DIPLOMARBEIT

Optimales Energiemanagement für eine vernetzte lokale Energiegemeinschaft

unter Anwendung fortgeschrittener Modelle des maschinellen Lernens

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Masterstudiums

Elektrische Energietechnik und nachhaltige Energiesysteme

eingereicht von

Mikita Balbukou
Matrikelnummer: 12140071

am Institut für Energiesysteme und Elektrische Antriebe
im Forschungsbereich Energiesysteme und Netze (E370-01)
der TU Wien

Betreuung: **Univ.-Prof. DI. Dr.sc.techn. Bernd Klöckl**
Univ. Assist. Dr. Ibrahim Brahmia

Wien, am August 20, 2025

DIPLOMA THESIS

Optimal Energy Management for an Interconnected Local Energy Community

using Advanced Machine Learning Models

for the attainment of the degree

Master of Science

as part of the master studies

Electrical Power Engineering and Sustainable Energy Systems

submitted by

Mikita Balbukou
matriculation number: 12140071

at the Institute of Energy Systems and Electrical Drives
in the research area Energy Systems and Networks (E370-01)
at the TU Wien

Advisor: **Univ.-Prof. DI. Dr.sc.techn. Bernd Klöckl**
Univ. Assist. Dr. Ibrahim Brahmia

Vienna, on August 20, 2025

Kurzfassung

Der neue Trend zur Implementierung von Energiegemeinschaften bringt neue Herausforderungen für die Zuverlässigkeit der Energieversorgung mit sich. EGs nutzen hauptsächlich erneuerbare Energiequellen, wie beispielsweise Photovoltaikmodule, deren Leistung von den Umgebungsbedingungen abhängt. Während sonniger Stunden können PV-Anlagen überschüssige Energie erzeugen. Dies kann zu Überlastungen in Übertragungsleitungen und Überspannungen in Hochspannungsanlagen, Transformatoren und Generatoren führen.

Das Ziel dieser Masterarbeit ist die Entwicklung und Implementierung eines fortschrittlichen Energiemanagementsystems für lokale EGs und der Einsatz fortschrittlicher Optimierungstechniken zur Verbesserung der Netzstabilität, insbesondere zur Minimierung von Leitungsüberlastungen, zur Reduzierung von Busspannungsverletzungen und zur Optimierung der Stromverteilung.

Vier verschiedene Algorithmen wurden in der Python-Umgebung implementiert. Im ersten Schritt wird eine Monte-Carlo-Simulation angewendet, um die Systemzuverlässigkeit zu bewerten und Schwachstellen zu identifizieren. Im zweiten Schritt werden zwei fortschrittliche metaheuristische Optimierungsalgorithmen – Partikelschwarmoptimierung und Genetischer Algorithmus – eingesetzt. Diese Algorithmen führen eine Lastumverteilung durch, um optimale Lasten zu finden, unter denen das System stabil arbeitet. Im letzten Schritt wird ein Künstliches Neuronales Netz implementiert. Das KNN wird anhand eines Datensatzes aus PSO - oder GA-Ergebnissen trainiert.

Tritt im System ein Leistungsungleichgewicht oder eine Instabilität auf, kann der trainierte Algorithmus sofort neue Generatorleistungen vorhersagen, die mit der aktuellen Last im Gleichgewicht sind. Dies ermöglicht eine sofortige Wiederherstellung der Systemstabilität, ohne dass komplexe Optimierungsalgorithmen erneut ausgeführt werden müssen.

Die Ergebnisse zeigen, dass der kombinierte Ansatz die Betriebseffizienz steigert und die Systemstabilität auch bei variablen Last- und Erzeugungsbedingungen aufrechterhält. Diese Methodik bietet einen skalierbaren Rahmen für die zukünftige Entwicklung intelligenter Energiegemeinschaften. Diese Arbeit trägt zur Entwicklung intelligenter EMS-Frameworks für zukünftige dezentrale und auf erneuerbaren Energien basierende Energiesysteme bei.

Abstract

The new trend of implementing Energy Communities (ECs) brings new challenges related to the reliability of energy supply. ECs use mainly renewable energy sources (RES), such as photovoltaic panels (PV), whose output depends on environmental conditions. During sunny hours, PVs can generate excess power. It can lead to overloads in transmission lines and overvoltages in high-voltage equipment, transformers (TR), and generators.

This thesis aims to design and implement an advanced Energy Management System (EMS) for local ECs. The main objective is to use advanced optimization techniques to improve grid stability, namely minimizing line overloads, reducing bus voltage violations, and optimizing power distribution.

Four different algorithms were implemented in Python and Power Factory environments. The first stage applies a Monte Carlo simulation to assess system reliability and identify vulnerable areas. In the second stage, two advanced metaheuristic optimization algorithms, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), are used. These algorithms perform a load redispatching to find optimal loads under which the system operates stably. An Artificial Neural Network (ANN) is implemented in the final stage. The ANN is trained on a dataset obtained from PSO or GA results.

When a power imbalance between supply and demand or instability occurs, the trained algorithm can instantly predict a new set of generator outputs balanced with the current load. This allows in a short time for the restoration of system stability in a short time without rerunning complex optimization algorithms.

The results show that the combined approach enhances operational efficiency and maintains system stability, even under variable load and generation conditions. This methodology provides a scalable framework for the future development of intelligent energy communities.

This work contributes to the development of intelligent EMS frameworks for future decentralized and renewable energy systems.

Danksagung

An dieser Stelle möchte ich mich herzlich bei meinem Betreuer Herrn Ibrahim Brahmia für seine engagierte Unterstützung, seine wertvollen Anregungen sowie die kontinuierliche Begleitung während der gesamten Bearbeitungszeit dieser Diplomarbeit bedanken. Sein fachliches Wissen und seine konstruktive Rückmeldung haben maßgeblich zur erfolgreichen Umsetzung beigetragen.

Ebenso danke ich Herrn Univ.-Prof. Dipl.-Ing. Dr.sc.techn. Klöckl Bernd für seine Beteiligung an den Präsentationen und seine hilfreichen Rückmeldungen, die zur qualitativen Weiterentwicklung der Arbeit beigetragen haben.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Ich erkläre außerdem, dass ich beim Verfassen dieser Arbeit keine künstliche Intelligenz (KI) oder vergleichbare Software verwendet habe, mit Ausnahme eines wissenschaftlich anerkannten Übersetzungswerkzeugs zur Überprüfung der englischen Grammatik.

Wien, am 20. August 2025

[redacted]

Mikita Balbukou

Contents

List of Abbreviations

ANN	– Artificial Neural Network
CHP	– Combined Heat and Power
DG	– Diesel Generator
GA	– Genetic Algorithm
GHG	– Greenhouse Gas
EMS	– Energy Management System
EC	– Energy Community
FF	– Fitness Function
LFC	– Load Flow Calculation
MAE	– Mean Absolute Error
MC	– Monte Carlo
OPF	– Optimal Power Flow
PF	– DIgSILENT PowerFactory
PSO	– Particle Swarm Optimization
PV	– Photovoltaic
RES	– Renewable Energy Sources
REC	– Renewable Energy Community
TR	– Transformer

1. Introduction

Energy Communities are groups of people, companies, or local authorities that collaborate to produce and utilize electricity from renewable sources. They help reduce dependence on large energy companies and increase the use of clean energy. ECs can reduce energy costs and give local people more control over their energy supply. They also support the goals of the European Union for carbon neutrality. This section provides an overview of ECs and explains why they are becoming increasingly important. Their emergence will be discussed from two different perspectives: public and technical.

1.1. Background

Over the last decade, Europe's energy structure has undergone significant changes in response to issues of CO_2 emissions and environmental protection. The first serious step was taken in 2015 in Paris, where the so-called "Paris Agreement" was signed [1].

The Paris Agreement aims to limit global warming and combat climate change. The EU and all its member states have signed and ratified the Paris Agreement and are firmly committed to its implementation. By approving the agreement, the EU and its countries are legally bound by the goal of keeping global temperatures within safe limits.

To reach this goal, the EU has, among other things, launched the European Green Deal strategy, which puts in place measures and rules to drastically reduce emissions and transform its economy into a climate-neutral one by 2050. As required by the Paris Agreement, the EU submitted its long-term emissions reduction strategy and its updated climate plans (nationally determined contribution, NDC) before the end of 2020. The EU aims to reduce EU emissions by 2030 by at least 55% compared to 1990 levels as a step towards reaching neutrality by 2050 [2].

To achieve climate neutrality, EU countries have gradually begun to adopt the widespread use of Renewable Energy Sources (RES). To successfully and timely achieve climate goals, the number of renewable and non-dispatchable sources of electricity generation in European countries continues to increase. Furthermore, some countries, such as Austria and Iceland, aim to achieve carbon neutrality by 2040 [3].

Such goals have led to the emergence of new and flexible types of energy systems, including Virtual Power Plants (VPPs), Microgrids, Digital Grids, and Energy Communities (ECs). The present study will focus on the latter type.

In a general sense, the EC is an association of groups of citizens, companies, local businesses, and local government agencies that have decided to cooperate in producing electricity from RES and thus be able to meet their energy needs with a clean, affordable, and locally produced alternative. ECs can be viewed from both public and technical perspectives.

1.2. Energy communities from a public perspective

ECs motivate citizens to support the transition to clean energy, increasing public acceptance of renewable energy projects and promoting private investment. ECs can effectively restructure existing energy systems, empowering citizens to manage the energy transition locally and benefit from improved energy efficiency, lower bills, reduced energy poverty, and greater local opportunities for green jobs.

Recognizing the potential contribution of Energy Communities (ECs) to achieving a safer, more affordable, and cleaner energy system for Europe, the REPowerEU plan has set out a common policy objective of establishing one energy community per municipality with a population of over 10,000 by 2025 [4].

By acting as a single entity, ECs can access all relevant energy markets equally with other market participants. Under EU law, an EC can be any legal entity, including an association, cooperative, partnership, non-profit organization, or limited liability company.

Renewable ECs are a significant step towards energy independence and climate neutrality, and solving associated technical issues, such as line overload and overvoltages, is a top priority.

1.3. Energy communities from a technical perspective

ECs are groups of electricity producers and consumers, generally located in the same area, with a high share of RES. ECs can operate autonomously or remain interconnected with the main (external) grid during peak demand periods. In the present work, three different ECs with different energy sources are linked via the external grid.

ECs' key ideas are their focus on cooperation, energy sharing through transmission lines, reducing energy costs through advanced energy management, minimizing the carbon footprint through the use of RES, and improving energy independence.

However, RES highly depend on weather conditions, leading to unstable generation. Moreover, with the increased share of RES, specific technical issues can arise, such as overloads in transmission lines and voltage violations on buses.

Thus, an important task is promptly responding to changes in generation and consumption and restoring the balance between them, thereby preventing excessive current through transmission lines and other expensive equipment, such as TR or generators. As a rule, load flow calculations were performed for the traditional assessment of reliability and flexibility of the power system in worst-case scenarios. This approach has worked well in the past when variations in consumption levels were slow, and the largest part of electricity generation was dispatchable. However, introducing more non-dispatchable and variable distributed power generation into the power grid means that the parameters in the power system are no longer deterministic, but stochastic to a larger extent.

Considering the changes in the power system and the expected continuation of these changes, the European Commission recommends using probabilistic methods for load flow calculations in the electricity system. According to this report, the preferred methods are Monte Carlo (MC) methods [5].

MC simulation is a mathematical technique for predicting the possible outcomes of an uncertain event. This method analyzes past data and predicts a range of future outcomes. For the current work, MC is actively used to perform a large number of load flow calculations with different parameters of loads and generators. As a result, the most vulnerable areas of the power grid were identified.

The next stage will be to optimize the overloaded power grid and bring it to a stable state. Since the investigated model includes a large number of different electrical elements, traditional mathematical optimization techniques, such as linear programming (LP), quadratic programming (QP), and the gradient descent method (GD), may not be applicable.

So, more advanced techniques should be implemented for more complex and extended power grids, where meta-heuristic methods should be applied.

Two of the most commonly used meta-heuristic algorithms are:

- Genetic Algorithm (GA)
- Particle Swarm Optimization (PSO)

These methods enable the finding of new, optimized parameters for loads and generators, which will serve as the basis for applying an ANN. Ultimately, the ANN model was trained on this new, optimized dataset, allowing for the rapid prediction of generator setpoints and ensuring a balance between demand and supply without the need to rerun PSO or GA.

This work can also be seen as part of an EMS. An EMS is a framework used by power grid operators to monitor, control, and optimize the operation of the power system, ensuring reliability and efficiency. Current research is focusing on these goals.

All algorithms contributed to the EMS by performing load management (redispatch) and enhanced grid stability, while considering all technical system constraints.

1.4. Research questions

As the title of the thesis suggests, "Optimal Energy Management for Interconnected Local Energy Communities," the work aims to address the following central research question:

How can modern modeling, optimization, and machine learning techniques be effectively applied to improve the operational reliability and efficiency of autonomous energy communities?

To answer this question, the following sub-questions are investigated:

1. How can Monte Carlo simulation be utilized to evaluate power system reliability and pinpoint critical vulnerabilities under uncertain load and generation conditions?
2. How accurately can machine learning algorithms predict future PV generation based on historical data?
3. Which optimization method - PSO or GA - provides better performance in mitigating overloads and voltage violations? What are the comparative strengths and limitations of each?
4. To what extent can ANNs replace traditional optimization tools (such as PSO and GA) in real-time power distribution?

1.5. Overview of the thesis

To answer the above research questions, the thesis is divided into the following chapters:

Chapter 1 provides a general overview of ECs from a technical and public perspective, and also motivates the use of ECs.

Chapter 2 provides an overview of the primary optimization methods used in the work: PSO, GA, ANN, and Random Forest Regression.

Next, Chapter 3 describes the methodology. The studied energy grid is described in detail, along with all the preparatory stages for optimization. Specifically, the necessary data frames for the MC simulation are created, the parameterization of the PV panels is carried out using machine learning methods, and finally, the application of PSO, GA, and ANN is described.

Chapter 4 presents the results of the methods mentioned above. Furthermore, the calculation results are compared not only numerically but also graphically.

Chapter 5 is devoted to comparing the effectiveness of the methods used, assessing their scalability, practical potential, and identifying their limitations.

The final chapter 6 summarizes the study's results, formulates the main findings, discusses the limitations of the approaches used, and suggests directions for future work.

1.6. Nomenclature

F	Function of interest (e.g., power flow)
S_i	i-th set of input variables in MC simulation
n_S	Total number of simulations (Monte Carlo)
$f(x)$	Probability density function
μ	Mean value of the distribution
σ	Standard deviation of the distribution
λ	Rate parameter of exponential distribution
F_1	Objective function for minimizing power losses
V_i	Voltage phasor magnitude at bus i
θ_i	Voltage angle at bus i
Y_{ij}	Admittance magnitude between bus i and j
F_2	Voltage deviation objective
v_i	Per-unit voltage magnitude at bus i
NG	Number of generators
NB	Number of buses
P_{gi}	Active power generation of generator i
P_{Dj}	Active power demand at bus j
P_L	Total active power losses in the system
Q_{gi}	Reactive power generation of generator i
Q_{Dj}	Reactive power demand at bus j
Q_L	Total reactive power losses in the system
P_{gi}^{min}	Minimum active power generation of generator i
P_{gi}^{max}	Maximum active power generation of generator i
Q_{gi}^{min}	Minimum reactive power generation of generator i
Q_{gi}^{max}	Maximum reactive power generation of generator i
$\|V_i\|^{min}$	Minimum voltage magnitude at bus i
$\|V_i\|^{max}$	Maximum voltage magnitude at bus i
F	Overall objective function (fitness)
L	Line overloads (in % or MW)
T	Transformer overloads
V	Voltage deviations from nominal
S_{active}	Unbalanced active power in the system
$S_{reactive}$	Unbalanced reactive power in the system
$w_1, ..., w_5$	Weights for individual objective terms

2. State-of-the-art and progress beyond

This chapter provides an overview of the legislative documents signed at the level of EU Member States, with the primary purpose of accelerating the emergence of ECs without excessive bureaucratic obstacles. In particular, the RED I, RED II, and RED III documents will be discussed. Additionally, the European Green Deal and the REPowerEU strategy will be briefly reviewed as part of the broader EU policy aimed at reducing CO_2 emissions.

2.1. Historical background and legislative evolution of renewable energy communities

ECs represent a modern energy grid design model in which citizens, businesses, and governments cooperate to generate and exchange energy locally. These communities are typically located in one area. In traditional grid management, all energy is centralized, meaning it is supplied from a single source (e.g., a cogeneration plant). In the case of ECs, energy production is based on sustainability, participation, and decentralization principles.

This approach aims to transform passive consumers into active participants in the energy transition, helping to overcome the technical, legal, and social challenges that often hinder the large-scale deployment of renewable energy.

ECs are distinguished from other distributed generation models by their strong focus on community involvement, environmental sustainability, and local benefits, including energy independence, reduction of greenhouse gas (GHG) emissions, and social inclusion.

Although the 2015 Paris Agreement provided renewed political momentum for climate action, the first significant step towards combating GHG was taken with the ratification of the Kyoto Protocol (1997), which inspired European nations to promote decentralized energy production by supporting local cooperative projects [6]. For example, at this time, Germany adopted the Renewable Energy Act, which introduced feed-in tariffs to stimulate small-scale producers and encourage the development of local projects based on solar and wind technologies [7].

In parallel, the EU launched a standard energy policy. The Green Paper on Energy Security [8] emphasized the importance of diversifying energy sources and promoting citizen participation. These initiatives led to growing political support for renewable energy across Europe.

A significant step in the EU's development was Directive 2009/28/EC (RED I) [9], which established legally binding targets for the share of renewable energy within the EU. According to this Directive, 20% of Europe's energy consumption should be utilized by renewable energy by 2020. This directive created the foundation for integrating citizen participation models, such as ECs, into national energy frameworks.

The next step in this evolution was Directive 2018/2001/EU (RED II), which formally introduced the legal definition of renewable ECs and encouraged their integration into national energy markets. This was the culmination of legislative evolution. RED II removed numerous regulatory and technical barriers, enabling citizens to produce, consume, store, and share energy.

2.2. European regulatory framework: RED II and the Clean Energy Package

Directive 2018/2001/EU, commonly known as RED II, is a key component of the EU Clean Energy Package. This legislative package has significantly impacted the development and legal recognition of ECs across EU Member States. RED II introduces a clear legal framework for promoting and integrating renewable energy into national energy systems [10].

According to RED II, an EC is a legal entity controlled by local members whose primary objective is to generate social, environmental, or economic benefits at the community level rather than financial profit.

RED II, which entered into force in December 2018, requires Member States to achieve a minimum of 32% renewable energy in their gross final energy consumption by 2030. The directive includes several measures to achieve this goal, including explicit support for RECs. RED II focuses on three main elements:

- Self-consumption: Under the directive, citizens and communities have the right to generate, store, and share renewable energy.
- Access to networks: Member States must ensure fair and non-discriminatory access for ECs to the electricity distribution networks. This includes the obligation to remove bureaucratic and technical barriers that typically prevent small producers from participating in the energy market.
- Economic support: National governments are encouraged to promote ECs by adopting economic support schemes such as feed-in tariffs, tax reliefs, or fixed compensation models.

These elements, in general, aim to empower local actors, simplify participation in the energy transition, and accelerate sustainable development at the community level. RED II is a key policy for building a decentralized, citizen-driven energy system across Europe.

2.3. International regulatory framework: RED III

The third revision of the Renewable Energy Directive was RED III [11]. It was officially adopted in October 2023 as part of the Fit for 55 package to boost the use of sustainable energy in combating climate change. The directive updates RED II to achieve the more ambitious targets of the European Green Deal and the REPowerEU plan to reduce GHG emissions by 55% by 2030 and strengthen the EU's energy independence.

RED III sets an ambitious target: 42.5% of Europe's total energy consumption must come from renewable sources by 2030, with an additional push toward 45%. This is a significant increase compared to the 32% target set by RED II. Member states are required to update their National Energy and Climate Plans (NECPs) to align with these targets.

In addition, the Directive sets specific targets for essential sectors: First, it simplifies and speeds up the permitting process for renewable energy projects to address delays. Second, it introduces binding sub-targets for renewable hydrogen use in Industry 42% and transport 5.5% by 2030. Third, supports the development of renewable fuels of non-biological origin to bring down dependence on fossil fuels. Fourth, encourages collaboration between EU member states to meet renewable energy goals.

RED III's most notable features are simplifying and shortening administrative approval timelines for new renewable energy installations. For renewable installations located in designated "go-to" areas, the maximum permit duration is 12 months, while for projects outside these zones, the limit is 24 months. These measures were created to eliminate unnecessary bureaucracy and facilitate the rapid growth of renewable energy infrastructure.

Furthermore, RED III strengthens the role of ECs by recognizing them as central actors in the decentralized energy transition. By promoting distributed energy generation, particularly through PV, wind, and biomass installations at a local scale, ECs can help achieve energy autonomy and support the EU's climate neutrality goals.

The Directive is already influencing progress in countries such as Germany and Denmark, where people have already participated in the energy market through energy cooperatives and state incentives.

2.4. The Austrian EAG and Its Role in Energy Communities

The Renewable Energy Expansion Act - EAG (Erneuerbaren-Ausbau-Gesetz) was created to support green energy systems in Austria. This Act is a new law that aims to increase the share of renewable energy in Austria's electricity generation to 100% by 2030 and to achieve climate neutrality in the country by 2040. The EAG regulates key issues related to the promotion of renewable energy and is thus the successor to the Green Electricity Act (ÖSG, also known as ÖkostromG).

The EAG enables the creation of **energy communities**. For example, people from all over Austria can join to form an energy community and thus share the energy they generate.

To achieve the target of 100% renewable energy in electricity production, the Austrian Ministry for Climate Protection, Environment, Energy, Mobility, Innovation and Technology has set the goal of increasing electricity production from renewable sources by 27 terawatt hours (TWh) by 2030, taking into account the projected electricity consumption. This increase in green electricity production must be achieved under strict environmental criteria. For context: Austria's total electricity consumption in 2021 was around 72 TWh [12].

The planned increase of 27 TWh of green electricity is to be distributed among the individual renewable energy technologies as follows:

- Photovoltaics: 11 TWh
- Wind power: 10 TWh
- Hydropower: 5 TWh
- Biomass: 1 TWh

The EAG aims to achieve the desired increase of a total of 27 TWh of green electricity through various, partly new, instruments [12]:

1. Promoting the expansion of renewable energy through investment grants
2. Promoting the expansion of renewable energy through a market premium model
3. Promoting energy communities

The study will only consider the fourth EAG instrument - promoting energy communities.

With the introduction of the EAG, the legislature is also promoting the formation of energy communities for the first time. These communities, usually at the local level, strive to achieve the highest possible utilization of renewable energies within their membership. They can also store surplus green electricity or divert it to higher-level grids and sell it. Energy communities can receive a market premium for surplus green electricity—that is, electricity fed into the general grid rather than consumed by the energy community—as long as the share of surplus electricity does not exceed 50% of the electricity produced within the energy community [12].

In an energy community, different actors can come together and share the energy they generate; they can consume it jointly, store it, and/or sell it. A built-in smart meter and the measurement and storage of quarter-hourly values (opt-in) are required for billing within the energy community. A contractual/membership relationship exists between you and the energy community.

There are two types of ECs: renewable and citizen energy communities. This study reveals the significance of Citizen Energy Communities. According to [13]: The concept of an energy community can also be implemented beyond the local or regional level, even beyond the boundaries of a grid area; in this case, a **Citizen Energy Community** is established. It can have participants in different grid areas. It may only generate, share, store, consume, or sell electrical energy, and this electricity does not have to be generated exclusively from RES. Since there is no geographical proximity here and, in extreme cases, the energy must be transported across Austria, there are no reduced grid tariffs – the normal grid fees apply, just as when purchasing from a supplier. The renewable energy subsidy must also be paid for the electricity shared in the citizen energy community.

This legal context provides the basis for modeling the coordinated energy distribution between different actors. In this thesis, such coordination is realized by modeling three ECs with centralized optimization mechanisms. These actors do not represent formal legal ECs, but are technically inspired by the citizen ECs structure as authorized by the EAG.

Thus, while the legal framework defines the boundaries and rights of the ECs in Austria, this study examines their technical and operational potential in the modeled microgrid structures.

2.5. Clarification of Framework Assumptions

In this thesis, the concept of interconnected energy systems is inspired by Energy Communities as promoted in the European legislative context (e.g., RED II and RED III directives). However, it is essential to emphasize that the presented model does not imply mandatory compliance with the European Union's legal rules or regulatory requirements. Instead, the study considers **private microgrids** operating in coordination, to evaluate the potential of optimization algorithms under defined supply conditions, rather than reproducing the regulatory environment.

The model assumes a purely technical structure in which three systems are interconnected via an external grid and exchange electricity in a way that resembles the behavior of an energy community.

This clarification is included to ensure the reader does not misinterpret the simulation as a regulatory implementation. References to RED I, II, III, and the EC framework illustrate the relevance and possible application areas of the studied optimization methods, not to imply legal enforcement.

2.6. Methodology: Monte Carlo Simulation, PSO, GA, and ANN optimization algorithms

In this section, a planning strategy based on a stochastic optimization method is proposed to solve optimal power management problems, such as improving total cost, minimizing total power losses, reducing voltage violations, and reducing overloads. Two optimization algorithms, PSO and GA, are reviewed and compared.

2.6.1. Monte Carlo Simulation

The output can be regulated and controlled by traditional generation, such as from a combined heat and power (CHP) plant, representing a deterministic value. Therefore, load flow calculations can be performed using standard methods, including the Newton-Raphson, Gauss-Seidel, or Fast Decoupled Load Flow (FDLF) [14].

However, as mentioned earlier, ECs contain a high share of RES, the generation of which directly depends on weather conditions, such as sunlight and wind. Unlike CHP plants, the output from RES is stochastic in nature. Moreover, with an increase in the number of RES, the behavior becomes increasingly probabilistic and less predictable.

For performing load flow calculations of such networks, the EU Commission proposes [5] the use of probabilistic methods for load flow analysis, particularly MC simulation. This method has found wide application in many fields of science. In the context of power systems, especially with the growing deployment of RES, MC is used to simulate and evaluate the likelihood of different operational outcomes under uncertainty.

Monte Carlo simulation is a mathematical technique for predicting the possible outcomes of uncertain events by running many scenarios. It involves repeatedly performing load flow calculations in power systems, each time with different randomized input parameters such as RES generation and load demand. Thus, MC simulation does not replace traditional power flow methods; instead, it builds upon them by applying them iteratively under stochastic input conditions to evaluate the system's behavior under uncertainty. Each iteration was based on a pre-calculated mean value μ and the standard deviation σ [15].

In this project, instead of historical data, a dataset with six pre-calculated different scenarios was used. Each scenario has different initial parameters for generators and loads. For example, Scenario 1 represents 100% base load and generation, Scenario 2 reduced both to 70%. Scenario 3 -increases to 110%, and so on. The data frame created in this way considers the probabilistic distribution of generation and load and serves as the input basis for the MC simulation.

The number of MC iterations is independent of the system size. The stopping criterion is usually a fixed number of iterations. In the project, 1000 iterations were performed. The primary objective of the MC calculation was to assess the system's stability, flexibility, and reliability, and to determine which lines are most susceptible to overvoltage or voltage violations under uncertain operating conditions.

2.6.2. Overview of Particle Swarm Optimization (PSO)

After identifying unstable elements in the power grid, which consists of three interconnected ECs, the system must be optimized to meet key technical objectives: minimizing line overloading, reducing voltage violations on buses, and improving overall system stability. The optimal power flow (OPF) is the primary and most effective method for operating and managing the electrical network.

Unlike standard load flow analysis, which determines the system's state based on given inputs (generation, load, etc.), OPF adds intelligence and has an optimization objective [16]. It calculates the "best possible" values to optimize the objective function, such as minimizing total power losses or reducing total voltage violations, taking into account all technical constraints and network features.

There are different mathematical techniques for solving the OPF problem. For example, linear programming (LP), quadratic programming (QP), the gradient descent method (GM), and the Newton method can be implemented. However, they have limitations [17]:

- They can be used for small or simplified power systems. Real-world systems are too complex for them.
- These methods are designed for continuous objective functions. So, they may fail if the problem contains discrete or non-continuous values.
- They require smooth (differentiable) functions. For example, Newton's Method requires smooth functions; otherwise, the method cannot find an optimal solution.

To conduct OPF calculations for modern, complex, and nonlinear power systems, meta-heuristic algorithms should be used. Two of the most commonly used meta-heuristic algorithms are: Particle Swarm Optimization (PSO) and Genetic Algorithm (GA).

PSO is initially inspired by nature. This method is based on how bird flocks or insect swarms move in groups. The primary property of such populations is the exchange of information. If one participant in this group finds a better point (e.g., a food source), the entire group moves in this direction.

In the context of power systems, each "particle" in PSO represents a possible candidate solution – a unique set of parameters such as active power generation, reactive power, and load reduction factors. The algorithm iterates through all possible sets of system parameters under which the power grid model can reach a stable and optimal state.

The quality of each solution is evaluated using a fitness function (FF). The FF evaluates how good the current solution is. In this project, the FF penalizes [18]:

- Line overloads
- TR or generator overloading,
- Voltage violations beyond allowed limits.

The optimization process continues until convergence, when the value of the FF becomes smaller and new iterations do not change its value. At this point, PSO has found a new set of optimized parameters under which the power system operates stably without any technical issues.

These parameters are then used in a final load flow calculation, which confirms that overloads and voltage violations have been successfully mitigated.

2.6.3. Overview of Genetic Algorithm (GA)

While the previous algorithm, PSO, was inspired by swarm behavior in nature, GA is based on the principles of biology, particularly genetics. GA describes its operations using biological terminology such as chromosomes, genes, and mutations [19].

The initial population with a certain number of individuals (called chromosomes) is created as the first step. In the context of power systems, each "chromosome" in GA represents a possible candidate solution—a unique set of parameters, such as active power generation, reactive power, and load reduction factors.

Each chromosome consists of multiple genes. Genes are properties of chromosomes. In the context of power systems, genes refer to the parameters of electrical elements, such as the active and reactive power of generators or loads.

If the value of FF is high, the power grid should be optimized. The population undergoes a process of selection, crossover (recombination), and mutation, where two parent chromosomes combine to produce a new offspring. In the recombination process, each new generation becomes better than each previous one, since only the best characteristics are taken from the parent chromosomes. This process is repeated many times, and GA gradually improves the final solution.

The quality of each solution is evaluated using an FF, as described in the PSO section, which penalizes line overloads, voltage violations, and instability. The optimization continues until convergence - that is, when the value of the FF becomes smaller and new iterations do not change its value.

In the study, the **best1bin** differential evolution strategy was chosen for crossover. According to the [20] the 'best1bin' strategy is a good starting point for many systems. This approach ensures that the best solution is always used to generate new candidates, thereby accelerating convergence and providing more stable results than purely random strategies.

