with numerous variables and the possibility of providing near-real-time or real-time decision-making capabilities that could significantly improve grid operation, stability and efficiency. Moreover, apart from quantum computing, other quantum technologies could be introduced into the power grid, such as quantum sensors or quantum communications and quantum cryptography, which would also enhance the security of the grid. Nevertheless, it is important to remember this is not possible yet with current quantum technologies.

5.7 Current Implementation of Quantum Technologies in Smart Grids Internationally

Seen the incredible potential quantum computing has for Smart grids, several electric utility companies around the world are investing towards introducing quantum computing in different power system applications. Some examples are presented next:

- Enel in collaboration with Data Reply: Using QUBO, they have designed a model that creates an optimal plan for maintenance with crew mobilisation. For this, they have used D-Wave quantum computers and AWS cloud [58].
- E-ON in collaboration with IBM: Both companies work towards developing control and coordination algorithms to manage DERs using quantum computers [60].
- Dubai Electricity and Water Authority in collaboration with D-Wave and Microsoft Azure: They are exploring different quantum-inspired solutions that focus on power and energy systems optimisation to improve the country's efforts to become more sustainable [58], [61]. They aim to make the most of the available energy resources.
- USA Department of Energy's Los Alamos and Oak Ridge Laboratories in collaboration with EPB, a utility from Tennessee: Their project consists of demonstrating the effectiveness of using metro-scale Quantum Key Distribution (QKD) for ensuring secure communications in the power system. This system would be able to operate on the existing infrastructure even in extreme weather conditions. It encrypts and authenticates data by taking advantage of the randomness of quantum mechanics [62].
- USA National Renewable Energy Laboratory (NREL) in collaboration with RTDS Technologies Inc. and Atom Computing: They have developed an open-source application that allows to perform "quantum-in-the-loop" experiments. In this sense, the interface allows connecting power grids and quantum computers to perform real-time tests [63]. This will be extremely useful to validate quantum algorithms.

6

Computation Time Analysis

To determine where quantum computing could be beneficial, it is important to identify which parts of the code are more computationally complex. In particular, this section will analyse the computation time needed to execute the different parts of the algorithm and the scalability of the problem. To do this, it is important to reset the random number generator in each execution of the algorithm, to ensure the reproducibility of the results and properly compare the execution time in each situation.

6.1 Computation Time Distribution using MATLAB Profiler

To identify which parts of the code take longer to execute, MATLAB profiler is used. To better understand the results, the following graph shows the hierarchy of the different functions that are going to be analysed in this section, as they are expected to be the most computationally complex:

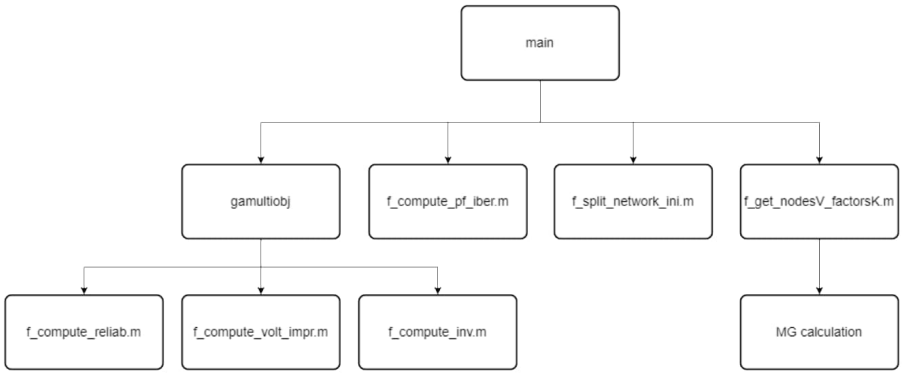


Figure 10: Hierarchy of the relevant Functions for the Computation Time Analysis

The base case, which corresponds to the one previously analysed in project FLEXENER has a total of 263 buses, which are divided into 24 zones. This corresponds to the second largest island belonging to the data from network 1. Including the function which calculates the power flow, the code takes

approximately a total of 212.57s to execute. This total computation time can be divided into three groups: data pre-processing, optimisation, and post-processing.

Starting from this base case, different scenarios are analysed. The first two scenarios correspond to the base case, and a modification of the base case which implies calculating an auxiliary matrix, which will be explained later in this section. The third scenario corresponds to the base case but running the function for computing the power flow beforehand and loading the results obtained as a matrix volt into the system.

This way, the total computation time has the following distribution among the three previously mentioned groups:

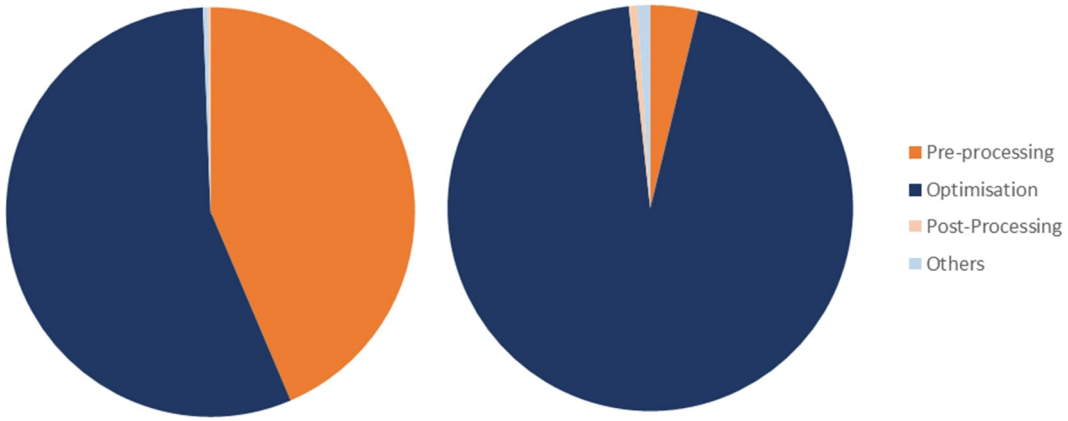


Figure 11: High-Level Distribution of the Total Computation Time for the Base Case Calculating the Power Flow (left) and Loading volt (right)

As can be seen, the optimisation takes up most of the computation time of the algorithm. Nevertheless, when calculating the power flow, the time dedicated to data pre-processing increases considerably. This way, two functions, in particular, take much longer than the rest to execute, as seen in the following graph. These are the power flow and the genetic algorithm.

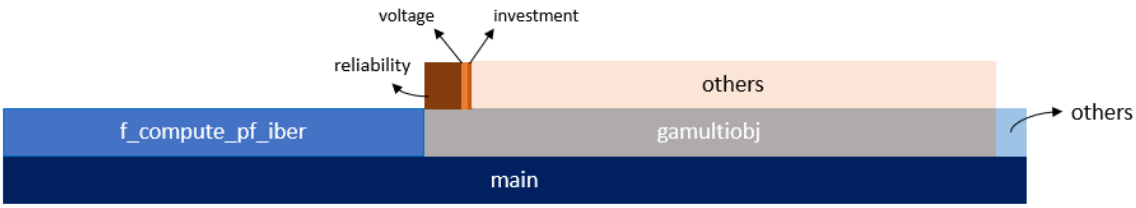


Figure 12: Distribution of the total Computation Time for the Base Case at a Function Level

The computation time distribution for each of the scenarios is shown in more detail in the next table:

Table 3: Computation Time Distribution for the Base Case

Running the power flow function	Not pre-calculating MG	Total Execution Time			212.57 s	100%	-
		Power Flow			87.39 s	41.11%	-
		Optimization using Genetic Algorithms		Total	118.70 s	55.84%	100%
				Reliability	7.63 s	3.59%	6.4%
				Voltage	0.43 s	0.2%	0.36%
				Investment	0.34 s	0.16%	0.29%
	Pre-calculating MG	Total Execution Time			214.26 s	100%	-
		Power Flow			88.55 s	41.33%	-
		Optimization using Genetic Algorithms		Total	119.16 s	55.61%	100%
				Reliability	9.68 s	4.52%	8.12%
				Voltage	0.63 s	0.29%	0.53%
				Investment	0.48 s	0.22%	0.4%
		MG Calculation			0.0039 s	0.0018%	-
Loading Volt	Not pre-calculating MG	Total Execution Time			137.77 s	100%	-
		Optimization using Genetic Algorithms		Total	130.23 s	94.5%	100%
				Reliability	8.73 s	6.34%	6.7%
				Voltage	0.49 s	0.36%	0.38%
				Investment	0.42 s	0.3%	0.32%
		f_load_case			4.73 s	3.43%	-
	Pre-calculating MG	Total Execution Time			125.35 s	100%	-
		Optimization using Genetic Algorithms		Total	118.69 s	94.7%	100%
				Reliability	6.53 s	5.21%	5.5%
				Voltage	0.43 s	0.34%	0.36%
				Investment	0.33 s	0.26%	0.28%
		f_load_case			4.3 s	3.43%	-
		MG Calculation			0.0037 s	0.003%	-

Regarding the computation time needed to execute them, first is the optimisation function gamultiobj. It takes up 55.84% of the execution time. As explained in previous chapters, the optimisation problem is made up of three different objective functions, which need to be calculated for every iteration of the algorithm. The calculation of voltage and investment metrics does not imply a great computational burden. However, the calculation of the reliability metric does. As previously mentioned, this complexity comes from the need to calculate each element of the interaction matrix, which depends on the faulted zone, the topology of the network and the optimisation variable which is the battery capacity that is going to be installed. This way, the reliability function takes up 3.59% of the total computation time, while voltage control and investment take up less than 1% of the total computation time each. With respect to the time needed for the optimisation, reliability represents

around 6%, voltage around 0.38% and investment around 0.3%. The reason why these percentages are so low is that gamultiobj calls a great number of internal functions to perform the genetic algorithm.

Looking at the specific code lines in the reliability function which take longer to execute, it can be seen a lot of the computation time is going towards locating which zones correspond to the same microgrid for every fault case. As this does not depend on the optimisation variable, it could also be pre-computed. To analyse the effect this would have on the computation time and determine if it would imply memory issues, instead of using the MATLAB function *find*, a binary matrix representing whether each analysed zone belongs to a certain microgrid in case of a fault (represented as MG in Table 3) is calculated during data pre-processing using the following formulation which is added to the pre-existing MATLAB algorithm in the function f_get_nodesV_factorsK:

```
for b2=1:num_zonas %Fault in b2
   num_mgs = max(mpc.mgs_affect(:,b2));
   if (num_mgs==0)
       MG(:,b2,:)=0;
   else
        for b1=1:num_zonas %b1 is the analyzed zone
                for mg=1:num_mgs
                     if(mpc.mgs_affect(b1, b2)==mg)
                        MG(b1,b2,mg)=1;
                     end
                end
        end
   end
end
mpc.MG=MG;
```

This matrix is then accessed during the reliability optimisation to calculate the ENS and total capacity in a microgrid for a fault in a certain zone. This is done by using the following code lines:

```
zonas_en_mg=mpc_opt.MG(:,i,j); %Vector de 0 y 1 que dice para fallo en
                                  i si pertenecen al microgrid j
c_mg=sum(capacidad_total'*zonas_en_mg);
ens_mg= sum(mpc_opt.zonal_table(:,9)'*zonas_en_mg);
VOLL = VOLL + fail*sum(mpc_opt.zonal_table(:,3)'*zonas_en_mg*ZI);
```

The results obtained show that this change slightly modifies the computation time, as can be seen in Table 3. Also, for the base case, it does not imply a memory problem. Nevertheless, it does not improve the total computation time, neither loading the volt matrix nor calculating the power flow, and loading this matrix MG during the optimisation still is the most computationally complex part of the reliability function.

There is another function that takes more time to execute than the rest. That is the function f_compute_pf_iber. It calculates the hourly power flow, for which it uses 41.11% of the total computation time. It is important to note that this function can also be run beforehand, and the resulting data saved into a matrix volt, which would then be loaded into the system. This would reduce the total computation time. It has been checked by running the code with the volt matrix, and the results obtained show a 35% reduction in the total computation time, as can be seen in Table 3. In this case, the most computationally complex part of the code is the optimisation and, in particular, the reliability function, followed by the f_load_case function.

In the base case, f_compute_pf_iber represents a big part of the total computation time. However, it is expected to perform better than the optimisation function for bigger networks. Therefore, to analyse the necessity of implementing the power flow using quantum computing as well, it is important to also analyse the scalability of this function.

In conclusion, the most computationally complex part of the problem is the optimisation. It is followed by the power flow. Nevertheless, this can be treated as a separate problem. Therefore, as expected, the optimisation is the part that should be implemented using quantum computing. The pre-processing part of the algorithm can stay in classical computation, as well as the power flow which can be pre-calculated.

6.2 Scalability Analysis – Big O Analysis

A Big O analysis of the algorithm is carried out to determine the time complexity of the algorithm. With this purpose, the algorithm can be divided into three main parts:

- Post-Processing: It contains the function f_plot_results and is expected to be the least computationally complex part of the algorithm, having linear time O(n). The time this part takes to execute is related to the number of iterations it takes for the genetic algorithm to converge, and to the number of buses.
- Pre-Processing: This is the biggest section of the algorithm and contains several functions, most of which increase their size linearly as O(n). Moreover, the scalability of most of the functions in this group depends on the number of buses. Nevertheless, there are some functions which are worth taking a closer look at:
    - *f_split_network_ini:* This function has a time complexity of O(n^2), as it has a while loop inside of a for loop for calculating the matrix mgs_affect. This is one of the only functions in the pre-processing group whose computation time depends on the number of zones.
    - *f_compute_pf_iber:* The growth rate of the power flow calculations is difficult to estimate, as it depends on several factors such as the network size regarding the number of both buses and branches, network topology, the number of loops, the presence of transformers with tap-changing capabilities and the algorithm used (Newton Raphson as default in this problem), among others. Nevertheless, power flow equations solved by the classical iterative methods generally scale with O(n) [55]. Moreover, in this problem, repetitive power flow calculations need to be carried out to have the hourly power flow profile, which takes up a large amount of computation time. Nevertheless, as all the iterations are completely independent, to reduce the computation time this task could be parallelised.
- Optimisation: This section contains the genetic algorithm and, therefore, the three objective functions. The time complexity of this group is related to the number of zones, rather than to the number of buses. First, regarding genetic algorithms, these kinds of algorithms are inherently chaotic. Therefore, determining their O() can be misleading. Next, focusing on each of the objective functions:

- *f_compute_volt_impr* and *f_compute_inv:* Both functions have a linear time complexity O(n), as there is a summation on the optimisation variable x, whose size increases with the number of zones.
- *f_compute_reliab:* This function has a time complexity of O(n^3) as it has two nested for loops, and a summation inside of them, which is equivalent to a third loop.

