Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ХОМЕНКО ОЛЕКСАНДР МИКОЛАЙОВИЧ

УДК 004.82:004.89

ДИСЕРТАЦІЯ

МЕТОДИ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СЕМАНТИЧНОГО МОДЕЛЮВАННЯ КАСКАДНИХ ЕФЕКТІВ У КРИТИЧНИХ ІНФРАСТРУКТУРАХ ІЗ ЗАСТОСУВАННЯМ ГРАФОВИХ НЕЙРОННИХ МЕРЕЖ

121 – Інженерія програмного забезпечення
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____О. М. Хоменко

Науковий керівник Коваль Олександр Васильович, доктор технічних наук, професор

Київ – 2026

АНОТАЦІЯ

Хоменко О.М. Методи та програмне забезпечення семантичного моделювання каскадних ефектів у критичних інфраструктурах із застосуванням графових нейронних мереж. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення». – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2026.

Критична інфраструктура – складана система, яка складається з множини об’єктів та зв’язків між ними. Електромережі забезпечують електроенергією побутових споживачів, промислові об’єкти та мають значний вплив на функціонування залежних об’єктів. Збої в роботі електромереж можуть призвести до каскадних ефектів, повного чи часткового знеструмлення мережі, що негативно впливає на функціонування пов’язаних інфраструктур:

– водопостачання: зменшення потужності роботи електромережі впливає на роботу насосів, що призводить до ускладнення постачання води до кінцевих споживачів;

– транспортної: проблеми з логістикою, ускладнення руху на дорогах, збільшення часу маршрутів.

Захист критичної інфраструктури від каскадних збоїв є важливим та комплексним завданням. У дослідженнях використовуються методи та програмне забезпечення, що надає можливість виявити потенційні умови виникнення каскадного ефекту, потенційні сценарії розвитку подій. Моделювання та симуляція розвитку каскадних збоїв в електромережі є складним процесом та потребує розуміння предметної області. Обсяг даних в електромережах збільшується в залежності від кількості об’єктів та зв’язків між ними, тому виникає потреба у формалізації даних для використання в аналітичних моделях.

Методи моделювання сценаріїв аналітичної діяльності досліджували О.Г. Додонов, О.В. Коваль, В.Р. Сенченко, А.В. Бойченко, А.Я. Гладун та інші. Методи виявлення збоїв в роботі критичної інфраструктури досліджували: A. Varbella, B. Gjorgiev, G. Sansavini, P. Hines, E.C. Sanchez, С.В. Толюпа та інші.

Теорія графів, мережі Баєса, мережі Петрі, ланцюги Маркова використовуються для аналізу роботи критичної інфраструктури в різних сценаріях. Структура критичної інфраструктури визначається за допомогою графу, що допомагає візуально представити мережу, взаємозв'язки між об'єктами та параметри компонентів. Під час аналізу стійкості та вразливості в електромережах використовуються топологічні та гібридні підходи для дослідженні властивостей мережі. Велика частина метрик графів надає статичну оцінку стійкості системи, не відображає динамічні зміни компонентів системи при виникненні збоїв, а результати аналізу залежать від якості початкових даних сценаріїв збоїв. Використання теорії графів та метрик є основою для розуміння стійкості критичної інфраструктури на ранніх етапах аналізу, але її ефективність зростає при поєднанні з моделями, що враховують динаміку та специфіку предметної області. Мережі Баєса використовуються для дослідження стійкості критичної інфраструктури, аналізу каскадних збоїв, моделювання динамічних процесів, але потребують значних обчислювальних ресурсів для великих та складних систем зі значною кількістю змінних і параметрів для обчислень та висновків. Мережі Петрі використовуються для моделювання та аналізу поведінки системи в сценаріях, але побудова та підтримка великих, складних моделей потребує спеціалізованих знань для перевірки коректності роботи моделі та її інтерпретації. Ланцюги Маркова використовуються для аналізу каскадних збоїв, але при моделюванні великої кількості комплексних взаємозалежностей відбувається спрощення сценарію для зменшення складності обчислення, що знижує точність аналізу. Ефективність роботи моделей залежить від якості та адекватності спрощення даних. Перспективним підходам аналізу каскадних ефектів в електромережах є використання графових нейронних мереж, що надають можливість враховувати зв'язки між компонентами та їхні ознаки.

Значна частина наявних програмних засобів, які використовуються для аналізу каскадних ефектів у системах (електромережах), має певну складність при моделюванні та порівнянні сценаріїв роботи систем. У зв'язку з цим виникає потреба в розробці програмного забезпечення обробки даних, створення представлень про стани роботи електромереж для визначення подібних сценаріїв та прийняття рішень.

На основі проведеного аналізу методів моделювання сценаріїв аналітичної діяльності, методів виявлення каскадних ефектів в електромережах визначено такі протиріччя:

– аналітичні моделі потребують консистентних даних, але відсутні формалізовані моделі перевірки даних;

– нейронні мережі потребують для навчання великих обсягів даних, але існує обмежена кількість доступних даних про роботу електромереж, каскадних ефектів та інструментів для створення сценаріїв;

– методи машинного навчання потенційно можуть виявляти каскадні ефекти в електромережі, але відсутній підхід порівняння сценаріїв;

– програмні засоби аналізу каскадних ефектів побудовані для обробки підготовлених наборів даних, але мають обмежений функціонал для обробки даних у режимі реального часу.

На основі визначених протиріч було сформовано наукове завдання: розробка науково-методичного апарату та програмного забезпечення семантичного моделювання розвитку каскадних ефектів у критичних інфраструктурах та їхнього порівняння для визначення альтернативних сценаріїв на основі представлення з графових нейронних мереж.

Метою дисертаційної роботи є підвищення ефективності порівняння сценаріїв каскадних ефектів в електромережах та визначення подібних станів систем на основі графових нейронних мереж із врахуванням семантики ознак та зв'язків між об'єктами.

Об’єктом дослідження є процеси аналізу вимог, розроблення, впровадження і супроводження програмного забезпечення виникнення каскадних ефектів та їхнє поширення в критичних інфраструктурах.

Предметом дослідження є методи та програмне забезпечення семантичного моделювання взаємозв’язків у критичних інфраструктурах та побудови графових нейронних мереж для створення та порівняння представлень про сценарії каскадних ефектів.

Дисертаційна робота виконана відповідно до науково-технічної діяльності Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» і кафедри інженерії програмного забезпечення в енергетиці.

Наукова новизна одержаних результатів досліджень:

1. Уперше розроблено метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі моделі потоку потужності, що базується на використанні онтологічної моделі та множині правил для формалізації, перевірки, доповнення даних про структуру електромережі та процеси розвитку каскадних ефектів у сценаріях. Розроблений метод дозволяє під час попереднього аналізу даних сценаріїв про каскадні ефекти в роботі електромереж виявляти протиріччя й помилки, що можуть бути усунені на ранньому етапі для покращення якості даних під час використання методів машинного навчання.

2. Уперше розроблено метод визначення подібності сценаріїв каскадних ефектів в електромережах на основі моделі автоенкодера та косинуса подібності, що відрізняється від подібних семантикою взаємодії між об’єктами та їхніми значеннями при порівнянні представлень про стани систем, що дозволяє враховувати структуру та вплив змін в роботі компонентів електромереж.

3. Удосконалено моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж у сценаріях на основі графових нейронних мереж, які, на відміну від наявних, враховують семантику станів компонентів. Удосконалена модель для створення представлень про стани систем

у кроках сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 5,6%, коефіцієнт детермінації ребер у середньому на 27,2%. Удосконалена модель для створення представлень про зміни станів систем у послідовності кроків сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 9,5%, коефіцієнт детермінації ребер у середньому на 0,5%.

4. Уперше розроблено архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, яка відрізняється від наявних порівнянням та визначенням подібності станів електромереж у сценаріях на основі розроблених методів та моделей автоенкодера, що дозволяє розробити програмне забезпечення, яке надає інформацію для прийняття рішення.

Практичне значення одержаних результатів:

1. Уперше розроблено програмне забезпечення аналізу каскадних ефектів в електромережах, де використовуються такі методи: метод семантичного моделювання сценаріїв каскадних збоїв в електромережах; метод визначення подібності сценаріїв каскадних ефектів в електромережі на основі графової нейронної мережі та косинуса подібності. Енкодер натренованої моделі нейронної мережі використовується для створення представлень про роботу електромереж у сценаріях, що порівнюються за допомогою коефіцієнта подібності на основі косинуса подібності із врахуванням семантики та значень ознак. Крім того, отримані представлення використовуються для визначення множини подібних станів електромереж на основі кластеризації. Одержані результати можуть використовуватися в освітньому процесі: матеріали для лекцій, лабораторних робіт. Програмне забезпечення може застосовуватися як система для прийняття рішень на основі наявних сценаріїв.

2. Створено базу даних, що містить інформацію про проведені експерименти, що використовуються при аналізі роботи моделей. Систематизація даних допомагає визначити приховані закономірності, помилки, аномалії, які важко побачити при поодиноких тестах. На основі накопичених даних про роботу моделей можна виявити ефективні та неефективні

налаштування гіперпараметрів. Порівняння роботи моделей з різними конфігураціями можна використати для визначення сильних і слабких сторін моделей та вибрати необхідні налаштування залежно від даних.

Отримані результати, які зазначені в дисертаційному дослідженні, є внеском у розвиток теоретичних і прикладних напрямків інформаційних технологій: використання семантичної моделі для формалізації предметної області (сценарії роботи електромереж); використання графових нейронних мереж для аналізу сценаріїв каскадних ефектів (каскадних збоїв) у критичних інфраструктурах (електромережах); використання програмного забезпечення для обробки даних, що відкриває нові можливості для розв’язання складних задач.

Перспективними напрямами досліджень є розробка нових методів й удосконалення наявних зі створення комплексних сценаріїв роботи електромереж, які розширюють множину подій, що впливають на мережу, варіанти розвитку каскадних збоїв, множину дій для стабілізації роботи мережі. Нові підходи можуть підвищити якість та варіативність даних для навчання нейронних мереж під час аналізу каскадних ефектів. Розробка нових моделей машинного навчання, які можуть додатково враховувати сторонні чинники в роботі електромереж та змінюються від моделювання середовища, є перспективним напрямом для майбутніх досліджень.

Ключові слова: інженерія програмного забезпечення, критична інфраструктура, електромережа, каскадний ефект, каскадні відмови, моделювання подій, семантична модель, формалізація знань, онтологічна модель, машинне навчання, нейронна мережа, графова нейронна мережа, програмне забезпечення, інформаційні технології.

ANNOTATION

Khomenko O.M. Methods and software for semantic modeling of cascading effects in critical infrastructures using graph neural networks. – Qualification scientific work in the form of a manuscript.

Thesis for the degree of Doctor of Philosophy in specialty 121 "Software Engineering". – National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2026.

Critical infrastructure is a complex system consisting of a set of objects and connections between them. Power grids provide electricity to household consumers, industrial facilities and have a significant impact on the functioning of dependent facilities. Failures in the operation of power grids can lead to cascading effects, complete or partial power outages of the network, which negatively affects the functioning of related infrastructures:

– water supply: a decrease in the power grid's operating capacity affects the operation of pumps, which complicates the supply of water to end users;

– transport: problems with logistics, traffic congestion, increased route times.

Protecting critical infrastructure from cascading failures is an important and complex task. The research uses methods and software that make it possible to identify potential conditions for the occurrence of a cascading effect, potential scenarios for the development of events. Modeling and simulation of the development of cascading failures in the power grid is a complex process and requires an understanding of the subject area. The volume of data in power grids increases depending on the number of objects and connections between them, therefore there is a need to formalize data for use in analytical models.

Methods for modeling analytical activity scenarios were studied by O.G. Dodonov, O.V. Koval, V.R. Senchenko, A.V. Boychenko, A.Y. Gladun and others. Methods for detecting failures in the operation of critical infrastructure were studied by: A. Varbella, B. Gjorgiev, G. Sansavini, P. Hines, E.C. Sanchez, S.V. Tolyupa and others.

Graph theory, Bayesian networks, Petri nets, Markov chains are used to analyze the operation of critical infrastructure in various scenarios. The structure of critical infrastructure is defined using a graph, which helps to visually represent the network, the relationships between objects, and the parameters of the components. When analyzing the stability and vulnerability of power grids, topological and hybrid approaches are used to study the properties of the network. Most graph metrics provide a static assessment of system stability, do not reflect dynamic changes in system components during failures, and the analysis results depend on the quality of the initial data of failure scenarios. The use of graph theory and metrics is the basis for understanding the stability of critical infrastructure in the early stages of analysis, but its effectiveness increases when combined with models that take into account the dynamics and specificity of the subject area. Bayesian networks are used to study the stability of critical infrastructure, analyze cascading failures, and simulate dynamic processes, but require significant computational resources for large and complex systems with a significant number of variables and parameters for calculations and conclusions. Petri nets are used to model and analyze system behavior in scenarios, but building and maintaining large, complex models requires specialized knowledge to verify the correctness of the model and its interpretation. Markov chains are used to analyze cascading failures, but when modeling a large number of complex interdependencies, the scenario is simplified to reduce the complexity of the calculation, which reduces the accuracy of the analysis. The effectiveness of the models depends on the quality and adequacy of the data simplification. Promising approaches to the analysis of cascading effects in power grids include the use of graph neural networks, which make it possible to take into account the connections between components and their characteristics.

A significant part of the existing software tools used to analyze cascading effects in systems (power grids) has a certain complexity in modeling and comparing system operation scenarios. In this regard, there is a need to develop data processing software, create representations of the states of power grids to identify such scenarios and make decisions.

Based on the analysis of methods for modeling analytical activity scenarios, methods for detecting cascading effects in power grids, the following contradictions were identified:

– analytical models require consistent data, but there are no formalized data verification models;

– neural networks require large amounts of data for training, but there is a limited amount of available data on the operation of power grids, cascading effects, and tools for creating scenarios;

– machine learning methods can potentially detect cascading effects in the power grid, but there is no scenario comparison approach;

– software tools for analyzing cascading effects are built to process prepared data sets, but have limited functionality for real-time data processing.

Based on the identified contradictions, a scientific task was formed: development of a scientific and methodological apparatus and software for semantic modeling of the development of cascading effects in critical infrastructures and their comparison to determine alternative scenarios based on a representation from graph neural networks.

The purpose of the thesis is to increase the efficiency of comparing cascade effect scenarios in power grids and to determine similar system states based on graph neural networks, taking into account the semantics of features and connections between objects.

The object of the study is the processes of requirements analysis, development, implementation and maintenance of software for the occurrence of cascading effects and their spread in critical infrastructures.

The subject of the study is methods and software for semantic modeling of relationships in critical infrastructures and construction of graph neural networks for creating and comparing representations of cascade effect scenarios.

The thesis was carried out in accordance with the scientific and technical activities of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" and the Department of Software Engineering in Energy.

Scientific novelty of the obtained research results:

1. For the first time, a method for semantic modeling of cascading failure scenarios in power grids has been developed based on a power flow model, which is based on the use of an ontological model and a set of rules for formalizing, verifying, and supplementing data on the structure of the power grid and the processes of developing cascading effects in scenarios. The developed method allows, during the preliminary analysis of data on cascading effects scenarios in the operation of power grids, to identify contradictions and errors that can be eliminated at an early stage to improve data quality when using machine learning methods.

2. For the first time, a method for determining the similarity of cascading effect scenarios in power grids has been developed based on the autoencoder model and the cosine of similarity, which differs from similar ones by the semantics of the interaction between objects and their values when comparing representations of system states, which allows taking into account the structure and impact of changes in the operation of power grid components.

3. Autoencoder models have been improved to create representations of the step and sequence of steps of the operation of electrical networks in scenarios based on graph neural networks, which, unlike existing ones, take into account the semantics of component states. The improved model for creating representations of system states in scenario steps allows to increase the coefficient of determination of nodes by an average of 5,6%, the coefficient of determination of edges by an average of 27,2%. The improved model for creating representations of system state changes in the sequence of scenario steps allows to increase the coefficient of determination of nodes by an average of 9,5%, the coefficient of determination of edges by an average of 0,5%.

4. For the first time, a software architecture has been developed for conducting experiments with power grid operation scenarios, in particular cascade effects, which differs from existing ones by comparing and determining the similarity of power grid states in scenarios based on developed methods and autoencoder models, which allows developing software that provides information for decision-making.

Practical significance of the results obtained:

1. For the first time, software for analyzing cascade effects in power grids has been developed, which uses the following methods: a method for semantic modeling of cascade failure scenarios in power grids; a method for determining the similarity of cascade effect scenarios in power grids based on a graph neural network and cosine similarity. The encoder of the trained neural network model is used to create representations of the operation of power grids in scenarios that are compared using a similarity coefficient based on cosine similarity taking into account semantics and feature values. In addition, the resulting representations are used to determine the set of similar states of power grids based on clustering. The results obtained can be used in the educational process: materials for lectures, laboratory work. The software can be used as a system for making decisions based on existing scenarios.

2. A database has been created containing information on the experiments conducted, which are used in the analysis of the operation of models. Systematization of data helps to identify hidden patterns, errors, anomalies that are difficult to see in single tests. Based on the accumulated data on the operation of models, effective and ineffective hyperparameter settings can be identified. Comparison of the performance of models with different configurations can be used to identify the strengths and weaknesses of the models and select the necessary settings depending on the data.

The results obtained, which are indicated in the dissertation research, are a contribution to the development of theoretical and applied areas of information technology: the use of a semantic model for formalizing the subject area (electricity grid operation scenarios); the use of graph neural networks for analyzing scenarios of cascading effects (cascading failures) in critical infrastructures (electricity grids); the use of data processing software, which opens up new opportunities for solving complex problems. Promising areas of research are the development of new methods and improvement of existing ones for creating complex scenarios of electric grid operation, which expand the set of events affecting the network, options for the development of cascading failures, and the set of actions to stabilize the network operation. New approaches can improve the quality and variability of data for training neural networks during the analysis of cascading effects. The development of new machine learning

models that can additionally take into account extraneous factors in the operation of electric grids and change from environmental modeling is a promising area for future research.

Keywords: software engineering, critical infrastructure, power grid, cascading effect, cascading failures, event modeling, semantic model, knowledge formalization, ontological model, machine learning, neural network, graph neural network, software, information technology.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Сенченко В. Р., Бойченко А. В., Коваль О. В., Бисько Р. М., Хоменко О. М. Огляд методів і технологій сценарного аналізу каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2024. Т. 26, № 1. С. 24–54. https://doi.org/10.35681/1560-9189.2024.26.1.308506.

Особистий внесок кожного автора:

– Сенченко В. Р.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Бойченко А. В.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Коваль О. В.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Бисько Р. М.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Хоменко О. М.: дослідження розвитку каскадних ефектів у критичній інфраструктурі, застосування методів машинного навчання для аналізу каскадних відмов, редагування тексту статті.

2. Хоменко О. М., Сенченко В. Р., Коваль О. В. Мережевий підхід при дослідженні каскадних ефектів критичних інфраструктур. *Реєстрація, зберігання і обробка даних.* 2024. Т. 26. № 2. С. 44–72. https://doi.org/10.35681/1560-9189.2024.26.2.316908.

Особистий внесок кожного автора:

– Хоменко О. М.: пошук та відбір релевантних публікацій, аналіз методів моделювання каскадних збоїв у роботі критичної інфраструктури, систематизація інформації, підготовка тексту статті;

– Сенченко В. Р.: перевірка та доопрацювання тексту статті;

– Коваль О. В.: перевірка тексту статті.

3. Сенченко В. Р., Бойченко А. В., Коваль О. В., Хоменко О. М. Методологія та архітектура платформи моделювання взаємозалежностей у критичних інфраструктурах при виникненні каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2025. Т. 27. № 1. С. 28–41. https://doi.org/10.35681/1560-9189.2025.27.1.335624.

Особистий внесок кожного автора:

– Сенченко В. Р.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Бойченко А. В.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Коваль О. В.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Хоменко О. М.: дослідження переваг та обмежень використання різних видів архітектури програмного забезпечення, редагування тексту статті.

4. Хоменко О. М., Коваль О. В. Побудова сценаріїв розвитку каскадних збоїв в інфраструктурі електромереж. *Зв'язок.* 2025. №5. С. 3–12. https://doi.org/10.31673/2412-9070.2025.050938.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, формування набору даних, проведення експериментів, формування результатів, підготовка та редагування статті;

– Коваль О. В.: перевірка та редагування тексту статті.

5. Хоменко О., Коваль О. Аналіз та порівняння сценаріїв каскадних ефектів в критичній інфраструктурі. *Інформаційні технології та суспільство*. 2025. № 3 (18). С. 176–185. https://doi.org/10.32689/maup.it.2025.3.24.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, розробка моделі, формування набору даних, проведення експериментів, формування результатів, підготовка статті;

– Коваль О. В.: перевірка та редагування тексту статті.

6. Хоменко О., Коваль О. Аналіз сценаріїв каскадних ефектів в критичній інфраструктурі на основі графових нейронних мереж. *Інформаційні технології*

та суспільство. 2025. №4 (19). С. 182–190. https://doi.org/10.32689/maup.it.2025.4.28.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, розробка моделі, формування набору даних, проведення експериментів, формування результатів, підготовка статті;

– Коваль О. В.: перевірка та редагування тексту статті.

7. Gavrilenko O., Khomenko O., Zhurakovska O., Kohan A., Piskun A., Khalus O. Application of association rules for formation of public (administrative) services portfolio. *Advanced Information Systems*. 2022. Vol. 6, No. 4. P. 63–68. https://doi.org/10.20998/2522-9052.2022.4.09.

Особистий внесок кожного автора:

– Gavrilenko O.: розробка концепції, визначення завдань в статті, редагування статті;

– Khomenko O.: розробка концепції, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті;

– Zhurakovska O.: розробка концепції, формалізація проблематики, редагування статті;

– Kohan A.: перевірка, виправлення та доопрацювання тексту статті;

– Piskun A.: перевірка, виправлення та доопрацювання тексту статті;

– Khalus O.: перевірка, виправлення та доопрацювання тексту статті.

8. Гавриленко О., Жураковська О., Коган А., Богданова Н., Хоменко О. Аналіз впливу коефіцієнтів подібності на склад портфелів публічних (адміністративних) послуг. Адаптивні системи автоматичного управління. 2023. Том 1, № 42. С. 49–58. https://doi.org/10.20535/1560-8956.42.2023.279089.

Особистий внесок кожного автора:

– Гавриленко О.: розробка концепції, алгоритму, редагування статті;

– Жураковська О.: розробка концепції, алгоритму, редагування статті;

– Коган А.: перевірка, виправлення та доопрацювання тексту статті;

– Богданова Н.: перевірка, виправлення та доопрацювання тексту статті;

– Хоменко О.: розробка концепції, алгоритму, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті.

9. Gavrilenko O., Khomenko O., Zhurakovska O., Kogan A., Matviichuk R., Piskun A., Khavikova Y. Establishing the grouping principle of public services based on the analysis of similarity coefficients. *Eastern-European Journal of Enterprise Technologies*. 2023 Vol. 3, No. 3 (123). P. 22–29. https://doi.org/10.15587/1729-4061.2023.280218.

Особистий внесок кожного автора:

– Gavrilenko O.: розробка алгоритму, визначення завдань в статті, редагування статті;

– Khomenko O.: розробка алгоритму, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті;

– Zhurakovska O.: розробка алгоритму, формалізація проблематики, редагування статті;

– Kogan A.: перевірка, виправлення та доопрацювання тексту статті;

– Matviichuk R.: перевірка, виправлення та доопрацювання тексту статті;

– Piskun A.: перевірка, виправлення та доопрацювання тексту статті;

– Khavikova Y.: перевірка, виправлення та доопрацювання тексту статті.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

10. Коваль О. В., Хоменко О. М. Вплив каскадних ефектів на роботу критичної інфраструктури. *Матеріали І Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 14 грудня 2023. С. 36–38.

Особистий внесок кожного автора: Коваль О. В. – перевірка та редагування тексту; Хоменко О. М. – розробка концепції публікації, редагування тексту.

11. Хоменко О. М., Сенченко В. Р., Коваль О. В. Програмні засоби аналізу каскадних ефектів в критичній інфраструктурі. *Матеріали XXI Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»*, Київ, 23–26 квітня 2024. С. 190–192.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення аналізу роботи критичної інфраструктури, редагування тексту; Сенченко В. Р. – перевірка тексту; Коваль О. В. – перевірка тексту.

12. Хоменко О. М., Коваль О. В. Підходи до аналізу каскадних ефектів в роботі електромереж. *Матеріали II Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 13 листопада 2024. С. 71–73.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення аналізу, систематизація інформації, редагування тексту; Коваль О. В. – перевірка тексту.

13. Хоменко О. М., Коваль О. В. Створення онтологічної моделі для опису моделювання електромережі. *Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»*, Київ, 24 квітня 2025. С. 469–472.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, розробка онтологічної моделі для опису моделювання електромережі, оформлення результатів; Коваль О. В. – перевірка тексту.

14. Хоменко О. М., Коваль О. В. Генерація даних для аналізу роботи пов'язаних критичних інфраструктур. Abstracts of XXIV International Scientific and Practical Conference «Latest theories and technologies for the development of scientific research», Zaragoza, 16–18 June 2025. С. 256–259.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, опис підходів для генерації даних про роботу критичної інфраструктури, оформлення результатів; Коваль О. В. – перевірка тексту.

15. Хоменко О. М., Коваль О. В. Використання онтологічної моделі та машинного навчання для аналізу каскадних ефектів в критичній інфраструктурі.

Матеріали III Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення», Київ, 13 листопада 2025. С. 61–63.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення експериментів, оформлення результатів; Коваль О. В. – перевірка тексту.

16. Гавриленко О. В., Хоменко О. М., Аналіз роботи алгоритмів для формування портфелів публічних (адміністративних) послуг на основі асоціативних правил. *Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, Київ, 25 листопада 2022. С. 89–90.

Особистий внесок кожного автора: Гавриленко О. В. – розробка концепції публікації, проведення експериментів, редагування тексту; Хоменко О. М. – розробка концепції публікації, проведення експериментів, редагування тексту.

17. Хоменко О. М., Гавриленко О. В. Концепція інформаційної системи для формування портфелів публічних (адміністративних) послуг. *Матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)»*, Київ, 9–11 травня 2023. С. 34–37.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, редагування тексту; Гавриленко О. В. – розробка концепції публікації, редагування тексту.

18. Gavrilenko O., Zhurakovska O., Khomenko O. Analysis of the feasibility of implementing service portfolios. *Abstracts of XXVIII International Scientific and Practical Conference «Unusual methods of development of science and thoughts»*, Madrid, 17–19 July 2023. С. 151–154.

Особистий внесок кожного автора: Gavrilenko O. – розробка концепції публікації, перевірка тексту; Zhurakovska O. – розробка концепції публікації, перевірка тексту; Khomenko O. – розробка концепції публікації, проведення експериментів, формування результатів, редагування тексту.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UML – Unified Modeling Language

BPMN – Business Process Model and Notation

OWL – Web Ontology Language

GNN – Graph Neural Network

GCN – Graph Convolutional Network

GAT – Graph Attention Network

GAT + Edges – Graph Attention Network with edges

GATV2 – Graph Attention Network v2

GATV2 + Edges – Graph Attention Network v2 with edges

LSTM – Long Short-Term Memory

MLP – Multilayer perceptron

JWT – JSON Web Token

ВСТУП

Актуальність теми. Відключення електроенергії впливає на роботу пов'язаних критичних інфраструктур. Збій в роботі компонентів електромережі може призвести до виникнення каскадного ефекту, що потенційно може вивести з ладу частину електромережі чи всі її компоненти. З часом складність мереж зростає, що породжує ризик виникнення тривалих знеструмлень. Каскадний ефект розвивається не лінійно та стрімко, що ускладнює процес прогнозування його розвитку. При дослідженні властивостей електромереж, розвитку збоїв та каскадних ефектів використовуються різні підходи: теорія графів, мережі Баєса, мережі Петрі, ланцюги Маркова, методи машинного навчання. Графові нейронні мережі – перспективний напрямок, що демонструє гарні результати для вирішення різних типів задач у роботі електромереж, зокрема при аналізі каскадних збоїв та ефектів. Програмне забезпечення, яке використовується під час аналізу роботи електромереж, має обмежений функціонал: порівняння роботи електромереж у різних сценаріях, використання наявного досвіду при прийнятті рішення.

На основі проведеного аналізу методів моделювання сценаріїв аналітичної діяльності, методів виявлення каскадних ефектів в електромережах визначено такі протиріччя:

– аналітичні моделі потребують консистентних даних, але відсутні формалізовані моделі перевірки даних;

– нейронні мережі потребують для навчання великих обсягів даних, але існує обмежена кількість доступних даних про роботу електромереж, каскадних ефектів та інструментів для створення сценаріїв;

– методи машинного навчання потенційно можуть виявляти каскадні ефекти в електромережі, але відсутній підхід порівняння сценаріїв;

– програмні засоби аналізу каскадних ефектів побудовані для обробки підготовлених наборів даних, але мають обмежений функціонал для обробки даних у режимі реального часу.

На основі визначених протиріч було сформовано наукове завдання: розробка науково-методичного апарату та програмного забезпечення семантичного моделювання розвитку каскадних ефектів у критичних інфраструктурах та їхнього порівняння для визначення альтернативних сценаріїв на основі представлення з графових нейронних мереж.

Зв’язок роботи з науковими програмами, планами, темами.

Дисертаційна робота виконана відповідно до науково-технічної діяльності Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» і кафедри інженерії програмного забезпечення в енергетиці.

Мета і завдання досліджень.

Метою дисертаційної роботи є підвищення ефективності порівняння сценаріїв каскадних ефектів в електромережах та визначення подібних станів систем на основі графових нейронних мереж із врахуванням семантики ознак та зв’язків між об’єктами.

Для досягнення мети були сформовані часткові **наукові завдання:**

1. Провести аналіз наявних методів та програмного забезпечення моделювання роботи електромереж та каскадних ефектів.

2. Розробити метод семантичного моделювання сценаріїв каскадних збоїв в електромережах для формалізації даних.

3. Розробити метод створення та порівняння представлень сценаріїв роботи електромережі для визначення подібності сценаріїв.

4. Удосконалити моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж в сценаріях.

5. Розробити архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, та моделями нейронними мереж для визначення подібних станів електромереж у сценаріях.

6. Провести обчислювальні експерименти створення та порівняння представлень сценаріїв каскадних ефектів в електромережах.

Об’єкт дослідження: процеси аналізу вимог, розроблення, впровадження і супроводження програмного забезпечення виникнення каскадних ефектів та їхнє поширення в критичних інфраструктурах.

Предмет дослідження: методи та програмне забезпечення семантичного моделювання взаємозв’язків у критичних інфраструктурах та побудови графових нейронних мереж для створення та порівняння представлень про сценарії каскадних ефектів.

Методи дослідження: методи порівняльного аналізу та систематизації інформації використані для визначення актуальності дослідження, методи моделювання використані для формування набору даних сценаріїв роботи електромереж при виникненні збоїв, методи семантичного аналізу використані для формалізації знань про предметну область, методи машинного навчання використані для формування представлень про стани електромереж у сценаріях та їхнього порівняння.

Наукова новизна одержаних результатів дослідження:

1. Уперше розроблено метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі моделі потоку потужності, що базується на використанні онтологічної моделі та множині правил для формалізації, перевірки, доповнення даних про структуру електромережі та процеси розвитку каскадних ефектів у сценаріях. Розроблений метод дозволяє під час попереднього аналізу даних сценаріїв про каскадні ефекти в роботі електромереж виявляти протиріччя й помилки, що можуть бути усунені на ранньому етапі для покращення якості даних під час використання методів машинного навчання.

2. Уперше розроблено метод визначення подібності сценаріїв каскадних ефектів в електромережах на основі моделі автоенкодера та косинуса подібності, що відрізняється від подібних семантикою взаємодії між об’єктами та їхніми значеннями при порівнянні представлень про стани систем, що дозволяє враховувати структуру та вплив змін в роботі компонентів електромереж.

3. Удосконалено моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж у сценаріях на основі графових нейронних мереж, які, на відміну від наявних, враховують семантику станів компонентів. Удосконалена модель для створення представлень про стани систем у кроках сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 5,6%, коефіцієнт детермінації ребер у середньому на 27,2%. Удосконалена модель для створення представлень про зміни станів систем у послідовності кроків сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 9,5%, коефіцієнт детермінації ребер у середньому на 0,5%.

4. Уперше розроблено архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, яка відрізняється від наявних порівнянням та визначенням подібності станів електромереж у сценаріях на основі розроблених методів та моделей автоенкодера, що дозволяє розробити програмне забезпечення, яке надає інформацію для прийняття рішення.

Практичне значення одержаних результатів:

1. Уперше розроблено програмне забезпечення аналізу каскадних ефектів в електромережах, де використовуються такі методи: метод семантичного моделювання сценаріїв каскадних збоїв в електромережах; метод визначення подібності сценаріїв каскадних ефектів в електромережі на основі графової нейронної мережі та косинуса подібності. Енкодер натренованої моделі нейронної мережі використовується для створення представлень про роботу електромереж у сценаріях, що порівнюються за допомогою коефіцієнта подібності на основі косинуса подібності із врахуванням семантики та значень ознак. Крім того, отримані представлення використовуються для визначення множини подібних станів електромереж на основі кластеризації. Одержані результати можуть використовуватися в освітньому процесі: матеріали для лекцій, лабораторних робіт. Програмне забезпечення може застосовуватися як система для прийняття рішень на основі наявних сценаріїв.

2. Створено базу даних, що містить інформацію про проведені експерименти, що використовуються при аналізі роботи моделей. Систематизація даних допомагає визначити приховані закономірності, помилки, аномалії, які важко побачити при поодиноких тестах. На основі накопичених даних про роботу моделей можна виявити ефективні та неефективні налаштування гіперпараметрів. Порівняння роботи моделей з різними конфігураціями можна використати для визначення сильних і слабких сторін моделей та вибрати необхідні налаштування залежно від даних.

Впровадження одержаних результатів:

Результати досліджень впроваджені в Інституті проблем реєстрації інформації НАН України та лекційних, лабораторних заняттях навчальних дисциплін Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»: «Інженерія даних та знань», «Інтелектуальний аналіз даних для задач енергетики».

Особистий внесок здобувача. Дисертаційна робота є результатом самостійного виконання наукового дослідження. Усі результати, представлені в дисертаційній роботі, отримані здобувачем одноосібно. У статтях із співавторами здобувачеві належать такі результати:

– у роботі [182] розроблено частину підходу формування наборів послуг з використанням асоціативних правил, проведено формування тестового набору даних та тестування підходу;

– у роботі [183] розроблено частину алгоритму формування портфеля послуг за допомогою коефіцієнтів подібності, проведено формування тестового набору даних та тестування алгоритму;

– у роботі [184] розроблено частину алгоритму групування послуг на основі аналізу їхнього складу з використанням коефіцієнтів подібності, проведено формування тестового набору даних та тестування роботи алгоритму;

– у роботі [39] проведено огляд методів машинного навчання під час аналізу каскадних ефектів;

– у роботі [41] проведено огляд методів та програмного забезпечення аналізу каскадних ефектів у критичній інфраструктурі та її стійкості: аналіз застосування теорії графів, мережі Баєса, мережі Петрі, ланцюгів Маркова, онтологічної моделі, визначено переваги та обмеження використання методів, програмних засобів;

– у роботі [40] проведено огляд дизайну архітектури програмного забезпечення застосунку аналізу каскадних ефектів;

– у роботі [134] розроблено метод семантичного моделювання сценаріїв каскадних збоїв в електромережі, що використовується для формалізації, перевірки, покращення якості даних симуляції, доповнення даних про структуру електромережі та процеси розвитку каскадних збоїв на основі онтологічної моделі та множини правил, побудовано сценарії розвитку каскадних збоїв в електромережах для тестування розробленого методу;

– у роботі [179] розроблено метод аналізу та порівняння сценаріїв каскадних ефектів у критичній інфраструктурі, розроблено модель автоенкодера на основі графової нейронної мережі, що враховує стани компонентів під час створення представлень про стан електромережі в кроках сценаріїв. Розроблена модель покращує коефіцієнт детермінації для вершин та ребер графу електромережі та зменшує похибку. Використано косинус подібності під час визначення подібності представлень графів: представлення містять інформацію про характеристики компонентів та використовуються для формування бази знань, що може використовуватися в подальшому аналізі;

– у роботі [180] вдосконалено метод аналізу та порівняння сценаріїв каскадних ефектів у критичній інфраструктурі на основі графових нейронних мереж. Розроблено модель автоенкодера на основі графової нейронної мережі, рекурентної нейронної мережі, механізму уваги, що враховує стани компонентів під час створення представлень про стан електромережі в послідовностях кроків сценаріїв. Розроблена модель покращує точність формування представлень. Окрім косинуса подібності, використано алгоритм кластеризації DBSCAN для пошуку множин подібних представлень. Розроблено архітектуру програмного

забезпечення на основі мікросервісної архітектури для обробки даних про роботу компонентів електромереж у режимі реального часу, що надає можливість швидко реагувати на зміни в системі.

Апробація матеріалів дисертації. Результати дисертаційного дослідження доповідалися та обговорювалися на конференціях:

1. IX Міжнародна науково-технічна Internet-конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», 25 листопада 2022 р., Київ, Україна.

2. IV Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)», 9–11 травня 2023 р., Київ, Україна.

3. XXVIII Міжнародна науково-практична конференція «Unusual methods of development of science and thoughts», 17–19 липня 2023 р., Мадрид, Іспанія.

4. I-а Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення», 14 грудня 2023 р., Київ, Україна.

5. XXI Міжнародна науково-практична конференція молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики», 23–26 квітня 2024 р., Київ, Україна.

6. 2 Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення», 13 листопада 2024 р., Київ, Україна.

7. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2025 р., Київ, Україна.

8. XXIV Міжнародна науково-практична конференція «Latest theories and technologies for the development of scientific research», 16–18 червень 2025 р., Сарагоса, Іспанія.

9. III Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення» MoSE-2025, Київ, Україна.

Публікації. За результатами досліджень опубліковано 18 наукових праць. Основні наукові положення опубліковано в 9 наукових статях [182–184, 39–41, 134, 179–180], що входять до переліку наукових фахових видань України. Із них одна наукова стаття [184] опублікована у виданні, що індексується в Scopus. За матеріалами виступів на науково-технічних конференціях опубліковано 9 тез доповідей [185–187, 190–195].

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел із 217 найменувань та 3 додатків. Повний обсяг дисертації складає 250 сторінок. Дисертація містить 72 рисунки, 11 таблиць.

РОЗДІЛ 1. АНАЛІЗ КАСКАДНИХ ЕФЕКТІВ У КРИТИЧНІЙ ІНФРАСТРУКТУРІ

Каскадні збої можуть виникати в різних типах мереж критичної інфраструктури (наприклад, у електричній, водопровідній, комунікаційній, логістичній) у результаті впливу природних явищ на об’єкти або цілеспрямованих атак на мережу. В електричній мережі каскадний ефект визначається як поломка одного компонента, яка має прямі наслідки для продуктивності мережі, але також може спричинити перевантаження та, як наслідок, часткову або повну поломку інших компонентів [1]. Терміни «каскадний ефект» та «ефект доміно» використовуються як взаємозамінні або один використовується для пояснення іншого. Ефект доміно – відносно незначна аварія, яка може ініціювати послідовність подій, які завдають шкоди на значно більшій території та призводять до набагато серйозніших наслідків [2]. Каскадний ефект і ефект доміно можуть не виникати відразу після тригерної події, а проявлятися протягом певного часу в інших об’єктах, які не були джерелом пошкодження.

1.1. Дослідження впливу каскадних ефектів на роботу критичної інфраструктури

Критична інфраструктура – складна система, що складається із множини об’єктів між якими існують взаємозалежності, основними з яких є [3]:

– фізична: стан однієї інфраструктурної системи залежить від матеріальних результатів іншої інфраструктурної системи. Наприклад, збій енергосистем впливає на роботу насосів для подачі води;

– кібернетична: стан однієї інфраструктурної системи залежить від інформації, що передається через інформаційну інфраструктуру. Наприклад, відсутність електроенергії призводить до перебоїв зв’язку, погіршується комунікація між об’єктами системи;

– географічна: локальна екологічна подія може спричинити зміни стану в усіх інфраструктурах. Наприклад, при виникненні ураганів можуть бути пошкоджені лінії електропередачі, зв’язку;

– логічна: стан однієї системи інфраструктури залежить від стану інших через механізм, який не є фізичним, кібернетичним або географічним. Наприклад, при тимчасовому закритті станцій метро зростає навантаження на наземний маршрутний транспорт, завантаженість доріг.

Приклад взаємозалежностей між електроенергетикою, водопостачанням та логістикою зображено на рисунку 1.1.

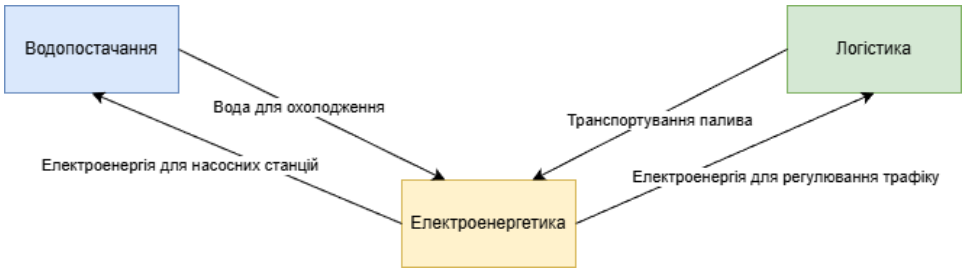

Рисунок 1.1 – Приклад взаємозалежностей в критичній інфраструктурі

Збої у функціонуванні об’єктів критичної інфраструктури негативно впливають на роботу залежних систем, в результаті виникають потенційно небезпечні ситуації, що можуть призвести до часткового чи повного виведення із ладу компонентів системи. Потенційні джерела небезпеки, які впливають на критичну інфраструктуру, можуть бути викликані різними початковими подіями, які класифіковані за характером ризику [4]:

а) природні:

1) геологічні: цунамі, землетруси, зсуви та селеві потоки, виверження вулканів;

2) погодні: повені, лавини, урагани та торнадо, сильні опади або шторми;

3) позаземні: астероїди, метеорити;

б) техногенні:

1) пожежі;

2) вибух;

3) викид токсичних хімікатів;

4) ядерні та радіологічні загрози;

в) антропогенні:

1) організаційні: людський фактор (людська помилка), неправильні / помилкові дії, відсутність дій;

2) злі наміри: саботаж (навмисний зрив роботи або несумлінне її виконання).

У дослідженні сценаріїв аварій, які пов'язані з ефектом доміно в базах даних MHIDAS, MARS, FACTS і ARIA події класифіковані за чотирма типами [5]:

– пожежа (fire);

– вибух (explosion);

– викид (release);

– хмара газу (gas cloud).

На основі даних було визначено: найчастішими причинами виникнення є зовнішні події (31%), механічні пошкодження (30%), людський фактор (21%); найпоширенішими місцями, де траплялися ефект доміно є склади (37%), переробні заводи (27%), транспортування (19%); найпоширенішими послідовностями в деревах подій є вибух-пожежа (21%), викид–пожежа–вибух (15%) і пожежа–вибух (14%) [5].

Прикладом небезпечних наслідків ефекту доміно є землетрус магнітудою 9,0 балів, що викликав цунамі та призвів до страшних наслідків в Японії 2011 року [4]:

– первинні ефекти: руйнування будівель, постраждав аеропорт, пошкоджено нафтопереробний завод, атомна електростанція;

– вторинні ефекти: автомобільний та міський транспорт (порушення роботи міського транспорту), повітряний транспорт (перебої в польотах), електрична мережа (порушення електропостачання), сільське господарство (забруднення ґрунту і води);

– третинні ефекти: фінансовий сектор (фінансові збитки), промисловість (зменшення виробництва).

Метод аналізу впливу каскадних ефектів в критичній інфраструктурі складається з п’яти основних кроків [4]:

– створення бази даних;

– визначення джерел небезпеки;

– визначення взаємозалежностей критичної інфраструктури;

– визначення можливих сценаріїв аварій;

– оцінка наслідків та зменшення ризиків.

Події, які породжують каскадний ефект чи ефект доміно поділяються на категорії, що можуть мати різний степінь деталізації. Події, які ініціювали ефект доміно в хімічній індустрії, на основі аналізу сценаріїв 225 аварій з 1961 по 2007 рік поділяються на категорії [5]: External events, Mechanical failure, Human factor, Impact failure, Violent reaction, Instrument failure, Upset process conditions, Services failure.

Відповідно до різних критеріїв ефекти доміно класифікуються на кілька категорій [6]: Accidental, Natech, Intentional, Fire-induced, Explosion-induced, Internal, External, Direct, Indirect, Temporal, Spatial, Serial, Parallel, Heat radiation-reduced, Overpressure-reduced, Fragment-induced, Coupled.

Аварії, які виникають у результаті ефекту доміно, спричиняють наслідки, які можуть вплинути на промислову інфраструктуру, економіку, людей та навколишнє середовище. Останні роки ймовірність ефекту доміно збільшується через зростання складності процесів, масштабів інфраструктури та наслідків помилки. Тому визначення ризиків та запобігання, пом’якшення наслідків ескалації помилки є актуальним завданням.

Одним із етапів дослідження каскадних ефектів та ефекту доміно в критичній інфраструктурі є аналіз даних про початкові події, їхній розвиток та наслідки, використовуючи сайти, бази даних зі звітами про хімічні аварії [7]:

– ARIA (Analysis, Research and information on Accidents);

– eMARS (Major Accident Reporting System);

– SOZOGAKU (JST Failure Knowledge Database);

– IOGP (International Association of Oil and Gas Producers-IOGP);

– PSID (Process safety incidents reported by member companies of the AICHE CCPS);

– Pro-cessNet. Process Net (German industry);

– ZEMA (Zentrale Melde- und Auswertestelle für Störfälle und Störungen inverfahrenstechnischen Anlagen – ZEMA);

– MAPIS (Major Accident Prevention Information System).

Збої в роботі критичної інфраструктури класифікуються як [3]:

– каскадний: збій в одній інфраструктурі, який спричиняє збій компонента в другій інфраструктурі, що згодом спричиняє збій у іншій інфраструктурі. Наприклад, результат аварії, стихійного лиха або навмисна дія може призвести до збою (порушення) роботи електростанції в місті. Ця подія може призвести до збоїв в інших секторах критичної інфраструктури (наприклад, водопостачання, зв'язок);

– зростання (ескалація): збій в одній інфраструктурі посилює незалежний збій в іншій інфраструктурі, збільшується тяжкість або час для відновлення в іншій інфраструктурі;

– звичайний (загальна причина): компоненти в кожній мережі виходять з ладу через певну загальну причину – основна причина збою широко поширена (наприклад, природна чи техногенна катастрофа).

Електромережі утворюють взаємозв'язані системи, де ризик виникнення каскадних збоїв є високим – збій компонента в мережі може призвести до ланцюга збоїв та до масштабних відключень електроенергії. Каскадний збій – послідовність залежних збоїв окремих компонентів, які послідовно послаблюють енергосистему [8]. Каскад збоїв може бути породжений через причини: нестабільність напруги та частоти, приховані збої систем захисту, помилки програмного забезпечення або оператора, а також перевантаження лінії [9]. Більшість великих збоїв електроенергії починаються з природних явищ: вітер і дощ (24%), блискавка (9%), крижаний шторм (9%); приблизно третина виникає

через неприродні події: несправність обладнання (22%), людська помилка (8%) [10]. Виникнення двох або більше одночасних збоїв може ініціювати каскадні збої, які призведуть до великих відключень електроенергії (наприклад, приблизно 50 мільйонів людей Північної Америки постраждали від відключення електроенергії 14 серпня 2003 року) [10].

Відключення електроенергії можуть вплинути на залежні інфраструктури. Однією з важливих інфраструктур в населених пунктах є водопостачання. Зменшення потужності роботи електромереж впливають на роботу таких систем:

– насосні станції: багато систем водопостачання використовують електричні насоси для транспортування води з очисних споруд до резервуарів, а потім до кінцевих споживачів. При відключенні електроенергії та без резервного живлення насоси перестають працювати, у результаті відбувається зниження тиску води або її відсутність у системі;

– водоочисні споруди: електроенергія необхідна для роботи та фільтрування води. У разі відключення електроенергії якість води потенційно може погіршитися;

– системи розподілу: транспортування води до кінцевих споживачів є важливою задачею, тому в деяких системах можуть бути встановлені резервні джерела живлення для непередбачуваних випадків відключення електроенергії. Проте ресурс цих генераторів є обмеженим. Якщо резервні джерела живлення системи виходять з ладу, це може призвести до перебоїв водопостачання.

Збої у системах водопостачання можуть призвести до катастрофічних наслідків: перебої в забезпеченні населення питною водою, проблеми з доступністю води для використання в домогосподарствах, проблеми з каналізаційними системами, що збільшує ризик поширення інфекцій в населених пунктах [11].

Відключення електроенергії впливає на транспортну інфраструктуру: організація та регулювання руху залізничного, автомобільного, водного та повітряного транспорту. У результаті збоїв в роботі електромереж виникають проблеми з логістикою, рух стає хаотичнішим, збільшується ризик виникнення

аварій, утворюються затори із автомобілів та громадського транспорту на дорогах, що ускладнює доставку товарів в магазини та мобільність населення. В аналізі транспортної мережі використовується Bureau of Public Roads Function (BPR) – математична модель, за допомогою якої можна оцінити затори на дорогах (час у дорозі), використовуючи відповідні параметри. Фактичний час подорожі по відрізку дороги (T) обчислюється за формулою (1.1) [12]:

$$T = T_0 * \left(1 + a * \left(\frac{D}{C}\right)^b\right), \quad (1.1)$$

де T_0 – час подорожі без затор по відрізку дороги;

a, b – параметри, які визначають чутливість часу в дорозі до змін інтенсивності руху;

D – обсяг трафіку на відрізку дороги;

C – пропускна здатність відрізка дороги.

Пропускна здатність відрізка дороги (C) – максимальна кількість транспортних засобів, які можуть бути на відрізку дороги, обчислюється за формулою (1.2). На визначення цього показника впливають різні фактори, наприклад: наявність заторів, правила дорожнього руху, геометрія дороги [12]:

$$C = C_0 * \left(1 - \left(\frac{D}{Q}\right)^p\right), \quad (1.2)$$

де C_0 – пропускна здатність відрізка дороги без заторів;

Q – пропускна здатність відрізка дороги;

p – швидкість зменшення пропускної здатності при збільшенні трафіку.

Обсяг трафіку на відрізку дороги (D) обчислюється за формулою (1.3) [12]:

$$D = C_0 * \left(1 - \left(\frac{C}{C_0}\right)^{\frac{1}{p}}\right). \quad (1.3)$$

Обернена функція BPR використовується для оцінки обсягу трафіку на відрізку дороги в певний момент часу та обчислюється за формулою (1.4) [12]:

$$D = Q * \left(1 - \left(\frac{T}{T_0}\right)^{\frac{1}{b}}\right). \quad (1.4)$$

Затори на дорогах ускладнюють екстреним службам виконувати свою роботу. Наприклад, при виникненні надзвичайної ситуації час відіграє важливу роль, оскільки вчасно надана медична допомога та госпіталізація людини може врятувати життя. При відключенні електроенергії операції в морських портах здебільшого припиняються, але робота аеропортів продовжується (можливі певні обмеження) завдяки системам аварійного живлення та постачання палива. У більшості ситуацій збої в електромережах мають короткочасні наслідки, тобто при відновлення роботи джерел живлення робота об’єктів продовжується, але існують довгострокові наслідки – черги в морських портах, які утворилися за час зупинки портів, оскільки відключення електроенергії впливає на операції завантаження та розвантаження суден [11]. Затримка в логістиці негативно впливає на роботу об’єктів в різних секторах економіки, що призводить каскадних ефектів в залежних секторах та негативних наслідків. Тому у критичних ситуаціях є важливим успішне управління логістикою для забезпечення роботи екстрених служб для вчасного реагування та забезпечення доставки товарів, які є важливими та необхідними для людей.

Каскадний ефект в електромережах – небезпечне явище, що впливає на роботу критичних інфраструктур. Тому для зменшення наслідків каскадних ефектів проводиться аналіз наявних та потенційних сценаріїв роботи електромереж при виникненні збоїв та зміни параметрів системи.

1.2. Аналіз стійкості системи та розвитку каскадних ефектів

Оцінка стійкості системи є важливим чинником для прийняття рішень, які залежать від контексту, у різних сценаріях для підвищення ефективності та оцінки витрат. Розвиток каскадного ефекту складається з фаз [13]:

– попередня фаза (precursor phase): відбувається повільний прогрес збоїв, диспетчеризація, розвантаження та контрольоване відключення можуть пом'якшити вплив каскаду;

– фаза ескалації (escalation phase): збої стрімко поширюються, запобігання відключень стає складнішим;

– фаза каскадного поетапного припинення роботи (cascade phase-out): швидкість розповсюдження збоїв сповільнюється, значна кількість компонентів системи вже вийшли з ладу.

Для визначення продуктивності системи визначається чотири фази та три переходи [14]:

– перша фаза ($t < t_d$): початкова стабільна фаза (original steady phase), у якій продуктивність системи приймає цільове значення;

– друга фаза ($t_d \leq t < t_r$): фаза руйнування (disruptive phase), точка переходу $TRNS(D)$, під час якої продуктивність системи починає падати до досягнення найнижчого рівня в момент часу t_r;

– третя фаза ($t_r \leq t < t_{ns}$): фаза відновлення (recovery phase), точка переходу $TRNS(R)$, під час якої продуктивність системи починає зростати до досягнення нового сталого рівня;

– четверта фаза ($t \geq t_{ns}$): нова стійка фаза (new steady phase), точка переходу $TRNS(NS)$, у якій продуктивність системи досягає та підтримується на новому стабільному рівні.

На основі показників системи в різних фазах обчислюється загальну метрику стійкості (GR), яка базується на кількісному визначенні основних можливостей системи [14]. GR можна використовувати для порівняння стійкості однієї системи до різних руйнівних подій, використовуючи різні стратегії

покращення. В результаті більш ефективні стратегії покращення системи повинні збільшити значення GR.

Крива стійкості відображає зміну показника ефективності $\mathcal{P}(t)$, який відображає стани системи $x \in X$ для значення часу у сценарії та обчислюється за формулою (1.5) [15]:

$$\mathcal{P}(t) = X \to R, t \in [t_0, t_c]. \tag{1.5}$$

Показники продуктивності зазвичай нормалізуються, щоб полегшити порівняння між системами та сценаріями. $\mathcal{P}(t)$ означає ненормалізовану продуктивність, а $p(t)$ позначає нормалізовану продуктивність, $p = \mathcal{P}/\mathcal{R}$, де $\mathcal{R}$ є еталонним, цільовим, базовим або номінальним значенням, яке може або не може змінюватися в часі [15]. Переходи між фазами в інтервалі часу $[t_0, t_c]$, метрики зображені на рисунку 1.2 та описуються як [15]:

– вплив небезпеки (exposure to a hazard), t_h;

– початкове порушення роботи системи (initial system disruption), t_e;

– кінець каскадних збоїв (end of cascading failures), t_d;

– початок відновлення системи (the beginning of system recovery), t_s;

– завершення відновлення системи (the completion of system recovery), t_f.