2.6.4. Comparison of PSO and GA

Both GA and PSO are meta-heuristic optimization techniques, but they differ in their concepts: PSO represents swarm intelligence, where particles update their positions based on individual and global best solutions.GA represents evolutionary principles (crossover and mutation) to generate improved offspring.

GA does not require the function to be continuous or differentiable because it explores many possible solutions randomly and through evolutionary operations (selection, crossover, mutation). In contrast, PSO updates particles based on gradient directions and is better suited for smooth and continuous functions.

Looking ahead, here are some important differences between the two algorithms: PSO typically converges faster (around 10 seconds) but may become stuck in local optima. At the same time, GA is better suited for global optimization, but can take significantly longer to compute, especially for 200 iterations. As will be seen in Chapter 4.1, GA found a better FF value of **2511**, compared to **3569**, indicating that it optimized the system better. This shows that both algorithms have reached the global minimum, and the difference in results is minimal, which means both methods work about equally well.

2.6.5. Application of Artificial Neural Network (ANN)

The final stage of the study involves the application of an artificial neural network (ANN) model [21], [22]. This algorithm serves as an add-on to the previously conducted PSO and GA optimizations. The ANN is trained using optimized data obtained from the PSO results.

The ANN is a non-parametric model, meaning it does not rely on predefined mathematical equations or system-specific parameters. Instead, it learns to approximate the input-output relationship directly from data samples. Once trained, it can generalize to new, unknown input conditions and provide fast predictions without additional system-level computation.

In this work, the ANN was trained on a dataset consisting of 400 samples, and then tested on 100 samples to evaluate its predictive performance. The model demonstrated a low Mean

Absolute Error (MAE), indicating a high level of accuracy.

The key advantage of ANN is its ability to perform calculations almost instantly if the load changes and the balance between demand and generation is disturbed.

This eliminates the need to re-run computationally and time-intensive metaheuristic optimization algorithms such as PSO or GA each time the load profile changes. It should be noted that the ANN model demonstrates high accuracy only within the range of conditions for which it was trained. The MAE (Mean Absolute Error) value remains low for small load fluctuations within 5–10%.

However, in unusual or critical situations—for example, when the load changes by more than ±30–50%—the neural network may produce incorrect values of generator output because such atypical values are beyond the trained sample. With deeper fluctuations, it is necessary to repeat the meta-heuristic calculations and retrain the ANN model.

3. Research Methodology and Technical Design

This chapter presents a sequential description of the project, including all calculations. The analysis and justification of the electrical experimental models used are given, and their verification is also carried out. Each mathematical algorithm used in the work is explained in more detail using flowcharts.

3.1. Overview of the Methodological Approach

The project begins by analyzing a simplified power system consisting of a single EC. As the study progresses, this model becomes more complex, increasing to three interconnected ECs with numerous electrical elements, including RES. The corresponding extended power grid model is shown in Appendix A.1.

All simulations and analyses were carried out in two software environments:

- DIgSILENT PowerFactory (PF) – a modern software for modeling electrical systems of any complexity
- Python – used for automation, optimization, and machine learning tasks.

In some stages, Microsoft Excel was also used to visualize the results.

Using the special library “powerfactory” [23], Python gains access to PF and expands its original capabilities. For example, as described above, 1000 iterations were calculated as part of the MC simulation. With Python, this was performed instantly. However, manually calculating so many iterations without Python would be extremely difficult and time-consuming. This method would not find application at real energy facilities, where it is necessary to perform calculations and optimize the network quickly.

Initially, a single-EC model was used to set up and verify the Python code. After verification, the model was expanded to three ECs, interconnected through an external grid. When connecting the three models, technical issues emerged, including overloads on the lines and voltage violations on buses. So, the optimization was necessary.

Meta-heuristic optimization methods—PSO and GA—were implemented to optimize the power grid. The final stage involved the use of a trained ANN, which enabled it to respond almost instantly to any change in load and restore the energy balance to the system. These methods will be described in detail in the following sections.

3.1.1. Difference from Existing Work

To better position the methodology applied in this work, a comparison is made with the 2024 Master’s thesis by Florian Strebl [24], which follows a different modeling approach to energy communities.

The main difference between the current approach and that of Florian Strebl lies in the methodological direction and focus. Florian Strebl investigates the impact of **large-scale**

deployment of energy communities on distribution networks using deterministic scenarios based on the Austrian and European legislative framework. A wide range of predefined configurations was created that combine different levels of urbanization and penetration rates of flexible resources, such as photovoltaics, batteries, electric vehicles, and heat pumps. Each scenario is optimized using `Femto.jl` with two objectives: minimizing individual costs (**"No Community"**) and optimizing costs at the community scale (**"Community"**). These optimized results are then analyzed using static load flow calculations in DIgSILENT PowerFactory for both low-voltage and medium-voltage networks. The main findings indicate that the impact of energy communities is greater in less urban areas and with higher levels of flexibility. Peer-to-peer trading within communities has only a limited effect due to economic constraints. Overall, communities can reduce residual load and improve self-consumption, but these effects are modest and context-dependent. Standard key performance indicators (KPIs), including self-sufficiency level, self-consumption level, transformer loading, voltage levels, and line utilization, were used to assess system behavior.

However, Streble's approach is limited by the lack of stochastic modeling and does not account for uncertainty, variability, or system behavior under stress. The analysis focuses on fixed assumptions and does not account for the unpredictability of the real world.

In contrast, the current work considers the stochastic nature of loads (utilizing a Monte Carlo Simulation). It aims to optimize the system at various load levels using modern machine learning techniques, including PSO, GA, and ANN.

3.2. Technical overview of employment Algorithms

The theoretical basis explaining the nature of MC Simulation, PSO, GA, and ANN is provided in Section 2.6. Next, the mathematical part will be considered.

3.2.1. Monte Carlo Simulation

Monte Carlo Simulation is widely used in power system studies due to the increasing presence of stochastic variables, such as renewable energy sources and varying loads. This numerical method is based on repeated random sampling, particularly useful for solving complex and nonlinear problems where analytical solutions are not readily available. The main idea is to perform multiple simulations by randomly varying the input parameters according to predefined probability distributions, and then analyzing the outcomes statistically [25, 26].

In the context of European power system planning, the application of probabilistic models, particularly Monte Carlo simulation, has been recommended for adequacy assessments [27].

It can use both historical and simulated data as input values for loads and generation in the power system. The input values are represented as distributions from which variables are randomly selected. The system is solved multiple times, each time for a different set of random variables. The results are then processed together as probability distributions describing the possibility of possible load flows [25].

The estimated expected value of a quantity $\hat{E}(F)$ can be computed as follows [14]:

$$\hat{E}(F) = \frac{1}{n_S} \sum_{i=1}^{n_S} F(S_i) \tag{3.1}$$

Here, F is the function of interest, S_i is the i^{th} set of input variables, and n_S is the total number of simulations.

In power system studies, MC Simulation is applied to nonlinear power flow equations, allowing the estimation of the probability distribution of power flows across transmission lines. Unlike deterministic approaches, MC Simulation enables the use of full nonlinear equations without requiring simplification. However, this comes at the cost of increased computational effort due to the large number of repeated simulations.

In probabilistic load flow studies, power system loads are typically modeled using a normal (Gaussian) distribution [28, 29]. The corresponding probability density function is given by:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.2}$$

where μ represents the mean value of the distribution, and σ^2 is the variance. As will be indicated further, within the framework of this work, these values will be calculated automatically using special Python libraries. The variable x may, for instance, denote the power demand at a particular bus.

In this study, the stochastic modeling of generator outputs is based on a predefined set of six deterministic scenarios (Figure 3.7) representing different operating levels of nominal capacity. These six values were used to compute the mean (μ) and standard deviation (σ) for each generator.

Subsequently, a normal (Gaussian) distribution $f(x)$ was fitted to this synthetic dataset, and 1000 random samples were drawn using the NumPy library. These samples served as input to the Monte Carlo simulation, providing variability in generation across iterations.

Although no outlier filtering was applied in this work, it is common in probabilistic modeling to limit the influence of the distribution tails. A typical approach is to constrain the values within the interval $[\mu - 3\sigma, \mu + 3\sigma]$, which covers 99.7% of all possible outcomes. This helps exclude extreme values that may not be physically feasible, especially when simulating critical infrastructure such as power systems.

Such truncation can be implemented programmatically using functions like `scipy.stats.truncnorm` from the SciPy library [30], which allows sampling from a normal distribution with specified lower and upper bounds. While the synthetic nature of the input data in this study already implies a bounded behavior, tail clipping may be added in future implementations to ensure numerical stability and realism in the generated samples.

3.2.2. Overview of Particle Swarm Optimization

The objective function of the PSO optimization aims to minimize voltage violations on buses, reduce losses, and mitigate overloads in transmission lines. The details of the objective function are provided below. The mathematical formulation used in this study is directly adapted from the approach proposed by Singh et al. [18], where the PSO algorithm was successfully applied to the IEEE 9-bus system. Their work served as a foundation for adapting the mathematical formulation to the system considered in this study.

Minimization of Transmission Losses

The active power losses in transmission lines can be calculated using the following equation:

$$F_1 = \sum_{i=1}^{NB} \sum_{\substack{j=1 \\ j\neq i}}^{NB} |V_i||V_j||Y_{ij}| \cos(\theta_{ij} + \delta_j - \delta_i) \tag{3.3}$$

where:

- $|V_i|\angle\delta_i$ is the voltage phasor in bus i,
- $Y_{ij}\angle\theta_{ij}$ is the admittance between bus i and j.

Voltage Deviation Minimization

To maintain the voltage profile close to nominal values (1.0 p.u.), the voltage deviation objective is expressed as:

$$F_2 = \sum_{i=1}^{N_L} |v_i - 1| \tag{3.4}$$

where v_i is the per-unit voltage magnitude at bus i, and N_L is the total number of buses considered in the voltage deviation calculation.

Constraints

The following restrictions must be considered when calculating the OPF:

Active Power Balance

The total active power supply should be equal to the power demand and losses:

$$\sum_{i=1}^{NG} P_{gi} - \left(\sum_{j=1}^{NB} P_{Dj} + P_L \right) = 0 \tag{3.5}$$

where, P_{Dj} is the active power demand and P_{L} is the active power losses.

Reactive Power Balance

$$\sum_{i=1}^{NG} Q_{gi} - \left(\sum_{j=1}^{NB} Q_{Dj} + Q_L \right) = 0 \tag{3.6}$$

where, Q_{Dj} is the reactive power demand and Q_{L} is the reactive power losses.

Losses

Active power demand and active power losses are given as:

$$P_L = \sum_{i=1}^{NB} \sum_{\substack{j=1 \\ j\neq i}}^{NB} |V_i||V_j||Y_{ij}| \cos(\theta_{ij} + \delta_j - \delta_i) \tag{3.7}$$

Reactive power demand and reactive power losses are given as:

$$Q_L = \sum_{i=1}^{NB} \sum_{\substack{j=1 \\ j\neq i}}^{NB} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} + \delta_j - \delta_i) \tag{3.8}$$

Operation limits

The active and reactive generation of all generators should be between the minimum and maximum values:

$$P_{gi}^{\min} \leq P_{gi} \leq P_{gi}^{\max}, \quad (i = 1, 2, ..., NG) \tag{3.9}$$

$$Q_{gi}^{\min} \leq Q_{gi} \leq Q_{gi}^{\max}, \quad (i = 1, 2, ..., NG) \tag{3.10}$$

Voltage limits

The voltage at all buses must be within the minimum limit and maximum limit.

$$|V_i^{\min}| \leq |V_i| \leq |V_i^{\max}|, \quad (i = 1, 2, ..., NB) \tag{3.11}$$

Objective function

The objective function (fitness) used in this study is designed to penalize several undesired conditions in the power system:

$$F = w_1 \cdot L + w_2 \cdot T + w_3 \cdot V + w_4 \cdot S_{\text{active}} + w_5 \cdot S_{\text{reactive}} \tag{3.12}$$

It represents a weighted sum of five components, each reflecting a specific network issue.

- L – total line overloads (in percent or MW),
- T – transformer overloads,
- V – voltage violations at buses (deviation from nominal value),
- S_{active} – unbalanced active power in the system,
- S_{reactive} – unbalanced reactive power in the system.

The weights w_1, w_2, w_3, w_4, w_5 [31] allow prioritizing the severity of each objective. In this work, the values used were:

$$w_1 = 10, \quad w_2 = 5, \quad w_3 = 8, \quad w_4 = 100, \quad w_5 = 100$$

This parametrization ensures that the optimization algorithm prioritizes balancing active and reactive power while also considering overloads and voltage profiles.

Algorithm for solution technique

Based on the flowchart shown in Figure 3.1, the standard GA is governed by the following phases:

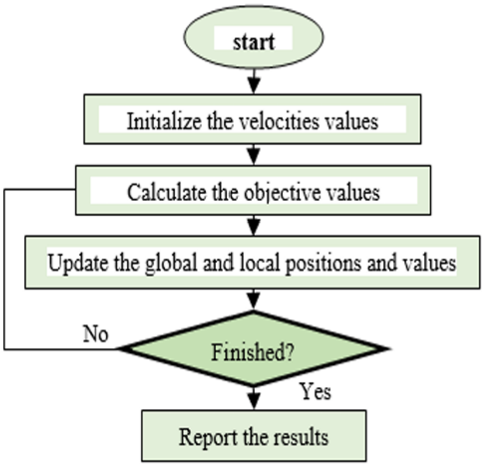

start
Initialize the velocities values
Calculate the objective values
Update the global and local positions and values
Finished?
No
Yes
Report the results

Figure 3.1.: Flowchart of PSO [17]

The PSO algorithm employed in this study follows a standard iterative procedure and comprises the following steps:

1. **Initialization**: Random initialization of particle velocities and positions within the defined search space.

2. **Velocities values:**

   In the PSO, each particle represents a candidate solution and is characterized by two vectors: a position vector x_i and a velocity vector v_i. The **velocity initialization** step refers to the random assignment of initial velocity values to all particles. These values determine the direction and speed at which each particle will explore the solution space. The particles' positions and velocities are then updated iteratively using the following standard equations [17]:

   $$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.13}$$

   $$v_i(t) = v_i(t-1) + c_1 r_1 [lb(t) - x_i(t-1)] + c_2 r_2 [gb(t) - x_i(t-1)] \tag{3.14}$$

   where:

   - $x_i(t)$, $v_i(t)$ are the position and velocity of particle i at iteration t,
   - $lb(t)$ is the best position found so far by particle i (local best),
   - $gb(t)$ is the best position found among all particles (global best),
   - $r_1, r_2 \in [0, 1]$ are uniformly distributed random numbers,
   - c_1, c_2 are acceleration coefficients.

   This initialization ensures diversity in the swarm and enables the algorithm to begin exploring the solution space effectively.

3. **Objective Function Evaluation**: Calculation of the objective (fitness) function for each particle.

4. **Update Step**: Updating both global and local best positions and fitness values based on performance.

5. **Convergence Check**: If the stopping condition (maximum number of iterations or sufficient fitness level) is met, the algorithm terminates.

6. **Output**: The best-found solution is reported.

This procedure was implemented in Python using a custom iterative loop. The particle update logic, boundary handling, and fitness evaluation were written manually, with support from a specialized PSO library for vectorized updates. Full implementation details are provided in Section 3.5.1.

This algorithm structure is adapted from [17] and applied to the power flow optimization in this study.

3.2.3. Overview of Genetic Algorithm

The second meta-heuristic algorithm applied in this study is the GA, which is inspired by the principles of natural selection and biological evolution. GA is particularly well-suited for solving nonlinear and complex optimization problems in power and energy systems.

For the GA, the objective function and system constraints remain unchanged, as previously defined in the PSO section.

The algorithmic framework used in this study is based on the approach in [32], which has shown successful application to benchmark power systems such as IEEE 6-bus, 14-bus, and 30-bus networks. In this study, the same methodology is adapted and applied to the developed ECs using a built-in Python library.

As recommended in [17], the standard GA procedure consists of five key phases, each of which is described below.

Algorithm for solution technique

Based on the flowchart shown in Figure 3.2, the standard GA is governed by the following phases:

1. **Generate an initial random solution**. A population of candidate solutions (individuals) is randomly generated. Each individual is represented as a chromosome - a vector containing potential solutions (data sets) for the electrical system.

2. **Evaluate the fitness function**. The objective function (e.g., minimizing active power loss, minimizing overload, achieving power balance between supply and demand) is calculated for each individual, determining how "fit" or suitable the solution is. Individuals with better performance have a higher chance of being selected.

3. **Perform the selection**. Based on the fitness values, the best chromosomes (sets of parameters) are selected.

4. **Application of genetic operators, such as crossover and mutation**. The best chromosomes are selected for the selection, and they undergo crossover, where they combine to create a new offspring. The next step is mutation, which introduces small random changes to add diversity, essential for exploring new regions of the search space. Moreover, additional diversity is essential for avoiding local minima and improving global optimization.

5. **Verification of the stopping criteria**. The algorithm iteratively evolves the population until a stopping criterion is met, such as a maximum number of generations or the convergence of fitness values.

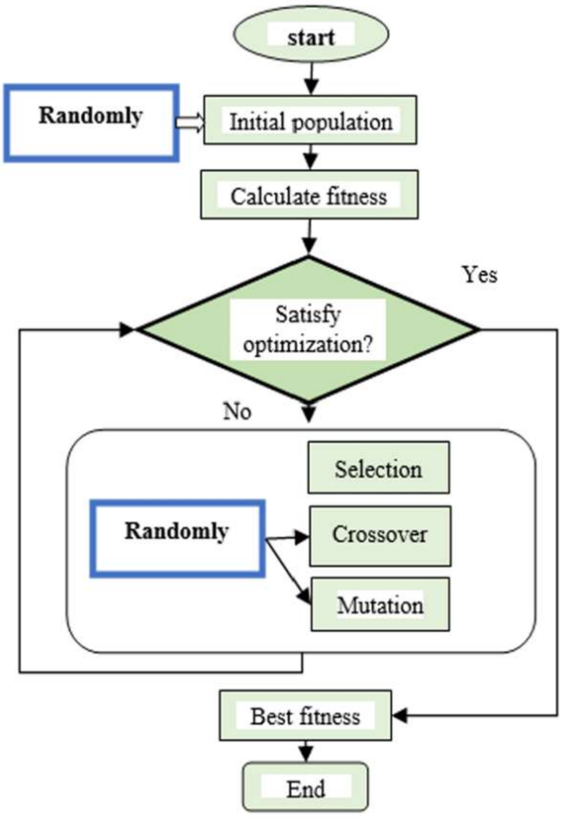


Figure 3.2.: Flowchart of GA [17]

A detailed explanation of the Python implementation will be presented in Section 3.5.2.

3.2.4. Overview of Artificial Neural Network

ANN was used as the final stage of the research, based on [22].

ANN are computational models that emulate the parallel processing capabilities of the human brain. ANNs process input data, learn patterns, and make predictions or classifications based on the knowledge they acquire.

An ANN consists of interconnected nodes, or artificial neurons, organized into layers. Each neuron receives inputs, processes them, and produces an output. The functioning of ANNs imitates how the human brain processes information. Just like biological synapses connect neurons in the brain, artificial neurons are linked through weights. These weights can be strengthened or weakened during training, similar to how synaptic strength changes in biological learning.

The artificial neuron model typically comprises three core components: input weights, a summation function, and an activation (or transfer) function. Collectively, these elements enable the network to compute weighted sums and apply nonlinear transformations to generate the desired output.

The architecture employed corresponds to the well-known Multilayer Perceptron (MLP), as illustrated in Figure 3.3. This MLP model has proven effective in capturing various types of nonlinear relationships in energy systems [33].

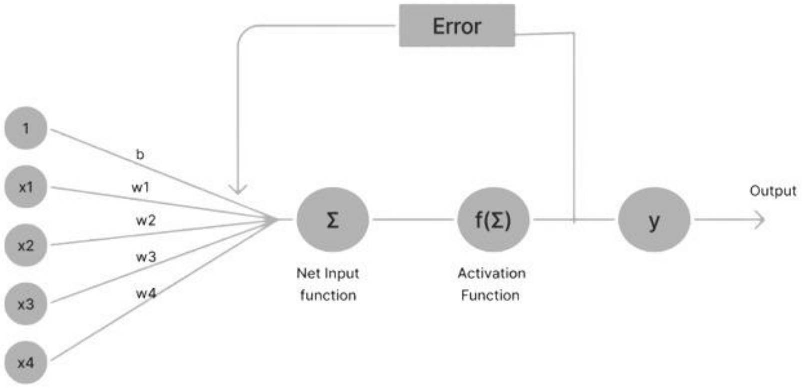


Figure 3.3.: Flowchart of ANN

In this study, the *input weights*, *summation*, and *activation functions* are automatically initialized and optimized using a dedicated Python library.

This study employs an ANN for regression purposes, specifically to predict the active power output $P_{\mathbf{gen}}$. The implementation relies on the **MLPRegressor** class from the scikit-learn Python library. The model structure, including the choice of activation functions, hidden layers, training iterations, and error evaluation, is guided by both the official documentation and example implementations from the open-source community:

- Official scikit-learn documentation [34].
- GitHub source of scikit-learn documentation [35].

These resources formed the basis for the ANN implementation in Python, which was utilized in this work. A detailed explanation of the code will be presented in Section 3.5.3.

3.2.5. Parametrization of PV installations

Parameterization of PV panels also requires a correct approach. One option for parameterizing in PF is to specify active powers manually.

To enhance the modeling's reliability, the active power values were based on predictions from historical PV generation data instead of being set arbitrarily. The active power data for 2023 in Austria (8760 hours) are utilized as input data. This data was sourced from the official ENTSO-E (European Network of Transmission System Operators for Electricity) website [36].

Initially, a *simple linear regression* was used to predict future active power values. However, this method showed a low level of correlation between actual and predicted values, which is also confirmed by unsatisfactory quality metrics - a low determination coefficient R^2_{test}, which indicates that the model cannot capture patterns.

Later, a more powerful algorithm, the *Random Forest Regression*, was selected, which ensured high prediction accuracy. The model reproduced the active power values with minimal deviations between the training and test samples, demonstrating a high level of correlation. A detailed explanation of the code will be presented in Section 3.4.1.

The linear regression model was implemented in Python, utilizing publicly available examples and documentation. In particular, the structure of the model and syntax were guided by:

- The official documentation of the LinearRegression class from the scikit-learn library [37].
- A practical Python example of linear regression published on GeeksforGeeks [38].

These sources provided step-by-step guidance for model definition, training, and performance evaluation using metrics such as R^2_{test} and MSE.

3.2.6. Comparative approach and modeling setup

Later in this chapter, a detailed review of the above methods and their implementation in Python and PF will be given.

A simplified EC is first modeled to verify the Python code at the first stage. Then the model is expanded to three ECs connected via the external network, including RES and diesel generators. Each model is first modeled without optimization (base case) and then optimized separately using PSO and GA optimizations. The last stage is the application of the ANN algorithm based on PSO results.

In critical operating modes, the optimization algorithms redistribute the load, considering the available capacity, reducing the overload on all network elements. In particular, one of the specified conditions is to limit the generator load to no more than 80% of their rated capacity, which avoids their operation in overload mode.

Running a diesel generator at a consistent and appropriate load level can significantly improve efficiency. Diesel generators often operate most efficiently at a load of around 80% of their rated capacity. Avoid overloading or underloading the generator, as this can lead to inefficiencies and shorten its lifespan [39]. Moreover, operating a diesel generator at 80% can lead to the most efficient fuel consumption. Running the generator at very low loads can increase fuel consumption per unit of electricity generated, making the operation less efficient [40]. Furthermore, continuously operating at or near 100% can also lead to overheating, accelerated wear, and, in extreme cases, equipment failure. This risk is especially pronounced in regions with high ambient temperatures—such as the Middle East—where cooling systems may become insufficient under peak loads. Therefore, many grid operators apply an operational limit of approximately 80% of the nominal power to ensure long-term reliability and thermal stability of the generation units.

The objective of this comparison is to analyze how optimization algorithms affect:

- reduction of voltage violations on buses;
- efficiency in reducing line loads;
- efficiency in reducing generator and transformer loads;
- balance of active and reactive power.

It is important to note that the total volume of generated energy from RES remains unchanged in all scenarios. Optimization only redistributes the internal load and generation to improve system operation.

3.3. Initial Power System Modeling and Model Verification

The first stage of the research is the verification of the Python code. The simulation model is based on the CIGRE medium-voltage reference network, which is included as a predefined example in the DIgSILENT PowerFactory [41]. Figure 3.4 and figure 3.5 show the initial simplified model of one EC before and after Load Flow calculation. Table 3.1 represents the quantity of electrical elements in the initial EC model. It contains:

Element	**Quantity**
Diesel generators (DG)	3
Transformers (TR)	5
Loads	11
Buses	18
External grid	1

Table 3.1.: Components of the initial EC model

The parameters of the elements are given in Table 3.2, 3.3, 3.4, 3.5:

#	Name	Active Power [MW]	Reactive Power [MVar]	Apparent Power [MVA]
1	Diesel Generator 1	3.892	0.0	3.892
2	Diesel Generator 2	2.048	0.0	2.048
3	Diesel Generator 3	2.048	0.0	2.048

Table 3.2.: Power parameters of diesel generators in the initial EC model

#	Name	Active Power [MW]	Reactive Power [MVar]	Apparent Power [MVA]	Power Factor
1	Load_1	1.401	0.5	1.5	0.949
2	Load_11	0.360	0.1	0.4	0.949
3	Load_12	0.944	0.2	1.0	0.970
4	Load_14	0.723	0.4	0.8	0.894
5	Load_3	0.724	0.4	0.8	0.894
6	Load_4	0.481	0.1	0.5	0.970
7	Load_5	0.959	0.2	1.0	0.970
8	Load_6	0.718	0.1	0.7	0.986
9	Load_7	0.120	0.1	0.2	0.707
10	Load_8	0.720	0.2	0.8	0.949
11	Load_9	0.840	0.5	1.0	0.868

Table 3.3.: Power parameters of loads in the initial EC model

#	Name	Type
1	Transformer_0_1	Transformer Type 25MVA 50Hz 110/20kV
2	Transformer_0_12	Transformer Type 25MVA 50Hz 110/20kV
3	Trf Conv 1	Transformer Type 5MVA 50Hz 10–20kV
4	Trf Conv 2	Transformer Type 5MVA 50Hz 10–20kV
5	Trf Conv 3	Transformer Type 5MVA 50Hz 10–20kV

Table 3.4.: List of transformers in the EC model

#	Name	Target Voltage [kV]	ΔV max [%]	ΔV min [%]
1	T0	110.0	5.0	-5.0
2	T1	20.0	5.0	-5.0
3	T10	20.0	5.0	-5.0
4	T11	20.0	5.0	-5.0
5	T12	20.0	5.0	-5.0
6	T13	20.0	5.0	-5.0
7	T14	20.0	5.0	-5.0
8	T2	20.0	5.0	-5.0
9	T3	20.0	5.0	-5.0
10	T4	20.0	5.0	-5.0
11	T5	20.0	5.0	-5.0
12	T6	20.0	5.0	-5.0
13	T7	20.0	5.0	-5.0
14	T8	20.0	5.0	-5.0
15	T9	20.0	5.0	-5.0
16	Term Conv 1	10.5	5.0	-5.0
17	Term Conv 2	10.5	5.0	-5.0
18	Term Conv 3	10.5	5.0	-5.0

Table 3.5.: Voltage parameters of buses in the EC model

After running the power flow calculation in PF, all elements are highlighted in green, indicating that there are no voltage violations or overloaded elements in the system [42]. The results of the power flow calculation are shown in Figure 3.6(left).

Next, the Python code is compiled. After running the code, a DataFrame with results was built. Figure 3.6(right) shows the output of the Python calculation, presenting the same results for line loading as obtained in PowerFactory. This confirms that the code has been correctly written and can be used for further simulations and optimizations.

Conclusions of the initial verification stage:

- A simplified EC model was successfully built in PF, and a Python code was written to automate calculations.
- The combination of PF and Python allows a flexible, fast, and efficient tool for power system analysis.
- The verification confirmed that both software have identical results, meaning that Python can be used for further research.