In conclusion, most of the functions in the algorithm have a linear time complexity. The optimisation is expected to be the most computationally complex function, for two reasons. First, because of the genetic algorithm, but also because of the reliability function. The latter has a time complexity of O(n^3). Therefore, the total time complexity of the problem will be at least O(n^3).

6.3 Verification of the Impact of the Network Size and Number of Zones in the Computation Time

As previously explained, the performance of the algorithm directly depends on the complexity and size of the analysed network. Therefore, this section aims to analyse the impact of changing the size of the network on the execution time. To carry out this analysis, both bigger and smaller networks than the base case will be studied.

Starting with a smaller network size, the system object of study is obtained by eliminating certain branches and analysing only one of the resulting islands. Therefore, the smaller case is included within the larger system. This way, different systems are analysed:

- Network with 29 buses: Starting from the base case, the branch which is deleted goes from bus 193129 to bus 195458. Two different islands result from this operation, one with 234 and another one with 29 buses. It is chosen to analyse the latter, as it is the one the slack bus belongs to. During pre-processing this network is divided into 4 different zones. The next figure shows the network topology:

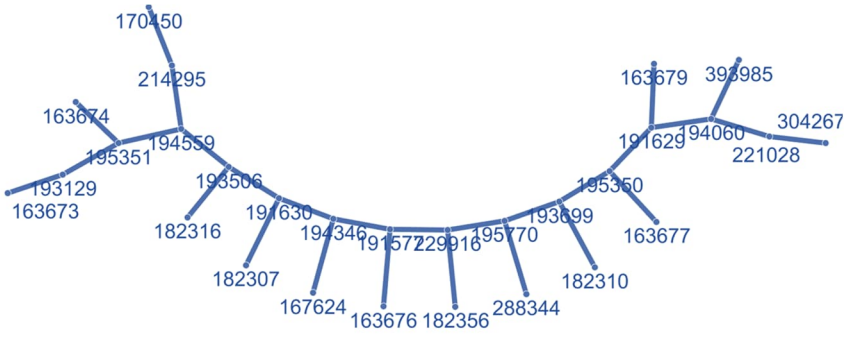


Figure 13: 29-bus Network Topology

- Network with 141 buses: To obtain this system, the branch going from bus 193398 to bus 229911 is eliminated. This results in two islands, one with 141 buses and another one with 122 buses. The one containing the slack bus is chosen as the object of study, which is the one with 141 buses. They are organised in 14 different zones during data pre-processing. The network topology is the following:

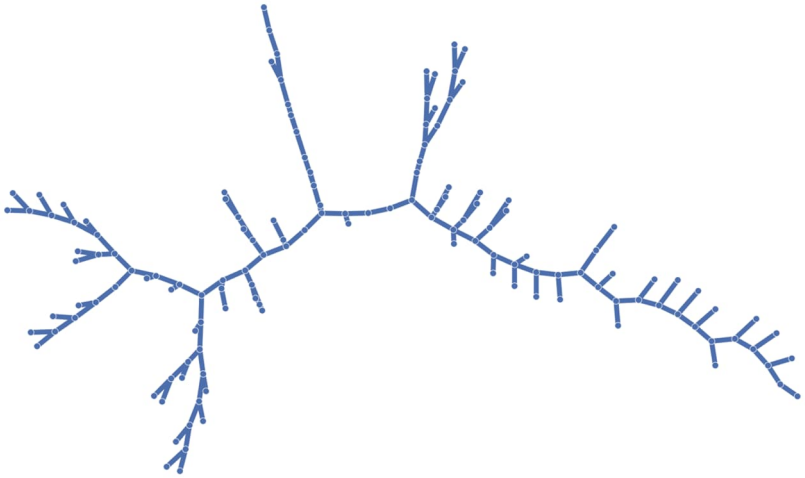


Figure 14: 141-bus Network Topology

Moving onto networks with a larger size, as there is not enough data available, and the optimisation is carried out at a zone level, the number of switches is modified to increment the number of zones. This way, the study is carried out in the same 263 bus network. In this sense, only the scalability of certain functions like the optimisation, the post-processing and certain functions that are part of the pre-processing group can be analysed.

First, the scalability of the calculation of the auxiliary MG matrix is analysed as it may be useful for problem reformulation. Its calculation time and size can be found in the following table:

Table 4: MG Size and Calculation Time

Buses – Zones	MG calculation	Size
29 – 4	0.0064 s	4x4x1
141 – 14	0.0081 s	14x14x3
263 – 24	0.0121 s	24x24x3
263 - 68	0.0248 s	68x68x12
263 – 108	0.04909 s	108x108x12
263 - 144	0.0476 s	144x144x6
263 - 180	0.0467 s	180x180x5
263 -225	0.0884 s	225x225x4
263 - 263	0.0805 s	263x263x3

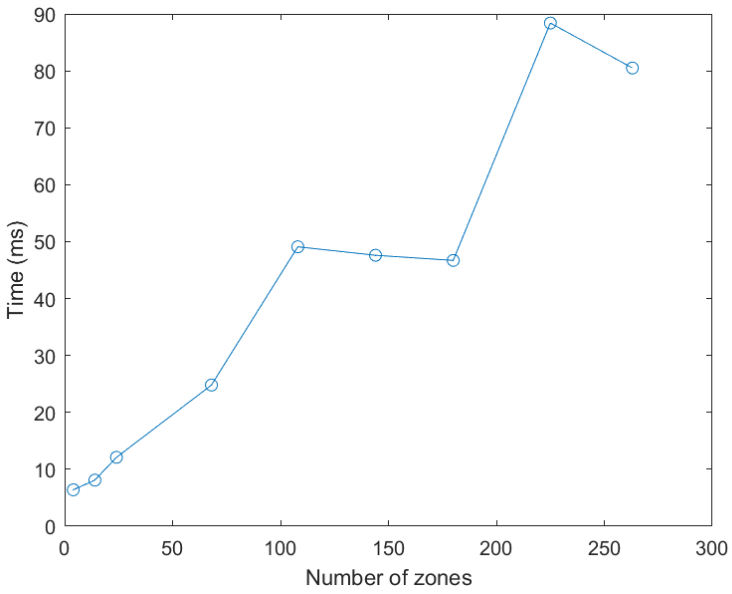


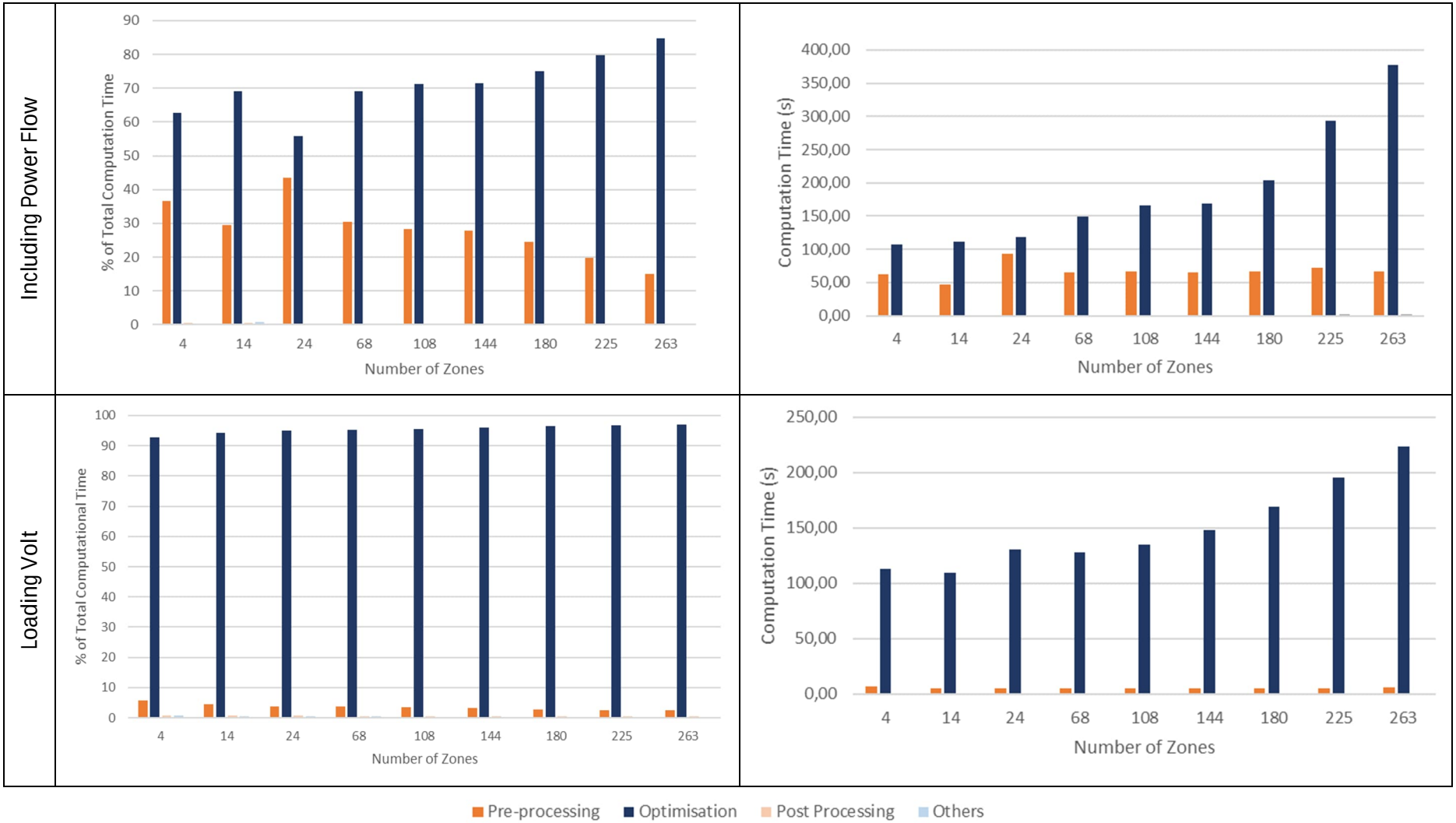
Figure 15: MG Scalability Analysis Results

As can be seen, even though the time needed for the calculation increases with the number of zones, it is always very low and therefore represents a very small proportion of the total computation time. Regarding the memory needed, as can be seen, it depends not only on the size of the network but also on the maximum number of microgrids generated in case of a fault in any of the zones. Nevertheless, it is not expected to be a problem for the implementation of this change into the algorithm.

In the previous section, it was concluded the calculation of the MG matrix did not imply great benefits either regarding the computation time or the results obtained. Therefore, the scalability of the classical problem will be analysed without the calculation of this matrix.

The following tables present a summary of the distribution of computation times for all the analysed network sizes. First, as done in the previous section, the computation time is distributed into three main groups: pre-processing, optimisation, and post-processing:

Table 5: Variation of the Computational Time with the number of Zones



As can be seen, in both cases and regardless of the network size, the optimisation is the group which takes up most of the total computation time, followed by the pre-processing. When increasing the number of zones, the proportion of the total computation time corresponding to the pre-processing decreases and the one corresponding to the optimisation increases. Nevertheless, it is important to remember that most of the analysed systems have the same number of buses, and a big part of the pre-processing is done at a bus level. Therefore, the percentage pre-processing represents should increase as the number of buses does so. However, as there is no data available for networks with a larger number of buses this cannot be analysed. Nevertheless, as explained in section 6.1, the computation time of the pre-processing is expected to increase more slowly than the one for the optimisation.

Once the high-level distribution of the time is known, the computation time for the different functions used in the algorithm is analysed. Some of them take much longer to execute than the rest. As identified in the previous section, these are the power flow, which belongs to the pre-processing, and the optimisation. Therefore, these functions are analysed in more depth in the following table.

Table 6: Computation Time Distribution for Different Network Sizes including Power Flow Calculation

Buses - Zones	Total Time	Power Flow	gamultiobj				
			Total	Reliability	Voltage	Investment	% of internal functions
29 - 4	170.325 s	57.56 s	106.39 s	1.94 s	0.535 s	0.405 s	97.29%
	100%	33.79%	62.46%	1.14%	0.31%	0.24%	
141 - 14	161.015 s	42.003 s	111.227 s	5.94 s	0.555 s	0.449 s	93.76%
	100%	26.09%	69.08%	3.69%	0.34%	0.28%	
263 - 24	212.57 s	87.39 s	118.7 s	7.63 s	0.43 s	0.34 s	92.93%
	100%	41.11 %	55.84%	3.59%	0.2%	0.16%	
263 - 68	216.197 s	60.85 s	148.635 s	35.806 s	0.855 s	0.756	74.82%
	100%	28.15%	68.75%	16.56%	0.4%	0.35%	
263 - 108	232.737 s	61.273 s	165.252 s	35.226 s	0.514 s	0.496 s	78.07%
	100%	26.33%	71%	15.14%	0.22%	0.21%	
263 - 144	234.833 s	61.43 s	167.75 s	56.704 s	0.616 s	0.58 s	65.48%
	100%	26.16%	71.43%	24.15 %	0.26%	0.25%	
263 - 180	271.43 s	61.33 s	203.20 s	70.18 s	0.603 s	0.61 s	64.87%
	100%	22.6%	74.86%	25.86%	0.22%	0.22%	
263 – 225	366.8 s	66.47 s	292.20 s	141.17 s	1.038 s	1.172 s	50.93%
	100%	18.12%	79.66%	38.49%	0.28%	0.32%	
263 – 263	444.375	61.59 s	376.19 s	172.45 s	0.98 s	1.001 s	53.63%
	100%	13.86%	84.66%	38.81%	0.22%	0.23%	

Table 7: Computation Time Distribution for Different Network Sizes Pre-calculating the Power Flow

Buses - Zones	Total Time	gamultiobj				
		Total	Reliability	Voltage	Investment	% of internal functions
29 – 4	122.087 s	112.773 s	1.515 s	0.428 s	0.324 s	97.99%
	100%	92.37%	1.24%	0.35%	0.27%	
141 – 14	115.803 s	108.66 s	4.422 s	0.421	0.335 s	95.24%
	100%	93.83%	3.82%	0.36%	0.29%	
263 – 24	137.77 s	130.23 s	8.73 s	0.49 s	0.42 s	92.59%
	100%	94.5%	6.34%	0.36%	0.3%	
263 - 68	134.324 s	127.534 s	28.097 s	0.635 s	0.542 s	77.04%
	100%	94.9%	20.92%	0.47%	0.4%	
263 – 108	141.261 s	134.437 s	29.359 s	0.451 s	0.416 s	77.52%
	100%	95.17%	20.78%	0.32%	0.29%	
263 - 144	154.716 s	147.96 s	41.043 s	0.45 s	0.438 s	71.66%
	100 %	95.63%	26.53%	0.29%	0.28%	
263 - 180	174.87 s	168.19 s	54.097 s	0.464 s	0.463 s	67.28%
	100%	96.18%	30.94%	0.27%	0.26%	
263 -225	201.72 s	194.69 s	77.35 s	0.52 s	0.53 s	59.72%
	100%	96.51%	38.35%	0.26%	0.26%	
263 - 263	230.38 s	222.46 s	97.14 s	0.544 s	0.566 s	55.82%
	100%	96.56%	42.17%	0.24%	0.25%	

For all the cases, the optimisation termites when the average change in the spread of the solutions in the Pareto Front is lower than the tolerance. This means the algorithm converges in a reasonable solution, as it is considered performing further iterations will not bring significant improvements. The results do not defer for loading the matrix volt or performing the power flow:

Table 8: Number of Generations needed for the Genetic Algorithm to Converge and Number of Solutions in the Pareto Front Depending on the Number of Zones

Buses - Zones	Number of Generations	Number of Solutions in the Pareto Front
29 – 4	103	105
141 – 14	153	105
263 – 24	148	105
263 - 68	305	105
263 – 108	556	105
263 - 144	102	105
263 - 180	209	105
263 -225	137	105
263 - 263	125	105

As can be seen in Table 6 and Table 7, for all cases the functions which are the most computationally challenging are the same: f_compute_pf_iber and the optimisation. Moreover, the lower the number of zones the faster the genetic algorithm reaches a solution, and therefore, the lesser the computation time. This is due to the great reduction in the time needed to perform the reliability metric. However, as can be seen in Table 8 this is not necessarily related to the number of generations in the genetic algorithm. In this sense, even though the computation time for the optimisation increases with the number of zones, this is not the case for the number of generations. This can be due to several reasons. Firstly, the time needed to perform selection, crossover and mutation operations and update the population can vary. This way, if the computational effort for one generation is high, it will take longer to execute. Moreover, the chosen initial population can impact significantly the number of iterations, depending on how much it resembles the optimal solution. It is also important to consider that in problems where there are multiple potential solutions, it can get harder for the genetic algorithm to converge. Finally, for larger networks, genetic algorithms have a risk of overfitting, meaning they converge to a localised solution, rather than searching for global optimums.