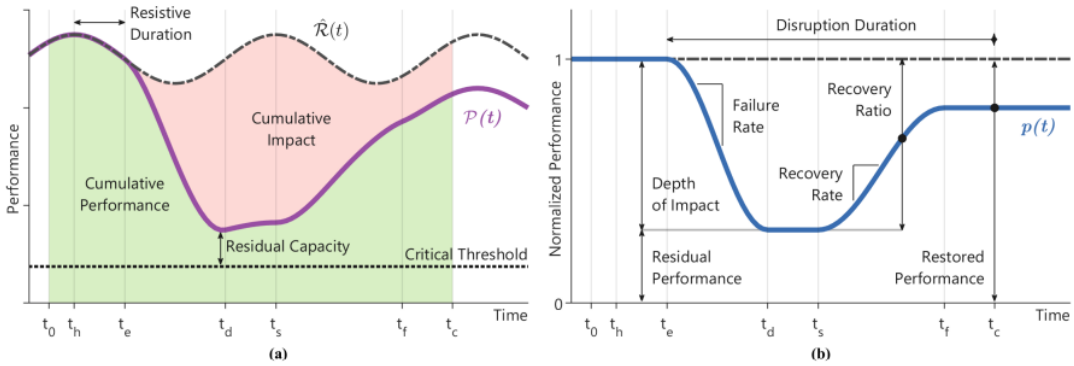

Рисунок 1.2 – Зведені показники для кривої стійкості в термінах (а) фактичних одиниць, $\mathcal{P}(t)$, і (б) нормалізованих одиниць, $p(t)$ [15]

Таксономія показників продуктивності (performance measures) кривої стійкості та зведених показників (summary metrics) складається із множини категорій [15].

Три категорії показників продуктивності [15]:

– доступність (availability): потужність або функціональність системи інфраструктури;

– продуктивність (productivity): кількість послуг, що надаються системою інфраструктури;

– якість (quality): характер послуг, що надаються системою інфраструктури.

Шість категорій підсумкових показників [15]:

– метрики на основі величини (magnitude-based): визначають продуктивність на певному етапі або в певний момент часу;

– метрики на основі тривалості (duration-based): кількісно визначають час між етапами;

– інтегральні (integral-based): включають час і продуктивність;

– метрики на основі швидкості (rate-based): кількісно визначають, як продуктивність системи змінюється з часом;

– порогові значення (thresholds): описують нелінійні переходи в кількісній інтерпретації;

– зведені метрики (ensemble summary): консолідації кількох метрик.

Вибір метрик залежить від кінцевих цілей та задач, які формуються під час аналізу системи. Оцінка стійкості системи може здійснюватися на основі концепції трапеції стійкості, яка відображає фази, у яких може перебувати критична інфраструктура (енергосистема), переходи між цими станами [16].

Трапеція стійкості зображена на рисунку 1.3 та складається з трьох фаз [16]:

– прогрес збою (disturbance progress): $t \in [t_{oe}, t_{ee}]$;

– погіршений стан після збою (post-disturbance degraded): operational resilience $t \in [t_{ee}, t_{or}]$ і infrastructure resilience $t \in [t_{ee}, t_{ir}]$;

– стан відновлення (restorative state): operational resilience $t \in [t_{or}, T_{or}]$ і infrastructure resilience $t \in [t_{ir}, T_{ir}]$.

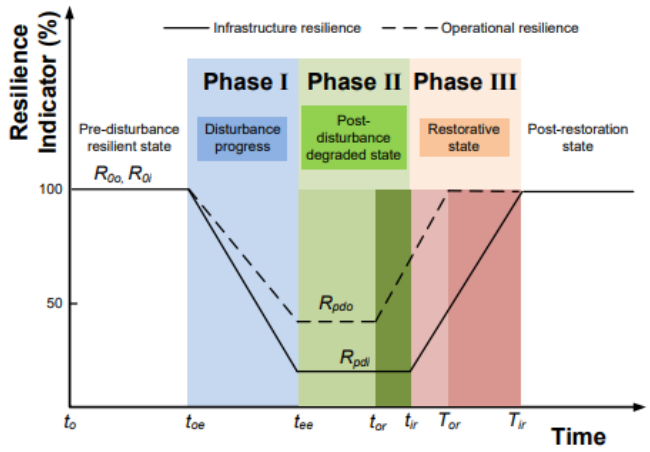

Рисунок 1.3 – Трапеція багатофазної стійкості [16]

Операційна стійкість (operational resilience) – характеристика забезпечення робочої міцності енергетичної системи (наприклад, забезпечення безперебійного постачання споживачам при виникненні катастрофи) [16]. Стійкість інфраструктури (infrastructure resilience) – характеристика фізичної міцності енергетичної системи [16]. Для визначення стійкості енергосистеми використовується набір метрик стійкості «ФЛЕП» [16]:

– Ф: швидкість зменшення стійкості (I фаза). Операційна стійкість Φ_o вимірюється за формулою (1.6) у одиницях (MW/hours):

$$\Phi_o = \frac{R_{pdo} - R_{0o}}{t_{ee} - t_{oe}}, \tag{1.6}$$

де R_{0o} – операційна стійкість до збою;

R_{pdo} – операційна стійкість після збою;

t_{oe} – час початку фази I;

t_{ee} – час кінця фази I.

Стійкість інфраструктури Φ_i вимірюється за формулою (1.7) у одиницях (Number of lines tripped/hours):

$$\Phi_i = \frac{R_{pdi} - R_{0i}}{t_{ee} - t_{oe}}, \quad (1.7)$$

де R_{0i} – інфраструктурна стійкість до збою;

R_{pdi} – інфраструктурна стійкість після збою.

– Λ: рівень падіння стійкості (I фаза). Метрика визначається рівнем деградації операційної стійкості Λ_o та інфраструктурної стійкості Λ_o, які визначені у формулах (1.8), (1.9) та вимірюються у одиницях (MW) та (Number of lines tripped) відповідно:

$$\Lambda_o = R_{0o} - R_{pdo}; \quad (1.8)$$

$$\Lambda_i = R_{0i} - R_{pdi}. \quad (1.9)$$

– E: час, протягом якого система залишається в стані погіршення (фаза II) після збурення та визначається для операційної стійкості E_o та інфраструктурної стійкості E_i відповідно у формулах (1.10) та (1.11) і вимірюється в (Hours):

$$\mathrm{E}_o = t_{or} - t_{ee}; \quad (1.10)$$

$$\mathrm{E}_i = t_{ir} - t_{ee}. \quad (1.11)$$

– П: швидкість відновлення системи до стійкого стану, який був до події (фаза III), обчислюється за допомогою формул (1.12) та (1.13) для операційної стійкості Π_o та інфраструктурної стійкості Π_i в одиницях виміру (MW/hours) та (Number of Lines restored/hours) відповідно:

$$\Pi_o = \frac{R_{0o} - R_{pdo}}{T_{or} - t_{or}}; \quad (1.12)$$

$$\Pi_i = \frac{R_{0i} - R_{pdi}}{T_{ir} - t_{ir}}. \quad (1.13)$$

За допомогою описаного підходу можливо моделювати поведінку інфраструктури, яка піддається впливу подіям, визначити стратегії для підвищення інфраструктурної та операційної стійкості [16].

Стійкість системи характеризується за допомогою термінів [17]:

– robustness: здатність об’єкту, який аналізується, витримувати певний рівень стресу без погіршення чи втрати функціонування;

– redundancy: ступінь існування елементів, які можуть бути замінені – здатні задовольнити функціональні вимоги у разі збою, зменшення або втрати функціональності;

– resourcefulness: здатність виявляти проблеми, визначати пріоритети та використовувати ресурси при виникненні умов, які можуть порушити роботу об’єкта;

– rapidity: здатність вчасно виконувати завдання для запобігання втрат та уникнення майбутніх збоїв.

Стійка система характеризується такими властивостями [17]:

– зменшена ймовірність виникнення збою;

– зменшення наслідків збоїв;

– зменшення часу для відновлення роботи системи до нормального рівня.

Моделювання та аналіз потенційних сценаріїв розвитку каскадних збоїв є важливим завданням для оцінки впливу подій на роботу компонентів системи, наслідків та стійкості системи при різних параметрах середовища.

1.3. Дослідження каскадних ефектів та знеструмлень електромереж

Електромережі є складними системами й відіграють велике значення в сучасному суспільстві: економічні та соціальні аспекти. Попит на електроенергію стрімко зростає, збільшується кількість споживачів, тому

виникає потреба в управлінні розподілу енергії, зниження виробничих витрат та балансування споживання та генерації. Для забезпечення роботи систем у стабільному режимі використовують методи моніторингу та захисту. Сучасні електромережі оснащені схемами захисту для уникнення і обробки непередбачуваних подій, збоїв, оператори тестують надійність та намагаються підтримувати критерій N–1, щоб забезпечити функціонування системи в межах прийнятного рівня роботи у випадку виникнення збоїв або планового відключення [18]. Але системи все ще є вразливими до аварійних ситуацій: помилки операторів, приховані збої, погодні умови, вихід з ладу ліній електропередачі через перегрів провідників ліній, які можуть спричинити каскадні процеси, і, як наслідок, – знеструмлення, що негативно впливає на роботу залежної інфраструктури. Крім того, відновлювані джерела енергії можуть збільшити ймовірність каскадних ефектів, що пов’язано з впливом низки факторів, наприклад: погода, час доби, навантаження (попит на електроенергію). Каскадний збій є небезпечним явищем, що ставить під загрозу функціонування частини або всієї системи. Тому аналіз знеструмлень є важливою частиною дослідження для формування стратегій захисту від каскадних збоїв у електромережах.

30 і 31 липня 2012 року було два знеструмлення в електроенергетиці Індії. На рисунку 1.4 зображено збій, що призвів до знеструмлення, 30 липня 2012 року.

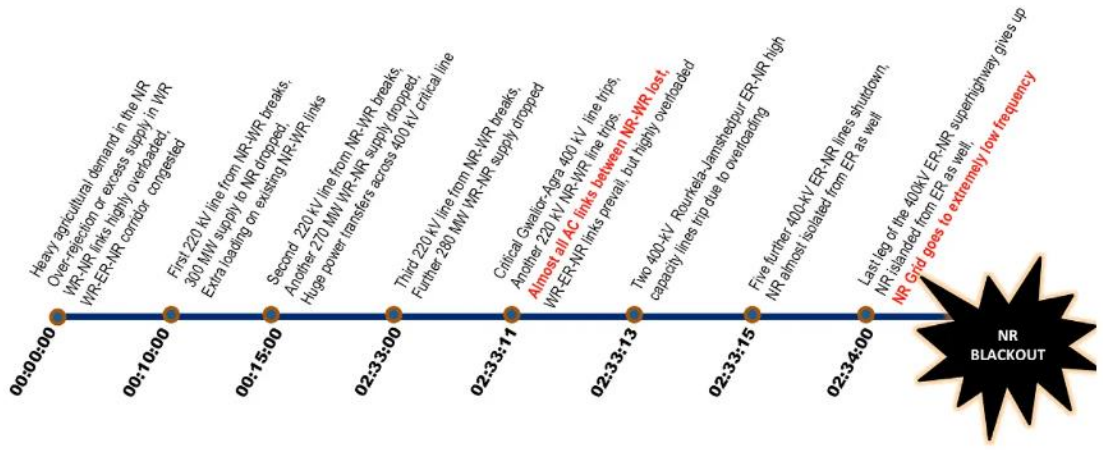

Рисунок 1.4 – Послідовність подій, що призвели до знеструмлення 30 липня 2012 року [19]

Знеструмлення 30 липня 2012 року вплинув на понад 400 мільйонів людей і тривав близько 13,5 годин, втрата потужності приблизно 36 ГВт [20]. Частота перед інцидентом становила 49,68 Гц [21], що зображено на рисунку 1.5.

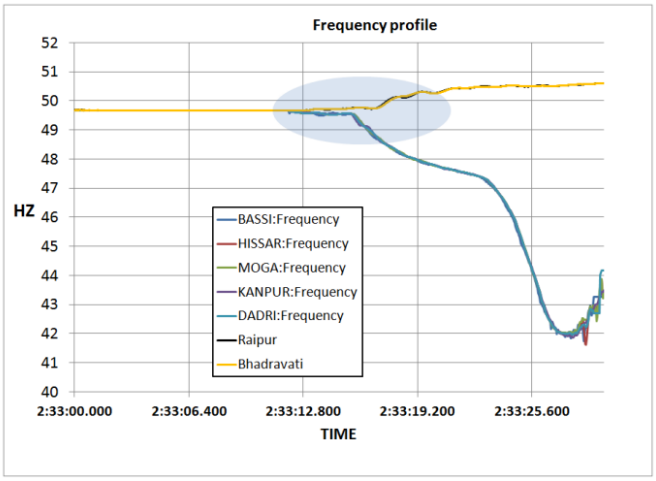

Рисунок 1.5 – Частота в різних місцях у NEW grid до та після інциденту [21]

Знеструмлення 31 липня 2012 року вплинув на понад 620 мільйонів людей, втрата потужності приблизно 48 ГВт [20]. Електропостачання було відновлено між 31 липня та 1 серпня 2012 року. Збій 31 липня 2012 року о 13:00, який зображено на рисунку 1.6, вплинув на Northern, Eastern та North-Eastern електромережі, а частота до інциденту становила 49,84 Гц [21], що зображено на рисунку 1.7.

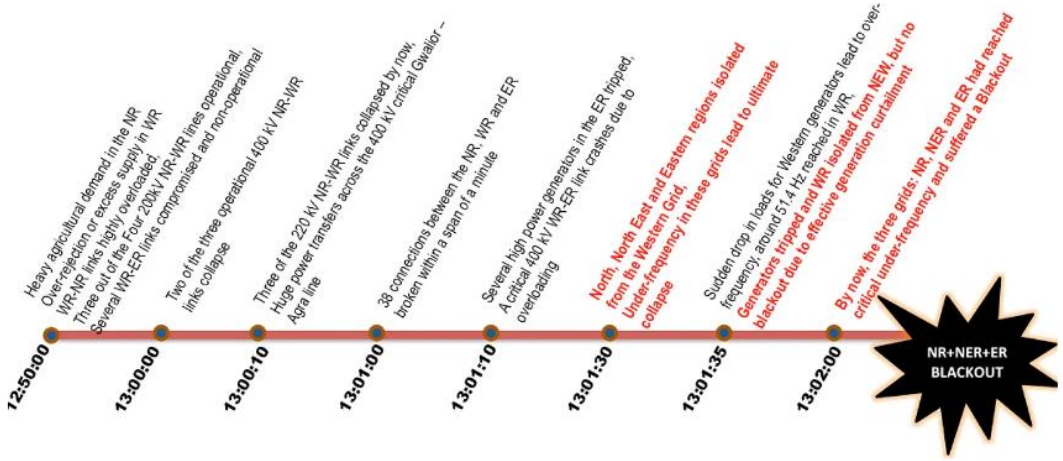

Рисунок 1.6 – Послідовність подій, що призвели до знеструмлення 31 липня 2012 року [19]

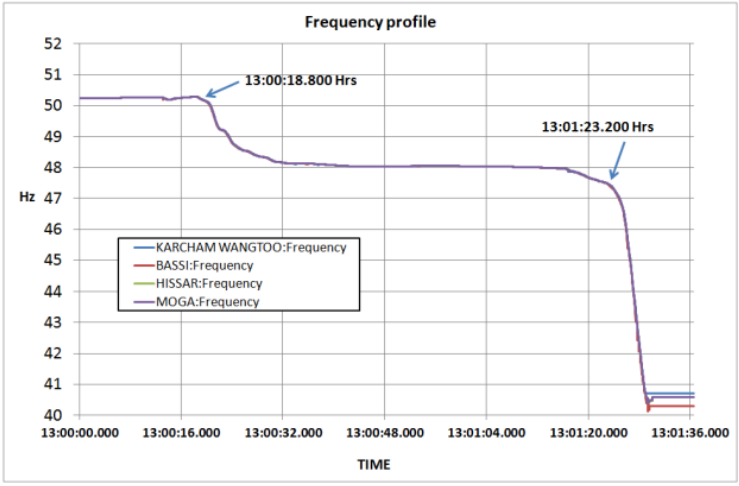

Рисунок 1.7 – Частота в різних місцях у NR grid до та після інциденту [21]

У звіті про збої в електромережі 30 липня 2012 року та 31 липня 2012 року [21] описано попередні умови в мережі 30 та 31 липня 2012 року: частота, потік потужності, напруга у важливих вузлах і відключення ліній перед збоєм у мережі. Попередній аналіз збоїв в електромережі містить аналіз аварійних ситуацій, послідовність подій, вихідні дані про збої, дії, вжиті після їх виникнення. Також описуються основні причини виникнення збоїв та підходи для зменшення ймовірності їхнього повторного виникнення та процес відновлення електромережі [21].

Знеструмлення 14–16 серпня 2003 року вплинуло приблизно на 55 мільйонів людей: 10 мільйонів людей у південному та центральному Онтаріо та 45 мільйонів людей у восьми штатах США, втрата потужності приблизно понад 70 ГВт [22]. У звіті [22] описано умови електричної системи до та під час знеструмлення, події, що призводять до каскаду енергосистеми, рекомендовані дії, які можуть вжиті, щоб запобігти або мінімізувати ймовірність такого збою в майбутньому. Збої з 15:05 на лінії в північно-східному штаті Огайо спричинили велике навантаження на паралельні ланцюги, що призвело до відключення лінії Sammis-Star о 16:05:57, що спричинило каскад відключень ліній високовольтної системи, спричинивши електричні флуктуації та відключення генератора, так що протягом семи хвилин знеструмлення охопило район Клівленд-Акрон [22]. Станом на 16:13 понад 508 енергоблоків на 265 електростанціях вийшли з ладу.

Каскадні події починалися повільно, але швидко поширювалися [22]. На рисунку 1.8 зображено зміну кількості відключених ліній електропередачі в часовому інтервалі.

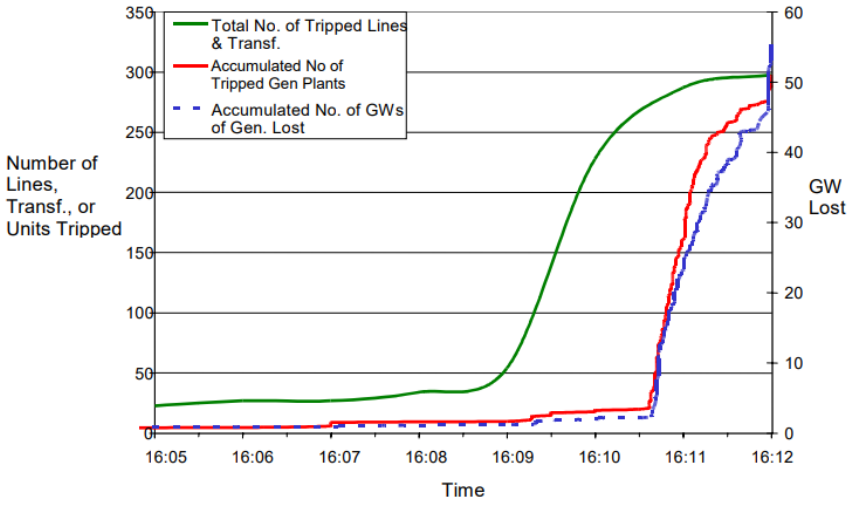

Рисунок 1.8 – Відключення ліній та генераторів під час каскаду [22]

На завершальній фазі, після 16:10:46, великий електричний острів на північному сході мав менше генерації, ніж навантаження, і був нестабільним із сильними стрибками напруги та коливаннями частоти та напруги. У результаті багато ліній і генераторів по всій зоні збоїв відключилися, утворилося кілька електричних островів. Генерація та навантаження на більшості менших островів були незбалансованими, що призводило до подальшого відключення ліній та генераторів, до тих пір поки на кожному острові не було встановлено рівновагу. У значній частині було знеструмлення, але деякі острови змогли досягти балансу між генерацією та навантаженням без повної втрати функціонування [22].

28 вересня 2003 року в Італії відбулося знеструмлення, що вплинуло на близько 56 мільйонів людей та описаний у звіті [23]. Каскадні збої почалися після відключення лінії електропередачі Swiss 380 кВ Mettlen-Lavorgo о 03:01, що було спричинене спалахом дерева. Лінія 380 кВ Mettlen-Lavorgo була навантажена приблизно на 86% від її максимальної потужності, як наслідок процес нагрівання провідників спричинив зменшення відстані до дерев. Спроба операторів ввести в експлуатацію цю лінію зазнала невдачі через занадто високий фазовий кут [23].

Навантаження відключеної лінії було розподілене між іншими лініями електропередачі. Відключення другої лінії 380 кВ Sils-Soazza (Швейцарія) відбулося о 03:25:21, оскільки вона працювала приблизно на 110% номінальної потужності протягом обмеженого інтервалу часу. Оператор мережі мав знизити навантаження цієї лінії до номінального значення протягом досить обмеженого часового проміжку [23]. У результаті втрати двох важливих ліній виникли перевантаження, почалися явища нестабільності: незадовільний рівень низької напруги на півночі Італії, відключення кількох генеруючих станцій в Італії, швидке падіння частоти в Італії було тимчасово зупинено приблизно на 49 Гц, але продовжувала знижуватися та досягла значення 47,5 Гц, що зображено на рисунку 1.9 [23].

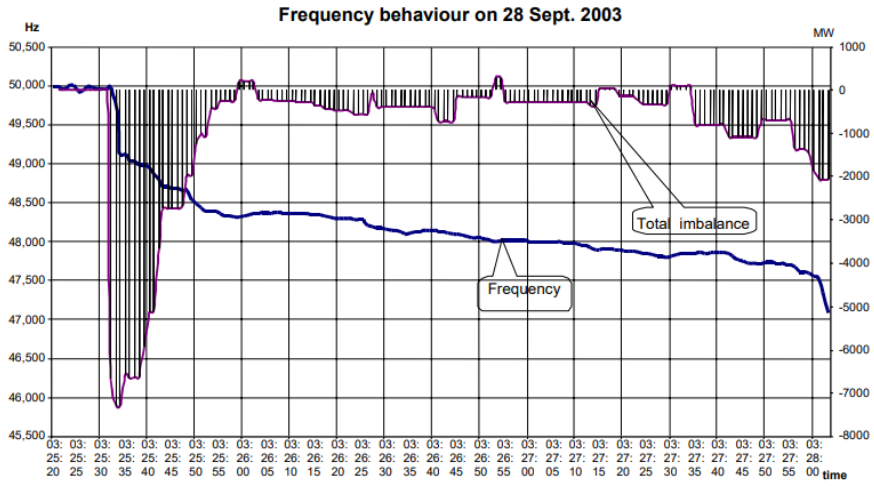

Рисунок 1.9 – Зміна частоти та дисбалансу 28 вересня 2003 року [23]

Значна частина потоку електроенергії була направлена через лінію 220 кВ Airolo-Mettlen (Швейцарія), що в результаті стала дуже перевантаженою і відключилася о 03:25:25. О 03:25:26 відбувся запуск пристрою автоматичного відключення Lienz-Soverzene [23]. В Італії процес відновлення почався відразу після відключення електроенергії: майже вся північна частина Італії була підключена до 08:00, центральна частина близько 12:00, інші частини материкової Італії о 17:00, Сицилія о 21:40 [23].

Знеструмлення електромережі Південної Австралії стався в результаті шторму 28 вересня 2016 року, що пошкодив інфраструктуру передачі електроенергії. В результаті каскадного збою 850000 споживачів штату втратили електроенергію, потреби яких у 1826 МВт забезпечувалися за допомогою [24]:

– 883 МВт вітрової генерації;

– 330 МВт теплової генерації;

– 613 МВт імпорту електроенергії з Вікторії (Victoria).

Погодні умови призвели до п’яти збоїв у системі передачі Південної Австралії з 16:16:46 по 16:18:13, з остаточним виходом з ладу трьох ліній електропередачі. Після цих збоїв відбулося скорочення на 456 МВт вітрової генерації на північ від Аделаїди [24]. Скорочення генерації та збільшення імпорту призвело до активації автоматичного механізму захисту від втрати синхронізації, що призвело до роз’єднання обох ланцюгів передачі Heywood Interconnector. В результаті приблизно 900 МВт постачання з Вікторії через Heywood Interconnector було втрачено, а генерація, що залишилася в Південній Австралії, не змогла задовольнити попит [24]. Раптовий і великий дефіцит генерації спричинив падіння частоти системи швидше, ніж початок дії схеми under frequency load shedding, що розроблена для швидкого відновлення балансу генерації та споживання, зображено на рисунку 1.10 [24].

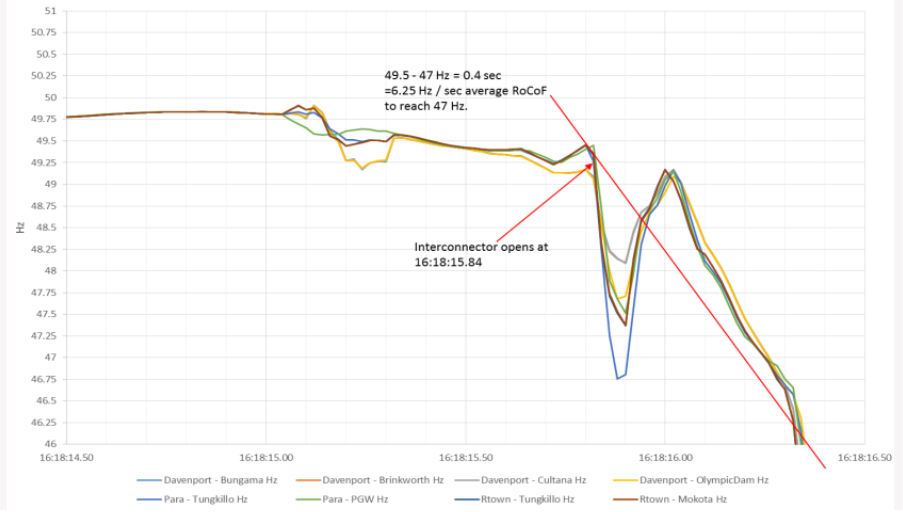

Рисунок 1.10 – Зміна частоти 28 вересня 2016 року [24]

Основною причиною падіння частоти була відсутність достатнього зменшення навантаження, а синхронні генератори та вітрові електростанції, що залишилися, не змогли підтримувати частоту ізольованої системи. У результаті о 16:18 сталося знеструмлення, що зображено на рисунку 1.11. Узагальнений ланцюг подій, потенційні заходи пом'якшення зображено на рисунку 1.12.

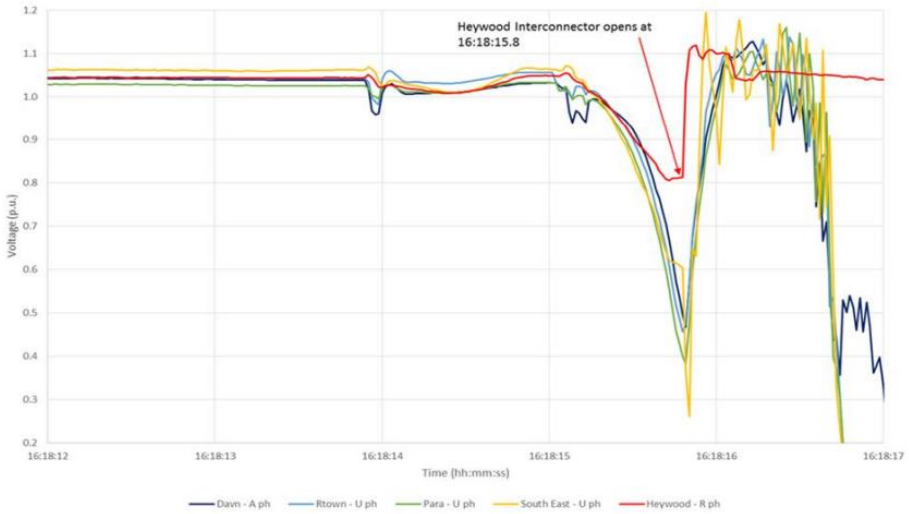

Рисунок 1.11 – Зміна напруги 28 вересня 2016 року [24]

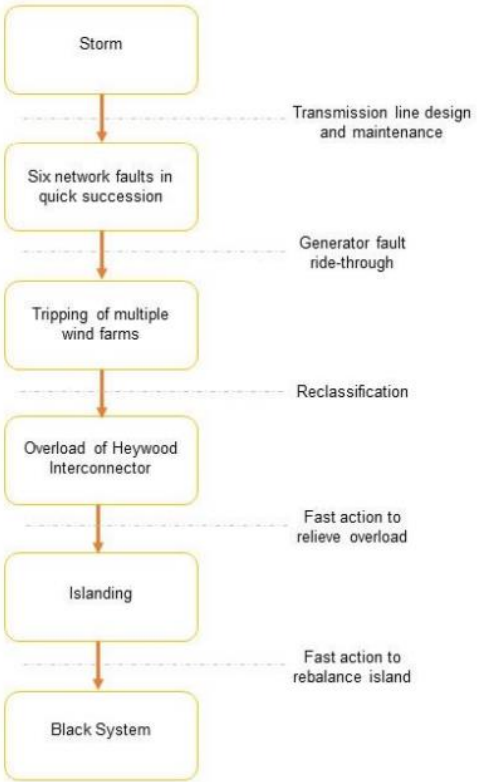

Рисунок 1.12 – Узагальнений ланцюг подій, потенційні заходи пом'якшення [24]

Після оцінки ділянок мережі, почався перезапуск системи. Послідовність відновлення складається з трьох основних, послідовних кроків [24]:

– забезпечити безпеку системи живлення;

– забезпечити допоміжне живлення електростанцій;

– відновити навантаження.

Час відновлення зайняв 7,5 годин, за який відновили постачання електроенергії 80–90% споживачам, приблизно 1 ГВт [24].

Знеструмлення в електромережі Туреччини сталося о 09:36 31 березня 2015 року в результаті послідовності подій [25], що зображені на рисунку 1.13. Перед знеструмленням основний коридор електропередачі зі Сходу на Захід був ослаблений через виведення з експлуатації з метою технічного обслуговування чотирьох ліній електропередачі 400 кВ і 16 послідовних конденсаторних батарей (series capacitor banks) [25], тому система не відповідала критерію безпеки (N–1). У зв'язку з перевантаженням о 09:36:09 відключилася лінія електропередачі Osmanca – Kursunlu, що ініціювало втрату синхронізації між Східною та Західною підсистемами Туреччини зі швидким (1,9 с) послідовним відключенням усіх паралельних ліній реле дистанційного захисту лінії, як наслідок Східна та Західна Турецькі підсистеми були розділені [25]. Дефіцит генерації 4700 МВт (21%) в Західній підсистемі спричинив втрату синхронізації з енергосистемою Continental European та від'єднання від неї шляхом відключення трьох ліній електропередачі з болгарською та грецькою мережами [25]. У Східній підсистемі був надлишок гідроенергії (41%), яка не могла транспортуватися на захід. Схеми відключення навантаження (Load shedding schemes) стабілізували частоту в Західній підсистемі, але деякі електростанції не справлялися з роботою на зниженій частоті (47,5 Гц), додаткова генерація була втрачена, що призвело до знеструмлення Західної підсистеми приблизно за 10 секунд [25]. Східна підсистема була прискорена з градієнтами приблизно 1,6 Гц/с до 52,3 Гц. Через підвищення частоти електростанції вийшли з ладу, а частота була менше 47,0 Гц. Після знеструмлення розпочалося відновлення

електромережі, у результаті чого о 18:30 відсоток відновлення системи дорівнював 95% [25].

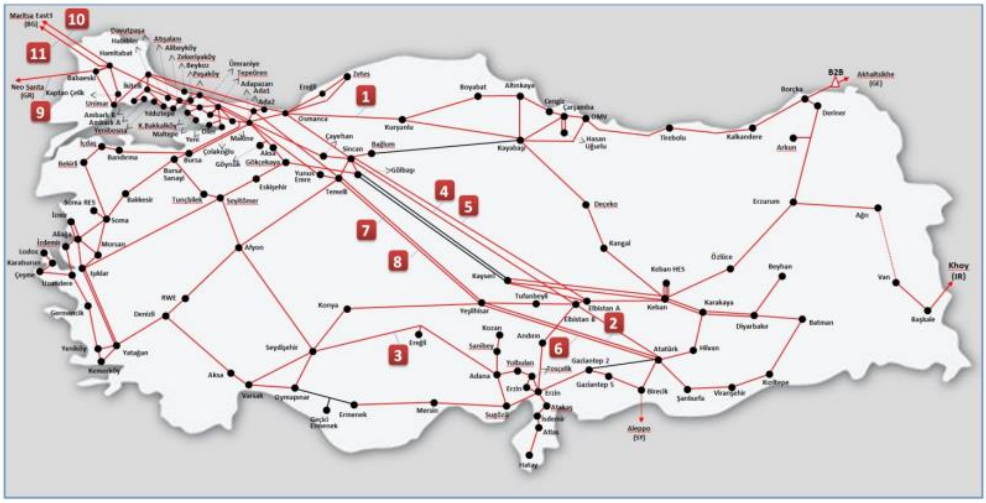

Рисунок 1.13 – Послідовність розвитку подій в електромережі [25]

У звіті про збитки інфраструктури від руйнувань внаслідок військової агресії Росії проти України станом на січень 2024 року [26] зазначено, що найбільше в енергетиці постраждала галузь виробництва та передачі електроенергії. За попередніми оцінками, загальна сума збитків, завданих цим об'єктам, становить понад 7,4 мільярда доларів [26]. Пошкодження критичної інфраструктури, впливає на роботу, функціонування залежних об’єктів, наприклад, без електроенергії та газопостачання залишаються підприємства та споживачі, статистика наведена в таблиці 1.1.

Таблиця 1.1 – Статистика про знеструмлення населених пунктів, відсутність електропостачання та газопостачання в споживачів

Дата	**Електромережі**	**Газопостачання**
20 травня 2022 року [27]	677 населених пунктів	Понад 161 тисяча абонентів
	635,8 тисяч споживачів	
1 червня 2022 року [28]	767 населених пунктів	169 тисяч абонентів
	632 тисячі споживачів	

Продовження таблиці 1.1

Дата	Електромережі	Газопостачання
10 червня 2022 року [29]	744 населені пункти 640,75 тисяч споживачів	172 тисячі абонентів
15 червня 2022 року [30]	722 населені пункти 613,5 тисяч споживачів	178,5 тисяч абонентів
20 червня 2022 року [31]	741 населений пункт 606,6 тисяч споживачів	176,7 тисяч абонентів
1 вересня 2022 року [32]	785 населених пунктів 615,2 тисяч споживачів	234,9 тисячі абонентів
27 вересня 2022 року [33]	1409 населених пунктів 740,7 тисяч споживачів	625,9 тисяч абонентів

Каскадні збої є небезпечним явищем у критичній інфраструктурі, особливо в електромережах. Вони виникають унаслідок різних чинників, тому важливо реагувати на ці події, протистояти їм та пом’якшувати наслідки.

1.4. Дослідження розвитку каскадних збоїв в електромережі

На розвиток каскадних збоїв впливає стан системи, її налаштування, рішення та дії, які приймаються автоматично та/або вручну в певний момент часу. Щоб сформувати порядок дій для зменшення або зупинення каскаду потрібно проаналізувати процес розвитку збоїв, що складається з наступних етапів:

– передумови виникнення збоїв;

– події, що породжують збої в електромережі;

– збої, які розвиваються в результаті каскадного ефекту;

– наслідки збоїв.

Передумови виникнення збоїв – це множина умов, при яких електромережа перебуває в стабільному стані, до виникнення збоїв. При дослідженні

знеструмлень визначаються та аналізуються передумови, що запустили ланцюг збоїв і призвели до знеструмлення мережі [34], основними з яких є:

1. Піковий попит. Мережа працює під великим навантаженням, близько до робочих лімітів і стабільності [34]. Частота виникнення знеструмлення значно зростає в кінці літа та в середині зими, спостерігається значна сезонна тенденція, протягом доби ймовірність зростає в години пік [35].

2. Помилки чи несправність у роботі обладнання. Обладнання або компонент системи може перебувати в станах, зображені на рисунку 1.14 [36]: нормальний стан (normal state) – компонент виконує призначену функцію; стан обмеженого функціонування (limited state) – компонент виконує призначену функцію, але з обмеженою потужністю; стан несправності (fault state) – компонент не виконує призначену функцію. Несправність – це будь-який дефект або відхилення, що призводить до того, що компонент не може виконувати свою призначену функцію в енергосистемі [36]. Несправності поділяються на три категорії: технічні, операційні та системні. Обладнання може вийти з ладу через несподівані технічні проблеми, вимушене відключення, заплановане відключення (технічні роботи).

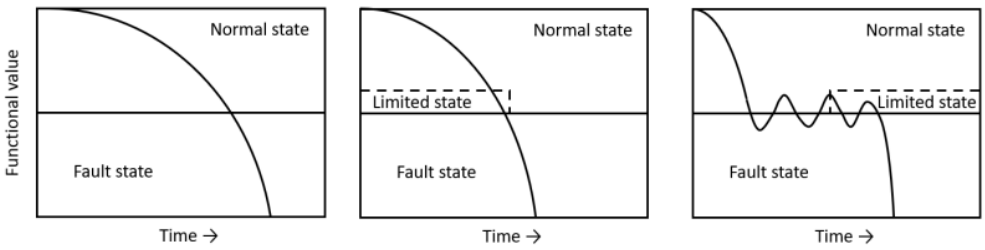

Рисунок 1.14 – Перехід між станами компонента системи [36]

3. Природні умови та навколишнє середовище, що можуть бути важливим фактором та передують виникненню знеструмлень [34, 35], наприклад: сильний вітер, сніг, дощ, блискавка, температура, повінь, рослинність. Погодні явища є найпоширенішими ініціаторами потенційних каскадних ефектів, що є найбільшою загрозою для електромереж [37].

4. Недостатній запас реактивної потужності. Недостатня реактивна потужність може призвести до падінь напруги, а надлишкова реактивна потужність може спричинити підвищення напруги, що може призвести до нестабільності напруги або навіть негативним наслідкам [34].

5. Невідповідність між запланованим і фактичним потоком електроенергії. Дисбаланс потужності в системі може призвести до нестабільності роботи електромережі. Тому здатність електромережі підтримувати стабільну роботу після збоїв є важливим завданням.

Події, що породжують збої можуть бути узагальнені у 5 категорій з підкатегоріями: екологічні причини, зовнішні впливи, експлуатація та обслуговування, технічне обладнання та невідомі [36]. Каскадний ефект виникає внаслідок однієї чи сукупності початкових подій [37], основними з яких є:

1. Коротке замикання: може статися через природні явища (наприклад, flashover) або технічні проблеми (наприклад, порушення ізоляції проводів).

2. Перевантаження: через лінію електропередачі транспортується занадто великий струм, що перевершую її ліміт, дріт нагрівається і може розплавитися, що може призвести до пожежі, пошкодження функціонування лінії. Тому використовуються захисні пристрої, які можуть відключити лінію, щоб уникнути пошкодження. Вимкнення перевантаженої лінії електропередачі як засіб захисту системи може призвести до каскадного ефекту.

3. Зниження напруги: події, в результаті та під час яких зниження напруги змінного струму відбувається протягом тривалого періоду часу. Це може призвести до поганої роботи електричного обладнання.

4. Підвищення напруги: події, в результаті та під час яких напруга підвищується і перевищує ліміт, що може призвести до пошкодження обладнання. В результаті термін експлуатації електричного обладнання зменшиться.

5. Приховані збої в пристроях захисту: пристрій захисту спрацьовує в тій ситуації, в якій не повинен спрацьовувати відповідно до налаштувань або не спрацьовує, тоді як мав би. Це може призвести до непередбачуваних наслідків.

6. Підвищення частоти: нестабільність частоти виникає в результаті дисбалансу між споживанням і генерацією електроенергії. Підвищення частоти може призвести до перегріву або пошкодження обладнання та інфраструктури. Це може дестабілізувати мережу, що призведе до потенційних каскадних збоїв.

7. Зниження частоти: зниження частоти може призвести до зменшення продуктивності обладнання, неефективної роботи або його пошкодження, нестабільності в електромережі. Якщо вчасно не вжити необхідних дій, це може спричинити каскадні збої, знеструмлення.

Стабілізація частоти, зображено на рисунку 1.15, є важливим завданням для забезпечення роботи електромережі, відхилення значення може призвести до серйозних аварій.

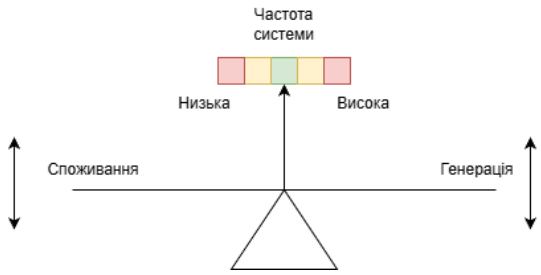

Рисунок 1.15 – Зменшення та підвищення частоти в електромережі

Збільшення стійкості системи, зменшення негативного впливу збоїв на систему потенційно зменшує розвиток каскадних ефектів, тому доцільно досліджувати роботу інфраструктури в різних сценаріях виникненні збоїв та розвитку каскадних ефектів [38 – 41].

1.5. Аналіз методів, вимог, розроблення, впровадження та супроводження програмного забезпечення моделювання роботи критичної інфраструктури

Під час аналізу роботи критичної інфраструктури використовуються різні методи, що дозволяють досліджувати різні властивості систем: стійкість, вразливість до каскадних ефектів, здатність до відновлення після збоїв.

1.5.1. Аналіз застосовування теорії графів для оцінки властивостей та моделювання роботи критичної інфраструктури

Об'єкти критичної інфраструктури та зв'язки між ними можливо описати за допомогою графу $G = (V, E)$, де $|V| = n$ – кількість вершин, $|E| = m$ – кількість ребер (дуг). Формальне визначення графу [42]: нехай скінчена множина $V = \{v_1, v_2, \dots, v_n\}$ невизначених елементів і нехай $V \otimes V$ – множина всіх упорядкованих пар $[v_i, v_j]$ елементів V. Відношення на множині V – будь-яка підмножина $E \subseteq V \otimes V$. Відношення E є симетричним, якщо $[v_i, v_j] \in E$ і $[v_j, v_i] \in E$. Відношення E є рефлексивним, якщо $\forall v \in, [v, v] \in E$. Відношення E є антирефлексивне, якщо $[v_i, v_j] \in E$ і $v_i \neq v_j$. Простий граф (simple graph) [42] визначається як $G = (V, E)$, де V – скінченна множина вершин, а E - симетричне та антирефлексивне відношення на V, елементи якого є ребра (дуги) графа. В орієнтованому графі (directed graph) [42] відношення E є несиметричним. Для дуг графа можуть бути встановлені значення (ваги) – дійсні числа, що визначають певну властивість ребра. У цьому випадку зручно наступне більш загальне визначення. Зважений граф (weighted graph) [42] визначається як $G = (V, E, W, f)$, де V – скінченна множина вершин, $E \subseteq V \otimes V = \{e_1, e_2, \dots, e_m\}$ – множина дуг, $W = \{w_1, w_2, \dots, w_r\}$ – множина ваг, де $w_i \in R$ і $f: E \to W$– вага кожного ребра. Якщо ваги є натуральними числами, то отриманий граф є мультиграфом (multigraph), у якому між парами вершин може бути кілька ребер. Якщо між вершинами p і q вага $k \in N$, то між цими вершинами існує k зв'язків. Вразливі та критичні компоненти в мережі можливо визначити за допомогою метрик теорії графів та провести аналіз подій, які можуть породжувати каскадні ефекти. Для аналізу графів використовуються матриця суміжності A і матриця Лапласа L. Матриця суміжності A визначається за формулою (1.14):

$$A_{ij} = \begin{cases} w_{ij}, \text{якщо } (i,j) \in E \\ 0, \text{в іншому випадку} \end{cases}. \quad (1.14)$$

Матриця Лапласа визначається за формулою (1.15):

$$L_{ij} = \begin{cases} d_i, \text{якщо } i = j \\ -1, \text{якщо } i \text{ є суміжною з } j, \\ 0, \text{в іншому випадку} \end{cases} \quad (1.15)$$

де d_i – степінь вершини (degree of node) i.

Для дослідження властивостей, характеристик графів вчені впроваджують і використовують різні метрики, які поділені на класи [43]:

– відстань (distance): hop count, closeness, eccentricity, diameter, radius, girth, expansion, betweenness;

– зв'язність (connection): degree, entropy, joint degree, assortativity, coreness, clique, rich club coefficient, giant component, reliability, chromatic number;

– спектри (spectra): algebraic connectivity, spectral radius, spectral partitioning, principal eigenvector.

Для оцінки міцності мережі використовуються метрики, що згруповані в шість категорій [44]:

– кластеризація (clustering);

– зв'язність (connectivity);

– відстань (distance);

– пропускна здатність (throughput);

– спектральні методи (spectral methods);

– географічні метрики (geographical metrics).

Перш ніж вибрати метрики для оцінки характеристик графу (мережі), потрібно враховувати тип графу, який досліджується, оцінити співвідношення між точністю та обмеженнями при обчисленні.

Графові спектральні методи (Graph spectral techniques) застосовуються для управління водопровідною мережею, які можуть бути використані для [45]:

1. Виявлення вузьких місць (bottlenecks). Вузькі місця визначаються за допомогою значень спектрального розриву (spectral gap), який визначається у

формулі (1.16) як різниця між найбільшим і другим за величиною власними значеннями матриці суміжності A:

$$\Delta\lambda = \lambda_1 - \lambda_2. \quad (1.16)$$

Ця метрика відображає міцність зв’язків мережі: наявність вузьких місць, мостів – ребро, видалення якого збільшує кількість зв’язаних компонентів графа й розділяє мережу на дві або більше частин. Що більше значення спектрального розриву, то стійкішою є мережа [45].

2. Вимірювання міцності (strength) мережі для розбиття на підмережі. Алгебраїчна зв’язність (Algebraic connectivity) λ_2 – друге найменше власне значення обчислюється на основі матриці Лапласа і визначає міцність з’єднань мережі і стійкість до збоїв. Чим більше значення алгебраїчної зв'язності, тим стійкішим є граф (мережа). Тобто мережу з більшою алгебраїчною зв’язністю важче роз’єднати [45]. Спектральний радіус або індекс (Spectral radius or index) – найбільше власне значення λ_1, яке обчислюється на основі матриці суміжності A і може використовуватися як метрика для оцінки рівня зв’язності мережі. Якщо граф добре зв’язаний, то більше значення спектрального радіусу і більша міцність мережі до збоїв [45].

Орієнтований граф, який не містить петель використовується для моделювання залежності між акторами критичної інфраструктури: вершина – актор (доповнений кінцевим автоматом, який відображає статус), ребро – зв’язок між двома акторами [46]. Актори можуть створювати події, які поширюються шляхами в графі і можуть впливати на системи. Моделювання критичної інфраструктури здійснюється за допомогою множини взаємодіючих автоматів, де стани змінюються на основі результатів інших автоматів. Кінцевий автомат актора Actor State Machine визначений у формулі (1.17) [46]:

$$ASM = (Q, \Sigma i, \Sigma o, T, O, D, S, q0), \quad (1.17)$$

де Q – скінченна множина станів;

$\sum_i$ – скінченна множина вхідних подій;

$\sum_o$ – скінченна множина вихідних подій;

$T: Q \times \sum_i \to Q$ – функція переходу;

$O: Q \times \sum_i \to \sum_o$ – вихідна функція;

$D: Q \times \sum_i \to Q \times R^+$ – функція переходу стану із затримкою;

$S: Q \to [0,1]$ – функція стану (0 – погане функціонування, а 1 – повна функціональність);

$q0$ – початковий стан.

ASM може містити три стани Q [46]: OK – актор здатний пропонувати ресурси залежним системам; PRE-FAIL (скорочено PF) – актор функціонує належним чином і може надавати ресурси своїм залежним особам, але одна або кілька його власних залежностей не функціонують, що вказує на можливий майбутній збій за короткий проміжок часу; FAIL (скорочено F) – актор не в змозі виробляти необхідні ресурси для підтримки акторів, які від нього залежать. На основі моделювання визначається вплив події на критичну інфраструктуру [46].

Визначення центральних вершин у мережі є важливою задачею для створення стійкої до збоїв або атак мережі. Метрики центральності класифікуються у три групи [47]:

– метрики центральності вершини (point centrality metrics);

– метрики центральності графа (graph centrality metrics);

– метрики центральності вибору групи (group selection centrality metrics).

Метрики оцінки центральності використовуються як додаткові критерії для визначення критичних вершин інфраструктури в графі для подальшого застосування методів пом’якшення ризиків [48].

Теорія графів застосовується для аналізу вразливостей у транспортних мережах [49]:

1. Середня відстань між усіма парами вершин графу (average distance) визначається за формулою (1.18) [50]:

$$\bar{d} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}\,, \tag{1.18}$$

де n – кількість вершин;

d_{ij} – довжина найкоротшого шляху між вершинами i та j.

Що нижче значення метрики, то сильніша зв'язність – більша міцність мережі. Середня відстань більш гнучка метрика, ніж діаметр $d(G)$ (найдовший найкоротший шлях між усіма парами вершин), яка визначена у формулі (1.19). Значення першої метрики зменшується при додаванні ребер, тоді як значення другої метрики може залишатися незмінним [50]:

$$d(G) = max_{x,y \in V(G)}\{d(x,y)\}. \tag{1.19}$$

2. Ефективність (Efficiency), визначається за формулою (1.20) [50]:

$$E = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{d_{ij}}. \tag{1.20}$$

При обчисленні значення ефективності використовуються зворотні величини довжини шлях: що вище значення метрики, то більша міцність мережі [50]. Додатковим показником ефективності є відносний розмір найбільшого компонента зв'язності (S): що вище значення метрики, то більша зв'язність і міцність графа. Відносний розмір найбільшого компонента зв'язності визначається за формулою (1.21) [49, 51]:

$$S = \frac{L}{n}, \tag{1.21}$$

де L – кількість вершин у найбільшому компоненті зв'язності;

n – кількість вузлів у вихідній мережі.

Додатковими метриками для оцінки міцності мережі є: Average Edge Betweenness та Average Vertex Betweenness: менше значення означає вищу міцність мережі. Метрики обчислюються за формулами (1.22) та (1.23) відповідно [50]:

$$\overline{b_v} = \frac{1}{2}(n-1)\left(\bar{d}+1\right); \tag{1.22}$$

$$\overline{b_e} = \frac{n(n-1)}{2m}\bar{d}. \tag{1.23}$$

Виявлення та запобігання виникнення вузьких місць в транспортних мережах є однією із важливих задач. Основними типами вузьких місць є [52]:

– вузькі місця інфраструктури (Infrastructure bottlenecks), які можуть виникати в результаті дії постійних або тимчасових умов;

– вузькі місця регулювання (Regulations bottlenecks), які можуть виникати в результаті правил/обмежень, які затримують рух;

– вузькі місця в ланцюзі поставок (Supply chain bottlenecks), які можуть виникати в результаті певних завдань в управлінні ланцюгами поставок.

Для характеристики вразливості транспортної мережі визначаються параметри на основі теорії графів – темах домінування [52]:

– кількість критичних вершин;

– кількість критичних ребер;

– топологічна (структурна) вразливість транспортної мережі;

– зважена структурна вразливість транспортної мережі;

– вузькі місця вершин транспортної мережі;

– вузькі місця дуг транспортної мережі;

– вразливість споживача.

Визначені алгоритми застосовуються для мереж різної розмірності (більше часу необхідно для аналізу великих мереж) та можуть використовуватися в динамічному аналізі вузьких місць у транспортних мережах.

Коефіцієнт кластеризації (Clustering coefficient) є корисною метрикою для визначення надмірності шляху та існування циклічних альтернативних шляхів водопостачання [53], і вимірює щільність транзитивних трикутників у мережі. Метрика наведена у узагальненому вигляді у формулі (1.24) та детальніше у формулі (1.25):

$$C = \frac{3N_{\Delta}}{N_3}; \tag{1.24}$$

$$C = \frac{1}{n} \sum_{i \in V; \delta_i > 1} \frac{2}{\delta_i(\delta_i - 1)} e_i\,, \tag{1.25}$$

де N_{Δ} – кількість трикутників (закритий триплет);

N_3 – кількість зв'язаних мережевих триплетів (три вершини, які з'єднані або двома ребрами (відкритий триплет), або трьома ребрами (закритий триплет));

δ_i – степінь (кількість сусідів) вершини i та e_i – кількість ребер (дуг) між сусідами i.

При збільшенні коефіцієнта кластеризації утворюються надлишкові шляхи між сусідами та підвищується міцність мережі.

Метрики оцінки стійкості мережі групуються в категорії на основі їхніх властивостей та цільового застосування. Наприклад, під час аналізу стійкості в телекомунікаційних мережах таксономія метрик складається з [54]:

– структурні метрики (structural metrics): використовуються для визначення рівня стабільності у мережі та визначення поширення збоїв мережею під час видалення вершини/ребра [54];

– метрики центральності (centrality metrics): використовуються для визначення елементів в мережі, які є найважливішими або центральними [54];

– функціональні метрики (Functional metrics): використовуються для визначення зміни продуктивності мережі у результаті численних збоїв [54].

Важливим етапом дослідження каскадних збоїв є аналіз впливу збоїв на роботу компонентів в системі. Виникнення збоїв в об’єктах електромережі може спричинити незбалансоване навантаження в системі, виникає каскадний ефект, який в результаті може призвести до відключення електроенергії та руйнування електромережі. Процес розвитку каскадних збоїв складається з трьох етапів [55]:

1. Початковий етап. Через вплив зовнішніх чинників пошкоджені об’єкти виходять з ладу через обмежену потужність. При вчасному виявленні збоїв можливо вжити відповідних заходів для локалізації та зменшення впливу аварії [55].

2. Етап розширення. Збої продовжують поширюватися та збільшується в масштабі, що призводить до зміни навантаження, збільшується кількість пошкоджень в електромережі. На етапі розширення ще можливо частково контролювати процес розвитку аварії в мережі [55].

3. Етап колапсу. Збої накопичуються, збільшується навантаження, що з часом може призвести до виведення із ладу системи електромережі [55].

Аналіз стійкості та вразливості в електромережах здійснюється за допомогою концепцій теорії складних мереж, які складається з двох різних підходів: топологічний та гібридний [56]. Топологічний підхід базується на структурі та її топологічних концепціях (структура електромережі представлена у вигляді графа, де вершини – станції та підстанції, ребра – зв’язки між вершинами. Для оцінки характеристик об’єктів використовуються різні метрики, поширеними є: середня довжина шляху (average path length), node degree distribution, betweenness, розмір найбільшого компонента (size of the largest component) та ефективність мережі [56]. У гібридному підході використовуються концепції з електротехніки (наприклад, імпеданс лінії) для покращення топологічного підходу, мережа представлена орієнтованим, зваженим графом, оскільки лінії мають різний імпеданс та обмеження потоку електроенергії, а електроенергія тече від шин генератора до шин навантаження, тому метрики із

топологічного підходу змінюються на відповідні аналоги: net-ability, electrical degree centrality, electrical betweenness centrality, entropic degree, effective graph resistance [56].

Мережева здатність (net-ability) використовується для оцінки ефективності електромережі з урахуванням особливостей електричних мереж (обмеження та розподіл потоку електроенергії в мережі, які зумовлені законами фізики), визначається за формулою (1.26) [56–59]:

$$A = \frac{1}{N_G N_D} \sum_{i \in G} \sum_{j \in D} \frac{C_{ij}}{Z_{ij}}, \tag{1.26}$$

де G, D – множини вершин генератора та навантаження (елементи, що споживають електроенергію, підключені до генератора);

N_G, N_D – загальна кількість генераторів і навантажень;

C_{ij} – можливості передачі електроенергії пари генерації та навантаження (i, j);

Z_{ij} – еквівалентний імпеданс кола, кінцями якого є вершини i та j.

Вразливість лінії l визначається за формулою (1.27) як падіння «net-ability», спричинене відключенням (розривом) лінії l [57]:

$$V_A(l) = \frac{A - A_l}{A}. \tag{1.27}$$

Electrical betweenness лінії [56, 58] визначається за формулою (1.28):

$$B_e(v) = \frac{1}{2} \sum_{g \in G} \sum_{d \in D} C_g^d \sum_{l_{ij} \in L^V} f_{l_{ij}}^{gd}, \tag{1.28}$$

де L^V – множина ліній, які з'єднують шину (bus) v.

«Пропускна здатність передачі електроенергії» від шини g до d, визначається за формулою (1.29):

$$C_g^d = min\left[\frac{P_{l_{ij}}^{max}}{\left|f_{l_{ij}}^{gd}\right|}\right], \tag{1.29}$$

де $P_{l_{ij}}$ – фізичне обмеження потоку потужності лінії l;

$f_{l_{ij}}^{gd}$ – потужність на лінії l для одиниці потужності, що надходить на шину генерації g і споживається на шині навантаження d [56, 58].