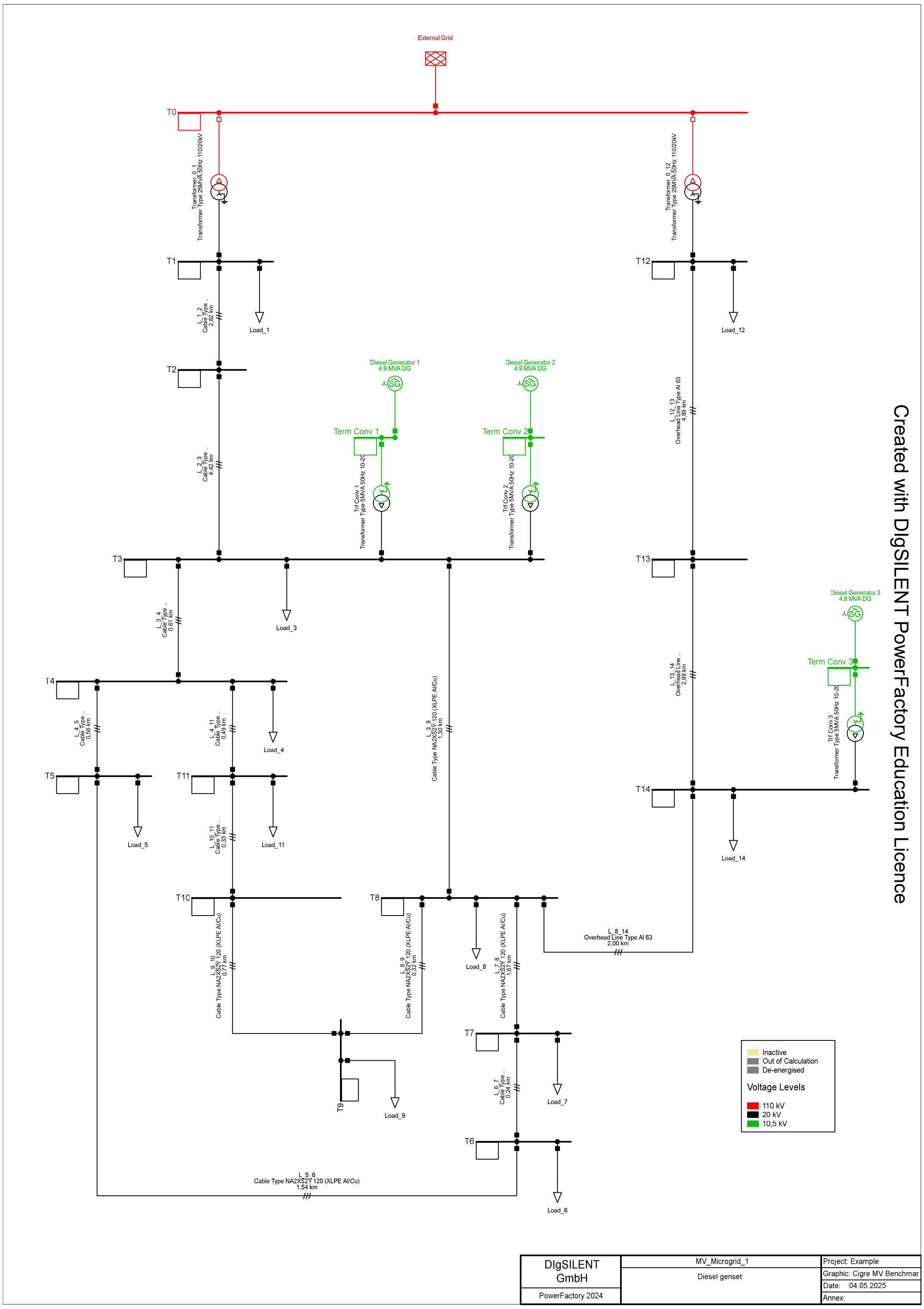


Figure 3.4.: Initial simplified model of one EC before Load Flow calculation [41]

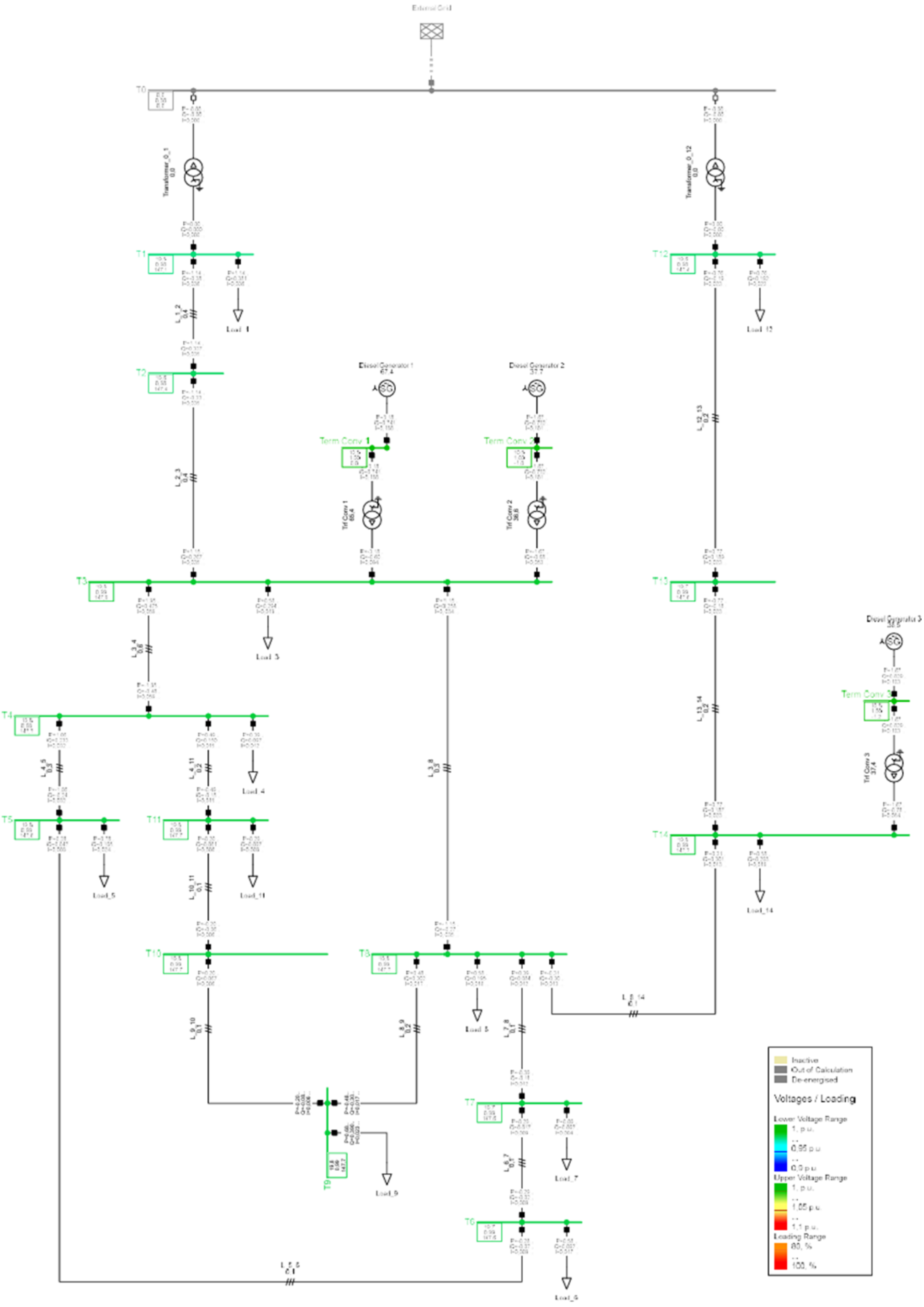
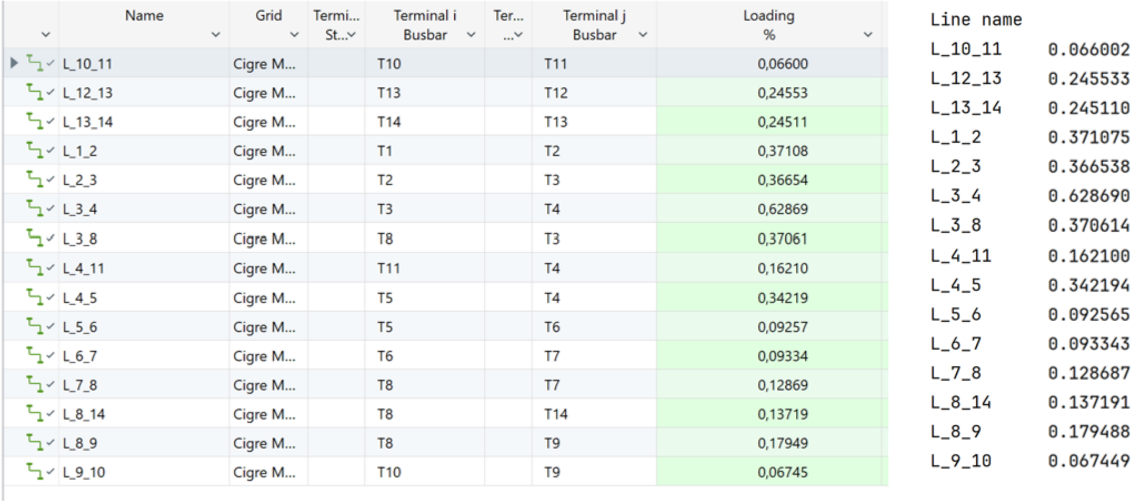


Figure 3.5.: Initial simplified model of one EC after Load Flow calculation



Name	Grid	Termi...	Terminal i	Ter...	Terminal j	Loading
		St...	Busbar	...	Busbar	%
L_10_11	Cigre M...		T10		T11	0,06600
L_12_13	Cigre M...		T13		T12	0,24553
L_13_14	Cigre M...		T14		T13	0,24511
L_1_2	Cigre M...		T1		T2	0,37108
L_2_3	Cigre M...		T2		T3	0,36654
L_3_4	Cigre M...		T3		T4	0,62869
L_3_8	Cigre M...		T8		T3	0,37061
L_4_11	Cigre M...		T11		T4	0,16210
L_4_5	Cigre M...		T5		T4	0,34219
L_5_6	Cigre M...		T5		T6	0,09257
L_6_7	Cigre M...		T6		T7	0,09334
L_7_8	Cigre M...		T8		T7	0,12869
L_8_14	Cigre M...		T8		T14	0,13719
L_8_9	Cigre M...		T8		T9	0,17949
L_9_10	Cigre M...		T10		T9	0,06745

```
Line name
L_10_11     0.066002
L_12_13     0.245533
L_13_14     0.245110
L_1_2       0.371075
L_2_3       0.366538
L_3_4       0.628690
L_3_8       0.370614
L_4_11      0.162100
L_4_5       0.342194
L_5_6       0.092565
L_6_7       0.093343
L_7_8       0.128687
L_8_14      0.137191
L_8_9       0.179488
L_9_10      0.067449
```

Figure 3.6.: Results of Load Flow calculation using PF and Python

3.3.1. Preparing data for Monte Carlo simulation

After the model is successfully verified for operation and all preparatory calculations are performed, the next step is to analyze the probability of overloads in transmission lines and generators using the MC simulation [43].

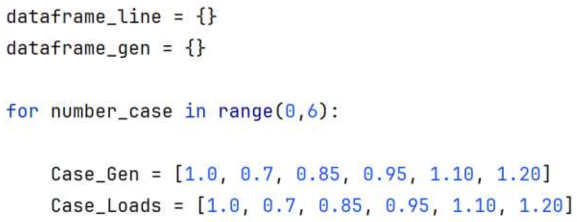
As shown in Figure 3.5, the current model operates without technical issues. To simulate stressed operating conditions for the model, a loop was applied in the Python code to gradually increase the load by 1% per iteration until any line in the system exceeds the defined overload threshold.

In this research, the loop's stopping criterion is set to 10% overload on at least one transmission line. This ensures the system experiences significant stress, providing a realistic base for the MC simulation.

MC simulation typically requires historical data to predict output. However, in this work, instead of historical data, a custom dataset of six different power flow scenarios is used. These scenarios have different active power generation and load combinations, providing a realistic spread of input conditions.

As shown in Figure 3.7, the power values for generators and loads are changed six times, each time filling the DataFrame for lines and generators. This DataFrame serves as the input data for the MC simulation.

This modeling choice allows a more realistic assessment of potential overloads and voltage issues under diverse operating conditions.



```
dataframe_line = {}
dataframe_gen = {}

for number_case in range(0,6):

    Case_Gen = [1.0, 0.7, 0.85, 0.95, 1.10, 1.20]
    Case_Loads = [1.0, 0.7, 0.85, 0.95, 1.10, 1.20]
```

Figure 3.7.: Code block for generating the six scenarios

Generation and load levels were varied simultaneously in six scenarios (70%, 85%, ..., 120% of nominal values). These bounds reflect the physical limitations of the units and ensure that the simulated injections remain within technically feasible ranges. This approach is explained by the partial autonomy of the studied Energy Communities, where generation is based on renewable sources and diesel generators with uncertain and unstable output. In such systems, neither generation nor consumption can be considered strictly controlled or deterministic, which justifies the simultaneous stochastic variation of both processes.

The scenarios were assumed to be stochastically independent across generators, meaning that variations in one generator's output were not correlated with variations in others. This assumption simplifies the sampling process and avoids the need to model interdependencies between generation units.

The scenario outputs for all generators are summarized in 3.6, which provides the basis for the subsequent statistical analysis.

name	**GenLoad0**	**GenLoad1**	**GenLoad2**	**GenLoad3**	**GenLoad4**	**GenLoad5**
Diesel Generator 1	3.892	2.724	3.3082	3.697	4.2812	4.671
Diesel Generator 2	2.048	1.433	1.741	1.946	2.2528	2.457
Diesel Generator 3	2.048	1.433	1.741	1.946	2.2528	2.457

Table 3.6.: Resulting values for generator outputs after the six scenarios

To better understand the variation in generator outputs, a probability density function (PDF) was calculated for DG1, as shown in Figure 3.8. The PDF follows an approximately Gaussian distribution, characterized by the mean value (μ) and the standard deviation (σ) derived from the scenario data.

The mean value indicates the central tendency of the generator's output, while the standard deviation quantifies the variability across the scenarios. A small σ reflects stable operation with low fluctuation in power output, whereas a large σ suggests significant variability and potential instability in generator performance. In Figure 3.8, the mean is marked by a vertical red dashed line, and the σ boundaries are indicated by orange dashed lines.

The PDF follows an approximately Gaussian distribution. The bounded range (70%–120% of nominal) ensures that extreme or unrealistic values are excluded, while the use of multiple scenarios captures a realistic spread of operational states.

For a more direct comparison between all generators, Figure 3.9 presents the output values for each case (Case0–Case5) in a bar chart. This visual format highlights that DG1 consistently operates at higher levels compared to DG2 and DG3, and in certain scenarios (e.g., Case 5) exceeds its nominal capacity. This condition represents a potential overload, which, in real-world operation, could lead to accelerated wear or even damage to the generator.

Together, Figures 3.8 and 3.9 provide complementary insights: Figure 3.8 captures the statistical distribution of DG1's output, while Figure 3.9 shows the discrete case-by-case values for all generators. These statistical profiles will be used as probabilistic inputs for the Monte Carlo simulation, enabling a realistic estimation of overload probabilities under diverse operating conditions.

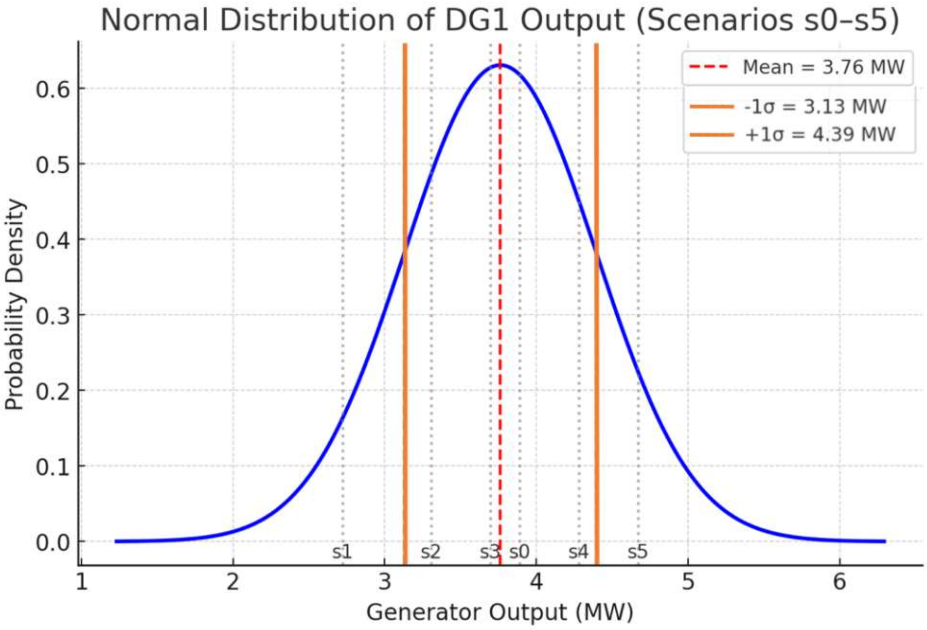


Figure 3.8.: Normal Distribution of DG1

Probabilistic model for DG1. Based on the six scenario values from Table 3.6, for DG1 the sample mean and standard deviation are:

$$\mu_{DG1} = 3.762 \text{ MW}, \qquad \sigma_{DG1} = 0.63 \text{ MW}.$$

In the Monte Carlo study we model the DG1 active-power injection as a *truncated normal random variable*,

$$P_{G1} \sim \mathcal{N}(\mu, \sigma^2) \text{ truncated to } [L, U],$$

where the truncation bounds **L** (Lower bound) and **U** (Upper bound) are taken directly from the scenario design:

$$L = 2.724 \text{ MW } (70\%), \qquad U = 4.671 \text{ MW } (120\%).$$

A common approach is to constrain the values within the interval: $[\mu - 3\sigma, \ \mu + 3\sigma] = [1.874, \ 5.65]$ MW, which covers 99.7% of all possible outcomes under a normal distribution. However, this range exceeds the feasible operating limits of DG1. Therefore, sampling is restricted to **[L,U]**, ensuring compliance with the generator's technical limits and preventing unrealistic injections. The same probabilistic modeling framework is applied to the remaining generators (and, analogously, to loads).

Thus, the necessary framework for conducting the MC simulation has been created.

A bar chart was created based on the DataFrame to better understand the data distribution and the differences between the six scenarios.

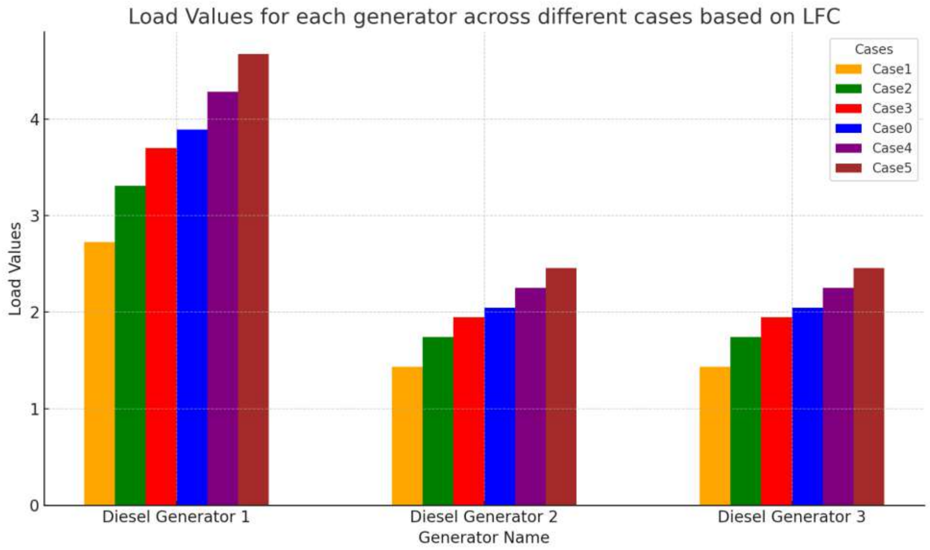


Figure 3.9.: Generator load values for each scenario (Case0 to Case5)

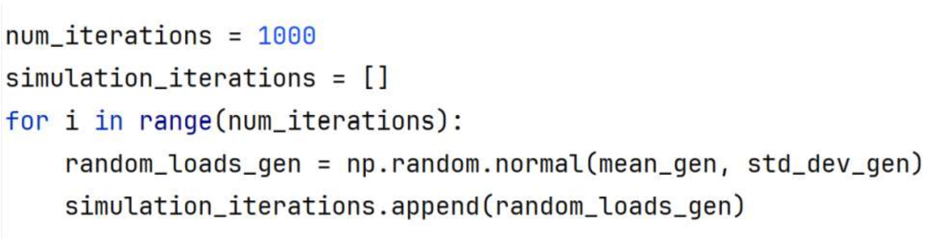
3.3.2. Estimation of the probability of overloads using Monte Carlo Simulation

The simulation can begin once the required input data for the MC has been successfully obtained. The simulation requires two parameters:

- The mean value μ,
- The standard deviation σ.

These two parameters define the normal (Gaussian) distribution. In this project, they can be calculated automatically based on the six previously defined load flow scenarios, which can be seen in Table 3.6.

To simulate the stochastic variation in generation or load, a Monte Carlo loop was implemented using the NumPy library, as shown in the Figure 3.10. At each iteration, a new sample of active power values was drawn from a normal distribution based on previously computed mean and standard deviation values. These samples are later used as input to load flow calculations to analyze the network response.



```
num_iterations = 1000
simulation_iterations = []
for i in range(num_iterations):
    random_loads_gen = np.random.normal(mean_gen, std_dev_gen)
    simulation_iterations.append(random_loads_gen)
```

Figure 3.10.: Parametrization using NumPy library

The variable **random__loads__gen** represents a single realization of generator power values for one Monte Carlo scenario. These values are inserted into the power system model to compute the resulting power flows, line loadings, and potential overloads.

Detailed numerical and graphical results of the MC simulation will be given in the next chapter.

3.3.3. System modernization: Installing an additional parallel line

The result obtained from the MC simulation showed that certain lines, in particular **Line 3__4**, experience overloads more often than others.

One cost-effective and most straightforward option for reducing the line load is to install a second parallel line with identical electrical parameters. In PF, this second line is installed parallel to **L__3__4** and called **L__3__4__parallel**. A fragment of the modified PowerFactory topology is shown in Figure 3.11.

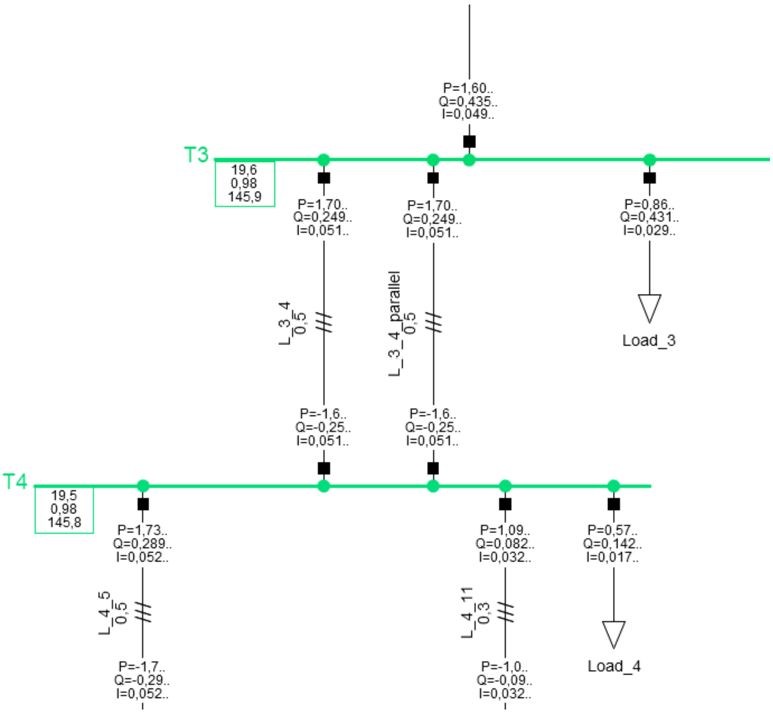


Figure 3.11.: Fragment of the model with a parallel line installed

Repeated calculations did not reveal any overloaded lines. The following section will provide a detailed evaluation of the effectiveness of this upgrade. Conclusions of the MC-Simulation:

- The issue of power line overload using the Monte Carlo approach was investigated.
- By modeling different scenarios of initial conditions, the probability of overloads was estimated, and the most vulnerable sections of the network were identified.
- As a result of the analysis, the solution to the overload problem of **L__3__4** was to install an additional parallel line **L__3__4__parallel**, which reduced the load and improved the reliability of the entire system.

3.4. Extended Model: Three Energy Communities - Increased Complexity and Overloads

The next step is to increase the complexity of the model. In real-world systems, several ECs can be linked through the external grid and operate in coordination. During peak load hours, surplus energy from one EC can flow to another.

The new extended model consists of three ECs with an identical structural configuration. They differ in their power sources:

- EC1 operates only on DGs,
- EC2 includes only PV panels,
- EC3 combines one DG and one PV panel.

As mentioned before, the large number of RES is the main characteristic of ECs. All ECs are interconnected through a common external grid. The current system includes:

- 45 transmission lines
- 13 TR
- 33 loads
- 8 PV installations
- 4 DG

The PF model diagram is provided in the Appendix A.1.

3.4.1. Parametrization of PV installations

The generator and load parameters in the second and third ECs were taken from the already configured EC1. In addition, these communities contain PV panels, which must also be correctly parameterized with real operating conditions.

The parameterization of PV will be carried out using machine learning techniques for the greatest accuracy and realistic representation. The model will predict future active power values for the PV panel based on historical data.

3.4.2. Simple Linear Regression prediction algorithm

To assess the correctness of the model's prediction, three metrics are typically used: the coefficient of determination of the training data R^2_{train}, the coefficient of determination of the test data R^2_{test}, and the mean square error (MSE).

The coefficient of determination of the training data R^2_{train} shows how well the model has learned from the training set (usually 80% of the set is allocated for training and 20% for testing). A value of R^2_{train} close to 1 indicates a high degree of fit to the training data.

The coefficient of determination for the test data R^2_{test} shows how well the model predicts new data it has not seen before. A value close to 1 indicates high prediction accuracy.

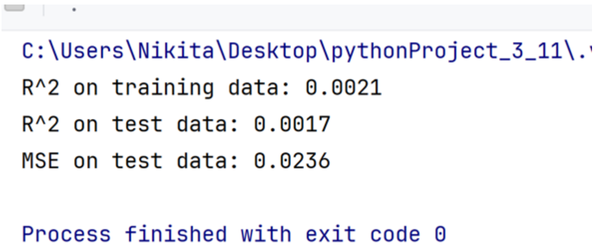
In some cases, R^2_{train} may be high, while R^2_{test} is much lower. This often means overfitting, when the model remembers the training data too well but fails to generalize to new data.

The third parameter is the value of the mean square error (MSE), which represents the average error between the predicted and actual values, squared. The smaller the MSE, the better the model's performance.

Initially, for forecasting, simple linear regression was used for the analysis, where the active power of the PV panels was a function of time (in hours). However, the limited data set (only active power and hours) led to an extremely low coefficient of determination:

- $R^2_{\text{train}} = 0.0021$ on training data,
- $R^2_{\text{test}} = 0.0017$ on test data,
- $MSE = 0.0236$ on test data.

Such low values indicate no correlation between the real and predicted data. This means that linear regression cannot be used for forecasting. The performance of linear regression can be seen in Figure 3.12.



```
C:\Users\Nikita\Desktop\pythonProject_3_11\.
R^2 on training data: 0.0021
R^2 on test data: 0.0017
MSE on test data: 0.0236

Process finished with exit code 0
```

Figure 3.12.: Performance of Linear Regression algorithm

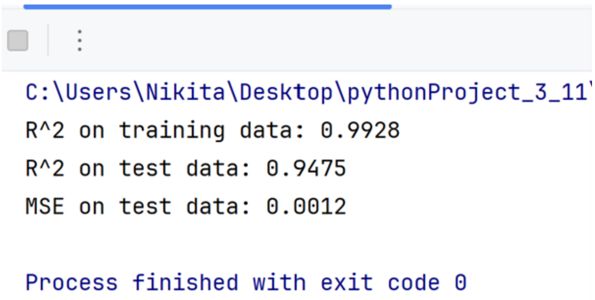
3.4.3. Random Forest Regression algorithm

To improve the model's prediction accuracy, new parameters were added to Table 3. Initially containing only time and power, the table was expanded with new columns: day, month, and holidays, reflecting seasonality, day of the week, and weekends. These parameters will help the model to find more meaningful patterns in the data that were previously hidden and not obvious.

Since the data is highly nonlinear and dispersed, a more powerful machine learning algorithm was used – Random Forest Regression [44].

The results show excellent learning performance. The model predicts the power values with high accuracy and low error. The results are shown in Figure 3.13.

- $R^2_{\text{train}} = 0.9928$ on training data,
- $R^2_{\text{test}} = 0.9475$ on test data,
- $MSE = 0.0012$ on test data.



```
C:\Users\Nikita\Desktop\pythonProject_3_11'
R^2 on training data: 0.9928
R^2 on test data: 0.9475
MSE on test data: 0.0012

Process finished with exit code 0
```

Figure 3.13.: Performance of Random Forest Regression algorithm

A plot for estimating the prediction quality is shown in Figure 3.14. The blue points represent the real data, while the red lines show the model's predictions. The plot shows that the red lines lie close to the blue dots, indicating a strong fit. The dataset was split with 80% used for training and 20% for testing, which is a common practice in machine learning.

The red lines lie on the blue dots, demonstrating accurate convergence. Appendix B.1 contains detailed Python code with explanations.

Table 3.7.: Expanded input dataset for Random Forest Regression algorithm

Time	Power	Month	Day	Holidays
1	0	1	1	1
2	0	1	1	1
3	0	1	1	1
4	0	1	1	1
5	0	1	1	1
6	0	1	1	1
7	0	1	1	1
8	0	1	1	1
9	0.0020169	1	1	1
10	0.013406	1	1	1
11	0.027662	1	1	1
12	0.036961	1	1	1
13	0.042793	1	1	1
14	0.042995	1	1	1
15	0.032854	1	1	1
16	0.0133	1	1	1
17	0.0014256	1	1	1
18	0	1	1	1
19	0	1	1	1
20	0	1	1	1
21	0	1	1	1
22	0	1	1	1
23	0	1	1	1
24	0	1	1	1

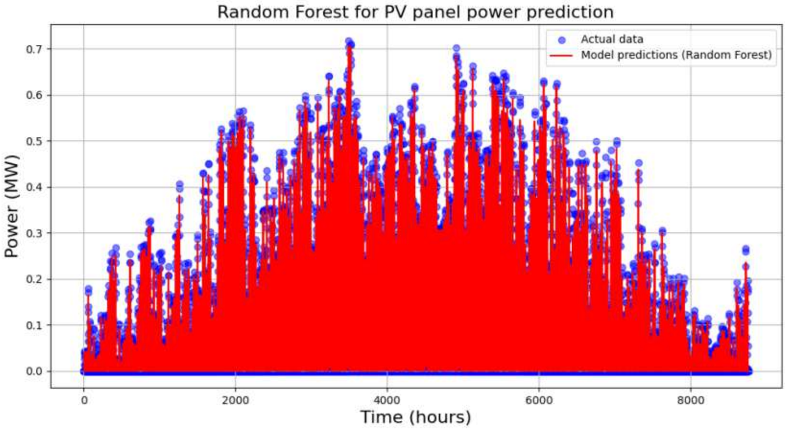


Figure 3.14.: Comparison of predicted and actual power values

To visualize the seasonality of PV power generation, the overall plot was divided into six subplots for different periods of the year, each representing a specific period. This helps to visualize the power generation in different seasons of the year. The following periods were selected:

- Winter: 0–1500 hours
- Spring: 1500–3500 hours
- Summer: 3500–5500 hours
- Autumn: 6000–8500 hours
- Peak Summer: 4400–5100 hours with the most active solar period
- Valley Winter: 0–750 hours with minimum solar activity

Figure 3.15 shows the predicted power for each season.

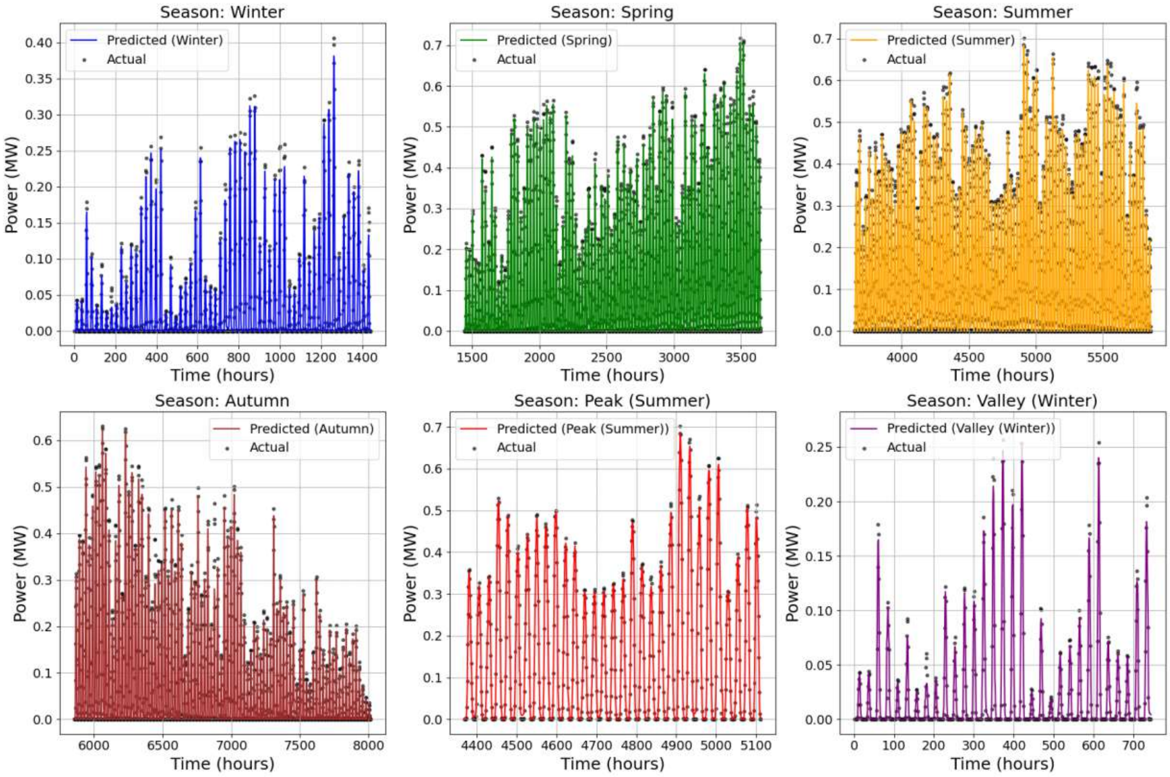


Figure 3.15.: Seasonal segmentation of forecasted vs actual PV power generation

This stage plays one of the key roles in the accurate parameterization of PV panels. The model was trained on real PV data for 2023 and proved high accuracy in forecasting.