Nevertheless, it is important to note that, in this case, the number of solutions in the Pareto Front stays constant for all network sizes. This is because it is directly related to the population size set in the genetic algorithm, following a linear relation, as shown in the following graph:

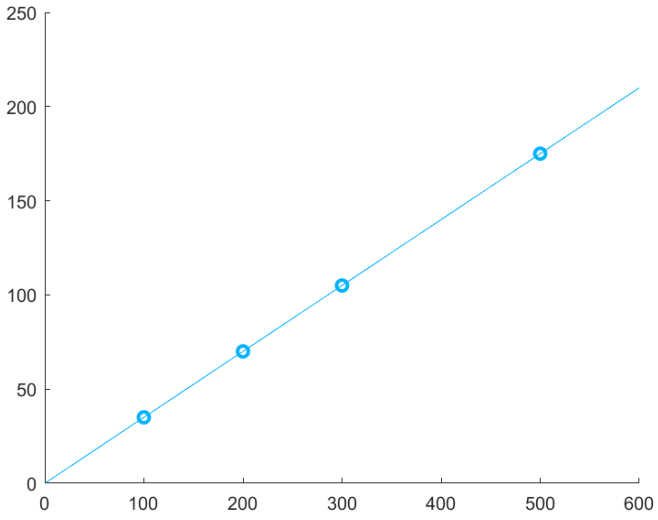


Figure 16: Relation between the population size of the Genetic Algorithm and the Number of Solutions in the Pareto Front

In this case, the population size is set to 300. Therefore, as the ParetoFraction option of gamultiobj has not been assigned a specific value and keeps the default value, which is 0.35, for all generations the number of solutions in the Pareto front is 105. By increasing the population size, the number of solutions in the Pareto front increases, while the number of iterations needed tends to get lower.

It is important to remember the power flow is run at a bus level. Therefore, it makes sense that for most all the scenarios with the same number of buses, the computation time of this function stays practically constant. Nevertheless, it is expected to grow more slowly than the optimisation.

All the data in the tables above is represented in the following figures:

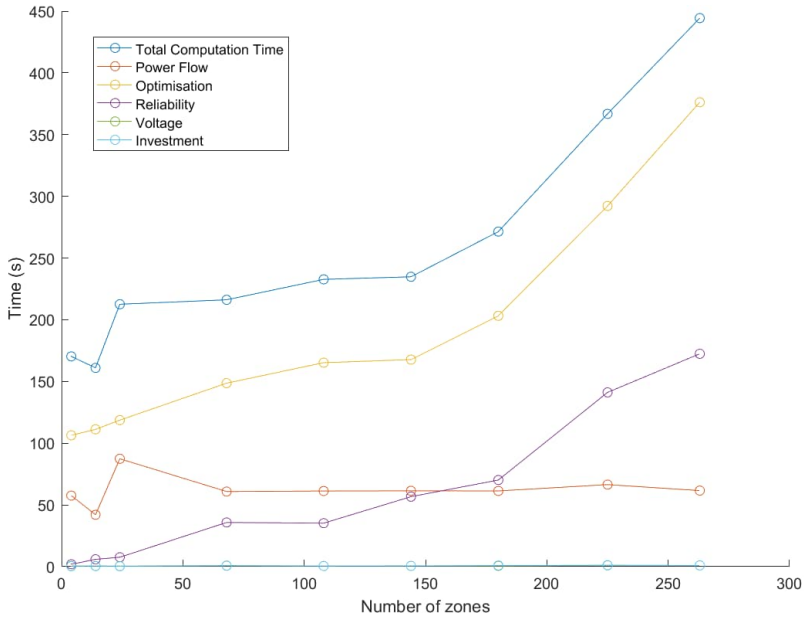


Figure 17: Scalability Analysis Results for the Algorithm Including the Power Flow Calculation

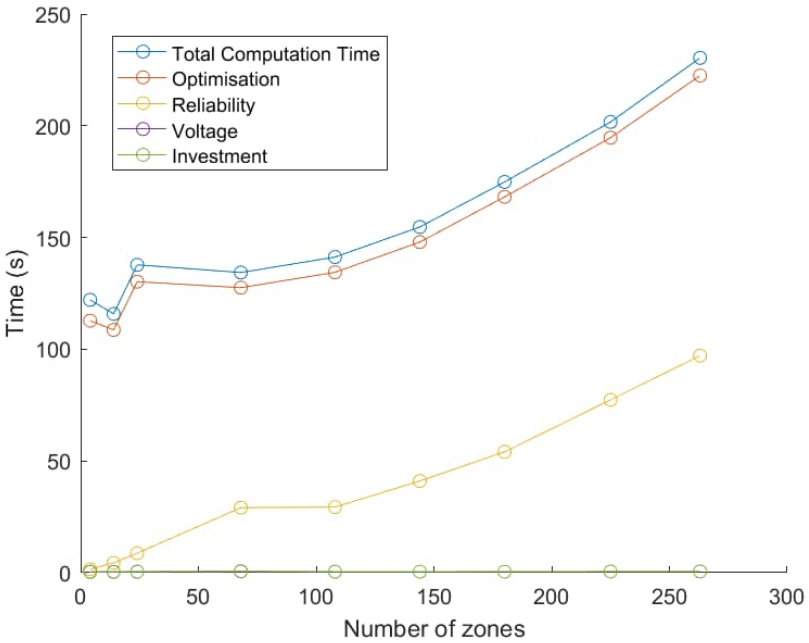


Figure 18: Scalability Analysis Results for the Algorithm Pre-Calculating the Power Flow

For the analysed scenarios, both figures show that the increment in time is directly related to the optimisation function and in particular to the reliability function. It is important to note that as the number of zones increases, the proportion of the optimisation time which is dedicated to internal functions tends to get lower. This is mainly because of the increasing importance of the reliability function.

As can be seen in Figure 17, the time needed to perform the power flow calculations does not experience big variations as expected because the number of buses stays constant. Nevertheless, its computation time is expected to increase when the number of buses does so. In this sense, some research has been carried out regarding the application of quantum computing for the calculation of power flows. As shown in [46], [47], and [48], different attempts to implement power flow in quantum computing have been made, achieving promising results. However, there are still many challenges to face until power flows can achieve a quantum advantage. In this sense, as explained in the previous section, even though the growth rate is expected to be smaller than in other parts of the algorithm, it will still take up a large amount of computation time. Therefore, it might be interesting to translate it into quantum computing in the future.

There is another part of the algorithm that is worth taking a closer look at. As previously explained, the scalability of the pre-processing cannot be analysed due to a lack of data. Nevertheless, there is a specific part of the f_split_network_ini function which calculates matrix mgs_affect which is also expected to be strongly affected by the increase in the number of zones, as explained in the previous section. As the different cases have the same number of buses, the execution time for most of this function should stay constant. The measured times for f_split_network_ini are:

Table 9: Computation Time of f_split_network_ini for Different Numbers of Zones

Number of zones	Time
4	0.119 s
14	0.147 s
24	0.151 s
68	0.164 s
108	0.172 s
144	0.207 s
180	0.189 s
225	0.253 s
263	0.286 s

The slight variation shown should be associated with the calculation of mgs_affect, as well as with the inherent variation when measuring computation times. However, the number of points and recorded times are too small to draw any conclusions regarding its growth rate. Nevertheless, it is not expected to be more time complex than the optimisation and the needed time for the matrix calculation is expected to grow more slowly. Therefore, it is not expected to be a problem.

The next graphs show the computational complexity determined in the previous section for the different functions coincides with the growth tendency presented by the algorithm. In particular, the reliability is approximated using a cubic function:

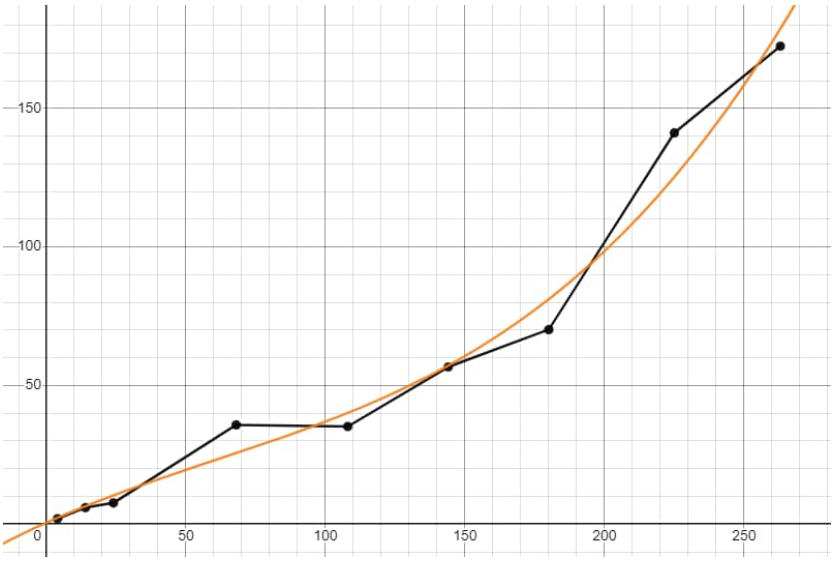


Figure 19: Reliability Computation Time Approximation to a Cubic Function (Number of zones – Total Computation Time)

This approximation has an R^2 of 0.9814. Therefore, it is concluded the reliability function has a time complexity of O(n^3).

For the total computation time, it was expected to be at least cubic, as the time complexity of the genetic algorithm is difficult to estimate. This way, it has been approximated by both an exponential function and a cubic function:

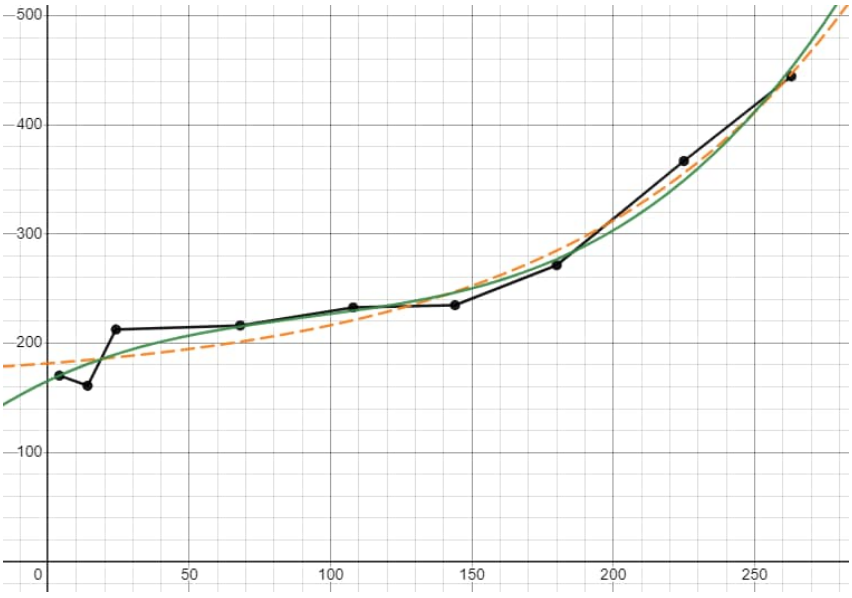


Figure 20: Total Computation Time Approximation to an Exponential and a Quadratic Function (Number of Zones - Total Computation Time)

The approximation using an exponential function, represented in orange, has an R^2 of 0.9685. On the other hand, the approximation to a cubic function, represented in green, has an R^2 of 0.9788. This way, it seems computation time has a time complexity of O(n^3). Nevertheless, to be completely

certain, data for bigger networks would be needed, as the genetic algorithm has the potential of having exponential growth, as they are inherently chaotic. Therefore, it is concluded the growth rate of the total computation time is at least cubic.

In conclusion, the optimisation is the part of the code where quantum computing would be most useful. Applying quantum computing to calculate the power flow could also be interesting. Nevertheless, this is not something that can be done for large networks with the current quantum computing technologies.

7

Choice of Quantum Technology and Formulation

Once the part of the algorithm where quantum computing would be most beneficial is identified, then a strategy for the implementation of this technology can be developed.

There are two ways in which this problem can be approached:

- Heuristic Approach: This way of solving the problem is more based on common sense rather than on specific calculations. In this sense, it will imply a less complex analysis of the network, without the need to perform power flow calculations. Some of the assumptions which could be useful in this approach are that installing two batteries in the same node will reduce the total installation costs and that installing batteries as far away as possible from the main feeder will benefit the largest number of nodes, for example.
- Quantitative Approach: This approach is the one originally followed in project FLEXENER and is based on numerically studying the impact of installing a new battery into the system. This way, the quantitative approach will allow for quantifying change, meaning how much every metric improves when installing a new battery.

In this sense, it will be attempted to solve the problem using a quantitative approach, leaving the more heuristic one as a backup.