Ступінь ентропії (entropic degree) e_i^w вимірює критичність вершини i в мережі та враховує наступні критерії: міцність зв'язку між вершинами i та j, враховуючи вагу w_{ij}; кількість ребер, які з'єднані із вершиною; розподіл ваг між ребрами [56, 58]. Ступінь ентропії визначається за формулою (1.30), нормалізована вага ребра (p_{ij}), яке з'єднує вершини i та j визначається за формулою (1.31):

$$e_i^w = \left[1 - \sum_{j=1}^{N} p_{ij} \log\left(p_{ij}\right)\right] \sum_{j=1}^{N} w_{ij}\,; \tag{1.30}$$

$$p_{ij} = \frac{w_{ij}}{\sum_{j=1}^{N} w_{ij}}, \tag{1.31}$$

де w_{ij} – обмеження потоку на лінії l_{ij}.

Для визначення критичних компонентів (ліній електропередачі в електромережі) може використовуватися ефективний опір графа (effective graph resistance), який враховує електричні властивості електромереж – розподіл потоку електроенергії відповідно до законів Кірхгофа [60, 61]. Ефективний опір графа досліджується як метрика для розширення мережі для підвищення стійкості до каскадних збоїв [61]. Ефективний опір графа R_G можна обчислити за формулою (1.32) або за формулою (1.33) та має обмеження, що визначається за формулою (1.34) [62]:

$$R_G = \sum_{1 \le i \le j \le N} R_{ij}\,; \tag{1.32}$$

$$R_G = N \sum_{i=2}^{N} \frac{1}{\mu_i}; \tag{1.33}$$

$$\frac{N}{\mu_2} < R_G \le \frac{N(N-1)}{\mu_2}, \tag{1.34}$$

де R_{ij} (Effective resistance) – це електричний опір між вершинами i та j;

μ_i – ненульове власне значення оператора Лапласа.

Два ребра, які з'єднані послідовно і відповідають резисторам з опором r_1 та r_2 Ом, можна замінити одним ребром з ефективним опором $(r_1 + r_2)$. Якщо два ребра з'єднані паралельно, то їх можна замінити ребром з ефективним опором $({r_1}^{-1} + {r_2}^{-1})^{-1}$. Що нижчий ефективний опір графа, то надійніша електромережа та стійкіша каскадних збоїв. Додавання ребер або збільшення ваги ребер зменшує ефективний опір графа [62]. Стійкість електромереж оцінюється за допомогою показника критичності додавання лінії, що визначається за формулою (1.35) [61], або видалення лінії, що визначається за формулою (1.36), [60] на основі ефективного опору графа:

$$\triangle R_G^l = \frac{R_G - R_{G+l}}{R_G}; \tag{1.35}$$

$$\triangle R_G^l = \frac{R_{G-l} - R_G}{R_G}, \tag{1.36}$$

де R_G – вихідний ефективний опір графа G;

R_{G+l} ефективний опір графа G після додавання лінії l;

R_{G-l} ефективний опір графа G після видалення лінії l.

Крім аналізу стійкості мережі у вигляді графу за допомогою відповідних метрик, важливо досліджувати стратегії мережевих атак, вибір методів захисту для зменшення наслідків атак у мережах. Для атаки на мережу є два можливі підходи: видалення вершин і видалення ребер, які є критичними, важливими для функціональності мережі. На основі двох метрик (degree centrality, betweenness centrality) існує чотири стратегії атаки [63, 64]:

– initial degree removal (ID): вибір вершин для атаки здійснюється на основі її степеню (degree), видалення вершин починається із найвищим степенем одна за одною, що призводить до якнайшвидшого зменшення загальної кількості ребер у мережі [64];

– initial betweenness removal (IB): атака спрямована на вершини з високим значенням betweenness centrality, видалення вершин починається із найвищим значенням одна за одною, що призводить до руйнування якомога більшої кількості шляхів [64];

– recalculated degree removal (RD): стратегія атаки аналогічна до initial degree removal, але додатково обчислюється degree distribution вершин після видалення кожної вершини, що дає можливість повторно оцінити важливість цілей [64], але збільшується обчислювальна складність [63];

– recalculated betweenness removal (RB): аналогічна стратегія до initial betweenness removal, крім того обчислюється node centrality вершин на кожному кроці, що дає можливість повторно оцінити важливість цілей [64], але збільшується обчислювальні витрати [63].

Стратегії (ID, IB, RD, RB) можуть бути застосовані для атаки – видалення ребер із мережі, де степінь ребра (edge degree) k_e можна визначити формулами (1.37 – 1.40) за допомогою степенів двох вершин, які воно з'єднує (ребро e з'єднує дві вершини v і w зі відповідними степенями вершин k_v і k_w) [64]:

$$k_e = k_v k_w; \tag{1.37}$$

$$k_e = k_v + k_w; \tag{1.38}$$

$$k_e = \min(k_v, k_w); \tag{1.39}$$

$$k_e = \max(k_v, k_w)\,. \tag{1.40}$$

Визначення степеню ребра за формулою (1.37) найкраще корелює з метрикою betweenness centrality, тому ефективно використовується в стратегіях атаки. Для виявлення збитків, які були завдані атаками, використовується обернена середня довжину найкоротшого шляху (average inverse geodesic length) та гігантська компонента (giant component) [64]. Атаки на мережу можуть спричиняти каскадні збої, тому важливо визначити стратегії захисту, враховуючи властивості графа та тип атаки чи збою. Техніки для покращення захисту мережі згруповані в категорії [63]:

а) додавання ребер (edge addition): спричиняє додаткові витрати на мережу, ніж перебудова ребер (edge rewiring), але майже завжди забезпечує кращий результат [63];

1) випадкове додавання (random addition): додати ребро, що з'єднує дві випадкові вершини;

2) переважне додавання (preferential addition): додати ребро, з'єднавши дві вершини з найнижчими степенями;

б) перебудова ребер (edge rewiring): популярний метод покращення надійності мережі, витрати нижчі, ніж при додаванні ребер (edge addition) [63];

1) перебудова випадкового ребра (random edge rewiring): видалити випадкове ребро та застосувати випадкове додавання ребра;

2) перебудова випадкового сусіда (random neighbor rewiring): вибрати випадкову вершину, випадкового сусіда обраної вершини, видалити ребро, яке їх з'єднує та додати випадкове ребро (random addition);

3) переважна перебудова (preferential rewiring): від'єднати випадкове ребро від вершини з найвищим степенем та з'єднати це ребро до випадкової вершини;

4) переважна перебудова випадкового ребра (preferential random edge rewiring): від'єднати випадкове ребро від вершини з вищим степенем та з'єднати ребро до випадкової вершини;

в) визначення важливих вершин і ребер у мережі для моніторингу підозрілої активності (node monitoring): деякі підходи, які можуть використовуватися для атаки на мережу, можливо використовувати для її захисту, визначення критичних місць, зменшення вразливості, збільшення стійкості та контролю критичних вершин і ребер [63].

Отже, дослідження використання метрик стійкості, стратегій атаки, захисту для збільшення стійкості критичних інфраструктур (електромереж), які представлені у вигляді графів, до різних типів подій (каскадних збоїв) є актуальним завданням [41].

1.5.2. Аналіз застосовування мереж Баєса для моделювання роботи критичної інфраструктури

Для оцінки ризиків та моделювання каскадних, доміно ефектів використовуються мережі Баєса. Дослідження в підходах на основі мережі Баєса та розробка програмного забезпечення для моделювання та аналізу надає можливість формувати більш точні оцінки ризиків. Мережа Баєса (Bayesian Network, BN) – це імовірнісна графова модель, яка є одним з інструментів для аналізу, отримання знань з даних та може використовуватися для моделювання складних недетермінованих систем. Формально мережа Баєса визначається як орієнтований ациклічний граф $G(V, E)$, де V – множина вершин або величин, а $E \subseteq V \times V$ – множина орієнтованих ребер. Кожна вершина $X_i \in V$ відповідає випадковій величині X_i, а кожне ребро $(X_i, X_j) \in E$ представляє прямий вплив X_i на X_j. Тобто X_i є батьком (parent) X_j, а X_j є дочірнім (child) елементом X_i. Кожна вершина X_i має відповідний розподіл ймовірностей $P(X_i|pa(X_i))$, де $pa(X_i)$ – множина батьків X_i в G [41]. Спільний розподіл ймовірностей (joint probability

distribution) по всім величинам $P(X)$ в мережі визначається за формулою (1.41) [65]:

$$P(X_1, \dots, X_n) = \prod_{i=1}^{n} P(X_i | pa(X_i)). \quad (1.41)$$

Імовірність одночасного виникнення незалежних подій A, B визначається за формулою (1.42):

$$P(A, B) = P(A)P(B). \quad (1.42)$$

Якщо події A, B залежні, то виникнення однієї із них надає певну інформацію про можливість появи іншої, де $P(B|A)$ – імовірність виникнення події B при умові виникненні події A визначається за формулою (1.43):

$$P(A, B) = P(A)P(B|A). \quad (1.43)$$

Концепція мереж Баєса побудована на теоремі Баєса, за допомогою якої можна сформувати висновок і прийняти рішення, визначається за формулою (1.44):

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}. \quad (1.44)$$

У мережі Баєса існують три типові класи причинно-наслідкових зв'язків, які зображені на рисунку 1.16 на прикладі із трьома змінними A, B, C:

– причинно-наслідковий ланцюг (каскад) (causal chain (cascade)): вершина A з'єднана ребром з вершиною B, а B з'єднана з C;

– спільний батьківський вузол (common parent (common cause)): дві вершини A та C мають спільну батьківську вершину B;

– спільний ефект (v-структура): вершина C має двох батьків A і B, форма структури схожа на літеру V.

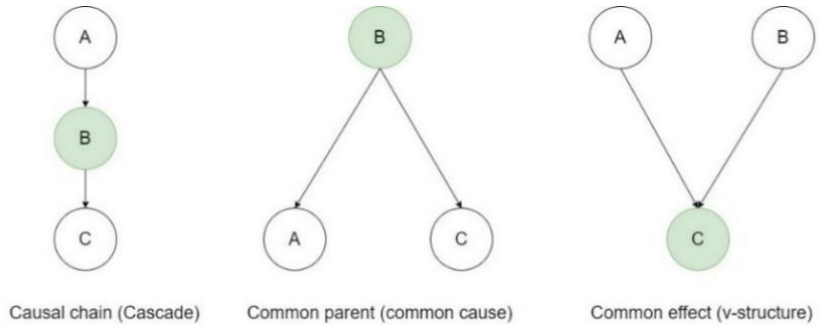

Рисунок 1.16 – Можливі класи причинно-наслідкових зв’язків

Існують різновиди мережі Баєса, які використовуються у випадках, коли потрібна розширена структура [66]:

– динамічні мережі Баєса (Dynamic Bayesian Networks): використовуються для моделювання динамічних і часових процесів, складається з попередньої мережі Баєса, яка визначає початкові умови, та перехідної мережі Баєса, яка визначає, як змінні змінюються час від часу [66];

– діаграми впливу (influence diagrams): мережі Баєса вказують ймовірність появи певної події, але не мають алгоритму дії при виникненні певної ситуації, тому мережа доповнюється двома типами вершин: прийняття рішень (decision) і корисності (utility) [66, 67];

– моделі причинної взаємодії (causal interaction models): Noisy-Or [68].

Побудова мережі Баєса складається з двох основних фаз: визначення графової структури; визначення параметрів Р. Граф і параметри мережі можуть бути визначені на основі експертних знань, які можуть бути отримані з даних або їх поєднанням. Аналіз для виведення висновків в мережах Баєса є NP-складною проблемою частково тому, що простір розв’язків графів зростає надекспоненціально (super-exponentially) з кількістю величин [69]. Мережі Баєса є візуально-інтуїтивно зрозумілими, що робить їх придатними для аналізу та модифікації експертами. На запитання, яке може бути сформовано в імовірнісній формі, можна відповісти правильно та з певним рівнем впевненості: «З огляду на

деякі наслідки, які були причини?» «Як щось контролювати з урахуванням поточних показань?» «У випадку причинно-наслідкових байєсівських мереж, що станеться, якщо в систему втрутитися?» [66]. Вивчення мережі Баєса складається з визначення структури та параметрів мережі з даних. Алгоритми вивчення структури мережі поділяються на два основні класи: методи на основі обмежень (constraint-based methods), які вилучають і орієнтують ребра на основі серії тестів умовної незалежності (conditional independence); методи на основі оцінок (score-based methods), спрямовані на вивчення структури мережі шляхом пошуку структури, яка максимізує функцію оцінки, але існують гібридні алгоритми, які поєднують підходи на основі оцінок і обмежень [69]. Існують різні програмні засоби (бібілотеки) для вивчення мереж Баєса, які мають різні характеристики, набір інструментів та обмеження у використанні [70]: BANJO, BayesiaLab, Bayesian Network Classifiers in Weka, Bayes Server, B-Course, BKD/BD, blip, bnlearn, BNJ, bnstruct, BNT, DEAL, ELVIRA, Free BN, GeNie and Smile, GOBNILP, Hugin Expert, LibB, Libpgm, OpenMarkov, PNL and PyPNL, Pomegranate, ProBT, Tetrad, UnBBayes, WinMine/MSBN. Тому існує потреба для розширення існуючих інструментів, функціоналу, налаштування та використання. Для моделювання сценаріїв ефектів доміно застосовуються мережі Баєса, динамічні мережі Баєса, діаграми впливу [67]. Динамічні Баєсівські мережі можуть використовуватися для аналізу (діагностування та прогнозування) кіберстійкості енергетичних систем, визначення причинно-наслідкових та часових залежностей для виявлення аномалій для вчасного реагування. Використовуючи різні типи алгоритмів для формування висновків, за допомогою динамічних мереж Баєса можливо виконувати наступні завдання: моніторинг (засоби аналітики безпеки системи для раннього виявлення або в реальному часі ймовірності спроби атаки відповідно до спостережень, які були зібрані експертами чи аналітиками); прогнозування (допоміжний інструмент для аналітика/системи з можливістю визначити стратегії, які зловмисник, імовірно, використовуватиме, що допоможе визначити стратегії захисту); офлайн-діагностика (визначення імовірних причин

подій безпеки, використовуючи фільтрації (онлайн) і згладжування (офлайн)) [71].

Останнім часом кількість досліджень інфраструктури розумного міста в науковій літературі збільшилося. Для аналізу ризиків та моделювання загроз в інфраструктурі застосовуються різні методи, частина яких заснована на мережах Баєса та їхніх модифікаціях [72]. Процес аналізу поведінки комплексних взаємозалежних мереж потребує інформації про топологію та зв'язки між мережами, які бувають неявними і можуть виявляються після збою, що призводить до виникнення каскадних збоїв. На основі спостережень за каскадними збоями та інших даних можливо визначити структури мереж взаємозалежної критичної інфраструктури: одна послідовність каскадних збоїв може не містити достатньої інформації про мережу, яка досліджується, але за допомогою поєднання каскадних ланцюгів збоїв збільшується можливість реконструкції мережі, щоб зробити якісні висновки про її функціональність [73]. Іноді при моделюванні взаємозалежних мереж критичної інфраструктури для оцінки їхніх властивостей та характеристик є обмеження або відсутність вхідних даних, тому одним із рішень є моделювання синтетичних взаємозалежних мереж критичної інфраструктури [74]. Підходи реконструкції мережі на основі даних поділяються на три категорії: підхід на основі вкладення графів (graph embedding-based approach), підхід на основі оптимізації (optimization-based approach) та Баєсівський підхід (Bayesian approach). Баєсівські підходи реконструкції мережі поділяються на дві категорії: параметричні та непараметричні. Для реконструкції мережі за допомогою байєсівського підходу потрібно оцінити щільність розподілу ймовірностей (probability distribution density) топології мережі з урахуванням спостережень – множину початкових сценаріїв збоїв та наступних послідовностей каскадних збоїв [73]. Висновки формуються за допомогою алгоритму Markov Chain Monte Carlo залежно від типу мережі з урахуванням її топологічних обмежень. Для підвищення обчислювальної ефективності підходу використовується три методи: Likelihood calculation, Graph validation, Tie-No-Tie sampler [73]. За допомогою фреймворку

[75] можна провести аналіз та оптимізацію стійкості критичної інфраструктурі, де стан продуктивності інфраструктури моделюється за допомогою ланцюга Маркова (Markov chain).

1.5.3. Аналіз застосовування мереж Петрі для моделювання роботи критичної інфраструктури

При моделюванні ефекту доміно та/або каскадних ефектів у комплексних системах використовуються різні методи, проте деякі з них мають обмеження для використання в певних ситуаціях. Мережі Петрі (Petri nets) – перспективний інструмент графічного та математичного моделювання для опису та дослідження систем обробки інформації, які характеризуються як одночасні (concurrent), асинхронні (asynchronous), розподілені (distributed), паралельні (parallel), недетерміновані (nondeterministic) та/або стохастичні (stochastic) [76]. Мережі Петрі використовуються в інженерії програмного забезпечення для вирішення різних задач [77], наприклад, можливо зробити висновки про структуру та динамічну поведінку системи, яка моделюється. Мережа Петрі – це орієнтований, зважений дводольний граф, який складається з двох типів вершин, які називаються позиціями та переходами, де дуги йдуть або від місця до переходу, або від переходу до місця, визначається за формулою (1.45) [76]:

$$PN = (P, T, F, W, M_0), \tag{1.45}$$

де $P = \{p_1, p_2, p_3, \dots, p_m\}$ – скінчена множина позицій (n – кількість позицій), які графічно зображуються колом;

$T = \{t_1, t_2, t_3, \dots, t_n\}$ – скінчена множина переходів (n – кількість переходів), графічно зображуються прямокутною смугою;

$F \subseteq (P \times T) \cup (T \vee P)$ – скінченна множина дуг від позицій до переходів або переходів до позицій;

$W: F \to \{1,2,3,\dots\}$ – вагова функція (цілі додатні числа), де k–зважена дуга може інтерпретуватися як множина з k паралельних дуг;

$M_0: P \to \{1,2,3,\dots\}$ – початкове маркування моделі мережі Петрі, що представляє кількість маркерів (ціле невід'ємне число) у кожній позиції моделі, графічно зображуються чорними крапками на позиції [41].

У моделюванні використовуються поняття умов (позиції) і подій (переходи), де перехід (подія) має певну кількість вхідних і вихідних позицій, що представляють відповідно передумови (які повинні бути виконані для спрацювання переходу) та постумови (результат переходу) події, наявність k маркерів в позиції може означати кількість доступних ресурсів, стан або маркування в мережах Петрі змінюється відповідно до правила переходу [76]. Приклад мережі Петрі зображено на рисунку 1.17.

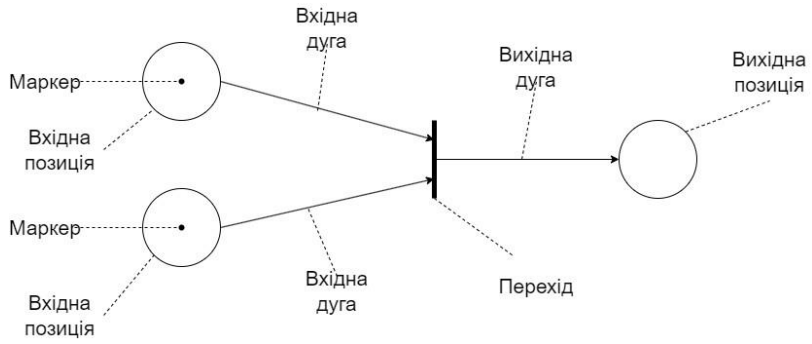

Рисунок 1.17 – Приклад мережі Петрі [41]

Мережі Петрі мають певні обмеження у використанні в деяких випадках. Щоб подолати обмеження та розширити можливості моделювання були розроблені різні модифікації, які пропонують унікальний функціонал, тому вибір типу мережі Петрі залежить від вимог і характеристик системи, наприклад:

– стохастична мережа Петрі (Stochastic Petri Net): розширена версія мережі Петрі, в якій визначається імовірнісна поведінка переходів, що дозволяє моделювати недетерміновані та невизначені системи [78];

– узагальнена стохастична мережа Петрі (Generalized Stochastic Petri Net): розширена версія стохастичної мережі Петрі, яка має дві нові функції: активація

переходу без затримки та інгібіторні дуги, які використовуються для вимкнення переходу, коли маркер присутній у вхідних позиціях [79];

– часова мережа Петрі (Timed Petri Net): розширена версія мережі Петрі, де використовується концепція часових затримок переходів, тому можливо представляти залежну від часу поведінку [78];

– кольорова мережа Петрі (Colored Petri Net): розширена версія мережі Петрі, яка містить маркери з певними кольорами або атрибутами, які можуть представляти різні властивості, тому може бути використана для детального моделювання та аналізу складних систем [78];

– нечітка мережа Петрі (Fuzzy Petri Net): поєднує концепції мережі Петрі з нечіткою логікою, забезпечує структуру моделювання для обробки невизначеності [78];

– гібридна мережа Петрі (Hybrid Petri Net): розширена версія мережі Петрі, яка поєднує моделювання дискретних і неперервних подій [78];

Існує множина інструментів для побудови, редагування, моделювання, оцінки та візуалізації систем за допомогою мереж Петрі з різним функціоналом та обмеженнями [54]: AlPiNA, CoopnBuilder, GreatSPN, LoLA, PEP, SNOOPY, MARCIE, CHARLIE, JSARP, MIST, PETRUCHIO, PNEditor, Yasper, PAPETRI, Xpetri, PROD, ARP, JPetriNet, Petri .NET Simulator, QPME.

Одним із способів моделювання ймовірності аварії (ефект доміно) та патерна її поширення у дослідженні є узагальнена стохастична модель мережі Петрі (Generalised Stochastic Petri-net model) – DOMINO-GSPN, за допомогою якої можливо графічно представити взаємодію між компонентами, підготувати заходи контролю, прийняття рішень, щоб запобігти майбутнім аваріям [79]. Крім того, модель каскадного поширення вразливості інтегрованих енергетичних систем у транспортній галузі, правила поширення каскаду вразливостей моделюється за допомогою мережі Петрі [80]. Також мережі Петрі та модифікації (розширені мережі Петрі) використовуються в різних сферах розумних електромереж (Smart Grid): управління (management), надійність (reliability), мережа (networking) та безпека (security), можуть бути використані для

моделювання та імітації різних процесів (наприклад, аналіз надійності мережі, діагностика проблем, розподіл ресурсів, планування споживання енергії, забезпечення безперервного електропостачання, запобігання можливим каскадним ефектам) [78]. Надійність і стійкість є важливими складовими в телекомунікаційних мережах для забезпечення роботи складних систем. Розширена мережа Петрі використовується для моделювання 5G і інших телекомунікаційних мереж: фізичної інфраструктури, віртуальної інфраструктури, мережевих сервісів, їх поведінки та залежностей для аналізу, оцінки продуктивності та стійкості [81].

1.5.4. Аналіз застосовування ланцюгів Маркова для моделювання роботи критичної інфраструктури

Ланцюги Маркова використовується для оцінки стохастичних процесів, можуть бути використані для аналізу надійності комплексних систем. На основі аналізу 37 blackout подій в енергосистемі визначено, що знеструмлення енергосистеми (blackout) зазвичай складаються з п’яти фаз: передумова (precondition), початкова подія (initiating event), каскадні події (cascading events), кінцевий стан (final state), відновлення (restoration) та сформовано припущення, що превентивні дії повинні бути вжиті до того, як відбулися тригерні події, бо після початку високошвидкісного каскаду ситуація може стати неконтрольованою, а blackout може статися протягом дуже короткого часу [82]. Аналіз, прогнозування та запобігання каскадних послідовностей / ланцюгів збоїв в електромереж є важливим завданням при розробці стратегій запобігання каскадних аварій в енергосистемах (blackouts). Для вирішення поставленої проблеми використовується модель ланцюга Маркова з урахуванням невизначеностей і стохастичних факторів [83]. Ланцюг Маркова визначається за формулою (1.46) як послідовність випадкових величин, що задовольняє Марковську властивість [41], приклад зображений на рисунку 1.18:

$$P(X_{t+1} = x_{t+1} \mid X_t = x_t, \dots, X_1 = x_1) = P(X_{t+1} = x_{t+1} \mid X_t = x_t), \qquad (1.46)$$

де X_{t+1} – стан системи в наступний момент часу;

X_t – стан системи у поточний момент часу.

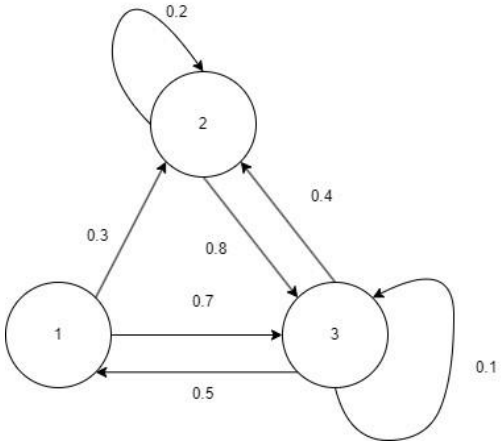

Рисунок 1.18 – Приклад ланцюга Маркова [41]

Для спрощення моделювання та аналізу каскадних ефектів можуть використовуватися графи взаємодії (interaction graphs), за допомогою яких зображають базові взаємодії та впливи між компонентами під час каскадних ефектів. Для моделювання та аналізу розвитку розміру каскадів використовується модель ланцюга Маркова з використанням структур спільноти (community structures) в графі взаємодії електромережі [84]. Аналіз та прогнозування розмірів каскаду надає можливість оцінювати стан системи для прийняття рішень щодо пом’якшення каскадних ефектів та підвищення стійкості системи. Для прогнозування k наступних ймовірних подій у потоках (streams) використовується event precedence model, яка будує поглинаючий ланцюг Маркова першого порядку, механізм причинного висновку в реальному часі вивчає причинно-наслідкові зв’язки [85]. Для оцінки ризику багаточасових каскадних відключень (multi-timescale cascading outages) використовується методологія на основі пошуку по дереву Маркова (Markovian tree search), яка дозволяє уникнути повторного моделювання каскадних шляхів, заощаджуючи обчислювальні ресурси та підвищуючи ефективність [86]. Управління ризиками

каскадних відключень можливо за допомогою підходу на основі градієнта ризику та пошуку по дереву Маркова [87].

1.5.5. Аналіз програмного забезпечення для оцінки ризиків та наслідків у роботі критичній інфраструктурі при виникненні збоїв

Існує множина методологій, програмних засобів, які застосовуються для захисту критичної інфраструктури та класифіковані відповідно до характеристик [88]: тип інфраструктури, техніка моделювання (multi-agent systems, system dynamics, rating matrices, relational data-bases and the network theory. Методи моделювання можуть поєднуватися з обчислювальними методами та техніками: continuous time-step simulation, discrete time-step simulation, Monte Carlo simulation, decision trees, geographic information systems, risk management techniques, event monitoring or real time record), завершеність і доступність, етапи управління ризиками [88].

Для аналізу ефектів доміно використовуються методології, програмні засоби та інструменти [89]: MAXCRED (оцінка та запобігання аварій на підприємствах хімічної промисловості, використовуючи метод дерева подій і аналіз дерева збоїв), DOMIFFECT (програмне забезпечення з об'єктно-орієнтованою архітектурою, розроблене на C++ для аналізу ефекту доміно), ARIPAR (імовірнісна методологія для оцінки ризиків), ATLANTIDE (програмне забезпечення для оцінки наслідків аварійних подій, використовуючи дерева подій для оцінки можливих сценаріїв), DOMINOXL (визначення можливих ефектів доміно), GeOsiris (моделювання аварій з ефектом доміно), MiniFFECT (визначення положення та розташування хімічних об'єктів для мінімізації наслідків каскадних подій за допомогою підходів нелінійного програмування), DomPrevPlanning (програмне забезпечення може виконувати аналіз ризиків ефекту доміно для запобігання їх виникнення) [89].

Для аналізу каскадних збоїв в електромережах використовується MATCASC – інструмент на основі MATLAB, який має функціонал для оцінки

потоку електроенергії в лінії, для оцінки пропускної здатності лінії, а також для застосування варіантів видалення лінії (випадкове або на основі атаки), моделювання каскадних відключень через перевантаження лінії, визначення збитків через каскадні збої [9].

Для оцінки вразливості та міцності графів може використовуватися Tiger – відкритий пакет інструментів Python, який містить 22 показники надійності графіка з оригінальними та швидкими наближеними версіями (коли це можливо); 17 механізмів збоїв та атак; 15 евристичних та оптимізаційних методів захисту; 4 інструменти моделювання [90].

Таблиця 1.2 – Порівняння програмного забезпечення для аналізу каскадних ефектів та ефекту доміно

Програмний інструмент	**Переваги**	**Недоліки**
ARIPAR	Інтеграція з геоінформаційними системами для оцінки ризиків	Складність налаштування та необхідність експертних знань. Потреба в значному обсязі даних
MATCASC	Гнучкість в модифікації алгоритмів для дослідження каскадних збоїв	Необхідність навичок програмування / моделювання для налаштування середовища
Tiger	Моделювання надійності та каскадних збоїв у складних систем	Складність побудови, валідації великих графів, які мають складні зв’язки

Більшість наявних програмних засобів для моделювання та аналізу каскадних / доміно ефектів використовуються для конкретної (однієї) галузі критичної інфраструктури з врахуванням особливостей процесів та потребують додаткового налаштування, впровадження на об’єктах. Технології, алгоритми, підходи до розробки програмного забезпечення стрімко розвиваються, що робить

можливим покращення наявних рішень – створення програмного забезпечення для аналізу каскадних ефектів з таким функціоналом: збільшена точність результатів, зменшення часу на виконання обчислень, гнучкий функціонал налаштування, розширення для інтеграції та застосування в різних секторах критичної інфраструктури [41].

1.5.6. Аналіз застосування методів машинного навчання в роботі критичної інфраструктури

В останні роки машинне навчання (machine learning) та глибоке навчання (deep learning) стрімко розвиваються, з'являються нові підходи, методи, які можуть бути використані для аналізу каскадних збоїв в енергосистемах [13]. Одним із підходів для прогнозування каскадних збоїв є використання графових нейронних мереж (Graph Neural Networks) [91], які можуть використовуватися для онлайн-прогнозування каскадних збоїв в електромережах [92]. Для підвищення надійності енергосистеми виконується пошук каскадних збоїв в онлайн-режимі: використовується фреймворк на основі згорткової графової нейронної мережі (graph convolutional network) [93]. Крім того, для вивчення каскадних збоїв може використовуватися deep convolutional generative adversarial network [94]. Графові нейронні мережі – перспективний підхід, що демонструє хороші результати в дослідженнях каскадних збоїв в електромережах. Для навчання моделі нейронної мережі потрібно сформувати набір даних, який міститиме інформацію про роботу компонентів електромережі при виникненні збоїв. Дані про роботу критичної інфраструктури є обмеженими або недоступними для аналізу у зв'язку з підвищеним рівнем захисту роботи об'єктів, тому виникає потреба у створенні синтетичних (штучних) даних роботи системи для її аналізу. Синтетичні дані – штучно створені дані, які наближені до реальних сценаріїв роботи системи (імітують закономірності та характеристики реальних даних) та створені за допомогою комп'ютерних алгоритмів, симуляції або статистичних моделей. Моделі машинного навчання, глибоко навчання

потребують великих обсягів даних для їхнього тренування та покращення ефективності, тому набори даних створюються або доповнюються за допомогою штучних даних. Однією з переваг синтетичних даних є можливість створювати різноманітні набори даних: генерувати у великих обсягах, з різними характеристиками, з різноманітними сценаріями. Для генерації синтетичних даних використовуються підходи/методи:

– симуляція даних з використанням середовищ та процесів, які є схожими до реальних;

– генерація даних на основі попередньо визначених правил, обмежень, алгоритмів, що імітують певні патерни чи розподіли;

– додавання шумів (контрольовану випадковість) в наявні дані для створення нових;

– процедурна генерація з використанням алгоритмів, які автоматично створюють середовища або дані;

– генеративна змагальна мережа: генератор створює синтетичні дані, дискримінатор оцінює їх (порівнює їх з реальними даними);

– варіаційний автокодувальник: кодер стискує вхідні дані в прихований простір, декодер створює нові, синтетичні дані.

1.6. Постановка наукового завдання та часткових завдань досліджень

Захист критичної інфраструктури від каскадних збоїв є важливим та комплексним завданням. Збій у роботі компонентів критичної інфраструктури може призвести до виникнення ланцюгу подій – каскадного ефекту (ефекту доміно), що може вивести з ладу частину об’єктів системи чи всю систему загалом. Тому важливо аналізувати роботу об’єктів критичної інфраструктури, стійкість до впливу подій, щоб прийняти превентивні дії чи сформувати план дій у подібних сценаріях. Електромережі – складні системи, які є важливою частиною критичної інфраструктури, забезпечують електроенергією побутових споживачів, промислові об’єкти. Крім того, вони є вразливими до збоїв та

каскадних ефектів, що призводять до повного або часткового знеструмлення мережі. У роботі електромереж використовуються інструменти моніторингу, які збирають дані про роботу компонентів для аналізу та прийняття рішень.

Було проведено аналіз методів моделювання сценаріїв аналітичної діяльності, методів виявлення каскадних ефектів в електромережах та виявлено такі протиріччя:

– аналітичні моделі потребують консистентних даних, але відсутні формалізовані моделі перевірки даних;

– нейронні мережі потребують для навчання великих обсягів даних, але існує обмежена кількість доступних даних про роботу електромереж, каскадних ефектів та інструментів для створення сценаріїв;

– методи машинного навчання потенційно можуть виявляти каскадні ефекти в електромережі, але відсутній підхід порівняння сценаріїв;

– програмні засоби аналізу каскадних ефектів побудовані для обробки підготовлених наборів даних, але мають обмежений функціонал для обробки даних у режимі реального часу.

На основі визначених протиріч було сформовано наукове завдання: розробка науково-методичного апарату та програмного забезпечення семантичного моделювання розвитку каскадних ефектів у критичних інфраструктурах та їхнього порівняння для визначення альтернативних сценаріїв на основі представлення з графових нейронних мереж.

Метою дисертаційної роботи є підвищення ефективності порівняння сценаріїв каскадних ефектів в електромережах та визначення подібних станів систем на основі графових нейронних мереж із врахуванням семантики ознак та зв’язків між об’єктами.

Об’єктом дослідження є процеси аналізу вимог, розроблення, впровадження і супроводження програмного забезпечення виникнення каскадних ефектів та їхнє поширення в критичних інфраструктурах.

Предметом дослідження є методи та програмне забезпечення семантичного моделювання взаємозв’язків у критичних інфраструктурах та побудови графових

нейронних мереж для створення та порівняння представлень про сценарії каскадних ефектів.

Для досягнення мети були сформовані часткові наукові завдання:

1. Провести аналіз наявних методів та програмного забезпечення моделювання роботи електромереж та каскадних ефектів.

2. Розробити метод семантичного моделювання сценаріїв каскадних збоїв в електромережах для формалізації даних.

3. Розробити метод створення та порівняння представлень сценаріїв роботи електромережі для визначення подібності сценаріїв.

4. Удосконалити моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж в сценаріях.

5. Розробити архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, та моделями нейронними мереж для визначення подібних станів електромереж у сценаріях.

6. Провести обчислювальні експерименти створення та порівняння представлень сценаріїв каскадних ефектів в електромережах.

Розв’язання визначеного наукового завдання можливо шляхом застосування розроблених методів, моделей та програмного забезпечення, що визначило структуру та зміст дисертаційної роботи.

1.7. Висновки до розділу 1

Проведено дослідження розвитку та впливу каскадних ефектів на роботу критичної інфраструктури, зокрема електромереж. В результаті було визначено, що каскадний ефект розвивається не лінійно та стрімко. Нелінійність процесу ускладнює прогнозування його розвитку. Проведено аналіз методів та програмного забезпечення моделювання роботи критичної інфраструктури в різних сценаріях, визначено їхні переваги та обмеження. Структура критичної

інфраструктури визначається за допомогою графу, що допомагає візуально представити мережу, взаємозв’язки між об’єктами та параметри компонентів.

Під час аналізу стійкості та вразливості в електромережах використовуються топологічні та гібридні підходи для дослідження властивостей мережі. Велика частина метрик графів надає статичну оцінку стійкості системи, не відображає динамічні зміни компонентів системи при виникненні збоїв, а результати аналізу залежать від якості початкових даних сценаріїв збоїв. Мережі Баєса використовуються для дослідження стійкості критичної інфраструктури, аналізу каскадних збоїв, моделювання динамічних процесів, але потребують значних обчислювальних ресурсів для великих та складних систем зі значною кількістю змінних і параметрів для обчислень та висновків. Мережі Петрі використовуються для моделювання та аналізу поведінки системи в сценаріях, але побудова та підтримка великих, складних моделей потребує спеціалізованих знань для перевірки коректності роботи моделі та її інтерпретації. Ланцюги Маркова використовуються для аналізу каскадних збоїв, але при моделюванні великої кількості комплексних взаємозалежностей відбувається спрощення сценарію для зменшення складності обчислення, що знижує точність аналізу. Значна частина наявних програмних засобів для аналізу каскадних ефектів (ефектів доміно) є складною в налаштуванні та має обмеження для обробки та аналізу даних в режимі реального часу.

На основі проведеного аналізу методів і програмного забезпечення моделювання роботи критичної інфраструктури сформульовано актуальність наукового завдання щодо розробки науково-методичного апарату та програмного забезпечення семантичного моделювання розвитку каскадних ефектів у критичних інфраструктурах та їхнього порівняння для визначення альтернативних сценаріїв на основі представлення з графових нейронних мереж.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ СЕМАНТИЧНОГО МОДЕЛЮВАННЯ СЦЕНАРІЇВ КАСКАДНИХ ЗБОЇВ В ЕЛЕКТРОМЕРЕЖАХ

Електроенергетична система відіграє важливе значення в сучасному світі: забезпечує виробництво, передачу та розподіл електроенергії, що використовується в різних секторах економіки. Складність електромереж стрімко зростає, тому аналіз роботи компонентів та взаємозв'язків між ними є важливим аспектом для побудови підходів зменшення наслідків каскадних збоїв. Модель електромережі є важливим інструментом для розуміння складних взаємодій між об'єктами системи, аналізу їх роботи та прийняття обґрунтованих рішень. Оператори систем передачі та розподілу мають власну інформацію про мережу, але, зазвичай, ці дані не є загальнодоступними, мають обмеження у використанні, деталізації характеристик, які є необхідними для проведення академічних досліджень.

2.1. Визначення ознак та зв'язків між компонентами електромережі

Узагальнена електромережа зображена на рисунку 2.1 та складається з наступних елементів: джерела електроенергії, передача та розподіл, навантаження.

Рисунок 2.1 – Приклад електромережі

Більшість електромереж використовують змінний струм частотою 50 Гц або 60 Гц, який виробляється на великих електростанціях. Електростанції – це об’єкти, які виробляють електроенергію та перетворюють її у різні форми енергії (наприклад, теплової, механічної, ядерної енергії) в електричну енергію, розташовуються переважно подалі від населених пунктів. На це впливають різні фактори: наявність палива (транспортування електроенергії на великі відстані вигідніше, ніж транспортування палива), рельєф (гідроелектростанція розташовується у відповідному місці річки або вітрова електростанція розташовується на березі, щоб отримати додаткову енергію від вітру). Основні типи електростанцій [95]:

– теплові електростанції: теплова енергія використовується для виробництва пари, яка приводить у дію турбіну, з’єднану з генератором, для виробництва електроенергії;

– атомні електростанції: використовуються ядерні реакції для отримання теплової енергії, яка використовується для роботи парових турбін;

– гідроелектростанції: використовується енергії води для обертання турбін;

– відновлювальна енергетика: сонячна електростанція, вітрова електростанція. Відновлювані джерела енергії є стохастичними, що ускладнює контроль над електричною мережею.

Електроенергія передається під високою напругою змінного струму на великі відстані (резистивні втрати менші) від електростанцій до підстанцій через високовольтні мережі електропередачі (110–380 кВ). Підстанції знижують напругу до 10–20 кВ та з’єднані з розподільною мережею, де розподільні трансформатори знижують напругу до нижчого рівня <1 кВ, яка передається до кінцевих споживачів (домашнє та комерційне використання).

Мережа передачі електроенергії ефективно транспортує електроенергію на великі відстані від електростанцій до систем розподілу, мінімізуючи втрати та зберігаючи її якість. Електроенергія виробляється на електростанціях при відносно низькій напрузі приблизно від 2,3 кВ до 30 кВ, що залежить від розміру об’єкта. Для транспортування електроенергії на великі відстані напруга

підвищується трансформатором електростанції до вищої напруги (від 115 кВ до 765 кВ змінного струму), щоб зменшити втрати енергії. Вона переміщується лініями електропередачі, які часто проходять через високі опори за межами населених пунктів до електричних підстанцій та утворюють мережу передачі електроенергії [96]. Якщо загальний опір лінії електропередачі від електростанції до населеного пункту дорівнює R, а необхідна потужність $P = I \times V$, тому чим вища напруга в лінії електропередачі, тим менший струм. Оскільки втрата потужності лінії електропередачі визначається як $P_{loss} = I^2 \times R = \frac{P^2 R}{V^2}$, а при сталих значення P та R втрати в лінії електропередачі зменшуються при збільшенні напруги V. Отже, найменше значення величини струму, яку можливо використати для транспортування електроенергії, призводить до найменших втрат потужності. В Україні для передачі та розподілу електричної енергії використовують класи змінної напруги [97]:

– низька напруга: 127/220 В, 230/400 В;

– середня напруга: 6 кВ, 10 кВ, 20 кВ, 35 кВ;

– висока напруга: 110 кВ, 150 кВ;

– надвисока напруга: 220 кВ, 330 кВ, 400 кВ, 500 кВ, 750 кВ.

При транспортуванні електроенергії використовують первинні і вторинні системи передачі, які визначаються на основі довжини лінії. В первинній системі передачі використовується висока напруга для великих відстаней, щоб транспортувати електроенергію до певного центрального регіону. На підстанціях за допомогою трансформаторів напруга знижується, оскільки відстань до населених пунктів суттєво зменшилася. У вторинній системі передачі використовується нижча напруга для менших відстаней, щоб транспортувати електроенергію до кожного міста [98].

Розподільні підстанції розташовані поблизу населених пунктах або в них та з’єднані з мережами передачі. На підстанціях за допомогою трансформаторів напруга знижується для розподілу та постачання місцевими лініями електропередачі з нижчою напругою до кінцевих споживачів з відповідною напругою: промислові, житлові об’єкти тощо. Висока напруга з лінії

електропередачі знижується за допомогою трансформаторів до напруги первинного рівня розподілу (11 кВ, але може коливатися від 2,4 кВ до 33 кВ залежно від регіону чи споживача) [99]. Вторинна розподільна система – лінії низької напруги, які проходитимуть в певних житлових зонах між будинками, щоб забезпечувати енергією побутових користувачів (однофазна напруга 50 Гц 220 або 230 В або 3-фазна напруга 400 В для великих побутових об'єктів, комерційних будівель, невеликих підприємств).

Мережі розподілу електроенергії відіграють важливу роль у передачі електроенергії кінцевим споживачам та складаються з фідерів, які використовуються для передачі електроенергії від підстанції до розподільних трансформаторів, які знижують напругу для передачі до будинків, підприємств тощо. Вибір типу системи розподілу електроенергії залежить від різних факторів: масштаб мережі, надійність, ефективність та економічні витрати. Мережі розподілу електроенергії класифікуються в залежності від схем підключення фідерів або топології, основними та поширеними є [100]:

1. Радіальна розподільна система (Radial distribution system). Цей тип розподільної системи складається з єдиного фідера, що прокладений від підстанції та подає електроенергію кінцевим споживачам, тобто потік потужності відбувається в одному напрямку від джерела до навантажень, зазвичай використовується в невеликих системах розподілу електроенергії [100]. Переваги: простота у проєктуванні та реалізації, низька початкова вартість побудови мережі; простота експлуатації та технічного обслуговування; висока масштабованість архітектури. Недоліки: низька надійність постачання електроенергії (несправність у фідері призведе до відключення електроенергії для пов'язаних споживачів, потенційні відключення для великих груп споживачів до завершення ремонтних робіт); підтримка рівномірного розподілу навантаження вздовж фідера може бути проблематичним, що призводить до перепадів напруги; низька варіативність налаштувань підключень різних груп споживачів.

2. Паралельний розподіл фідерів (Parallel feeders distribution). Надійність та ефективність радіальної системи може бути підвищена за допомогою використання паралельних фідерів, що рівномірно розподіляють навантаження, знижують ризик перевантаження і перепадів напруги, але початкова вартість цієї системи зростає. Якщо один із фідерів вийде з експлуатації або станеться збій, то навантаження може бути направлено на інші фідери, використовуючи перемикачі, що забезпечує безперебійне електропостачання. Цей тип системи використовується у великих системах розподілу електроенергії, наприклад: житлові райони, комерційні будівлі, промислові комплекси, зменшуючи ризик відключень і збоїв [100].

3. Кільцева магістральна розподільна система (Ring main distribution system). Рівень надійності кільцевої системи подібний до рівня паралельних фідерів. Кожен розподільний трансформатор живиться від декількох фідерів, але різними шляхами, що збільшує надійність мережі (відновлення електропостачання після збою відбувається швидше в кільцевих мережах, ніж у радіальних), але збільшується складність в обслуговувані мережі. Зазвичай використовуються в міських і промислових середовищах [100].

4. Взаємопов’язана система розподілу (Interconnected distribution) – тип мережі, де розподільна фідерна мережа живиться від двох або більше підстанцій або електростанцій, що забезпечує резервні шляхи живлення (безперебійне живлення, якщо одне із джерел вийде з ладу через збій або відключення, ізоляцію збоїв), використання додаткових джерел електроенергії для забезпечення підвищених потреб споживачів. Цей тип системи має високу надійність. Зазвичай він використовується для забезпечення функціонування важливих об’єктів. Складність обслуговування та експлуатації системи може бути зменшена за допомогою моніторингу й автоматизації процесів прийняття рішень та керування мережею [100].

Значення навантаження (споживання електроенергії) може коливатися протягом доби через певні чинники: сезонність, погодні умови, час доби. Тому прогнозування попиту електроенергії та забезпечення відповідної генерації є

важливими завданням операторів мереж для безперебійного постачання електроенергії. Баланс генерації та навантаження є важливим критерієм для забезпечення стабільності мережі: якщо генерація перевищує навантаження, то це може призвести до перенапруги, до пошкодження обладнання (виведення з експлуатації). Якщо навантаження перевищує генерацію, то це може призвести до відключень електроенергії та знеструмлення.

Розуміння взаємодії між компонентами мережі, балансування навантаження і виробництва електроенергії має вирішальне значення для ефективного управління електромережею. Оператори повинні постійно контролювати роботу електромережі, миттєво реагувати та проводити відповідні дії, щоб забезпечити надійне та ефективне електропостачання.

Побудова якісної моделі електромережі є важливою умовою для проведення ефективного дослідження. OpenStreetMap [101] – відкритий інструмент, який може бути використаний для вивчення топології електромережі. Проте в процесі побудови електромережі можуть виникнути проблеми, які зазначені в роботі [102]: обмеженість тегів (недостатня деталізація); деталізація схем розгалуження ліній електропередачі; неправильні, відсутні дані. Для моделювання та аналізу мережі електропередачі існують відкриті інструменти: PyPSA-Eur [103, 104, 105], PyPSA-Earth [106, 107], які мають обширний функціонал для аналізу електромереж великої множини країн.

Для збору характеристик електростанцій України використовується глобальна база даних електростанцій [108], статистика зображена на рисунку 2.2.

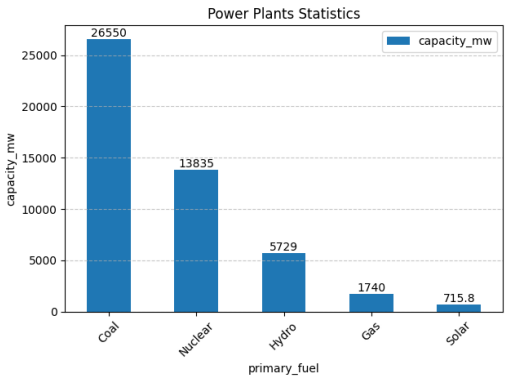

Рисунок 2.2 – Статистика електростанцій України

Електростанції, що працюють на вугіллі вироблять найбільшу кількість електроенергії – 26550,0 МВт, атомні електростанції – 13835,0 МВт, гідроелектростанції – 5729,0 МВт, електростанції на газу – 1740,0 МВт, сонячні електростанції – 715,8 МВт.

Для роботи з картографічним сервісом OpenStreetMap є різні інструменти:

– OSMnx [109, 110] – пакет, який використовується з Python для завантаження, моделювання, аналізу та візуалізації даних з OpenStreetMap;

– earth-osm [111] – інструмент, що надає зручний та простий інструмент для роботи з картою: можливість завантажувати, працювати з даними про об'єкти інфраструктури з OpenStreetMap (OSM), експортувати дані у форматах csv і geojson, підтримка багатопроцесорності, використовуючи Python API та CLI.

За допомогою бібліотек для роботи з OpenStreetMap можемо визначити, які природні об'єкти знаходяться в радіусі 1,5 км від електростанцій, інформація про них зазначена в довіднику [112].

Для побудови мережі передачі та розподілу електроенергії України на основі відкритих даних OpenStreetMap використовується інструмент PyPSA-Eur та складається з наступних етапів:

– зчитування даних OSM;

– очищення даних OSM;

– побудова мережі OSM;

– створення мережі PyPSA [113, 114];

– експорт даних в CSV-файли.

У результаті була сформована електромережа, яка складається з наступних компонентів: шини (buses), трансформатори, лінії електропередачі.

У результаті аналізу відкритих даних було визначено 9962 шини (підстанцій), які мають різний рівень напруги: 750,0 кВ – 16 одиниць; 500,0 кВ – 5 одиниць; 400,0 кВ – 5 одиниць; 330,0 кВ – 174 одиниці; 220,0 кВ – 92 одиниці; 154,0 кВ – 100 одиниць; 150,0 кВ – 367 одиниці; 110,0 кВ – 2201 одиниці; 35,0 кВ – 6997 одиниць; 10,0 кВ – 3 одиниці; 6,0 кВ – 2 одиниці, статистика зображена на рисунку 2.3.

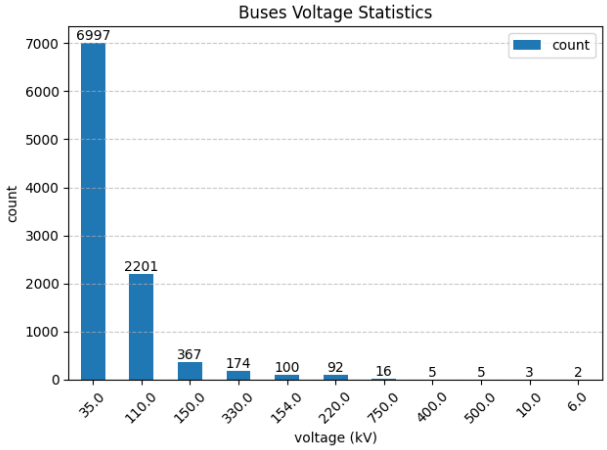

Рисунок 2.3 – Статистика шин (підстанцій) в електромережі України

Лінії електропередачі та розподілу з’єднані за допомогою трансформаторів з різним рівнем напруги (одиниця виміру кВ), які можуть бути класифіковані у 2 категорії: знижувальний трансформатор, підвищувальний трансформатор, статистика зображена на рисунку 2.4.

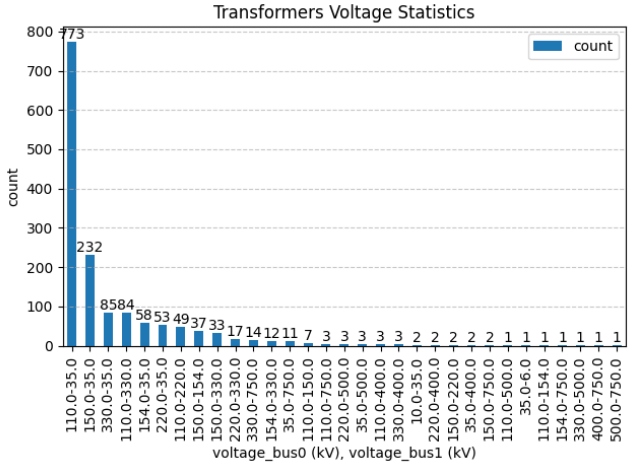

Рисунок 2.4 – Статистика трансформаторів в електромережі України

Лінії електропередачі з’єднують шини (підстанції), трансформатори для передачі електроенергії на великі відстані, використовуючи різні рівні напруги (одиниця виміру кВ): 330, 110, 35, 750, 220, 150, 500, 400, 154, 6, 10. Найбільша загальна довжина ліній електропередачі середньої напруги – 61913,94025 (км), високої напруги – 38039,47148 (км), надвисокої напруги – 20571,89504 (км), статистика зображена на рисунку 2.5.

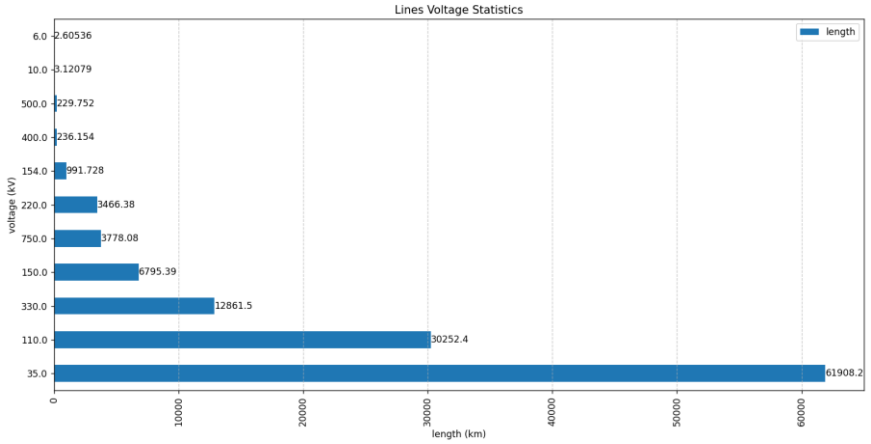

Рисунок 2.5 – Статистика ліній електропередачі в електромережі України

У результаті сформовано граф, де вершини – шини (9962 одиниці), а лінії електропередачі (11233 одиниці) та трансформатори (1497 одиниці) – ребра.

2.2. Вибір засобу моделювання роботи електромережі та сценаріїв розвитку каскадних ефектів

Під часу аналізу електромережі, яка складається з великої кількості об’єктів, виникає потреба у формалізованому опису компонентів, зв’язків між ними, щоб полегшити розуміння предметної області та перевірити коректність даних.

Уніфікована мова моделювання (Unified Modeling Language) – це мова моделювання загального призначення, яка є стандартом для дизайну складних систем, процесів та їхню візуалізацію з використанням множини графічних нотацій. UML діаграми поділяються на дві категорії: статичні (визначають структуру системи) та динамічні (визначають поведінку та взаємодію компонентів системи). Моделювання систем у вигляді діаграм дозволяє аналізувати, покращувати моделі перед реалізацією, що призводить до більш якісних результатів. В UML є 14 типів діаграм, які використовуються для різних цілей, мають різні набори конструкцій і позначень та поділяються на дві категорії [115]:

– 7 типів діаграм, які використовуються для представлення структурної інформації (Structural diagrams);

– 7 типів діаграм, які використовуються для поведінкового моделювання (Behavioral diagrams), які включають чотири типи діаграм, які описують взаємодію (Interaction diagram).

Більшість користувачів використовують лише обмежену множину типів та нотифікацій UML діаграм для розв'язання поставлених завдань: деякі діаграми використовуються частіше, інші використовуються іноді в певних випадках, статистика наведена у таблиці 2.1 [116].