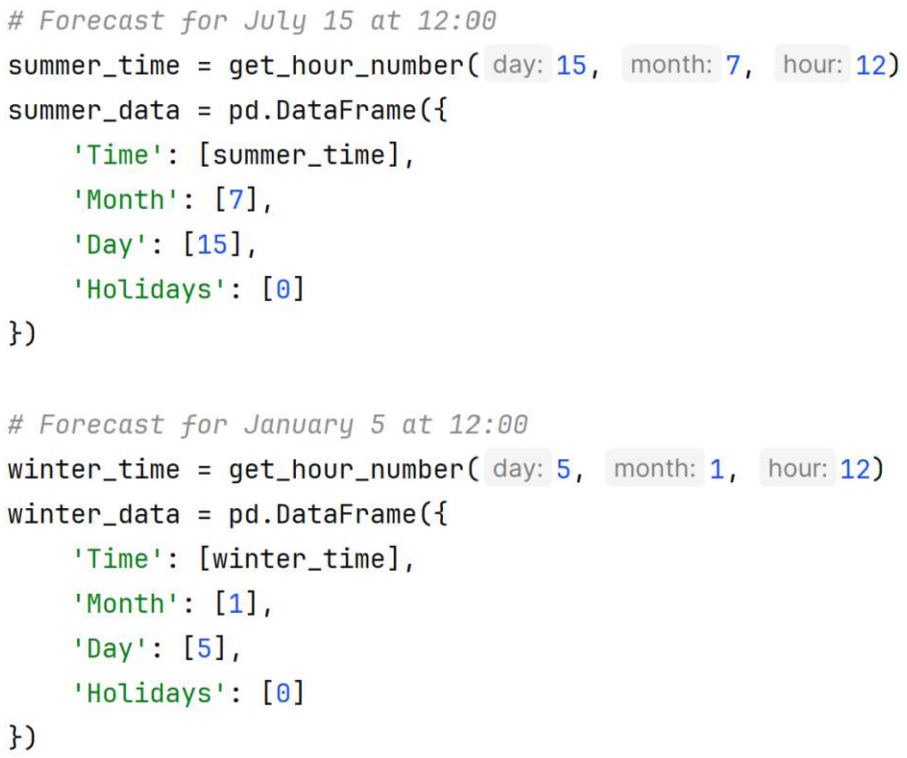
As a result, the grid operator can now utilize the algorithm in a power plant to estimate the expected power at any future time. Without this stage, the power of PV would be manually selected, reducing the reliability of further calculations.

The use of machine learning to predict future output is used in professional power system studies, which increases the reliability of all the following stages, including PSO, GA, and ANN.

As a test of the model's forecasting performance, the predicted PV power will be presented for two dates:

- Summer forecast: July 15, 12:00,
- Winter forecast: January 5, 12:00.

The figure 3.16 shows the forecasting procedure using the trained model in Python. The operator only needs to enter the desired date and time (e.g., July 15 or January 5 at 12:00), and the result will be generated automatically. This allows for the forecasted PV power value to be calculated for any specified moment.

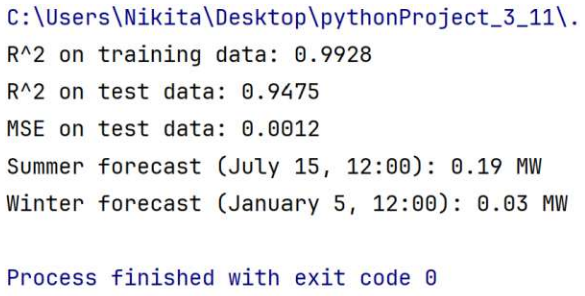


```
# Forecast for July 15 at 12:00
summer_time = get_hour_number( day: 15,  month: 7,  hour: 12)
summer_data = pd.DataFrame({
    'Time': [summer_time],
    'Month': [7],
    'Day': [15],
    'Holidays': [0]
})

# Forecast for January 5 at 12:00
winter_time = get_hour_number( day: 5,  month: 1,  hour: 12)
winter_data = pd.DataFrame({
    'Time': [winter_time],
    'Month': [1],
    'Day': [5],
    'Holidays': [0]
})
```

Figure 3.16.: Power forecast results

The numerical results of the PV forecast are presented in Figure 3.17:



```
C:\Users\Nikita\Desktop\pythonProject_3_11\.
R^2 on training data: 0.9928
R^2 on test data: 0.9475
MSE on test data: 0.0012
Summer forecast (July 15, 12:00): 0.19 MW
Winter forecast (January 5, 12:00): 0.03 MW

Process finished with exit code 0
```

Figure 3.17.: Power forecast results

The visualization of the predicted PV power for two selected points is shown in Figure 3.18:

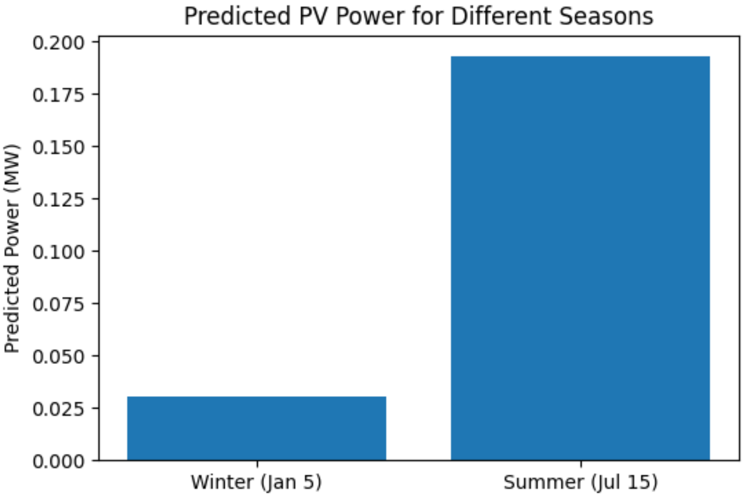
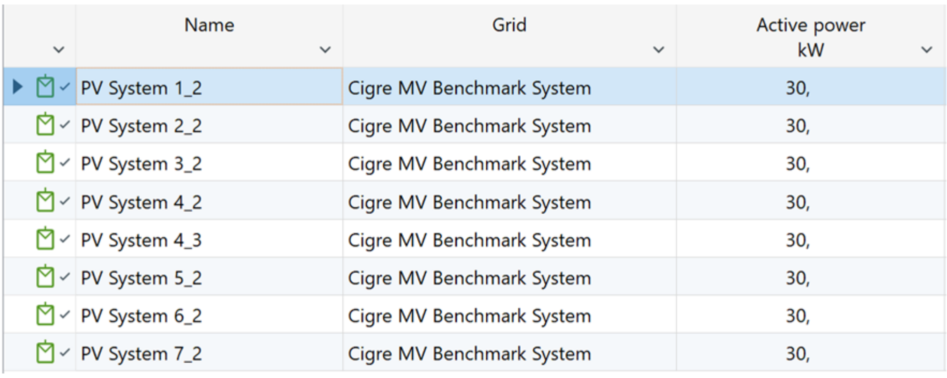


Figure 3.18.: Visualization of forecast results

The results confirm that the model provides adequate forecasts. In winter, the predicted power value is **0.03 MW**, which indeed reflects low solar radiation. In summer, the power is significantly higher and accounts **0.19 MW**, which indicates a higher level of solar radiation.

For the parameterization of the PV panels, the value of the winter period was selected, in which the load on the electrical system increases due to additional heating and lighting. Solar generation drops significantly due to reduced irradiation and daylight hours. Thus, the worst case is “built in", making the model resistant to peak loads. Figure 3.19 shows the parameterized PV panels in the PF.



	Name	Grid	Active power kW
▶ ✓	PV System 1_2	Cigre MV Benchmark System	30,
✓	PV System 2_2	Cigre MV Benchmark System	30,
✓	PV System 3_2	Cigre MV Benchmark System	30,
✓	PV System 4_2	Cigre MV Benchmark System	30,
✓	PV System 4_3	Cigre MV Benchmark System	30,
✓	PV System 5_2	Cigre MV Benchmark System	30,
✓	PV System 6_2	Cigre MV Benchmark System	30,
✓	PV System 7_2	Cigre MV Benchmark System	30,

Figure 3.19.: Parameterization of PV panels in the PF

With the successful parameterization of PV panels, the new extended model is now ready for analysis (Appendix A.1). However, after integrating the three ECs, new challenges have emerged. Particularly, EC1 has overloads in several lines and voltage violations, indicating potential reliability issues. These elements are highlighted in red, indicating technical issues. It means that optimization methods are needed to improve grid stability.

3.5. Estimation of the probability of overloads using Monte Carlo Simulation

Similar to the previous part, an MC analysis will be performed for a new model consisting of three interconnected ECs.

The repeated MC simulation is essential for assessing reliability and identifying potential vulnerabilities. The power grid is now more complex, with RES, DGs, and an external grid. The degree of uncertainty is higher, mainly due to PV and variable loads.

Six different cases were calculated by changing the initial data of generators, PV installations, and loads as part of the simulation. All calculation results were combined into one single DataFrame, which serves as the basis for the Monte Carlo analysis.

Detailed numerical and graphical results of the MC simulation will be presented in the next chapter.

3.5.1. Application of PSO Optimization

After completing all preparatory work, an OPF can be carried out to optimize the system and bring it to a stable state, without voltage violations and overloaded elements.

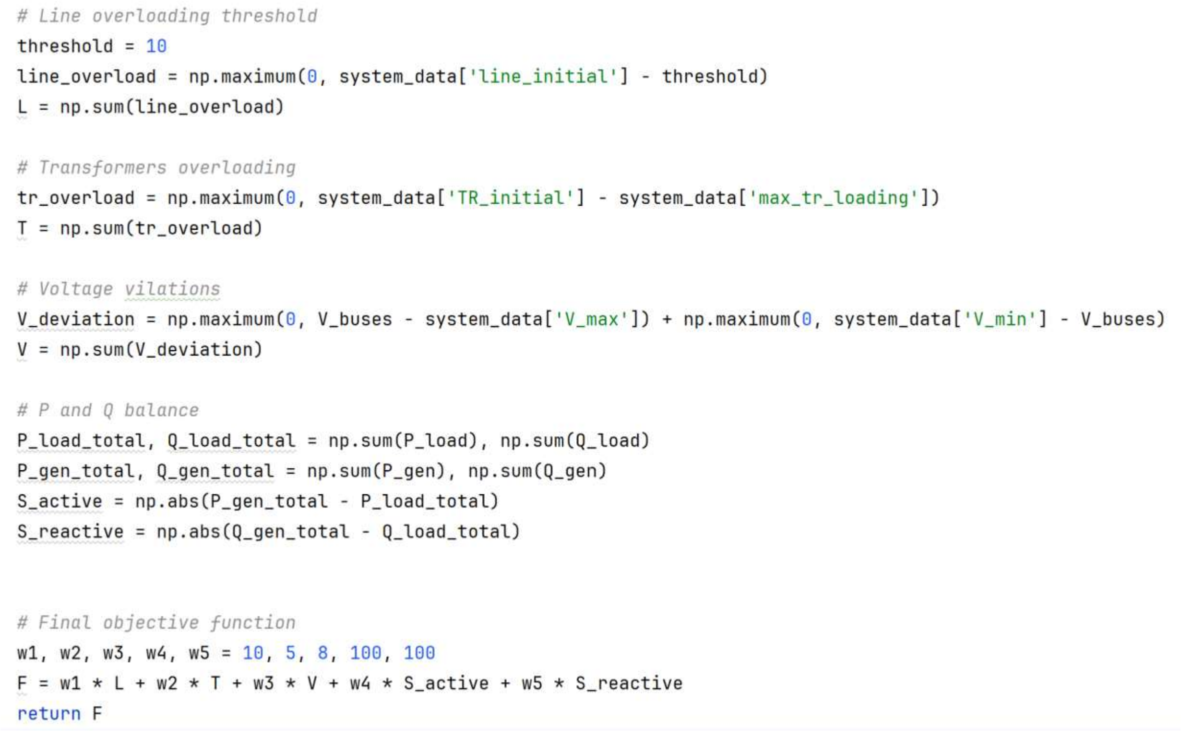
Since the current power grid is too complex for traditional methods, such as the Newton-Raphson approach, a metaheuristic algorithm should be applied.

The first chosen algorithm is PSO. As previously described, PSO iterates through all possible sets of system parameters, aiming to minimize the value of FF. Consequently, PSO helps to find the optimal power distribution and reduce overloads.

Currently, the energy system operates in an unstable mode (see Appendix A.1). Overloads are observed on all elements. On most buses, especially in EC1, voltage drops are more than 50%. Considering all technical constraints and limitations, the PSO optimization will be applied to the system. The parametrization of FF is shown in Figure 3.20.

In the context of power systems, PSO has been widely used for various optimization problems such as economic load distribution (ELD), voltage control, and overload mitigation. For example, [31] demonstrates the effectiveness of PSO in overload control in transmission systems under both normal and critical operating conditions. The selection of PSO parameters (weight coefficients) was based on literature guidelines [45] and adjusted empirically for convergence.

The implementation in this study follows the standard PSO procedure, which includes particle initialization, fitness evaluation, updating of personal and global records, and updates to position and velocity. The algorithm terminates when the convergence criteria are met or a predefined number of iterations is reached.



```
# Line overloading threshold
threshold = 10
line_overload = np.maximum(0, system_data['line_initial'] - threshold)
L = np.sum(line_overload)

# Transformers overloading
tr_overload = np.maximum(0, system_data['TR_initial'] - system_data['max_tr_loading'])
T = np.sum(tr_overload)

# Voltage vilations
V_deviation = np.maximum(0, V_buses - system_data['V_max']) + np.maximum(0, system_data['V_min'] - V_buses)
V = np.sum(V_deviation)

# P and Q balance
P_load_total, Q_load_total = np.sum(P_load), np.sum(Q_load)
P_gen_total, Q_gen_total = np.sum(P_gen), np.sum(Q_gen)
S_active = np.abs(P_gen_total - P_load_total)
S_reactive = np.abs(Q_gen_total - Q_load_total)


# Final objective function
w1, w2, w3, w4, w5 = 10, 5, 8, 100, 100
F = w1 * L + w2 * T + w3 * V + w4 * S_active + w5 * S_reactive
return F
```

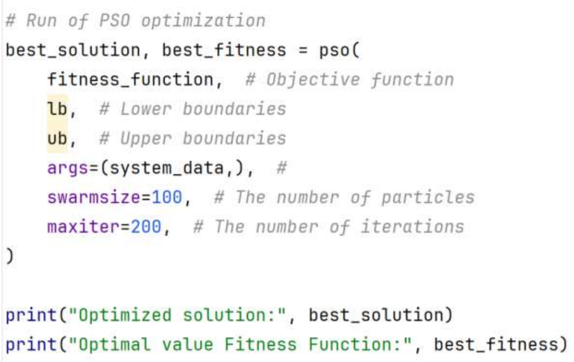
Figure 3.20.: Parametrization of the FF in PSO algorithm

The weight coefficients prioritize the following objectives:

- w1 – line overloads,
- w2 – transformer overloads,
- w3 – voltage violations,
- w4 – power imbalance,
- w5 – excessive load shedding.

Thus, the objective is to minimize the overall value of **"F"**. Optimization will be needed if the value of FF is high, indicating a lot of technical issues.

The Python code for running the PSO algorithm [46], which will attempt to find the optimal distribution of power and loads, reducing losses, overloads, and voltage violations, is shown in Figure 3.21.



```
# Run of PSO optimization
best_solution, best_fitness = pso(
    fitness_function,  # Objective function
    lb,  # Lower boundaries
    ub,  # Upper boundaries
    args=(system_data,),  #
    swarmsize=100,  # The number of particles
    maxiter=200,  # The number of iterations
)

print("Optimized solution:", best_solution)
print("Optimal value Fitness Function:", best_fitness)
```

Figure 3.21.: Parametrization of the PSO algorithm

This code initializes PSO by finding the best set of parameters that minimizes the value of FF. It performs 200 iterations, during which the particles will update their position based on the best solutions found, and returns two values:

- **best_solution** – a vector of optimized values found by PSO (new optimized parameters).
- **best_fitness** – the minimum value of the FF after optimization.

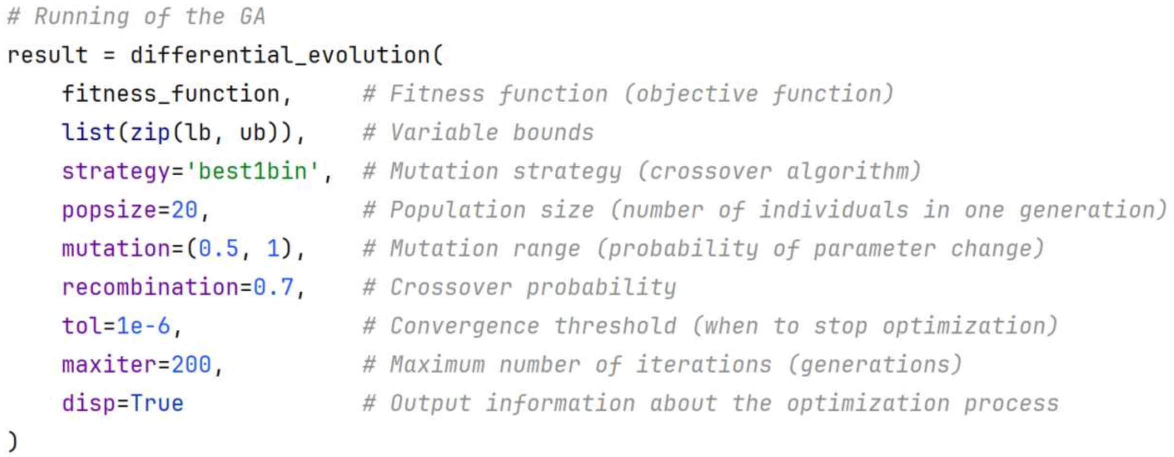
After the PSO optimization, new parameters for generators and loads were determined, which will ultimately be automatically configured in PF. These new parameters ensure that the power grid experiences no technical issues.

The next chapter will present results and tables with new optimized parameters.

3.5.2. Application of GA Optimization

The Python code for running the GA algorithm, which will attempt to find the optimal distribution of power and loads, reducing losses, overloads, and voltage violations, is shown in Figure 3.22.

The parametrization of FF remains unchanged and is shown in Figure 3.20.



```
# Running of the GA
result = differential_evolution(
    fitness_function,       # Fitness function (objective function)
    list(zip(lb, ub)),      # Variable bounds
    strategy='best1bin',    # Mutation strategy (crossover algorithm)
    popsize=20,             # Population size (number of individuals in one generation)
    mutation=(0.5, 1),      # Mutation range (probability of parameter change)
    recombination=0.7,      # Crossover probability
    tol=1e-6,               # Convergence threshold (when to stop optimization)
    maxiter=200,            # Maximum number of iterations (generations)
    disp=True               # Output information about the optimization process
)
```

Figure 3.22.: Parametrization of the GA algorithm

Table 3.8 provides explanations for each parameter.

Table 3.8.: Description of GA parameters

Parameter	**Description**
strategy='best1bin'	This strategy ensures that the best solution is always used for generating new candidates. This accelerates convergence and provides more stable results compared to other random strategies.
popsize=20	Population size represents the number of possible solutions in each generation.
mutation=(0.5, 1)	Adds more randomness and can help avoid getting stuck in a local minimum.
recombination=0.7	Specifies how often parent solutions will be mixed for new individuals. Increasing the crossover helps find a good solution faster, but reduces diversity.
tol=1e-6	Convergence threshold. When the difference between generations becomes less than 1e-6, the algorithm stops.
maxiter=200	Maximum number of iterations. 200 was defined as the sufficient minimum.
disp=True	Enables output of optimization progress information during execution.

After the GA optimization, new parameters for the power grid were achieved, and the system does not experience any technical issues when using them.

The next chapter will present results and tables with new optimized parameters from PSO and GA algorithms.

3.5.3. Application of ANN

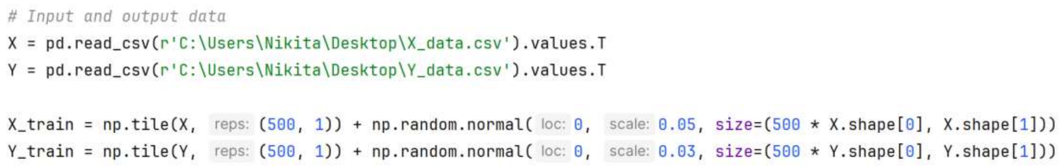
PSO and GA are sources of "ideal" solutions. They found values of P_{gen} that are balanced with the load, but they require a long time to calculate, especially if many real-time simulations are needed [47].

ANN is a fast substitute for PSO and GA. It was trained on the results obtained from PSO: P_{load} serves as the input, and P_{gen} is the output. Once trained, the ANN can, in a short time, predict the optimal P_{gen} value for a given P_{load}, without running complex iterative optimization each time.

The optimized values of loads and generators from PSO were saved into two separate `.csv` files:

- X – input data (loads)
- Y – output data (generators)

500 synthetic input-output pairs were generated by repeating the base case and adding Gaussian noise to train the model. Data from 400 samples were used for training, while the remaining 100 were used for testing the forecasting accuracy. The Python code for ANN is shown in Figure 3.23.



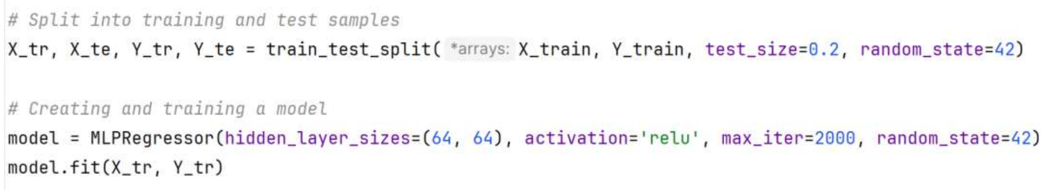
```
# Input and output data
X = pd.read_csv(r'C:\Users\Nikita\Desktop\X_data.csv').values.T
Y = pd.read_csv(r'C:\Users\Nikita\Desktop\Y_data.csv').values.T

X_train = np.tile(X,  reps: (500, 1)) + np.random.normal( loc: 0,  scale: 0.05, size=(500 * X.shape[0], X.shape[1]))
Y_train = np.tile(Y,  reps: (500, 1)) + np.random.normal( loc: 0,  scale: 0.03, size=(500 * Y.shape[0], Y.shape[1]))
```

Figure 3.23.: Parametrization of ANN

The dataset was split, with 80% (i.e., 400 samples) used for training and 20% (100 samples) for testing, which is a common practice in machine learning.

The neural network model is based on the **MLPRegressor** from the scikit-learn library, using two hidden layers, each containing 64 neurons [48]. The **ReLU** activation function is applied, and the training is performed over a maximum of 2000 iterations. The model weights are not manually specified but automatically initialized and optimized during training. The Python code for model creation and training is shown in Figure 3.24.



```
# Split into training and test samples
X_tr, X_te, Y_tr, Y_te = train_test_split( *arrays: X_train, Y_train, test_size=0.2, random_state=42)

# Creating and training a model
model = MLPRegressor(hidden_layer_sizes=(64, 64), activation='relu', max_iter=2000, random_state=42)
model.fit(X_tr, Y_tr)
```

Figure 3.24.: Model creation and training

For assessing the model's prediction performance, the value of MAE [49] was calculated, and plots for actual and predicted power were built.

Figure 3.25 shows four plots corresponding to four generators. The blue line represents the value of active power (in MW) obtained after PSO optimization. The orange line shows the predicted value calculated by the neural network.

The value of MAE is **0.0398 MW** and indicates the high accuracy of the model. This means that the ANN can reproduce the power distribution of generators obtained from PSO (or GA) much faster.

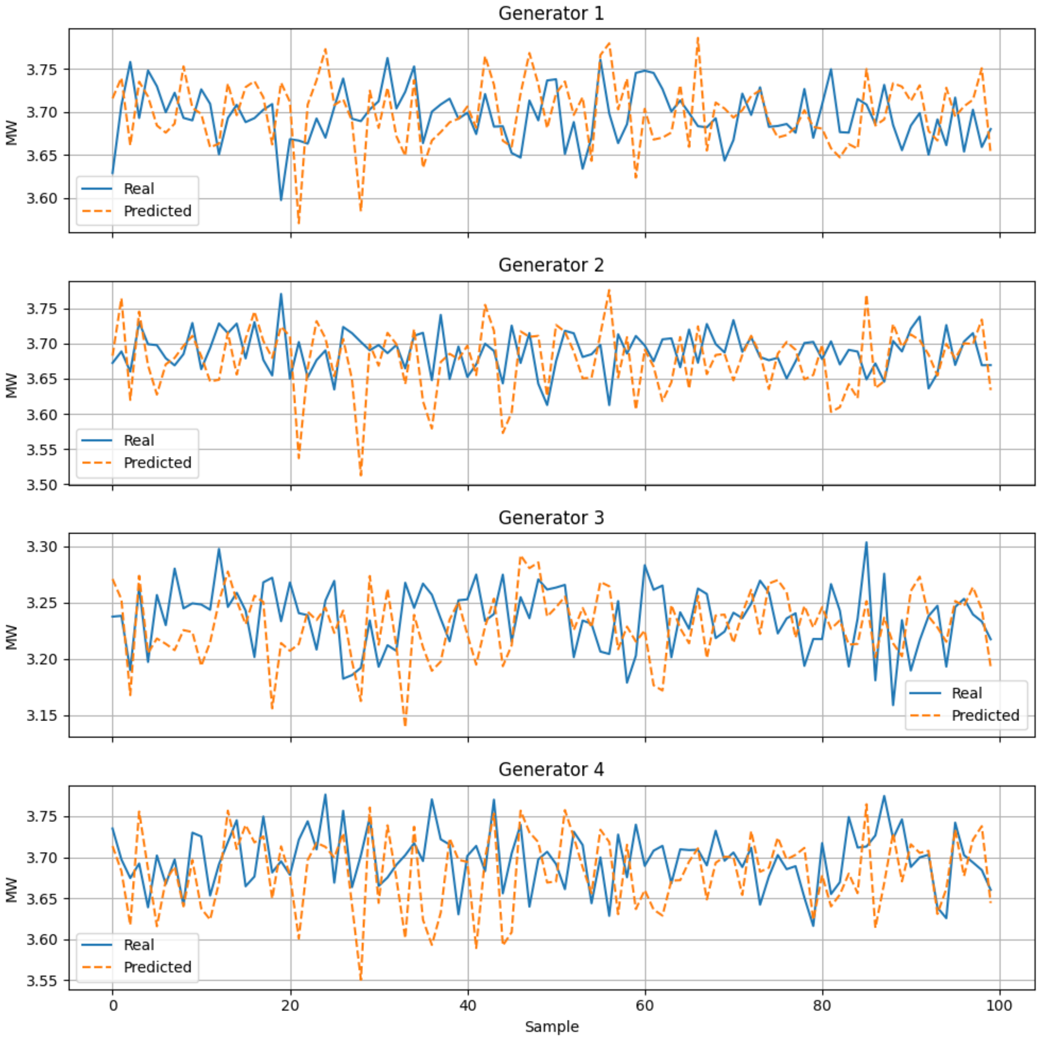


Figure 3.25.: Comparison of predicted and actual power values

The obtained power values were automatically exported to PowerFactory. Recalculation of the Load Flow confirmed the mode's stability and the absence of overloads. As a result, the grid operator at the power plant can now use the algorithm to estimate reliable generator output in real time for any updated load scenario. Without this approach, the operator would have to repeat the PSO or GA calculation for every change in load, which is computationally inefficient.

As mentioned, the ANN model demonstrates high accuracy only within the range of conditions for which it was trained. The MAE (Mean Absolute Error) value remains low for small load fluctuations within +—5–10% .

However, in unusual or critical situations—for example, when the load changes by more than ±30–50%—the neural network may produce incorrect values of generator output because such atypical values are beyond the trained sample. In such cases, it is recommended that optimization be rerun using PSO or GA.

Figure 3.26 shows the ANN's applicability limits for different levels of load change:

- The red line indicates the initial load level (100%), where the MAE is **0.0398 MW.**
- The blue line shows the growth of the MAE with an increase or decrease in load relative

to the trained level.

The plot shows that when the load changes by more than 30%, the MAE value increases significantly, making the model unreliable. If the load changes beyond ±40–50%, the neural network becomes incorrect.

This indicates that the ANN model works effectively within the trained range, but is not designed to extrapolate to extreme conditions. In such cases, the ANN model must be retrained, using updated data from PSO or GA.