As previously mentioned, there are several ways of solving optimisation problems using quantum computers. These include QAOA, QIOA and Quantum Annealers. Nevertheless, for this problem, the most suitable technology would be the latter.

QAOA is run in Quantum Gate Computers, which are still very sensitive to noise. This may compromise the optimality of the solution. On the other hand, QIOA is not very mature yet and requires a great number of computational resources. Therefore, the chosen technology to solve the problem is a Quantum Annealer, as this kind of computer is specific for optimisation and can tackle much larger problems than Quantum Gate Computers. In particular, D-Wave's Quantum Annealer will

be used, which is programmed using the company's SDK Ocean. This SDK contains different open-source Python tools that will allow for developing the quantum algorithm.

When using Quantum Annealers, the preferred formulation is QUBO, which stands for Quadratic Unconstrained Binary Optimisation. QUBO formulation allows for tackling a wide variety of combinatorial optimisation problems [64] which are generally NP-hard. As its name indicates, with QUBO formulation decision variables need to be binary. This way, every problem can be expressed as:

$$max/min\, y = x^t \cdot Q \cdot x$$

Equation 5: QUBO formulation

Where X represents the optimisation variable. In this sense, for this specific problem, when transforming the optimisation variable from continuous to discrete, it no longer represents the battery capacity, but rather if a certain battery capacity previously set is installed in a certain zone. This way, the vector X will have dimensions N·C x 1, being N the number of zones and C the number of different battery capacities. Regarding matrix Q, it is a square matrix containing constants and representing the problem. For the studied problem a matrix of N x N number of zones will be needed for each battery capacity. These matrixes will be the main diagonal of the Q matrix. The off-diagonal matrixes will represent the interaction between the different battery capacity scenarios, meaning if there are batteries of both types installed. As an example, the structure of the QUBO formulation for a system with four zones and two different battery capacities is shown in the next graph:

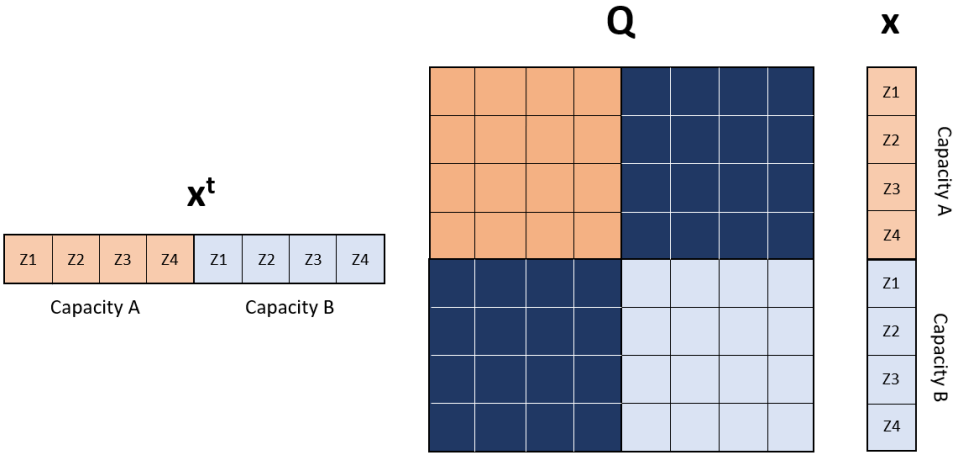


Figure 21: Structure of the QUBO Formulation for a System with Four Zones and Two Battery Capacities

Moreover, to be classified as QUBO, the highest power of the optimisation variable needs to be two, and no hard constraints can be applied. Therefore, all constraints will need to be implemented as penalties to the objective function. The weight of this penalty will represent the importance of meeting the constraint. This weight is usually empirically set, meaning it is determined after a trial-and-error process.

Additionally, it is important to note that in order to solve the problem using a Quantum Annealer, the genetic algorithm will be let go and it will be solved as an optimisation problem with a single objective function. In this sense, as the original problem has three objective functions, two constraints and uses a continuous variable, it needs to be reformulated to be able to solve it using a quantum computer. This is further explored in the following section.

8

Problem Reformulation

The parts of the algorithm that are going to be implemented using quantum computers need to be transformed to adapt them to this new environment. For this, several steps are needed. First, the optimisation problem is presented using an operational-research-like formulation to then be translated into QUBO formulation.

8.1 Discrete Optimisation Variable Using Genetic Algorithms

To be able to run the optimisation using a Quantum Annealer, the preferred formulation is QUBO. In this sense, to transform the algorithm into QUBO, the optimisation variable needs to be binary. To do this the optimisation variable is discretised. This way, it will represent whether a certain battery capacity, which will be previously chosen, will be installed in a zone or not. This is the first step is carried out in the following sections:

8.1.1 MATLAB Implementation

As the optimisation variable now needs to be discrete, most of the algorithm can stay intact. Nevertheless, some changes are needed as explained next. First, the population type for the genetic algorithm needs to be changed to 'bitstring' so that X receives binary values. This is done by adding the following code line:

```
options = optimoptions(options,'PopulationType', 'bitstring');
```

When using this population type, gamultiobj ignores all constraints. As the battery capacities are set beforehand by the user, it is assumed they will always be less than the maximum capacity. Nevertheless, the total capacity installed is not guaranteed to be less than the maximum. In a first approximation, the constraint is not implemented when doing the optimisation. This way, once the different possible solutions are generated the ones that do not meet the constraints are eliminated. However, with this approach, some possible solutions may be ignored as they did not belong to the Pareto front for being dominated solutions.

Moreover, the battery capacity in each case needs to be set by the user in the main. Different capacities will be studied, each of which will be studied separately, as the gamultiobj function does not allow for the evaluation of the three capacities at the same time. One option to solve this problem would be to evaluate each of the capacities consecutively, starting with the biggest one and finishing with the smallest one, and taking into account the solution for the previous battery size. This way, the optimisation for the biggest battery size would be computed. Then the batteries installed would be saved into the zonal table column 10, with the rest of the originally installed capacity. Next, the optimisation for the following capacity would be computed, and the process would be repeated. There is one main drawback to this strategy. The genetic algorithm does not return a unique solution, but rather a matrix with several solutions whose size depends on the population size, being this the maximum number of solutions. For this specific problem, in some cases, there can be 300 solutions as this is the population size set. This way, there are two options: evaluating all the possible combinations of solutions for all battery sizes, which will drastically increase the computation time to months, or randomly selecting a set of solutions for each of them. The latter has the risk of losing optimality. For this reason, it is decided each of the capacities is evaluated independently.

For simplicity for the user, as well as to avoid executing the pre-processing each time, the different capacities are implemented using a vector. In this case, four different capacities are analysed: 10, 500, 1000 and 5000 kWh, all of them smaller than the maximum battery capacity installed, which is 10000kWh. The value of the capacity is passed to the different optimisation functions, where it is implemented by changing a few code lines as shown next:

- Main → The entire optimisation needs to be implemented inside a for loop which goes through all the positions of the capacity vector. This is done as follows:

```
C=[10 500 1000 5000];

for i=1:size(C,2)

   %Calculation of the initial metrics
   [metric_volt] = f_compute_volt_impr(x0, C(i))
   [metric_reliab] = f_compute_reliab(x0, C(i))
   [metric_inv] = f_compute_inv(x0, C(i))

   %Optimization
   %Parameter definition
   nvars = mpc_opt.num_zonas;
   A = [ones(1,mpc_opt.num_zonas)];
   b = [10000]; %Maximum battery size
   Aeq = [];
   beq = [];
   nonlcon = [];
   lb = zeros(mpc_opt.num_zonas,1)';
   ub = 10000*ones(mpc_opt.num_zonas,1)';
   rng='default';

   %Objective Function Definition

   fitnessfcn =
   @(x)[f_compute_volt_impr(x,C(i)),f_compute_reliab(x,C(i)),f_compute_in
   v(x,C(i))];

   %Options Definition
   options = optimoptions('gamultiobj');
   options = optimoptions(options,'PopulationSize', 300);
   options = optimoptions(options,'PopulationType', 'bitstring');
```

```
%Optimización
tic
[x,fval,exitflag, output]  = gamultiobj(fitnessfcn,nvars, A,b,
Aeq,beq, lb,ub,nonlcon, options);
toc

%Delete solutions that do not meet the constraints
for j=1:size(x,1)
    cap(j)=sum(x(j,:)*C(i));
end

not_valid = find(cap>b)
x(not_valid,:)=[];
fval(not_valid,:)=[];

x=x.*C(i)

%Plot the results
figure;
f_plot_results(fval,x,metric_volt,metric_reliab);
txt=['C=', num2str(C(i))];
title(txt);
fprintf('The number of points on the Pareto front was: %d\n',
size(x,1));
fprintf('The number of generations was : %d\n', output.generations);

end
```

- Reliability → `function [metric_reliab] = f_compute_reliab(x,C)`

  `capacidad_total=x(:).*C+mpc_opt.zonal_table(:,10)`

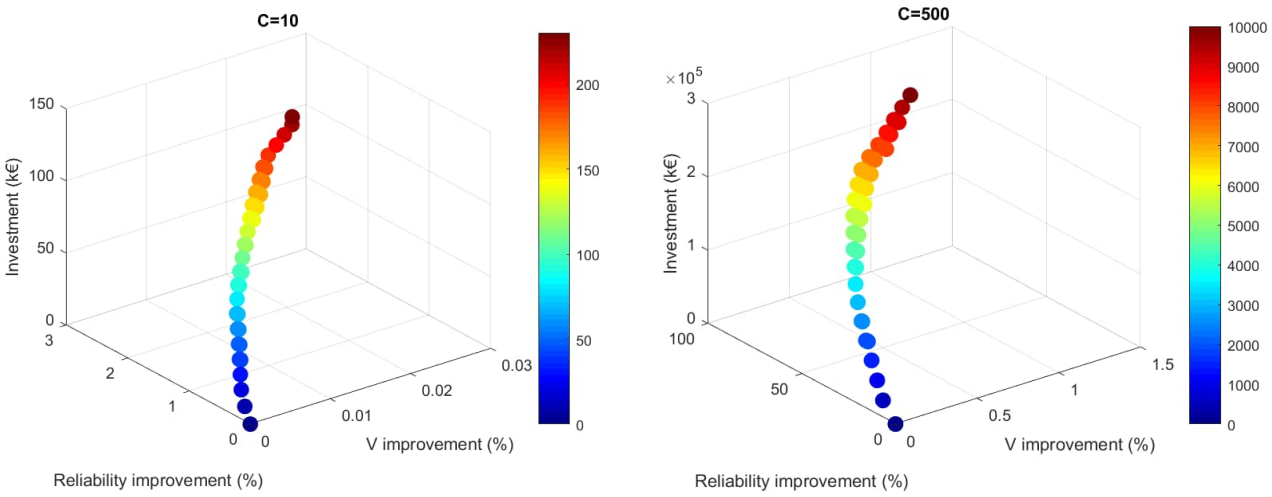
- Voltage → `function [metric_volt] = f_compute_volt_impr(x,C)`

  `capacidad_total=x(:).*C+mpc_opt.zonal_table(:,10)`

- Investment → `function [metric_inv] = f_compute_inv(x,C)`

  `metric_inv= sum(x(:).*((a)*C^2+b*C^2+c));`

The optimisation termites because the average spread of the solution gets smaller than the function tolerance. Therefore, the algorithm is able to reach a possibly optimal solution. The results obtained show the following distribution:



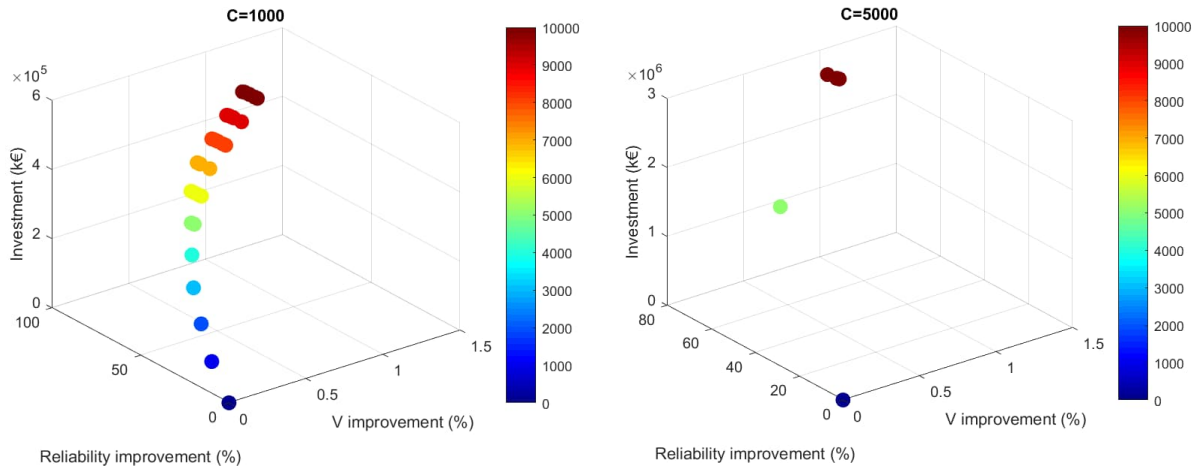


Figure 22: Results for solving the Problem with Genetic Algorithms and a Discrete Optimisation Variable and no constraints

Looking at the graphs obtained for the solutions, it can be clearly seen that now the optimisation variable is discrete, as each point represents the installation of another battery. This way, it makes sense there are more points in the graphs for smaller capacities than for large capacities. For example, for a capacity of 10kWh 1000 batteries can be installed and therefore there can be 1001 groups of points, while for a capacity of 5000kWh a maximum of two batteries can be installed and therefore, a maximum of 3 groups can appear, including the option of not installing any batteries. It is important to emphasize the notation "groups of points", as for each capacity there may be more than one possible solution, not one being better than the other. In this sense, there might be one solution which does slightly better in the voltage metric, while another one does so in the reliability metric, not one being better than the other in the overall analysis.

As shown in the following table, for bigger battery sizes a lot of options are being deleted because they do not meet the constraint:

Table 10: Number of Generations in the Genetic Algorithm, Number of solutions deleted for not meeting the constraint and Number of Solutions that Remain in the Pareto Front for different Battery Sizes

	Number of Generations	Initial number of Points in the Pareto Front	Number of solutions deleted	Number of Points in the Pareto Front after Deletion
C=10 kWh	102	105	0	105
C=500 kWh	102	105	6	99
C=1000 kWh	102	105	50	55
C=5000 kWh	121	105	96	9

Therefore, it is decided it is necessary to include the constraint into the optimisation somehow. In this sense, different strategies can be followed.