Таблиця 2.1 – Використання діаграм UML [116]

UML діаграма	**Всі джерела**
Class	100%
Composite Structure	52%
Component	80%
Deployment	80%
Object	71%
Package	70%
Activity	98%
Sequence	97%
Communication	82%
Interaction Overview	39%
Timing	40%
Use Case	96%
State Machine	96%
Profile	11%

Кожна UML діаграма використовується з певною метою. Широковідомими діаграмами є:

– діаграма класів (class diagram) – тип структурної діаграми UML, яка описує структуру системи за допомогою класів, їхніх атрибутів, методів і зв'язків між класами;

– діаграма діяльності (activity diagram) – тип поведінкової діаграми UML, яка використовується для моделювання та візуалізації бізнес-процесів у системі, змін у системі з часом, переходи між різними станами з використанням подій або умов, розвитку сценаріїв, взаємодію різних компонентів системи;

– діаграма послідовності (sequence diagram) використовується для візуалізації послідовності процесів, взаємодії між об'єктами в часовій послідовності при моделюванні динамічної поведінки в системі, аналізу процесів, щоб виявити помилки та потенційні вузькі місця;

– діаграма варіантів використання (use case diagram) використовується для представлення взаємодії між акторами (користувачами) і системою, формування функціональних вимог системи (відображенні основних функцій системи без надлишкових деталей), корисна для розуміння роботи системи;

– діаграма станів (state machine diagram) використовується для моделювання динамічної системи, представлення стану системи або частини системи в певні моменти часу, як об'єкт реагує на події, як змінюється стан.

Отже, UML мають низку переваг: стандартизований спосіб представлення систем, використовується для зрозумілого опису, документації та візуального представлення при проєктуванні складних концепцій. Але UML має і недоліки: складність в проєктуванні та підтримці моделей складних систем; потенційна неоднозначність при суб'єктивній інтерпретації діаграми; ризик надмірної чи недостатньої деталізації моделі.

Systems Modeling Language (SysML) – це мова моделювання з відкритим кодом, яка розширює функціонал UML, усуває певні обмеження. Вона надає можливість моделювати широкий діапазон складних систем та використовується в системній інженерії. Для цього існують 9 типів діаграм, які мають свої особливості та використовуються для опису різних аспектів системи [117]:

– діаграма вимог (requirement diagram): визначає вимоги, зв’язки між ними та іншими елементами моделі; немає аналога UML;

– діаграма діяльності (activity diagram): використовуються для опису динамічної поведінки, функціональних можливостей системи, сценаріїв; activity diagram – аналогічна модель в UML;

– діаграма послідовності (sequence diagram): описує взаємодію між об’єктами (події, послідовність повідомлень) при виконанні сценарію; sequence diagram – аналогічна модель в UML;

– діаграма кінцевого автомата (state machine diagram): застосовуються для опису станів об’єкта або взаємодій, переходів між цими станами, що використовуються для моделювання поведінки компонента системи з часом (життєвого циклу об’єкта); state machine diagram – аналогічна модель в UML;

– діаграма прецедентів (use case diagram): представлення системи, описує взаємодію між системою та її середовищем, використовується для визначення системних вимог; use case diagram – аналогічна модель в UML;

– діаграма визначення блоків (block definition diagram): визначає компоненти системи, їхні властивості, обмеження, поведінку, інтерфейси та зв’язки; діаграма класів – аналогічна модель в UML;

– діаграма внутрішніх блоків (internal block diagram): описують внутрішню структуру блоку (частини, властивості, інтерфейси); composite structure – аналогічна модель в UML;

– параметрична діаграма (parametric diagram): застосовується для визначення обмежень системи, аналізу параметрів системи, вплив змін в дизайні на поведінку системи; немає аналога UML;

– діаграма пакетів (package diagram): використовується для відображення зв’язків між пакетами та їхніми елементами; package diagram – аналогічна модель в UML.

SysML має кілька переваг порівняно з UML: семантика SysML є більш гнучкою (можливість видалення непотрібних конструкцій UML), додає два нових типи діаграм (діаграми вимог і параметричні діаграми), зменшує

обмеження UML, але успадковує також багато труднощів і недоліків UML, крім того, для освоєння та ефективного застосування інструменту потрібен досвід.

Дерево сценаріїв (Scenario tree) – це орієнтований граф, де кожна вершина визначає можливий стан та має унікальну батьківську вершину, а кожне ребро представляє перехід між станами, приклад зображений на рисунку 2.6. У багатьох випадках використання дерева сценаріїв пов'язано з моделюванням стохастичного процесу. Множина вершин визначається за формулою (2.1) [118]:

$$V = \dot{\cup} N_t, \tag{2.1}$$

де N_t – вершини, що належать етапу t.

Послідовність спостережень $\xi^t = \{\xi_1, \dots, \xi_t\}$ – сценарій на етапі t. Scenario fan – окремий випадок дерева сценаріїв, де кожен сценарій $\xi^s = \{\xi_1^s, \dots, \xi_T^s\}$ має ймовірність $p_s = P(\xi_1^s)$ і не залежить від інших сценаріїв. Для побудови дерев сценаріїв використовуються різні підходи, які мають переваги та недоліки, оскільки ці методи залежать від поставленої задачі оптимізації. При виборі методу потрібно враховувати співвідношення між обчислювальними ресурсами й точністю рішень. Дерева сценаріїв можуть використовуватися у сфері електроенергетики, наприклад: дерева сценарію попиту та ціни [119], Power Management Problems [120].

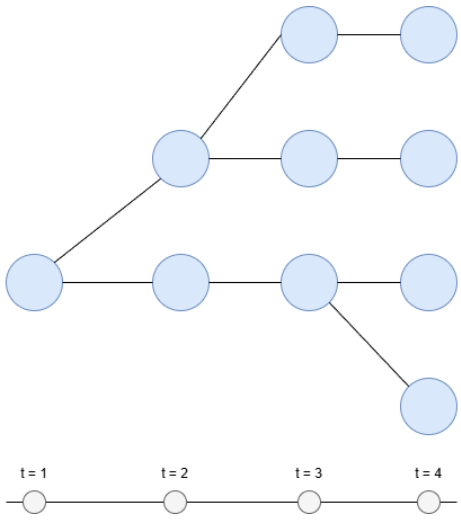

Рисунок 2.6 – Приклад дерева сценаріїв

Business Process Model and Notation (BPMN) – стандарт візуалізації та проєктування бізнес-процесів. Символи BPMN використовуються для представлення різних аспектів бізнес-процесів, що забезпечує для користувачів можливість отримати чітке та зрозуміле представлення робочих процесів, документувати, аналізувати та вдосконалювати модель. Символи BPMN класифікуються на основі їх функцій [121, 122]:

– flow objects, які використовуються для моделювання логіки процесу від початку до кінця з використанням елементів: events (start, intermediate, end), activities (task, sub-process, transaction, call activity), gateways (exclusive, inclusive, parallel, complex, event-based, exclusive event-based, parallel event-based);

– connecting objects, які використовуються для визначення зв'язків між елементами, візуалізації послідовності та залежності в процесі; основними типами є: sequence flows, message flows, associations flows.

– swimlanes, які використовується для організації дій в бізнес-процесі – процес розділяється на окремі секції, визначаються чіткі межі відповідно до учасників процесу, що полегшує розуміння виконання процесу; основними типами є: pools, lanes.

– artifacts, які використовуються як додаткові елементи для зазначення додаткової інформації до діаграми: уточнення та документування деталей процесу; основними типами є: data symbols (data object, data input/output, data store), group, annotation, association.

Моделювання бізнес-процесів за допомогою нотації BPMN надає користувачам знання про процеси, що допомагають тестувати різні сценарії, оцінювати потенційні наслідки, приймати рішення щодо покращення процесу, розподілу ресурсів. Помилки або неточності при моделюванні бізнес-процесів несуть потенційні ризики, що можуть негативно відобразитися при реалізації моделі.

Онтології є інструментом, який використовується для формалізації даних про предметну область, забезпечує структуру для зв'язки між різними компонентами, полегшує обмін та інтеграцію даних та виведення нових знань на

основі визначених концептів. Для опису онтології використовуються мови онтології, які мають різні можливості. Популярною мовою для опису онтології є OWL 2 (Web Ontology Language 2), що має розширений та покращений функціонал порівняно з попередньою версією OWL. Онтологія складається з наступних основних компонентів [123]:

1. Класи (Classes) – множини, що містять індивідуумів. Класи описуються за допомогою формальних (математичних) тверджень, які визначають вимоги приналежності індивідуумів до класу. Простим прикладом використання класів є побудова таксономії у вигляді ієрархії суперклас-підклас. Ієрархія класів OWL підтримує множинне наслідування – в одного класу може мати декілька батьківський класів.

2. Властивості (Properties) використовуються для визначення зв'язків між індивідуумами. В OWL для властивостей є можливість додати характеристики:

2.1. Функціональність (Functional). Для даного індивідуума може існувати щонайбільше один індивідуум, що пов'язаний з індивідуумом за допомогою властивості. Функціональна властивість еквівалентна властивості з обмеженням кількості (cardinality restriction), зі значенням max = 1. В математиці відношення $f \subset X \times Y$ є функціональним Кожному $x \in X$: $(x, y) \in f$ відповідає один і тільки один елемент $y \in Y$.

2.2. Обернена функціональність (Inverse Functional). Якщо властивість є обернено функціональною, то обернена властивість є функціональною.

2.3. Транзитивність (Transitive). Відношення $R \subseteq X \times X$ на множині X називається транзитивним, якщо для будь-яких x_i, x_j, x_k з $x_i R x_j$ і $x_j R x_k$ слідує $x_i R x_k$. Крім того транзитивна властивість не може бути функціональною, а обернена властивість також має бути транзитивною. Наприклад, якщо x > y і y > z, то x > z. Оберненим знаком до > є <, і відношення з < також є транзитивним: x < y і y < z, то x < z.

2.4. Симетричність (Symmetric). Відношення $R \subseteq X \times X$ на множині X називається симетричним, якщо для будь-якої пари (x_i, x_j) $x_i R x_j$ слідує $x_j R x_i$.

2.5. Асиметричність (Asymmetric). Відношення $R \subseteq X \times X$ на множині X називається асиметричним, якщо $R \cap R^{-1} \subseteq \emptyset$ - не може мати симетричних значень.

2.6. Рефлексивність (Reflexive). Відношення $R \subseteq X \times X$ на множині X називається рефлексивним, якщо виконується умова $\forall x(xRx)$. Наприклад, рівність є найпоширенішим прикладом рефлексивної властивості.

2.7. Антирефлексивне (Irreflexive). Відношення $R \subseteq X \times X$ на множині X називається антирефлексивним, якщо для всіх $x \in X$ не виконується умова xRx.

3. Індивіди (Individuals) – екземпляри, що можуть належати до кількох класів, мають властивості, які описують їх взаємодію між собою.

Для виведення нових висновків на основі побудованої онтології використовується механізм міркування (Reasoner) та різні інструменти [124]. Розробка онтології є важливим етапом, що допомагає формалізувати знання про предметну область, робити нові висновки на основі визначених концептів, полегшує розуміння та інтерпретацію роботи елементів складних систем [125]. Підходи на основі онтології використовується для захисту критичної інфраструктури [126], для аналізу операцій та прийняття рішень на основі даних в інформаційних та комунікаційних системах [127]. Онтологія складається з трьох частин, які мають зв’язки між собою та використовуються для побудови бази знань, яка може використовуватися користувачами для виявлення вразливостей та операцій для їхній протидії [127]:

– онтологія інформаційно-комунікаційних технологій;

– онтологія вразливості;

– онтологія атак.

Для представлення знань, семантичної сумісності компонентів критичної інфраструктури та впливу небезпечних природних подій використовується онтологія InfraRisk, яка може застосовуватися для прийняття рішень при виникненні відповідних подій [128]. Каскадні ефекти в критичній інфраструктурі можуть бути описані за допомогою онтології для кращого розуміння їхньої поведінки: створення, поширення та впливу на підвищення стійкості системи [129]. Існують різні методології, патерни та рекомендації для побудови онтології [130, 131]. Крім патернів, існують ще антипатерни, які встановлюються в процесі аналізу наявних онтологічних моделей [132]. Аналіз подібних досліджень надає можливість зменшити кількість помилок при побудові онтологічної моделі, що потенційно збільшить її якість. Онтології рекомендовано оновлювати, розширювати, адаптувати до змін у предметній області, оскільки ефективність онтологічної моделі залежить від актуальності та рівня деталізації знань. Обмеженість інформації про процеси в предметній області може стати вузьким місцем при аналізі та побудові онтологічної моделі. У результаті моделі можуть бути значно спрощені (рівень деталізації менший), що потенційно впливає на втрату важливих деталей. Онтології з обмеженими рівнями деталізації використовуються для полегшення процесу моделювання та доповнюються даними за необхідності.

Граф знань (Knowledge graph) використовується для представлення фактів та семантичних зв’язків між ними та визначається за формулою (2.2):

$$G = (x_1, r_1, x_2), \tag{2.2}$$

де x_1 – вершина (суб’єкт);

r_1 – відношення (предикат);

x_2 – вершина (об’єкт).

Онтологія може бути використана для визначення схеми для графа знань, оскільки визначає класи, відношення, властивості об’єктів у предметній області, тому відіграє важливу роль при побудові графа знань [133].

2.3. Розробка методу семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі онтологічної моделі та множини правил

Моделювання та симуляція сценаріїв каскадних збоїв в електромережах є складним процесом, який потребує структуризації знань про предметну область. Метод семантичного моделювання сценаріїв каскадних збоїв в електромережах використовується для формалізації, перевірки та доповнення даних про структуру електромережі й процеси розвитку каскадних збоїв на основі онтологічної моделі та множини правил.

2.3.1. Побудова онтологічної моделі електромережі

Онтологічна модель предметної області визначається за формулою (2.3):

$$O = \{C, P, R, S\}, \tag{2.3}$$

де $C = \{C_1, \dots, C_n\}$ – множина класів;

$P = \{P_1, \dots, P_n\}$ – множина властивостей;

$R = \{R_1, \dots, R_n\}$ – множина відношень між класами та властивостями;

$S = \{S_1, \dots, S_n\}$ – множина правил-обмежень [134].

Онтологічна модель електромережі визначається у вигляді графу, який відображає компоненти та зв’язки між ними, що визначаються за формулами (2.4 – 2.8): шини (Bus) – вузли (Node) в графі; лінії електропередачі (PowerLine) та трансформатори (Transformer) – гілки (Branch) і відповідно є ребрами (Edge) в графі, що з’єднують (isConnectedTo) вершини графа – шини, до яких можуть бути підключені компоненти [134]:

$$Branch \equiv Edge, \qquad Bus \equiv Node; \tag{2.4}$$

$$Branch \equiv PowerLine \sqcup Transformer,$$
$$Branch \sqsubseteq GraphComponent,$$
$$Branch \sqsubseteq GridComponent; \quad (2.5)$$

$$Edge \equiv PowerLine \sqcup Transformer; \quad (2.6)$$

$$Edge \sqsubseteq GraphComponent, Edge \sqsubseteq GridComponent,$$
$$Node \sqsubseteq GraphComponent, Node \sqsubseteq GridComponent; \quad (2.7)$$

$$Edge \sqsubseteq \exists isConnectedTo.Node. \quad (2.8)$$

Шина є компонентом графу (GraphComponent), мережі (GridComponent) та має з'єднання (hasConnectionWith) з генератором (Generator), навантаженням (Load), лінією електропередачі (PowerLine), трансформатором (Transformer) [134], що визначається за формулами (2.9 – 2.10):

$$Bus \sqsubseteq GraphComponent, Bus \sqsubseteq GridComponent; \quad (2.9)$$

$$Bus \sqsubseteq \exists hasConnectionWith.\begin{pmatrix} Generator \sqcup Load \\ \sqcup PowerLine \sqcup Transformer \end{pmatrix}. \quad (2.10)$$

Шина містить властивості, які мають дробове значення (double) та визначені допустимі значення: hasActivePowerMWValue – значення активної потужності (МВт), hasKiloVoltValue – значення напруги (кВ), hasReactivePowerMVArValue – значення реактивної потужності (МВАр), hasVoltageAngleDegreeValue – кут напруги (градуси). Значення hasKiloVoltValue, hasVoltageAngleDegreeValue можуть бути ≥ 0 [134], визначається за формулами (2.11 – 2.14):

$$Bus \sqsubseteq \exists hasActivePowerMWValue.double; \quad (2.11)$$

$$Bus \sqsubseteq \exists\, hasKiloVoltValue.double,\ Range\colon double[\geq\ "0.0"\hat{}\hat{}double]; \quad (2.12)$$

$$Bus \sqsubseteq \exists\, hasReactivePowerMVArValue.double; \quad (2.13)$$

$$Bus \sqsubseteq \exists\, hasVoltageAngleDegreeValue.double,\ Range\colon double[\geq\ "0.0"\hat{}\hat{}double]. \quad (2.14)$$

Джерелом електроенергії в мережі є генератори – електростанції, що підключені до шин, визначається за формулами (2.15 – 2.17):

$$Generator \equiv PowerStation; \quad (2.15)$$

$$Generator \sqsubseteq GridComponent, Generator \sqsubseteq PowerGeneration; \quad (2.16)$$

$$Generator \sqsubseteq \exists\, isConnectedTo.Bus. \quad (2.17)$$

Для джерела живлення визначено властивості, що визначаються за формулами (2.18 – 2.24): hasActivePowerMWValue, hasMaxActivePowerMWValue – максимальне значення активної потужності (МВт), hasMaxReactivePowerMVArValue – максимальне значення реактивної потужності (МВАр), hasMinActivePowerMWValue – мінімальне значення активної потужності (МВт), hasMinReactivePowerMVArValue – мінімальне значення реактивної потужності (МВАр), hasReactivePowerMVArValue, hasVoltageMagnitudePUValue – величина напруги в умовних одиницях:

$$Generator \sqsubseteq \exists\, hasActivePowerMWValue.double; \quad (2.18)$$

$$Generator \sqsubseteq \exists\, hasMaxActivePowerMWValue.double; \quad (2.19)$$

$$Generator \sqsubseteq \exists\, hasMaxReactivePowerMVArValue.double; \quad (2.20)$$

$$Generator \sqsubseteq \exists\, hasMinActivePowerMWValue.double; \quad (2.21)$$

$$Generator \sqsubseteq \exists\, hasMinReactivePowerMVArValue.double; \quad (2.22)$$

$$Generator \sqsubseteq \exists\, hasReactivePowerMVArValue.double; \quad (2.23)$$

$$Generator \sqsubseteq \exists\, hasVoltageMagnitudePUValue.double, \; Range: double[\geq \text{"0.0"}\hat{}\hat{}double]. \quad (2.24)$$

Для позначення навантаження (об'єкти споживають електроенергію) використовується клас Load. Споживачі електроенергії (PowerConsumer) поділяються на 3 типи:

а) комерційні споживачі (CommercialConsumer): офіси, магазини, які мають низький рівень напруги, визначаються за формулою (2.25):

$$CommercialConsumer \sqsubseteq \forall\, hasVoltageLevel.Low-Voltage. \quad (2.25)$$

б) індустріальні споживачі (IndustrialConsumer), які узагальнені у 3 класи:

1) великі виробництва використовують надвисокий та високий рівні напруги, визначаються за формулою (2.26):

$$LargeScaleIndustry \sqsubseteq \forall\, hasVoltageLevel.\begin{pmatrix} Extra-High-Voltage \\ \sqcup\ High-Voltage \end{pmatrix}. \quad (2.26)$$

2) середні виробництва використовують високий та середній рівні напруги, визначаються за формулою (2.27):

$$MediumScaleIndustry \sqsubseteq \forall\, hasVoltageLevel.\begin{pmatrix} High-Voltage \\ \sqcup\ Medium-Voltage \end{pmatrix}. \quad (2.27)$$

3) малі виробництва використовують низький та середній рівні напруги, визначаються за формулою (2.28):

$$SmallScaleIndustry \sqsubseteq \forall\, hasVoltageLevel.\begin{pmatrix} Low-Voltage \\ \sqcup\ Medium-Voltage \end{pmatrix}. \quad (2.28)$$

в) побутові споживачі (ResidentialConsumer), що використовують низький рівень напруги, визначаються за формулою (2.29):

$$ResidentialConsumer \sqsubseteq \forall\, hasVoltageLevel.Low-Voltage. \quad (2.29)$$

Навантаження в електромережі при моделюванні підключається (isConnectedTo) до шини та має властивості hasActivePowerMWValue, hasReactivePowerMVArValue, визначаються за формулами (2.30 – 2.33):

$$Load \equiv PowerConsumer, Load \sqsubseteq GridComponent; \quad (2.30)$$

$$Load \sqsubseteq \exists\, isConnectedTo.Bus; \quad (2.31)$$

$$Load \sqsubseteq \exists\, hasActivePowerMWValue.double; \quad (2.32)$$

$$Load \sqsubseteq \exists\, hasReactivePowerMVArValue.double. \quad (2.33)$$

Рівень напруги (VoltageLevel) класифікується в 4 категорії, які мають визначені інтервали, що використовуються для перевірки коректності даних при наповненні онтології:

– надвисока напруга (Extra-High-Voltage), інтервал (230,0; 765,0] кВ, визначається за формулою (2.34):

$$Extra - High - Voltage \sqsubseteq \forall\, hasKiloVoltValue.double\begin{bmatrix} > & "230.0"\text{^^}double, \\ \leq & "765.0"\text{^^}double \end{bmatrix}. \quad (2.34)$$

– висока напруга (High-Voltage), інтервал (35,0; 230,0] кВ, визначається за формулою (2.35):

$$High - Voltage \sqsubseteq \forall\, hasKiloVoltValue.double\begin{bmatrix} > & "35.0"\text{^^}double, \\ \leq & "230.0"\text{^^}double \end{bmatrix}. \quad (2.35)$$

– середня напруга (Medium-Voltage), інтервал (1,0; 35,0] кВ, визначається за формулою (2.36):

$$Medium - Voltage \sqsubseteq \forall\, hasKiloVoltValue.double\begin{bmatrix} > & "1.0"\text{^^}double, \\ \leq & "35.0"\text{^^}double \end{bmatrix}. \quad (2.36)$$

– низька напруга (Low-Voltage), інтервал [0,0; 1,0] кВ, визначається за формулою (2.37):

$$Low - Voltage \sqsubseteq \forall\, hasKiloVoltValue.double\begin{bmatrix} \geq & "0.0"\text{^^}double, \\ \leq & "1.0"\text{^^}double \end{bmatrix}. \quad (2.37)$$

Важливим елементом в електромережі є трансформатор, що з’єднує шини та підвищує/знижує напругу, визначається за формулами (2.38 – 2.40):

$$Transformer \sqsubseteq Branch, Transformer \sqsubseteq Edge; \quad (2.38)$$

$$Transformer \sqsubseteq \exists\, isConnectedTo.Bus; \quad (2.39)$$

$$Transformer \sqsubseteq \exists\, hasKiloVoltValue.double, \quad Range: double[\geq\ "0.0"\text{^^}double]. \quad (2.40)$$

Лінія електропередачі є важливим компонентом, що з'єднує шини та має властивості: hasLengthKMValue – довжина (км), hasResistancePUValue – опір (умовні одиниці), hasReactancePUValue – реактивний опір (умовні одиниці), hasShuntSusceptancePUValue – уявна частина адмітансу (умовні одиниці), визначаються за формулами (2.41 – 2.46):

$$PowerLine \sqsubseteq Branch, PowerLine \sqsubseteq Edge; \quad (2.41)$$

$$PowerLine \sqsubseteq \exists\, isConnectedTo.Bus; \quad (2.42)$$

$$PowerLine \sqsubseteq \exists\, hasLengthKMValue.double, \\ Range: double[\geq \textit{"0.0"}\text{^^}double]; \quad (2.43)$$

$$PowerLine \sqsubseteq \exists\, hasReactancePUValue.double; \quad (2.44)$$

$$PowerLine \sqsubseteq \exists\, hasResistancePUValue.double; \quad (2.45)$$

$$PowerLine \sqsubseteq \exists\, hasShuntSusceptancePUValue.double. \quad (2.46)$$

Електромережі поділяються на 2 типи: транспортування та розподіл електроенергії. Відповідно до типу мережі можуть використовуватися спеціальний тип обладнання та застосовуються обмеження в мережі, тому модель розширюється за допомогою нових термінів, визначаються за формулами (2.47 – 2.51):

$$PowerGridModel \equiv DistributionGrid \sqcup TransmissionGrid; \quad (2.47)$$

$$PowerLine \equiv DistributionLine \sqcup TransmissionLine; \quad (2.48)$$

$$Transformer \equiv DistributionTransformer \sqcup TransmissionTransformer; \quad (2.49)$$

$$DistributionGrid \sqsubseteq \forall\, hasKiloVoltValue.double, Range: double[> "0.0"^^double, \leq "110.0"^^double]; \quad (2.50)$$

$$TransmissionGrid \sqsubseteq \forall\, hasKiloVoltValue.double, Range: double[> "110.0"^^double, \leq "765.0"^^double]. \quad (2.51)$$

На основі визначених класів, властивостей об'єктів, властивостей даних створюються екземпляри, що описують предметну область електромережі.

Тестова мережа (IEE9GRID) зображена на рисунку 2.7, складається з 9 шин, що описані за допомогою екземплярів: Bus1, Bus2, Bus3, Bus4, Bus5, Bus6, Bus7, Bus8, Bus9; 3 генератори: Gen1, Gen2, Gen3; 9 ліній електропередачі: Line1, Line2, Line3, Line4, Line5, Line6, Line7, Line8, Line9; 3 навантажень: Load1, Load2, Load3 [134].

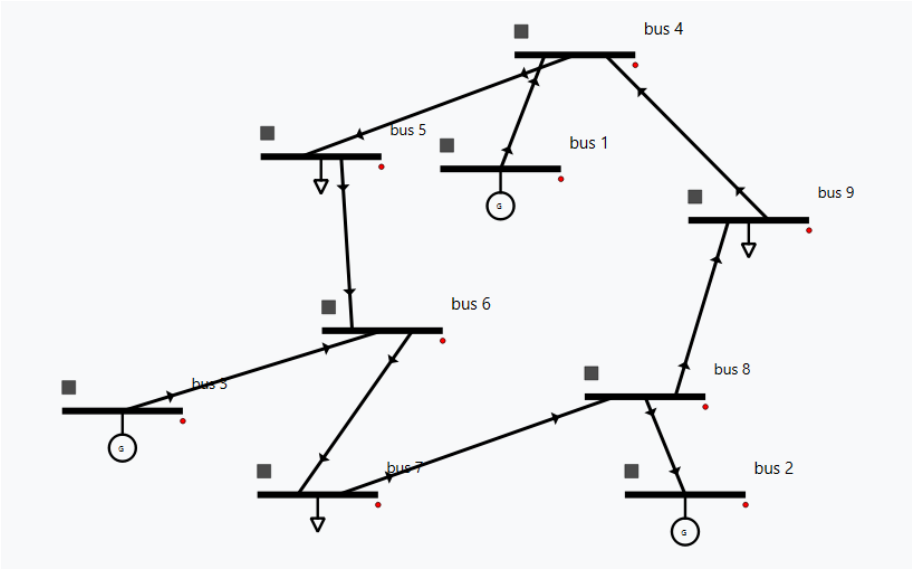

Рисунок 2.7 – Приклад електромережі case9, яка міститься в MATPOWER [135], візуалізована за допомогою програмного забезпечення VeraGrid [136]

Приклад характеристик шини Bus1, яка зображена на рисунку 2.8:

– тип Bus;

– isPartOf(Bus1, IEEE9GRID) – Bus1 входить до мережі IEEE9GRID;

– hasActiveStatusValue(Bus1, true) – статус шини (активний);

– hasBusTypeValue(Bus1, 3) – тип шини (3 – slack bus);

– hasKiloVoltValue(Bus1, 345.0) – напруга 345.0 (кВ);

– hasMaxVoltageMagnitudePUValue(Bus1, 1.1) – максимальна напруга 1.1 (умовні одиниці);

– hasMinVoltageMagnitudePUValue(Bus1, 0.9) – мінімальна напруга 0.9 (умовні одиниці);

– hasVoltageAngleDegreeValue(Bus1, 0.0) – кут напруги 0.0 (градуси);

– hasVoltageMagnitudePUValue(Bus1, 1.0) – напруга 1.0 (умовні одиниці).

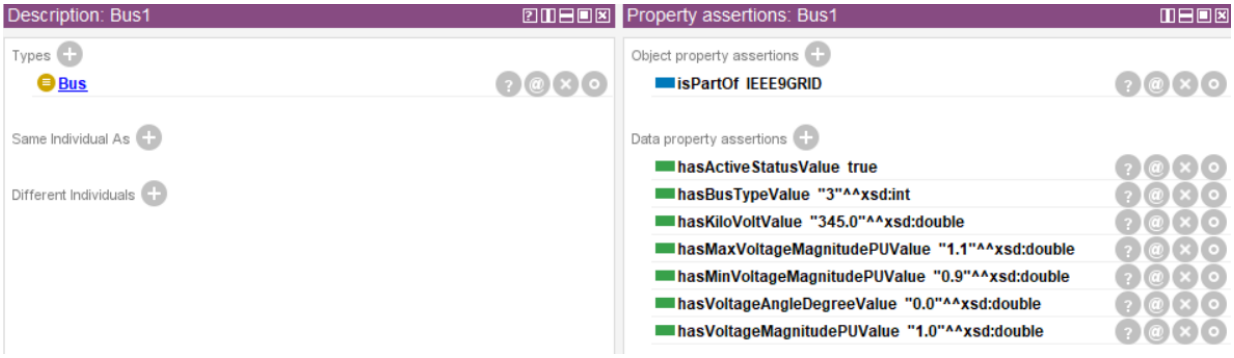

Рисунок 2.8 – Опис екземпляра Bus1 в онтології електромережі [134]

Приклад характеристик генератора Gen1:

– тип Generator;

– isConnectedTo(Gen1, Bus1) – генератор Gen1 під’єднаний до шини Bus1;

– isLocatedAt(Gen1, OutOfCity) – генератор розташований за межами міста;

– isPartOf(Gen1, IEEE9GRID) – Gen1 входить до мережі IEEE9GRID;

– hasActivePowerMWValue(Gen1, 72.3) – активна потужність 72.3 МВт;

– hasActiveStatusValue(Gen1, true) – активний статус роботи генератора;

– hasMaxActivePowerMWValue(Gen1, 250.0) – максимальна активна потужність 250.0 МВт;

– hasMaxReactivePowerMVArValue(Gen1, 300.0) – максимальна реактивна потужність 300.0 МВАр;

– hasMinActivePowerMWValue(Gen1, 10.0) – мінімальна активна потужність 300.0 МВт;

– hasMinReactivePowerMVArValue(Gen1, -300.0) – мінімальна реактивна потужність -300.0 МВАр;

– hasReactivePowerMVArValue(Gen1, 27.03) – реактивна потужність 27.03 МВАр;

– hasVoltageMagnitudePUValue(Gen1, 1.04) – напруга 1.04 умовних одиниць.

Приклад характеристик лінії електропередачі Line1:

– тип PowerLine;

– isConnectedTo(Line1, Bus1), isConnectedTo(Line1, Bus4) – лінія електропередачі Line1 з'єднує шини Bus1 та Bus4;

– isPartOf(Line1, IEEE9GRID) – Line1 входить до мережі IEEE9GRID;

– hasReactancePUValue(Line1, 0.0576) – реактивний опір 0.0576 (умовні одиниці);

– hasResistancePUValue(Line1, 0.0) – опір 0.0 (умовні одиниці);

– hasShuntSusceptancePUValue(Line1, 0.0) – уявна частина адмітансу 0.0 (умовні одиниці).

Приклад характеристик навантаження Load1:

– тип Load;

– isConnectedTo(Load1, Bus5) – навантаження Load1 під'єднане до шини Bus5;

– isLocatedAt(Load1, City) – навантаження розташовано в населеному пункті;

– isPartOf(Load1, IEEE9GRID) – Load1 входить до мережі IEEE9GRID;

– hasActivePowerMWValue(Load1, 90.0) – активна потужність 90.0 (МВт);

– hasActiveStatusValue(Load1, true) – активний статус роботи навантаження;

– hasReactivePowerMVArValue(Load1, 30.0) – реактивна потужність 30.0 (МВАр).

2.3.2. Перевірка коректності та повноти онтологічної моделі

Резонер Pellet [137] у редакторі онтології Protégé [138] перевіряє коректність даних, виявляє помилки, додає висновки (додаткові метадані) [134], приклад зображений на рисунку 2.9. При помилці в даних резонер виводить помилку. Приклад помилки зображено на рисунку 2.10.

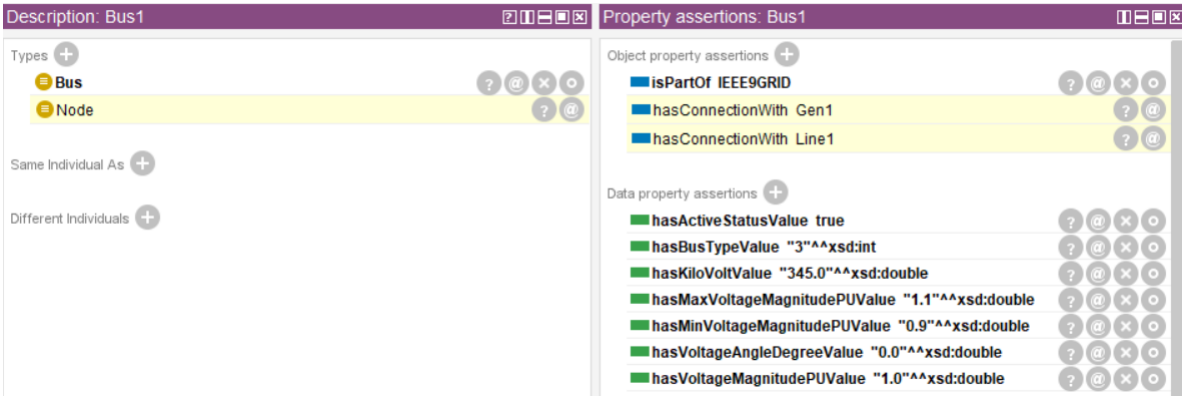

Рисунок 2.9 – Виведення висновків для екземпляра Bus1 [134]

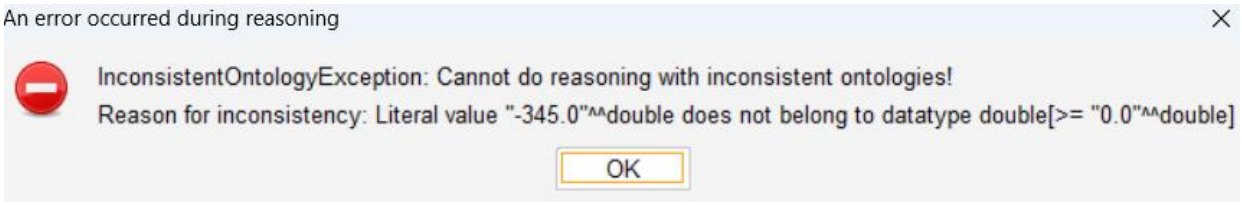

Рисунок 2.10 – Помилка в даних екземпляра Bus1 [134]

Якщо дані можуть інтегруватися з різних джерел, то додатково можна встановити правила перевірки формату вхідних даних за допомогою SHACL. За допомогою обмежень встановлюється кількість властивостей для екземплярів класу. Наприклад, для класу Line обов'язковими є властивості: isPartOf, hasReactancePUValue, hasResistancePUValue, hasShuntSusceptancePUValue, isConnectedTo, що визначаються за допомогою обмежень minCount, maxCount [134]. Приклад обмежень зображено на рисунку 2.11, результати валідацій – на рисунку 2.12 та рисунку 2.13.

```
powergrid:LineShape
    a sh:NodeShape ;
    sh:targetClass powergrid:PowerLine ; # Applies to all PowerLines
    sh:property [
        sh:path powergrid:isPartOf ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path powergrid:isConnectedTo ;
        sh:minCount 2 ;
        sh:maxCount 2 ;
    ] ;
    sh:property [
        sh:path powergrid:hasReactancePUValue ;
        sh:datatype xsd:double ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path powergrid:hasResistancePUValue ;
        sh:datatype xsd:double ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path powergrid:hasShuntSusceptancePUValue ;
        sh:datatype xsd:double ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:closed false ;
    sh:ignoredProperties ( rdf:type owl:topDataProperty owl:topObjectProperty )
    .
```

Рисунок 2.11 – Обмеження для екземплярів класу лінії електропередачі в електромережі, які візуалізовані за допомогою shacl-playground [139]

```
<!-- http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Line1 -->

<owl:NamedIndividual rdf:about="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Line1">
    <rdf:type rdf:resource="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Powerline"/>
    <untitled-ontology-56:isConnectedTo rdf:resource="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Bus1"/>
    <untitled-ontology-56:isConnectedTo rdf:resource="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Bus4"/>
    <untitled-ontology-56:isConnectedTo rdf:resource="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Bus5"/>
    <untitled-ontology-56:isPartOf rdf:resource="http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#IEEE9GRID"/>
```

Validation Report

Success

No

Errors found

- http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#Line1:
  - http://www.semanticweb.org/alexandr/ontologies/2025/2/untitled-ontology-56#isConnectedTo:
    - More than 2 values

Рисунок 2.12 – Результат валідації онтології з помилковими даними [134]

Оскільки в екземплярі Line1 класу PowerLine визначено властивість isConnectedTo 3 рази (лінія електропередачі під’єднана до Bus1, Bus4, Bus5), то виникає відповідно помилка при валідації згідно визначених обмежень SHACL. Поєднання онтологічної моделі та SHACL покращує якість даних та доповнює їх, що полегшує аналіз предметної області [134]. У випадку коректних даних згідно визначених обмежень формується успішний звіт.

Рисунок 2.13 – Результат успішної валідації онтології

2.3.3. Побудова сценаріїв каскадних збоїв електромережі на основі онтологічної моделі

Онтологічна модель складається з класів, властивостей та екземплярів, які описують сценарії роботи електромережі. Під час визначення сценаріїв додаються метадані про функціонування електромережі в сценарії: виконується множина дій (Action) для забезпечення роботи мережі [140], які можна визначити як класи в онтології [134]:

– ExcitationLimiter – захисний механізм для запобігання пошкодження генератора;

– GneratorTrpippingDueToUnderload – генератори в шинах відключаються через недовантаження;

– IslandTripping – відключення острова мережі;

– OverfrequencyGeneratorShedding – захисний механізм зниження частоти генератора використовується для відновлення балансу механічної та електричної потужності, щоб частота залишалася в межах допустимих значень;

– OverloadProtection – лінії електропередачі, в яких перевищується значення номінального навантаження, відключаються від перевантаження;

– RemoveShuntDevices – відключення шунтуючих пристроїв;

– UnderfrequencyLoadShedding – система захисту, що використовується для балансування механічної та електричної потужності, щоб забезпечити значення частоти у визначеному інтервалі;

– UndervoltageLoadShedding – зменшення навантаження для підтримки рівня напруги;

– VoltageCollapseLoadShedding – механізм використовується для перетворення нерозв’язуваного потоку потужності в розв’язуваний.

Події (Event), які виникають в процесі моделювання та симуляції роботи електромережі класифікуються у категорії:

– збій (Failure) – початковий збій компонентів (визначаються як неактивні);

– поділення мережі на острови (Islanding) – утворюються автономні підмережі;

– стабілізація мережі (NetworkStability) – мережа перебуває в стабільному стані;

– мережа без збоїв (NoFailure);

– знеструмлення (Blackout) – мережа відключена;

– початок каскаду (CascadeStarted) – в результаті збоїв почався каскад;

– продовження каскаду (CascadeContinues) – каскадні збої впливають на роботу електромережі та може викликати множину послідовних подій;

– завершення каскаду (CascadeFinished) – робота електромережі була стабілізована або вона була знеструмлена для захисту її компонентів.

Онтологічна модель доповнюється важливими характеристиками, які відображаються послідовність зміни в роботі електромережі:

– ідентифікатором сценарію (hasscenario_number);

– ідентифікатором похідного сценарію (hassubscenario_number);

– назвою похідного сценарію (hassubscenario_name), що описує процес, який відбувається в острові;

– ідентифікатором острову (hasisland_id) – мережа може бути поділена на множину островів;

– ідентифікатором первинного острову (hasparent_island_id) – острів, який поділився на декілька островів.

На основі даних моделювання та симуляції створюються екземпляри компонентів електромережі. За допомогою SWRL [141] онтологічна модель доповнюється новими знаннями [134], наприклад:

– визначення початкової структури електромережі (наприклад, шини): Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ swrlb:equal(?sname, "Init"^^rdf:PlainLiteral) ^ PowerGridModel(?a) -> hasPart(?a, ?b);

– зв’язування сценаріїв та островів (острова, з яких складається сценарій): Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ Scenario(?s) ^ hasscenario_number(?b, ?sn) ^ hassubscenario_number(?b, ?ssn) ^ hasscenario_number(?s, ?s_sn) ^ hassubscenario_number(?s, ?s_ssn) ^ swrlb:equal(?sn, ?s_sn) ^ swrlb:equal(?ssn, ?s_ssn) ^ Island(?isl) ^ hasisland_id(?isl, ?isl_id) ^ hasisland_id(?b, ?b_isl_id) ^ swrlb:equal(?isl_id, ?b_isl_id) -> hasPart(?s, ?isl);

– визначення для острову первинного та похідних островів: Island(?islnd1) ^ hasparent_island_id(?islnd1, ?p_islnd) ^ Island(?islnd2) ^ hasisland_id(?islnd2, ?islnd_id) ^ swrlb:equal(?islnd_id, ?p_islnd) -> hasParent(?islnd1, ?islnd2);

– визначення для сценарію первинного та похідних сценаріїв: Scenario(?b) ^ Scenario(?s) ^ hasscenario_number(?b, ?sn) ^ hassubscenario_number(?b, ?ssn) ^ hasscenario_number(?s, ?s_sn) ^ hassubscenario_number(?s, ?s_ssn) ^ swrlb:equal(?sn, ?s_sn) ^ swrlb:add(?new_s_sn, ?s_ssn, 1) ^ swrlb:equal(?ssn, ?new_s_sn) -> hasParent(?b, ?s);

– визначення шин, які належать до островів: Bus(?b) ^ Scenario(?s) ^ hasscenario_number(?b, ?sn) ^ hassubscenario_number(?b, ?ssn) ^ hasscenario_number(?s, ?s_sn) ^ hassubscenario_number(?s, ?s_ssn) ^ swrlb:equal(?sn, ?s_sn) ^ swrlb:equal(?ssn, ?s_ssn) ^ Island(?isl) ^ hasisland_id(?isl, ?isl_id) ^ hasisland_id(?b, ?b_isl_id) ^ swrlb:equal(?isl_id, ?b_isl_id) -> hasPart(?isl, ?b);

– визначення процесу утворення островів: Island(?islnd1) ^ hasChild(?islnd1, ?p_islnd) ^ Islanding(?isl) -> hasEvent(?islnd1, ?isl);

– визначення знеструмлення: Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ swrlb:equal(?sname, "Blackout"^^rdf:PlainLiteral) ^ Blackout(?a) ^ isPartOf(?b, ?isl) -> hasEvent(?isl, ?a);

– визначення збоїв компонентів: Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ swrlb:equal(?sname, "initial_contingency"^^rdf:PlainLiteral) ^ Failure(?a) ^ isPartOf(?b, ?isl) -> hasEvent(?isl, ?a);

– застосування захисного механізму UndervoltageLoadShedding: Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ swrlb:equal(?sname, "UVLS"^^rdf:PlainLiteral) ^ UndervoltageLoadShedding(?a) ^ isPartOf(?b, ?isl) -> hasAction(?isl, ?a);

– визначення шин, які були відключені у острові: Island(?isl) ^ Bus(?b) ^ hasPart(?isl, ?b) ^ hasBusType(?b, ?tr) ^ swrlb:equal(?tr, 4) -> hasTrippedBus(?isl, ?b);

– визначення початку каскадного ефекту (початковий острів поділився на множину островів): Island(?islnd1) ^ hasChild(?islnd1, ?p_islnd) ^ hasisland_id(?islnd1, ?id) ^ hasparent_island_id(?islnd1, ?pid) ^ CascadeStarted(?isl) ^ swrlb:equal(?id, ?pid) -> hasEvent(?islnd1, ?isl);

– визначення процесу каскаду (продовження каскаду). Каскадний ефект вже почався, але ще не закінчився: Island(?islnd1) ^ hasChild(?islnd1, ?c_islnd) ^ hasParent(?islnd1, ?p_islnd) ^ CascadeContinues(?isl) -> hasEvent(?islnd1, ?isl);

– завершення каскаду (острів стабілізувався): Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ Scenario(?s) ^ hasscenario_number(?b, ?sn) ^ hassubscenario_number(?b, ?ssn) ^ hasscenario_number(?s, ?s_sn) ^ hassubscenario_number(?s, ?s_ssn) ^ swrlb:equal(?sn, ?s_sn) ^ swrlb:equal(?ssn, ?s_ssn) ^ Island(?isl) ^ hasisland_id(?isl, ?isl_id) ^ hasisland_id(?b, ?b_isl_id) ^ swrlb:equal(?isl_id, ?b_isl_id) ^ CascadeFinished(?cf) ^ swrlb:equal(?sname, "success"^^rdf:PlainLiteral) -> hasEvent(?isl, ?cf);

– завершення каскаду (знеструмлення острову): Bus(?b) ^ hassubscenario_name(?b, ?sname) ^ Scenario(?s) ^ hasscenario_number(?b, ?sn) ^ hassubscenario_number(?b, ?ssn) ^ hasscenario_number(?s, ?s_sn) ^ hassubscenario_number(?s, ?s_ssn) ^ swrlb:equal(?sn, ?s_sn) ^ swrlb:equal(?ssn, ?s_ssn) ^ Island(?isl) ^ hasisland_id(?isl, ?isl_id) ^ hasisland_id(?b, ?b_isl_id) ^ swrlb:equal(?isl_id, ?b_isl_id) ^ CascadeFinished(?cf) ^ swrlb:equal(?sname, "Blackout"^^rdf:PlainLiteral) -> hasEvent(?isl, ?cf).

2.3.4. Моделювання роботи електромережі

Процес моделювання роботи електромережі складається з наступних кроків, що зображені на рисунку 2.14 [134]:

– визначити перелік компонентів та зв’язки між ними, що формують структуру електромережі;

– визначити параметри компонентів електромережі;

– запустити модель потоку потужності;

– перевірити коректність структури та параметрів електромережі;

– перевірити коректність даних: за потреби оновити структуру електромережі, параметри компонентів електромережі;

– зберегти налаштування електромережі.

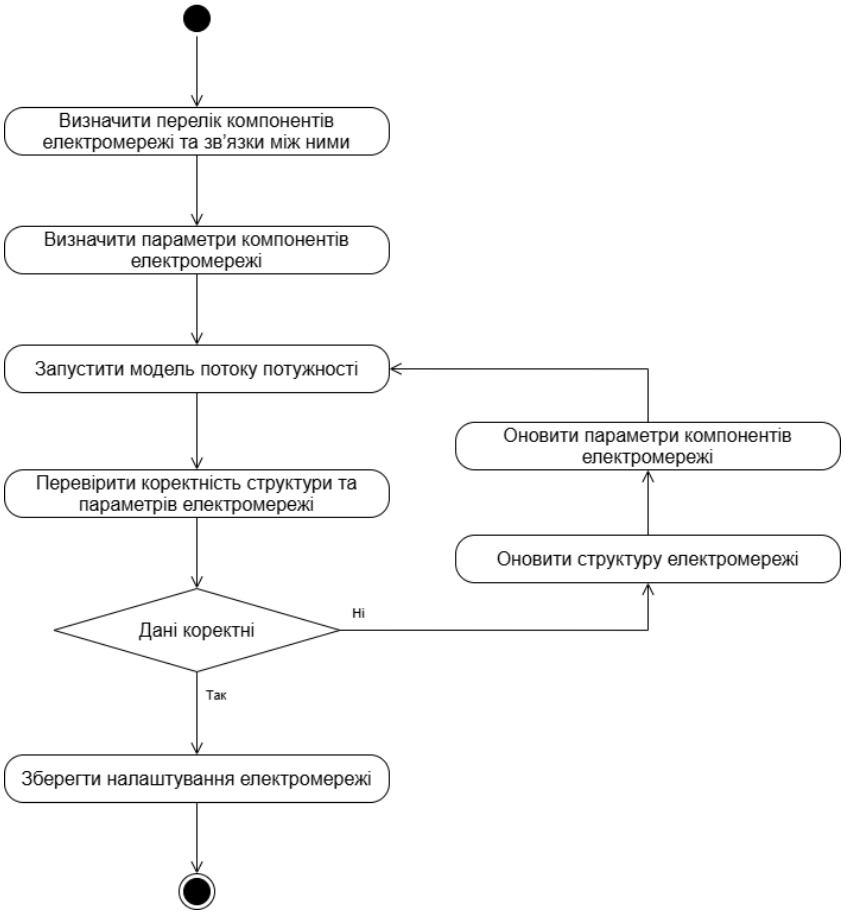

Рисунок 2.14 – Процес створення моделі електромережі [134]

2.3.5. Симуляція сценаріїв каскадних збоїв в електромережі

Сформовані сценарії запускаються в моделі симуляції, щоб визначити стани та параметри компонентів на кожному етапі розвитку можливого каскадного ефекту [134]. Приклад сценарію: на початку виводяться з ладу гілки з індексом 5 та 7, утворюється множина островів: початкова електромережа – це острів Island0, який після збою поділився на 2 острови Island1 та Island5. Потім Island5 був знеструмлений, а Island1 поділився на 3 острови: Island2, який був стабілізований, Island3, який був знеструмлений, Island4, який був знеструмлений [134]. У результаті було сформовано дерево розвитку подій у мережі, що зображено на рисунку 2.15 та відображає розвиток та вплив каскаду: листки відображають кінцеву структуру та стан підмережі (стабілізація чи знеструмлення).

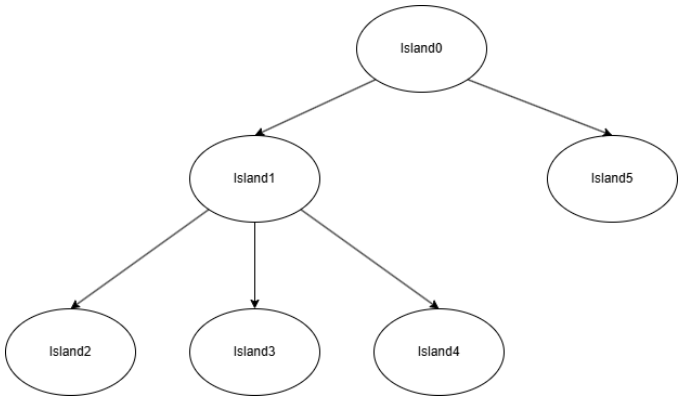

Рисунок 2.15 – Дерево розвитку подій в мережі, поділ мережі на підмережі

У результаті роботи Pellet Reasoner були отримані висновки про острови. Острів з ідентифікатором 5 (Island5) доповнений метаданими, шо зображено на рисунку 2.16 [134]:

– події: знеструмлення, каскад закінчився;

– компоненти: шина Bus_1_2_2, генератор Generator1_2_2;

– первинний острів – острів, який поділився на менші острови: Island0;

– сценарій, де використовується острів: Scenario1_2 – основний сценарій з ідентифікатором 1, та похідним сценарієм з ідентифікатором 2;

– компоненти, які були відключені: шина Bus_1_2_2, генератор Generator1_2_2.

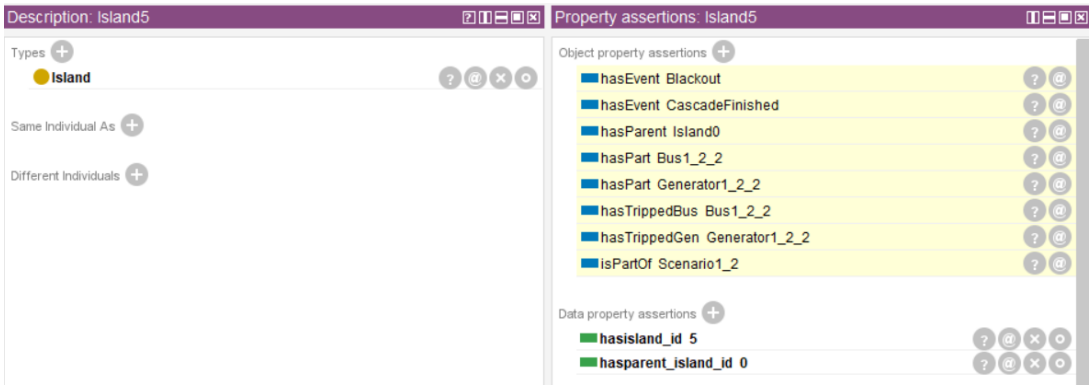

Рисунок 2.16 – Результат висновків про острів 5 (Island5) [134]

Сценарій (Scenario1_2) доповнений метаданими, що зображено на рисунку 2.17:

– складається з островів: Island1, Island5;

– має зв'язки з сценаріями: має первинний сценарій Scenario1_1, має похідний сценарій Scenario1_3.

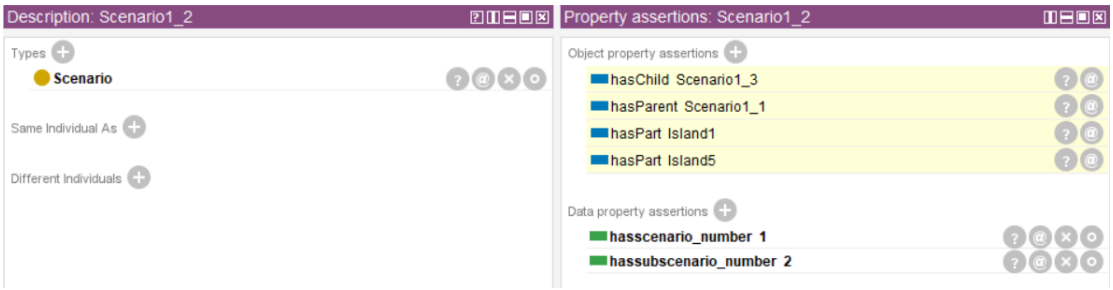

Рисунок 2.17 – Результат висновків про сценарій Scenario1_2

2.4. Висновки до розділу 2

Моделювання та симуляція розвитку каскадних збоїв в електромережі є складним процесом та потребує розуміння предметної області. Обсяг даних в електромережах збільшується в залежності від кількості об'єктів та зв'язків між ними, тому виникає потреба у формалізації даних, щоб забезпечити зручне

представлення даних для використання в аналітичних моделях. Було проведено аналіз структури та зв’язків між компонентами електромережі. Для опису предметної області було розглянуто наявні інструменти, їхні варіанти застосування.

Розроблено онтологічну модель, яка формалізує дані про структуру електромережі, характеристики компонентів, сценарії роботи при виникненні збоїв (каскадні ефекти). Розроблено семантичні правила, які використовуються для перевірки коректності даних, виявлення протиріч, доповнення даних новими висновками, що полегшує розуміння предметної області. Представлення даних у вигляді графу полегшує візуальне сприйняття зв’язків між об’єктами. Розроблена модель може бути доповнена даними (додатковими концепціями та правилами), масштабуватися відповідно до потреб користувачів.

Розроблено метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі моделі потоку потужності, що базується на використанні онтологічної моделі та множині правил для формалізації, перевірки, доповнення даних про структуру електромережі та процеси розвитку каскадних ефектів у сценаріях. Розроблений метод дозволяє під час попереднього аналізу даних сценаріїв про каскадні ефекти в роботі електромереж виявляти протиріччя й помилки, що можуть бути усунені на ранньому етапі для покращення якості даних під час використання методів машинного навчання. Створено тестові сценарії каскадних збоїв в електромережах для тестування розробленого методу.