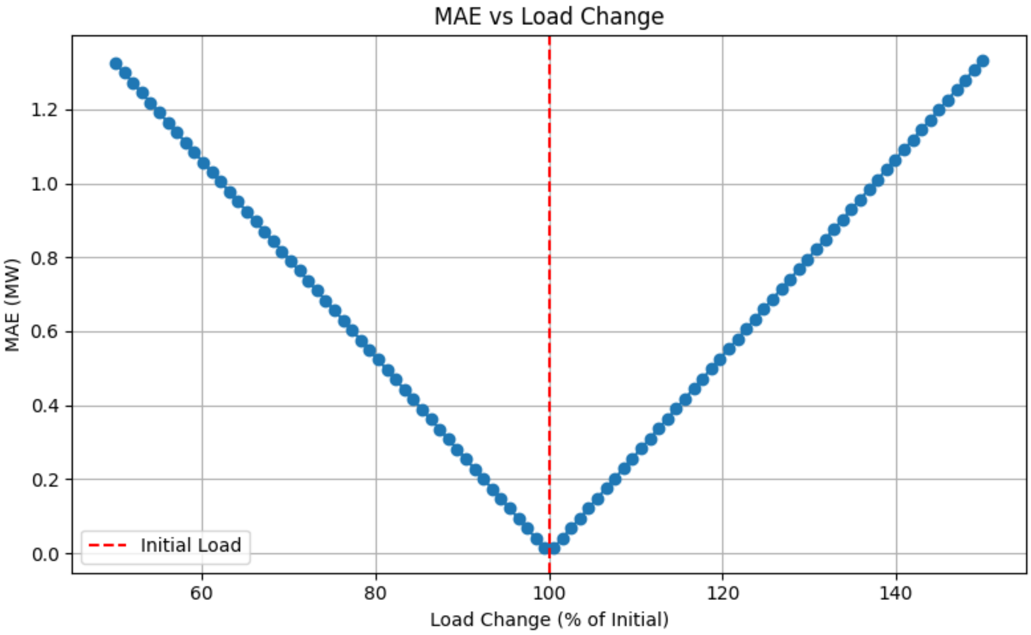


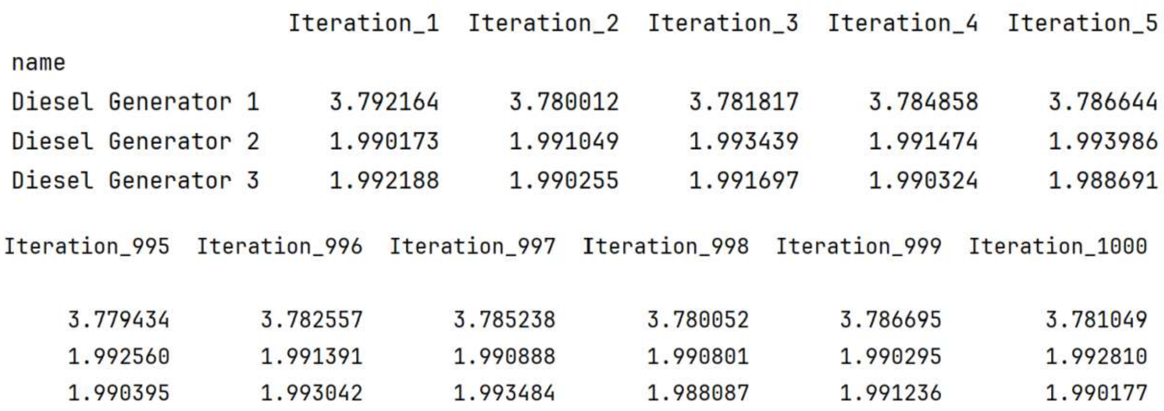
Figure 3.26.: Applicability limits for different levels of load change

4. Results and Discussion

This chapter presents a structured overview of all the obtained results. The subsection numbers correspond to the methodological steps of the previous part. Each section contains a graphical, numerical, and theoretical interpretation of the results. Additional diagrams and models can be found in Appendix A.

4.1. Result of Monte Carlo Simulation

Table 4.1 presents the input data for generators used in 1000 iterations for Load Flow calculations. The grid's most vulnerable and potentially critical lines can be identified based on these values.



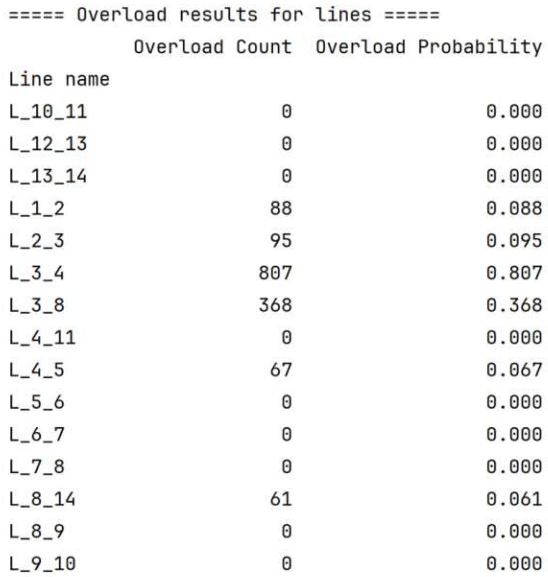
```
                    Iteration_1  Iteration_2  Iteration_3  Iteration_4  Iteration_5
name
Diesel Generator 1     3.792164     3.780012     3.781817     3.784858     3.786644
Diesel Generator 2     1.990173     1.991049     1.993439     1.991474     1.993986
Diesel Generator 3     1.992188     1.990255     1.991697     1.990324     1.988691

Iteration_995  Iteration_996  Iteration_997  Iteration_998  Iteration_999  Iteration_1000

     3.779434       3.782557       3.785238       3.780052       3.786695        3.781049
     1.992560       1.991391       1.990888       1.990801       1.990295        1.992810
     1.990395       1.993042       1.993484       1.988087       1.991236        1.990177
```

Figure 4.1.: Input data for generators for 1000 iterations

A threshold of **10%** line overload was defined to classify a line as overloaded. Table 4.2 shows the results of the MC simulation. According to Table 4.2, lines **L__3__4** and **L__3__8** were overloaded in **80.7%** and **36.8%** of the simulated cases, respectively. This indicates that these lines are the most critical regarding reliability and should be monitored during grid operation.



```
===== Overload results for lines =====
           Overload Count  Overload Probability
Line name
L_10_11                 0                 0.000
L_12_13                 0                 0.000
L_13_14                 0                 0.000
L_1_2                  88                 0.088
L_2_3                  95                 0.095
L_3_4                 807                 0.807
L_3_8                 368                 0.368
L_4_11                  0                 0.000
L_4_5                  67                 0.067
L_5_6                   0                 0.000
L_6_7                   0                 0.000
L_7_8                   0                 0.000
L_8_14                 61                 0.061
L_8_9                   0                 0.000
L_9_10                  0                 0.000
```

Figure 4.2.: Overload results for lines

The visualization of results is shown in Figure 4.3, clearly indicating that line **L__3__4** and **L__3__8** are in a critical state.

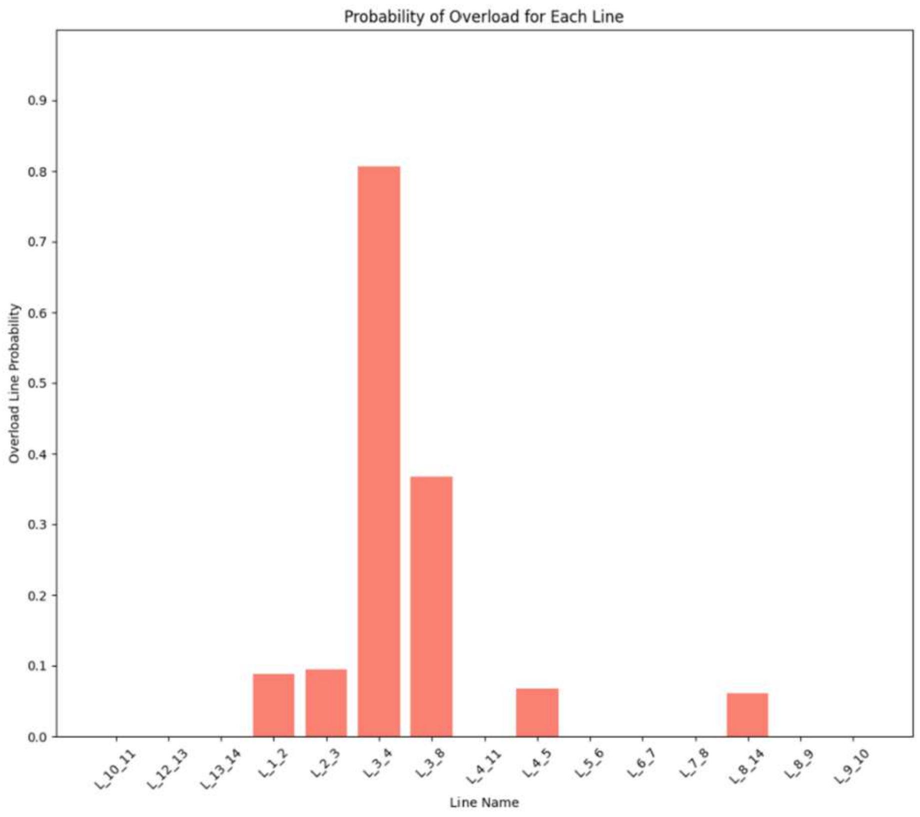
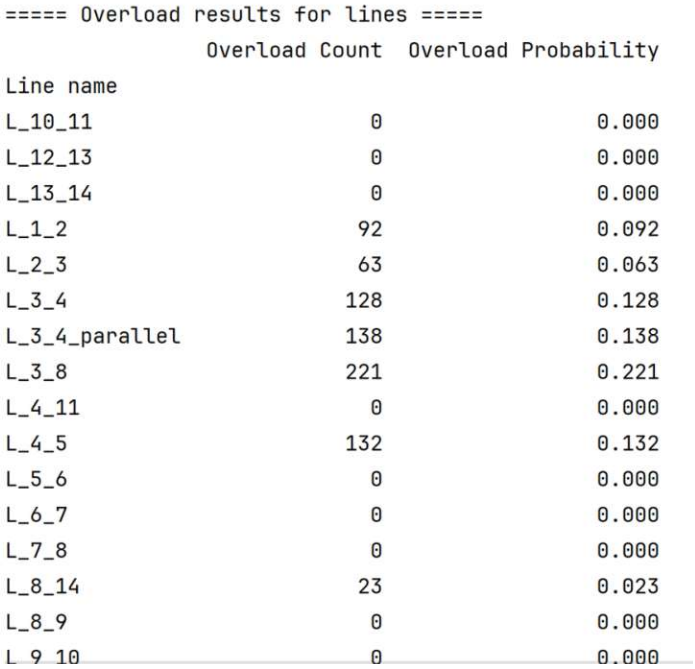


Figure 4.3.: Probability of line overloads across 1000 Monte Carlo simulations

4.2. Result for modernized system with an additional parallel line

For reducing line overloads, the new additional line **L__3__4__parallel** was installed parallel to line **L__3__4** . Table 4.4 represents the results of the MC simulation for the modernized network. According to the Table 4.4 , the overload probabilities for lines **L__3__4** and **L__3__8** were significantly decreased to 12.8% and 22.1% of the simulated cases, respectively, compared to the previous case 80.7% and 36.8%.



```
===== Overload results for lines =====
                Overload Count  Overload Probability
Line name
L_10_11                      0                 0.000
L_12_13                      0                 0.000
L_13_14                      0                 0.000
L_1_2                       92                 0.092
L_2_3                       63                 0.063
L_3_4                      128                 0.128
L_3_4_parallel             138                 0.138
L_3_8                      221                 0.221
L_4_11                       0                 0.000
L_4_5                      132                 0.132
L_5_6                        0                 0.000
L_6_7                        0                 0.000
L_7_8                        0                 0.000
L_8_14                      23                 0.023
L_8_9                        0                 0.000
L_9_10                       0                 0.000
```

Figure 4.4.: Overload results for lines

Figure 4.5 shows the new distribution of overload probabilities. It shows that the additional parallel line led to load redistribution across multiple lines. Not only was the overload probability in line **L__3__4** decreased, but also in line **L__3__8** from 36.8% to 22.1%. As a result, overall grid stability increased.

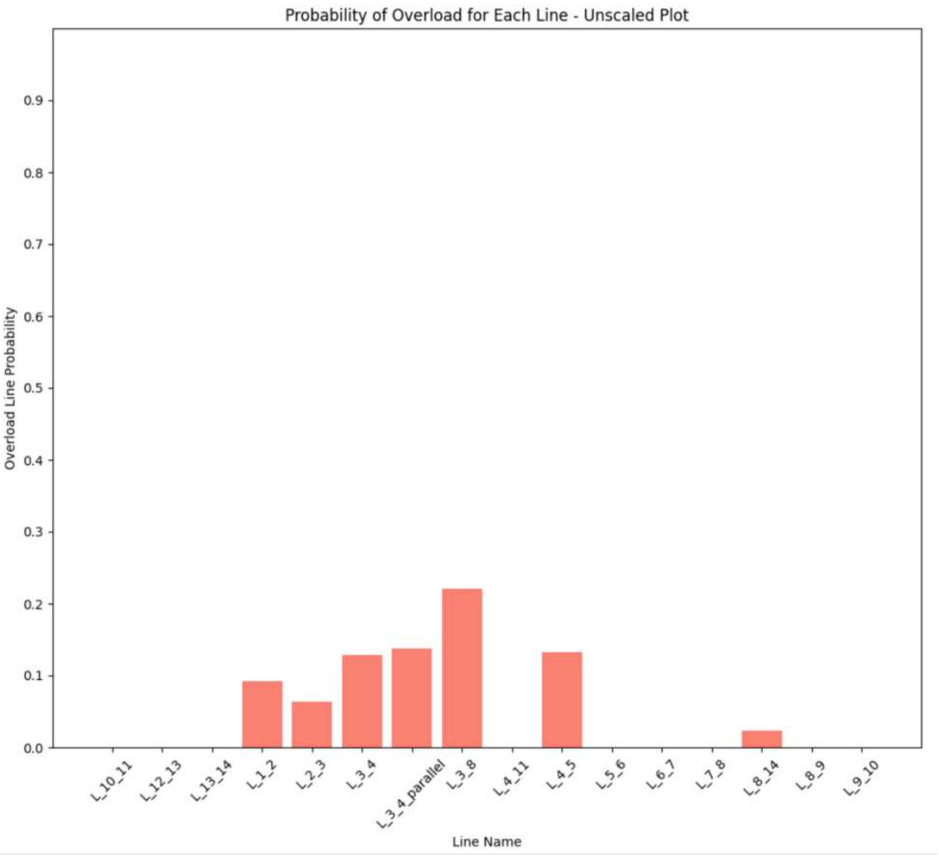


Figure 4.5.: Probability of line overloads across 1000 Monte Carlo simulations

4.3. Results of the extended model with three interconnected Energy Communities

The three Energy Communities (EC1, EC2, and EC3) are interconnected via a common external grid node, which serves as a simplified representation of the higher-level transmission network. This connection allows for power exchange between the communities while maintaining their semi-independent operation in most scenarios. Figure A.1 in Appendix A illustrates this configuration in detail. The External Grid element ensures voltage reference and acts as a slack bus, enabling simulation of realistic grid-interconnected behavior.

The MC simulation is carried out similarly for this model, precisely as in Subsection 4.1. Based on the calculated mean value and standard deviation, 1000 iterations were performed. The results are presented in Figure 4.6.

As shown in the figure, most of the lines in EC1 demonstrate a high probability of overload, indicating potential reliability issues. Most of the lines in EC1 and several lines in EC3 operate in a potentially critical state in more than 50% of the simulated cases.

The naming of the lines has the following logic:

- Lines in EC1 contain only two numbers (e.g., **L__3__4**, **L__1__2**).
- Lines in EC2 end with the number 2 (e.g., **L__3__4__2**, **L__1__2__2**).
- Lines in EC3 end with the number 3 (e.g., **L__3__4__3**, **L__1__2__3**).

This naming logic helps to distinguish which EC each line belongs to.

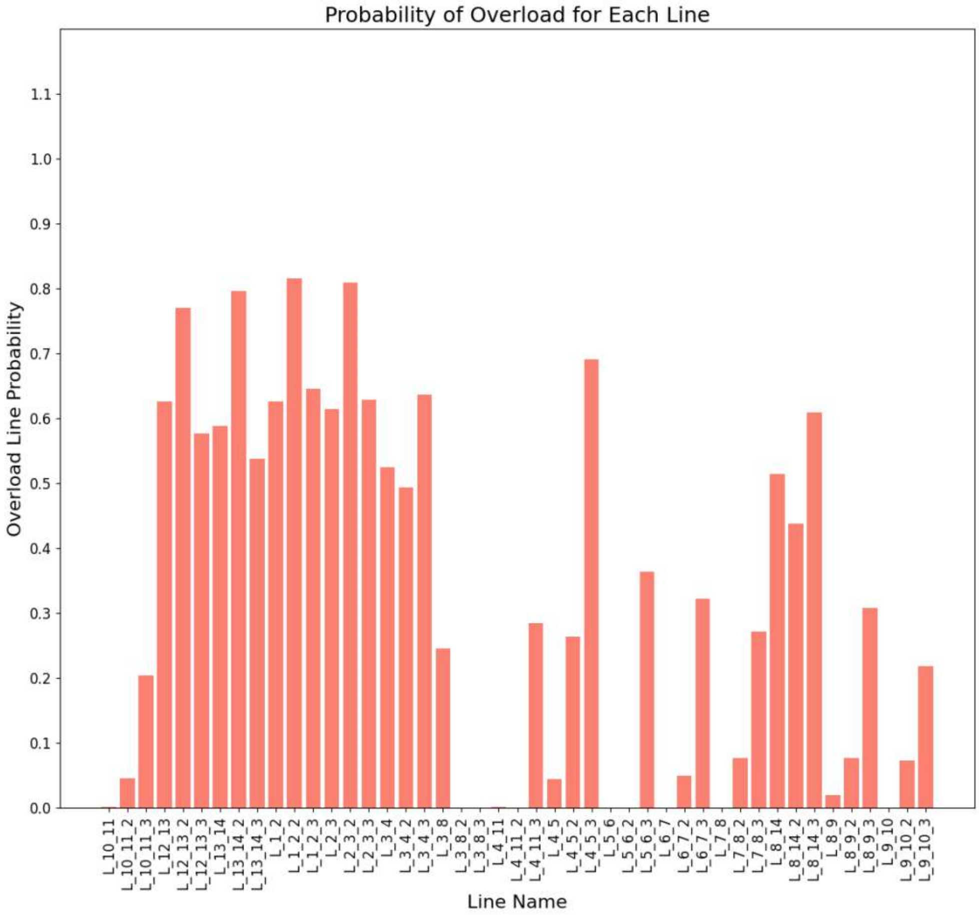


Figure 4.6.: Probability of line overloads across 1000 Monte Carlo simulations

Figure 4.7 shows the voltage magnitude at the buses. The logic of the bus names is similar – the last number represents the corresponding EC. Two different plots were built:

- Top chart: voltage magnitude in per unit (p.u.),
- Bottom chart: voltage magnitude in kilovolts (kV).

As the figure shows, EC1 experiences extreme voltage drops of over 50% on several buses.

The low voltage values in this scenario result from a deliberately non-optimized stress situation in which there is a sharp increase in load with no voltage regulation mechanisms. Since the model does not include reactive power compensation or voltage regulation devices (e.g., OLTCs or capacitor banks), the system exhibits significant voltage dips at high loads.

The model includes scenarios with a sharp increase in demand and insufficient local generation or voltage control. Since the network is weakly connected at the medium voltage level and most generation is decentralized, voltage violations on the buses can occur if the load exceeds local capabilities. Such unrealistic voltages were part of the motivation for applying optimization algorithms (PSO, GA). This simulation intentionally creates a **critical scenario** in which the network operates at its limits. This allows for identifying weak points in the system, assessing its resilience to overloads, and testing the effectiveness of optimization algorithms under extreme loads. Without optimization or redistribution, the extended system cannot actually maintain acceptable voltage levels at critical nodes. A detailed explanation is provided in Section 4.3.2.

The complete network topology of the interconnected energy communities, including the external network connection, is presented in Appendix A Figure A.1. As shown, the EC1, EC2, and EC3 models are connected via a **110kV** high-voltage bus **T0**. This centralized backbone structure enables the exchange of energy between communities and serves as a single point of interconnection.

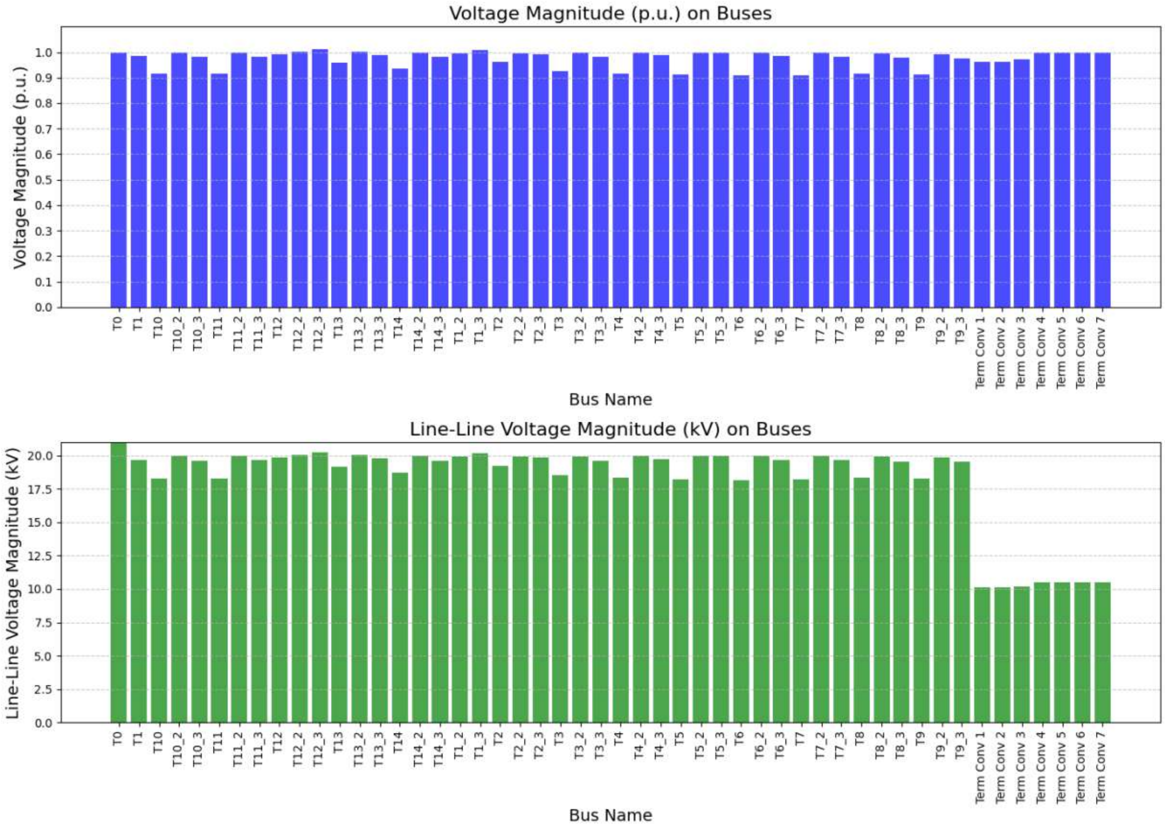
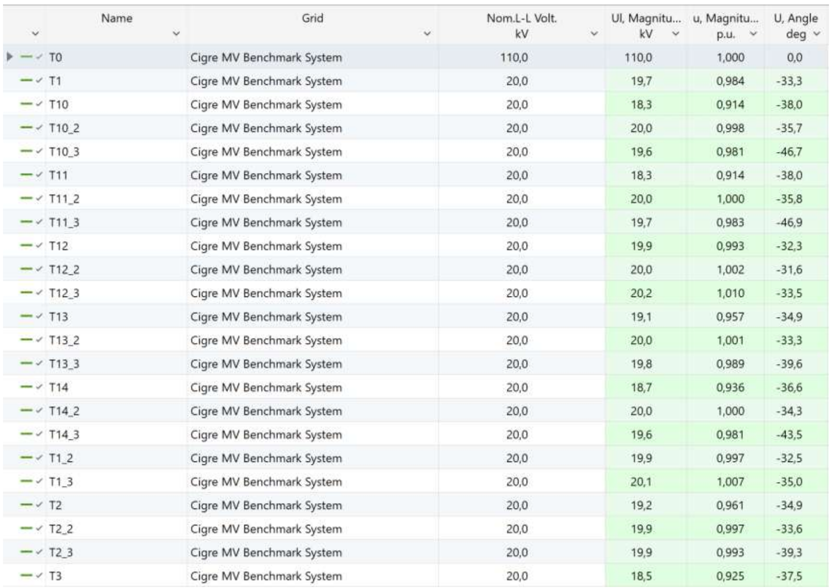


Figure 4.7.: Voltage magnitude at the buses

To confirm the data, Figure 4.8 shows the busbar voltage results from PowerFactory.



Name	Grid	Nom.L-L Volt. kV	Ul, Magnitu... kV	u, Magnitu... p.u.	U, Angle deg
T0	Cigre MV Benchmark System	110,0	110,0	1,000	0,0
T1	Cigre MV Benchmark System	20,0	19,7	0,984	-33,3
T10	Cigre MV Benchmark System	20,0	18,3	0,914	-38,0
T10_2	Cigre MV Benchmark System	20,0	20,0	0,998	-35,7
T10_3	Cigre MV Benchmark System	20,0	19,6	0,981	-46,7
T11	Cigre MV Benchmark System	20,0	18,3	0,914	-38,0
T11_2	Cigre MV Benchmark System	20,0	20,0	1,000	-35,8
T11_3	Cigre MV Benchmark System	20,0	19,7	0,983	-46,9
T12	Cigre MV Benchmark System	20,0	19,9	0,993	-32,3
T12_2	Cigre MV Benchmark System	20,0	20,0	1,002	-31,6
T12_3	Cigre MV Benchmark System	20,0	20,2	1,010	-33,5
T13	Cigre MV Benchmark System	20,0	19,1	0,957	-34,9
T13_2	Cigre MV Benchmark System	20,0	20,0	1,001	-33,3
T13_3	Cigre MV Benchmark System	20,0	19,8	0,989	-39,6
T14	Cigre MV Benchmark System	20,0	18,7	0,936	-36,6
T14_2	Cigre MV Benchmark System	20,0	20,0	1,000	-34,3
T14_3	Cigre MV Benchmark System	20,0	19,6	0,981	-43,5
T1_2	Cigre MV Benchmark System	20,0	19,9	0,997	-32,5
T1_3	Cigre MV Benchmark System	20,0	20,1	1,007	-35,0
T2	Cigre MV Benchmark System	20,0	19,2	0,961	-34,9
T2_2	Cigre MV Benchmark System	20,0	19,9	0,997	-33,6
T2_3	Cigre MV Benchmark System	20,0	19,9	0,993	-39,3
T3	Cigre MV Benchmark System	20,0	18,5	0,925	-37,5

Figure 4.8.: Voltage magnitude at the buses in PowerFactory

For example, bus **T14** has the voltage magnitude **18.7 kV**, which can be seen in Figure 4.7. In the Figure 4.9, a grid fragment from PowerFactory. This bus is underloaded.

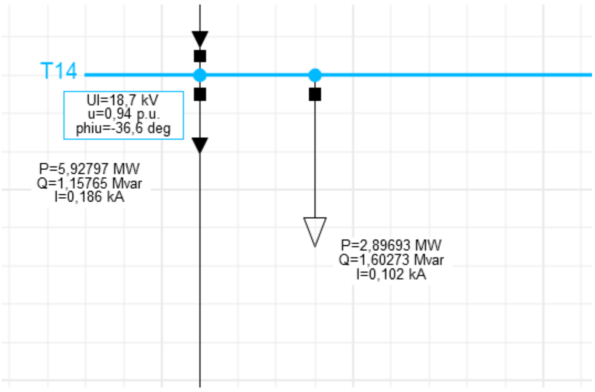


Figure 4.9.: Voltage magnitude at the bus T13 in PowerFactory

Appendix A.1 represents the power grid, and EC1 is indeed highlighted in red and blue, indicating bus overloads and voltage violations.

The results of the MC simulation confirm the presence of significant reliability problems within the power grid, which requires optimizing and stabilizing the entire system.

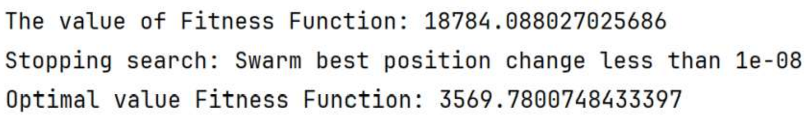
4.3.1. Application of PSO Optimization

To evaluate the instability of the current power grid, the FF was first calculated, resulting in a high value of **18784**, indicating significant system imbalance. After applying the PSO optimization algorithm, the FF was reduced to **3569**, demonstrating a substantial improvement in system stability. Although the FF does not directly correspond to a physical parameter, it serves as a metric to evaluate how far the system deviates from optimal operation. In real-world power systems, minor deviations are expected; thus, the objective is not to achieve a perfect zero but to ensure a reliable and stable operating point.

However, some errors are still present:

- PSO could not eliminate all overloads, but significantly reduced them.
- Voltage drops on the buses are present, but do not exceed unacceptable limits.
- Generators still operate at maximum power, but the limits are not exceeded.

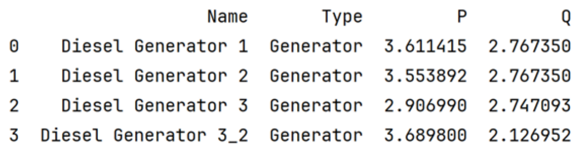
Results from Python are presented in Figure 4.10.



```
The value of Fitness Function: 18784.088027025686
Stopping search: Swarm best position change less than 1e-08
Optimal value Fitness Function: 3569.7800748433397
```

Figure 4.10.: Results of PSO from Python

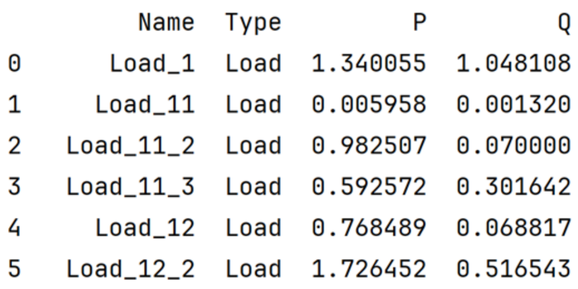
Table 4.11 and Table 4.12 represent the best new parameters for generators and loads found by PSO, under which the system doesn't have any technical issues.



```
                  Name       Type         P         Q
0    Diesel Generator 1  Generator  3.611415  2.767350
1    Diesel Generator 2  Generator  3.553892  2.767350
2    Diesel Generator 3  Generator  2.906990  2.747093
3  Diesel Generator 3_2  Generator  3.689800  2.126952
```

Figure 4.11.: Optimized parameters for generators

Note: **Diesel Generator 3_2** corresponds to the EC2.



```
        Name  Type         P         Q
0     Load_1  Load  1.340055  1.048108
1    Load_11  Load  0.005958  0.001320
2  Load_11_2  Load  0.982507  0.070000
3  Load_11_3  Load  0.592572  0.301642
4    Load_12  Load  0.768489  0.068817
5  Load_12_2  Load  1.726452  0.516543
```

Figure 4.12.: Optimized parameters for loads

These new parameters of generators and loads were exported to PF, where they were used for Load Flow analysis. After performing the Load Flow calculation, the system doesn't experience any technical issues. New optimized power grid from PF can be seen in Appendix A.2, indicating that there are no more overloaded or critical operation elements.