First, the constraint is included inside all three objective functions as a penalty in the cases it is not satisfied. To do this, an if statement is included in each objective function. This way, if the summation of all the batteries, so all the elements of x, is greater than the maximum value M, then the objective function is directly assigned an infinite value. If it is smaller than M, then the rest of the code in the objective function can be executed to obtain the true value of the objective function. This way, as it is a minimisation problem, all solutions with a total capacity higher than the maximum will be discarded. Using this approach, practically no variation can be seen in graphs for the overall results. Nevertheless, it is verified all the solutions obtained satisfy the constraints.

For this model, there is one particularity for big capacities which is worth mentioning. As the biggest capacity is of 5000kWh and the maximum capacity is 10000kWh, a maximum of two batteries can be installed. As such a big penalty is imposed for not meeting the constraint and it is so easy to go above the maximum capacity, the genetic algorithm tends to find the optimal by not installing any batteries at all, as shown in the following figure:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 23: Results of the Discrete Problem Including the Constraint in all three Objective Functions for a Battery Capacity of 5000kWh

To solve this problem, another constraint is included, to guarantee at least one battery is installed. To do so, the previously implemented if statement needs to be modified:

```
if sum(x(:).*C)<=10000 && sum(x(:))>0
    … %Original code for the objective Function
else
     metric_inv=inf;
end
```

Moreover, the initialisation vector x0 can no longer be 0, as an infinite value for all metrics will be obtained. Therefore, it must include one battery in a random position.

With these changes, the solution obtained for small capacities stays the same, while for big capacities it significantly improves, as shown in the following figure:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
9	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	5000	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	5000	0	5000	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	5000	0	5000	0	0	0	0	0	0	0	0	0	0	0	0

Figure 24; Results of the Discrete Problem Including the Constraint in all three Objective Functions and Guaranteeing the Installation of at least one battery for a Battery Capacity of 5000kWh

As can be seen, the solution reached shows that if only one battery can be installed, it must be done so in zone 12, while if a second one can be installed it should be done so in zone 10.

Another way to do this would be to include the constraint as an objective function. Nevertheless, it is important to remember this will not be a hard constraint, and therefore there might still be some cases where the constraint is not met to ensure the minimisation of the rest of the objective functions. As it is an inequality constraint, a slack variable will be needed. The binary expansion of this variable needs to be computed to obtain a set of binary variables that will be introduced into the optimisation problem. The binary expansion will depend on the battery capacity set as the maximum. The number of binary variables is determined by:

$$N = \log_2 M$$

Equation 6: Calculation of the Number of Variables in the Binary Expansion of the Slack Variable

With M being the maximum installed capacity and N being the number of binary variables needed for the expansion of the slack variable. If the result is not a natural number, then the immediate superior must be chosen. This way, for a maximum capacity of 10000kW, 14 binary variables would be needed. To simplify the following equation, a maximum capacity of 30kW is chosen, which will imply the need for 5 binary variables. This way, the constraint would be modelled as follows:

$$\sum_j \sum_i x_{i,j} \cdot B_j \leq M$$

$$\text{Const}\left(x_{i,j}\right) = \left(\sum_j \sum_i x_{i,j} \cdot B_j - M - s\right)^2$$

$$\text{Const}\left(x_{i,j}\right) = \left(\sum_j \sum_i x_{i,j} \cdot B_j - M - y_1 - 2 \cdot y_2 - 4 \cdot y_3 - 8 \cdot y_4 - 16 \cdot y_5\right)^2$$

Equation 7: Transformation of the Inequality Constraint to Introduce it as an Objective Function

As previously mentioned, this can be implemented into the code by including a new objective function. The number of binary variables needed for the expansion of the slack variable is calculated in the main as:

```
num_slack=ceil(log2(M);
```

Some other changes need to be made in the main, such as the number of variables in the genetic algorithm. The constraint itself is implemented in a function that can then be called by gamultiobj, as done with the rest of the objective functions:

```
function [max_capac]=f_contraint(x,num_slack,C)

      global mpc_opt;

      gamma=1;

      %Calculation of the binary expansion
      expansion=x(mpc_opt.num_zonas+1)*1
      for k=1:num_slack
          expansion=expansion+x(mpc_opt.num_zonas+k+1)*(2^k);
      end

      max_capac=gamma*(sum(x(1:num_zonas))*C-b-expansion)^2;

end
```

Gamma represents a parameter that allows for modifying the importance of the constraint. As previously explained, with this formulation it is not a hard constraint. Therefore, the higher gamma the more weight is given to this objective function and, therefore, the more important it is to meet the constraint. The value of gamma is set empirically. Nevertheless, it is important to keep in mind that for high values of gamma, the minimisation of the rest of the objective functions may be compromised.

8.1.2. Scalability Analysis

The evolution of computation time using a discrete optimisation variable is analysed next for the system with 263 buses and 24 zones without including the calculation of the auxiliary matrix MG. The option which uses a vector for the capacities, includes the constraints as a penalty to all three objective functions and a minimum of one battery installed is used.

As this formulation resembles more the problem that is going to be solved with quantum computing, its scalability is going to be analysed. This way, it will serve as a benchmark to study the benefits provided by quantum computing. The same network sizes used for the original formulation are analysed and the results obtained are presented in the following tables. It is important to note the critical functions in both formulations coincide:

Table 11: Computation Time Distribution for Different Network Sizes including Power Flow Calculation for a Discrete Optimisation Variable

Buses – Zones	Total Time	Power Flow	gamultiobj				
			Total	Reliability	Voltage	Investment	% of internal functions
29 – 4	68.53 s	50.89 s	10.29 s	4.44 s	1.37 s	0.99 s	33.92 %
	100 %	74.26 %	15.02 %	6.48 %	2 %	1.44 %	
141 – 14	91.55 s	63.67 s	20.04 s	13.22 s	1.44 s	1.11 s	21.31 %
	100 %	69.55 %	21.89 %	14.44 %	1.57 %	1.21 %	
263 – 24	115.63 s	81.22 s	26.58 s	19.54 s	1.33 s	1.05 s	17.14%
	100 %	71.1 %	22.99 %	16.9 %	1.15 %	0.91 %	
263 - 68	149.27	85.48 s	55.08 s	30.31 s	2.42 s	2.15 s	36.67 %
	100 %	57.27 %	36.9 %	20.31 %	1.62 %	1.44 %	
263 - 108	215.39 s	81.87 s	125.69 s	39.38 s	5.68 s	5.17 s	60.04 %
	100 %	38.01 %	58.35 %	18.28 %	2.64 %	2.4 %	
263 - 144	245.17 s	82.63 s	154.79 s	58.23 s	5.39 s	4.9 s	55.79 %
	100 %	33.7 %	63.14 %	23.75 %	2.2 %	2%	
263 - 180	402.64 s	91.74 s	301.81 s	78.22 s	10.24 s	9.51 s	67.54 %
	100 %	22.78 %	74.96 %	19.43 %	2.54 %	2.36 %	
263 – 225	401.86 s	81.72 s	312.39 s	69.95 s	9.02 s	8.11 s	72.12 %
	100 %	20.34 %	77.74 %	17.41 %	2.24 %	2.02 %	
263 – 263	775.65 s	80.34 s	687.46 s	180.16 s	17.35 s	15.37 s	69.03 %
	100 %	10.36 %	88.63 %	23.23 %	2.24 %	1.98 %	

The results above are represented in the following graph:

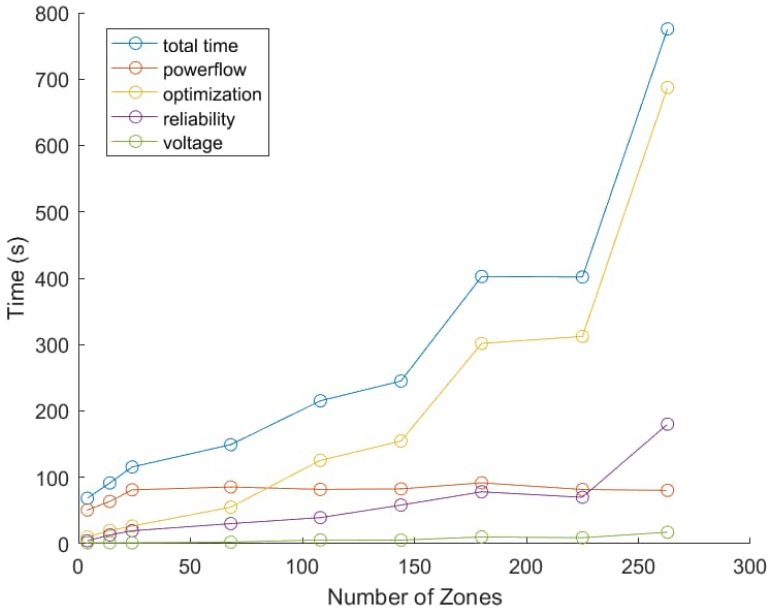


Figure 25: Scalability Analysis Results Using a Discrete Optimisation Variable and Including the Power Flow Calculation

As expected, the computation time for the power flow remains constant with respect to the problem with a continuous optimisation variable, as the pre-processing part of the algorithm has not been modified. However, this is not the case for the optimisation. First, as it can be seen the growth rate of the total computational time is heavily influenced by the genetic algorithm itself, rather than by the reliability function. This way, gamultiobj grows faster than the reliability function. In this sense, the proportion of the optimisation time which is related to internal functions of gamultiobj increases with the number of zones.

Table 12: Comparison of the Number of Generations Needed to Reach a Solution Depending on the Number of Zones for a Continuous and a Discrete Optimisation Variable

Buses - Zones	Number of Generations					Number of Solutions in the Pareto Front				
	Continuous	Discrete (C in kWh)				Continuous	Discrete (C in kWh)			
		10	500	1000	5000		10	500	1000	5000
29 – 4	103	102	103	103	103	105	105	105	105	105
141 – 14	153	102	103	107	105	105	105	105	105	105
263 – 24	148	102	102	102	102	105	105	105	105	105
263 - 68	305	116	115	264	626	105	105	105	300	300
263 – 108	556	146	254	336	2306	105	105	300	300	300
263 - 144	102	166	1362	917	364	105	105	300	300	300
263 - 180	209	140	2792	520	1353	105	105	300	300	300
263 -225	137	111	580	1496	2585	105	105	300	300	300
263 - 263	125	242	1008	2553	5026	105	105	300	300	300

As can be seen, the number of generations needed to solve a problem and the number of solutions in the Pareto Front tend to increase with the number of zones and the battery capacity. First, regarding the number of solutions in the Pareto Front, taking into consideration the Population Size is 300, it can be seen that for smaller networks the default value of the ParetoFraction option in gamultiobj is 0.35, while for larger networks it is 1. This way, for larger networks the number of solutions in the Pareto Front is 300, while for smaller networks it is 105. This is done to find a well-distributed set of solutions that allow for obtaining a result which is closer to the optimum. In this sense, using a higher Pareto fraction for small networks could lead to premature convergence of the algorithm towards a limited set of solutions, not allowing for finding the global optimum. For larger networks, the opposite occurs. The solution space is very large and the algorithm needs to focus on refining the quality of the solutions rather than ensuring their diversity, accelerating the convergence of the algorithm. This way, by changing the value of the ParetoFraction the genetic algorithm tries to find a balance between exploration and exploitation.

Regarding the number of generations, it can be seen that, generally, it tends to get higher with higher battery capacities. This may be because for higher capacities the number of feasible solutions is lower, as there is a lower number of batteries than can be placed to meet the constraint for maximum capacity. Therefore, for higher capacities, there is a larger number of solutions that are discarded and therefore, it takes the genetic algorithm longer to converge. Nevertheless, it is important to remember that many aspects can affect the number of generations needed to find a solution, such as problem complexity, which depends on the chosen network, or how close the initial population is to the optimal solution.

Finally, the time complexity of the algorithm is analysed. The results are shown in the following graph:

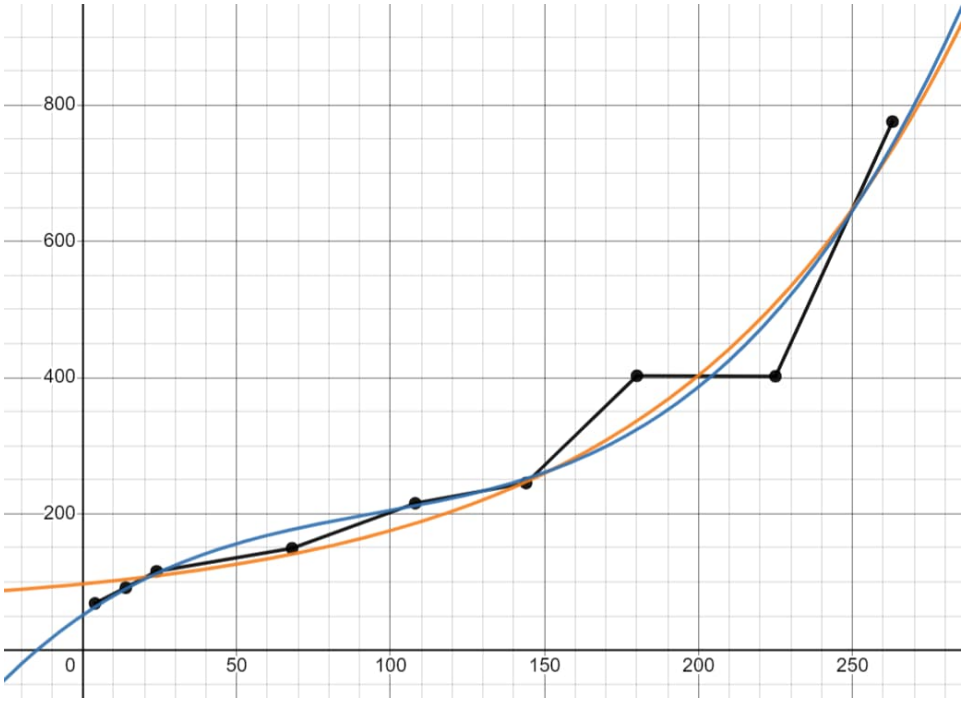


Figure 26: Approximation of the Total Computation Time for a Discrete Optimisation Variable Using a Cubic and an Exponential Function (Number of zones – Total Computation Time)

The higher variability in the computational time coming from the genetic algorithm makes it harder to identify the growth rate of the function. As done with the algorithm with a continuous optimisation variable, it is approximated using a cubic and an exponential function, represented in blue and orange, respectively. The results obtained for the R^2 for the cubic and exponential functions are 0.958 and 0.951, respectively.

As the optimisation problem is now inside a for loop depending on the battery capacity, the time complexity of the problem is expected to increase with respect to the continuous problem. Moreover, as seen in Table 11, the internal functions of the genetic algorithm, of which the time complexity is unknown, are responsible for a great part of the total computational time. Therefore, taking this into consideration and as the R^2 for both approximations is practically the same, it is determined the time complexity of the problem is exponential, as it is more aligned with the expected behaviour for this kind of allocation problem. Nevertheless, more data would be needed to confirm this statement.