РОЗДІЛ 3. РОЗРОБКА МЕТОДУ ВИЗНАЧЕННЯ ПОДІБНОСТІ СЦЕНАРІЇВ КАСКАДНИХ ЕФЕКТІВ В ЕЛЕКТРОМЕРЕЖАХ

При виникненні збоїв на підстанції або лінії електропередачі електроенергія перенаправляється відповідно до законів Кірхгофа та Ома, які переважно ігноруються в простих топологічних моделях, що може призвести до помилкових висновків [142, 143]. Каскадні збої в електричних мережах поширюються не локально (зображено на рисунку 3.1). Наступний збій може виникнути далеко від попереднього, що ускладнює моделювання процесу, зокрема визначення шаблонів поширення збоїв та розробку алгоритмів для запобігання та пом'якшення їх наслідків. Дослідження впливу топології мережі, каскадних механізмів (фізики) і зв'язку на вразливість мережі інфраструктури є важливим для зниження ризику каскадних збоїв [143].

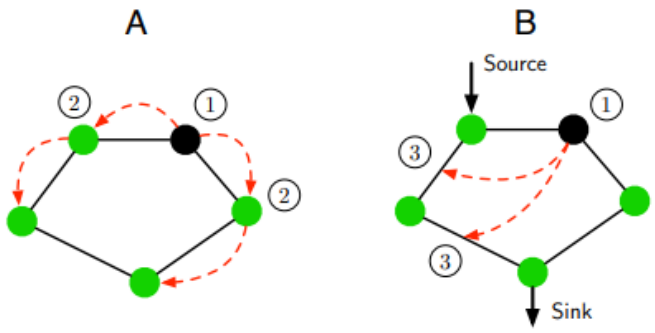

Рисунок 3.1 – Розвиток каскаду в (A) моделях топологічного зараження (topological contagion model) та (B) моделях електромережі [143]

3.1. Моделювання роботи електромережі

Аналіз потоку потужності (Power Flow Analysis або Load Flow Analysis) є основою для аналізу енергосистеми, відіграє важливу роль при проєктуванні, розширенні мережі, модифікації характеристик компонентів (наприклад, як функціонує енергосистема, як система працює з різними вхідними даними) та використовується для моніторингу аварійних ситуацій. Електромережа

моделюється за допомогою графу, який складається з вузлів (nodes) та гілок (branches), які представляють елементи системи: гілки – трансформатори та кабелі, вузли – навантаження, генератори та шунти. Також при моделюванні застосовуються терміни та рівняння на основі теорії електричних кіл. Струм і напруга є синусоїдальними функціями в системі змінного струму та визначаються на основі амплітуди, частоти та фази. При обчисленні потоку потужності використовуються середні значення струму і напруги на основі середньоквадратичних значень функцій струму та напруги [144, 145]. Миттєве значення змінного струму та напруги у момент часу t визначаються за формулами (3.1) та (3.2) відповідно:

$$i(t) = \sqrt{2}|I|\sin(\omega t + \phi_I); \tag{3.1}$$

$$v(t) = \sqrt{2}|V|\sin(\omega t + \phi_V), \tag{3.2}$$

де ω - кутова частота, рад/с;

ϕ_V і ϕ_I – фазові кути напруги і струму відповідно, які вимірюються в радіанах.

$|I|$ і $|V|$ – середньоквадратичні значення струму та напруги визначаються за формулами (3.3) та (3.4) відповідно:

$$|I| = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt}\,; \tag{3.3}$$

$$|V| = \sqrt{\frac{1}{T}\int_0^T v^2(t)dt}\,, \tag{3.4}$$

де $T = 2\pi/\omega$ – період синусоїди, с.

Для ефективної передачі електроенергії використовуються трифазні системи. У збалансованих трифазних системах струм і напруга однакові за

величиною в усіх трьох фазах, але зсунуті по фазі на $2\pi/3$ радіани. Тому задача потоку потужності трифазної системи вирішується шляхом розв'язку системи з однофазним еквівалентом, а інші дві фази обчислюються, враховуючи фазовий зсув [144, 145]. У збалансованій трифазній системі змінного струму змінна напруга та струм визначаються за формулами (3.5) та (3.6) відповідно:

$$v(t) = \sqrt{2}|V|\cos(\omega t - \delta); \tag{3.5}$$

$$i(t) = \sqrt{2}|I|\cos(\omega t - \phi - \delta), \tag{3.6}$$

де $\delta = \{0, 2\pi/3, 4\pi/3\}$ – фазовий зсув між трьома фазами, рад;

ϕ – фазовий зсув між струмом і напругою, рад.

Напруга і струм для кожної фази a, b, c визначаються за формулами (3.7–3.12):

$$v_a(t) = \sqrt{2}|V|\cos(\omega t); \tag{3.7}$$

$$i_a(t) = \sqrt{2}|I|\cos(\omega t - \phi); \tag{3.8}$$

$$v_b(t) = \sqrt{2}|V|\cos\left(\omega t - \frac{2\pi}{3}\right); \tag{3.9}$$

$$i_b(t) = \sqrt{2}|I|\cos\left(\omega t - \phi - \frac{2\pi}{3}\right); \tag{3.10}$$

$$v_c(t) = \sqrt{2}|V|\cos\left(\omega t - \frac{4\pi}{3}\right); \tag{3.11}$$

$$i_c(t) = \sqrt{2}|I|\cos\left(\omega t - \phi - \frac{4\pi}{3}\right). \tag{3.12}$$

Миттєва потужність при значенні $\delta = 0$ може бути визначена як [144, 145]:

$$p(t) = P[1 + \cos(2\omega t)] + Q[\sin(2\omega t)], \quad (3.13)$$

де $P = |V||I|\cos(\phi)$ – активна потужність, Вт;

$Q = |V||I|\sin(\phi)$ – реактивна потужність, ВАр.

Активна потужність (справжня потужність) передається від джерела до навантаження та використовується для корисної роботи (наприклад, живлення пристроїв, освітлення, механічна робота та нагрівання). Реактивна потужність використовується для підтримки електромагнітних полів в електричному ланцюзі, що дозволяє передавати активну потужність, коливається між джерелом і реактивними компонентами. Тому обидва типи потужності є важливими для функціонування системи змінного струму.

Складність сучасних електромереж зростає: збільшується кількість компонентів та їх взаємодія, потрібно підтримувати баланс між генерацією і споживанням електроенергії з урахуванням фізичних, економічних та інших умов. Наприклад, диспетчеризація генерації для задоволення попиту з мінімальними витратами, є одним із завдань, які вимагають розв'язання складної задачі оптимізації. Система передачі електроенергії моделюється у вигляді графу, який складається з таких компонентів: вузлів – шини, які є місцями, де підключаються компоненти системи (генератори, навантаження, підстанції тощо); ребра – гілки (лінії передачі, трансформатори тощо), які з'єднують шини. Обчислення потоку потужності в електромережі є одним із важливих підходів для аналізу поведінки системи в стаціонарному стані. Це дає можливість визначити, що відбувається в системі: значення напруги у вузлах та гілках енергосистеми, перевантаження елементів, слабкі місця в мережі. Метою аналізу потоку потужності є обчислення для кожної шини:

– величини напруги (voltage magnitude, |V|);

– кута напруги (voltage phase angle, δ);

– реальної потужності (injected real power, P);

– реактивної потужності (injected reactive power, Q).

Для кожної шини мережі є задані дві з цих чотирьох величин, а інші дві є невідомими. Залежно від того, які дві величини вказані, кожна із шин може бути класифікована відповідно до однієї із трьох категорій, які наведені в таблиці 3.1 [146]:

1. Slack bus (swing bus або reference bus). У системі, що аналізується, є тільки одна slack bus – це шина, для якої задано величину напруги та кут напруги, а реальна і реактивна потужність невідомі (обчислюються). Шина повинна мати джерело реальної потужності і реактивної потужності для балансування рівнянь потоку потужності (power flow equations).

2. PQ шина (шина навантаження) – шина системи, для якої задана реальна та реактивна потужність та невідомі (обчислюються) величина напруги та кут напруги. Моделюється як споживач електроенергії, що має від'ємну вхідну активну та реактивну потужність, задану у вузлі.

3. PV шина (генераторна шина) – шина системи, для якої визначено величину напруги та вхідну реальну потужність та невідомі (обчислюються) реактивна потужність та кут напруги.

Таблиця 3.1 – Категорії шин в електромережі при моделюванні

Тип шини	P	Q	|V|	δ
Slack	Не задано	Не задано	Задано	Задано
PQ	Задано	Задано	Не задано	Не задано
PV	Задано	Не задано	Задано	Не задано

У кожному вузлі (шині) *i* мережі, яка моделюється, комплексна потужність визначається за формулою (3.14) [144, 145]:

$$S_i = V_i * I_i^* = P_i + jQ_i, \tag{3.14}$$

де V_i – різниця потенціалів між вузлом *i* та землею;

I_i – струм, що надходить у вузол i;

I_i^* – спряжене комплексне значення струму;

$j = \sqrt{-1}$.

У ланцюгах змінного струму комплексний опір називається імпедансом, який визначається за формулою (3.15) та вимірюється в Омах:

$$Z = R + jX, \tag{3.15}$$

де R – активний опір;

X – реактивний опір.

Якщо значення $X > 0$, то реактивний опір є індуктивним. Якщо X < 0, то реактивний опір є ємнісним. Якщо $X = 0$, то імпеданс є резистивним, тобто в ланцюзі відсутні котушки індуктивності та конденсатори. Лінія передачі моделюється як імпеданс між двома вузлами. Обернена величина до імпедансу є адміттанс, який визначається за формулою (3.16) та вимірюється в сименсах:

$$Y = \frac{1}{Z} = G + jB = \frac{R}{Z^2} - j\frac{X}{Z^2}, \tag{3.16}$$

де G – дійсна складова повної провідності (conductance);

B – уявна складова повної провідності (susceptance).

Топологія та електричні характеристики компонентів мережі можуть бути представлені за допомогою матриці адміттансу. Використовуючи 1-ий закон Кірхгофа (Kirchhoff's Current Law) та закон Ома, система може бути описана за формулою (3.17) [144, 145]:

$$I = YV = \begin{bmatrix} I_1 \\ \dots \\ I_n \end{bmatrix} = \begin{bmatrix} Y_{11} & \dots & Y_{1n} \\ \dots & \dots & \dots \\ Y_{n1} & \dots & Y_{nn} \end{bmatrix} \begin{bmatrix} V_1 \\ \dots \\ V_n \end{bmatrix}, \tag{3.17}$$

де I – вектор вхідного струму у вузлах;

V – вектор напруги у вузлах;

Y – матриця адміттансу.

Комплексна потужність у вузлі i визначається за формулою (3.18, 3.19) [144]:

$$S_i = V_i(YV)_i^* = V_i\left(\sum_{k=1}^{N} Y_{ik}V_k\right)^*; \tag{3.18}$$

$$S_i = \sum_{k=1}^{N} |V_i||V_k|(cos\delta_{ik} + jsin\delta_{ik})(G_{ik} - jB_{ik}), \tag{3.19}$$

де δ_{ik}=$\delta_i - \delta_k$ – різниця фазових кутів між вузлом i та k.

Рівняння активної та реактивної потужності визначаються за формулами (3.20) та (3.21) відповідно:

$$P_i = \sum_{k=1}^{N} |V_i||V_k|(G_{ik}cos\delta_{ik} + B_{ik}sin\delta_{ik}); \tag{3.20}$$

$$Q_i = \sum_{k=1}^{N} |V_i||V_k|(G_{ik}sin\delta_{ik} - B_{ik}cos\delta_{ik}). \tag{3.21}$$

Рівняння потоку потужності утворюють систему нелінійних рівнянь, які можна розв’язати за допомогою чисельних методів. Існує множина методів для розв’язання рівнянь потоку потужності, які мають переваги та недоліки. На вибір методів впливає різні фактори, основними з яких є: збіжність (існування розв’язку), обчислювальна складність (використання обчислювальних ресурсів та пам’яті), гнучкість реалізації.

Метод Ньютона-Рафсона (Newton–Raphson) є одним з найпоширеніших методів для вирішення рівнянь потоків потужності, використовується для

розв'язування системи нелінійних рівнянь $F(x) = 0$. Пошук розв'язку відбувається ітеративним шляхом, поки не буде задоволений задана збіжність. Робота методу починається з початкових припущень усіх невідомих змінних, а далі на кожній ітерації відбувається обчислення корекції та оновлення попередніх значень, визначається за формулами (3.22 – 3.24) [144, 145]:

$$-J\left(x^i\right)\Delta x^i = F\left(x^i\right); \tag{3.22}$$

$$-\begin{bmatrix} \frac{\partial \Delta P}{\partial \delta} & \frac{\partial \Delta P}{\partial |V|} \\ \frac{\partial \Delta Q}{\partial \delta} & \frac{\partial \Delta Q}{\partial |V|} \end{bmatrix} \begin{bmatrix} \Delta \delta \\ \Delta |V| \end{bmatrix} = \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}; \tag{3.23}$$

$$x^{i+1} = x^i + \Delta x^i, \tag{3.24}$$

де i – номер ітерації;

$J(x)$ – матриця Якобі;

$J_{ik} = \frac{\partial F_i(x)}{\partial x_k}$.

Потік постійного струму (DC loadflow) – спрощена модель потоку потужності, яка використовується для зменшення часу обчислення за рахунок наближень, що перетворюють нелінійну систему рівнянь потоку потужності до системи лінійних рівнянь та вливає на точність результатів. Проте цей метод може застосуватися для пошуку початкових значень для інших ітераційних методів. У моделі потоку постійного струму ігноруються рівняння балансу реактивної потужності, визначається лише потік реальної потужності [144, 145]. Модель може використовуватися у випадках, де потрібна приблизна оцінка реальної потужності потоку в системі. Існують також інші методи та модифікації для розв'язку системи рівнянь потоку потужності в електромережі [147].

Оптимальний потік потужності (Optimal Power Flow) – задача оптимізації, яка складається з цільової функції, множини обмежень та змінних. Множина

змінних і обмежень, а також цільова функція відрізняються залежно від типу задачі. Обчислення оптимального потоку потужності є важливим завданням для забезпечення ефективної роботи електромережі. Загальна форма оптимального потоку потужності визначається за формулами (3.25 – 3.28) [148]:

$$\min_x f(x)\,; \tag{3.25}$$

$$g(x) = 0; \tag{3.26}$$

$$h(x) \leq 0; \tag{3.27}$$

$$x^{min} \leq x \leq x^{max}, \tag{3.28}$$

де $f(x)$ – цільова функція мінімізації вартості генерації та втрат активної та реактивної потужності навантаження;

$g(x)$ – рівняння балансу потоку потужності вузла;

$h(x)$ – обмеження нерівності, що моделює ліміти потоку гілки;

x^{min}, x^{max} – граничні значення для параметрів компонентів системи [148].

Одним із найпоширеніших застосувань оптимального потоку потужності є мінімізація витрат на генерацію активної потужності (хоча цільова функція може складатися з кількох цілей), визначається за формулою (3.29) [149, 150]:

$$\min_{P_{G_i}} \sum_i c_{G_i} P_{G_i}\,, \tag{3.29}$$

де c_{G_i} – вартість виробництва енергії генератором G_i;

P_{G_i} – кількість потужності, яка виробляється генератором G_i.

Існують різні варіанти моделей оптимального потоку потужності, які використовуються для аналізу електромережі [151–154]. Моделі потоку потужності постійного струму є швидкими в обчисленні, але вони не враховують

потоки реактивної потужності, що може стати критичним у деяких ситуаціях. При аналізі моделей потоку потужності змінного струму для масштабних систем зі складними зв'язками обчислювальна складність зростає. Динамічні моделі можуть надавати детальну інформацію про каскади в системі, але для їх роботи необхідна множина параметрів, що описують характеристики системи, що може бути вузьким місцем через обмеженість даних, крім того, динамічні моделі є дорогими в обчисленні, тому можуть бути непрактичними при роботі з складними мережами. У результаті для аналізу каскадних збоїв в електромережах виникає необхідність у методах, що забезпечують розв'язок з прийнятною похибкою для різних сценаріїв і повинні задовольняти такі вимоги: аналіз мереж з великою розмірністю, гнучкість та швидкість в обчисленні.

3.2. Створення сценаріїв роботи електромережі при виникненні збоїв

Дані про роботу електромереж є закритими, тому виникає задача для генерації синтетичних даних, які будуть наближеними до реальних. Для симуляції сценаріїв роботи електромережі, зокрема каскадних збоїв та їх розвиток, використовуються фізичні моделі та ймовірнісні моделі, які мають свої особливості та обмеження [155]. Генерація сценаріїв для різних мереж складається з вибірки випадкових рівномірних або нормальних розподілів для зміни параметрів системи [156]:

– значення активного та реактивного опору лінії вибрані в межах 80% і 120% від їхніх початкових значень (використовується рівномірний розподіл);

– величина напруги для генераторів встановлюється в інтервалі [1,00; 1,05] у відносних одиницях (використовується рівномірний розподіл);

– початкові активні потужності для генераторів змінюються (використовується нормальний розподіл), де середнє значення відповідає початковій активній потужності, а стандартне відхилення становить 10% від неї;

– активна потужність та реактивна потужність для шин із навантаженням змінюються випадковим чином (використовується нормальний розподіл), де

середнє значення відповідає початковій потужності, а стандартне відхилення становить 10% від неї відповідно для кожної.

Енергосистеми не розраховані на численні непередбачувані ситуації, тому оператори електромереж повинні розуміти наслідки (сценарії розвитку збоїв) та підходи до стабілізації роботи мережі. У зв'язку з цим виконується аналіз роботи системи при різних умовах для виявлення потенційно-небезпечних випадків, щоб уникнути втрат у функціонуванні мережі:

– перевантажені гілки;

– шини з підвищеною або зниженою напругою;

– виникнення островів: система розділяється на дві або більше підсистем і кожна з них потенційно може мати проблеми з балансом потужності.

Для моделювання та аналізу сценаріїв каскадних збоїв в електромережі може використовуватися каскадна модель збоїв на основі потоку потужності змінного струму, яка складається з 12 кроків [156]:

– ініціювати параметри системи (початковий стан);

– ініціювати збій k ліній/трансформаторів;

– ідентифікувати острів (нова топологія системи);

– визначити відхилення частоти;

– застосувати UFLS (under-frequency load shedding) для підтримки стабільності енергосистеми;

– відновити баланс потужності;

– розрахунок потоку потужності (перевірка стабільності напруги);

– застосувати UVLS (under-voltage load shedding) для підтримки напруги;

– оцінити острова (кроки 4–8 застосовуються до всіх островів);

– перевірити перевищення лімітів в гілках;

– перевірити критерії припинення симуляції;

– сформувати вихідні результати.

Для аналізу стійкості електромереж у роботі використовується модель каскадних відмов змінного струму, яка включає динамічні явища, механізми захисту в статичному представленні, показує як механізми захисту працюють під

час розвитку каскадних збоїв [140]. Модель може працювати з великими системами, з різними параметрами та моделювати сценарії розвитку каскадів. Формування сценаріїв про розвиток збоїв в електромережі здійснюється на основі моделі [140]: у кожному кроці визначаються параметри та стан компонентів електромережі, формується послідовність кроків, що зберігаються у файлі для аналізу.

3.3. Вибір нейронних мереж для аналізу роботи електромереж

Нейронні мережі – моделі машинного навчання, які складаються із взаємопов'язаних нейронів, використовуються для виявлення складних закономірностей в даних та вирішення різних задач в багатьох предметних галузях (оптимізація процесів, підвищення ефективності та прийняття рішень). Приклад нейрону зображено на рисунку 3.2.

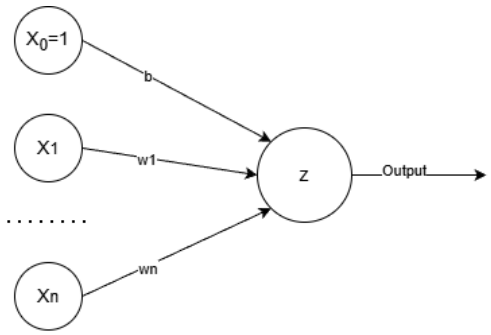

Рисунок 3.2 – Структура нейрону в нейронній мережі

Для обчислення виходу нейрону використовується зважена сума N входів $(x_1, \dots, x_n)$, які є зваженими за вагами з'єднань $(w_1, \dots, w_n)$ від входів нейрону, до якої додається зміщення (bias, b) та застосовується функція активації [157].

Результат лінійного перетворення (z) визначається за формулою (3.30):

$$z = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n + b. \qquad (3.30)$$

Функція активації (f) визначається за формулою (3.31) та відіграє важливу роль: додається нелінійність, що дозволяє мережі вивчати складні шаблони:

$$Output = f(z). \tag{3.31}$$

Нейронна мережа складається із множини нейронів, що утворюються шари та зв'язки між ними:

1. Вхідний шар (Input Layer). Вхідні дані передаються в нейронну мережу через вхідний шар, де ознаки представляється як нейрони у шарі ($A^{[0]} = X$).

2. Приховані шари (Hidden Layers). Шари між вхідним та вихідним шарами виконують перетворення та вивчення вхідних даних для вирішення поставленого завдання. Наприклад, для прихованого шару 1 трансформація визначається за формулою (3.32):

$$A^{[1]} = f\left(W^{[1]}X + b^{[1]}\right), \tag{3.32}$$

де $W^{[1]}$ – матриця ваг;

X – вхідний вектор;

$b^{[1]}$ – вектор зміщення;

f – функція активації.

Для прихованого шару 2 трансформація визначається за формулою (3.33):

$$A^{[2]} = f\left(W^{[2]}A^{[1]} + b^{[2]}\right). \tag{3.33}$$

3. Вихідний шар (Output Layer). Наприклад, в шарі обчислюються дані, які необхідні для вирішення поставленої задачі (регресії, класифікації тощо), визначається за формулою (3.34):

$$\bar{Y} = A^{[3]} = f\left(W^{[3]}A^{[2]} + b^{[3]}\right). \tag{3.34}$$

Приклад нейронної мережі зображено на рисунку 3.3, повне рівняння для прямого поширення (forward propagation) визначається за формулою (3.35):

$$\bar{Y} = f(f(f(XW^{[1]} + b^{[1]})W^{[2]} + b^{[2]})W^{[3]} + b^{[3]}). \quad (3.35)$$

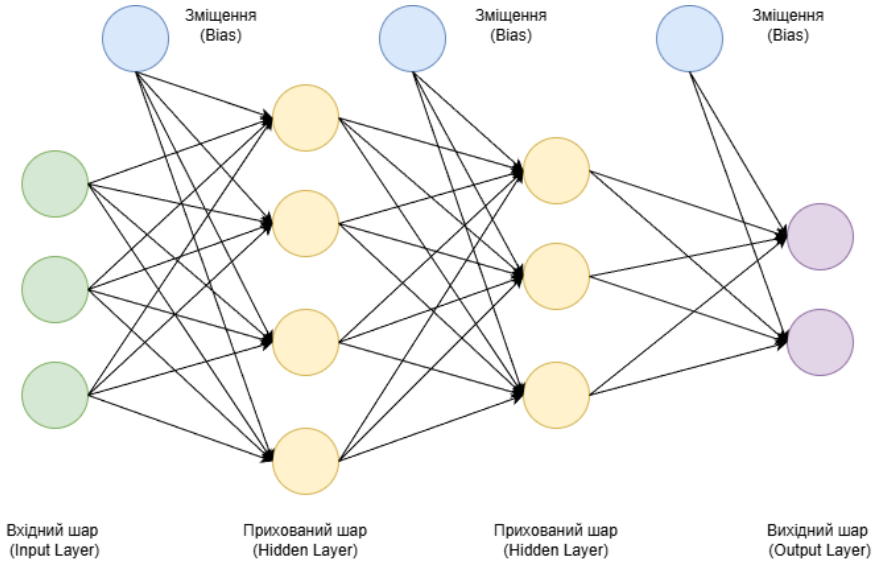

Рисунок 3.3 – Приклад нейронної мережі

При навчанні нейронної мережі використовується зворотне поширення (Backpropagation) [157]:

– до вихідних даних нейронної мережі застосовується функція втрат (L) для визначення різниці між фактичними та прогнозованим даними;

– використання градієнтного спуску: обчислюється градієнт похибки відносно вагових коефіцієнтів W та зміщень b, використовується ланцюгове правило [157] для обчислення похідної композиції двох і більше функцій. Наприклад, обчислення похідної функції $y = y(x)$, де $x = x(t)$ виглядає у формулі (3.36):

$$\frac{dy}{dt} = \frac{dy}{dx} * \frac{dx}{dt}. \quad (3.36)$$

– після обчислення градієнтів відбувається оновлення вагових коефіцієнтів W та зміщень b, які визначені за формулами (3.37), (3.38):

$$W_{new}^{[n]} = W_{old}^{[n]} - \alpha \frac{dL}{dW^{[n]}}; \tag{3.37}$$

$$b_{new}^{[n]} = b_{old}^{[n]} - \alpha \frac{dL}{db^{[n]}}. \tag{3.38}$$

Процес прямого та зворотного поширення повторюється повторюються, поки не буде досягнуто цільової оптимізації (з кожною ітерацією ваги та зміщення мережі наближаються до значень, які мінімізують похибку) [157].

При проєктуванні нейронної мережі вибір функції активації відіграє важливу роль, оскільки впливає на продуктивність моделі: функції активації мають різні властивості (область визначення, обчислювальна складність тощо), які можуть бути перевагами або недоліками в залежності від задачі [158]. Поширеними функціями активації є:

– Logistic Sigmoid: вихідні дані в діапазоні (0; 1), використовується в задачах бінарної класифікації, в задачах прогнозування ймовірностей – вихідні дані інтерпретується як ймовірність настання події, обчислюється за формулою (3.39):

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{3.39}$$

– Tanh: вихідні дані в діапазоні (-1; 1), використовується для прогнозування часових рядів, обчислюється за формулою (3.40):

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{3.40}$$

– ReLU: вихідні дані в діапазоні [0; +∞), простота, ефективність обчислення, обчислюється за формулою (3.41):

$$f(x) = \max(0, x)\,. \tag{3.41}$$

– Softmax обчислюється за формулою (3.42), використовується на вихідному рівні задач класифікації з кількома класами – ймовірність належності до певного класу:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}. \tag{3.42}$$

Існує багато інших функцій активації і їхнє використання залежить від задачі та архітектури нейронної мережі.

Функції втрат (Loss functions) відіграють важливу роль в навчанні нейронних мереж та використовуються у різних завданнях [159]. В завданнях регресії поширеними є функції втрат:

– середньоквадратична похибка (Mean Squared Error (MSE) Loss) обчислюється за формулою (3.43): середнє значення квадратів різниць між очікуваними та отриманими вихідними даними, є простою для розуміння, але чутлива до викидів:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2\,, \tag{3.43}$$

де n – кількість спостережень в наборі даних;

y_i – фактичне значення i-го спостереження;

$\overline{y_i}$ – прогнозоване значення i-го спостереження.

– середня абсолютна похибка (Mean Absolute Error (MAE) Loss) обчислюється за формулою (3.44): середнє значення абсолютних різниць між

очікуваними та отриманими вихідними даними. Функція менш чутлива до викидів порівняно з MSE, часто використовується при великій кількості викидів в даних. Недоліком функції є не диференційованість в нулі, що може бути проблемою для алгоритмів оптимізації:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \overline{y_i}|\,. \tag{3.44}$$

– похибка Губера (Huber Loss) поєднує переваги MSE та MAE, обчислюється за формулою (3.45):

$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \overline{y_i})^2, \text{де } |y_i - \overline{y_i}| \le \delta \\ \delta|y_i - \overline{y_i}| - \frac{1}{2}\delta^2, \text{де } |y_i - \overline{y_i}| > \delta \end{cases}, \tag{3.45}$$

де δ – це гіперпараметр, який визначає поріг, при якому функція втрат змінюється з квадратичної (середньоквадратична похибка) на лінійну (середня абсолютна похибка).

У завданнях класифікації поширеними є функції втрат:

– Binary Cross-Entropy Loss (Log Loss) обчислюється за формулою (3.46), використовується для задач бінарної класифікації:

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\overline{y_i}) + (1 - y_i)\log(1 - \overline{y_i})]\,, \tag{3.46}$$

де n – кількість спостережень в наборі даних;

y_i – фактичне значення (0 або 1) для класу i-го спостереження;

$\overline{y_i}$ – прогнозована ймовірність i-го спостереження.

– Categorical Cross-Entropy Loss обчислюється за формулою (3.47), використовується в задачах класифікації із кількома класами (більше двох), функція активації у вихідному шарі SoftMax:

$$\text{Categorical Cross-Entropy Loss} = -\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log\left(\overline{y_{ij}}\right), \tag{3.47}$$

де n – кількість спостережень в наборі даних;

k – кількість класів;

y_{ij} – фактичне значення (0 або 1) класу j в i-му спостереженні;

$\overline{y_{ij}}$ – прогнозована ймовірність класу j в i-му спостереженні.

Графові нейронні мережі використовуються в різних галузях. Вони ефективно моделюють зв'язки та залежності в даних на основі графів, які складаються з вузлів та ребер, що фіксують зв'язки між ними. Завдання на основі графових даних можуть бути класифіковані в наступні категорії: node level task (завдання рівня вузла), edge level task (завдання рівня ребра), graph level task (завдання рівня графа) [160].

Завдання рівня вузла діляться на чотири категорії:

– класифікація вузлів (node classification) – визначення класів вузлів на основі їхніх ознак та зв'язків;

– регресія вузлів (node regression) – прогнозування числових значень для вузлів;

– кластеризація вузлів (node clustering) – поділ вузлів на класи та групування вузлів з подібними характеристиками;

– виявлення аномалій вузлів (node anomaly detection) – ідентифікація вузлів графа, які відрізняються від визначених характеристик.

Завдання рівня ребра поділяються на категорії:

– прогнозування зв’язків (link prediction) – визначення ймовірності створення ребра між двома вузлами на основі структури графа та характеристик вузлів;

– класифікація ребер (edge classification) – прогнозування класу для ребер в графі (наприклад, категорія або статус ребра);

– регресія ребер (edge regression) – завдання прогнозування числових значень для ребер у графі;

– кластеризація ребер (edge clustering) – групування подібних ребер у графі в кластери на основі їхніх характеристик4

– виявлення аномальних ребер (edge outlier detection) – визначення ребер у графі, які відрізняються від характеристик у графі (наприклад, незвичайні ознаки або зв’язки).

Завдання рівня графа визначаються на рівні цілих графів (регресія, класифікація, висновки для графу або підграфу, а не окремих вузлів чи ребер). Графові нейронні мережі використовуються для агрегації інформації з вузлів і ребер для створення векторного представлення (embedding vector) всього графа, що може використовуватися для побудови моделей для різних завдань.

Основою графових нейронних мереж (GNN) є передача повідомлень (message-passing neural network) – процес ітеративної передачі повідомлень між вузлами та агрегації інформації від їхніх сусідів, що дозволяє моделі розуміти комплексні зв'язки та залежності в даних на основі графів [161]. Задано неорієнтований граф $G = (V, E)$, де V – множина вузлів, E – множина ребер, кожен вузол $v \in V$ має відповідний вектор ознак x_V, мета GNN – вивчити представлення h_V для кожного вузла v, яке збирає інформацію про сусідні вузли.

Передача повідомлень складається з наступних кроків:

1. Агрегація повідомлень (Aggregation of messages). Кожен вузол використовує функцію для створення повідомлення для кожного сусіднього вузла. Для кожного вузла v збирається інформація від його сусідніх вузлів $N(v)$. Агреговане повідомлення m_v для вузла v обчислюється як агрегування інформації від сусідніх вузлів за формулою (3.48):

$$m_v = aggregate(\{h_u, u \in N(v)\}). \quad (3.48)$$

Поширеними функціями агрегування є:

– агрегація сум (sum aggregation);

– агрегація середніх значень (mean aggregation);

– max pooling;

– агрегація на основі уваги (attention-based aggregation).

2. Оновлення (Update). Оновлення представлення вузла h_V на основі агрегованого повідомлення m_v та поточного представлення вузла h_V виконується за допомогою функції update – шар нейронної мережі з параметрами, які можна навчати, визначається за формулою (3.49):

$$h'_v = update(h_v, m_v). \quad (3.49)$$

Цей процес є ітеративним в графовій нейронній мережі. Вибір функції $aggregate$, $update$ та кількості ітерацій залежить від архітектури GNN. У загальному вигляді процес визначається за формулою (3.50):

$$h_v^l = update^{(l)}\left(h_v^{l-1}, aggregate^{(l)}(\{h_u^{l-1}, u \in N(v)\})\right), \quad (3.50)$$

де $l \in \{1, \dots, L\}$ – шари нейронної мережі.

Для завдань на рівні графів використовується інваріантна до перестановок функція readout, яка агрегує представлення вузлів в один вектор. Він може бути використаний для подальшої класифікації або регресії, визначається за формулою (3.51):

$$h_G = readout(\{h_v^L, v \in V\}). \quad (3.51)$$

Графові нейронні мережі (GNN) з механізмом передачі повідомлень показують кращі результати на наборі даних [162] з графовою структурою чим використання моделі з лінійними шарами (MLP). У таблиці 3.2 наведено результати використання моделі нейронної мережі, яка використовує різні шари (графові та лінійні) в завданні класифікації вузлів. Модель GNN демонструє кращі результати, ніж модель MLP, оскільки GNN використовує зв'язки між вузлами (структуру графа), а MLP оброблює ознаки вузлів окремо, що призводить до втрати топологічної інформації.

Таблиця 3.2 – Точність моделей на тестових даних

Dataset	MLP Accuracy	GNN Accuracy
CITESEER	50,10%	68,30%
CORA	53,30%	79,70%
PUBMED	72,30%	78,70%

Існують різні модифікації та архітектури графових нейронних мереж, кожна з яких має особливості та застосування [163, 164].

Графові згорткові нейронні мережі (Graph Convolutional Networks) можуть застосовуватися для задач на рівні вузлів, ребер та графів. Операція згортки графа в шарі l визначається за формулою (3.52) [165]:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \quad (3.52)$$

де $\tilde{A} = A + I_N$ – матриця суміжності неорієнтованого графа G з self-connections;

I_N – одинична матриця;

$\widetilde{D}$ – діагональна матриця степенів $\tilde{A}$;

$W^{(l)}$ – матриця ваг шару;

$\sigma(\cdot)$ – нелінійна функція активації;

$H^{(l)} \in R^{N\times D}$ – матриця представлень вузлів у l-му шарі;

$H^{(0)} = X$ (вхідні ознаки).

Графові нейронні мережі із механізмом уваги (Graph Attention Networks) використовують динамічні вагові коефіцієнти, які враховують ознаки вузлів, їхню важливість (вузли по-різному впливають на сусідні вузли) та механізм самоуваги (self-attention). Цей підхід є гнучкішим, ніж GCN, де використовуються статичні коефіцієнти нормалізації, враховуються лише ступені вузлів. Коефіцієнти уваги (attention coefficients) обчислюються для всіх сусідів вузла. Коефіцієнт уваги між вузлами i та j визначається за формулою (3.53) [166]:

$$a_{ij} = \frac{\exp\left(LeakyReLU(\vec{a}^T[W\vec{h}_i||W\vec{h}_j])\right)}{\sum_{k\in N_i}\exp\left(LeakyReLU(\vec{a}^T[W\vec{h}_i||W\vec{h}_k])\right)}, \quad (3.53)$$

де $\cdot^T$ – транспонування;

$||$ – конкатенація.

Оновлене представлення вузла i визначається за формулою (3.54)

$$h_i' = \sigma\left(\sum_{j\in N_i} a_{ij} * h_j\right). \quad (3.54)$$

Самоувага не дуже стабільна, тому використовується багатоголова увага (multi-head attention) [167, 168]. Кроки (лінійне перетворення, функція активації, нормалізація Softmax) повторюються кілька разів, де кожен екземпляр (k – індекс головки уваги) створює представлення h_i^k, які потім формують вихідний результат за допомогою функцій [168]:

– середнє значення (average), використовується у вихідному шарі, обчислюється за формулою (3.55):

$$h_i = \frac{1}{n}\sum_{k=1}^{n} h_i^k . \quad (3.55)$$

– конкатенація (concatenation), використовується у прихованому шарі, обчислюється за формулою (3.56):

$$h_i = ||_{k=1}^{n} h_i^k. \quad (3.56)$$

Шари графової нейронної (GCN, GAT) можуть об'єднуватися (на вхід шару передаються представлення з попереднього шару) для вивчення складних зв'язків та даних графу. При великому збільшенні глибини графової нейронної мережі виникає надмірне згладжування (oversmoothing) [169], яке негативно впливає на продуктивність моделі.

Для обробки великих графів використовується архітектура GraphSAGE, яка має кращу масштабованість та усуває деякі обмеження GCN, GAT. Основними компонентами GraphSAGE є [170]:

– вибірка сусідів (neighbor sampling): для визначених вузлів використовується підмножина сусідніх вузлів, що може потенційно призвести до втрати важливої інформації та зниження продуктивності моделі;

– агрегація (aggregation): можуть використовуватися інші агрегатори: Mean aggregator, LSTM aggregator, Pooling aggregator.

При аналізі зміни характеристик мережі в часі використовують гібридні моделі, які складаються з блоків, що вивчають структурні та часові характеристики історичних даних для прогнозування майбутньої поведінки мережі [171–176]. Графи з часовою складовою поділяються на два типи:

– статичні графи (static graphs) – структура графу не змінюється із часом, але ознаки змінюються;

– динамічні графи (dynamic graphs) – структура графу змінюється із часом (кількість вузлів, ребер), ознаки змінюються.

Під час роботи з даними, які містять часову складову (послідовні дані у часі) використовується тип рекурентної нейронної мережі – Long Short-Term Memory (LSTM). Рекурентні нейронні мережі мають проблеми при обробці довгих послідовностей, при навчанні може виникати проблема зниклого градієнта (Vanishing Gradient) або проблема вибухового градієнта (Exploding Gradient), що призводить до нестабільності моделі (мережа може «забути» важливу інформацію в довгій послідовності). LSTM була розроблена, щоб усунути обмеження рекурентної нейронної мережі та складається з компонентів [177, 178]:

– Forget Gate визначається за формулою (3.57): інформація, яка більше не є корисною видаляється:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big), \tag{3.57}$$

де W_f – вагова матриця;

$[h_{t-1}, x_t]$ – об'єднання поточного вхідного даних та попереднього прихованого стану;

b_f – зміщення;

σ – функція активації сигмоїда.

– Input gate визначається за формулами (3.58 – 3.60): додавання корисної інформації до стану:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i); \tag{3.58}$$

$$\hat{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c); \tag{3.59}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t), \tag{3.60}$$

де $\odot$ – поелементне множення.

– Output gate визначається за формулою (3.61): визначення корисної інформації для виходу:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o). \quad (3.61)$$

Навчання LSTM моделі є повільнішим та вимагає більше пам'яті, ніж RNN, але має кращу обробку довгих послідовностей.

3.4. Розробка методу визначення подібності сценаріїв каскадних ефектів в електромережах на основі графових нейронних мереж

При формуванні представлення про роботу електромережі в сценарії потрібно враховувати взаємодію компонентів та їхні статуси. Компоненти електромережі можуть бути виведені із ладу чи відключені в процесі розвитку каскадного ефекту, тому важливо враховувати ці особливості при обробці даних графу. У розроблених шарах використовується фреймворк передачі повідомлень, використовується інформація про з'єднання вершин, ребер та їхні статуси, що є корисним при формуванні представлення про вершини та ребра графу електромережі.

3.4.1. Створення даних про роботу електромережі в сценаріях

Для створення каскадних ефектів в електромережі використовується модель потоку потужності та захисні механізми [140], які використовуються при виконанні певних умов компонентами системи. Кожен сценарій складається з множини похідних сценаріїв, які відображають зміни компонентів системи в момент часу. Стан роботи електромережі в момент часу t визначається за допомогою орієнтованого графу $G_t = (V_t, E_t)$, де V_t – шини, $E_t \in V \times V$ – гілки, які мають відповідні ознаки, де $X_t \in R^{|V| \times F_V}$ – матриця ознак шин, F_V – кількість ознак шини; $B_t \in R^{|E| \times F_E}$ – матриця ознак гілок, F_E – кількість ознак гілки. Також

до компонентів електромережі додаються метадані, які використовуються для розуміння розвитку сценарію: номер сценарію, номер похідного сценарію, номер підмережі, номер батьківської мережі, яка була поділена на підмережі.

Електромережа складається з компонентів, що мають визначені ознаки:

а) шина (bus):

1) чиста активна потужність (net real power) (P_i) на шині i обчислюється за формулою (3.62):

$$P_i = P_{gi} - P_{di}, \tag{3.62}$$

де P_{gi} – реальна потужність, що генерується на шині i;

P_{di} – реальна потужність, що споживається навантаженнями.

У залежності від значень P_{gi} та P_{di} чиста активна потужність P_i може набувати значень:

– позитивне значення: шина є джерелом енергії, генерується потужності більше, ніж споживається;

– від'ємне значення: шина є навантаженням, споживається потужності більше, ніж генерується;

– нульове значення: збалансована потужність між генерацією та споживанням на шині.

2) чиста реактивна потужність (net reactive power) (Q_i) на шині i обчислюється за формулою (3.63):

$$Q_i = Q_{gi} - Q_{di}, \tag{3.63}$$

де Q_{gi} – реактивна потужність, що генерується на шині i;

Q_{di} – реактивна потужність, що споживається навантаженнями.

Значення чистої реактивної потужності Q_i можливо інтерпретувати як:

– позитивне значення: шина споживає більше реактивної потужності, ніж надходить, характерно для шини навантаження (load bus);

– від’ємне значення: у шину надходить більше реактивної потужності, ніж споживається, що характерно для генератора;

– нульове значення: баланс між реактивною потужністю, що надходить від джерел (наприклад, генератори), і реактивною потужністю, що споживається навантаженнями.

3) величина напруги (Vm_i) на шині i, що визначається на основі потоку потужності;

4) кут напруги (Va_i) на шині i, що визначається на основі потоку потужності.

б) гілка (branch), що моделюються у вигляді орієнтованого графу для моделі потоку гілок, щоб точно відобразити напрямок потоку потужності від однієї шини до іншої в певному напрямку, розраховувати величини потоку потужності. Для гілки, яка з’єднує шини j та i, визначаються ознаки:

1) Pf: реальна потужність, яка походить від шини j на початку гілки;

2) Pt: реальна потужність, яка надходить на шину i в кінці гілки;

3) Qf: реактивна потужність, яка походить від шини j на початку гілки;

4) Qt: реактивна потужність, яка надходить на шину i в кінці гілки.

Приклад електромережі, що складається із 3 шин та 2 гілок із ознаками зображено на рисунку 3.4.

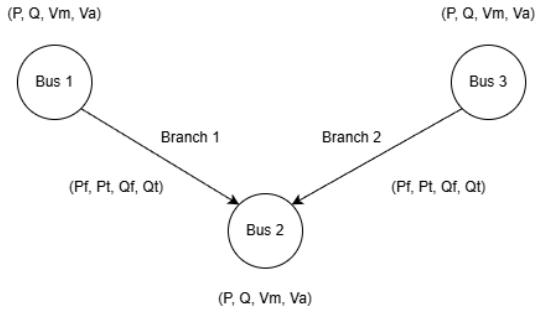

Рисунок 3.4 – Приклад графу електромережі в момент часу

3.4.2. Розробка шару для формування представлення про вузли графу

Вхідні дані шару для формування представлення про вершини графу зображено на рисунку 3.5 [179, 180]:

– ознаки вершин графу x (кожна шина має ознаки: активна потужність, реактивна потужність, величина напруги, кут напруги);

– ознаки ребер графу $edge\_attr$ (реальна потужність, яка походить від шини j на початку гілки; реальна потужність, яка надходить на шину i в кінці гілки; реактивна потужність, яка походить від шини j на початку гілки; реактивна потужність, яка надходить на шину i в кінці гілки);

– структура графу $edge\_index$;

– маска $node\_mask$, яка визначає активні та неактивні вершини графу;

– маска $edge\_mask$, яка визначає активні та неактивні ребра графу.

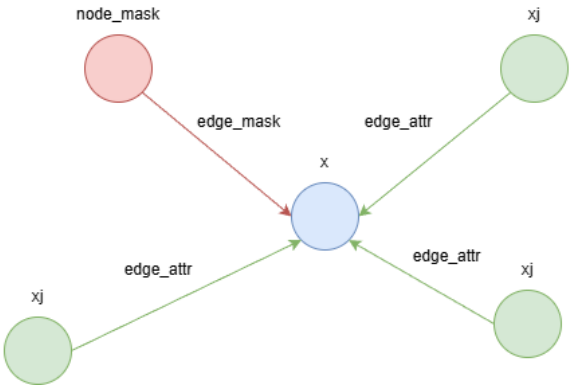

Рисунок 3.5 – Фрагмент графу електромережі мережі

Процес обробки даних в шарі складається із наступних етапів [179, 180]:

1. Формування повідомлення:

– застосувати маску до ознак ребер графу, визначається за формулою (3.64):

$$edge\_attr = edge\_attr * edge\_mask; \quad (3.64)$$

– застосувати маску до вихідних ознак вершин графу, визначається за формулою (3.65):

$$x_j = x_j * node\_mask; \quad (3.65)$$

– об’єднати ознаки вихідної вершини x_j та ребра $edge\_attr$, визначається за формулою (3.66):

$$combined\_features = x_j || edge\_attr; \quad (3.66)$$

– застосувати лінійне перетворення, яке трансформує розмірність об’єднаних ознак до вхідної розмірності ознак вершин графу, визначається за формулою (3.67):

$$message = mlp\_message(combined\_features); \quad (3.67)$$

– сформувати повідомлення для агрегації.

2. Агрегація повідомлень:

– застосувати маску для повідомлень, які отримують цільові вершини x_i, визначається за формулою (3.68):

$$input\_message = message * node\_mask[x_i]; \quad (3.68)$$

– застосувати функцію агрегації «сума», що підсумовує повідомлення для кожного цільового вузла, визначається за формулою (3.69):

$$aggregated\_message = sum(input\_message, x_i); \quad (3.69)$$

3. Оновлення представлення вершини:

– поєднати агреговане повідомлення з оригінальними ознаками вершини, визначається за формулою (3.70):

$$node\_embedding = x + aggregated\_message; \quad (3.70)$$

– застосувати лінійне перетворення, яке трансформує розмірність отриманого представлення до визначеної вихідної розмірності, визначається за формулою (3.71):

$$final\_node\_embedding = mlp\_embedding(node\_embedding). \quad (3.71)$$

3.4.3. Розробка шару для формування представлення про ребра графу

Вхідні дані шару для формування представлення про ребра графу [179, 180]:

– ознаки вершин графу x;

– ознаки ребер графу $edge\_attr$;

– структура графу $edge\_index$;

– маска $node\_mask$, яка визначає активні та неактивні вершини графу;

– маска $edge\_mask$, яка визначає активні та неактивні ребра графу.

Процес обробки даних в шарі складається із наступних етапів [179, 180]:

1. Формування повідомлення:

– застосувати маску до ознак вершин графу, що визначається за формулою (3.72):

$$x = x * node\_mask; \quad (3.72)$$

– застосувати маску до ознак ребер графу, що визначається за формулою (3.73):

$$edge\_attr = edge\_attr * edge\_mask; \quad (3.73)$$

– об’єднати ознаки вихідної вершини x_j та цільової вершини x_i, що визначається за формулою (3.74):

$$combined\_features = x_j || x_i; \quad (3.74)$$

– застосувати лінійне перетворення, яке трансформує розмірність об’єднаних ознак до вхідної розмірності ознак ребер графу, що визначається за формулою (3.75):

$$message = mlp\_message(combined\_features); \quad (3.75)$$

2. Оновлення представлення ребра:

– поєднати повідомлення з оригінальними ознаками ребра, що визначається за формулою (3.76):

$$edge\_embedding = edge\_attr + message; \quad (3.76)$$

– застосувати лінійне перетворення, яке трансформує розмірність отриманого представлення до визначеної вихідної розмірності, визначається за формулою (3.77):

$$final\_edge\_embedding = mlp\_embedding(edge\_embedding). \quad (3.77)$$

3.4.4. Удосконалення моделі автоенкодера для створення представлень про крок роботи електромережі в сценаріях

Формування представлення про електромережу в кроці сценарію відбувається за допомогою моделі автоенкодера, яка зображена на рисунку 3.6.

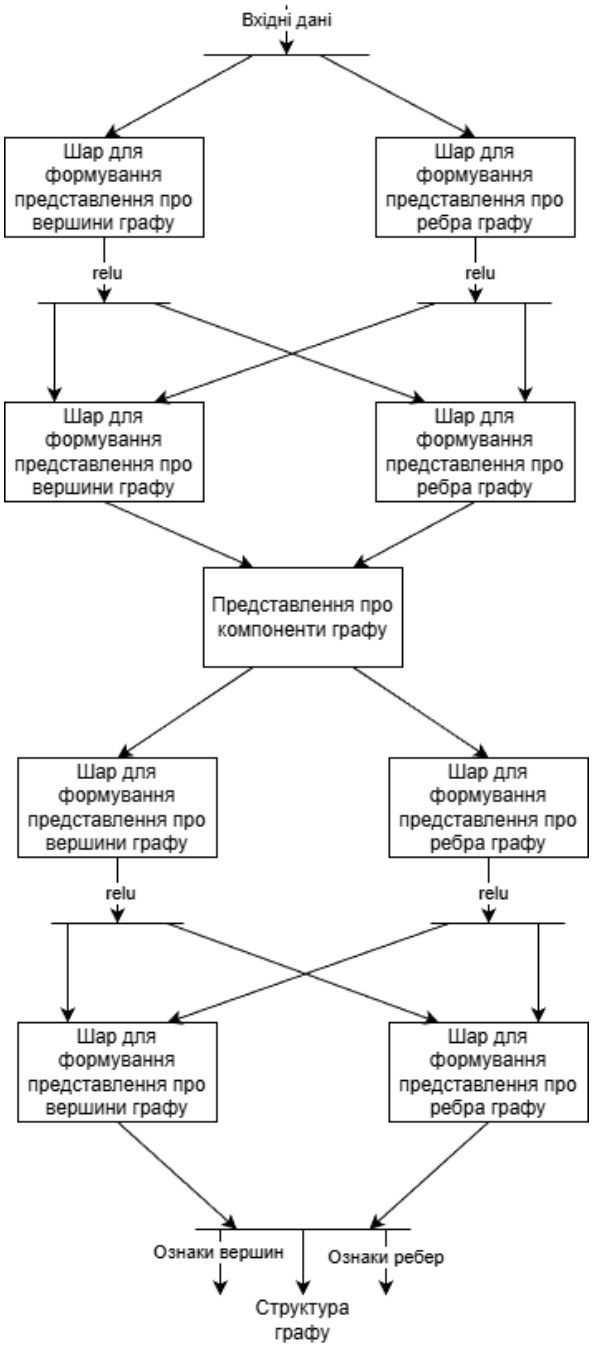

Рисунок 3.6 – Структура автоенкодера для формування представлення про вершини та ребра електромережі в кроці сценарію [180]

Модель автоенкодера складається з таких компонентів [179, 180]:

– енкодер (encoder) – блок, де формується представлення про компоненти електромережі в певний крок сценарію;

– шари для формування представлення про вершини графу електромережі;

– шари для формування представлення про ребра графу електромережі;

– декодер (decoder) – блок, де відбувається реконструкція ознак вершин, ребер та структури графу електромережі в певний крок сценарію;

– шари для відтворення ознак вершин графу електромережі;

– шари для відтворення ознак ребер графу електромережі.

Графові шари використовуються для відтворення ознак та структури графу із представлення, яке формується в середині автоенкодера (embedding). Відтворення структури графу в декодері визначається за формулою (3.78) [179, 180]:

$$edges = sum(x_j * x_i * edge\_attr). \tag{3.78}$$

Під час тренування моделі використовується функція втрат (loss fucntion), яка складається з: функції втрат ознак вершин графу (mse_loss), функції втрат структури графу (binary_cross_entropy_with_logits), функції втрат ознак ребер графу (mse_loss) [179, 180], зміна помилки зображена на рисунку 3.7. Під час тестування роботи моделі використовуються наявні графові шари та розроблені шари. Для оцінки роботи моделі використовується значення помилки моделі (loss) та коефіцієнт детермінації R^2, обчислюється за формулою (3.79):

$$R^2 = 1 - \frac{\sum\left(Y_i - \widehat{Y}_i\right)^2}{\sum(Y_i - \bar{Y})^2}, \tag{3.79}$$

де Y_i – фактичне значення залежної змінної для i-го спостереження;

$\widehat{Y}_i$ – прогнозоване значення залежної змінної для i-го спостереження;

$\bar{Y}$ – середнє значення залежної змінної.

Значення R^2 лежить в діапазоні від 0 до 1 і інтерпретується наступним чином: якщо значення близьке до 1, то модель добре пояснює варіацію залежної змінної; якщо значення близьке до 0, то модель погано пояснює варіацію залежної змінної. Результати роботи моделей наведено в таблицях 3.3, 3.4, 3.5.

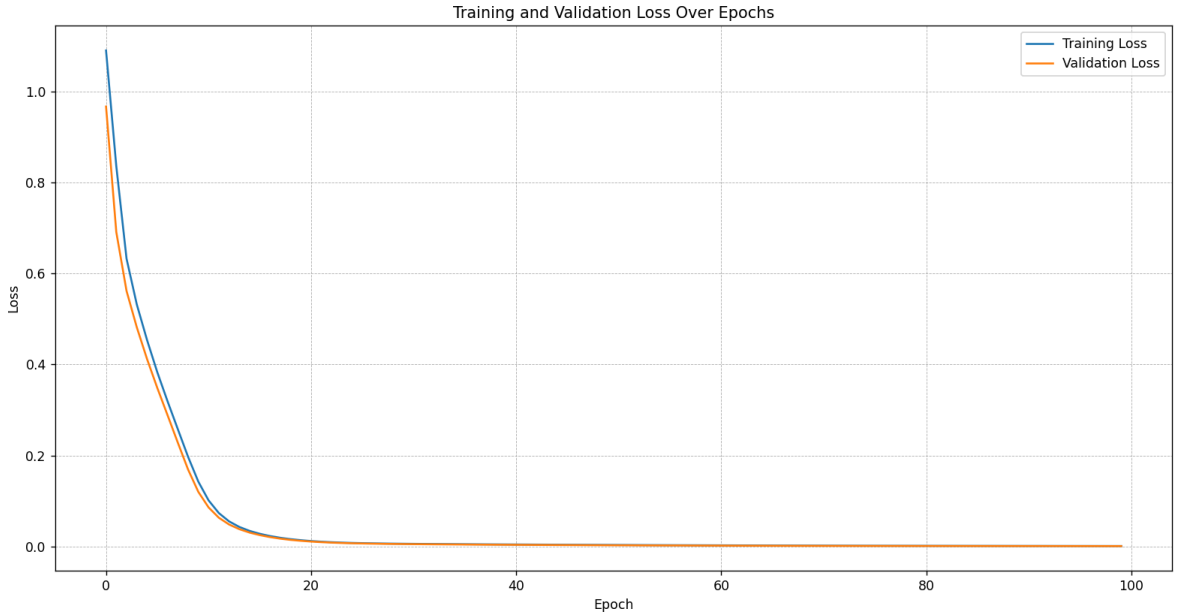

Рисунок 3.7 – Зміни значення функції втрат під час навчання моделі автоенкодера для формування представлення про крок сценарію

Таблиця 3.3 – Результати роботи моделі на електромережі case9 при значенні epoch = 100

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedOneStep	**0,0007**	**0,0006**	**0,0006**	**0,9932**	**0,9685**
GCN	0,0183	0,0189	0,0158	0,9043	0,4475
GAT	0,0153	0,0151	0,0149	0,9145	0,5476
GAT + Edges	0,0124	0,0122	0,0133	0,9207	0,6229
GATV2	0,0129	0,0134	0,0138	0,9263	0,6322
GATV2 + Edges	0,0118	0,0129	0,0103	0,9255	0,7703

Таблиця 3.4 – Результати роботи моделі на електромережі case30 при значенні epoch = 100

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedOneStep	**0,0017**	**0,0016**	**0,0020**	**0,9821**	**0,9212**
GCN	0,0163	0,0174	0,0150	0,8935	0,4712

Продовження таблиці 3.4

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
GAT	0,0131	0,0135	0,0123	0,9231	0,5925
GAT + Edges	0,0128	0,0122	0,0117	0,9146	0,6217
GATV2	0,0086	0,0087	0,0097	0,9405	0,6857
GATV2 + Edges	0,0091	0,0085	0,0089	0,9255	0,6534

Таблиця 3.5 – Результати роботи моделі на електромережі case94pi при значенні epoch = 200

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedOneStep	**0,0016**	**0,0020**	**0,0016**	**0,9878**	**0,9996**
GCN	0,0661	0,0664	0,0678	0,9025	0,6884
GAT	0,0516	0,0497	0,0446	0,8997	0,8208
GAT + Edges	0,0532	0,0570	0,0599	0,9377	0,7163
GATV2	0,0470	0,0489	0,0559	0,9213	0,7875
GATV2 + Edges	0,0500	0,0486	0,0500	0,9222	0,7680

Модель потенційно може бути покращена, при збільшенні кількості шарів, нейронів та зміні гіперпараметрів, але потрібно слідкувати за процесом навчання, щоб уникнути перетренування моделі.