4.3.2. Visualization of PSO optimization results

Figure 4.13 presents the impact of PSO on line loading. The plot shows the reduction in line overload on transmission lines after optimization. A bar chart is used, where the blue bars show the loading values before optimization, and the orange bars show the values after optimization.

For example, the overload on **Line 2__3** was reduced from **12%** to **2%**.

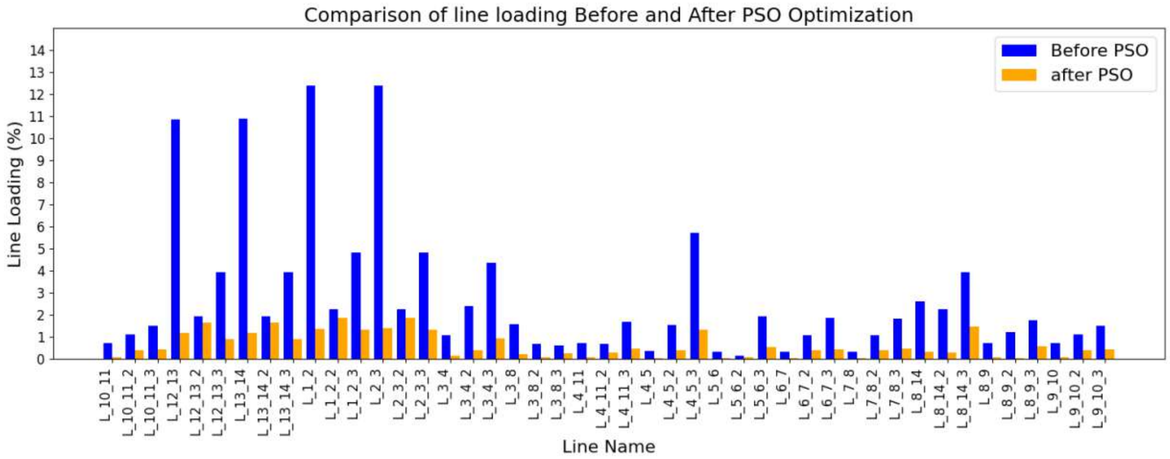


Figure 4.13.: Impact of PSO on line loading

Before the PSO optimization (Figure A.1), the power flows were primarily directed from the external grid through the upper buses and downward into EC1, supplying the loads along the path. This reflects a critical scenario in which the demand in EC1 exceeds the generation limits of the local diesel generators. As a result, additional energy is drawn from neighboring EC2 and EC3 via the external grid to compensate for the deficit in EC1.

Figures 4.14 and 4.15 show pieces of the electrical network at scale, with the directions of power flows indicated.

For example, in Figure 4.14, the active power flows on lines **L__1__2** (between **T1** and **T2**), **L__2__3**, and **L__3__4** are clearly directed toward the internal loads of EC1.

After applying PSO optimization in Figure 4.15, the direction of power flows changes. Power is now primarily supplied by local diesel generators within EC1, and flows are redirected outward, away from the generators and toward the distribution nodes. The algorithm effectively reduced the system load to acceptable limits (while respecting the 80% generation constraint), enabling EC1 to become self-sufficient.

As shown, the Line **L__1__2**, **L__2__3**, **L__3__4**, and **L__3__8** now carry power away from the generators in EC1. This redistribution reduces loading on key transmission lines, as seen in comparing line utilization before and after PSO.

That is, the algorithm reduced the generator power to a safe level and simultaneously redistributed the loads to ensure a power balance.

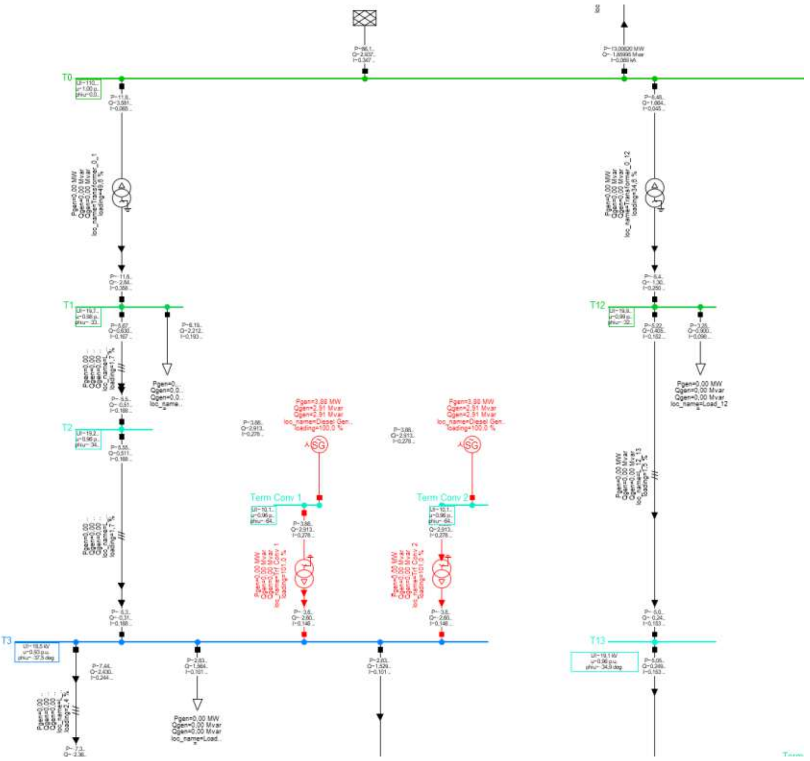


Figure 4.14.: Network fragment at scale before PSO optimization

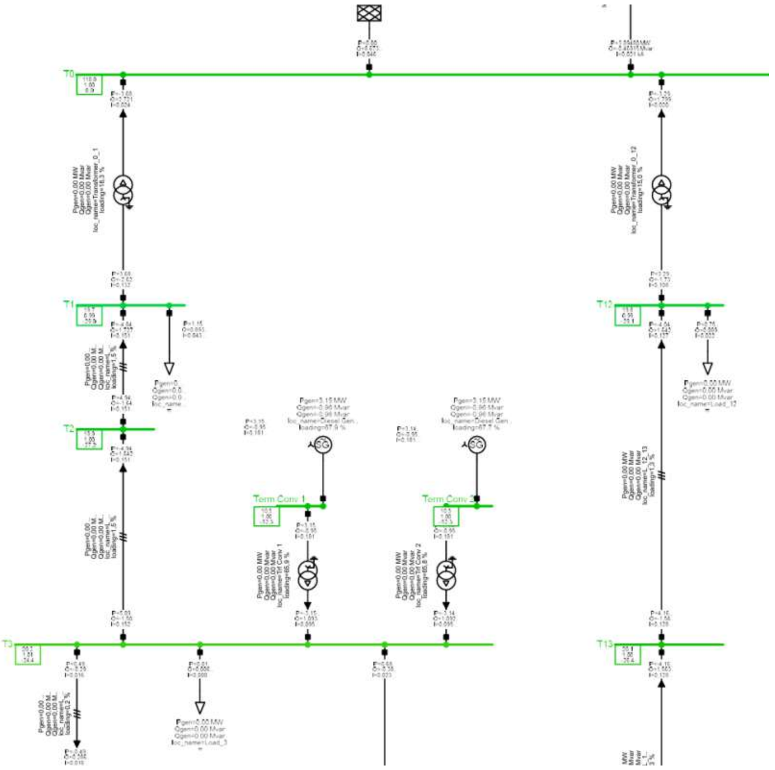


Figure 4.15.: Network fragment at scale after PSO optimization

In PowerFactory, the line loading percentage is defined as the ratio of the actual apparent power flow to the rated (thermal) apparent power capacity of the line. Mathematically, the loading L_i of line i is computed as:

$$L_i = \frac{|S_{\text{actual},i}|}{S_{\text{rated},i}} \times 100\%,$$

where:

- $|S_{\text{actual},i}| = \sqrt{P_i^2 + Q_i^2}$ is the magnitude of the actual apparent power (in MVA) flowing through line i,
- $S_{\text{rated},i}$ is the rated (thermal limit) apparent power capacity of the line, commonly referred to as `Rate A` in PowerFactory [42].

This means that a loading value of 100% corresponds to the maximum allowable thermal capacity of the line under normal operating conditions. The values presented in the optimization results (e.g., 10–12%) indicate that the lines are operating well below their thermal limits.

However, low thermal loading does not necessarily imply that the system is voltage-stable. Voltage violations can still occur due to poor reactive power distribution, inadequate voltage control, high penetration of distributed generation (e.g., PV), or weak network topology. Additionally, in this study, some scenarios involve sudden increases in active power demand, which locally stress the voltage profile and can cause temporary undervoltage situations. Therefore, voltage quality issues can emerge even when current loading values remain low, and it is possible to observe significant voltage drops or rises even when line loadings remain within safe thermal limits.

Figure 4.16 presents the impact of PSO on generator loading. The plot evaluates the efficiency of load redistribution among the available generators, demonstrating the effectiveness of PSO in balancing loads.

While the generator loading is visibly reduced after PSO optimization, the algorithm **simultaneously reduces** the total system load as shown in Figure 4.17. This is a key principle of the implemented strategy — the PSO adjusts both generation and load to maintain power balance while avoiding overloads and voltage violations.

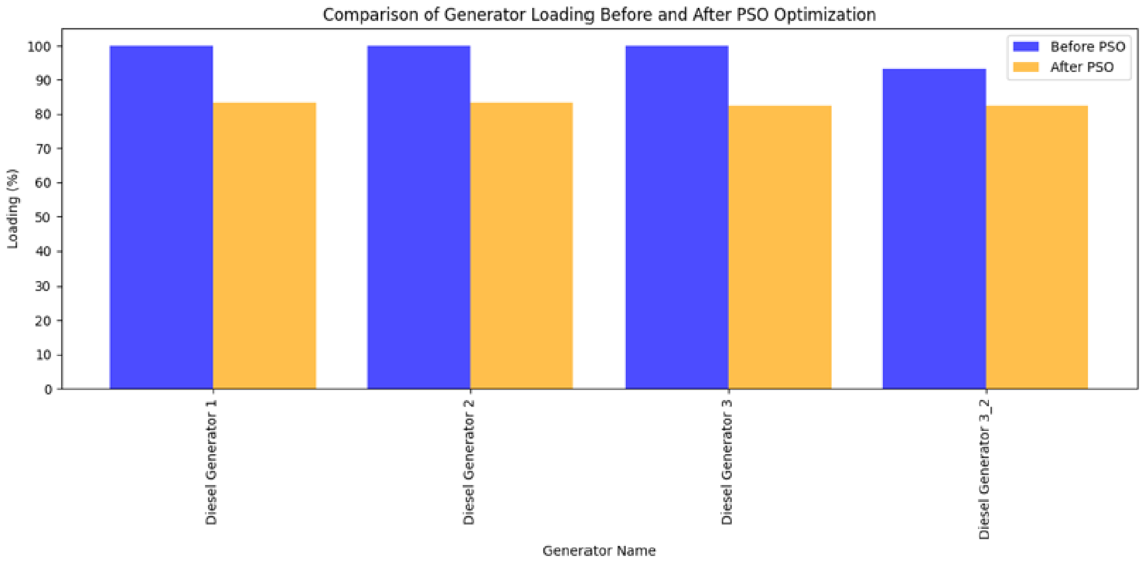


Figure 4.16.: Impact of PSO on generator loading

As can be seen, the generators' output was reduced to approximately **80%** of their capacity. A detailed explanation of this action is presented in section 3.2.6.

Figure 4.17 presents the impact of PSO on load redistribution. The algorithm performed load redispatch so that overloads and voltage violations did not emerge in the system. Under these parameters, the system operates stably.

For example, the **Load_5** was reduced from **4 MW** to **2 MW**, and the **Load_1_2** from **3.2 MW** to **1 MW**.

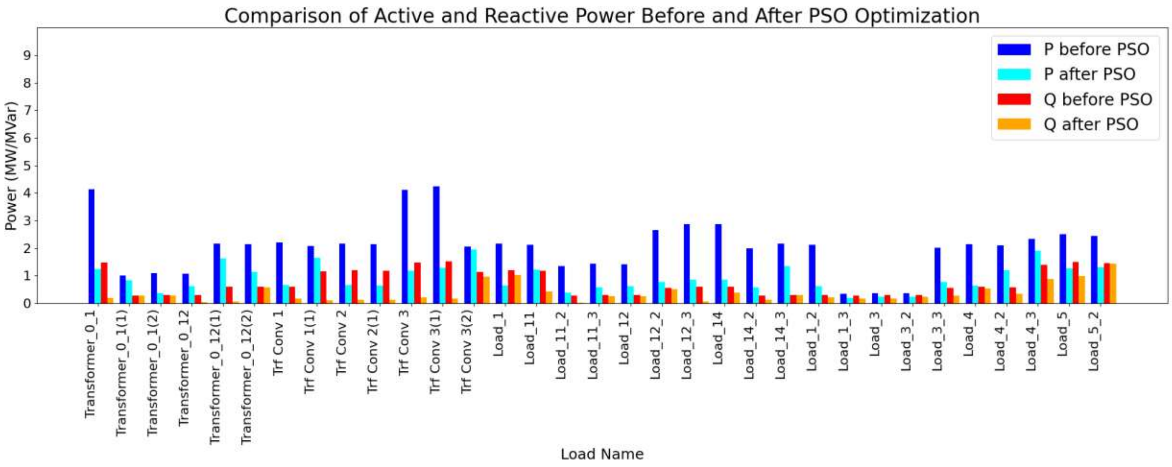


Figure 4.17.: Impact of PSO on loads redisctribution

Figure 4.18 demonstrates the impact of PSO on bus voltages. This plot evaluates the voltage levels on the power system buses before and after optimization. Before optimization, several buses experienced voltage violations, falling below the target level of 20 kV. After applying PSO, voltage levels were successfully stabilized.

For example, the voltage at **T10** and **T11** increased from **18.25 kV** to **20 kV** .

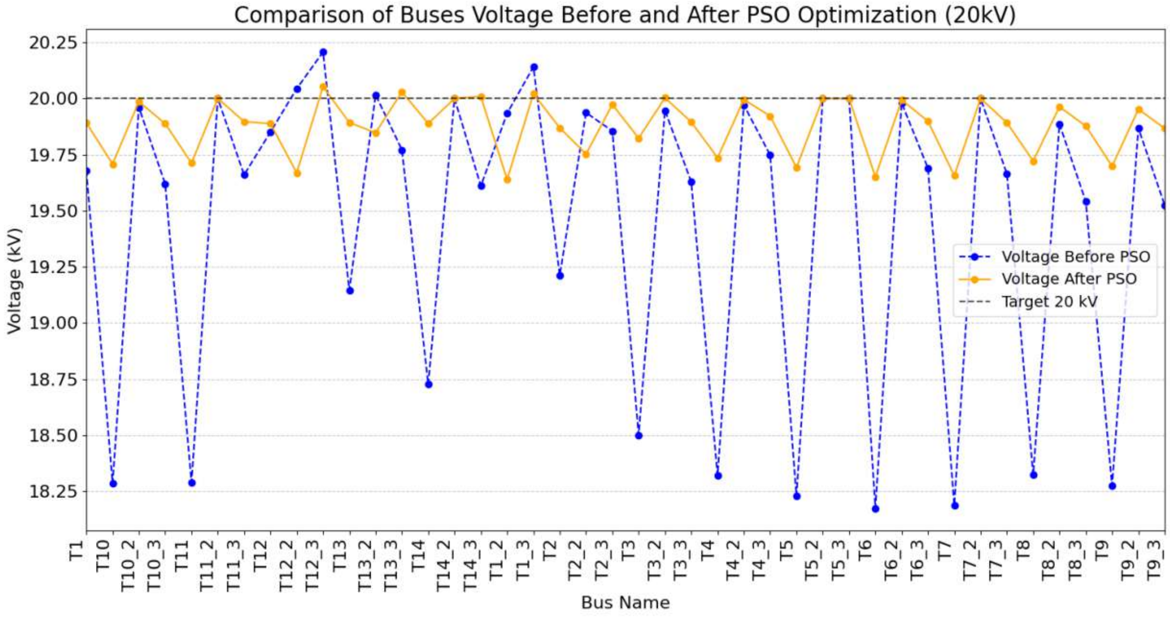


Figure 4.18.: Impact of PSO on bus voltages

4.3.3. Applying PSO optimization for 85% scenario

As an additional test of the algorithm's performance and proving its resilience, PSO was applied to a scenario with an 85% load and generation. The new power grid is presented in Appendix A.5. As you can see, there are no elements highlighted in red, which means that the new grid does not have any severe technical issues, however the optimization is needed to improve overall grid stability.

Figure 4.19 presents the impact of PSO on load redistribution.

For example, the **Load__5** was reduced from **2.4 MW** to **1.5 MW**, and the **Load__1__2** from **2.1 MW** to **0.95 MW**.

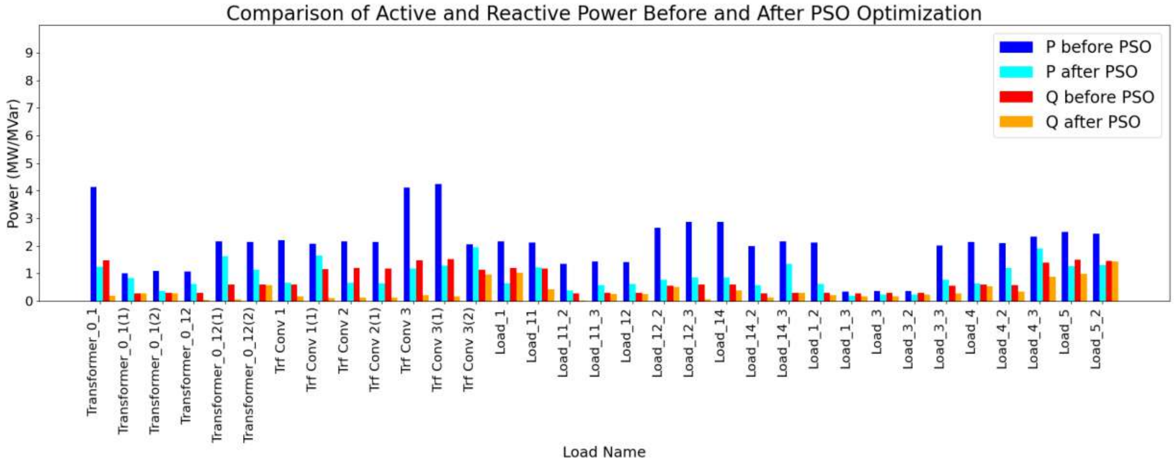


Figure 4.19.: Impact of PSO on loads redisctribution

Figure 4.20 demonstrates the impact of PSO on bus voltages. This plot evaluates the voltage levels on the power system buses before and after optimization. Before optimization, several buses experienced voltage violations, falling below the target level of 20 kV. After applying PSO, voltage levels were successfully stabilized.

For example, the voltage at **T6** and **T7** increased from **19.2 kV** to **19.7 kV** .

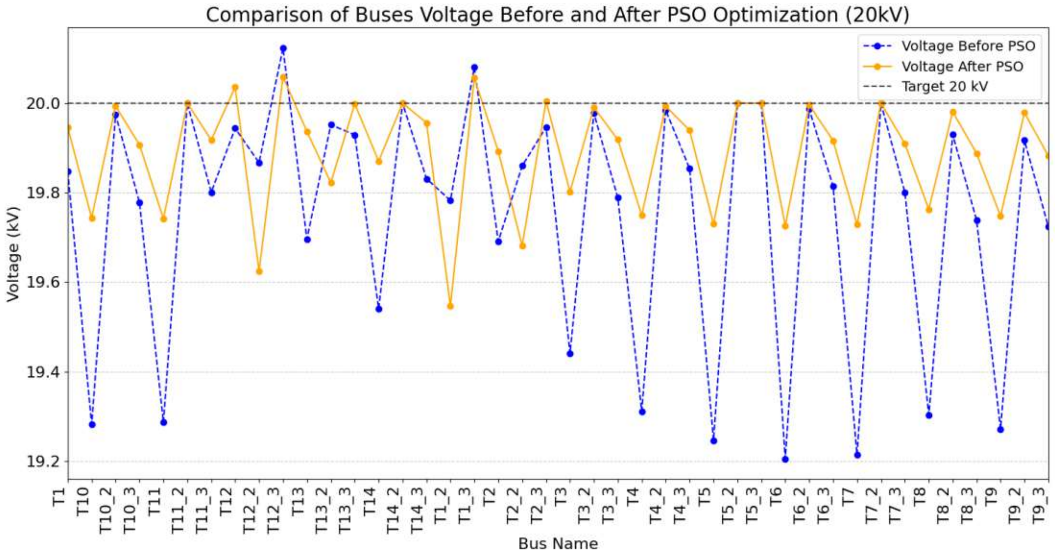


Figure 4.20.: Impact of PSO on bus voltages

According to European standards, particularly **EN 50160** ([50, 51]), which defines voltage characteristics in public electricity supply systems, the permissible voltage violations for medium-voltage grids (typically 1 kV to 36 kV) are as follows:

- The standard allows a voltage variation of ±10% of the nominal voltage under normal operating conditions.
- This means the voltage must remain within 90% to 110% of the nominal value (e.g., for a 20 kV system, the acceptable range is 18 kV to 22 kV).

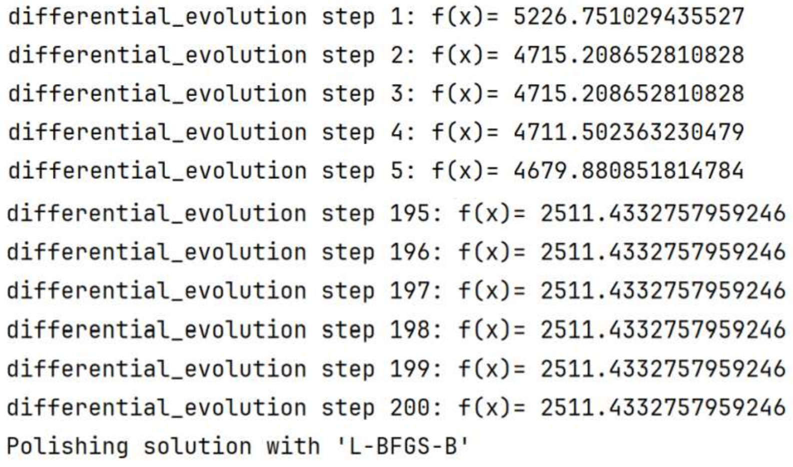
As can be seen from the figures 4.18 and 4.20, all bus voltages remain within acceptable limits, thus complying with the **EN 50160** standard.

4.3.4. Application of GA Optimization

The first step is evaluating the FF. Before applying GA optimization, the FF value remains **18784** and equals the initial state (as in PSO).

As specified in the code (see Figure 3.22), the number of iterations was set to **200**.

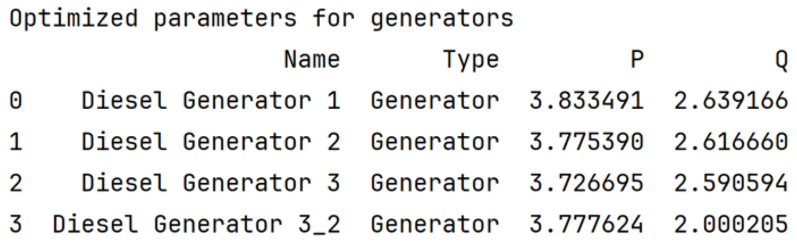
Figure 4.21 presents the results from Python during the first five and last five iterations. The FF gradually decreased, resulting in a value of **2511**.



```
differential_evolution step 1: f(x)= 5226.751029435527
differential_evolution step 2: f(x)= 4715.208652810828
differential_evolution step 3: f(x)= 4715.208652810828
differential_evolution step 4: f(x)= 4711.502363230479
differential_evolution step 5: f(x)= 4679.880851814784
differential_evolution step 195: f(x)= 2511.4332757959246
differential_evolution step 196: f(x)= 2511.4332757959246
differential_evolution step 197: f(x)= 2511.4332757959246
differential_evolution step 198: f(x)= 2511.4332757959246
differential_evolution step 199: f(x)= 2511.4332757959246
differential_evolution step 200: f(x)= 2511.4332757959246
Polishing solution with 'L-BFGS-B'
```

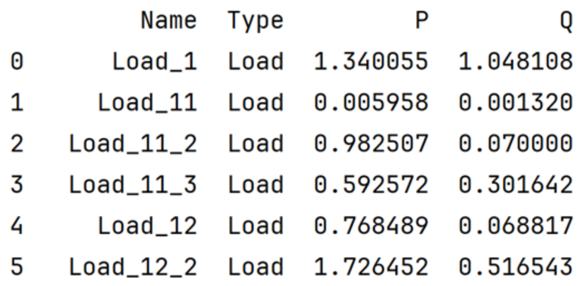
Figure 4.21.: Results of GA from Python during 200 iterations

Table 4.22 and Table 4.23 represent the best new parameters for generators and loads found by GA, under which the system has no technical issues.



```
Optimized parameters for generators
                   Name       Type         P         Q
0     Diesel Generator 1  Generator  3.833491  2.639166
1     Diesel Generator 2  Generator  3.775390  2.616660
2     Diesel Generator 3  Generator  3.726695  2.590594
3   Diesel Generator 3_2  Generator  3.777624  2.000205
```

Figure 4.22.: Optimized parameters for generators



```
       Name  Type         P         Q
0    Load_1  Load  1.340055  1.048108
1   Load_11  Load  0.005958  0.001320
2 Load_11_2  Load  0.982507  0.070000
3 Load_11_3  Load  0.592572  0.301642
4   Load_12  Load  0.768489  0.068817
5 Load_12_2  Load  1.726452  0.516543
```

Figure 4.23.: Optimized parameters for loads

The system operated without technical issues after performing a Load Flow calculation with these new parameters. New optimized power grid from PF can be seen in Appendix A.2, indicating that there are no more overloaded or critical operation elements.

4.3.5. Visualization of GA optimization results

Visualization of the GA optimization results for line loading, load redistribution, and bus voltages shows patterns almost identical to those achieved with PSO. Therefore, the plots are not included again to avoid redundancy. Detailed illustrations of such effects can be seen in Figures 4.13, 4.17, and 4.18.

However, the plot of generator power distribution in Figure 4.24 differs from the previous one in Figure 4.16. After GA optimization, generators operate at an average load of **60%**, compared to an average of **80%** with PSO.

This plot shows the better efficiency of load redistribution compared to PSO.

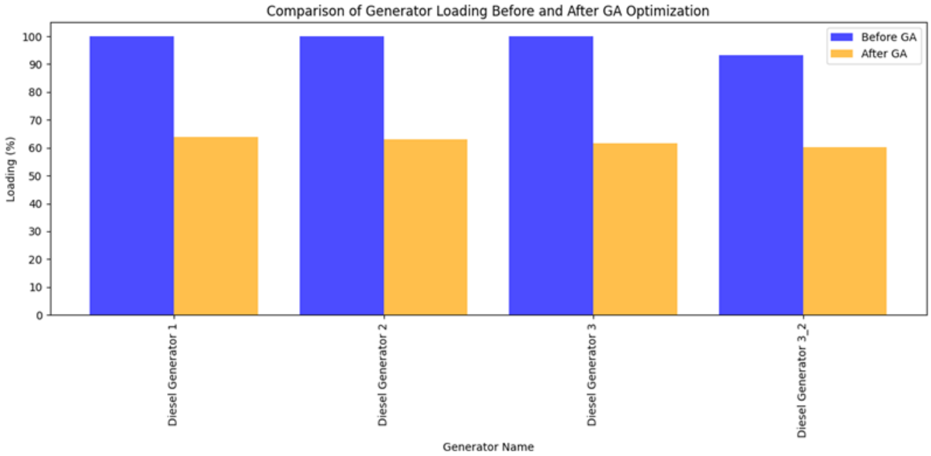


Figure 4.24.: Impact of GA on generator loading

The observed reduction in generator load and line load is due to the fact that the GA algorithm simultaneously regulates generator outputs and consumption, ensuring that the power balance is maintained.

4.4. Technical comparison of two algorithms

Both GA and PSO are meta-heuristic optimization techniques, but they differ in their approach:

- PSO represents swarm intelligence, where particles update their positions based on individual and global best solutions.
- GA represents evolutionary principles (selection, crossover, and mutation) to generate improved solutions.

GA does not require the function to be continuous or differentiable because it explores many possible solutions randomly and through evolutionary operations (selection, crossover, mutation). In contrast, PSO updates particles based on gradient directions and is better suited for smooth and continuous functions.

Table 4.1 presents the main parameters for comparison of the two algorithms.

Table 4.1.: Technical comparison between PSO and GA

Method	Fitness Function	Convergence Speed	Local Optima
PSO	3569	Medium (10 seconds)	High
GA	2511	Slow (3 minutes)	Low

PSO typically converges faster but may get stuck in local optima, while GA is better suited for global optimization but can take significantly longer to compute, especially for **200** iterations. GA found a better FF value of **2511**, compared to **3569**, optimizing the system. However, since both algorithms reached values near the global minimum, the overall difference in results is slight, so both approaches can be considered adequate.

Interestingly, their core optimization strategies differ: for optimization, PSO starts from a random position in space and then gradually improves the global solution by following the best particle. In contrast, GA immediately begins with two parent solutions with the best characteristics. This means that, after the first iteration, PSO still has a high FF value, while GA already has a low value. This can be seen from Figure 4.21, where exactly after the first iteration, the FF value drops significantly from **18784** to **5226**. This presents the core difference between the two algorithms.

4.4.1. Visualization of technical comparison

To better understand and visualize the results, plots were built for lines, generators, and bus voltages. Each plot includes three parameters: before optimization, after GA, and after PSO.

Figure 4.25 shows that the orange bars (after GA) are, in most cases, lower than the green bars (after PSO). The GA optimization showed better results, reducing line overloads more effectively.

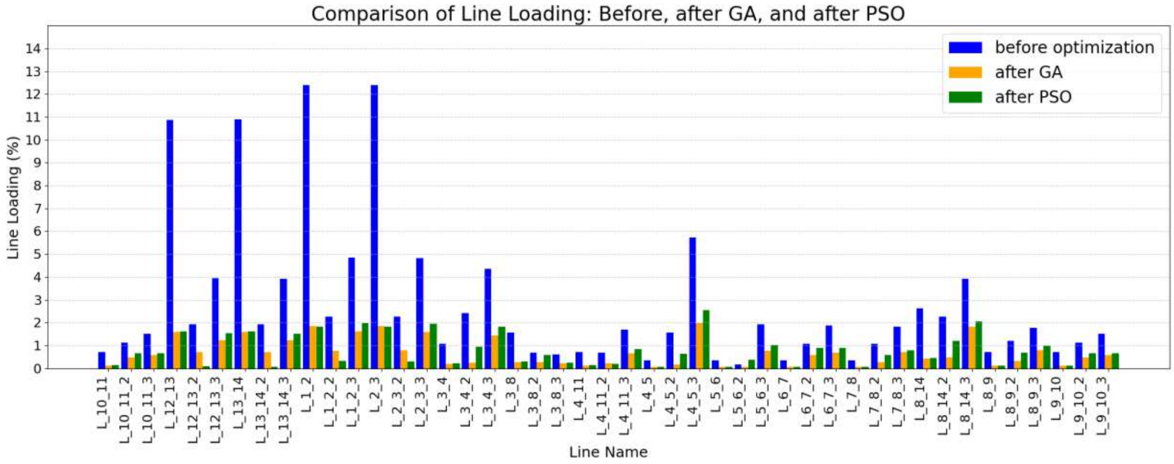


Figure 4.25.: Comparison of line loading bar chart

Figure 4.26 represents a pie chart with a percentage comparison of cases in which GA outperformed PSO. As shown, GA reduced line loading more effectively in **84.4%** of the cases.