8.2 Classical Formulation

As explained in the previous chapter, the implementation of this problem in Quantum Computing will lose the genetic algorithms. Therefore, in this section, the problem will be reformulated in that sense.

8.2.1 Continuous Optimisation Variable

The first step is to reformulate the problem in classical formulation, to create one single objective function and clearly identify the constraints of the problem. When combining the three objective functions into one it is important to include a set of parameters that represent the importance of solving each of them, while allowing for unifying the units. With this approach, the solution obtained is no longer a Pareto Front, but rather a unique solution. This way, the weight parameters can be modified depending on what wants to be achieved with that solution.

In this case, the same approach as the one taken in the FLEXENER algorithm is taken, using a continuous variable as the optimisation variable to represent the battery capacity that is going to be installed:

- Indexes:

  i Zones (b1= analysed zones, b2=faulted zone)

  mg Microgrids

- Parameters:

  M Maximum Installed Battery Capacity

  fr_{b2} Fault rate in zone b2

  $hrep_{b2}$ Reparation time in zone b2

  U Unitary Cost of the Batteries

  cap_c Battery capacity at which functions with and without economies of scale cross.

  Z_3_i Zonal table column 3. VOLL for zone i.

  Z_8_i Zonal table column 8. Value of voltage improvement for zone i

  Z_9_i Zonal table column 9. Total ENS in zone i.

  Z_10_i Zonal table column 10. Total generation capacity (not from the batteries) installed in zone i

  $MG_{b1,b2,mg} \in \{0,1\}$ $\forall$ b1, b2, mg If b1$\in$mg for fault in b2 $\Rightarrow$ MG=1

  aux $\in \{0,1\}$ If ENS=0 $\Rightarrow$ aux=1

  G_{b2} Number of microgrids for a fault in b2

- Variables:

  x_i Battery Capacity installed in zone i

- Objective Function:

$$\min a \cdot V(x_i) + b \cdot I(x_i) + c \cdot VOLL(x_i)$$

$$V(x_i) = -\sum_i Z\_8_i \cdot (x_i + Z\_10_i)$$

$$I(x_i) = \sum_i \frac{U}{(cap\_c - 2 \cdot M)} \cdot (x_i^2 - 2 \cdot M \cdot x_i)$$

$$VOLL(x_i) = \sum_{b2=1}^{i} \sum_{mg=1}^{G_{b2}} fr_{b2} \cdot hrep_{b2} \cdot IE_{b2,mg} \cdot ZI_{b2,mg}$$

$$ZI_{b2,mg} = \left(1 - \frac{C_{b2,mg}}{Z\_9_{b2} + aux}\right) \cdot (1 - aux) \qquad \forall\ b2, mg$$

$$C_{b2,mg} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot (Z\_10_{b1} + x_{b1}) \qquad \forall\ b2, mg$$

$$IE_{b2,mg} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot Z\_3_{b1} \qquad \forall\ b2, mg$$

Equation 8: Classical Formulation of the Problem Using Continuous Variables

- Constraints:

$x_I \geq 0$	Battery Capacity always positive
$\sum_i x_I \leq M$	Maximum total battery capacity

8.2.2 Discrete Optimisation Variable

Next, the optimisation variable is modified from continuous to binary. With this, the proposed formulation is the following:

- Indexes:

  i Zones (b1= analysed zones, b2=faulted zone)

  mg Microgrids

  j Battery Capacity

- Parameters:

  M Maximum Installed Battery Capacity

  fr_{b2} Fault rate in zone b2

  $hrep_{b2}$ Reparation time for zone b2

  U Unitary Cost of the Batteries

  cap_c Battery capacity at which functions with and without economies of scale cross.

  Z_3_i Zonal table column 3. VOLL for zone i.

  Z_8_i Zonal table column 8. Value of voltage improvement for zone i

  Z_9_i Zonal table column 9. Total ENS in zone i.

  Z_10_i Zonal table column 10. Total generation capacity (not from the batteries) installed in zone i

  $MG_{b1,b2,mg} \in \{0,1\}$ $\forall$ b1, b2, mg If b1$\in$mg for fault in b2 $\Rightarrow$ MG=1

  $aux \in \{0,1\}$ If ENS=0 $\Rightarrow$ aux=1

  G_{b2} Number of microgrids for a fault in b2

  B_j Battery Capacity j

- Variables:

  $x_{i,j} \in \{0,1\}$ $\forall$ i, j $x_{i,j}$ = 1 if Battery Capacity j is installed in zone i

- Objective Function:

$$\min a \cdot V(x_{i,j}) + b \cdot I(x_{i,j}) + c \cdot VOLL(x_{i,j})$$

$$V(x_{i,j}) = -\sum_{j}\sum_{i} Z\_8_i \cdot B_j \cdot x_{i,j}$$

$$I(x_{i,j}) = \sum_{j}\sum_{i} \frac{U}{(cap\_c - 2 \cdot M)} \cdot x_{i,j} \cdot (B_j^2 - 2 \cdot M \cdot B_j)$$

$$VOLL(x_{i,j}) = \sum_{j}\sum_{b2=1}^{i}\sum_{mg=1}^{G_{b2}} fr_{b2} \cdot hrep_{b2} \cdot IE_{b2,mg} \cdot ZI_{b2,mg,j}$$

$$ZI_{b2,mg,j} = \left(1 - \frac{C_{b2,mg,j}}{Z\_9_{b2} + aux}\right) \cdot (1 - aux) \qquad \forall\ b2, mg, j$$

$$C_{b2,mg,j} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot (Z\_10_{b1} + B_j \cdot x_{b1,j}) \qquad \forall\ b2, mg, j$$

$$IE_{b2,mg} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot Z\_3_{b1} \qquad \forall\ b2, mg$$

Equation 9: Classical Formulation Using a Binary Optimisation Variable

- Constraints:

  $\sum_j \sum_i x_{I,j} \cdot B_j \leq M$ Maximum total battery capacity

8.3 QUBO Formulation

For the QUBO formulation, the previously developed discrete algorithm needs to be transformed into an unconstrained problem. In this sense, the maximum battery size constraint is already implemented into the discrete code as the capacity is set by the user. Nevertheless, the maximum total installed capacity needs to be implemented as a penalty. In this sense, the formulation for the constraint developed in section 8.1 is used to transform the problem into QUBO:

- Indexes:

  i Zones (b1= analysed zones, b2=faulted zone)

  mg Microgrids

  j Battery Capacity

- Parameters:

  M Maximum Installed Battery Capacity

  fr_{b2} Fault rate in zone b2

$hrep_{b2}$ Reparation time for zone b2

U Unitary Cost of the Batteries

cap_c Battery capacity at which functions with and without economies of scale cross.

Z_3_i Zonal table column 3. VOLL for zone i.

Z_8_i Zonal table column 8. Value of voltage improvement for zone i

Z_9_i Zonal table column 9. Total ENS in zone i.

Z_10_i Zonal table column 10. Total generation capacity (not from the batteries) installed in zone i

$MG_{b1,b2,mg} \in \{0,1\}\ \forall$ b1, b2, mg If b1∈mg for fault in b2 ⇒ MG=1

$aux \in \{0,1\}$ If ENS=0 ⇒ aux=1

G_{b2} Number of microgrids for a fault in b2

B_j Battery Capacity j

- Variables:

$x_{i,j} \in \{0,1\}\ \forall$ i, j $x_{i,j}$ = 1 if Battery Capacity j is installed in zone i

s Slack Variable. Later expanded to a set of binary variables

- Objective Function:

$$\min a \cdot V(x_{i,j}) + b \cdot I(x_{I,j}) + c \cdot VOLL(x_{I,j}) + d \cdot Const(x_{i,j})$$

$$V(x_{i,j}) = -\sum_j \sum_i Z\_8_i \cdot B_j \cdot x_{i,j}$$

$$I(x_{i,j}) = \sum_j \sum_i \frac{U}{(cap\_c - 2 \cdot M)} \cdot x_{i,j} \cdot (B_j^2 - 2 \cdot M \cdot B_j)$$

$$VOLL(x_{i,j}) = \sum_j \sum_{b2=1}^{i} \sum_{mg=1}^{G_{b2}} fr_{b2} \cdot hrep_{b2} \cdot IE_{b2,mg} \cdot ZI_{b2,mg,j}$$

$$ZI_{b2,mg,j} = \left(1 - \frac{C_{b2,mg,j}}{Z\_9_{b2} + aux}\right) \cdot (1 - aux) \qquad \forall\ b2, mg, j$$

$$C_{b2,mg,j} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot (Z\_10_{b1} + B_j \cdot x_{b1,j}) \qquad \forall\ b2, mg, j$$

$$IE_{b2,mg} = \sum_{b1=1}^{i} MG_{b1,b2,mg} \cdot Z\_3_{b1} \qquad \forall\ b2, mg$$

$$Const(x_{I,j}) = \left(\sum_{j} \sum_{i} x_{I,j} \cdot B_j\text{-}M\text{-}s \right)^2$$

Equation 10: Formulation Using a Binary Variable and No Constraints (QUBO)

Depending on the value given to the parameter d, the importance of meeting the constraint can be modified as it is not implemented as a hard constraint.

As explained in Section 8.1.1, s represents a slack variable, of which a binary expansion needs to be performed. This will depend on the value of M.

To finish, to be expressed as a QUBO, it needs to have the following structure, as explained in section 7:

$$\max/\min y = x^t \cdot Q \cdot x$$

x represents the variables, which in the case of the problem are the slack variables and for each battery capacity if it is installed in a zone or not. The constant matrix Q is obtained from the equations previously presented and its size depends on the number of zones, the number of battery capacities studied, and the number of binary variables needed for the slack variable. This way, it will be a square matrix of dimensions: ((zones + slack) · capacities) x ((zones + slack) · capacities).

9

Quantum Computing Approach

In this chapter, several strategies for solving the problem using quantum computing are presented. Nevertheless, the final solution for the problem using quantum computing is yet to be determined. There are a couple of strategies that may be interesting as intermediate steps for reaching a solution.

9.1 Quantitative Approach

As previously mentioned, there are two ways of approaching this problem using quantum computing. The quantitative approach is related to solving the problem using the equations presented in Section 8.3.

Taking away the genetic algorithms, reliability seems to be the most complex function of the original problem. Nevertheless, when translating the problem into QUBO formulation the complexity of the problem varies. This is explained with the following equations:

$$Metric\_Reliab = \sum_{b1=1}^{b1=B} \sum_{b2=1}^{b2=B} fr_{b2} * h\,rep_{b2} * IE_{b1} * ZI_{b1,b2}^{ns}$$

$$ZI_{b1,b2} = 1 - \frac{C}{\sum ENS_Z} \rightarrow ZI_{b1,b2} = \frac{C}{\sum ENS_Z}$$

Equation 11: Transformation of the Reliability Function into QUBO Formulation

As we are trying to minimise the variable x, constant terms can be taken away to facilitate the calculations. This way, instead of minimising the VOLL we would be maximising the losses that would be prevented with the installation of the battery. Joining these two functions, we obtain the new metric of reliability for one battery size:

$$Metric\_Reliab = \sum_{b1=1}^{b1=B} \sum_{b2=1}^{b2=B} \frac{fr_{b2} * h\,rep_{b2} * IE_{b1} * C}{\sum ENS_Z} \cdot x_{b1} = \sum_{b1=1}^{b1=B} Q_{b1} \cdot x_{b1}$$

Equation 12: QUBO Formulation of the Reliability Metric

As can be seen, there is an interaction between faulted zones and batteries, but not between batteries. Consequently, there are no quadratic terms affecting the optimisation variable x, meaning the problem is linear and, therefore, the QUBO matrix diagonal. This way, it would be hard to show quantum advantage as it can be solved using classical computation. Therefore, the introduction of new terms into the problem should be studied to increase the complexity and make it more suitable for quantum computing. This could be studying the interaction between batteries and allowing for more than one battery of the same size to be placed in the same zone.

9.2 Heuristic Approach

Even if using the previously discussed formulation it could be difficult to show a quantum advantage, there are other more heuristic methods which could be used to solve the problem using quantum computing and that could bring good results.

9.2.1 Energy Community Detection

Firstly, quantum computing can be used to create logical partitions of the network known as Energy Communities, which would allow for simplifying the network even more to work with large power systems, as shown in [65]. This is known to be an NP-complete problem; therefore, quantum computing could be extremely useful to solve it.

In this sense, the previously calculated zones will be divided into energy communities, simplifying the allocation problem that will later be solved using either quantum or classical optimisation algorithms depending on the size of the resulting network. As mentioned, this would allow for working with larger networks. Energy communities would be formed by a group of zones, each of which is formed by a group of nodes, among which the most representative node is chosen during pre-processing attending to the maximisation of voltage control as previously explained.

Literature shows using quantum annealing to partition a large network has achieved very positive results. Therefore, it could be explored for the set problem. To do this, complex network and graph theory can be used to facilitate the clustering process by using modularity optimisation. This way, the electrical grid is represented as a graph of buses connected by lines. The optimisation problem proposed with this purpose will try to maximise a quality metric for modularity, which will show how connected the elements of one partition are compared to other partitions.

Another way of applying the concept of energy communities would be setting a certain number of batteries N that can be installed attending to the maximum investment. Then, the network would be divided into said number of communities to place a battery in each of them. Nevertheless, this may not be the most optimal solution, as batteries may be placed in nodes which have a lower risk of outage.

9.2.2 Eigencentrality

Once the network is simplified, then the possible locations for the batteries can be analysed. As a first approach, the end of the graph branches or the pendant nodes are expected to be the most critical because they are connected to the least number of nodes and, therefore, are more likely to suffer an outage in case of fault. In this sense, eigencentrality would allow for locating this kind of node. This is the opposite of what is done in [66], where the t-most central nodes are identified. Nevertheless, as the importance of nodes is ranked, the t-least central nodes can also be determined by looking at the t-least important nodes in the rank. This ranking is done by attending to the scores from eigenvectors. This way, the problem of eigencentrality consists of finding the eigenvector which corresponds to the leading eigenvalue of the adjacency matrix of the network. In the case of [66], the problem was addressed both using a quantum annealer and QAOA on an IBM gate quantum computer. In the case of our problem, it would be done using a quantum annealer as previously explained.