3.4.5. Визначення подібності сценаріїв роботи електромереж

Для формування представлення про граф (електромережу) використовується механізм пулінгу (graph-level pooling), що застосовується до ознак графу, визначається за формулою (3.80):

$$scenario = global\_mean\_pool(nodes) * global\_mean\_pool(edges). \quad (3.80)$$

Global_mean_pool для вершин графу G_i обчислюється за формулою (3.81) [181]:

$$r_{G_i} = \frac{1}{|V_i|} \sum_{v \in V_i} x_v \,, \tag{3.81}$$

де V_i – множина вузлів у графі G_i;

$|V_i|$ – кількість вузлів у графі G_i;

x_v – вектор ознак вузла v у графі G_i.

Представлення графу використовується в аналітичних моделях порівняння сценаріїв за допомогою коефіцієнтів подібності. Коефіцієнти подібності, асоціативні правила застосовуються в різних сферах для оцінки подібності, визначення зв'язків між об'єктами [182–187]. Поширеними коефіцієнтами подібності є:

– коефіцієнт Жаккара, який вимірює подібність між множинами, обчислюється за формулою (3.82). Значення коефіцієнта знаходяться між 0 та 1: якщо значення дорівнює 0 – відсутність подібності; значення, що наближаються до 1 – зростання подібності; 1 – однакові множини даних [188]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}, \tag{3.82}$$

де $|A \cap B|$ – кількість спільних елементів в множинах A та B;

$|A \cup B|$ – загальна кількість елементів в множинах A та B.

– косинус подібності обчислюється за формулою (3.83): косинус кута між ненульовими векторами. При значенні кута 0 градусів косинус подібності дорівнює 1 – однаковий напрямок, максимальна подібність; при значенні кута 90 градусів косинус подібності дорівнює 0 – вектори ортогональні, відсутність подібності; діаметрально направлені вектори мають подібність –1 [189]:

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \quad (3.83)$$

де A_i та B_i – координати вектору A та B відповідно.

Важливо крім кута між векторами враховувати ще їхні величини, тому порівняння електромережі в кроці сценарію обчислюється за формулою (3.84):

$$similarity = cos(\theta) * \frac{\min\left(\sqrt{\sum_{i=1}^{n} A_i^2}, \sqrt{\sum_{i=1}^{n} B_i^2}\right)}{\max\left(\sqrt{\sum_{i=1}^{n} A_i^2}, \sqrt{\sum_{i=1}^{n} B_i^2}\right)}. \quad (3.84)$$

Можливі значення коефіцієнту подібності розділяються на інтервали:

– [0,0; 0,35] – низька подібність станів електромережі;

– (0,35; 0,5] – нижче-середнього подібність станів електромережі;

– (0,5; 0,75] – середня подібність станів електромережі;

– (0,75; 1,0] – висока подібність станів електромережі.

Визначення подібних станів електромережі порівняно з наявними сценаріями надає додаткову інформацію під час прийняття рішень: розвиток каскадного ефекту, характеристики компонентів [179, 180]. Дослідження роботи критичної інфраструктури в різних сценаріях відіграє важливу роль для прийняття відповідних рішень [190–195].

У сценарії 1, який зображено на рисунку 3.8, виведені із ладу шини: 2, 8 та гілки між шинами: 8–2, 8–9, 7–8, електромережа продовжує функціонувати.

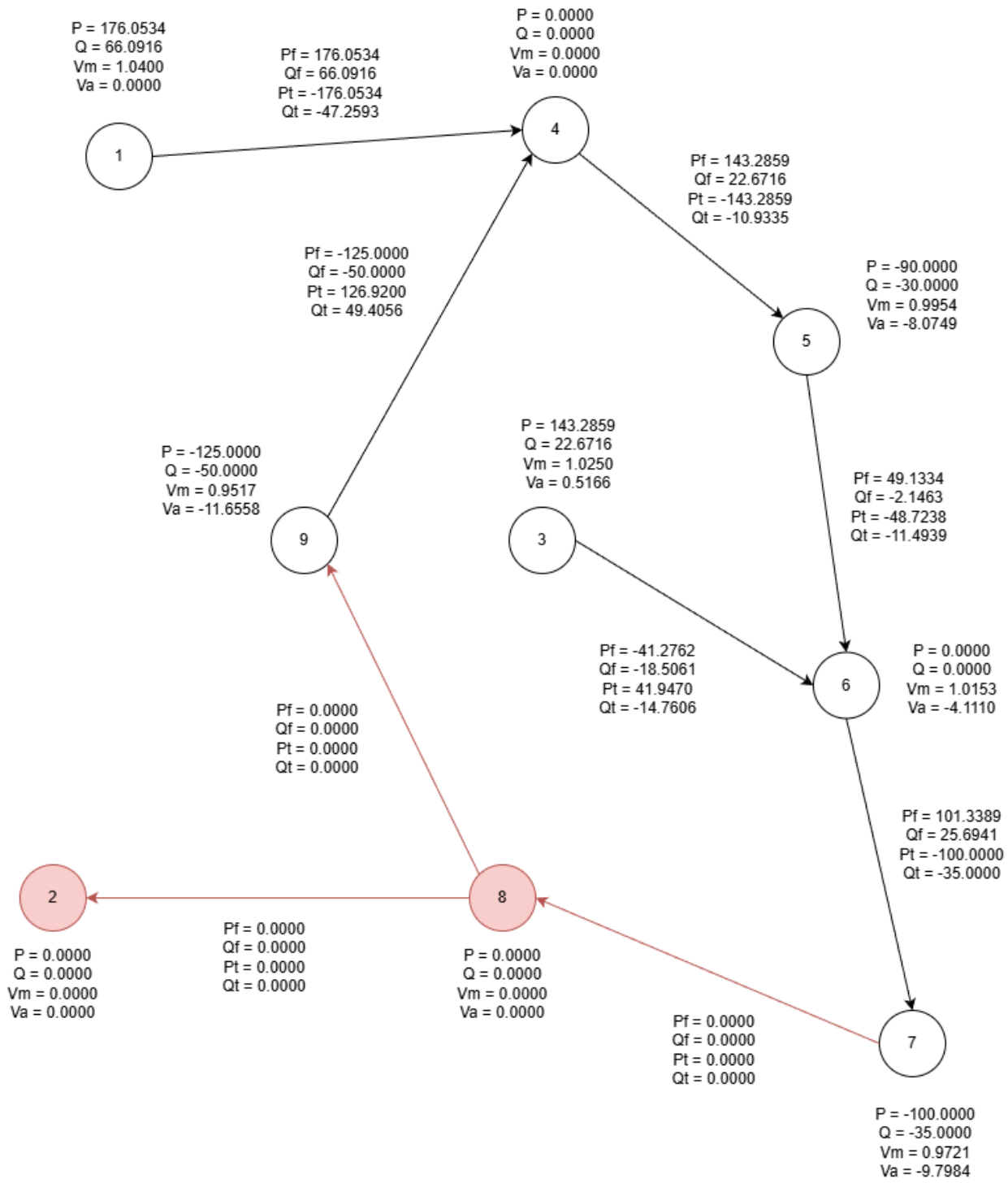

Рисунок 3.8 – Приклад стану електромережі в сценарії 1

У сценарії 2, який зображено на рисунку 3.9, виведені із ладу шини: 1, 2, 3, 4, 5, 6, 7, 8, 9 та гілки між шинами: 1–4, 4–5, 5–6, 3–6, 6–7, 7–8, 8–2, 8–9, 9–4, електромережа знеструмлена. Подібність сценаріїв 1 та 2 дорівнює 0,3050 – низька подібність станів електромережі.

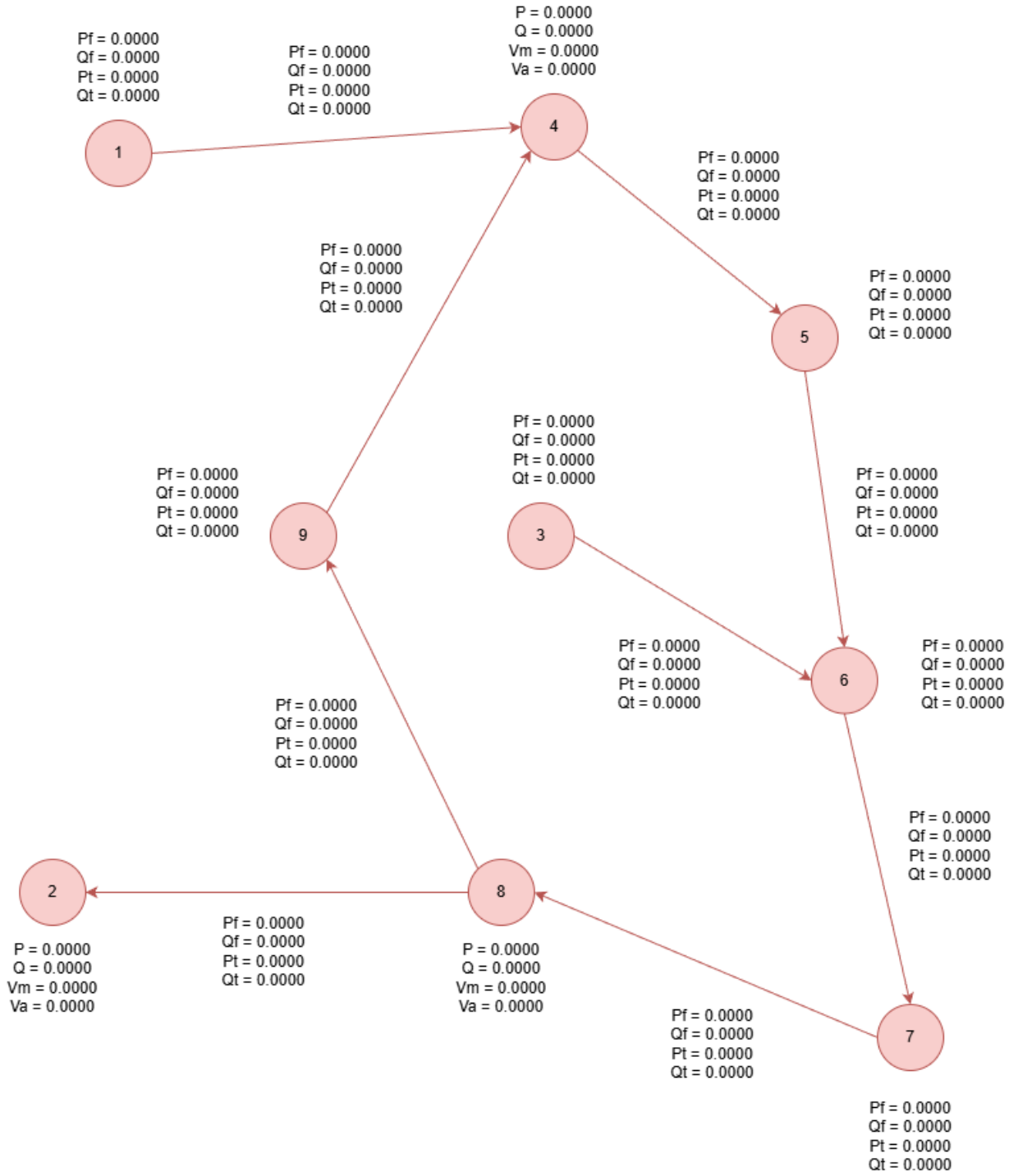

Рисунок 3.9 – Приклад стану електромережі в сценарії 2

У сценарії 3, який зображено на рисунку 3.10, виведені із ладу шини: 2, 9, 4 та гілки між шинами: 1–4, 9–4, 8–9, 8–2, 5–6, електромережа продовжує функціонувати. Подібність сценаріїв 1 та 3 дорівнює 0,7925 – висока подібність станів електромережі.

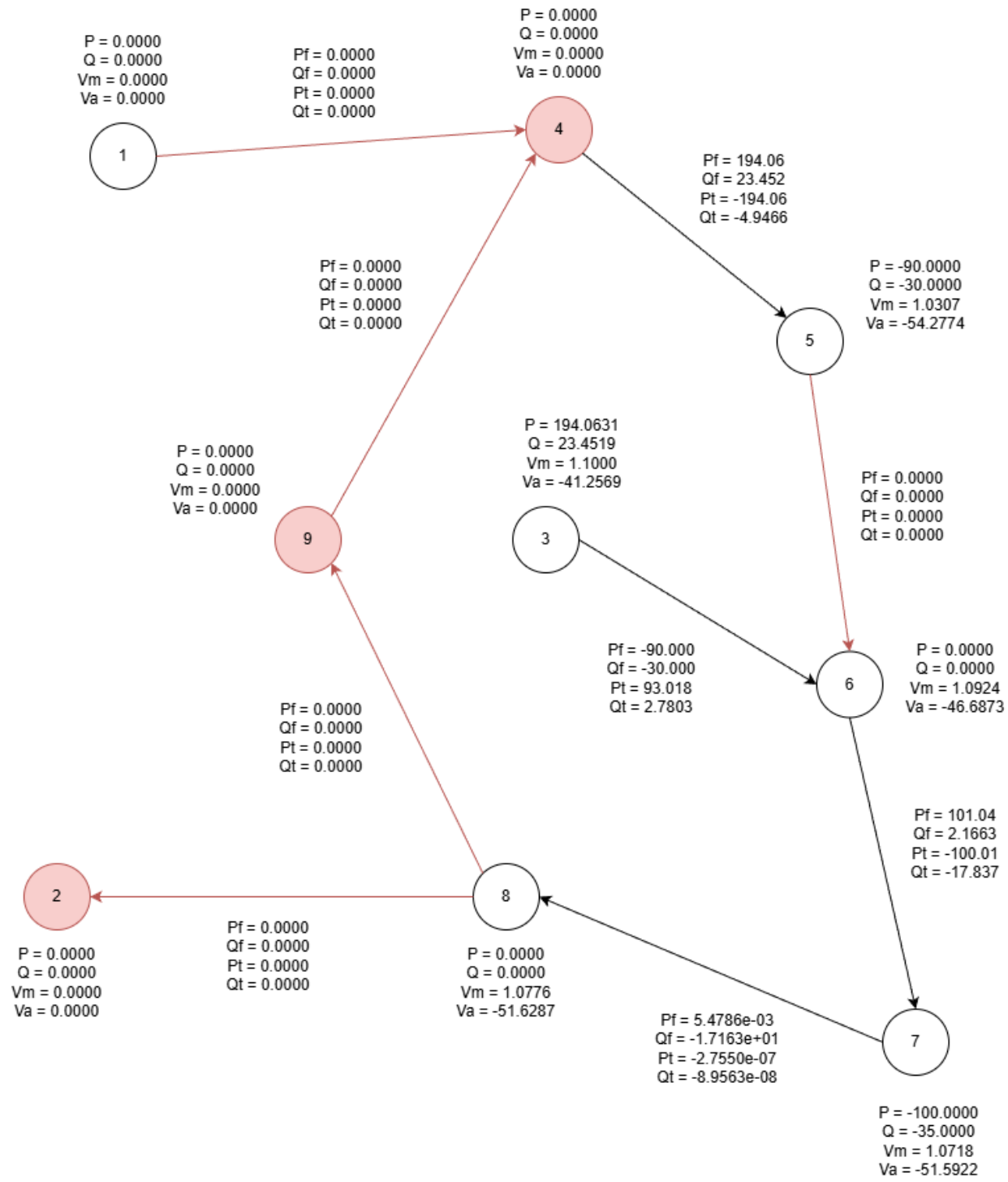

Рисунок 3.10 – Приклад стану електромережі в сценарії 3

Визначення подібних сценаріїв надає можливість визначити схожі умови функціонування електромережі. Якщо електромережа краще функціонує в одному із визначених сценаріїв, то доцільно розглянути варіанти досягнення покращеного стану: зміна параметрів системи, додавання резервних компонентів для забезпечення функціонування об’єкта при виникненні збоїв.

Представлення роботи компонентів електромережі використовуються для визначення множини подібних станів електромережі [180] за допомогою алгоритму кластеризації DBSCAN [196], що визначає автоматично кількість кластерів на основі щільності даних: eps – максимальна відстань між двома точками, які вважаються сусідніми; min_samples – мінімальна кількість точок, яка необхідна для формування кластера. Значення min_samples визначається експериментальний шляхом, eps визначається за допомогою кроків:

1. Обчислити відстань до k-го найближчого сусіда для кожної точки, де k = min_samples, що зображено на рисунку 3.11.

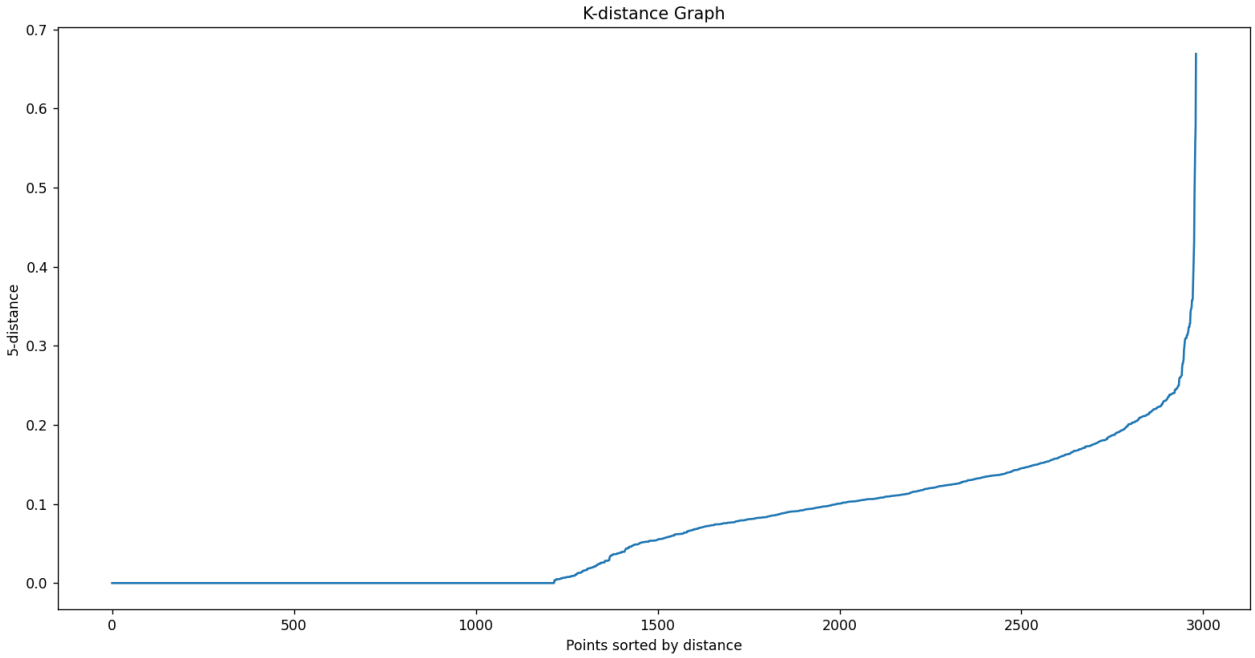

Рисунок 3.11 – Візуалізація k-відстаней сценаріїв електромережі case9

2. Знайти «elbow point». У результаті визначаються кластери: «Cluster 0», «Cluster 1», «Cluster 2», «Cluster 3» та «Noise». Кластери можуть інтерпретуватися як: «Cluster 0» – знеструмлені електромережі, «Cluster 1» – стан електромережі, що призводить до знеструмлення електромережі, «Cluster 2» – стан електромережі, що призводить до часткового знеструмлення електромережі, «Cluster 3» – проміжний чи кінцевий робочий стан електромережі, «Noise» – шуми, кроки сценаріїв (стани електромереж), які не належать до кластерів.

Сценарій 1 належить до «Cluster 2», Сценарій 2 – «Cluster 0», Сценарій 3 – «Cluster 2». Для візуалізації результатів кластеризації використовується PCA [197] для зменшення розмірності даних до 2 вимірів [180], що зображено на рисунку 3.12.

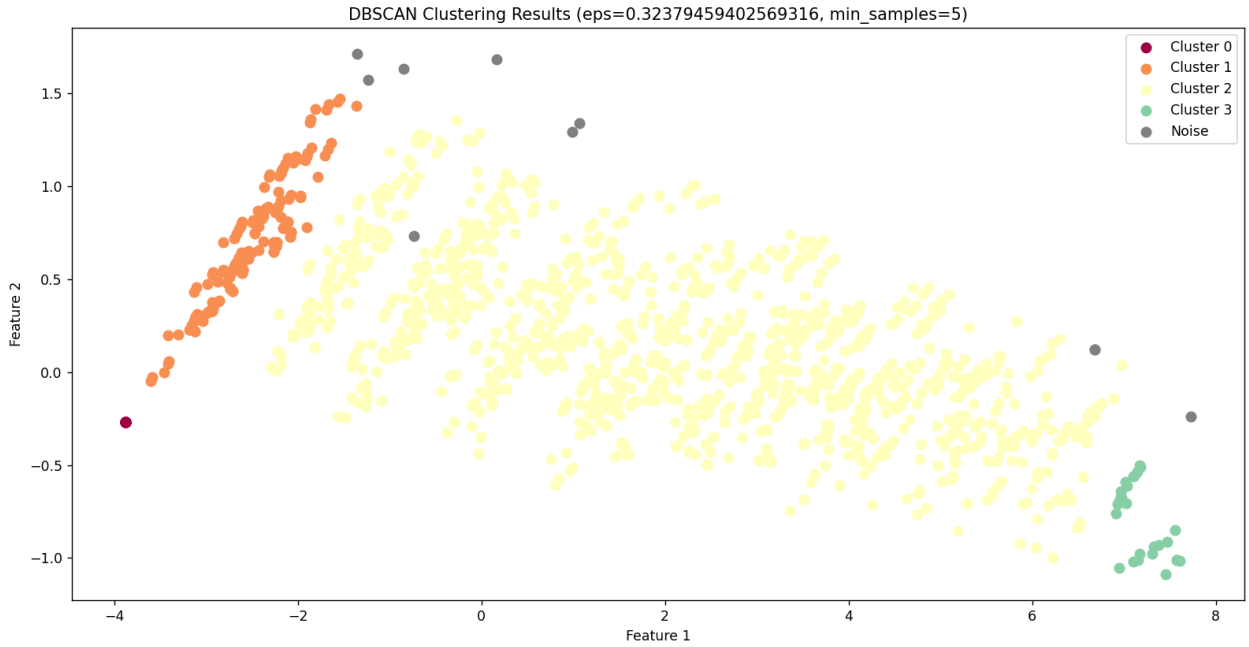

Рисунок 3.12 – Візуалізація кластерів сценаріїв електромережі case9

3.4.6. Удосконалення моделі автоенкодера для створення представлень про послідовності кроків роботи електромережі в сценаріях

Сценарії можуть містити різну кількість кроків – послідовність подій. Перед навчанням моделі потрібно сформувати послідовності подій в сценаріях з максимальною допустимою кількістю кроків. Оскільки послідовність подій в сценарії може бути меншою за визначену максимальну довжину, тому потрібно використовувати маску для визначення реальних та доповнених даних в послідовності. Нехай заданий сценарій S_i, який складається з 5 послідовних кроків: X_1, X_2, X_3, X_4, X_5. Наприклад, при визначенні максимальної кількості кроків в послідовності $maxLength = 3$ формуються послідовності, що зображено на рисунку 3.13.

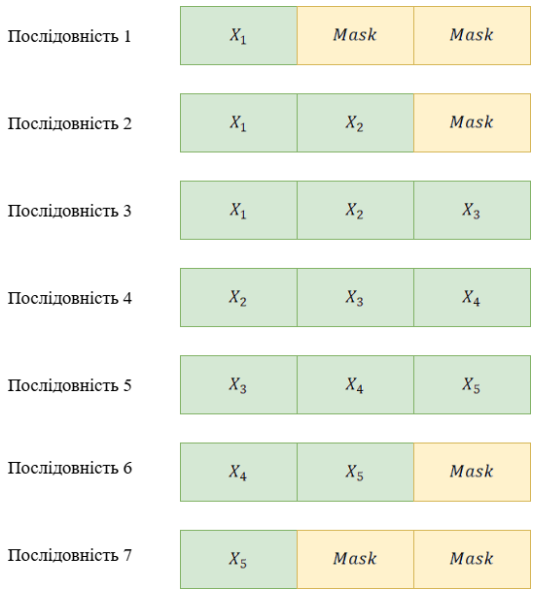

Рисунок 3.13 – Приклад послідовностей кроків сценарію

Для потенційного збільшення ефективності вивчення зв’язків у послідовностях доцільно використовувати LSTM у поєднанні з MultiheadAttention. Це поєднання покращує здатність моделі зосереджуватися на відповідних частинах послідовності завдяки механізму уваги [167]. Багатоголова увага розширює самоувагу (self-attention), вхідні дані розділяються між кількома головами (heads), що дозволяє моделі фіксувати різноманітні зв'язки та закономірності, визначається за формулами (3.85), (3.86):

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O; \quad (3.85)$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \quad (3.86)$$

де Q, K, V – матриці запитів (Query), ключів (Key) и значень (Value);

W_i^Q, W_i^K, W_i^V – матриці ваг проекцій i-ї голови (head);

W^O – матриця ваг, яка об'єднує результати всіх голів (heads).

Модель автоенкодера для формування представлення про зміни в роботі електромережі в сценарії зображена на рисунку 3.14.

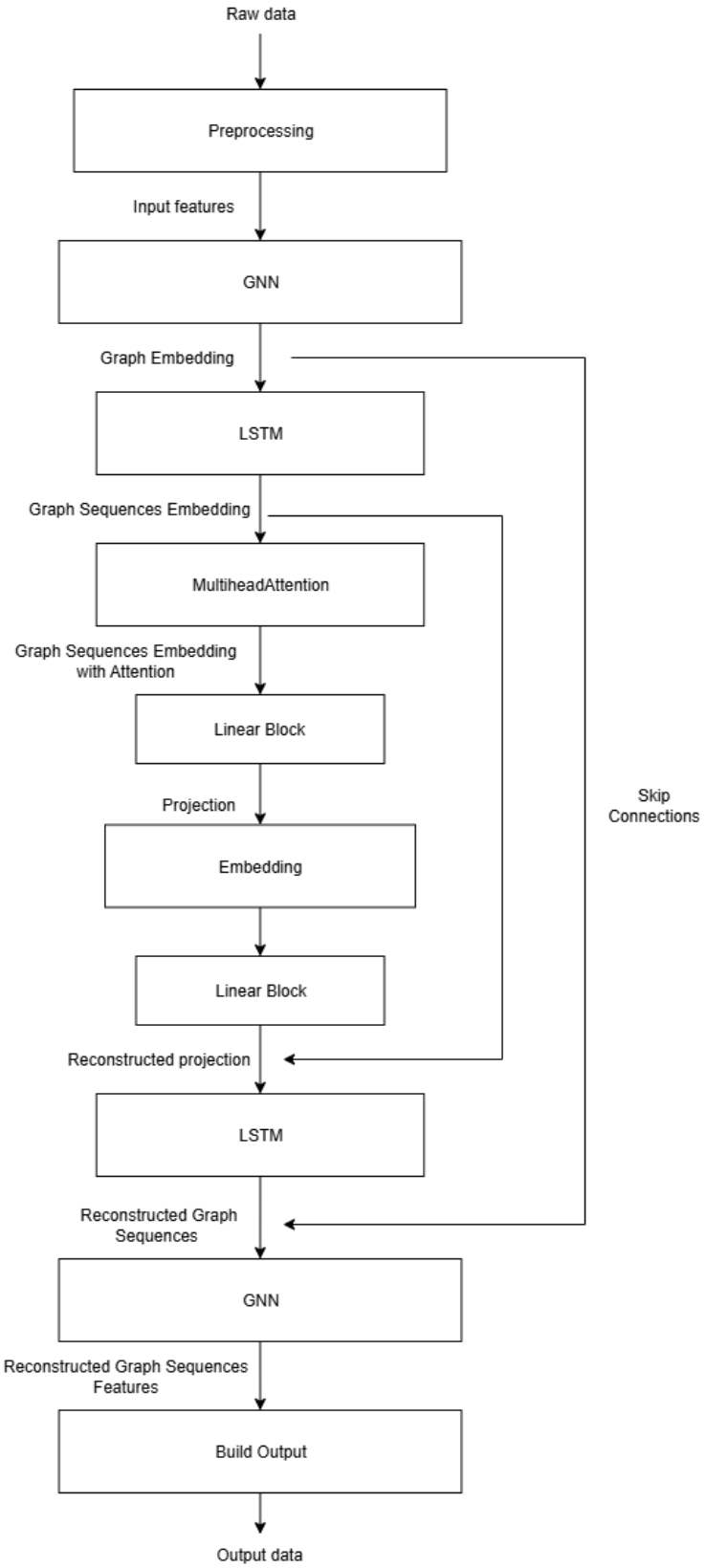

Рисунок 3.14 – Структура автоенкодера для формування представлення про зміни в роботі електромережі в сценарії [180]

Модель автоенкодера складається з таких блоків [180]:

1. Енкодер:

1.1. Блок GNN складається із графових шарів для обробки послідовностей графів. Граф складається із ознак вершин, ознак ребер та структури (зв'язки між вершинами). $V = (Batch\_size, Seq\_length, Num\_nodes, Node\_features)$ – дані про вершини, $E = (Batch\_size, Seq\_length, Num\_Edges, Edge\_features)$ – дані про ребра, $E = (Batch\_size, Seq\_length, 2, Num\_Edges)$ - зв'язки між вершинами, де $Batch\_size$ – розмір пакету даних при пакетній обробці даних, Seq_length - розмір послідовності, Num_nodes – кількість вершин в графі, $Node\_features$ – кількість ознак у вершини графу, Num_Edges – кількість ребер в графі, $Edge\_features$ – кількість ознак у ребра графу. При визначенні реальних та доповнених кроків в послідовностях використовується маска $Mask = (Batch\_size, Seq\_length)$. У блоці для кожного кроку послідовності формуються представлення про вершини та ребра графу з використанням розроблених шарів. Представлення графу формується за допомогою пулінгу, що поєднує ознаки вершин та ребер: $Output\_G = (Batch\_size, Seq\_length, Latent\_dim) = node\_mean\_pool * edge\_mean\_pool$, де $Latent\_dim$ – розмірність вихідних ознак, $node\_mean\_pool$ – пулінг вершин графу, $edge\_mean\_pool$ – пулінг ребер графу.

1.2. Блок LSTM використовується для обробки послідовностей, що були отримані після обробки в попередньому блоці.

1.3. Для додаткового наповнення контексту про кроки в послідовностях (розуміння складних закономірностей) використовується шар MultiheadAttention.

1.4. Лінійний шар застосовується для перетворення даних у визначену розмірність для формування представлення про графи в кроках сценаріїв.

2. Декодер:

1. Лінійний шар трансформує дані до розмірності, яка необхідна в обробці в блоці LSTM декодера.

2. Блок LSTM відновлює ознаки в послідовностях кроків сценаріїв.

3. Лінійні та графові шари відновлюють вершини та ребра графів в послідовності кроків до початкової розмірності.

Для покращення процесу тренування моделі використано skip connections. Результати роботи моделей наведено в таблицях 3.6, 3.7, 3.8.

Таблиця 3.6 – Результати роботи моделі на електромережі case9 при значенні epoch = 150

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedMultiStep	**0,0065**	**0,0036**	**0,0031**	**0,9804**	**0,9910**
GCN	0,0139	0,0114	0,0158	0,8933	0,9876
GAT	0,0154	0,0146	0,0160	0,8853	0,9797
GAT + Edges	0,0161	0,0149	0,0141	0,8909	0,9875
GATV2	0,0161	0,0139	0,0148	0,8860	0,9833
GATV2 + Edges	0,0146	0,0124	0,0137	0,9058	0,9827

Таблиця 3.7 – Результати роботи моделі на електромережі case30 при значенні epoch = 150

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedMultiStep	**0,0073**	**0,0088**	**0,0113**	**0,9038**	**0,9866**
GCN	0,0176	0,0210	0,0228	0,8086	0,9623
GAT	0,0154	0,0231	0,0230	0,8038	0,9842
GAT + Edges	0,0162	0,0216	0,0196	0,8312	0,9721
GATV2	0,0143	0,0202	0,0196	0,8474	0,9847
GATV2 + Edges	0,0154	0,0185	0,0176	0,8387	0,9775

Таблиця 3.8 – Результати роботи моделі на електромережі case94pi при значенні epoch = 150

Layer	Train loss	Val loss	Test loss	Node R^2	Edge R^2
MaskedMultiStep	**0,0021**	**0,0029**	**0,0015**	**0,9774**	**0,9997**
GCN	0,0210	0,0259	0,0156	0,8121	0,9864
GAT	0,0198	0,0214	0,0183	0,8190	0,9818
GAT + Edges	0,0202	0,0194	0,0192	0,8582	0,9793
GATV2	0,0199	0,0219	0,0150	0,8322	0,9893
GATV2 + Edges	0,0206	0,0244	0,0178	0,8219	0,9833

Представлення про послідовність кроків у сценаріях формується в енкодері та використовуються при обчисленні подібності та кластеризації. У результаті визначаються кластери: «Cluster 0», «Cluster 1», «Cluster 2», «Cluster 3», «Cluster 4» та «Noise», що характеризують різний ступінь знеструмлення електромережі в сценаріях (зображені на рисунку 3.15).

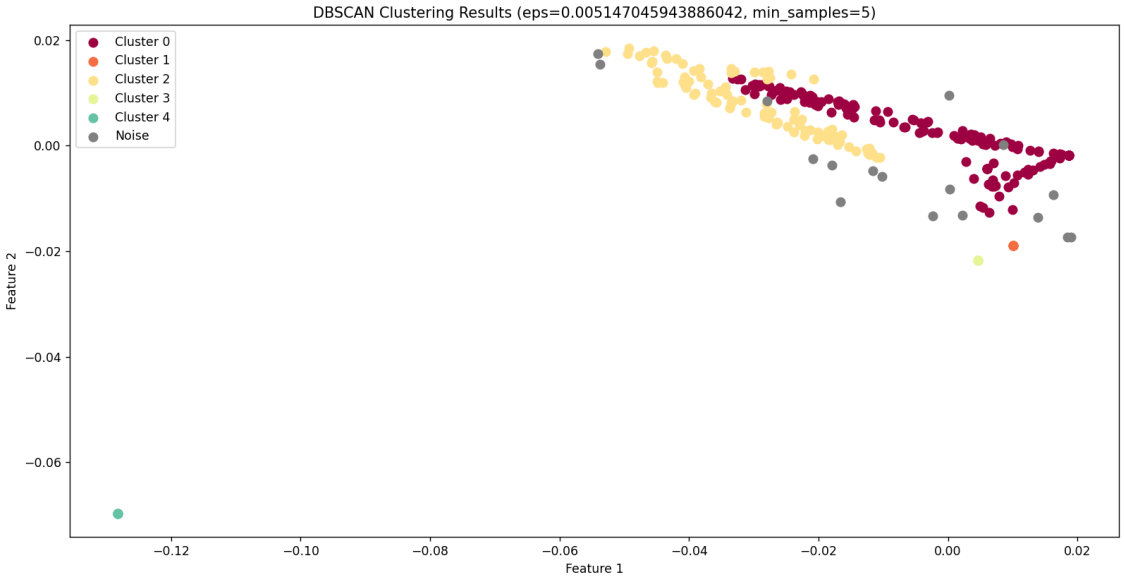

Рисунок 3.15 – Візуалізація кластерів сценаріїв електромережі case94pi

Приклад визначення подібності кроків у сценаріях: у «Сценарії 1» на кроці 3 електромережа case94pi знеструмлена повністю (неактивні всі шини та гілки),

у «Сценарії 2» на кроці 2 в електромережі функціонують 42 шини та 19 гілок, але застосовуються дії для стабілізації роботи системи, у результаті на кроці 3 електромережа знеструмлюється (неактивні всі шини та гілки). У результаті подібність кроків в сценаріях дорівнює 0,9692 – висока подібність станів електромережі.

3.5. Висновки до розділу 3

Розроблено метод визначення подібності сценаріїв каскадних ефектів в електромережах на основі моделі автоенкодера та косинуса подібності, що відрізняється від подібних семантикою взаємодії між об’єктами та їхніми значеннями при порівнянні представлень про стани систем, що дозволяє враховувати структуру та вплив змін в роботі компонентів електромереж. Енкодер натренованої моделі нейронної мережі використовується для створення представлень про роботу електромереж в сценаріях, що порівнюються за допомогою коефіцієнта подібності на основі косинуса подібності із врахуванням семантики та значень ознак. Крім того, отримані представлення використовуються для визначення множини подібних станів електромереж на основі кластеризації.

Удосконалено моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж у сценаріях на основі графових нейронних мереж, які, на відміну від наявних, враховують семантику станів компонентів. Удосконалена модель для створення представлень про стани систем у кроках сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 5,6%, коефіцієнт детермінації ребер у середньому на 27,2%. Удосконалена модель для створення представлень про зміни станів систем у послідовності кроків сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 9,5%, коефіцієнт детермінації ребер у середньому на 0,5%.

РОЗДІЛ 4. РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АНАЛІЗУ СЦЕНАРІЇВ КАСКАДНИХ ЕФЕКТІВ В КРИТИЧНІЙ ІНФРАСТРУКТУРІ

Архітектура програмного забезпечення активно розвивається. З'являються нові підходи, які дозволяють впроваджувати моделі штучного інтелекту, що підвищують якість надання послуг та полегшують процес прийняття рішень. Тому комплексні архітектури програмного забезпечення поєднують особливості побудови традиційного програмного забезпечення та практики використання штучного інтелекту. Для побудови архітектури програмного забезпечення потрібно визначити відповідні вимоги для вирішення поставлених задач, що складається з таких пунктів:

1. Аналіз предметної області та визначення вимог:
    1.1. Визначити тип проблеми (класифікація, регресія тощо).
    1.2. Визначити тип навчання моделі (навчання з вчителем, навчання без вчителя тощо).
    1.3. Визначити вимоги до обробки даних (обробка в реальному часі, пакетна обробка даних).
    1.4. Визначити інструментарій для взаємодії із застосунком.
2. Нефункціональні вимоги:
    2.1. Визначити вимоги до продуктивності системи.
    2.2. Визначити вимоги до точності моделі.
3. Обмеження та переваги інструментів при реалізації системи.
4. Загальна архітектура застосунку.
5. Процес функціонування моделі штучного інтелекту.
6. Архітектура сховища даних.
7. Механізми автентифікації та авторизації.

Визначення архітектури програмного забезпечення є важливими етапом перед його розробкою: визначаються компоненти, їхня взаємодія, обробка запитів та зберігання даних.

Архітектури та патерни проєктування [198, 199] мають свої переваги та недоліки. Їх використовують залежно від вимог до функціоналу. Поширеними архітектурами програмного забезпечення є:

1. Монолітна архітектура. Усі компоненти застосунку (інтерфейс, бізнес-логіка, обробка даних) утворюють єдиний блок. Цей підхід є простим у розробці, підтримці, розгортанні. Проте при зростанні навантаження та складності бізнес логіки процес масштабування та підтримки може викликати певні складнощі. З часом застосунки та навантаження збільшуються. Це призводить до збільшення споживання ресурсів через обмежену гнучкість в масштабуванні, а великі оновлення – до тимчасової недоступності функцій.

2. Багаторівнева архітектура. Структура застосунку визначається у вигляді шарів, що мають відповідні функції та зв’язані, наприклад: шар представлення, шар бізнес-логіки, шар доступу до даних. Модифікації логіки в одному із шарів, зазвичай, не впливає на інші, що спрощує процес оновлення, підтримки та тестування. Можливість масштабувати незалежно окремі шари підвищує масштабованість застосунку. Крім того, компоненти кожного шару можуть бути повторно використані в різних частинах програми, а шари можуть бути розширені, оновлені, що не впливає на всю систему. Цей шаблон часто використовується в клієнт-серверних системах. Він розділяє функціональність на фізичні рівні, що покращує продуктивність.

3. Мікросервісна архітектура. Застосунок поділяється на невеликі незалежні сервіси – мікросервіси, кожен з яких виконує певну функцію (бізнес-логіку) та взаємодіють через API. Кожен із мікросервісів може використовувати різні технології для виконання поставлених завдань. Вони можуть оновлюватися, розгортатися та масштабуватися відповідно до потреб, що забезпечують гнучкість і стійкість застосунку.

4. Подійно-орієнтована архітектура (Event-Driven Architecture). Сервіси працюють незалежно, взаємодіють через брокер подій, що забезпечує можливість обробки даних у режимі реального часу, гнучкість при проєктуванні системи. Event Producer створює подію, а Event Consumer отримує подію та

відповідно реагує. Система може легко масштабуватися: додаються або видаляються сервіси, а збій в одному з компонентів не зупиняє роботу всього застосунку.

5. Безсерверна архітектура (Serverless architecture). У безсерверній архітектурі постачальник хмарних послуг керує ресурсами сервера, де виконується код. Розробники можуть зосередити увагу над бізнес-логікою та її реалізацією, а масштабування ресурсів відповідно до потреб виконується провайдером.

Для обробки та аналізу даних про сценарії роботи електромереж було розроблено архітектуру програмного забезпечення, що зображено на рисунку 4.1.

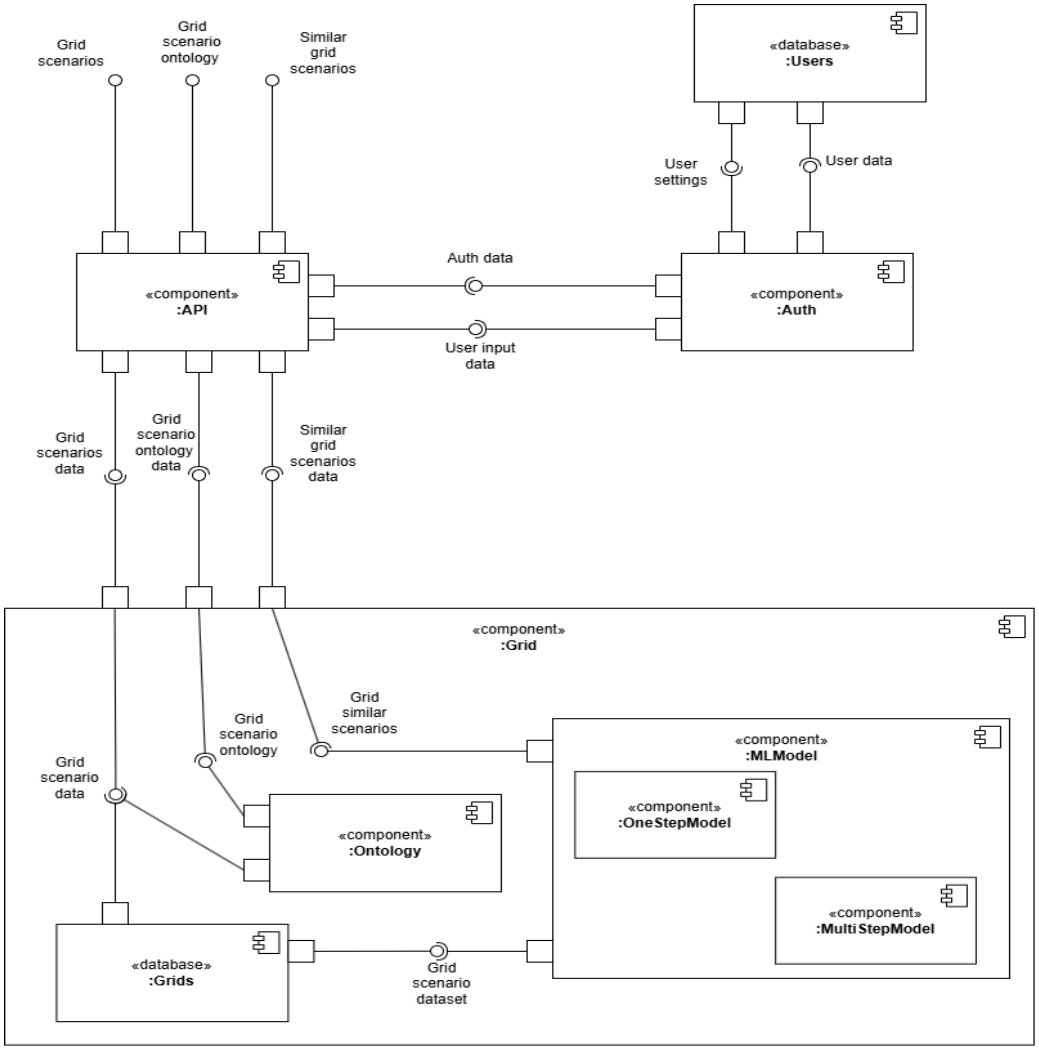

Рисунок 4.1 – Діаграма компонентів архітектури програмного забезпечення

Для реалізації програмного застосунку обрано мікросервісну та подійно-орієнтовану архітектури, що надають можливість обробляти, аналізувати великі обсяги даних та масштабувати сервіси залежно від навантаження. Взаємодія об’єктів у системі зображено на рисунку 4.2.

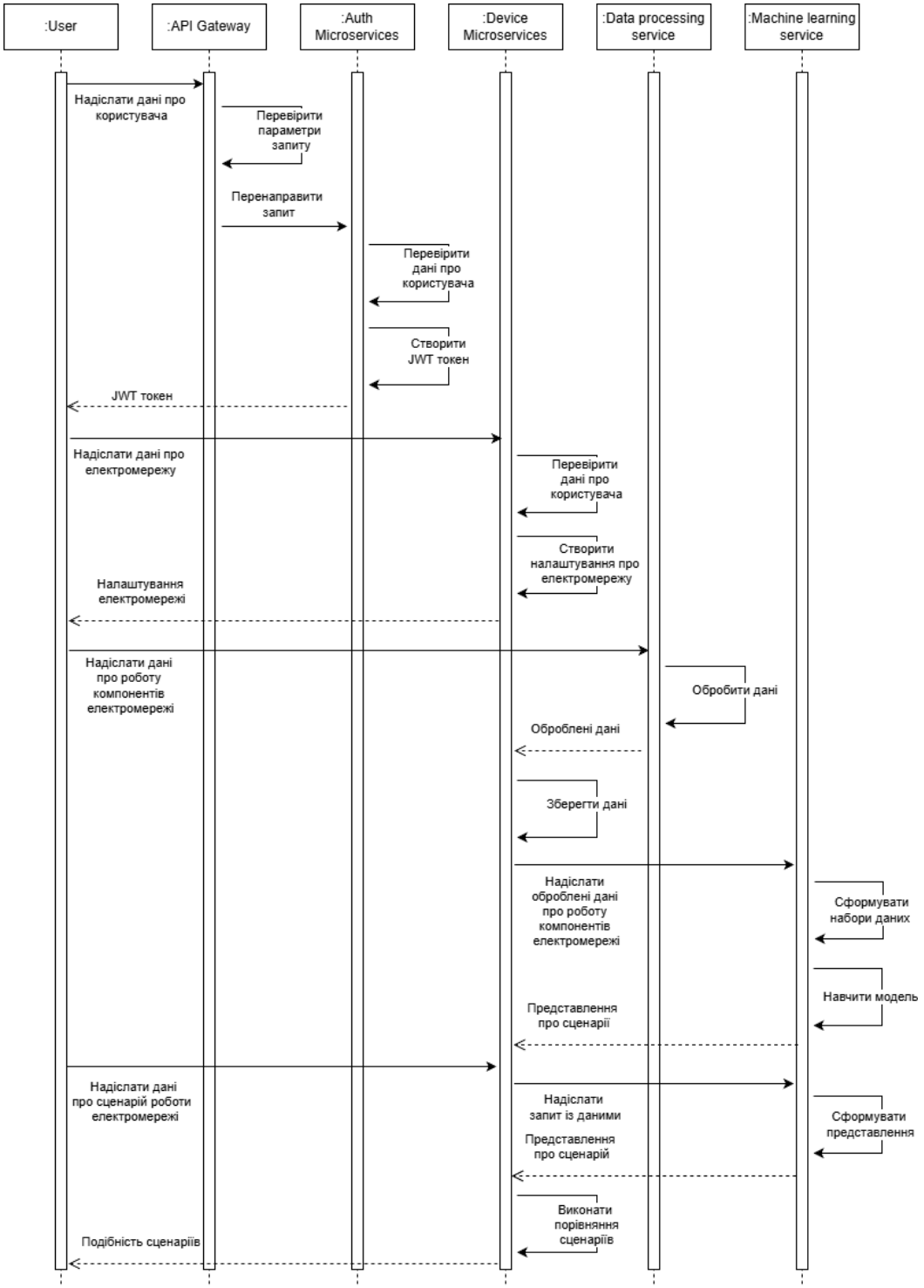

Рисунок 4.2 – Діаграма послідовності взаємодії об’єктів у системі при визначенні подібності станів електромереж в сценаріях

4.1. Налаштування обробки вхідного запиту

API шлюз (API gateway), зворотний проксі (reverse proxy) та балансувальник навантаження (load balancers) використовуються в мікросервісній архітектурі та забезпечують створення масштабованих продуктів, які потенційно здатні витримувати велике навантаження.

Балансувальник навантаження розподіляє вхідний трафік між кількома мікросервісами, що забезпечує високу доступність, надійність, масштабованість. Запити відправляються до робочих екземплярів і перерозподіляються, якщо він виходить з ладу. Балансувальник навантаження використовує інтелектуальну маршрутизацію запитів з використанням алгоритмів, наприклад: Round Robin, Least Connections, IP Hash та працює на рівні 4 рівні (Transport) або 7 рівні (Application) моделі OSI. Розподіл запитів забезпечує оптимальну потужність, запобігає перевантаженню будь-якого з мікросервісів (перетворенню на вузьке місце) та забезпечує стабільну продуктивність застосунку при різних навантаженнях [200].

Зворотний проксі-сервер (reverse proxy) використовується в запитах між клієнтом та серверами як додатковий рівень безпеки: маскувати IP-адреси серверів бекенду, запити від клієнтів перенаправляються на один або кілька серверів на основі визначених правил, логіки та повертає відповідь сервера клієнту. Зворотний проксі-сервер працює на 7-му рівні моделі OSI та використовується для балансування навантаження, підвищення безпеки: приховує внутрішні сервери від прямих запитів зовнішнього клієнта, використовує налаштування для обробки запитів (фільтрація трафіку, IP-адрес, блокування DoS/DDoS-атак, що зменшує ймовірність негативних наслідків), розподіл трафіку та маршрутизацію на основі контексту запиту, SSL termination, кешування (кешування відповіді від серверів, для повторних запитів, що зменшує навантаження на сервери та зменшує затримку для користувачів) [200].

API шлюз (API Gateway) використовується як вхідна точка для клієнтських запитів (розташований між клієнтами та серверними), діє як зворотний проксі-

сервер, працює на 7-му рівні моделі OSI та має функціонал для обробки запитів, наприклад: централізована маршрутизація запитів до відповідного мікросервісу на основі контексту, перетворення запитів і відповідей, централізована автентифікація та авторизація, обмеження швидкості (запобігання надмірній кількості запитів, які клієнт може надіслати протягом певного періоду часу), логування подій, моніторинг (збирання метрик про запити та відповіді) для подальшого аналізу. Одним із недоліків використання API шлюзу є додаткова складність через організацію налаштувань.

Для реалізації вхідної точки в програмному забезпеченні був обраний NGINX [201], який використовується як балансувальник навантаження та зворотний проксі-сервер, що забезпечує високу продуктивність, безпеку, ефективність та масштабованість при обробці великої кількості запитів. Обробка вхідного запиту зображено на рисунку 4.3.

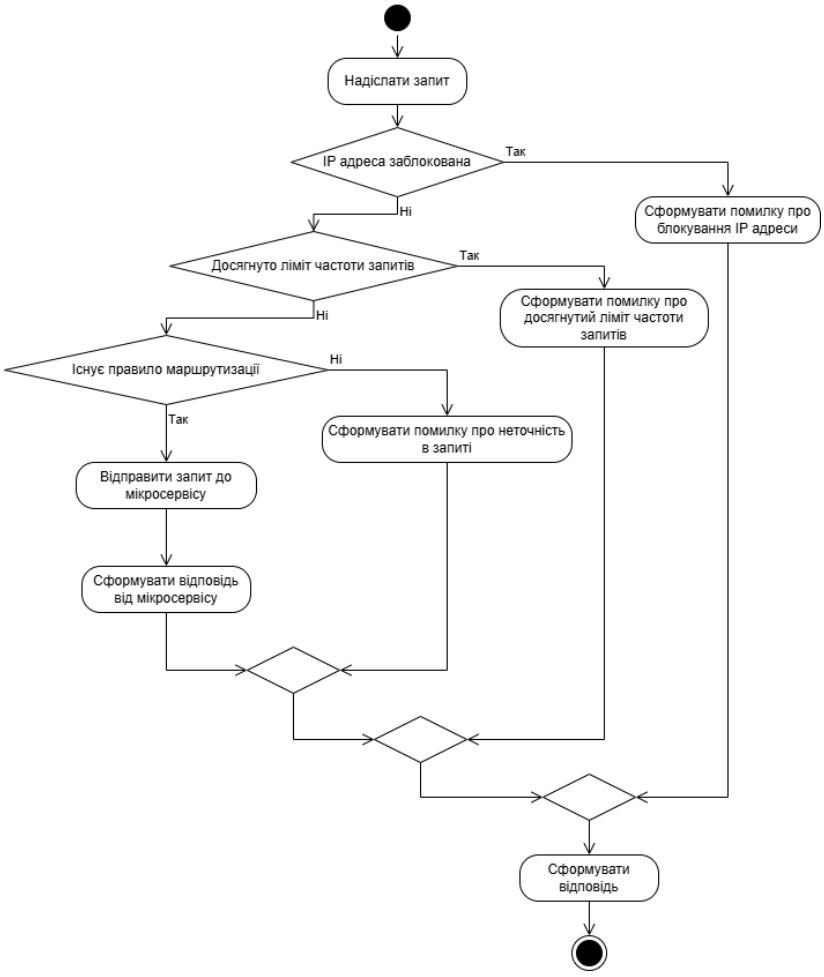

Рисунок 4.3 – Діаграма діяльності обробки вхідного запиту

4.2. Налаштування автентифікації та авторизації

Безпека даних відіграє важливу роль при проєктуванні систем. Автентифікація та авторизація є важливими компонентами для забезпечення безпеки в системі. Автентифікація – це процес перевірки особи користувача або сервісу. Авторизація – це процес визначення дій, які дозволені виконувати автентифікованому користувачеві або сервісу [202]. Мікросервіси потребують впровадження заходів безпеки при обробці запитів: контроль доступу, шифрування даних (використання SSL). У мікросервісній архітектурі метод автентифікації вибирається відповідно до вимог та обмежень.

Поширеним методом автентифікації та авторизації в мікросервісній архітектурі є JWT (JSON Web Tokens) [203]. JWT токен складається з трьох частин, які розділені крапками (.) [204]:

1. Header, що складається з двох частин: типу токена, алгоритму, який використовується для підписання, наприклад HMAC SHA256 або RSA. Потім цей JSON кодується Base64Url, щоб сформувати першу частину JWT [204].