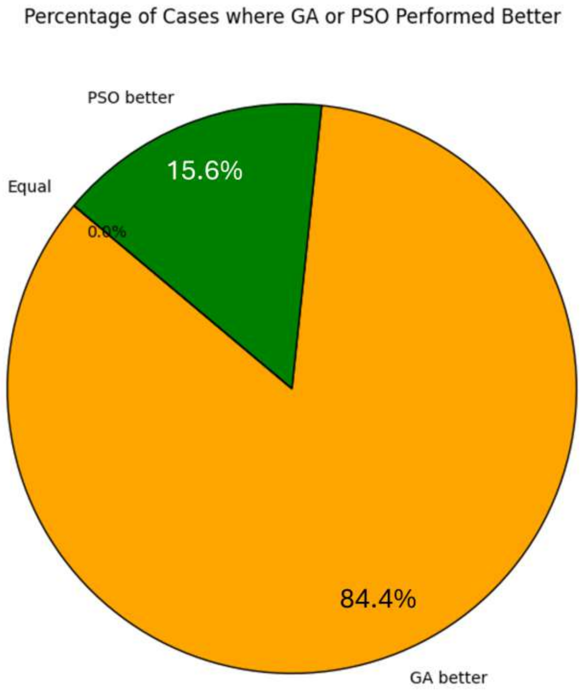


Figure 4.26.: Percentage comparison of line loading pie chart

Therefore, GA is more efficient in optimizing line loading.

Figure 4.27 presents the generator loading. In both cases, the generators do not experience overload operating conditions.

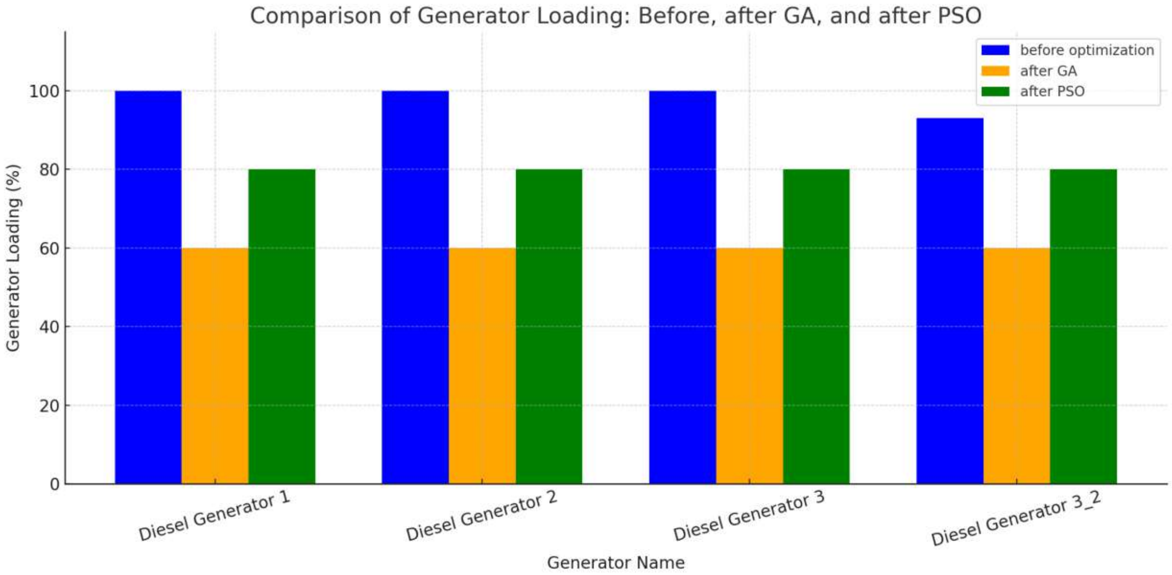


Figure 4.27.: Generator loading comparison

Although the total generator output power decreases after optimization, the system remains balanced. This is because the optimization algorithm simultaneously reduces the load (as shown in the Figure 4.28) values on certain buses to maintain power balance. This way, the total demand is adapted to the available generation power.

Figure 4.28 shows the comparison of active and reactive power of the loads. The power was reduced to approximately the same values in both algorithms. This indicates that both GA and PSO found the minimum acceptable load level, considering all the system's technical limitations.

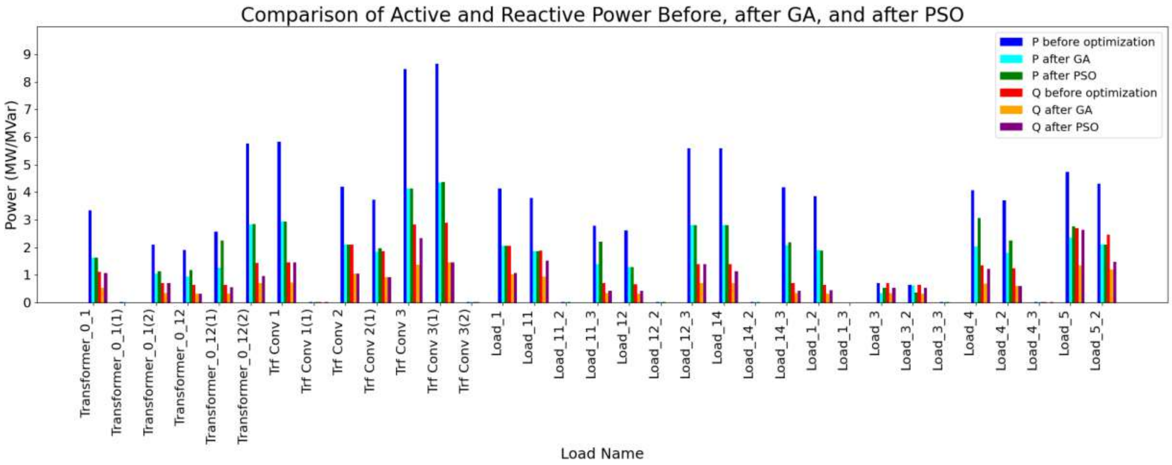


Figure 4.28.: Active and reactive power comparison bar chart

Figure 4.29 clearly shows the percentage comparison. As shown, GA performed slightly better than PSO in active power reduction, while GA significantly outperformed PSO in reactive power, achieving better values in **87.9%** of the cases.

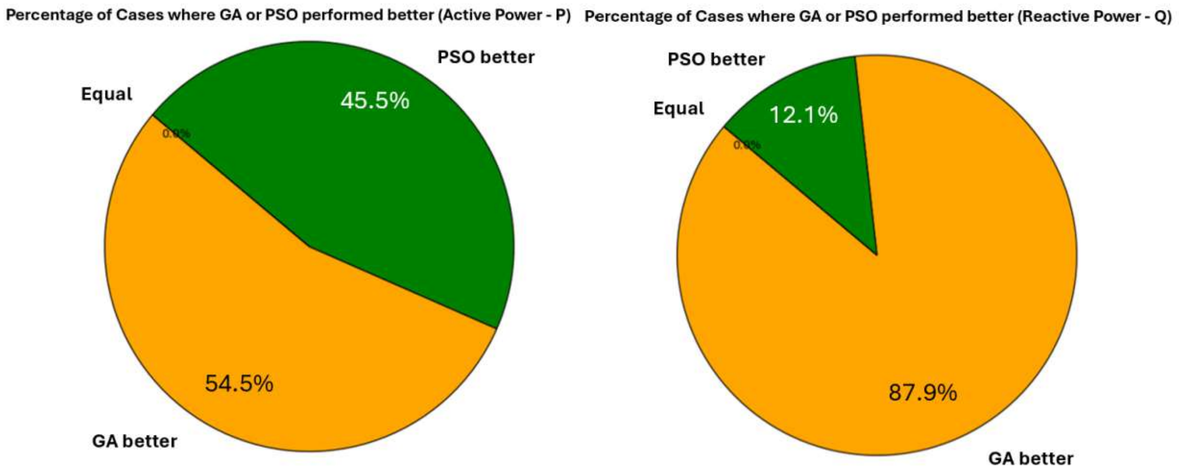


Figure 4.29.: Active and reactive power comparison pie chart

Figures 4.30 and 4.31 illustrate busbar voltage violations. The voltage levels are nearly equal in both cases and do not exceed the acceptable limits. This confirms that both algorithms effectively maintain voltage stability in the system.

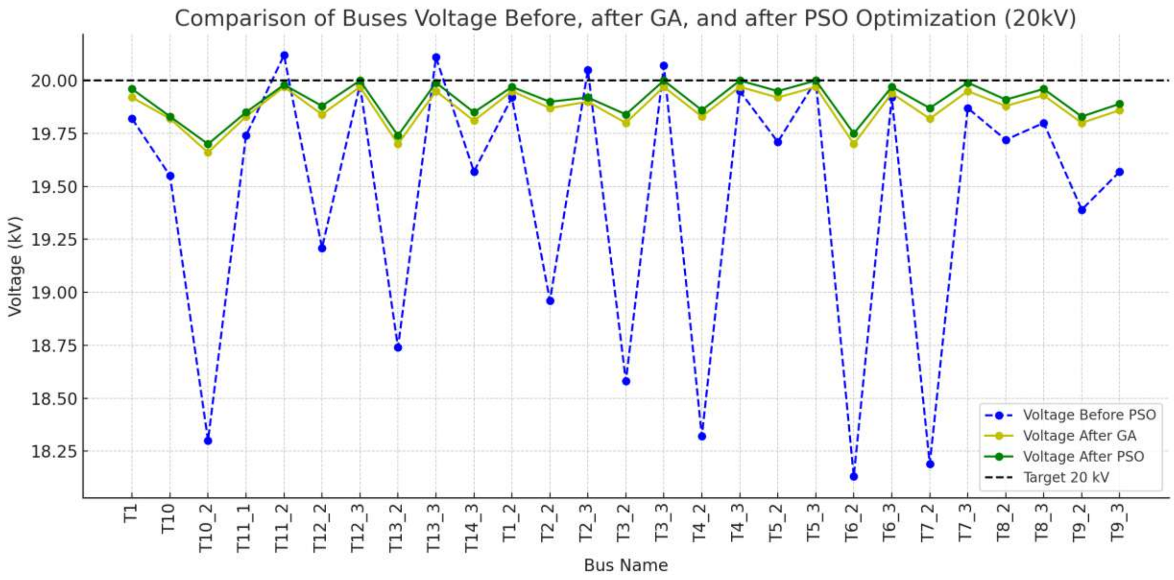


Figure 4.30.: Bus voltage comparison 20 kV

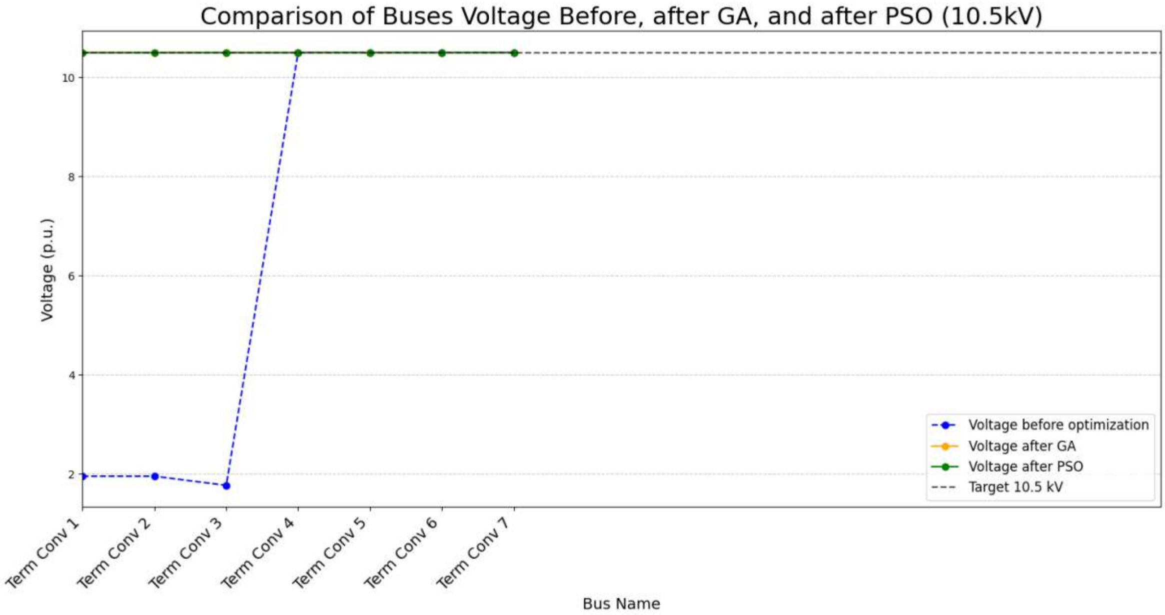


Figure 4.31.: Bus voltage comparison 10.5 kV

Thus, both algorithms effectively executed load dispatch, minimized line overloading and voltage violations, and attained a more balanced generation distribution.

4.4.2. Economic comparison of PSO and GA

The Day Ahead Price **150 EUR/MWh** (23.02.2025) of electricity was taken from APG [52]. The example of calculation for GA is presented in Figure 4.32. The calculation for PSO is identical.

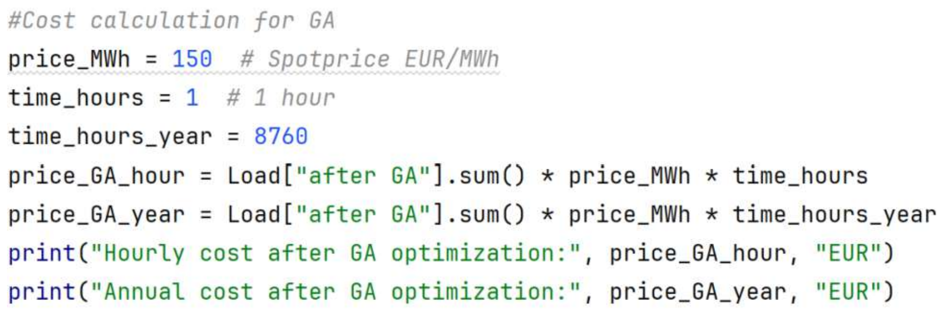

```
#Cost calculation for GA
price_MWh = 150  # Spotprice EUR/MWh
time_hours = 1  # 1 hour
time_hours_year = 8760
price_GA_hour = Load["after GA"].sum() * price_MWh * time_hours
price_GA_year = Load["after GA"].sum() * price_MWh * time_hours_year
print("Hourly cost after GA optimization:", price_GA_hour, "EUR")
print("Annual cost after GA optimization:", price_GA_year, "EUR")
```

Figure 4.32.: Example for calculation from Python

The total cost of electricity consumed after optimization was calculated for both algorithms, considering the spot price of electricity. This calculation was performed by summing up the entire load after optimization and multiplying it by the spot price. The calculation was conducted for both hourly and annual cases.

The first case provides an estimate of the immediate effect of optimization, and the second case allows for estimating the global economic impact, which is important for long-term analysis. The results are shown below:

- Hourly cost after GA optimization: **4,347 EUR**

- Annual cost after GA optimization: **38.079 million EUR**
- Hourly cost after PSO optimization: **4,959 EUR**
- Annual cost after PSO optimization: **43.441 million EUR**

Figure 4.33 shows a bar plot with a comparison of two algorithms.

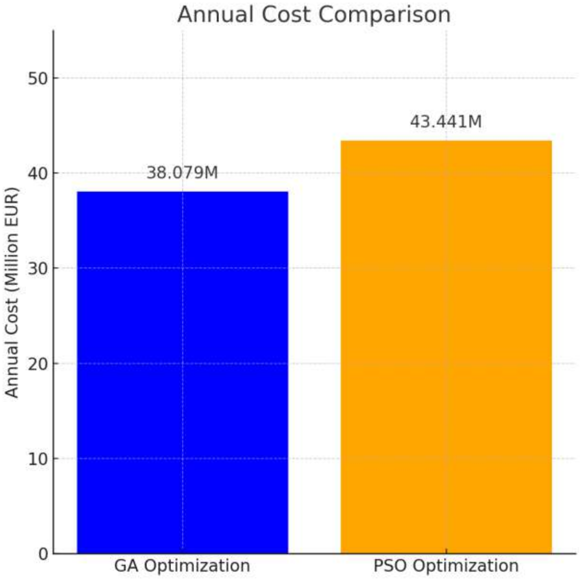


Figure 4.33.: Annual cost comparison

The annual difference is around **5.362 million EUR** and shows that the economic effect could be significant in the long term.

4.5. Applying of ANN

As discussed previously, the ANN was trained on the optimized results from PSO data (from two .csv files) (see Figure 3.23) and will try to predict the output. Once trained, the ANN can, in a short time, predict the optimal $P_{\mathbf{gen}}$ value, which is calculated based on the given $P_{\mathbf{load}}$ without running complex iterative optimization each time. However, the ANN algorithm is applicable only for minor disturbances.

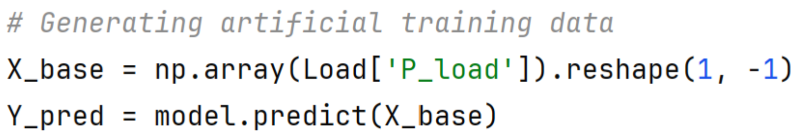
The algorithm for checking the functionality of the ANN is as follows:

1. New loads were created manually. In Table 4.2, previous and new load parameters are presented. For example, new loads are now 80% of the initial value.

Table 4.2.: Active power of loads

P_{initial}	**80%**
1.893142	1.5145136
0.00866921	0.006935368
0.6547431	0.52379448
0.5737207	0.45897656
0.9410952	0.75287616
4.17507	3.340056
1.751664	1.4013312
0.00722706	0.005781648

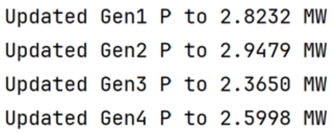
2. New loads can be inserted in PF.
3. Already learned and trained ANN is used for the prediction of new parameters for generators. From Figure 4.34, it can be seen that $X_{\textbf{base}}$ serves as data input and imports data directly from PF. The variable $Y_{\textbf{pred}}$ is then used to predict the new generator outputs.



```
# Generating artificial training data
X_base = np.array(Load['P_load']).reshape(1, -1)
Y_pred = model.predict(X_base)
```

Figure 4.34.: Generator values prediction

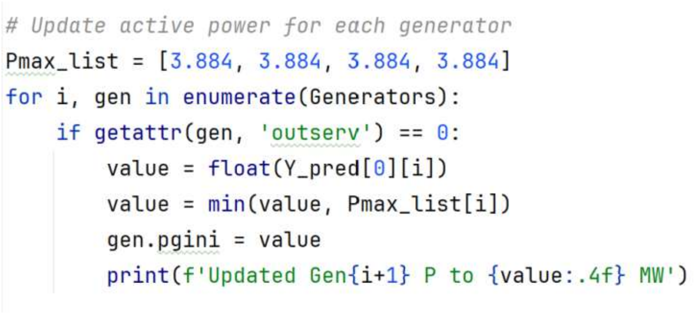
4. The predicted active power values for generators $Y_{\textbf{pred}}$ are presented in the Figure 4.35.



```
Updated Gen1 P to 2.8232 MW
Updated Gen2 P to 2.9479 MW
Updated Gen3 P to 2.3650 MW
Updated Gen4 P to 2.5998 MW
```

Figure 4.35.: Updated generator parameters in Python

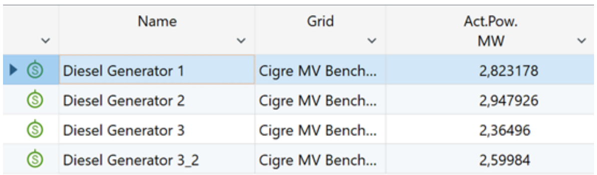
These data can now be directly inserted into PF for further Load Flow analysis. This occurs automatically using a block in the Figure 4.36.



```
# Update active power for each generator
Pmax_list = [3.884, 3.884, 3.884, 3.884]
for i, gen in enumerate(Generators):
    if getattr(gen, 'outserv') == 0:
        value = float(Y_pred[0][i])
        value = min(value, Pmax_list[i])
        gen.pgini = value
        print(f'Updated Gen{i+1} P to {value:.4f} MW')
```

Figure 4.36.: Updated generator parameters in Python

Updated data in the PF interface are represented in the Figure 4.37.

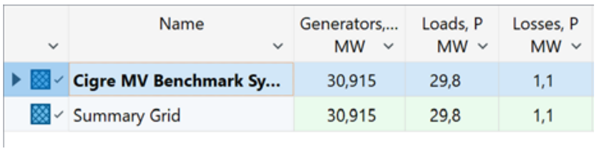


	Name	Grid	Act.Pow. MW
▶	Diesel Generator 1	Cigre MV Bench...	2,823178
	Diesel Generator 2	Cigre MV Bench...	2,947926
	Diesel Generator 3	Cigre MV Bench...	2,36496
	Diesel Generator 3_2	Cigre MV Bench...	2,59984

Figure 4.37.: Updated generator parameters in PowerFactory interface

5. Performing Load Flow calculation to confirm that the model works without technical issues (see Appendix A.2). This confirms the ANN can respond to minor disturbances and automatically re-dispatches generator output, considering all technical limitations.

The Figure 4.38 illustrates the quality assessment of load and generation balancing.



	Name	Generators,... MW	Loads, P MW	Losses, P MW
▶ ▨ ✓	**Cigre MV Benchmark Sy...**	30,915	29,8	1,1
▨ ✓	Summary Grid	30,915	29,8	1,1

Figure 4.38.: Power balance in PowerFactory after applying ANN-predicted generator setpoints

- Total active generation = **30.915 MW**
- Total load + losses = 29.8 + 1.1 = **30.9 MW**

This confirms that the balance is nearly perfect ($\Delta P \approx \mathbf{0.015\ MW}$), validating the correctness of the ANN.

The value of MAE can also be calculated, which represents the comparison of the absolute error (**0.015 MW**) with the total load (**29.8 MW**) or how accurately the algorithm matched the new generation values to the consumption:

$$MAE = \frac{0.015}{29.8} \cdot 100 = \mathbf{0.0503\%} \tag{4.1}$$

The deviation represents less than **0.05%** of the total system load, demonstrating that the ANN redispatch was performed with high precision [53].

Conclusions of the optimization chapter:

In this chapter, PSO and GA algorithms contributed to the Energy Management System (EMS) by performing **load management**, optimally adjusting demand and generation, and considering all technical system constraints. Both algorithms successfully minimized transmission line overloads and improved system stability.

Further comparisons demonstrated that GA required more computational time but provided slightly better fitness function results, making it a viable alternative for solving complex OPF problems.

As a final stage, an ANN was trained on optimized PSO data. Once trained, it allowed the system to react quickly to minor disturbances. The ANN found new generator parameters and ensured the balance between supply and demand, avoiding calculating long and complex meta-heuristic algorithms. This demonstrates the potential of the ANN as a fast and reliable tool for real-time OPF applications.

5. Synthesis of results

5.1. Reference to the Research Question

The research aimed to develop a robust optimization framework for three ECs interconnected through an external grid. Many RES, especially PV installations, can lead to overloaded conditions. Two classical meta-heuristic (PSO, GA) and one machine learning (ANN) approach were selected for power and load dispatch.

The aim was successfully achieved. By comparing results from tables or PF models, it was proved that all line overloads and voltage violations in the system were reduced, and overall grid stability was improved. The optimization algorithms ensured that the power balance was preserved: load levels were dynamically adjusted in line with the reduced generator capacities, avoiding excess demand and maintaining system equilibrium. Moreover, the initial value of FF in both algorithms was also significantly reduced, indicating a decrease in technical stress in the power grid.

5.2. Transferability and Scalability of Results

As previously discussed, this framework can be utilized by grid operators in power plants to rapidly respond to energy disturbances and stabilize the system within seconds. Once trained, the ANN model is capable of conducting real-time generator dispatch.

Since the entire model was built in PF, it can be easily extended to more complex systems, including even more ECs with other electrical elements. Importantly, this does not require reconfiguring the Python code, as the model is deeply integrated with Python. All system parameters are imported into the code and can be used for further optimization. All implemented tools (MC, PSO, GA, ANN) are automated and linked.

Moreover, by changing different key parameters—such as the **threshold** for MC simulation, **number of iterations** for PSO and GA, or the target **date in the future** for PV output (see 3.4.3 Random Forest Regression algorithm)—the system can be broadly and deeply investigated in multiple scenarios, considering numerous factors.

5.3. Strengths and limitations of the methodology

Table 5.1 compares the three implemented algorithms based on their FF performance, convergence speed, and ability to escape local optima.

Table 5.1.: **Comparison of implemented algorithms**

Method	Fitness Function	Convergence speed	Local optima
PSO	3569	Medium (10 seconds)	High
GA	2511	Slow (3 minutes)	Low
ANN	-	Instant	Low

PSO typically converges faster but may become stuck in local optima, while GA is better suited for global optimization but can take significantly longer to compute, especially for 200 iterations. GA found a better FF value. ANN requires pre-optimized data from either PSO or GA for training. However, it cannot be applied to severe technical disturbances without retraining.

Thus, while PSO and GA are more suitable for optimizing full-scale systems, ANN can be used in real-time to balance power supply and demand under mild fluctuations.

Moreover, as seen before, PSO is a more favorable approach for covering energy demand. However, if the priority is loss minimization and generation efficiency, GA works better, although it leads to lower load satisfaction.

5.4. Acquired skills

While working on this project, several important engineering and analytical skills were acquired. First, strong programming skills in Python and confident use of PF for power system modeling and simulation were achieved. Using these two programs in combination has expanded their initial boundaries and allowed them to investigate the power grid more deeply and perform more serious calculations that would be impossible manually (e.g., 1000 MC load flow calculations).

A deep understanding of various optimization algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Artificial Neural Networks (ANN), was gained, enabling the development and testing of complex operational strategies. These skills are highly transferable and applicable to real-world power systems and control centers.

MC simulation provided insights into power grid reliability analysis. Additionally, the project included machine learning, specifically forecast regression algorithms, to predict future PV outputs, contributing to better system planning and predictability.

An important lesson was also understanding which algorithm is appropriate for which scenario. For example, an ANN is best suited for real-time use under moderate disturbances, while PSO and GA are more suitable for solving complex optimization problems under critical system conditions.

Overall, this work provided a holistic view of modern energy system optimization and demonstrated how classical methods and AI can complement each other in developing future-ready energy solutions.

6. Conclusions

6.1. Appropriateness of the research question and methodology

The primary research question was how to ensure the reliable and sustainable operation of ECs with a high share of RES. To address this, a robust optimization framework was required. Since traditional deterministic optimization algorithms are not suitable for complex, dynamic systems, two meta-heuristic algorithms (PSO and GA) and one machine learning algorithm (ANN) were implemented.

PSO and GA were used for full-scale optimization under various operating scenarios, while ANN was used as a fast-response tool for daily power balancing fluctuations. Both numerical results and visualizations confirmed that these algorithms optimized the system and ensured optimal power flow.

6.2. Lessons learned from the synthesis of results

MC simulation can be a powerful tool for predicting potential vulnerabilities in a system. Moreover, as proven above, it can be successfully applied to simplified and complex systems. The possibilities are especially expanded when combining Python and PF.

Meta-heuristic algorithms inspired by natural and biological processes (PSO and GA) successfully fulfilled the network optimization tasks. The FF values were significantly reduced in both cases, indicating improved operating conditions and enhanced system stability.

6.3. Overcoming the Limitations in Future Work

Although the implemented framework demonstrated high efficiency in power flow optimization and generation redistribution in various scenarios, certain limitations were identified that may guide future research and improvements.

First, the ANN model cannot provide accurate results under significant system disturbances. To overcome this limitation, the model can be trained on a broader and more diverse dataset, not only on PSO, as in this research, but also on other algorithms under all possible system operation scenarios. Thus, ANN will be able to analyze a larger number of patterns and will not go beyond the usual values.

Second, despite its fast convergence, the PSO algorithm may become stuck in local optima. This limitation can be addressed by applying hybrid approaches, such as combining PSO with GA or reinforcement learning (RL) methods, which enable fast exploration and improved global search capabilities.

Third, the GA computation time increases significantly with the number of iterations. Future research may consider integrating parallel computing or cloud optimization techniques to reduce computational overhead and improve scalability.

Finally, while the integration between Python and PF has been successfully established, the framework can be even more flexible by connecting it to external data sources such as SCADA systems or weather APIs. This will enable real-time predictions, thereby improving the accuracy and applicability of the optimization.

An additional extension of the study could be to include multi-energy systems, where renewable sources such as wind, biomass, or hydrogen could be trained using an ML approach to predict the combined output of these sources, with optimization algorithms able to maintain a stable operation and balance between generation and consumption in the power grid.

6.4. Optimization of Peer-to-Peer (P2P) Energy Trading

Peer-to-peer (P2P) energy trading allows members of ECs to exchange surplus renewable energy directly with each other. Optimization techniques such as PSO, GA, and alternative metaheuristics such as Artificial Bee Colony (ABC) and Gray Wolf Optimization (GWO) can be applied to improve the efficiency of such transactions. These algorithms support the development of adaptive pricing mechanisms and optimal trading strategies tailored to local supply and demand conditions.

6.5. Policy and Regulatory Alignment in the EU Context

Future research can address regulatory and policy-related challenges ECs face within the European Union. This includes aspects such as grid access, tariff structures, and participation in local energy markets. By incorporating predictive modeling and scenario analysis, the impact of different regulatory frameworks can be evaluated, providing valuable insights for aligning technical developments with evolving EU energy policies.

6.6. Socio-Economic and Environmental Impact Assessment

The research can be further extended to assess the broader socio-economic and environmental implications of deploying optimized PV systems within ECs. Machine learning models may be utilized to predict long-term energy savings and reductions in carbon emissions. In parallel, optimization algorithms can help evaluate cost-benefit trade-offs, supporting decision-making processes that balance individual incentives with collective sustainability goals.

6.7. Outlook on Future Work in this Field of Research

This research can be expanded much more widely and in-depth. For example, to predict the output of PV panels, other machine learning algorithms may be applied in place of Random Forest Regression to potentially improve accuracy, such as Extreme Gradient Boost (XG-Boost), Support Vector Regression (SVR), Long-Short-Term Memory Network (LSTM), and multiple linear regression (MLR). They can achieve more accurate results than Random Forest Regression.