It is important to remember the set problem has several sources of generation: the main feeder and the distributed generation. DERs can be installed in different parts of the network, including pendant nodes. In this case, the eigencentrality algorithm may choose them as a not-important node, and, therefore, as a candidate for placing a battery. However, in some cases, they shouldn't be. To avoid the later used optimisation algorithm choosing these nodes for placing the batteries, a penalty can be included. Moreover, depending on the chosen communities and the resulting network topology there might be interesting nodes which are not considered when calculating the eigencentrality. As an example, the following network from [65] is presented:

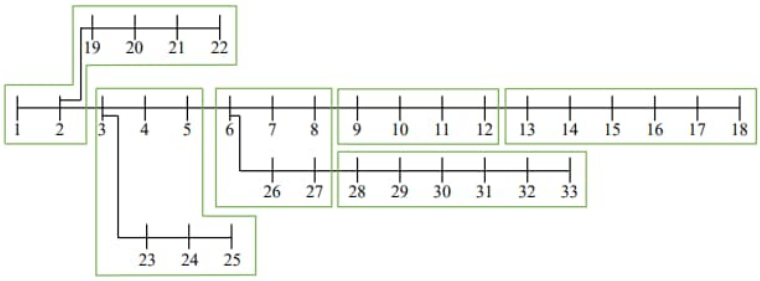


Figure 27: Example of Energy Communities Partitioning of a Radial Network [65]

Attending to the resulting communities which are represented in the diagram, the following graph can be obtained:

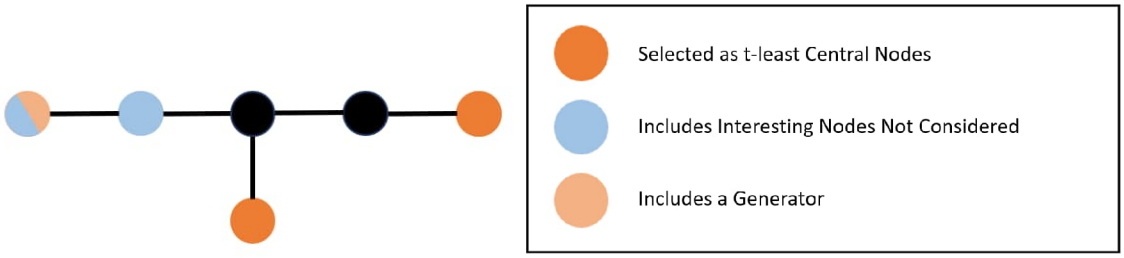


Figure 28: Example of the graph obtained from Energy Communities in Figure 27

As can be seen, some of the communities contain pendant nodes which may be interesting (represented in blue). Nevertheless, when translated into the graph this is not reflected and,

therefore, they are not considered in the eigencentrality calculations. This is the case of node 25 and node 22, which would be discarded for including generation in the zone. To avoid this, some metrics for each community can be pre-computed before using eigencentrality in the entire network.

9.2.3 Distance Between Nodes

Finally, to solve the allocation problem, another important consideration is that when placing batteries and generators as far as possible from each other, the largest portion of the network is covered. Therefore, the distance between these nodes should be maximised. Moreover, the loss of supply in every node does not affect the total reliability in the same way. In this sense, as reliability is measured in economic terms, attending to the VOLL, the loss of supply in those nodes with a higher VOLL will have a higher negative impact on reliability. In this sense, the distance between this type of node and a source of generation, like batteries, should be minimised.

To calculate the distance between nodes several strategies can be followed. One of them is Dijkstra's classical algorithm. However, other quantum alternatives allow for solving the longest and shortest path problems, which essentially means maximising and minimising the distance. The following table summarises the different QUBO formulations that could be used to solve each kind of problem using a Quantum Annealer:

Table 13: Alternatives for Solving Shortest and Longest Path Problems Using Quantum Annealing

Shortest Path Problem [67]	
Hop-based Approach	• It is not really a Shortest Path Algorithm. For a set number of steps, meaning a set number of vertices that need to be visited (|P|), the algorithm returns the lowest-cost path of that length. • Suitable for both directed and undirected graphs. • Uses one qubit per vertex per possible position, so per node and per step: |P|·|V|
Directed Edge-Based Approach	• Is more of a Shortest Trail Algorithm, as it allows for visiting vertices more than once. However, with a constraint for no negative-cost cycles, both problems are equivalent. • Used for directed graphs. Undirected graphs can be converted to directed graphs, doubling the qubit usage. • More economical in terms of the use of qubits: |E|
Undirected Edge-Based Approach	• Modifies some of the constraints of the previous method to make it suitable for undirected graphs. • Used for undirected graphs • Uses one qubit for each edge and each vertex: |V| +|E|. When |E|>|V|, this approach is more economical in terms of qubits than the previous one for undirected graphs.

Longest Path Problem [68]	
Order-based Formulation	• The algorithm looks for the longest path that visits a set number of vertices • Paths are described as an ordered sequence of vertices which has no repetitions • The objective is to maximise the objective function, but Quantum Annealing only minimises, so it must be negated. • Uses one qubit per vertex per possible position: $\|P\|\cdot\|V\|$ • Suitable for radial networks • The construction time of matrix Q for long paths can be too demanding
Degree-based Formulation	• Paths are described as a set of vertices and edges with degree constraints. • Uses one qubit for each vertex and each node: $\|V\| + \|E\|$. • Suitable for radial networks • More suitable than the previous one for large networks.

For both of the proposed algorithms for solving the longest path problems, the start and end nodes need to be fixed. This is not the case for this project. This way, they should be transformed to allow for arbitrary starting and ending. This is fairly easy for the order-based formulation, and more complex for the degree-based formulation, as extra variables need to be included.

It is not clear that any of these quantum alternatives provide significant benefits compared to classical algorithms for large networks. The limited availability of qubits in current quantum computers is one of the main obstacles. Therefore, the suitability of these tools for the set problem should be analysed. Nevertheless, this is not expected to be a problem as the size of the networks is reduced in previous steps. In addition, alternatives like Hybrid Solvers could be used to reduce the number of qubits needed. In this sense, the quadratic parts of the QUBO problem would be solved using a quantum computer, while the linear parts could be solved using a classical computer.

To finish, with all this data, a resource allocation problem can be solved to find the final solution among the possible locations identified in the previous steps. This kind of problem was explored in further detail in section 5.6.

As previously mentioned, this is just a first approximation that might change as the project develops and starts being implemented.

10

Conclusions

Quantum computing has the potential of bringing huge benefits for different sectors, the power systems among others, as it allows for solving problems which are too complex for classical computing. This project aimed to analyse the applicability of this new computing paradigm to a specific problem related to electric networks. This problem is the design of a battery network that maximises voltage control in normal operation and reliability in case of fault while minimising the investment. To do so, firstly, the classical algorithm developed in MATLAB by Universidad Pontificia Comillas in collaboration with i-DE has been analysed. The pre-processing part of the algorithm mainly depends on the number of nodes in the network, while the optimisation depends on the number of zones into which the network is divided. Taking this into consideration, the scalability analysis carried out does not picture the growth rate of the pre-processing, as it is done mainly at a zone level because of a lack of data. Nevertheless, looking at the formulation of the algorithm, most of it is not expected to be a computational burden. Said scalability analysis, however, has shed some light on what parts of the algorithm show the highest growth rates and, therefore, could be the most suitable for translating into quantum computing: the optimisation and the power flow. Nevertheless, current quantum computers do not allow for calculating the power flow for large systems. Therefore, it is concluded quantum computing should be used for the optimisation.

The computation time needed for the optimisation seems to grow with a cubic function. In this sense, small problems will be able to be executed in classical computers relatively easily. Nevertheless, for large networks quantum computing could be useful to reduce the total computation time. In other words, it could be benefited by the quantum speedup. Moreover, the analysis of the classical problem has shown genetic algorithms may struggle to find globally optimal solutions. In this sense, quantum computing would allow for achieving better results.

The size of the network that can be analysed would be limited by the number of qubits of the quantum computer. As it is an optimisation problem, the chosen technology to implement this into quantum computing is a quantum annealer. Also, this kind of computer allows for working with larger networks than gate computers as their number of qubits is higher. As a reference, there are quantum

annealers which have as many as 5000 qubits [69]. Moreover, Hybrids Solvers could be used to reduce the qubit requirements, solving with a quantum computer just the part of the QUBO problem which is quadratic and could benefit from quantum speedup.

The total number of qubits needed to solve the problem will also depend on the strategy chosen for its implementation in the quantum annealer. As previously mentioned, as this project is part of a larger project developed by i-DE in collaboration with Multiverse Computing, the algorithm is yet to be translated into quantum computing. Nevertheless, this technology provides many different strategies and tools that could be used to do so.

For its implementation into quantum computing, two different strategies can be followed as previously explained: a quantitative approach and a heuristic approach. Regarding the quantitative approach, the reliability function, which is the most computationally complex shows a linear behaviour when translating it into QUBO. This may make it difficult to find a quantum advantage using this approach. Nevertheless, new hypotheses could be introduced to increase the complexity and transform it into a quadratic problem, such as allowing for placing more than one battery of the same type in the same node and taking into account their interaction. The heuristic approach shows a completely different way of solving the problem, using different hypotheses to the original problem but reaching the same solution: an optimal battery network design. With this approach, it may be easier to find a quantum advantage, but it would be harder to find a benchmark to analyse the actual improvement obtained by using quantum computing.

Finally, it is important to remember quantum computing is still very new and in the process of maturation. Therefore, exploring the different uses they may have for grid optimisation can pave the way for future advancements in the application of quantum technologies in the electrical sector.

In conclusion, this problem is suitable to be solved with a quantum computer, in particular a quantum annealer, as with its cubic computation time growth it could benefit from the quantum speedup, boosting at the same time the introduction of quantum technologies into the power sector. Moreover, as genetic algorithms do not guarantee optimality, the results obtained using quantum computing are expected to improve.

11

Future Work

As previously mentioned, this project is developed in collaboration with the Spanish DSO i-DE, part of the Iberdrola group, and Multiverse Computing. In this sense, it is just the start of a larger project that will finish with the implementation of the problem in quantum computing. This will allow for comparing the results obtained with both computing paradigms, to determine whether quantum computing has provided any benefits and, therefore, is suitable for solving this kind of problem. The full strategy for the implementation is yet to be determined. Some useful tools to do so have been presented in this project. Nevertheless, other strategies must be explored to find the best approach for solving the problem.

At the moment, the project is focused on translating the optimisation part of the algorithm to quantum computing, as it is the most computationally complex. Nevertheless, as mentioned in Chapter 4, there is another function which belongs to the pre-processing which takes up a lot of the computation time. This is the power flow. Current quantum technologies are not able to solve this problem for large networks. However, in the set problem, the biggest computational burden comes from performing the power flow calculations multiple times to obtain the hourly power flow in a year. As each of the power flow calculations is completely independent, these iterations could be parallelised.

On another note, as explained in Chapter 4, two main simplifications were made in the classical algorithm. The first is related to the network partition, which is done by attending to tele-controlled switches. This may not be the best approach as one of the objectives of the algorithm is to minimise the investment and installing new tele-controlled switches is much cheaper than installing batteries. Therefore, other ways of partitioning the network should be studied. An interesting approach would be using quantum computing, in particular the energy communities previously explained.

Another one of the simplifications made is regarding the formulation of the objective function related to the investment. As previously mentioned, there are many aspects which have not been taken into account, such as the cost of other elements such as inverters or the installation and

maintenance cost, which may be lower if batteries are connected in the same place or if mobile batteries are used. This way, the formulation of this objective function should be revised.

References

[1] Ortega Manjavacas. A, "Power electronics and applications for grid control", Operation and Planning of Future Distribution Networks, Universidad Pontificia Comillas ICAI, October 2021.

[2] Aggarwal. S and Orvis. R. "Grid Flexibility: Methods for Modernizing the Power Grid", Energy Innovation, March 2016. Grid-Flexibility-report.pdf (energyinnovation.org)

[3] Reihani.E, Sepasi,S, Roose. L.R., and Matsuura. M, "Energy management at the distribution grid using a Battery Energy Storage System (BESS)", December 2015. Energy management at the distribution grid using a Battery Energy Storage System (BESS) - ScienceDirect

[4] Zeraati. M, Hamedani Golshan. M. E., and Guerrero. J.M., "Distributed Control of Battery Energy Storage Systems for Voltage Regulation in Distribution Networks With High PV Penetration," in IEEE Transactions on Smart Grid, vol. 9, no. 4, pp. 3582-3593, July 2018, doi: 10.1109/TSG.2016.2636217.

[5] Lawder, M.T., et al., "Battery Energy Storage System (BESS) and Battery Management System (BMS) for Grid-Scale Applications," in Proceedings of the IEEE, vol. 102, no. 6, pp. 1014-1030, June 2014, doi: 10.1109/JPROC.2014.231745

[6] Ahmad. S, and Asar. A.u."Reliability Enhancement of Electric Distribution Network Using Optimal Placement of Distributed Generation". Sustainability 2021, 13, 11407. https://doi.org/10.3390/su132011407

[7] Sigrit. L, "Islanded Operations and Islanding of Microgrids", Operation and Planning of Future Distribution Networks, Universidad Pontificia Comillas ICAI, February 2022.

[8] Idlbi. B, von Appen. J, Kneiske. T, and Braun. M, "Cost-Benefit Analysis of Battery Storage System for Voltage Compliance in Distribution Grids with High Distributed Generation", Energy Procedia, Vol 99, 2016. Cost-Benefit Analysis of Battery Storage System for Voltage Compliance in Distribution Grids with High Distributed Generation (sciencedirectassets.com)

[9] Choi. D, Shamim. N, Crawford. A, Huang. Q, Vartanian. C.K., Viswanathan. V.V., Paiss. M. D., Alam. M.J.E., Reed. D.M. , Sprenkle.V.L., "Li-ion battery technology for grid application", Journal of Power Sources, Vol 511, 2021. Li-ion battery technology for grid application (sciencedirectassets.com)

[10] Postigo Marcos. F, and Mateo Domingo. C, "T.3.1.2. Algoritmo que determina el tamaño y la ubicación de baterías", Instituto de Investigación Tecnológica, Universidad Pontificia Comillas ICAI, February 2022.