2. Payload, що складається із claims – твердження про користувача та додаткові дані. Існує три типи claims: registered, public та private. Потім payload кодується Base64Url для формування другої частини JWT [204].

3. Signature, що створюється за допомогою закодованого header, закодованого payload та секретного ключа [204]. Signature використовується для перевірки цілісності даних при відправці.

Для підпису JWT токена можна використовувати асиметричне шифрування: застосування пари приватного ключа (private key) та публічного ключа (public key), що підвищує рівень безпеки. Приватний ключ використовується при створенні токена в мікросервісі автентифікації, а публічний ключ використовується в інших мікросервісах для верифікації токена. JWT токен є statelessness, що спрощує масштабування системи. Обробка запиту користувача до Auth мікросервісу зображено на рисунку 4.4.

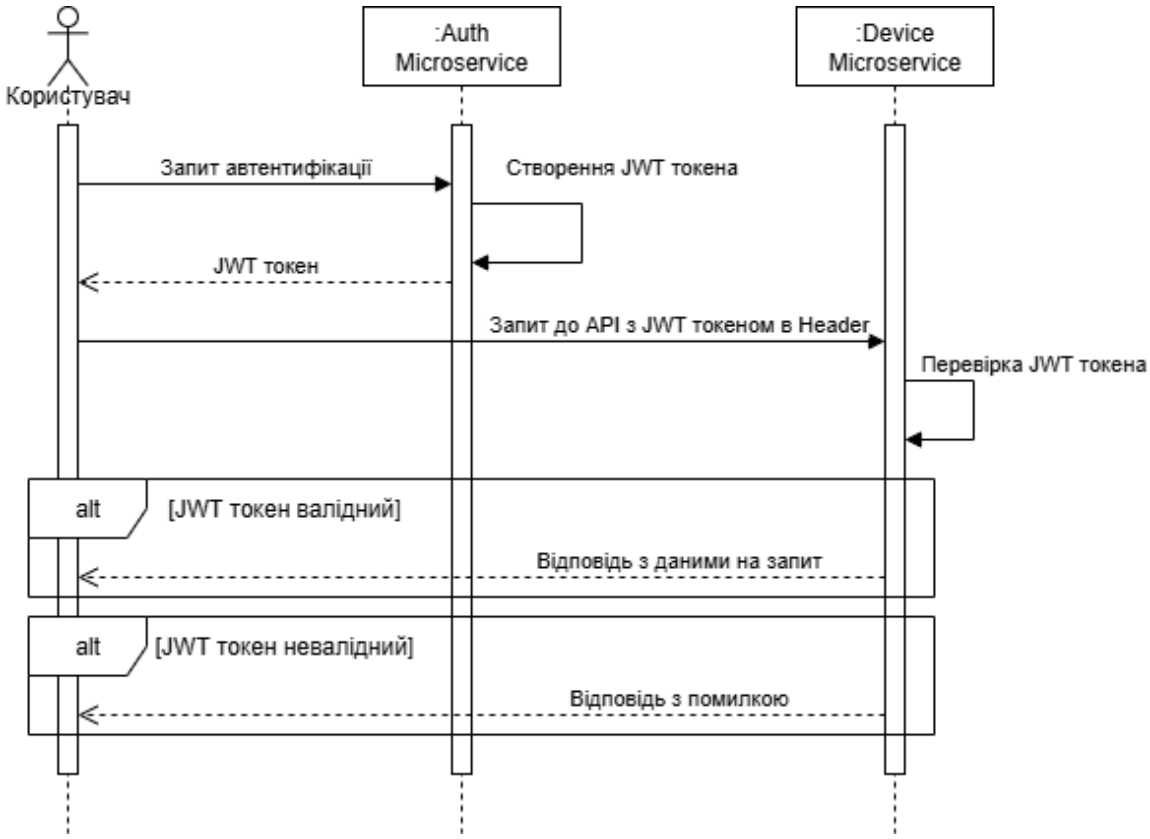

Рисунок 4.4 – Обробка запиту користувача до Auth мікросервісу

4.3. Розробка API мікросервісів для аналізу сценаріїв каскадних ефектів в критичній інфраструктурі

Розробка API починається з проєктування бази даних. Вибір бази даних залежить від масштабу та складності задач. PostgreSQL – об’єктно-реляційна система баз даних з відкритим кодом [205], що зберігає дані в таблицях, яка складається з рядків (записи про дані) та стовпців (атрибутів). Між таблицями можуть бути встановлені зв’язки за допомогою ключів, що дозволяє створювати складні структури даних та ефективно виконувати запити. PostgreSQL підтримує ACID [206] – набір властивостей, що гарантують надійність обробки транзакцій у базі даних:

– атомарність (atomicity): транзакція є єдиною, неподільною. Вона виконується повністю або не виконується взагалі;

– узгодженість (consistency): транзакція переводить базу даних зі стану в стан, який відповідає всім правилам і обмеженням;

– ізольованість (isolation): паралельні транзакції не впливають одна на одну, результати транзакції не видно іншим транзакціям до моменту її завершення;

– довговічність (durability): результати успішних транзакцій зберігаються, навіть у разі виникнення збоїв у системі.

PostgreSQL має функціонал для обробки великих обсягів структурованих та неструктурованих даних. Вони використовуються для подальшого аналізу (наприклад, підготовка даних для навчання нейронних мереж), що підходить для організації сховища даних про електромережу.

Для користувача необхідно забезпечити можливість додавати та змінювати налаштування про компоненти електромережі. Для збереження налаштувань про електромережу було створено базу даних grid_database з таблицями: grids – налаштування про мережу, buses – налаштування про шини, gens – налаштування про генератори, branches – налаштування про гілки, що зображені на рисунку 4.5.

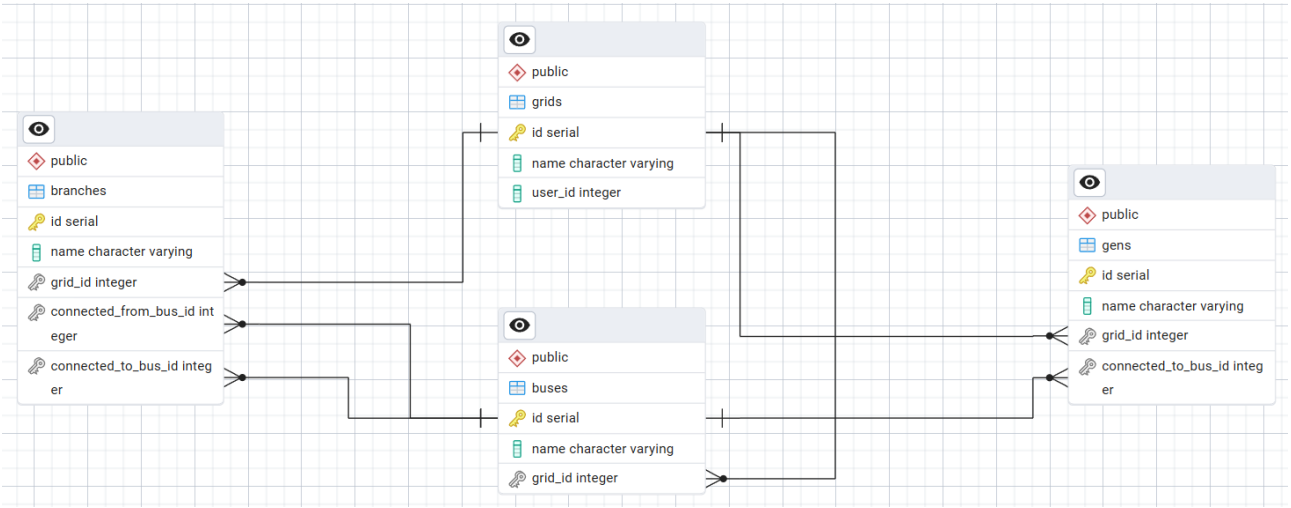

Рисунок 4.5 – Структура таблиць компонентів електромережі та зв’язки між ними в базі даних

Дані про користувача зберігаються в окремій базі даних auth_database в таблиці users, що використовується в мікросервісі для аутентифікації та авторизації при використанні методів API.

Для взаємодії з базою даних використовується SQLAlchemy. Вона допомагає будувати ефективні SQL запити, використовуючи мову програмування Python та об'єктно-орієнтований підхід [207]. При створенні API мікросервісів використовується FastAPI – високопродуктивний вебфреймворк для створення API за допомогою Python [208].

У мікросервісі автентифікації та авторизації (auth_service) створено API endpoints, що зображені на рисунку 4.6.

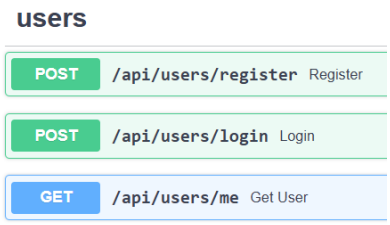

Рисунок 4.6 – API endpoints мікросервісу автентифікації та авторизації

Реєстрація користувача (/api/users/register). Користувач вводить логін та пароль і надсилає POST запит до API. Логін повинен бути унікальним, тому відбувається перевірка введеного логіну. При успішній валідації даних створюється запис в базі даних про нового користувача та повертається відповідь про успішне виконання запиту, що зображено на рисунку 4.7.

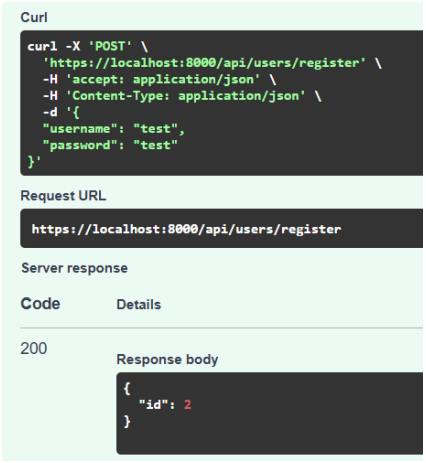

Рисунок 4.7 – Результат успішного виконання запиту register

У результаті виконання запиту повертається ідентифікатор створеного користувача. Якщо користувач з введеним логіном існує, то повертається помилка у відповіді, що зображено на рисунку 4.8.

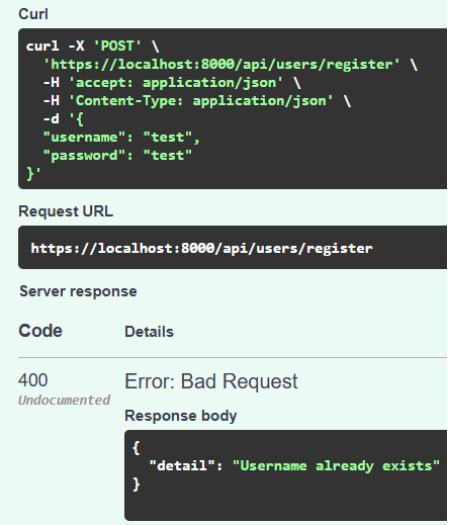

Рисунок 4.8 – Результат виконання запиту register, якщо користувач вже існує

Логін користувача (/api/users/login), отримання access token. Користувач вводить облікові дані для отримання access token. У результаті обробки запиту відбувається перевірка даних про користувача та генерація токену, що зображений на рисунку 4.9. Він містить необхідні дані для надання доступу до відповідної електромережі. Цей токен використовується в запитах до мікросервісу електромережі (grid_service).

Рисунок 4.9 – Access token після успішного виконання запиту login

При введені некоректних даних користувача відповідь повертається з описом помилки, що зображено на рисунку 4.10.

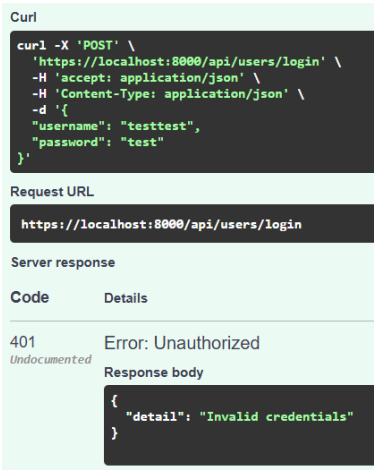

Рисунок 4.10 – Опис помилки при виконанні запиту login

Отримання даних про користувача (/api/users/me). У токену можуть міститися дані (зображено на рисунку 4.11), які можуть бути використані в запитах, де потрібна додаткова інформація про користувача. При виникненні помилки при обробці запиту користувач отримує повідомлення з описом помилки, зображено на рисунку 4.12.

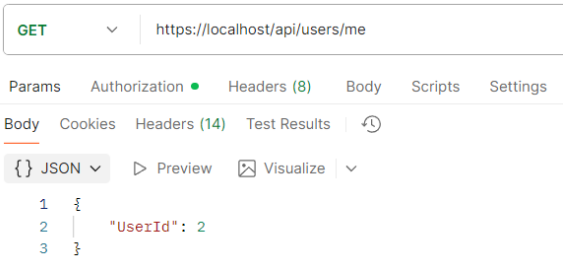

Рисунок 4.11 – Дані про користувача, які отримані з access token

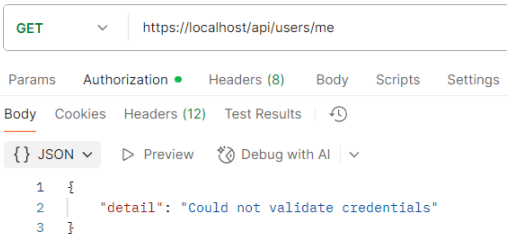

Рисунок 4.12 – Інформація про помилку при зчитуванні даних з access token

У мікросервісі електромережі (grid_service) створено API endpoint для роботи з компонентами та даними електромережі (у заголовку запиту передається access token, який отримується від auth_service по запиту login):

1) GET запити:

– /api/grids/ – отримати дані про електромережі, які доступні користувачу;

– /api/grids/{grid_id} – отримати дані про електромережу по вказаному ідентифікатору;

– /api/grids/{grid_id}/buses – отримати дані про шини електромережі;

– /api/grids/{grid_id}/gens – отримати дані про генератори електромережі;

– /api/grids/{grid_id}/branches – отримати дані про гілки електромережі;

2) POST запити:

– /api/grids/ – створити структуру електромережі;

– /api/grids/{grid_id}/buses – додати шину для вказаної електромережі;

– /api/grids/{grid_id}/gens – додати генератор для вказаної електромережі;

– /api/grids/{grid_id}/branches – додати гілку для вказаної електромережі;

3) PUT запити:

– /api/grids/{grid_id} – оновити дані про електромережу по вказаному ідентифікатору;

– /api/grids/{grid_id}/buses/{bus_id} – оновити дані про шину для вказаної електромережі;

– /api/grids/{grid_id}/gens/{gen_id} – оновити дані про генератор для вказаної електромережі;

– /api/grids/{grid_id}/branches/{branch_id} – оновити дані про гілку для вказаної електромережі;

4) DELETE запити:

– /api/grids/{grid_id} – видалити дані про електромережу по вказаному ідентифікатору;

– /api/grids/{grid_id}/buses/{bus_id} – видалити дані про шину для вказаної електромережі;

– /api/grids/{grid_id}/gens/{gen_id} – видалити дані про генератор для вказаної електромережі;

– /api/grids/{grid_id}/branches/{branch_id} – видалити дані про гілку для вказаної електромережі.

4.4. Налаштування обробки даних про роботу компонентів електромережі

Збір даних є початковим етапом для обробки даних з різних джерел: бази даних, API, файли, пристрої Інтернету речей. Вхідні дані проходять через процес обробки, зберігаються в сховищі даних та використовуються при подальшому аналізі. Існує пакетна обробка даних (Batch), обробка даних у реальному часі (Real-Time, Streaming) та гібридна (Hybrid), що використовується залежно від швидкості обробки та обсягу даних.

Пакетна обробка даних опрацьовує дані фіксованими фрагментами (можлива паралельна обробка). Вона використовується в завданнях, які є менш чутливими до часу та виконуються з певною періодичністю (наприклад, щоденна статистика). При обробці даних може бути застосована горизонтальна масштабованість: додавання ресурсів за потреби при виконанні обчислень. Пакетна обробка даних відносно проста у впровадженні, оскільки не потребує складних механізмів обробки даних у режимі реального часу. Проте пакетна обробка даних має недоліки:

– затримка при обробці даних (аналітика недоступна одразу, що може стати критичним для програм);

– застарілі аналітичні дані для аналізу (потенційно може негативно вплинути на прийняття рішень).

Обробка даних у реальному часі (потокова обробка даних) відбувається одразу після їхнього створення. Цей тип обробки даних застосовується у задачах, де важлива низька затримка при обробці даних (наприклад, негайна аналітика даних із датчиків). Потокова обробка даних надає можливість формувати аналітику та швидко реагувати на критично важливі події (аналітика актуальна та релевантна). Але реалізація може бути складнішою, ніж у пакетній обробці:

– потребує забезпечення узгодженості даних;

– потребує обчислювальних ресурсів та інфраструктури для обробки безперервних потоків даних.

Для надсилання даних про стан та показники компонентів електромережі використовується POST запит (/api/grids/data). Вхідні дані перевіряються та зберігаються в таблицях бази даних, які зображені на рисунку 4.13. Ідентифікаторами події є стовпчики scenario_id, sub_scenario_id та компонент електромережі (gri_id / bus_id /gen_id, branch_id).

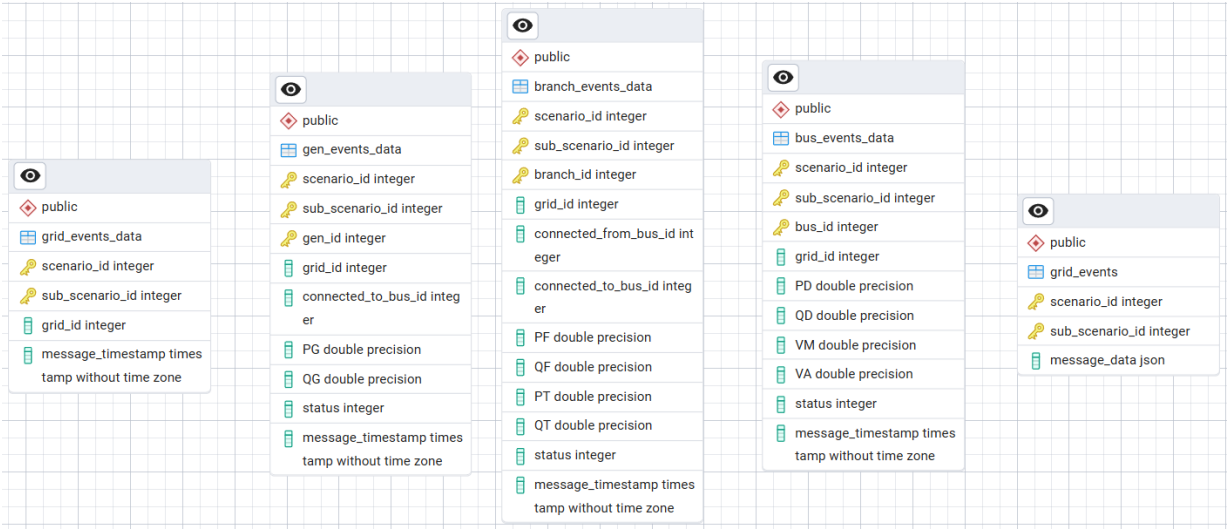

Рисунок 4.13 – Структура таблиць з даними про компоненти електромережі, які надходять від користувача

Захоплення змін даних (Change Data Capture) – це метод, що використовується в базах даних для відстеження змін в даних (вставка, оновлення, видалення) та допомагає підтримувати узгодженість даних у системах [209]. Для виявлення зміни даних у режимі реального часу використовується Debezium [210], що публікує події в брокер повідомлень Apache Kafka [211] для подальшої потокової обробки та проведення аналітики в реальному часі. Дані з таблиць компонентів електромережі надсилаються у відповідний topic, які зображені на рисунку 4.14:

– grid_server.public.grid_events_data – дані про електромережу;

– grid_server.public.bus_events_data – дані про шини електромережі;

– grid_server.public.gen_events_data – дані про генератори електромережі;

– grid_server.public.branch_events_data – дані про гілки електромережі.

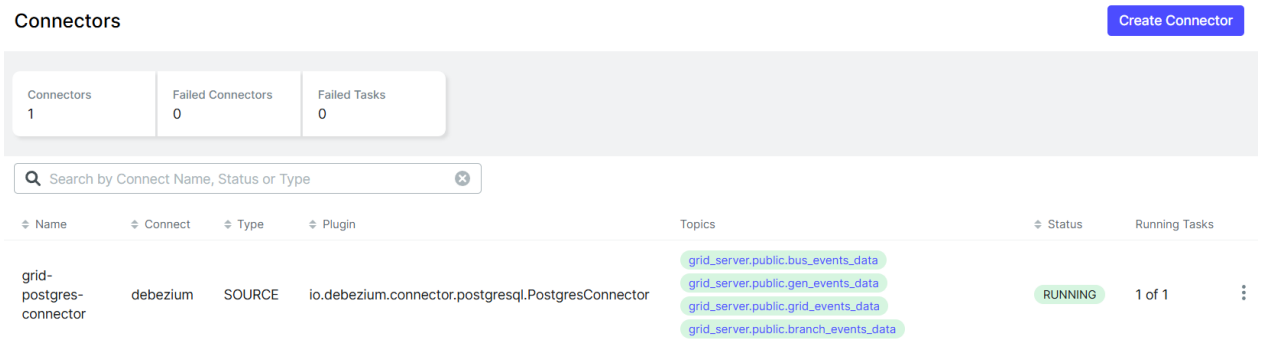

Рисунок 4.14 – Налаштування Debezium для зчитування змін в таблицях

Для обробки потоків даних в режимі реального часу використовується Apache Flink [212]. Apache Flink має такі переваги:

– можливість оброблювати як необмежені потокові дані (unbounded, streaming data), так обмежені пакетні дані (bounded, batch data);

– висока продуктивність, низька затримка та висока пропускна здатність;

– висока відмовостійкість, гарантує надійність та доступність даних;

– масштабованість: обробка великих наборів даних;

– розширений функціонал для обробки даних у вікні (windowing);

– інтеграція з іншими інструментами для роботи з великими даними, підтримка мов програмування, наприклад: Java, Scala та Python.

Налаштуванням процесів обробки даних, конфігурації і розгортання є комплексним процесом, що потребує спеціалізованих навичок.

Середовище виконання Flink (Flink runtime) – це розподілений механізм, який відповідає за виконання визначених користувачем завдань (jobs) обробки даних (потокових, пакетних) у кластері Flink (зображено на рисунку 4.15).

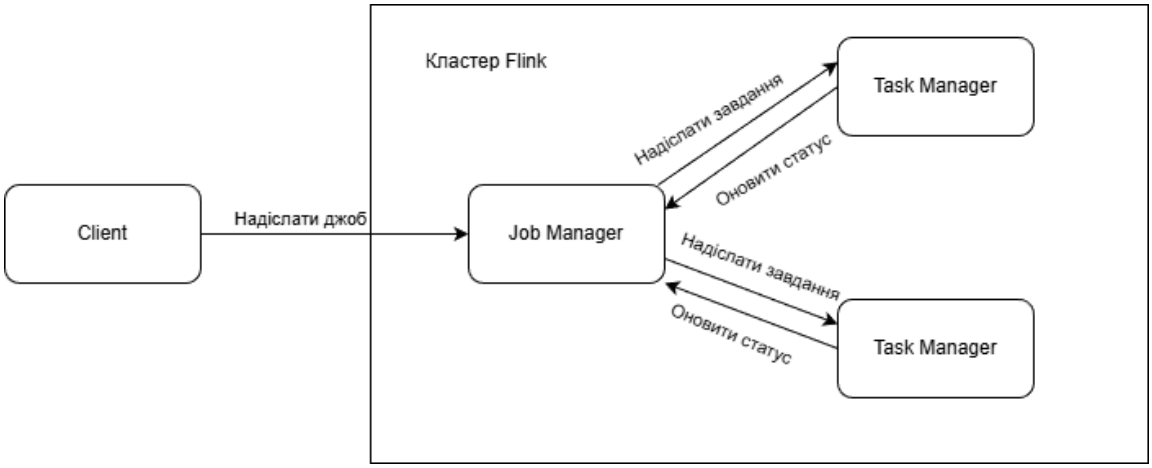

Рисунок 4.15 – Середовище виконання Flink (Flink runtime)

Клієнт (Client) не є частиною середовища виконання Flink, а відповідає за створення графу завдання Flink (Flink job graph) та його надсилання (submitting) менеджеру завдань (Job manager). Менеджер завдань отримує граф завдання від клієнта, планує його та контролює виконання в менеджерах виконання завдань (Task manager). Менеджер виконання завдань відповідає за виконання всіх завдань, які були призначені йому та повідомляє про їхній статус.

Важливим компонентом при обробці даних у потоці є часова ознака (timestamp). Flink підтримує три варіації часу в даних [213]:

1. Час події (Event time) – це час, коли подія відбулася в джерелі даних. Він використовується в ситуаціях, де потрібна точність результатів, оскільки використовується реальний час події. У процесі обробки даних потрібно враховувати невпорядковані дані та дані із затримкою, використовуючи водяні знаки (watermarks).

2. Час отримання (Ingestion time) – це час, коли подія потрапляє до Flink. Він використовується, коли час події недоступний, але для даних потрібна позначка часу. Час отримання менш точний, ніж час події, оскільки не відображає початковий час її виникнення.

3. Час обробки (Processing time) – це час, коли оператор у Flink обробляє подію. Він використовується, коли продуктивність у режимі реального часу є важливішою за точність. Простий та швидкий у використанні, але найменш точний, оскільки на нього впливають затримки мережі та поточне навантаження системи (використовує системний годинник машини, на якій оброблюється подія).

При обробці даних про компоненти електромережі використовується час події: поле message_timestamp, що міститься в json, який надсилає користувач до API. При використанні часу події потрібно налаштувати обробку позначок часу та генератор водяних знаків, які використовуватимуться для відстеження перебігу часу події. У системах події можуть надходити в різному порядку, зокрема й затримкою, оскільки можуть бути збої в системі. Без реалізації належного механізму обробки подібних подій це може призвести до неправильних результатів під час аналізу даних.

Водяні знаки (Watermarks) – це спеціальні маркери в потоці даних, які повідомляють Flink про перебіг часу події та є лімітом, який вказує, що всі події з попередньою позначкою часу (менша за водяний знак), ймовірно, настали. Водяні знаки генеруються з використанням часу подій, з визначеною затримкою для врахування подій, які надходять із запізненням. Водяні знаки повідомляють про можливість обробки даних у вікнах (windows).

Віконна обробка (Windowing) використовуються для поділу потоку на фрагменти скінченого розміру на основі визначених критеріїв – вікна (windows). Для вікна можна застосувати різні операції, наприклад: перетворення, агрегацію чи комплексну бізнес-логіку. Flink має функціонал для роботи з такими типами вікон [214]:

1. Tumbling windows – це вікна фіксованого розміру, що не перекриваються (not overlap). Кожна подія належить одному вікну. Вони використовуються для простих, періодичних агрегацій протягом визначеного періоду часу.

2. Sliding windows – це вікна фіксованого розміру, що перекриваються (overlap). Вони використовуються для аналізу даних з перекриваючими часовими інтервалами.

3. Session windows – це вікна динамічного розміру, які групують події на основі періодів активності. Вікна сеансу не перекриваються та не мають фіксованого часу початку та завершення. Вони використовуються для групування активності користувача в сесії.

4. Global windows – вікна, які охоплюють весь потік. Вікно закривається і відбувається обчислення, коли виконується певна умова (trigger).

Процес обробки потоків даних складається із кроків:

1. Визначити джерело даних: дані зчитуються з вказаними налаштуваннями із topic Apache Kafka, формуються потоки: bus_data_stream, gen_data_stream, branch_data_stream.

2. Визначити цільове сховище даних. Дані вставляються у визначені таблиці бази даних PostgreSQL: таблиця «alert_messages» містить повідомлення із інформацією про помилку в компоненті електромережі; таблиця «nodes» містить інформацію про вершини графу електромережі; таблиця «edges» містить інформацію про вершини графу електромережі.

3. Перетворити вхідні дані.

    3.1. Зчитати json про оновлені рядки в таблицях.

    3.2. Визначити дані про вставку нових рядків в таблиці.

    3.3. Вибрати необхідні поля із вхідних даних.

4. Порівняти фактичні та допустимі значення компонентів електромережі.

    4.1. Сформувати потік із повідомленнями про помилку у вхідних даних.

    4.2. Зберегти повідомленнями із деталями помилки у базі даних в таблиці alert_messages».

5. Визначити позначки часу події в потоці даних та створення водяних знаків. Події можуть надходити із затримкою, тому встановлюється максимальна кількість часу, протягом якого події можуть бути не в логічному порядку. У такому випадку встановлено значення for_bounded_out_of_orderness 5 секунд. Якщо остання подія має позначку часу T, генерується водяний знак, який повідомляє, що можна безпечно завершити вікна для подій з позначкою часу меншою або рівною T-5 секунд. Іноді дані можуть затримуватися (наприклад, збій partition), тому доцільно встановити тайм-аут with_idleness 1 хвилина (якщо протягом заданого часу очікування події не надходять, джерело позначається як неактивне), що запобігає зупинці завдання, виконання операцій. Час події визначається із поля «message_timestamp», але при виникненні помилки (позначка часу може бути відсутньою або мати неправильний формат) встановлюється позначка часу запису Kafka.

6. Застосування key_by – усі записи з однаковим ключем призначаються одному partition, перетворення DataStream на KeyedStream. Для потоків визначаються ключі: «bus_id», «gen_id», «branch_id».

7. Створення TumblingEventTimeWindows з тривалістю 1 хвилина. Вікна запускаються та обробляються на основі прогресії водяних знаків.

8. Визначити аномалії в показниках компонентів електромережі у визначеному вікні з використанням «Правила двох сигм»: для нормального розподілу приблизно 95% точок даних знаходяться в межах двох стандартних відхилень від середнього значення. Дані, що не знаходяться у визначених межах позначаються як аномалії.

8.1. Зберегти повідомлення із деталями помилки в базі даних у таблиці «alert_messages».

9. З’єднати потоки bus_data_stream та gen_data_stream, використовуючи логіку left join: до даних про шину у сценарії та похідному сценарії додати інформацію про підключені генератори та сформувати представлення про вузли графу, обчислюються поля P = PG–PD, Q = PD–QD.

10. Сформувати представлення про ребра на основі branch_data_stream.

11. Зберегти інформацію про вузли та ребра графу в таблицях бази даних.

Описаний процес обробки даних про роботу електромережі у визначених сценаріях запускається в завданні «Power Grid», що складається із 14 задач. Інформацію про завдання можливо перевірити в Apache Flink Dashboard, зображено на рисунку 4.16.

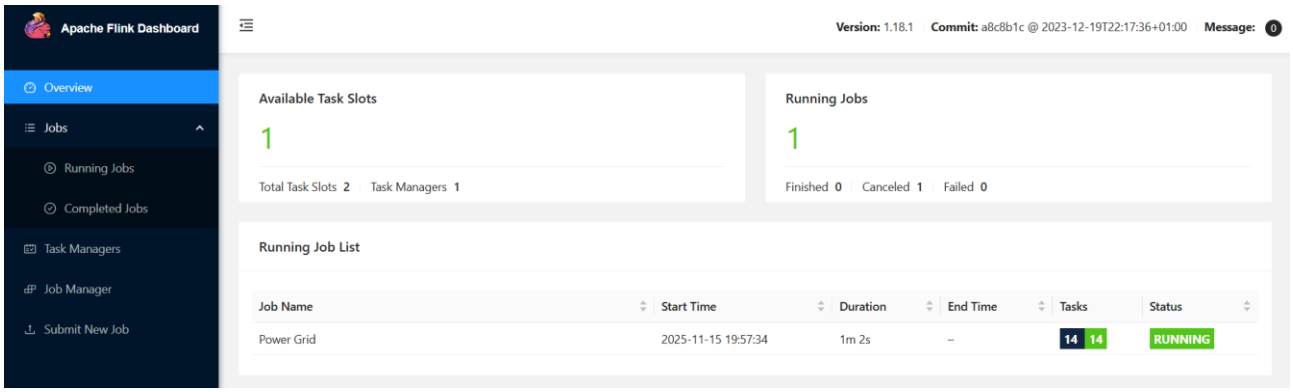

Рисунок 4.16 – Запущене завдання «Power Grid» із задачами

Завдання обробки даних про шини електромережі:

– зчитування даних з Kafka Bus topic, фільтрування, трансформація, визначення часу події, формування водяного знаку, встановлення ключа. Source: Kafka Bus topic -> Map, Filter, Map, Map -> (Map, Filter, Map -> Sink: Unnamed, Extract-Timestamp -> Timestamps/Watermarks -> Remove-Timestamp -> (_stream_key_by_map_operator, _stream_key_by_map_operator));

– визначення аномалій у значеннях «PD» у вікні даних (Tumbling windows) для шин та збереження повідомлення про помилку в таблиці бази даних. TumblingEventTimeWindows, Map -> Sink: bus_PD_anomaly;

– визначення аномалій у значеннях «QD» у вікні даних (Tumbling windows) для шин та збереження повідомлення про помилку в таблиці бази даних. TumblingEventTimeWindows, Map -> Sink: bus_QD_anomaly;

– визначення аномалій у значеннях «VM» у вікні даних (Tumbling windows) для шин та збереження повідомлення про помилку в таблиці бази даних. TumblingEventTimeWindows, Map -> Sink: bus_VM_anomaly;

– визначення аномалій у значеннях «VA» у вікні даних (Tumbling windows) для шин та збереження повідомлення про помилку в таблиці бази даних. TumblingEventTimeWindows, Map -> Sink: bus_VA_anomaly;

– з’єднання даних з потоків шин та генераторів по визначеному ключу (до даних шин додаються дані генераторів, які підключені до них), збереження даних про вузли мережі. Keyed Co-Process -> Sink: node_data.

Завдання обробки даних про генератори електромережі:

– зчитування даних з Kafka Gen topic, фільтрування, трансформація, визначення часу події, формування водяного знаку, встановлення ключа. Source: Kafka Gen topic -> Map, Filter, Map, Map -> (Map, Filter, Map -> Sink: Unnamed, Extract-Timestamp -> Timestamps/Watermarks -> Remove-Timestamp -> (_stream_key_by_map_operator, _stream_key_by_map_operator));

– визначення аномалій у значеннях «PG» у вікні даних (Tumbling windows) для генераторів та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: gen_PG_anomaly;

– визначення аномалій у значеннях «QG» у вікні даних (Tumbling windows) для генераторів та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: gen_QG_anomaly.

Завдання обробки даних про гілки електромережі:

– зчитування даних з Kafka Branch topic, фільтрування, трансформація, визначення часу події, формування водяного знаку, встановлення ключа, збереження даних про ребра мережі. Source: Kafka Branch topic -> Map, Filter, Map, Map -> (Map, Filter, Map -> Sink: alarms_invalid_branch, Extract-Timestamp -> Timestamps/Watermarks -> Remove-Timestamp -> (_stream_key_by_map_operator, Map -> Sink: edge_data));

– визначення аномалій у значеннях «PF» у вікні даних (Tumbling windows) для гілок та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: branch_PF_anomaly;

– визначення аномалій у значеннях «QF» у вікні даних (Tumbling windows) для гілок та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: branch_QF_anomaly;

– визначення аномалій у значеннях «PT» у вікні даних (Tumbling windows) для гілок та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: branch_PT_anomaly;

– визначення аномалій у значеннях «QT» у вікні даних (Tumbling windows) для гілок та збереження повідомлення про помилку в таблиці базі даних. TumblingEventTimeWindows, Map -> Sink: branch_QT_anomaly.

Механізм контрольних точок Apache Flink (checkpointing mechanism) забезпечує відмовостійкість, створюючи періодичні знімки стану завдання. Для завдання «Power Grid» встановленні налаштування для контрольних точок, зображено на рисунку 4.17:

– контрольна точка створюється кожні 100 секунд;

– встановити режим контрольних точок: EXACTLY_ONCE;

– пауза між контрольними точками становить не менше 5 секунд.

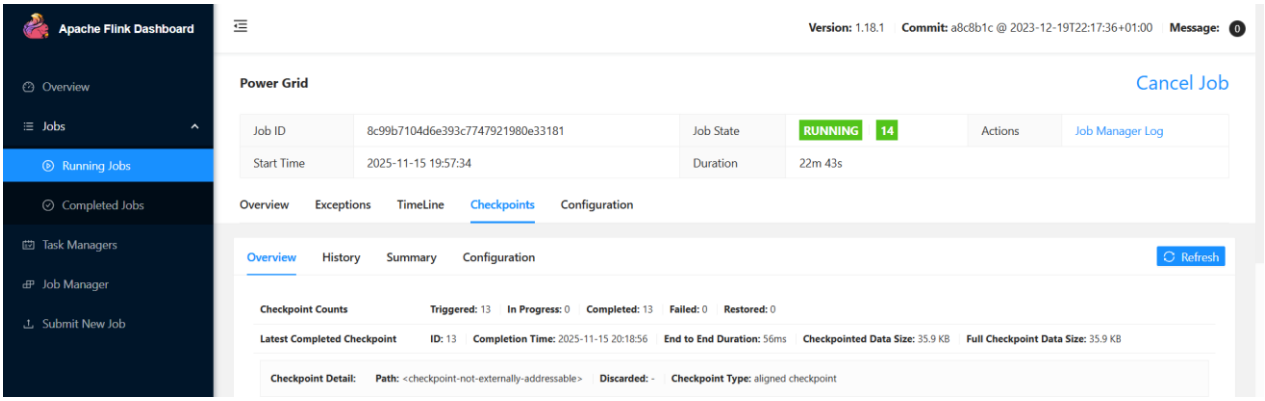

Рисунок 4.17 – Контрольні точки в завданні «Power Grid»

4.5. Тестування обробки даних про роботу електромережі

Для тестування була створена електромережа case9. Мережа складається з таких компонентів:

– 9 шин: bus1, bus2, bus3, bus4, bus5, bus6, bus7, bus8, bus9;

– 3 генераторів: gen1, gen2, gen3. Генератор gen1 підключений до шини bus1, gen2 – bus2, gen3 – bus3;

– 9 гілок: branch1, branch2, branch3, branch4, branch5, branch6, branch7, branch8, branch9. Гілка branch1 з'єднує шини bus1 та bus4, branch2 – bus4 та bus5, branch3 – bus5 та bus6, branch4 – bus3 та bus6, branch5 – bus6 та bus7, branch6 – bus7 та bus8, branch7 – bus8 та bus2, branch8 – bus8 та bus9, branch9 – bus9 та bus4.

У результаті оброблення даних створюється граф, який складається із вершин та ребер, що описують стан компонентів електромережі в кроці сценарію. Фрагменти даних про вершини графу та ребра графу наведені на рисунку 4.18 та рисунку 4.19 відповідно.

scenario_id integer	sub_scenario_id integer	bus_id integer	p double precision	q double precision	vm double precision	va double precision	status integer	message_timestamp bigint
39	1	1	-0	-0	1.0399999618530273	0	0	1763248214000
39	1	2	-0	-0	1.024999976158142	9.28000545501709	1	1763248214000
39	1	3	-0	-0	1.024999976158142	4.6647515296936035	1	1763248214000
39	1	4	-0	-0	1.025788426399231	-2.216787815093994	0	1763248214000
39	1	5	-90	-30	1.0126543045043945	-3.687396287918091	1	1763248214000
39	1	6	-0	-0	1.0323529243469238	1.9667160511016846	1	1763248214000
39	1	7	-100	-35	1.0158826112747192	0.7275360822677612	0	1763248214000
39	1	8	-0	-0	1.0257693529129028	3.719701051712036	0	1763248214000
39	1	9	-125	-50	0.9956308603286743	-3.9888052940368652	0	1763248214000

Рисунок 4.18 – Фрагмент даних про вершини графу

scenario_id integer	sub_scenario_id integer	branch_id integer	pf double precision	qf double precision	pt double precision	qt double precision	status integer	message_timestamp bigint
39	1	1	71.64102172851562	27.045923233032227	-71.64102172851562	-23.923126220703125	1	1763248214000
39	1	2	30.703670501708984	1.0300064086914062	-30.537261962890625	-16.543365478515625	1	1763248214000
39	1	3	-59.462738037109375	-13.456634521484375	60.816585540771484	-18.0748348236084	1	1763248214000
39	1	4	85	-10.859708786010742	-85	14.955327033996582	1	1763248214000
39	1	5	24.183414459228516	3.1195085048675537	-24.095417022705078	-24.295822143554688	1	1763248214000
39	1	6	-75.90457916259766	-10.704177856445312	76.37986755371094	-0.7973314523696899	1	1763248214000
39	1	7	-163	9.178149223327637	163	6.653660297393799	1	1763248214000
39	1	8	86.62013244628906	-8.380817413330078	-84.32015991210938	-11.312750816345215	1	1763248214000
39	1	9	-40.67983627319336	-38.68724822998047	40.93735122680664	22.89311981201172	1	1763248214000

Рисунок 4.19 – Фрагмент даних про ребра графу

Оброблені дані зберігаються в базі даних та можуть бути використані в подальшій аналітиці. Під час обробки сценаріїв роботи компонентів електромережі було визначені аномалії в даних, тому були сформовані повідомлення для користувача, що зображені на рисунку 4.20.

scenario_id integer	sub_scenario_id integer	element_type character varying (25)	element_id integer	element_param_name character varying (25)	element_param_value double precision	alert_message character varying (255)	message_timestamp bigint
40	2	bus	8	VA	0	Anomaly.	1763248256000
41	2	gen	2	PG	0	Anomaly.	1763248277000
41	2	branch	5	QT	0	Anomaly.	1763248277000
41	2	branch	6	QF	0	Anomaly.	1763248277000
41	2	bus	5	PD	0	Anomaly.	1763248277000
41	2	bus	7	PD	0	Anomaly.	1763248277000
41	2	bus	9	PD	0	Anomaly.	1763248277000
41	2	branch	8	PF	0	Anomaly.	1763248277000
41	2	branch	7	PF	0	Anomaly.	1763248277000
41	2	branch	8	PT	0	Anomaly.	1763248277000

Рисунок 4.20 – Приклад сформованих повідомлень із аномальними даними в сценаріях роботи електромережі

Для графічного представлення даних використовується інструмент Apache Superset [215], який має велику кількість налаштувань для дослідження та візуалізації даних, що полегшує їхнє сприйняття. На рисунку 4.21 зображено приклад агрегації кінцевих станів 9 шин у сценаріях: синій колір – загальна кількість сценаріїв, де використовується шина; фіолетовий колір – кількість сценаріїв, де шина була виведена з ладу; зелений колір – кількість сценаріїв, де шина не була виведена із ладу. Візуалізація даних полегшує процес дослідження даних, пошук аномалій, патернів, на які потрібно звернути увагу.

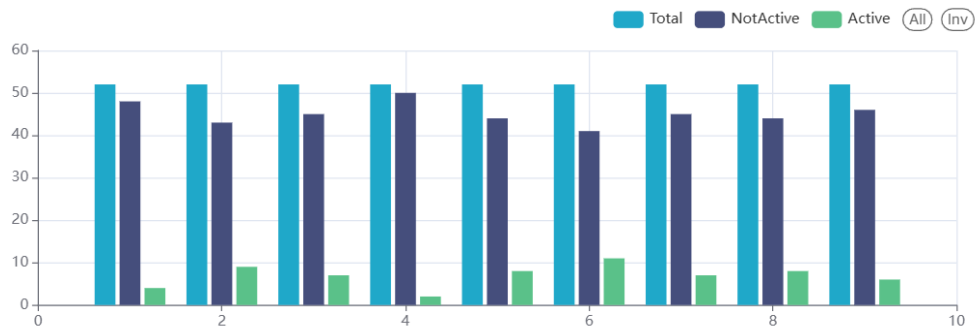

Рисунок 4.21 – Приклад візуалізації даних роботи електромережі

4.6. Підготовка даних та використання моделі нейронної мережі

Під час підготовки набору даних, а також створення та навчання моделі нейронної мережі використовуються бібліотеки PyTorch [216] та PyTorch Geometric [217]. Обробка даних для формування представлення про електромережі в сценаріях зображена на рисунку 4.22.

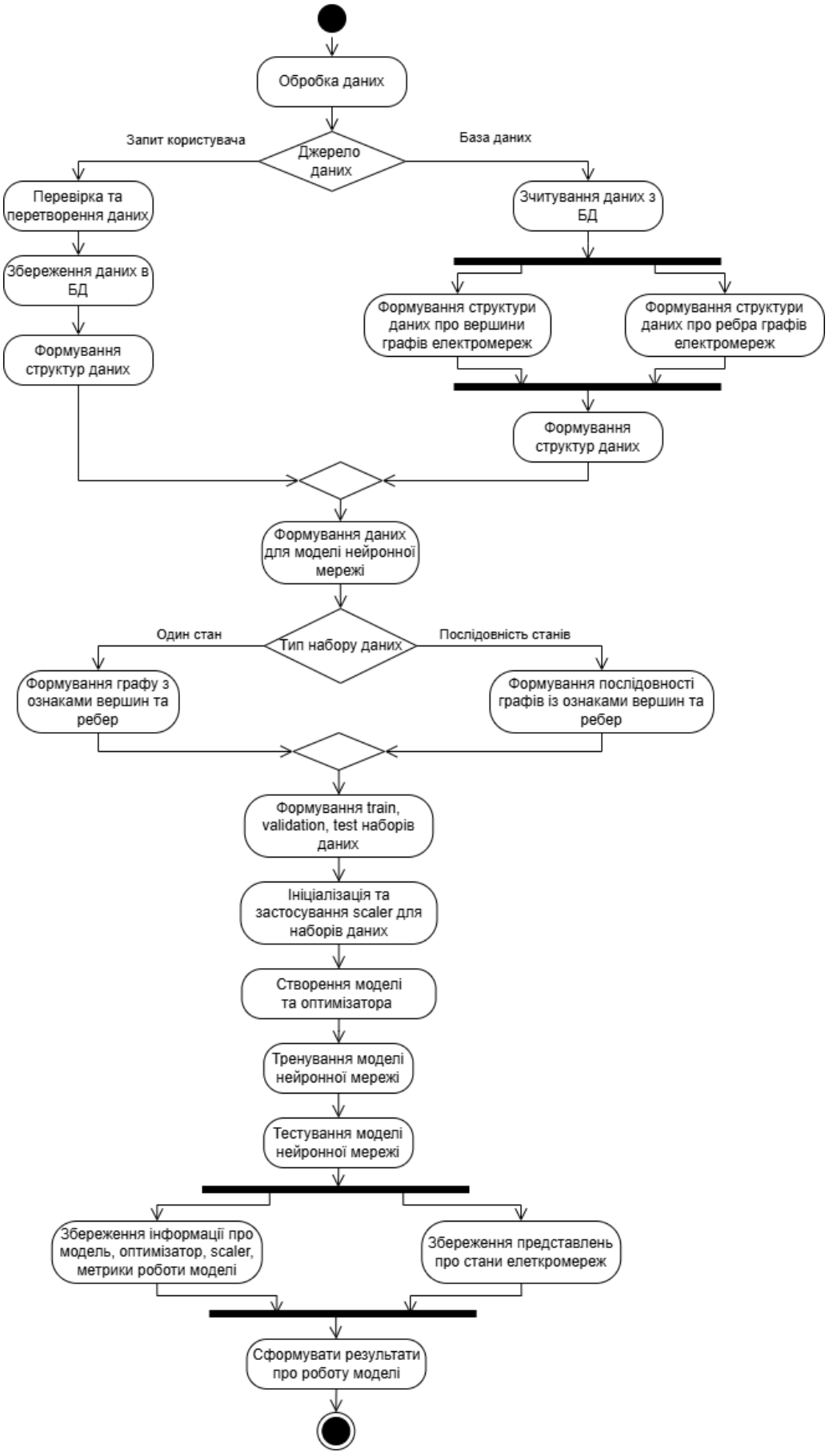

Рисунок 4.22 – Діаграма діяльності обробки даних для формування представлення про електромережі в сценаріях

Завдання обробки даних про роботу електромереж у сценаріях створюється, коли користувач надсилає Post запит /api/models/data, та складається з кроків:

– зчитати вхідні даних;

– перевіри, доповнити та трансформувати дані;

– зберегти дані в базі даних.

Завдання на навчання моделі нейронної мережі вказаного типу створюється, коли користувач надсилає Post запит /api/models/train. У процесі тренування моделі нейронної мережі використовується розподілене блокування для запобігання одночасного тренування типу моделі. Процес навчання моделі нейронної мережі складається із кроків:

– зчитати наявні сценарії про роботу електромережі з бази даних;

– трансформувати дані до необхідної структури;

– розділити дані на набори даних (Training Set, Validation Set, Test Set).

– нормалізувати даних;

– створити моделі нейронної мережі, визначити гіперпараметри;

– запустити тренування моделі нейронної мережі;

– запустити тестування моделі нейронної мережі;

– зберегти дані про модель, результати тренування та тестування;

– зберегти представлення про роботу електромереж у сценаріях.

Визначення подібних представлень здійснюється, коли користувач надсилає Post запит /api/models/embeddings з відповідними параметрами, та складається із кроків:

– зчитати наявні сценарії про роботу електромережі із бази даних;

– трансформувати дані до необхідної структури;

– завантажити модель, скейлер для вузлів та ребер графу;

– підготувати дані для моделі;

– сформувати представлення про роботу електромереж у сценаріях;

– знайти m подібних представлень у базі даних;

– сформувати відповідь з даними про подібні стани електромереж.

Детальна інформація про стан електромережі в сценарії доступна по запиту Get /api/models/scenarios з відповідними параметрами. Відповідь формується на основі даних в базі даних. Додатково користувач може переглянути інформацію про тренування моделі та сформовані представлення про роботу електромереж у сценаріях за допомогою відповідних Get запитів до API: /api/models/checkpoints та /api/models/embeddings. За необхідності користувач може сформувати онтологічну модель про сценарій роботи електромережі.

4.7. Висновки до розділу 4

Розроблено архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, яка відрізняється від наявних порівнянням та визначенням подібності станів електромереж у сценаріях на основі розроблених методів та моделей автоенкодера, що дозволяє розробити програмне забезпечення, яке надає інформацію для прийняття рішення. Програмне забезпечення складається із компонентів, які мають визначений функціонал для виконання завдань:

– вхідна точка, зворотний проксі-сервер, балансування навантаження, щоб забезпечити високу продуктивність при обробці запитів;

– мікросервіс автентифікації та авторизації, який використовує JWT токен із асиметричним шифруванням для підвищення безпеки;

– мікросервіс обробки даних про сценарії роботи електромереж, тренування моделей нейронних мереж для формування представлень про стани електромереж, що використовуються при визначенні подібних представлень.

Функціонал програмного забезпечення надає можливість: проводити експерименти зі сценаріями роботи електромереж; будувати онтологічні моделі сценаріїв роботи електромереж; створювати та тренувати моделі з різними конфігураціями для формування представлень про роботу електромереж; формувати базу даних про роботу електромереж, що використовується при прийнятті рішень на основі визначення подібності представлень.

ВИСНОВКИ

У результаті дисертаційного дослідження вирішено актуальне наукове завдання розробки науково-методичного апарату та програмного забезпечення семантичного моделювання розвитку каскадних ефектів у критичних інфраструктурах та їхнього порівняння для визначення альтернативних сценаріїв на основі представлення з графових нейронних мереж.

В дисертаційній роботі одержані такі результати:

1. Проведено дослідження розвитку та впливу каскадних ефектів на роботу критичної інфраструктури, зокрема електромереж. В результаті було визначено, що каскадний ефект розвивається не лінійно та стрімко. Нелінійність процесу ускладнює прогнозування його розвитку. Під час аналізу стійкості та вразливості в електромережах використовуються топологічні та гібридні підходи для дослідження властивостей мережі. Велика частина метрик графів надає статичну оцінку стійкості системи, не відображає динамічні зміни компонентів системи при виникненні збоїв, а результати аналізу залежать від якості початкових даних сценаріїв збоїв. Мережі Баєса, мережі Петрі, ланцюги Маркова використовуються для аналізу каскадних збоїв, але при моделюванні великої кількості комплексних взаємозалежностей відбувається спрощення сценарію для зменшення складності обчислення, що знижує точність аналізу. Ефективність роботи моделей залежить від якості та адекватності спрощень даних. Значна частина наявних програмних засобів для аналізу каскадних ефектів (ефектів доміно) є складною в налаштуванні та має обмеження для обробки та аналізу даних в режимі реального часу.

2. Уперше розроблено метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі моделі потоку потужності, що базується на використанні онтологічної моделі та множині правил для формалізації, перевірки, доповнення даних про структуру електромережі та процеси розвитку каскадних ефектів у сценаріях. Розроблений метод дозволяє під час попереднього аналізу даних сценаріїв про каскадні ефекти в роботі

електромереж виявляти протиріччя й помилки, що можуть бути усунені на ранньому етапі для покращення якості даних під час використання методів машинного навчання.

3. Уперше розроблено метод визначення подібності сценаріїв каскадних ефектів в електромережах на основі моделі автоенкодера та косинуса подібності, що відрізняється від подібних семантикою взаємодії між об'єктами та їхніми значеннями при порівнянні представлень про стани систем, що дозволяє враховувати структуру та вплив змін в роботі компонентів електромереж.

4. Удосконалено моделі автоенкодера для створення представлень про крок та послідовність кроків роботи електромереж у сценаріях на основі графових нейронних мереж, які, на відміну від наявних, враховують семантику станів компонентів. Удосконалена модель для створення представлень про стани систем у кроках сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 5,6%, коефіцієнт детермінації ребер у середньому на 27,2%. Удосконалена модель для створення представлень про зміни станів систем у послідовності кроків сценаріїв дозволяє підвищити коефіцієнт детермінації вузлів у середньому на 9,5%, коефіцієнт детермінації ребер у середньому на 0,5%.

5. Уперше розроблено архітектуру програмного забезпечення для проведення експериментів зі сценаріями роботи електромереж, зокрема каскадних ефектів, яка відрізняється від наявних порівнянням та визначенням подібності станів електромереж у сценаріях на основі розроблених методів та моделей автоенкодера, що дозволяє розробити програмне забезпечення, яке надає інформацію для прийняття рішення.

6. Практичне значення одержаних результатів в дисертаційній роботі полягає в тому, що вперше розроблено програмне забезпечення аналізу каскадних ефектів в електромережах, де використовуються такі методи: метод семантичного моделювання сценаріїв каскадних збоїв в електромережах; метод визначення подібності сценаріїв каскадних ефектів в електромережі на основі графової нейронної мережі та косинуса подібності. Енкодер натренованої моделі нейронної мережі використовується для створення представлень про роботу

електромереж у сценаріях, що порівнюються за допомогою коефіцієнта подібності на основі косинуса подібності із врахуванням семантики та значень ознак. Крім того, отримані представлення використовуються для визначення множини подібних станів електромереж на основі кластеризації. Одержані результати можуть використовуватися в освітньому процесі: матеріали для лекцій, лабораторних робіт. Програмне забезпечення може застосовуватися як система для прийняття рішень на основі наявних сценаріїв.

7. Створено базу даних, що містить інформацію про проведені експерименти, що використовуються при аналізі роботи моделей. Систематизація даних допомагає визначити приховані закономірності, помилки, аномалії, які важко побачити при поодиноких тестах. На основі накопичених даних про роботу моделей можна виявити ефективні та неефективні налаштування гіперпараметрів. Порівняння роботи моделей з різними конфігураціями можна використати для визначення сильних і слабких сторін моделей та вибрати необхідні налаштування залежно від даних.

8. Результати досліджень впроваджені в Інституті проблем реєстрації інформації НАН України та в навчальному процесі Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»: під час викладання лекційних і лабораторних занять з навчальних дисциплін «Інженерія даних та знань», «Інтелектуальний аналіз даних для задач енергетики».