Instead of PSO and GA, other equally powerful algorithms can be applied, such as Artificial Bee Colony (ABC), Gray Wolf Optimization (GWO), Flower Pollination Algorithm (FPA), Crow Search Algorithm (CSA), Group Search Optimization (GSO), Cuckoo Search Optimization, Moth-Swarm Algorithm (MSA), Bacterial Foraging Algorithm (BFA), SHADE, JAYA, and Moth-Flame Optimizer (MFO), among many others.

However, as an additional experiment, the parameters of PSO and GA's settings can be significantly changed, making results more accurate, but the calculation time will increase proportionally.

Additionally, the model could integrate real-time electricity market prices, and the ANN will instantly respond to fluctuations between load and generation, ensuring minimal costs.

Bibliography

[1] UNFCCC. The paris agreement. https://unfccc.int/process-and-meetings/the-paris-agreement, 2024. [Online; accessed 02/2025].

[2] Council of the European Union. The european green deal. https://www.consilium.europa.eu/en/policies/european-green-deal/, 2024. [Online; accessed 05/2025].

[3] European Commission. 2050 long-term strategy. https://climate.ec.europa.eu/eu-action/climate-strategies-targets/2050-long-term-strategy_en, 2024. [Online; accessed 02/2025].

[4] European Commission. Energy communities. https://energy.ec.europa.eu/topics/markets-and-consumers/energy-consumers-and-prosumers/energy-communities_en, 2025. Accessed: May 2025.

[5] European Commission. Identification of appropriate generation and system adequacy standards for the internal electricity market, 2015.

[6] UNFCCC. The kyoto protocol. https://unfccc.int/kyoto_protocol, 1997. [Online; accessed 03-May-2025].

[7] International Energy Agency. Germany's renewables energy act. https://www.iea.org/policies/12392-germanys-renewables-energy-act, 2021. [Online; accessed 03-May-2025].

[8] European Commission. Green paper on the security of energy supply. https://eur-lex.europa.eu/EN/legal-content/summary/green-paper-on-the-security-of-energy-supply.html?fromSummary=18, 2001. [Online; accessed 03-May-2025].

[9] Pinja. Renewable energy directive — what is the red and how does it affect industrial operators? https://blog.pinja.com/renewable-energy-directive, 2023. [Online; accessed 03-May-2025].

[10] Joint Research Centre (European Commission). Renewable energy recast 2030 (red ii). https://joint-research-centre.ec.europa.eu/welcome-jec-website/reference-regulatory-framework/renewable-energy-recast-2030-red-ii_en, 2023. [Online; accessed 03-May-2025].

[11] Austrian Federal Ministry for Climate Action. Red iii — renewable energy directive. https://energie.gv.at/glossary/red-iii, 2023. [Online; accessed 03-May-2025].

[12] Next Kraftwerke Austria. Erneuerbaren-ausbau-gesetz (eag) – Überblick, ziele, inhalte. https://www.next-kraftwerke.at/wissen/erneuerbaren-ausbau-gesetz-eag#ab-wann-gilt-das-erneuerbaren-ausbau-gesetz-eag, 2025. Accessed: May 2025.

[13] E-Control Austria. Energiegemeinschaften – informationen und rechtlicher rahmen. `https://www.e-control.at/energiegemeinschaften`, 2025. Accessed: May 2025.

[14] B. Prusty and D. Jena. A critical review on probabilistic load flow studies in uncertainty constrained power systems with photovoltaic generation and a new approach. *Renewable and Sustainable Energy Reviews*, 69:1286–1302, 2017.

[15] Alice Jansson and Emma Åkerman. Monte carlo simulations in load flow calculations: An application on a swedish 50 kv network. Division of Industrial Electrical Engineering and Automation, Lund University, 2021. CODEN: LTH/(IEA-5462)/1-55/(2021).

[16] DIgSILENT GmbH. Optimal power flow (opf) — digsilent powerfactory. `https://www.digsilent.de/en/optimal-power-flow.html`, 2025. Accessed: May 2025.

[17] Zahia Djeblahi, Belkacem Mahdad, and Kamel Srairi. Solving energy management optimization using various global metaheuristics methods. In *2022 19th International Multi-Conference on Systems, Signals Devices (SSD)*, Biskra, Algeria, 2022. IEEE.

[18] Harinder Pal Singh, Yadwinder Singh Brar, and D. P. Kothari. Optimal power flow solution using particle swarm optimization. *International Journal of Computational Engineering Research (IJCER)*, 8(11):1–7, 2018. ISSN (e): 2250–3005.

[19] Shilpa S. Shrawane and M. Diagavane. Application of genetic algorithm for power flow analysis. *International Journal of Engineering Research & Technology (IJERT)*, 2(9):1–6, 2013. ISSN: 2278-0181.

[20] SciPy Developers. scipy.optimize.differential_evolution — scipy documentation. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html`, 2025. Accessed: May 2025.

[21] Laben Imen, Boucherma Mouhamed, and Labed Djamel. Economic dispatch using classical methods and neural networks. *International Journal of Scientific and Research Publications*, 5(1):1–6, 2015.

[22] Muhammad Jamshed Abbass, Robert Lis, and Zohaib Mushtaq. Artificial neural network (ann)-based voltage stability prediction of test microgrid grid. *IEEE Access*, 2023.

[23] Francisco M. Gonzalez-Longatt and José Luis Rueda, editors. *PowerFactory Applications for Power System Analysis*. Springer, Berlin, Heidelberg, 2014.

[24] Florian Strebl. Impact of large-scale deployment of energy communities on distribution grids, 2024. Available online via TU Wien reposiTUm.

[25] Optimal power system planning under growing uncertainty. Technical brochure TB 820, CIGRE, 2020.

[26] Hanif Ramadhania, M. Shepero, J. Munkhammar, N. Widén, and M. Etherden. Review of probabilistic load flow approaches for power distribution systems with photovoltaic generation and electric vehicle charging. *International Journal of Electrical Power and Energy Systems*, 120, 2020.

[27] European Commission. Identification of appropriate generation and system adequacy standards for the internal electricity market, 2015. Technical Report.

[28] X.F. Wang, Y. Song, and M. Irving. *Modern Power Systems Analysis*. Springer Publishing, New York, 2008.

[29] D. Villanueva, J. Pazos, and A. Feijoo. Probabilistic load flow including wind power generation. *IEEE Transactions on Power Systems*, 26(3):1659–1667, 2011.

[30] The SciPy community. scipy.stats.truncnorm — truncated normal distribution. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.truncnorm.html, 2024. Accessed: June 2025.

[31] Muhammad Awais, Abdul Basit, Rana Adnan, Zahoor Ali Khan, Umar Qasim, Tamour Shafique, and Nadeem Javaid. Overload management in transmission system using particle swarm optimization. In *Procedia Computer Science*, volume 52, pages 858–865. Elsevier, 2015.

[32] T. Murali Mohan and T. Nireekshana. A genetic algorithm for solving optimal power flow problem. In *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1438–1440. IEEE, 2019.

[33] Barigye Osbert. Short-term load forecasting using ann model. In *Short Term Load Forecasting Using Artificial Neural Networks*, South Korea, June 2023. ResearchGate.

[34] scikit-learn developers. Mlpregressor — scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html, 2024. Accessed: May 2025.

[35] scikit-learn contributors. Github mirror of scikit-learn documentation - mlpregressor. https://github.com/scikit-learn/scikit-learn.github.io/blob/main/0.18/modules/generated/sklearn.neural_network.MLPRegressor.html, 2024. Accessed: May 2025.

[36] ENTSO-E. Entso-e transparency platform. https://transparency.entsoe.eu/, 2025. Accessed: May 2025.

[37] scikit-learn contributors. Github mirror of scikit-learn documentation - mlpregressor. https://scikit-learn.org/stable/modules/linear_model.html, 2024. Accessed: May 2025.

[38] GeeksforGeeks. Linear regression python implementation. https://www.geeksforgeeks.org/linear-regression-python-implementation/, 2023. Accessed: May 2025.

[39] ENEE.io. Maximizing diesel generator efficiency and runtime: Proven strategies. https://www.enee.io/post/maximizing-diesel-generator-efficiency-and-runtime-proven-strategies, 2024. Accessed: June 2025.

[40] Power International. Optimizing fuel consumption in diesel generators: A comprehensive guide. https://pi-dubai.com/optimizing-fuel-consumption-in-diesel-generators-a-comprehensive-guide/, 2024. Accessed: June 2025.

[41] DIgSILENT GmbH. Cigre mv benchmark network – powerfactory example project. https://www.digsilent.de/en/downloads.html, 2024. Accessed: May 2025. Built-in example in DIgSILENT PowerFactory software.

[42] DIgSILENT GmbH. *PowerFactory Tutorial 2024*. Gomaringen, Germany, 2024.

[43] Mini Thomas, Rakeshranjan, and Roma Raina. Load flow solution for unbalanced radial power distribution using monte-carlo simulation. *WSEAS Transactions on Power Systems*, 2010. Available online.

[44] Daniel Borup, Bent Jesper Christensen, Nicolaj Søndergaard Mühlbach, and Mikkel Slot Nielsen. Targeting predictors in random forest regression. *Journal of Econometrics*, 2023. Available online 6 April 2022, Version of Record 4 March 2023.

[45] Ali Haidar, Matthew Field, Jonathan Sykes, Martin Carolan, and Lois Holloway. Pspso: A package for parameters selection using particle swarm optimization. *SoftwareX*, 15:100722, 2021.

[46] G. Induraj. Implementing particle swarm optimization in python. https://induraj2020.medium.com/implementing-particle-swarm-optimization-in-python-c59278bc5846, 2020. Accessed: 2025-05-13.

[47] Mahdi Sharifzadeh, Alexandra Sikinioti-Lock, and Nilay Shah. Machine-learning methods for integrated renewable power generation: A comparative study of artificial neural networks, support vector regression, and gaussian process regression. *Renewable and Sustainable Energy Reviews*, 108:513–538, 2019.

[48] Antón Román-Portabales, Martín López-Nores, and José Juan Pazos-Arias. Systematic review of electricity demand forecast using ann-based machine learning algorithms. *Sensors*, 21(13):4544, 2021.

[49] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2011.

[50] Cenelec - european committee for electrotechnical standardization. https://www.cenelec.eu/standards-and-specifications. Accessed: August 2025.

[51] Austrian standards international. https://www.austrian-standards.at/en. Accessed: August 2025.

[52] Austrian Power Grid (APG). Day-ahead prices – cross-border exchange. https://markt.apg.at/en/transparency/cross-border-exchange/day-ahead-prices/, 2025. Accessed: May 6, 2025.

[53] Surekha R. Deshmukh. Feed-forward neural network based economic load dispatch. In *Proceedings of the International Conference at Indian Institute of Technology, Madras*, pages 635–639, Madras, India, Dec. 27–30 2010.

A. System Schematics

A.1. Initial interconnected power grid

A.2. Optimized power grid after PSO, GA, and ANN

A.3. Power grid in critical state before PSO optimization

A.4. Power grid in critical state after PSO optimization

A.5. Power grid for 85% scenario before PSO optimization

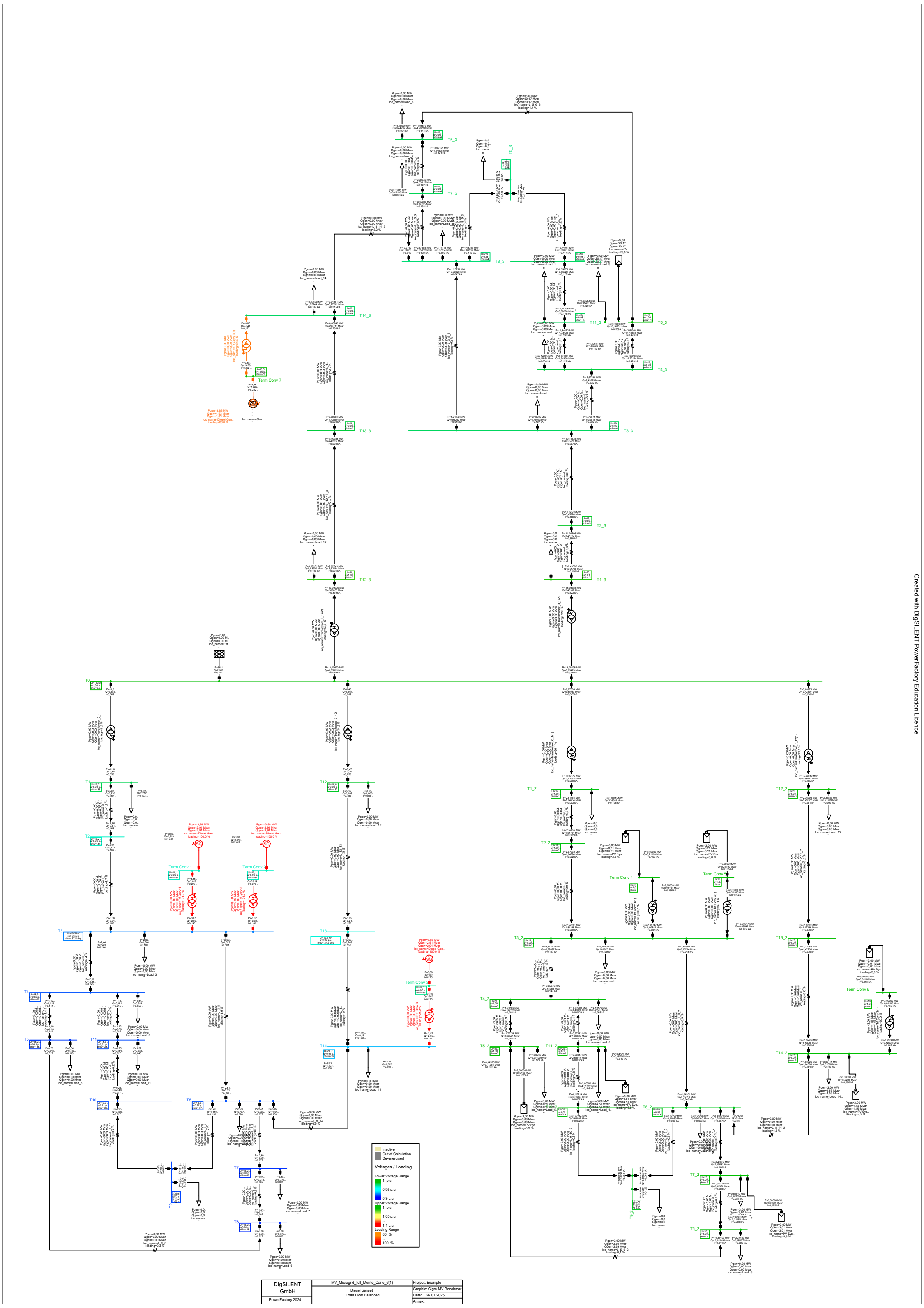


Figure A.1.: Initial interconnected power grid

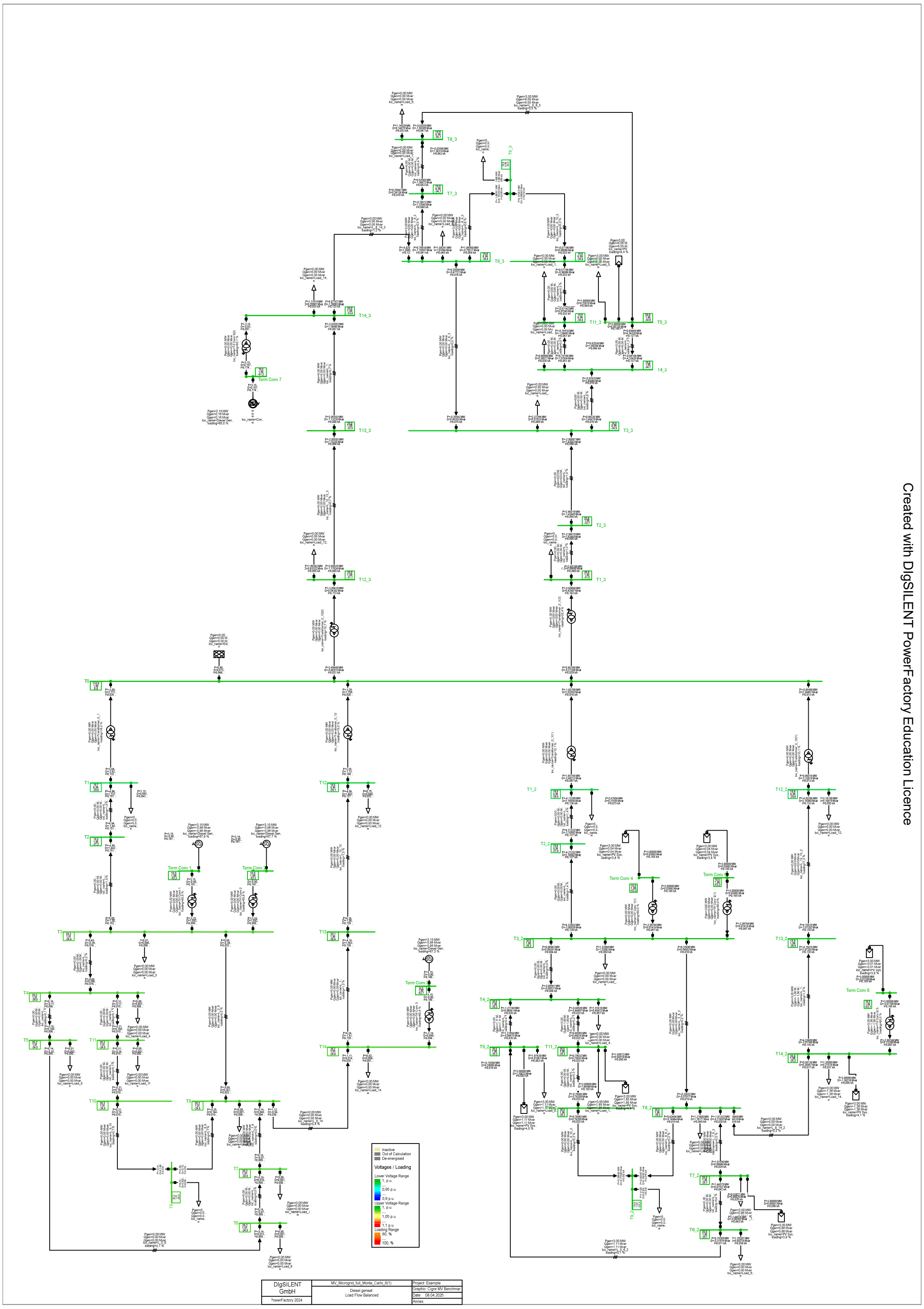


Figure A.2.: Optimized power grid after PSO, GA, and ANN

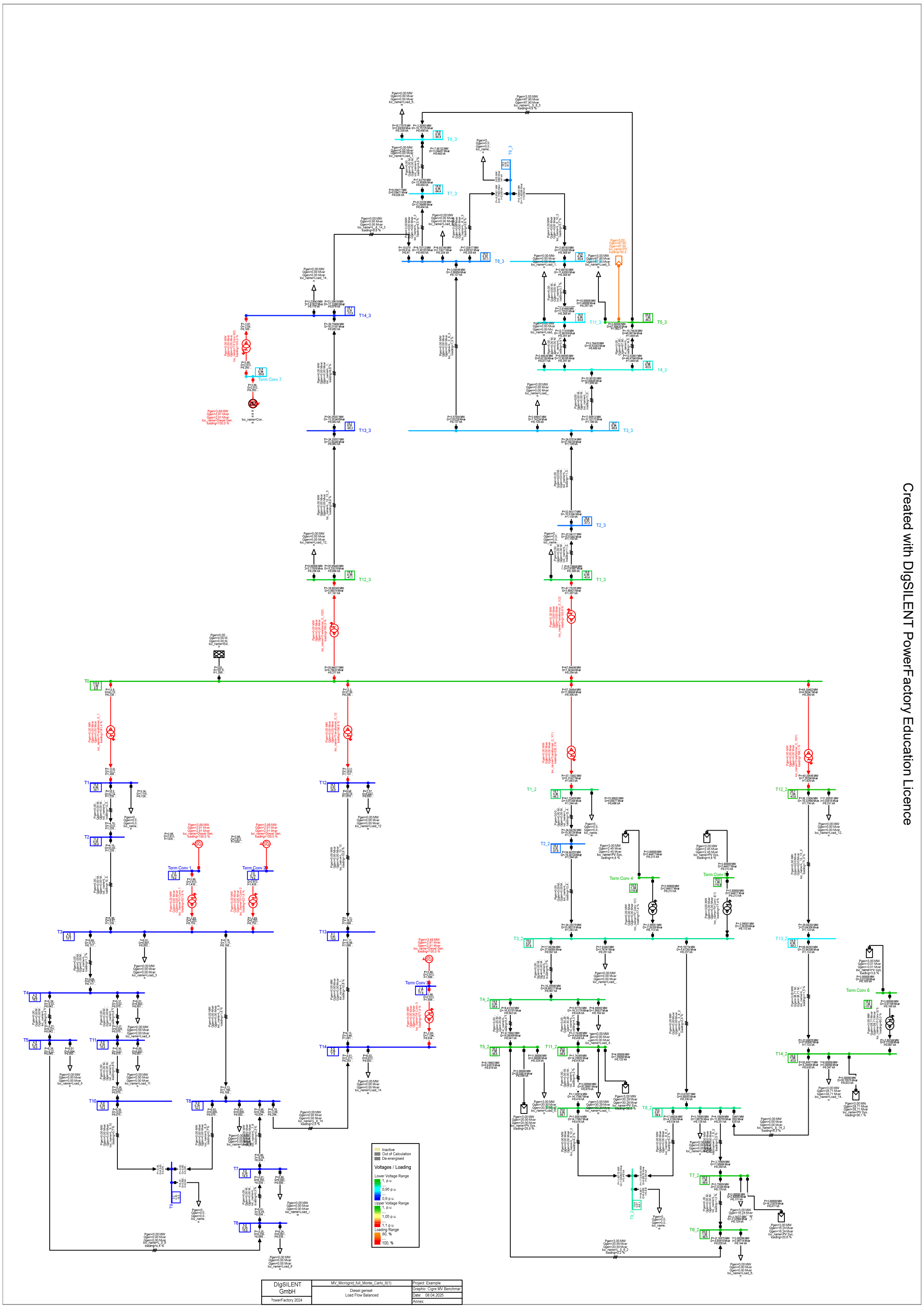


Figure A.3.: Power grid in critical state before PSO optimization

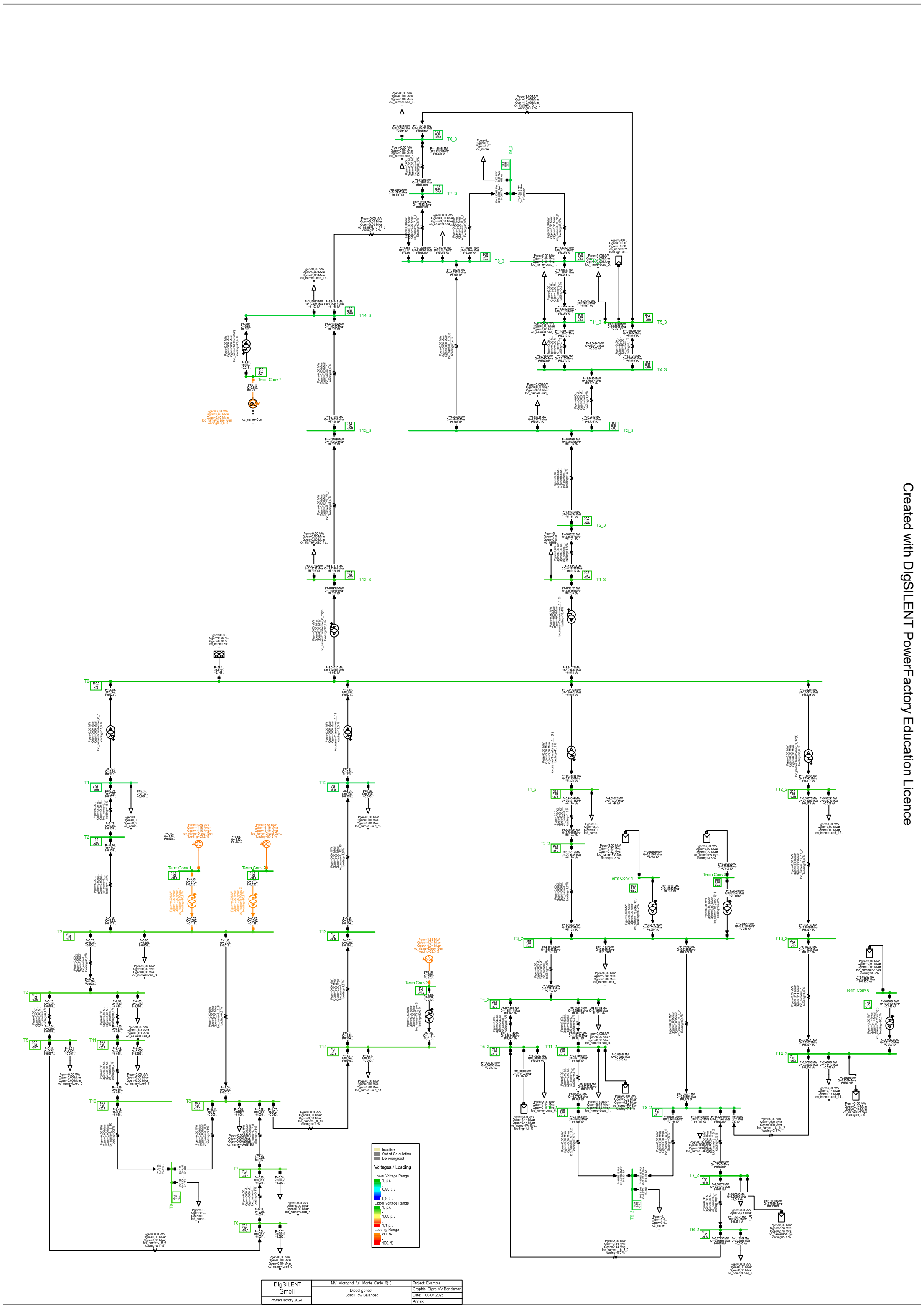


Figure A.4.: Power grid in critical state after PSO optimization

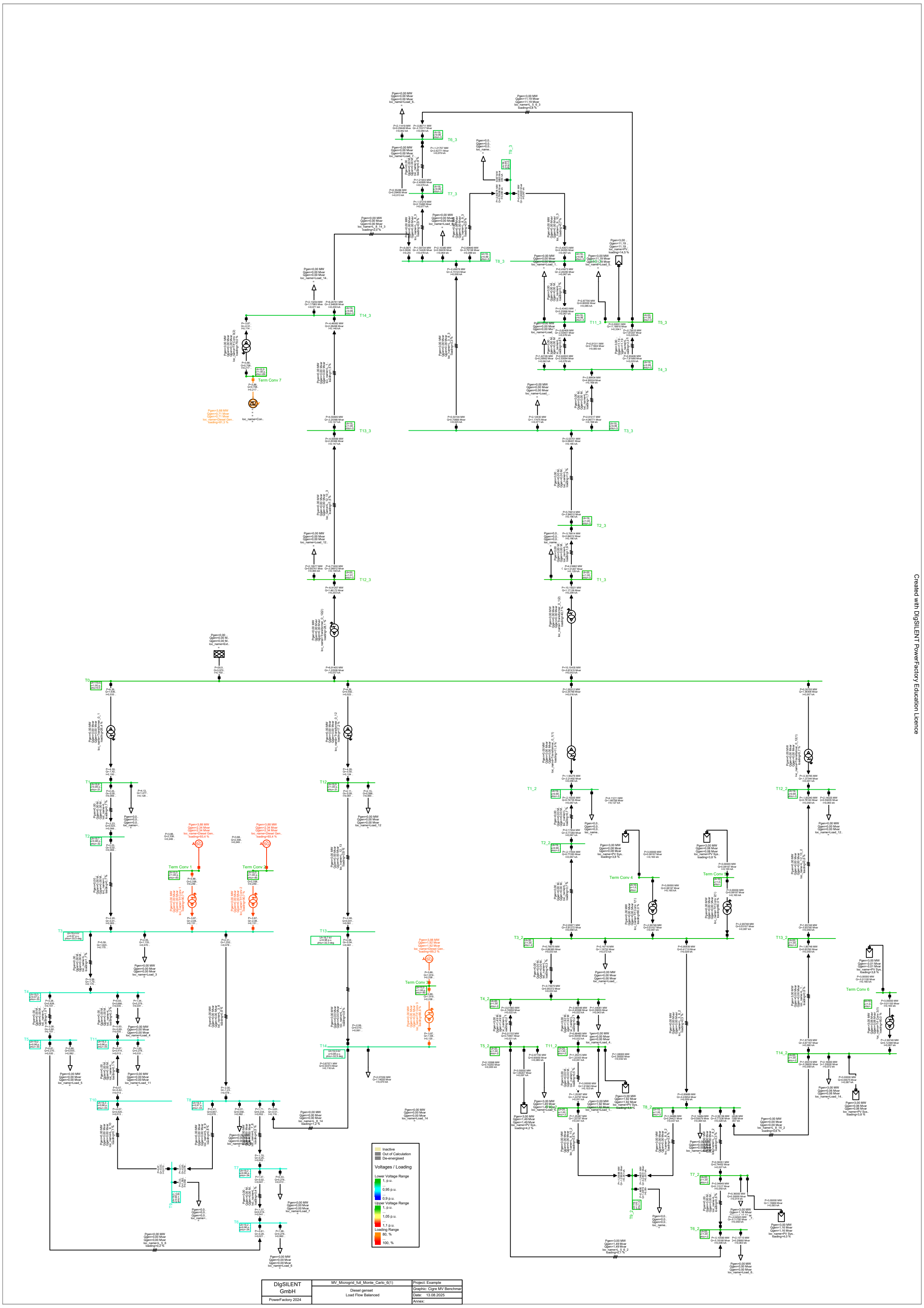


Figure A.5.: Power grid for 85% scenario before PSO optimization

B. Python Code

B.1. Random Forest Regression for PV prediction

Listing B.1: Python code for Random Forest Regression

```
# Loading of ENTSO-E data
PV_dataset = pd.read_csv(’C:/Users/Nikita/Desktop/PV_Dataset.csv
   ’, sep=’;’)

# X     input data, y     objective function (output)
X = PV_dataset[[’Time’, ’Month’, ’Day’, ’Holidays’]]
y = PV_dataset[’Power’]

# Convert "y" to a one-dimensional array
y = y.values.ravel()

# Split into training and test samples
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)

# Training the Random Forest model
model_rf = RandomForestRegressor(n_estimators=100, random_state
   =42)
# model_rf = LinearRegression()
model_rf.fit(X_train, y_train)

# Testing the Random Forest model
y_train_pred = model_rf.predict(X_train)
y_test_pred = model_rf.predict(X_test)

print(f"R^2 on training data: {r2_score(y_train, y_train_pred)
   :.4f}")
print(f"R^2 on test data: {r2_score(y_test, y_test_pred):.4f}")
print(f"MSE on test data: {mean_squared_error(y_test,
   y_test_pred):.4f}")
```