[11] "Iberdrola lidera el consorcio Flexener para investigar nuevas tecnologías que consoliden un sistema eléctrico 100% renovable", Iberdrola, August 2021. Iberdrola lidera el consorcio Flexener para investigar nuevas tecnologías que consoliden un sistema eléctrico 100% renovable - Iberdrola

[12] "Lithium-ion Battery Pack Prices Rise for the First Time to an Average of $151/kWh", Bloomberg NEF, December 2022. Lithium-ion Battery Pack Prices Rise for First Time to an Average of $151/kWh | BloombergNEF (bnef.com)

[13] "Trends in batteries", IEA, 2023 Trends in batteries – Global EV Outlook 2023 – Analysis - IEA

[14] Postigo Marcos. F., Mateo Domingo. C, Gómez San Román. T, "Improving distribution network resilience through automation, distributed energy resources, and undergrounding", International Journal of Electrical Power and Energy Systems, Vol 141, 2022. Improving distribution network resilience through automation, distributed energy resources, and undergrounding - ScienceDirect

[15] Abdel-Basset. M, Abdel-Fatah. L, Sangaiah. A.K. "Chapter 10 – Metaheuristic Algorithms: A Comprehensive Review", Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Intelligent Data-Centric Systems, 2018, Metaheuristic Algorithms: A Comprehensive Review - ScienceDirect

[16] Bhattacharyya. T, Chatterjee. B, Singh. P, Yoon. J, Geem. Z. W. and Sarkar. R. (2020). "Mayfly in Harmony: A New Hybrid Meta-Heuristic Feature Selection Algorithm". IEEE Access, 2020. (PDF) Mayfly in Harmony: A New Hybrid Meta-Heuristic Feature Selection Algorithm (researchgate.net)

[17] Kumar. M., Husian. M., Upreti. N. and Gupta. Deepti. "Genetic Algorithm: Review and Application", International Journal of Information Technology and Knowledge Management, Vol 2, No. 2, pp. 451-454, 2010. Genetic Algorithm: Review and Application by Manoj Kumar, Dr. Mohammad Husain, Naveen Upreti, Deepti Gupta :: SSRN

[18] Forrest. S. "Genetic Algorithms" Department of Computer Science, University of New Mexico, Albuquerque. Genetic algorithms (acm.org)

[19] Brital. A. "Genetic Algorithm Explained: Everything you need to know About Genetic Algorithm", 2021. Genetic Algorithm Explained :. Everything you need to know About... | by Anas BRITAL | Medium

[20] Grover-Silva. E., Girard. R., and Kariniotakis. G. "Optimal sizing and placement of distribution grid connected battery systems through an SOCP optimal power flow", Applied Energy, Vol 219, pp. 385-393, 2018 Optimal sizing and placement of distribution grid connected battery systems through an SOCP optimal power flow algorithm (sciencedirectassets.com)

[21] Prakash. P., Khatod. D. K. "Optimal sizing and siting techniques for distributed generation in distribution systems: A review", Renewable and Sustainable Energy Reviews, Vol 57, pp. 111-130, 2016. Optimal sizing and siting techniques for distributed generation in distribution systems_ A review (sciencedirectassets.com)

[22] Abdi. S., Afshar. K., "Application of IPSO-Monte Carlo for optimal distributed generation allocation and sizing", International Journal of Electrical Power and Energy Systems, Vol 44, pp. 786-797, 2013, Application of IPSO-Monte Carlo for optimal distributed generation allocation and sizing (sciencedirectassets.com)

[23] Kaur. S, Kumbhar. G, Sharma. J, "A MINLP technique for optimal placement of multiple DG units in distribution systems", International Journal of Electrical Power and Energy Systems, Vol 63, pp. 609-617, 2014. A MINLP technique for optimal placement of multiple DG units in distribution systems - ScienceDirect

[24] Kefayat. M., Lashkar Ara. A., Nabavi Niaki. S. A., "A hybrid of ant colony optimisation and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources", Energy Conversion and Management, Vol 92, pp. 149-161, 2015. A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources - ScienceDirect

[25] Rao Gampa. S., Jasthi. K., Goli. P., Das. D., Bnasal. R. C., "Grasshopper optimization algorithm based two-stage fuzzy multiobjective approach for optimum sizing and placement of distributed generations, shunt capacitors and electric vehicle charging stations", Journal of Energy Storage, Vol 27, 2020. Grasshopper optimization algorithm based two stage fuzzy multiobjective approach for optimum sizing and placement of distributed generations, shunt capacitors and electric vehicle charging stations (sciencedirectassets.com)

[26] Postigo Marcos. F. and Mateo Domingo. C. "T.3.1.1. Revisión del estado del arte de los mecanismos actuales de flexibilidad en la red de distribución", Instituto de Investigación Tecnológica, Universidad Pontificia Comillas ICAI, May 2021.

[27] "Quantum Computing. Progress and Prospects", National Academies of Sciences, Engineering, and Medicine. (2019). Quantum Computing: Progress and Prospects - National Academies of Sciences, Engineering, and Medicine, Division on Engineering and Physical Sciences, Intelligence Community Studies Board, Computer Science and Telecommunications Board, Committee on Technical Assessment of the Feasibility and Implications of Quantum Computing - Google Libros

[28] Microsoft. "What is quantum computing?". What is Quantum Computing | Microsoft Azure

[29] IBM, "What is Quantum Computing?", What is Quantum Computing? | IBM

[30] Roberts. J, "Quantum Computers will soon outperform classical machines", Horizon, The EU Research & Innovation Magazine, European Commission, June 2019 'Quantum computers will soon outperform classical machines' | Research and Innovation (europa.eu)

[31] Iberdrola. "All about Quantum Computing". What is quantum computing and how does it work? - Iberdrola

[32] Dargan. J, "Top 5 Quantum Programming Languages in 2022", 2022. Top 5 Quantum Programming Languages in 2022 (thequantuminsider.com)

[33] AWS, Amazon, "¿Qué es la Computación Cuántica?", ¿En qué consiste la computación cuántica? - Explicación sobre la computación cuántica - AWS (amazon.com)

[34] McKinsey Digital. "Quantum technology sees record investments, progress on talent gap", April 2023. Record investments in quantum technology | McKinsey

[35] Michielsen. K, Nocon. M, Willsch. D, Jin. F, Lippert. T and De Raedt. H, "Benchmarking gate-based quantum computers", Computer Physics Communications, Vol 220, pp. 44-55. main.pdf (sciencedirectassets.com)

[36] Quantum Zeitgeist, "Differences between quantum annealers and gate-based quantum computing", Differences Between Quantum Annealers And Gate-based Quantum Computing — Quantum Computing News And Features (quantumzeitgeist.com)

[37] Berman. G.P, Doolen G.D, Mainieri. R and Tsifrinovich.V.I, "Introduction to Quantum Computers", World Scientific, 1998. Introduction to Quantum Computers - Gennady P. Berman - Google Libros

[38] D-Wave, "What is Quantum Annealing?" What is Quantum Annealing? — D-Wave System Documentation documentation (dwavesys.com)

[39] Microsoft. "Hybrid Quantum Computing", 2023. Overview of hybrid quantum computing - Azure Quantum | Microsoft Learn

[40] Herman. A, "The Quantum Revolution Is Here, Its Name Is Hybrid", Forbes, 2022. The Quantum Revolution Is Here, Its Name Is Hybrid (forbes.com)

[41] Phillipson. K, Neumann. N and Wezeman. R, "Classification of Hybrid Quantum-Classical Computing", Computational Science - ICCS 2023, pp. 18-33, July 2023. Classification of Hybrid Quantum-Classical Computing | SpringerLink

[42] Heim. B, Soeken. M, Marshall. S, et al. "Quantum programming languages". Nat Rev Phys 2, pp. 709–722, 2020. Quantum programming languages | Nature Reviews Physics

[43] Kahane. T, "Quantum-Inspired Computing – What is it and why does it matter?" Fujitsu, 2020. Quantum-Inspired Computing – What is it and why does it matter? : FUJITSU BLOG - Global

[44] L e S Lopes. P, "Quantum Algorithms vs. Quantum-Inspired Algorithms", 2023. Quantum Algorithms vs. Quantum-Inspired Algorithms - The New Stack

[45] Hannan. M. A, Faisal. M, Pin Jern Ker, Begum. R. A, Dong. Z. Y and Zhang. C, "Review of optimal methods and algorithms for sizing energy storage systems to achieve decarbonisation in microgrid applications", Renewable and Sustainable Energy Reviews, Vol 131, 2020. Review of optimal methods and algorithms for sizing energy storage systems to achieve decarbonization in microgrid applications (sciencedirectassets.com)

[46] Feng. F, Zhou. Y and Zhang. P, "Quantum Power Flow", IEEE Transactions on Power Systems, vol 36, no.4, pp. 3810-3812, 2021. IEEE Xplore Full-Text PDF:

[47] Sævarsson. B, Chatzivasileiadis. S, Jóhannsson. H, and Østergaard. J, "Quantum computing for power flow algorithms: Testing on real quantum computers", arXiv preprint arXiv:2204.14028, 2022. 2204.14028.pdf (arxiv.org)

[48] Golestan. S, Habibi. M. R, Mousazadeh Mousavi. S. Y, Guerrero J.M, Vasquez J.C, "Quantum computation in power systems: An overview of recent advances", Energy Reports, Vol 9, pp. 584- 596, 2023. main.pdf (sciencedirectassets.com)

[49] Farhi. E, Goldstone. J and Gutmann. S, "A Quantum Approximate Optimization Algorithm", arXiv:1411.4028, 2014. 1411.4028.pdf (arxiv.org)

[50] Sreedhar. R, "QIOA: Quantum Inspired Optimisation Algorithm", Department of Microtechnology and Nanoscience, Chalmers University of Technology, 2020. QIOA: Quantum Inspired Optimisation Algorithm (chalmers.se)

[51] Ajagekar. A and You. F, "Quantum computing for energy systems optimization: Challenges and opportunities", Energy, Vol 179, pp 76-89, 2019. Quantum computing for energy systems optimization: Challenges and opportunities (sciencedirectassets.com)

[52] Harrow. A.W., Hassidim. A and Lloyd. S, "Quantum algorithm for solving linear systems of equations", arXiv:0811.3171

[53] Eskandarpour. R, Gokhale. P, Khodaei. A, Chong. F. T, Passo. A and Bahramirad. S, "Quantum Computing for Enhancing Grid Security", IEEE Transactions on Power Systems, Vol. 35, no. 5, pp. 4135-4137, 2020 IEEE Xplore Full-Text PDF:

[54] Hidary. D and Libenson. S, "The Application of QAOA on a Cloud-based Quantum Computer for Clean Energy Grid Optimization", 2020. (1) The Application of QAOA on a Cloud-based Quantum Computer for Clean Energy Grid Optimization | David Hidary and Samuel Libenson - Academia.edu

[55] Feng. F, Zhou. Y and Zhang. P, "Quantum Power Flow," IEEE Transactions on Power Systems, vol. 36, no. 4, pp. 3810-3812, July 2021 Quantum Power Flow | IEEE Journals & Magazine | IEEE Xplore

[56] Giani. A and Eldredge. Z, "Quantum Computing Opportunities in Renewable Energy". SN Computer Science, 2, no. 393, 2021. Quantum Computing Opportunities in Renewable Energy | SpringerLink

[57] Zhou. Y et al., "Quantum computing in power systems", iEnergy, vol. 1, no. 2, pp. 170-187, June 2022, IEEE Xplore Full-Text PDF:

[58] Ullah. M. H, Eskandarpour. R, Zheng. H and Khodaei. A, "Quantum computing for smart grid applications", IET Generation, Transmission & Distribution, Vol 16, 21, pp 4239-4257, 2022. Quantum computing for smart grid applications - Ullah - 2022 - IET Generation, Transmission & Distribution - Wiley Online Library

[59] Koretsky. S, Gokhale. P, Baker. J.M, Viszlai. J, Zheng. H, Gurung. N, et al, "Adapting Quantum Approximation Optimization Algorithm (QAOA) for Unit Commitment", arXiv:2110.12624

[60] Drepper. C, "E.ON allies with IBM Quantum to Advance Energy Transition Goals", E.On, 2021. E.ON allies with IBM Quantum to Advance Energy Transition Goals (eon.com)

[61] "Microsoft and DEWA bringing quantum computing to Dubai", Microsoft News Center, 2018. Microsoft and DEWA bringing quantum computing to Dubai - Stories

[62] "Los Alamos teams with Oak Ridge, EPB to demonstrate next-generation grid security tech", Los Alamos National Laboratory, 2019. Los Alamos teams with Oak Ridge, EPB to demonstrate next-generation grid security tech (lanl.gov)

[63] O'Neil. C, "Quantum Computers Can Now Interface With Power Grid Equipment", National Renewable Energy Laboratory (NREL), 2023 Quantum Computers Can Now Interface With Power Grid Equipment | News | NREL

[64] Glover. F, Kochenberger. G, and Du. Y, "Quantum Bridge Analytics I: A tutorial on Formulating and Using QUBO Models", 2019. arXiv:1811.11538

[65] Ferbández-Campoamor. M, O'Meara. C, Cortiana. G, Peric. V and Bernabé-Moreno. J, "Community Detection in Electrical Grids Using Quantum Annealing", 2021 arXiv:2112.08300

[66] Akrobotu. P. D, James. T. E, Negre. C. F. A and Mniszewski. S. M, "A QUBO formulation for top-t eigencentrality nodes", 2022. arXiv:2105.00172

[67] McCollum. J and Krauss. T, "Solving the Network Shortest Path Problem on a Quantum Annealer", IEEE Transactions on Quantum Engineering, vol. 1, pp. 1-12, 2020. Solving the Network Shortest Path Problem on a Quantum Annealer | IEEE Journals & Magazine | IEEE Xplore

[68] McCollum. J and Krauss. T, "QUBO formulations of the Longest Path Problem", Theoretical Computer Science, Vol 863, pp 86- 101, 2021. QUBO formulations of the longest path problem - ScienceDirect

[69] FORSCHUNGSZENTRUM JUELICH, "European Milestone: Quantum Computer with More Than 5000 Qubits Launched",2022 European Milestone: Quantum Computer With More Than 5,000 Qubits Launched (scitechdaily.com)

[70] Dilmegani. C, "Quantum Programming: Languages, SDKs and Algorithms in 2023", December 2022. Quantum Programming: Languages, SDKs & Algorithms in 2023 (aimultiple.com)

[71] Dilmegani. C, "Quantum Software Development Kits in 2023", December 2022. Quantum Software Development Kits in 2023 (aimultiple.com)

[72] "Digital Annealer -Quantum Computing Technology, Available Today", Fujitsu Digital Annealer: Fujitsu Global

A1

SDG Alignment

The SDGs are a compilation of seventeen universal goals set for 2030 by the United Nations. They seek to promote prosperity while safeguarding the environment. This project is aligned with one main goal, which corresponds to both the group of social and environmental goals:

- SDG 7: Affordable and Clean Energy:
  This SDG encompasses different objectives related to the energy field. On the one hand, goal 7.1 aims to "ensure universal access to affordable, reliable, and modern energy services", while objective 7.2 promotes a considerable increase in the global energy mix of the share of renewable energies. As renewable energy production is highly unpredictable and its connection to the grid as a distributed energy resource can disrupt system stability, energy storage systems such as the one designed in this project play a key part in their grid integration, providing improved control capabilities and flexibility. On the other hand, goal 7a promotes investment in energy infrastructure and clean energy technologies, among others, which perfectly aligns with the project's problem of designing a battery network.

On another note, the project could also be considered to be aligned with one of the economic goals:

- SDG 9: Industry, Innovation, and Infrastructure:
  The motto of this goal is "built resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation". Specifically, objective 9.4 aims to make the existing infrastructure more sustainable by using resources efficiently and promoting the adoption of cleaner technologies and industrial processes. As can be seen, this is also extremely related to the project's objective.