9. Мета досліджень щодо підвищення ефективності порівняння сценаріїв каскадних ефектів в електромережах та визначення подібних станів систем на основі графових нейронних мереж із врахуванням семантики ознак та зв’язків між об’єктами досягнута та всі часткові завдання вирішені повністю. Наукові результати досліджень є внеском у розвиток теоретичних і прикладних напрямів інформаційних технологій: використання семантичної моделі для формалізації предметної області (сценарії роботи електромереж); використання графових нейронних мереж для аналізу сценаріїв каскадних ефектів (каскадних збоїв) в критичних інфраструктурах (електромережах); використання програмного

забезпечення для обробки даних, що відкриває нові можливості для розв’язання складних задач.

10. Перспективними напрямами досліджень є удосконалення наявних та створення нових методів моделювання комплексних сценаріїв роботи електромереж, які враховують такі чинники: множину подій, що впливають на мережу; варіанти розвитку каскадних збоїв; множину дій для стабілізації роботи мережі. Нові підходи можуть підвищити якість та варіативність даних для навчання нейронних мереж під час аналізу каскадних ефектів. Крім того, в полі зору подальших наукових досліджень має бути розробка нових моделей машинного навчання, які можуть додатково враховувати сторонні чинники в роботі електромереж та адаптуватися до змін у середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kinney R., Crucitti P., Albert R., Latora V. Modeling Cascading Failures in the North American Power Grid. *arXiv*. 2004. https://doi.org/10.48550/arXiv.cond-mat/0410318.

2. Darbra R. M., Palacios A., Casal J. (2010) Domino effect in chemical accidents: Main features and accident sequences. *Journal of Hazardous Materials*. 2010. Vol. 183, Issues 1–3. P. 565-573. https://doi.org/10.1016/j.jhazmat.2010.07.061.

3. Rinaldi, S. M., Peerenboom J. P., Kelly T. K. Identifying, Understanding, and Analyz-ing Critical Infrastructure Interdependenciesю *IEEE Control Systems Magazine*. 2001. Vol. 21, No. 6. P. 11-25. https://doi.org/10.1109/37.969131.

4. Kadri F., Birregah B., Châtelet E. The Impact of Natural Disasters on Critical Infrastructures: A Domino Effect-based Study. *Homeland Security and Emergency Management*. 2014. Vol. 11, Issue 2. P. 217–241. https://doi.org/10.1515/JHSEM-2012-0077.

5. Clini F., Darbra R. M., Casal J. Historical Analysis of Accidents Involving Domino Effect. *Chemical Engineering Transactions*. 2010. Vol. 19. P. 335-340. https://doi.org/10.3303/CET1019055.

6. Chen C., Reniers G., Khakzad N. A thorough classification and discussion of approaches for modeling and managing domino effects in the process industries. *Safety Science.* 2020. Vol. 125. https://doi.org/10.1016/j.ssci.2020.104618.

7. Dobeš P., Martiníková B., Klecka V., Lepík P., Novotný P., Danihelka P. Review of selected chemical accident report sites regarding major accident investigation and lessons learnt needs. *Chemical Engineering Transactions*. 2022. Vol. 90. P. 613-618. https://doi.org/10.3303/CET2290103.

8. Baldick R. et al. Initial review of methods for cascading failure analysis in electric power transmission systems IEEE PES CAMS task force on understanding, prediction, mitigation and restoration of cascading failures. *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*. 2008. P. 1-8. https://doi.org/10.1109/PES.2008.4596430.

9. Koç Y., Verma T., Araujo N. A. M., Warnier, M. Matcasc: A tool to analyse cascading line outages in power grids. *arXiv.* 2013. https://doi.org/10.48550/arXiv.1308.0174.
10. Hines P., Balasubramaniam K., Sanchez E. C. Cascading failures in power grids. *IEEE Potentials*. 2009. Vol. 28, Issue 5. P. 24-30. https://doi.org/10.1109/MPOT.2009.933498.
11. Petermann T., Bradke H., Lüllmann A., Poetzsch M., Riehm U. What happens during a blackout: Consequences of a prolonged and wide-ranging power outage. 2011. https://doi.org/10.5445/IR/1000103292.
12. Yadav M. K. BPR Function in Transportation Network Analysis. 2023. URL: https://blog.truegeometry.com/app/geometry/2023/06/28/BPRFunctionNetworkAnalysis.html (дата звернення: 11.12.2025).
13. Sami N. Md, Naeini M. Machine Learning Applications in Cascading Failure Analysis in Power Systems: A Review. *arXiv.* 2023. https://doi.org/10.48550/arXiv.2305.19390.
14. Nan C., Sansavini G., Kröger W., Heinimann H. R. A Quantitative Method for Assessing the Resilience of Infrastructure Systems. *Probabilistic Safety Assessment and Management PSAM*. 12, June 2014, Honolulu, Hawaii.
15. Poulin C., Kane M. Infrastructure Resilience Curves: Performance Measures and Summary Metrics. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2102.01009.
16. Panteli M., Mancarella P., Trakas D. N., Kyriakides E., Hatziargyriou N. D. Metrics and Quantification of Operational and Infrastructure Resilience in Power Systems. *IEEE Transactions on Power Systems*. 2017. Vol. 32, Issue. 6, P. 4732–4742. https://doi.org/10.1109/TPWRS.2017.2664141.
17. Bruneau M., Chang S. E., Eguchi R. T., Lee G. C., O'Rourke T. D., Reinhorn A. M., Shinozuka M., Tierney K., Wallace W. A., von Winterfeldt D. A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities. *Earthquake Spectra*. 2003, Vol. 19, Issue 4. https://doi.org/10.1193/1.1623497.
18. Narimani M. R., Huang H., Umunnakwe A., Mao Z., Sahu A., Zonouz S., Davis K. Generalized Contingency Analysis Based on Graph Theory and Line Outage Distribution Factor. *arXiv*. 2020. https://doi.org/10.48550/arXiv.2007.07009.

19. Rath A. Indian Blackouts of July 2012: What Happened and Why? *Medium*. 2016. URL: https://medium.com/clean-energy-for-billions/indian-blackouts-of-july-2012-what-happened-and-why-639e31fb52ad (дата звернення: 11.12.2025).
20. Lai L. L., Zhang H. T., Mishra S., Ramasubramanian D., Lai C. S., Xu F. Y. Lessons learned from July 2012 Indian blackout. *9th IET International Conference on Advances in Power System Control, Operation and Management (APSCOM 2012)*. 2012. P. 1-6. https://doi.org/10.1049/cp.2012.2173.
21. Report on the grid disturbance on 30th July 2012 and grid disturbance on 31st July 2012. URL: https://www.cercind.gov.in/2012/orders/Final_Report_Grid_Disturbance.pdf (дата звернення: 11.12.2025).
22. North American Electric Reliability Council. Technical Analysis of the August 14, 2003, Blackout What Happened, Why, and What Did We Learn. URL: https://www.nerc.com/pa/rrm/ea/August%2014%202003%20Blackout%20Investigation%20DL/NERC_Final_Blackout_Report_07_13_04.pdf (дата звернення: 11.12.2025).
23. FINAL REPORT of the Investigation Committee on the 28 September 2003 Blackout in Italy. URL: https://eepublicdownloads.entsoe.eu/clean-documents/pre2015/publications/ce/otherreports/20040427_UCTE_IC_Final_report.pdf (дата звернення: 11.12.2025).
24. BLACK SYSTEM SOUTH AUSTRALIA 28 SEPTEMBER 2016. URL: https://www.aemo.com.au/-/media/Files/Electricity/NEM/Market_Notices_and_Events/Power_System_Incident_Reports/2017/Integrated-Final-Report-SA-Black-System-28-September-2016.pdf (дата звернення: 11.12.2025).
25. Report on Blackout in Turkey on 31st March 2015. URL: https://eepublicdownloads.entsoe.eu/clean-documents/SOC%20documents/Regional_Groups_Continental_Europe/20150921_Black_Out_Report_v10_w.pdf (дата звернення: 11.12.2025).

26. Report on damages to infrastructure from the destruction caused by Russia's military aggression against Ukraine as of January 2024. URL: https://kse.ua/wp-content/uploads/2024/05/Eng_01.01.24_Damages_Report.pdf (дата звернення: 11.12.2025).
27. Робота енергосистеми України станом на 20 травня 2022 року. URL: https://www.kmu.gov.ua/news/robota-energosistemi-ukrayini-stanom-na-20-travnya-2022-roku (дата звернення: 11.12.2025).
28. Робота енергосистеми України на 1 червня 2022 року. URL: https://www.kmu.gov.ua/news/robota-energosistemi-ukrayini-na-1-chervnya-2022-roku (дата звернення: 11.12.2025).
29. Робота енергосистеми України на 10 червня 2022 року. URL: https://mspu.gov.ua/news/robota-energosistemi-ukrayini-na-10-chervnya-2022-roku (дата звернення: 11.12.2025).
30. Робота енергосистеми України станом на 15 червня 2022 року. URL: https://www.kmu.gov.ua/news/robota-energosistemi-ukrayini-stanom-na-15-chervnya-2022-roku (дата звернення: 11.12.2025).
31. Робота енергосистеми України на 20 червня 2022 року. URL: https://mspu.gov.ua/news/robota-energosistemi-ukrayini-na-20-chervnya-2022-roku (дата звернення: 11.12.2025).
32. Робота енергосистеми України на 1 вересня 2022 року. URL: https://mev.gov.ua/statystychna-informatsiya/robota-enerhosystemy-ukrayiny-na-1-veresnya-2022-roku (дата звернення: 11.12.2025).
33. Робота енергосистеми України на 27 вересня 2022 року. URL: https://mev.gov.ua/statystychna-informatsiya/robota-enerhosystemy-ukrayiny-na-27-veresnya-2022-roku (дата звернення: 11.12.2025).
34. Velay M., Vinyals M., Besanger Y., Retiere N. An analysis of large-scale transmission power blackouts from 2005 to 2016. *2018 53rd International Universities Power Engineering Conference (UPEC)*. P. 1-6. https://doi.org/10.1109/UPEC.2018.8541901.

35. Hines P., Apt J., Talukdar S. Trends in the history of large blackouts in the United States. *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*. 2008. P. 1-8. https://doi.org/10.1109/PES.2008.4596715.

36. ENTSO-E Grid Disturbance Definitions for the Power System Above 100 kV. URL: https://eepublicdownloads.entsoe.eu/clean-documents/SOC%20documents/ENTSO-E_Grid_Disturbance_Definitions_for_the_Power_System_above_100_kV_-_to_be_published_version__1_.pdf (дата звернення: 11.12.2025).

37. Stankovski A., Gjorgiev B., Locher L., Sansavini G. Power blackouts in Europe: Analyses, key insights, and recommendations from empirical evidence. *Joule*. 2023. Vol. 7, Issue 11. P. 2468-2484. https://doi.org/10.1016/j.joule.2023.09.005.

38. Толюпа С. В., Кулько А. А. Нейро-нечітка системи виявлення вторгнень у інформаційну мережу критичної інфраструктури. *Кібербезпека: освіта, наука, техніка*. 2025. Т. 3, № 27. С. 233–247. https://doi.org/10.28925/2663-4023.2025.27.750.

39. Сенченко В. Р., Бойченко А. В., Коваль О. В., Бисько Р. М., Хоменко О. М. Огляд методів і технологій сценарного аналізу каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2024. Т. 26, № 1. С. 24–54. https://doi.org/10.35681/1560-9189.2024.26.1.308506.

40. Сенченко В. Р., Бойченко А. В., Коваль О. В., Хоменко О. М. Методологія та архітектура платформи моделювання взаємозалежностей у критичних інфраструктурах при виникненні каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2025. Т. 27. № 1. С. 28-41. https://doi.org/10.35681/1560-9189.2025.27.1.335624.

41. Хоменко О. М., Сенченко В. Р., Коваль О. В. Мережевий підхід при дослідженні каскадних ефектів критичних інфраструктур. *Реєстрація, зберігання і обробка даних*. 2024. Т. 26. № 2. С. 44-72. https://doi.org/10.35681/1560-9189.2024.26.2.316908.

42. Estrada E. Graph and Network Theory in Physics. *arXiv*. 2013. https://doi.org/10.48550/arXiv.1302.4378.

43. Hernandez J. M., Van Mieghem P. Classification of graph metrics. 2011. URL: https://www.nas.ewi.tudelft.nl/people/Piet/papers/TUDreport20111111_MetricList.pdf (дата звернення: 11.12.2025).

44. Oehlers M., Fabian B. Graph Metrics for Network Robustness—A Survey. *Mathematics*. 2021. Vol. 9, Issue 8. https://doi.org/10.3390/math9080895.

45. Di Nardo A., Giudicianni C., Greco R., Herrera M., Santonastaso G. F. Applications of graph spectral techniques to water distribution network management. *Water*. 2018. Vol. 10, No. 1. https://doi.org/10.3390/w10010045.

46. Puuska S., Kansanen K., Rummukainen L., Vankka J. Modelling and real-time analysis of critical infrastructure using discrete event systems on graphs. *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*. 2015. P. 1-5. https://doi.org/10.1109/THS.2015.7225330.

47. Wan Z., Mahajan Y., Kang B. W., Moore T. J., Cho J. A Survey on Centrality Metrics and Their Implications in Network Resilience. *arXiv*. 2020. https://doi.org/10.48550/arXiv.2011.14575.

48. Stergiopoulos G., Kotzanikolaou P., Theocharidou M., Gritzalis D. Risk mitigation strategies for critical infrastructures based on graph centrality analysis. *International Journal of Critical Infrastructure Protection*. 2015. Vol. 10. P. 34-44. https://doi.org/10.1016/j.ijcip.2015.05.003.

49. Mattsson L., Jenelius E. Vulnerability and resilience of transport systems – A discussion of recent research. *Transportation Research Part A: Policy and Practice*. 2015. Vol. 81. P. 16–34. https://doi.org/10.1016/j.tra.2015.06.002.

50. Ellens W., Kooij R.E. Graph measures and network robustness. *arXiv*. 2013. https://doi.org/10.48550/arXiv.1311.5064.

51. Berche B., Von Ferber C., Holovatch T., Holovatch Yu. Resilience of public transport networks against attacks. *arXiv*. 2009. https://doi.org/10.48550/arXiv.0905.1638.

52. Guze S. Graph Theory Approach to the Vulnerability of Transportation Networks. *Algorithms*. 2019. Vol. 12, Issue 12. https://doi.org/10.3390/a12120270.

53. Yazdani A., Jeffrey P. Complex network analysis of water distribution systems. *arXiv*. 2011. https://doi.org/10.48550/arXiv.1104.0121.

54. Rueda D. F., Calle E., Marzo J. L. Robustness Comparison of 15 Real Telecommunication Networks: Structural and Centrality Measurements. *Journal of Network and Systems Management*. 2017. Vol. 25. P. 269–289. https://doi.org/10.1007/s10922-016-9391-y.

55. Xie B., Tian X., Kong L., Chen W. The Vulnerability of the Power Grid Structure: A System Analysis Based on Complex Network Theory. *Sensors*. 2021. Vol. 21, Issue 21. https://doi.org/10.3390/s21217097.

56. Cuadra L., Salcedo-Sanz S., Del Ser J., Jiménez-Fernández S., Geem Z. W. A Critical Review of Robustness in Power Grids Using Complex Networks Concepts. *Energies*. 2015. Vol. 8, Issue 9. https://doi.org/10.3390/en8099211.

57. Arianos S., Bompard E., Carbone A., Xue F. Power grid vulnerability: A complex network approach. *arXiv*. 2009. https://doi.org/10.48550/arXiv.0810.5278.

58. Bompard E., Luo L., Pons E. A perspective overview of topological approaches for vulnerability analysis of power transmission grids. *International Journal of Critical Infrastructures*. 2015. Vol. 11, Issue 1. P. 15-26. https://doi.org/10.1504/IJCIS.2015.067397.

59. Abedi A., Gaudard L., Romerio F. Review of major approaches to analyze vulnerability in power system. *Reliability Engineering & System Safety*. 2019. Vol. 183. P. 153-172. https://doi.org/10.1016/j.ress.2018.11.019.

60. Koç Y., Warnier M., Kooij R. E., Brazier F. M. T. Structural Vulnerability Assessment of Electric Power Grids. *arXiv*. 2013. https://doi.org/10.48550/arXiv.1312.6606.

61. Wang X., Koç Y., Kooij R. E., Van Mieghem P. A network approach for power grid robustness against cascading failures. *arXiv*. 2015. https://doi.org/10.48550/arXiv.1505.06312.

62. Ellens W., Spieksma F. M., Van Mieghem P., Jamakovic A., Kooij R. E. Effective graph resistance. *Linear Algebra and its Applications*. 2011. Vol. 435, Issue 10. P. 2491-2506. https://doi.org/10.1016/j.laa.2011.02.024.
63. Freitas S., Yang D., Kumar S., Tong H., Chau D. H. Graph Vulnerability and Robustness: A Survey. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2105.00419.
64. Holme P., Kim B. J., Yoon C. N., Han S. K. Attack vulnerability of complex networks. *arXiv*. 2002. https://doi.org/10.48550/arXiv.cond-mat/0202410.
65. Vaniš M., Lokaj Z., Šrotýř M. A Novel Algorithm for Merging Bayesian Networks. *Symmetry.* 2023. Vol. 15, Issue 7. https://doi.org/10.3390/sym15071461.
66. Daly R., Qiang S., Aitken S. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review.* 2011. Vol. 26, Issue 2. P. 99-127. https://doi.org/10.1017/S0269888910000251.
67. Khakzad N. A Tutorial on Fire Domino Effect Modeling Using Bayesian Networks. *Modelling.* 2021. Vol. 2, Issue 2. https://doi.org/10.3390/modelling2020013.
68. Srinivas S. A Generalization of the Noisy-Or Model. *arXiv*. 2013. https://doi.org/10.48550/arXiv.1303.1479.
69. Kitson N. K., Constantinou A. C., Guo Z., Liu Y., Chobtham K. A survey of Bayesian Network structure learning. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2109.11415.
70. Scanagatta M., Salmeron A., Stella F. A survey on Bayesian network structure learning from data. *Progress in Artificial Intelligence.* 2019. Vol. 8. P. 425-439. https://doi.org/10.1007/s13748-019-00194-y.
71. Cerotti D., Codetta-Raiteri D., Dondossola G., Egidi L., Franceschinis G., Portinale L., Terruggia R. Evidence-Based Analysis of Cyber Attacks to Security Monitored Distributed Energy Resources. *Applied Sciences*. 2020, Vol. 10, Issue 14. https://doi.org/10.3390/app10144725.
72. Wright M., Chizari H., Viana T. A Systematic Review of Smart City Infrastructure Threat Modelling Methodologies: A Bayesian Focused Review. *Sustainability*. 2022. Vol. 14, Issue 16. https://doi.org/10.3390/su141610368.

73. Wang Y., Yu J., Baroud H. A Bayesian Approach to Reconstructing Interdependent Infrastructure Networks from Cascading Failures. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2211.15590.
74. Wang Y., Yu J., Baroud H. Generating synthetic systems of interdependent critical infrastructure networks. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2111.12742.
75. Eldosouky A., Saad W., Mandayam N. Resilient Critical Infrastructure: Bayesian Network Analysis and Contract-Based Optimization. *arXiv*. 2017. https://doi.org/10.48550/arXiv.1709.00303.
76. Murata T. 1989. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*. 1989. Vol. 77, Issue 4. P. 541–580. https://doi.org/10.1109/5.24143.
77. He X. A comprehensive survey of petri net modeling in software engineering. *International Journal of Software Engineering and Knowledge Engineering*. 2013. Vol. 23, No. 5. P. 589-625. https://doi.org/10.1142/S021819401340010X.
78. Ge M., Rossi B., Chren S., Blanco J. M. Petri Nets for Smart Grids: The Story So Far. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2401.05462.
79. Kamil M. Z., Taleb-Berrouane M., Khan F., Ahmed S. Dynamic domino effect risk assessment using Petri-nets. *Process Safety and Environmental Protection*. 2019. Vol. 124. P. 308-316. https://doi.org/10.1016/j.psep.2019.02.019.
80. Yin X., Li L., Liu Q. A Study on the Vulnerability Cascade Propagation of Integrated Energy Systems in the Transportation Industry Based on the Petri Network. *Energies.* 2022. Vol. 15, Issue 12. https://doi.org/10.3390/en15124320.
81. Li R., Decocq B., Barros A., Fang Y., Zeng Z. Petri Net-Based Model for 5G and Beyond Networks Resilience Evaluation. *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*. 2022. P. 131-135. https://doi.org/10.1109/ICIN53892.2022.9758134.
82. Lu W., Besanger Y., Zamai E., Radu D. Blackouts: Description, analysis and classification. *Proceedings of the 6th WSEAS International Conference on Power Systems,* Lisbon, Portugal, September 22-24, 2006.
83. Zhai C. A Robust Optimization Approach for Terminating the Cascading Failure of Power Systems. *arXiv*. 2019. https://doi.org/10.48550/arXiv.1907.13452.

84. Nakarmi U., Rahnamay-Naeini M. A markov chain approach for cascade size analysis in power grids based on community structures in interaction graphs. *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. 2020. P. 1–6. https://doi.org/10.1109/PMAPS47429.2020.9183579.

85. Acharya S., Lee B. S., Hines P. Real-time top-k predictive query processing over event streams. *arXiv*. 2015. https://doi.org/10.48550/arXiv.1508.06976.

86. Yao R., Huang S., Sun K., Liu F., Zhang X., Mei S., Wei W., Ding L. Risk assessment of multi-timescale cascading outages based on Markovian tree search. *arXiv*. 2016. https://doi.org/10.48550/arXiv.1603.03935.

87. Yao R., Sun K., Liu F., Mei S. Management of Cascading Outage Risk Based on Risk Gradient and Markovian Tree Search. *arXiv*. 2017. https://doi.org/10.48550/arXiv.1704.02556.

88. Yusta J. M., Correa G. J., Lacal- Arántegui R. Methodologies and applications for critical infrastructure protection: State-of-the-art. *Energy Policy*. 2011. Vol. 39, Issue 10. P. 6100–6119. https://doi.org/10.1016/j.enpol.2011.07.010.

89. Farid Kadri, Eric Chatelet. Domino Effect Analysis and Assessment of Industrial Sites: A Review of Methodologies and Software Tools. International Journal of Computers and Distributed Systems, 2013, 02 (III), pp.1-10. ffhal-01026495f

90. Freitas S., Yang D., Kumar S., Tong H., Chau D. H. Evaluating Graph Vulnerability and Robustness using TIGER. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2006.05648.

91. Chadaga S., Wu X., Modiano E. Power Failure Cascade Prediction using Graph Neural Networks. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2404.16134.

92. Varbella A., Gjorgiev B., Sansavini G. Geometric deep learning for online prediction of cascading failures in power grids. *Reliability Engineering & System Safety*. 2023. Vol. 237. https://doi.org/10.1016/j.ress.2023.109341.

93. Liu Y., Zhang N., Wu D., Botterud A., Yao R., Kang C. Searching for Critical Power System Cascading Failures With Graph Convolutional Network. *IEEE Transactions on Control of Network Systems*. 2021. Vol. 8, Issue 3. P. 1304-1313. https://doi.org/10.1109/TCNS.2021.3063333.

94. Huang S., Qi J. Learning Cascading Failure Interactions by Deep Convolutional Generative Adversarial Network. *2022 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 2022, P. 21-26. https://doi.org/10.1109/SmartGridComm52983.2022.9961045.

95. Історія енергетики. URL: https://mev.gov.ua/storinka/istoriya-enerhetyky (дата звернення: 11.12.2025).

96. ENTSO-E Transmission System Map. URL: https://www.entsoe.eu/data/map/ (дата звернення: 11.12.2025).

97. The Power System of UKRAINE. URL: https://www.cigre.org/userfiles/files/Community/NC/2018_National-power-system_Ukraine.pdf (дата звернення: 11.12.2025).

98. Basics of Electrical Power Transmission System. URL: https://www.electricaleasy.com/2016/03/basics-of-electrical-power-transmission.html (дата звернення: 11.12.2025).

99. Electric Power Distribution System Basics. URL: https://www.electricaleasy.com/2018/01/electric-power-distribution-system.html (дата звернення: 11.12.2025).

100. Electric Power Distribution System Topologies: Radial, Ring, Parallel & Interconnected. URL: https://www.electricaleasy.com/2018/02/radial-parallel-ring-main-interconneted-distribution.html (дата звернення: 11.12.2025).

101. OpenStreetMap. URL: https://www.openstreetmap.org (дата звернення: 11.12.2025).

102. Medjroubi W., Müller U. P., Scharf M., Matke C., Kleinhans D. Open Data in Power Grid Modelling: New Approaches Towards Transparent Grid Models. *Energy Reports*. 2017. Vol. 3. P. 14-21. https://doi.org/10.1016/j.egyr.2016.12.001.

103. Hörsch J., Hofmann F., Schlachtberger D., Brown T. PyPSA-Eur: An Open Optimisation Model of the European Transmission System. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1806.01613.

104. Xiong B., Fioriti D., Neumann F., Riepin I., Brown T. Modelling the High-Voltage Grid Using Open Data for Europe and Beyond. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2408.17178.
105. PyPSA-Eur: A Sector-Coupled Open Optimisation Model of the European Energy System. URL: https://github.com/PyPSA/pypsa-eur (дата звернення: 11.12.2025).
106. Parzen M., Abdel-Khalek H., Fedorova E., Mahmood M., Frysztacki M. M., Hampp J., Franken L., Schumm L., Neumann F., Poli D., Kiprakis A., Fioriti D. PyPSA-Earth. A New Global Open Energy System Optimization Model Demonstrated in Africa. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2209.04663.
107. PyPSA-Earth. A Flexible Python-based Open Optimisation Model to Study Energy System Futures around the World. URL: https://github.com/pypsa-meets-earth/pypsa-earth (дата звернення: 11.12.2025).
108. Global Power Plant Database. URL: https://datasets.wri.org/datasets/global-power-plant-database (дата звернення: 11.12.2025).
109. OSMnx. URL: https://wiki.openstreetmap.org/wiki/OSMnx (дата звернення: 11.12.2025).
110. Boeing G. Modeling and Analyzing Urban Networks and Amenities with OSMnx. *Geographical Analysis*. 2025. Vol. 57, Issue 4. P. 567-577. doi:10.1111/gean.70009. https://doi.org/10.1111/gean.70009.
111. earth-osm. URL: https://github.com/pypsa-meets-earth/earth-osm (дата звернення: 11.12.2025).
112. Map features. URL: https://wiki.openstreetmap.org/wiki/Map_features (дата звернення: 11.12.2025).
113. Brown T., Hörsch J., Schlachtberger D. PyPSA: Python for Power System Analysis. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1707.09913.
114. PyPSA. URL: https://docs.pypsa.org/latest/ (дата звернення: 11.12.2025).
115. Overview of the 14 UML Diagram Types. URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/overview-of-the-14-uml-diagram-types/ (дата звернення: 11.12.2025).

116. Reggio G., Leotta M., Ricca F., Clerissi D. What are the used UML diagrams? A Preliminary Survey. *Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013)*. Miami, USA, October 1, 2013.

117. What are the SysML diagram types? URL: https://sysml.org/sysml-faq/what-are-sysml-diagram-types.html (дата звернення: 11.12.2025).

118. Gorski V. Scenario Trees Models vs. Lattice Models in Stochastic Optimization. *Master's Thesis, Technische Universitat Munchen*, 2017. URL: https://mediatum.ub.tum.de/doc/1467568/file.pdf (дата звернення: 11.12.2025).

119. Heitsch H., Romisch W. Scenario tree generation for multi-stage stochastic programs. *Stochastic Optimization Methods in Finance and Energy*. 2011. P. 313-342. https://doi.org/10.1007/978-1-4419-9586-5_14.

120. Growe-Kuska N., Heitsch H., Romisch W. Scenario reduction and scenario tree construction for power management problems. *2003 IEEE Bologna Power Tech Conference Proceedings*. 2003. https://doi.org/10.1109/PTC.2003.1304379.

121. The Complete List of BPMN Symbols and Their Meanings. URL: https://creately.com/guides/bpmn-symbols/ (дата звернення: 11.12.2025).

122. BPMN Notation Overview. URL: https://www.visual-paradigm.com/guide/bpmn/bpmn-notation-overview/ (дата звернення: 11.12.2025).

123. DeBellis M. New Protégé Pizza Tutorial. URL: https://www.michaeldebellis.com/post/new-protege-pizza-tutorial (дата звернення: 11.12.2025).

124. Dentler K., Cornet R., ten Teije A., de Keizer N. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web: – Interoperability, Usability, Applicability*. 2011. Vol. 2, Issue 2. P. 71-87. https://doi.org/10.3233/SW-2011-0034.

125. Хала К. О., Гладун А. Я. Розширення можливостей онтологічного моделювання правових знань з допомогою елементів нечіткої логіки. *Cybernetics and Computer Engineering*. 2024. № 4 (218). С. 29-53. https://doi.org/10.15407/kvt218.04.029.

126. Henriques J., Caldeira F., Cruz T., Simões P. On the use of Ontology Data for Protecting Critical Infrastructures. *Journal of Information Warfare*. 2018. Vol. 17, Issue 4.

127. De Rosa F., Maunero N., Nicoletti L., Prinetto P., Trussoni M. Ontology for Cybersecurity Governance of ICT Systems. *Proceedings of the Italian Conference on Cybersecurity (ITASEC 2022)*. P. 52-63. Rome, Italy, June 20-23, 2022.

128. Roman D., Sukhobok D., Nikolov N., Elvesæter B., Pultier A. The InfraRisk Ontology: Enabling Semantic Interoperability for Critical Infrastructures at Risk from Natural Hazards. *The 16th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2017)*. 2017. P. 463-479. https://doi.org/10.1007/978-3-319-69459-7_31.

129. Toscano B., Fernandes A. D., da Silva M. M. A domain ontology on cascading effects in critical infrastructures based on a systematic literature review. *Int. J. Critical Infrastructures*. 2022. Vol. 18, No. 1. P. 79-103. https://doi.org/10.1504/IJCIS.2022.120679.

130. Noy N. F., McGuinness D. L. Ontology Development 101: A Guide to Creating Your First Ontology. URL: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (дата звернення: 11.12.2025).

131. Falbo R., Guizzardi G., Gangemi A., Presutti V. Ontology Patterns: Clarifying Concepts and Terminology. *Proceedings of the 4th Workshop on Ontology and Semantic Web Patterns co-located with 12th International Semantic Web Conference (ISWC 2013)*. Sydney, Australia, October 21, 2013.

132. Poveda M., Suárez-Figueroa M. C., Gómez-Pérez A. Ontology Analysis Based on Ontology Design Patterns. *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009)*. Washington D.C., USA, 25 October, 2009.

133. Peng C., Xia F., Naseriparsa M., Osborne F. Knowledge Graphs: Opportunities and Challenges. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2303.13948.

134. Хоменко О. М., Коваль О. В. Побудова сценаріїв розвитку каскадних збоїв в інфраструктурі електромереж. *Зв'язок.* 2025. №5. С. 3-12. https://doi.org/10.31673/2412-9070.2025.050938.

135 MATPOWER. URL: https://matpower.org/ (дата звернення: 11.12.2025).

136. VeraGrid. URL: https://github.com/SanPen/VeraGrid (дата звернення: 11.12.2025).

137. Sirin E., Parsia B. Pellet: An OWL DL Reasoner. *Proceedings of the DL Home 2004 International Workshop on Description Logics (DL2004)*. Whistler, British Columbia, Canada June 6-8, 2004.

138. Protégé. URL: https://protege.stanford.edu/ (дата звернення: 11.12.2025).

139. Zazuko SHACL Playground. URL: https://github.com/zazuko/shacl-playground (дата звернення: 11.12.2025).

140. Noebels M., Preece R., Panteli M. AC Cascading Failure Model for Resilience Analysis in Power Networks. *IEEE Systems Journal*. 2022. Vol. 16, Issue 1. P. 374-385. https://doi.org/10.1109/JSYST.2020.3037400.

141. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. URL: https://www.w3.org/submissions/SWRL/ (дата звернення: 11.12.2025).

142. Hines P., Cotilla-Sanchez E., Blumsack S. Do topological models provide good information about electricity infrastructure vulnerability? *arXiv*. 2010. https://doi.org/10.48550/arXiv.1002.2268.

143. Korkali M., Veneman J. G., Tivnan B. F., Hines P. D. H. Reducing Cascading Failure Risk by Increasing Infrastructure Network Interdependency. *arXiv*. 2015. https://doi.org/10.48550/arXiv.1410.6836.

144. Chipli S. Automating AC Power Flow Simulations The European Electricity Grid. *Thesis MSc Applied Mathematics. Delft University of Technology*. 2021. URL: https://diamhomes.ewi.tudelft.nl/~kvuik/numanal/chipli_afst.pdf (дата звернення: 11.12.2025).

145. Kootte M. E., Romate J. E., Vuik C. Load flow computations for (integrated) Transmission and Distribution systems: A literature review. *Reports of the Delft Institute of Applied Mathematics. Delft University of Technology.* 2020. URL:

https://pure.tudelft.nl/ws/portalfiles/portal/87138713/TP_MEK_Literature_Review_Energy_Intranets_NEAT.pdf (дата звернення: 11.12.2025).
146. Afolabi O. A., Ali W. H., Cofie P., Fuller J., Obiomon P., Kolawole E. S. Analysis of the Load Flow Problem in Power System Planning Studies. *Energy and Power Engineering*. 2015. Vol.7 No.10. P. 509-523. https://doi.org/10.4236/epe.2015.710048.
147. Montoya O. D., Molina-Cabrera A., Hernández J. C. A Comparative Study on Power Flow Methods Applied to AC Distribution Networks with Single-Phase Representation. *Electronics*. 2021. Vol. 10, Issue 21. https://doi.org/10.3390/electronics10212573.
148. Nair A. S., Abhyankar S., Peles S., Ranganathan P. Computational and numerical analysis of AC optimal power flow formulations on large-scale power grids. *Electric Power Systems Research*. 2022. Vol. 202. https://doi.org/10.1016/j.epsr.2021.107594.
149. Chatzivasileiadis S. Lecture Notes on Optimal Power Flow (OPF). *arXiv*. 2018. https://doi.org/10.48550/arXiv.1811.00943.
150. Baker K. Solutions of DC OPF are Never AC Feasible. *arXiv*. 2020. https://doi.org/10.48550/arXiv.1912.00319.
151. Nakiganda A. M., Dehghan S., Aristidou P. Comparison of AC Optimal Power Flow Methods in Low-Voltage Distribution Networks. *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. 2021. P. 1-5. https://doi.org/10.1109/ISGTEurope52324.2021.9639957.
152. Zhang H., Vittal V., Heydt G. T., Quintero J. A relaxed AC optimal power flow model based on a Taylor series. *2013 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*. 2013. P. 1-5. https://doi.org/10.1109/ISGT-Asia.2013.6698739.
153. Overbye T. J., Cheng X., Sun Y. A comparison of the AC and DC power flow models for LMP calculations. *37th Annual Hawaii International Conference on System Sciences*. 2004. https://doi.org/10.1109/HICSS.2004.1265164.
154. Yang C., Sun Y., Zou Y., Zheng F., Liu S., Zhao B., Wu M., Cui H. Optimal Power Flow in Distribution Network: A Review on Problem Formulation and Optimization Methods. *Energies*. 2023; Vol. 16, Issue 16. https://doi.org/10.3390/en16165974.

155. Guo Z., Sun K., Su X., Simunovic S. A review on simulation models of cascading failures in power systems. *iEnergy*. 2023. Vol. 2, Issue 4. P. 284-296. https://doi.org/10.23919/IEN.2023.0039.

156. Gjorgiev B., David A. E., Sansavini G. Cascade-risk-informed transmission expansion planning of AC electric power systems. *Electric Power Systems Research*. 2022. Vol. 204. https://doi.org/10.1016/j.epsr.2021.107685.

157. Mehlig B. Machine learning with neural networks. *arXiv*. 2021. https://doi.org/10.48550/arXiv.1901.05639.

158. Dubey S. R., Singh S. K., Chaudhuri B. B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2109.14545.

159. Elharrouss O., Mahmood Y., Bechqito Y., Serhani M. A., Badidi E., Riffi J., Tairi H. Task-based Loss Functions in Computer Vision: A Comprehensive Review. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2504.04242.

160. Waikhom L., Patgiri R. Graph Neural Networks: Methods, Applications, and Opportunities. *arXiv*. 2021. https://doi.org/10.48550/arXiv.2108.10733.

161. Gilmer J., Schoenholz S. S., Riley P. F., Vinyals O., Dahl G. E. Neural Message Passing for Quantum Chemistry. *arXiv*. 2017. https://doi.org/10.48550/arXiv.1704.01212.

162. Yang Z., Cohen W. W., Salakhutdinov R. Revisiting Semi-Supervised Learning with Graph Embeddings. *arXiv*. 2016. https://doi.org/10.48550/arXiv.1603.08861.

163. Ju W., Yi S., Wang Y., Xiao Z., Mao Z., Li H., Gu Y., Qin Y., Yin N., Wang S., Liu X., Yu P. S., Zhang M. A Survey of Graph Neural Networks in Real world: Imbalance, Noise, Privacy and OOD Challenges. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2403.04468.

164. Wu Z., Pan S., Chen F., Long G., Zhang C., Yu P. S. A Comprehensive Survey on Graph Neural Networks. *arXiv*. 2019. https://doi.org/10.48550/arXiv.1901.00596.

165. Kipf T. N., Welling M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv*. 2017. https://doi.org/10.48550/arXiv.1609.02907.

166. Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. Graph Attention Networks. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1710.10903.
167. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention Is All You Need. *arXiv*. 2023. https://doi.org/10.48550/arXiv.1706.03762.
168. Graph Attention Networks: Self-Attention for GNNs. URL: https://mlabonne.github.io/blog/posts/2022-03-09-Graph_Attention_Network.html (дата звернення: 11.12.2025).
169. Rusch T. K., Bronstein M. M., Mishra S. A Survey on Oversmoothing in Graph Neural Networks. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2303.10993.
170. Hamilton W. L., Ying R., Leskovec J. Inductive Representation Learning on Large Graphs. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1706.02216.
171. Yan S., Xiong Y., Lin D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1801.07455.
172. Yu B., Yin H., Zhu Z. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1709.04875.
173. Zhu J., Song Y., Zhao L., Li H. A3T-GCN: Attention Temporal Graph Convolutional Network for Traffic Forecasting. *arXiv*. 2020. https://doi.org/10.48550/arXiv.2006.11583.
174. Hu L., Liu S., Feng W. Spatial Temporal Graph Attention Network for Skeleton-Based Action Recognition. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2208.08599.
175. Al Sahili Z., Awad M. Spatio-Temporal Graph Neural Networks: A Survey. *arXiv*. 2023. https://doi.org/10.48550/arXiv.2301.10569.
176. Corradini F., Gerosa F., Gori M., Lucheroni C., Piangerelli M., Zannotti M. A Systematic Literature Review of Spatio-Temporal Graph Neural Network Models for Time Series Forecasting and Classification. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2410.22377.

177. Staudemeyer R. C., Morris E. R. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. *arXiv*. 2019. https://doi.org/10.48550/arXiv.1909.09586.

178. Understanding LSTM Networks. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 11.12.2025).

179. Хоменко О., Коваль О. Аналіз та порівняння сценаріїв каскадних ефектів в критичній інфраструктурі. *Інформаційні технології та суспільство*. 2025. № 3 (18). С. 176-185. https://doi.org/10.32689/maup.it.2025.3.24.

180. Хоменко О., Коваль О. Аналіз сценаріїв каскадних ефектів в критичній інфраструктурі на основі графових нейронних мереж. *Інформаційні технології та суспільство.* 2025. №4 (19). С. 182–190. https://doi.org/10.32689/maup.it.2025.4.28.

181. pool.global_mean_pool. URL: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.pool.global_mean_pool.html#torch_geometric.nn.pool.global_mean_pool (дата звернення: 11.12.2025).

182. Gavrilenko O., Khomenko O., Zhurakovska O., Kohan A., Piskun A., Khalus O. Application of association rules for formation of public (administrative) services portfolio. *Advanced Information Systems*. 2022. Vol. 6, No. 4. P. 63–68. https://doi.org/10.20998/2522-9052.2022.4.09.

183. Гавриленко О., Жураковська О., Коган А., Богданова Н., Хоменко О. Аналіз впливу коефіцієнтів подібності на склад портфелів публічних (адміністративних) послуг. Адаптивні системи автоматичного управління. 2023. Том 1, № 42. С. 49-58. https://doi.org/10.20535/1560-8956.42.2023.279089.

184. Gavrilenko O., Khomenko O., Zhurakovska O., Kogan A., Matviichuk R., Piskun A., Khavikova Y. Establishing the grouping principle of public services based on the analysis of similarity coefficients. *Eastern-European Journal of Enterprise Technologies*. 2023. Vol. 3, No. 3 (123). P. 22–29. https://doi.org/10.15587/1729-4061.2023.280218.

185. Гавриленко О. В., Хоменко О. М., Аналіз роботи алгоритмів для формування портфелів публічних (адміністративних) послуг на основі асоціативних правил.

Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», Київ, 25 листопада 2022. С. 89-90.

186. Хоменко О. М., Гавриленко О. В. Концепція інформаційної системи для формування портфелів публічних (адміністративних) послуг. *Матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)»*, Київ, 9-11 травня 2023. С. 34-37.

187. Gavrilenko O., Zhurakovska O., Khomenko O. Analysis of the feasibility of implementing service portfolios. *Abstracts of XXVIII International Scientific and Practical Conference «Unusual methods of development of science and thoughts»*, Madrid, 17 – 19 July 2023. С. 151-154.

188. Kosub S. A note on the triangle inequality for the Jaccard distance. *arXiv*. 2016. https://doi.org/10.48550/arXiv.1612.02696.

189. You K. Semantics at an Angle: When Cosine Similarity Works Until It Doesn't. *arXiv*. 2025. https://doi.org/10.48550/arXiv.2504.16318.

190. Коваль О. В., Хоменко О. М. Вплив каскадних ефектів на роботу критичної інфраструктури. *Матеріали I Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 14 грудня 2023. С. 36-38.

191. Хоменко О. М., Сенченко В. Р., Коваль О. В. Програмні засоби аналізу каскадних ефектів в критичній інфраструктурі. *Матеріали XXI Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»*, Київ, 23–26 квітня 2024. С. 190-192.

192. Хоменко О. М., Коваль О. В. Підходи до аналізу каскадних ефектів в роботі електромереж. *Матеріали II Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 13 листопада 2024. С. 71-73.

193. Хоменко О. М., Коваль О. В. Створення онтологічної моделі для опису моделювання електромережі. *Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»*, Київ, 24 квітня 2025. С. 469-472.
194. Хоменко О. М., Коваль О. В. Генерація даних для аналізу роботи пов'язаних критичних інфраструктур. Abstracts of XXIV International Scientific and Practical Conference «Latest theories and technologies for the development of scientific research», Zaragoza, 16-18 June 2025. С. 256-259.
195. Хоменко О. М., Коваль О. В. Використання онтологічної моделі та машинного навчання для аналізу каскадних ефектів в критичній інфраструктурі. *Матеріали III Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 13 листопада 2025. С. 61-63.
196. Wang D., Lu X., Rinaldo A. DBSCAN: Optimal Rates For Density-Based Cluster Estimation. *arXiv*. 2019. https://doi.org/10.48550/arXiv.1706.03113.
197. Gewers F. L., Ferreira G. R., De Arruda H. F., Silva F. N., Comin C. H., Amancio D. R., Da F. Costa L. Principal Component Analysis: A Natural Approach to Data Exploration. *arXiv*. 2018. https://doi.org/10.48550/arXiv.1804.02502.
198. Sharma A., Kumar M., Agarwal S. A Complete Survey on Software Architectural Styles and Patterns. *Procedia Computer Science*. 2015. Vol. 70. P. 16–28. https://doi.org/10.1016/j.procs.2015.10.019.
199. Software Architecture Guide. URL: https://martinfowler.com/architecture/ (дата звернення: 11.12.2025).
200. API Gateway vs. Load Balancer vs. Reverse proxy. URL: https://www.geeksforgeeks.org/system-design/api-gateway-vs-load-balancer-vs-reverse-proxy/ (дата звернення: 11.12.2025).
201. Nginx. URL: https://nginx.org/en/ (дата звернення: 11.12.2025).
202. Authentication and authorization in a microservice architecture: Part 1 – Introduction. URL: https://microservices.io/post/architecture/2025/04/25/microservices-authn-authz-part-1-introduction.html (дата звернення: 11.12.2025).

203. What is JSON Web Token? URL: https://www.jwt.io/introduction#what-is-json-web-token (дата звернення: 11.12.2025).
204. What is the JSON Web Token structure? URL: https://www.jwt.io/introduction#what-is-json-web-token-structure (дата звернення: 11.12.2025).
205. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 11.12.2025).
206. ACID. URL: https://www.postgresql.org/docs/current/glossary.html (дата звернення: 11.12.2025).
207. SQLAlchemy. URL: https://www.sqlalchemy.org/ (дата звернення: 11.12.2025).
208. FastAPI. URL: https://fastapi.tiangolo.com/ (дата звернення: 11.12.2025).
209. Change Data Capture (CDC). URL: https://www.geeksforgeeks.org/system-design/change-data-capture-cdc/ (дата звернення: 11.12.2025).
210. Debezium. URL: https://debezium.io/ (дата звернення: 11.12.2025).
211. Apache Kafka. URL: https://kafka.apache.org/ (дата звернення: 11.12.2025).
212. Apache Flink. URL: https://flink.apache.org/ (дата звернення: 11.12.2025).
213. Streaming Analytics. URL: https://nightlies.apache.org/flink/flink-docs-release-1.19/docs/learn-flink/streaming_analytics/ (дата звернення: 11.12.2025).
214. Apache Flink. Windows. URL: https://nightlies.apache.org/flink/flink-docs-release-2.1/docs/dev/datastream/operators/windows/ (дата звернення: 11.12.2025).
215. Apache Superset. URL: https://superset.apache.org/ (дата звернення: 11.12.2025).
216. PyTorch. URL: https://pytorch.org/ (дата звернення: 11.12.2025).
217. PyG (PyTorch Geometric). URL: https://pytorch-geometric.readthedocs.io/en/latest/ (дата звернення: 11.12.2025).

ДОДАТОК А

ЗАТВЕРДЖУЮ
проректор з навчальної роботи
КПІ ім. Ігоря Сікорського,
кандидат технічних наук, доцент
Тетяна ЖЕЛЯСКОВА
«20» 03 2026 р.

АКТ ВПРОВАДЖЕННЯ
результатів дисертаційного дослідження аспіранта кафедри інженерії програмного забезпечення в енергетиці, навчально-наукового інституту атомної та теплової енергетики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» Хоменка Олександра Миколайовича за темою «Методи та програмне забезпечення семантичного моделювання каскадних ефектів у критичних інфраструктурах із застосуванням графових нейронних мереж» на здобуття наукового ступеня доктора філософії за спеціальністю 121 «Інженерія програмного забезпечення» в освітній процес Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»

Комісія у складі:
- голови комісії, в.о. завідувача кафедри інженерії програмного забезпечення в енергетиці, професора, д. т. н. Коваля О.В.;
- професора, д. т. н. Федорова Н. В
- професора, д. т. н. Гаврилка Є. В.

засвідчує, що результати дисертаційного дослідження Хоменка Олександра Миколайовича на тему «Методи та програмне забезпечення семантичного моделювання каскадних ефектів у критичних інфраструктурах із застосуванням графових нейронних мереж», а саме:

1. Метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі онтологічної моделі та множини правил; онтологічна модель, що описує структуру електромережі, сценарії роботи електромережі при виникненні збоїв (каскадні ефекти); теоретичний підхід використання семантичної моделі з наборами даних для виявлення протирічь та доповнення метаданими, впроваджені у лекційні та лабораторні заняття з навчальної дисципліни «Інженерія даних та знань», що входить до нормативної підготовки здобувачів ступеня магістра за ОПП «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» за спеціальністю 121 «Інженерія програмного забезпечення»:

Лекція 4. Онтологічна інженерія.
Лекція 5. Онтологічна інженерія (продовження).
Лекція 6. Формування таксономії предметної області та атрибутів класів онтології, відношень між концептами.
Лекція 7. Розширений пошук інформації та знань на основі семантичної моделі.
Лабораторна робота 3.
Посилання на силабус: https://ipze.kpi.ua/

2. Метод визначення подібності сценаріїв на основі графової нейронної мережі; модель автоенкодера для формування представлень про стан електромереж в сценаріях; архітектуру програмного забезпечення обробки динамічних даних, формування представлень про сценарії для аналізу впроваджені в навчальний процес кафедри «Інженерія програмного забезпечення в енергетиці» у лекційні та лабораторні заняття навчальної дисципліни «Інтелектуальний аналіз даних для задач енергетики», що входить до нормативної підготовки здобувачів ступеня магістра за ОПП «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці» за спеціальністю 121 «Інженерія програмного забезпечення»:

Лекція 7. Опис, елементи і архітектура, процес навчання і явище перенавчання нейронної мережі. Моделі нейронних мереж, метод зворотного розповсюдження помилки.

Лабораторний практикум 4: Робота з нейронними мережами та іншими методами аналітичної обробки даних.

Посилання на силабус: https://ipze.kpi.ua/

Зазначене актуалізує зміст навчальних дисциплін до потреб практики.

В. о. завідувача кафедри
інженерії програмного забезпечення в енергетиці,
доктор технічних наук, професор _____ Олександр КОВАЛЬ

Професор, д.т.н. _____ Наталія ФЕДОРОВА

Професор, д.т.н. _____ Євген ГАВРИЛКО

ЗАТВЕРДЖУЮ
Заступник директора
ІПРІ НАН України,
доктор технічних наук, професор

_____ Олександр ДОДОНОВ
«24» 12 2025р

АКТ

Про впровадження результатів дисертаційної роботи Хоменко О.М. «Методи та програмне забезпечення семантичного моделювання каскадних ефектів у критичних інфраструктурах із застосуванням графових нейронних мереж»

Цей акт засвідчує те, що наукові та практичні результати дисертаційної роботи Хоменка Олександра Миколайовича за темою «Методи та програмне забезпечення семантичного моделювання каскадних ефектів у критичних інфраструктурах із застосуванням графових нейронних мереж» були впроваджені в Інституті проблем реєстрації інформації Національної академії наук України при здійснені відомчої науково-технічної роботи «Розроблення та дослідження методів і комп'ютерних технологій сценарного аналізу каскадних ефектів пов'язаних критичних інфраструктур (шифр: «SACE-24»), а саме:

1) Семантична модель, що описує об'єкти та зв'язки в сценаріях каскадних збоїв в електромережах, правила для перевірки коректності даних.
2) Метод семантичного моделювання сценаріїв каскадних збоїв в електромережах на основі онтологічної моделі та множини правил, що надає можливість виявляти протиріччя в даних на ранньому етапі проєктування.
3) Теоретичний підхід використання семантичної моделі з наборами даних, що надає можливість доповнити дані метаданими, які можуть бути використанні в подальшому аналізі.

Результати роботи рекомендовані до подальшого використання при розробленні програмних систем аналізу подій каскадних ефектів пов'язаних критичних інфраструктур та підготовки рішень щодо запобіганню наслідків цих подій.

Відповідальний виконавець
науково-технічної роботи «SACE-24»,
старший науковий співробітник
ІПРІ НАН України, к.т.н. В'ячеслав СЕНЧЕНКО

ДОДАТОК Б

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковані основні наукові результати дисертації:

1. Сенченко В. Р., Бойченко А. В., Коваль О. В., Бисько Р. М., Хоменко О. М. Огляд методів і технологій сценарного аналізу каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2024. Т. 26, № 1. С. 24–54. https://doi.org/10.35681/1560-9189.2024.26.1.308506.

Особистий внесок кожного автора:

– Сенченко В. Р.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Бойченко А. В.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Коваль О. В.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Бисько Р. М.: систематизація наявних досліджень аналізу каскадних ефектів, редагування тексту статті;

– Хоменко О. М.: дослідження розвитку каскадних ефектів у критичній інфраструктурі, застосування методів машинного навчання для аналізу каскадних відмов, редагування тексту статті.

2. Хоменко О. М., Сенченко В. Р., Коваль О. В. Мережевий підхід при дослідженні каскадних ефектів критичних інфраструктур. *Реєстрація, зберігання і обробка даних.* 2024. Т. 26. № 2. С. 44–72. https://doi.org/10.35681/1560-9189.2024.26.2.316908.

Особистий внесок кожного автора:

– Хоменко О. М.: пошук та відбір релевантних публікацій, аналіз методів моделювання каскадних збоїв у роботі критичної інфраструктури, систематизація інформації, підготовка тексту статті;

– Сенченко В. Р.: перевірка та доопрацювання тексту статті;

– Коваль О. В.: перевірка тексту статті.

3. Сенченко В. Р., Бойченко А. В., Коваль О. В., Хоменко О. М. Методологія та архітектура платформи моделювання взаємозалежностей у критичних інфраструктурах при виникненні каскадних ефектів. *Реєстрація, зберігання і обробка даних*. 2025. Т. 27. № 1. С. 28–41. https://doi.org/10.35681/1560-9189.2025.27.1.335624.

Особистий внесок кожного автора:

– Сенченко В. Р.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Бойченко А. В.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Коваль О. В.: розробка концепції моделювання взаємозалежностей у критичних інфраструктурах, редагування тексту статті;

– Хоменко О. М.: дослідження переваг та обмежень використання різних видів архітектури програмного забезпечення, редагування тексту статті.

4. Хоменко О. М., Коваль О. В. Побудова сценаріїв розвитку каскадних збоїв в інфраструктурі електромереж. *Зв'язок.* 2025. №5. С. 3–12. https://doi.org/10.31673/2412-9070.2025.050938.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, формування набору даних, проведення експериментів, формування результатів, підготовка та редагування статті;

– Коваль О. В.: перевірка та редагування тексту статті.

5. Хоменко О., Коваль О. Аналіз та порівняння сценаріїв каскадних ефектів в критичній інфраструктурі. *Інформаційні технології та суспільство*. 2025. № 3 (18). С. 176–185. https://doi.org/10.32689/maup.it.2025.3.24.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, розробка моделі, формування набору даних, проведення експериментів, формування результатів, підготовка статті;

– Коваль О. В.: перевірка та редагування тексту статті.

6. Хоменко О., Коваль О. Аналіз сценаріїв каскадних ефектів в критичній інфраструктурі на основі графових нейронних мереж. *Інформаційні технології та суспільство.* 2025. №4 (19). С. 182–190. https://doi.org/10.32689/maup.it.2025.4.28.

Особистий внесок кожного автора:

– Хоменко О. М.: розробка методу, розробка моделі, формування набору даних, проведення експериментів, формування результатів, підготовка статті;

– Коваль О. В.: перевірка та редагування тексту статті.

7. Gavrilenko O., Khomenko O., Zhurakovska O., Kohan A., Piskun A., Khalus O. Application of association rules for formation of public (administrative) services portfolio. *Advanced Information Systems*. 2022. Vol. 6, No. 4. P. 63–68. https://doi.org/10.20998/2522-9052.2022.4.09.

Особистий внесок кожного автора:

– Gavrilenko O.: розробка концепції, визначення завдань в статті, редагування статті;

– Khomenko O.: розробка концепції, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті;

– Zhurakovska O.: розробка концепції, формалізація проблематики, редагування статті;

– Kohan A.: перевірка, виправлення та доопрацювання тексту статті;

– Piskun A.: перевірка, виправлення та доопрацювання тексту статті;

– Khalus O.: перевірка, виправлення та доопрацювання тексту статті.

8. Гавриленко О., Жураковська О., Коган А., Богданова Н., Хоменко О. Аналіз впливу коефіцієнтів подібності на склад портфелів публічних (адміністративних) послуг. Адаптивні системи автоматичного управління. 2023. Том 1, № 42. С. 49–58. https://doi.org/10.20535/1560-8956.42.2023.279089.

Особистий внесок кожного автора:

– Гавриленко О.: розробка концепції, алгоритму, редагування статті;

– Жураковська О.: розробка концепції, алгоритму, редагування статті;

– Коган А.: перевірка, виправлення та доопрацювання тексту статті;

– Богданова Н.: перевірка, виправлення та доопрацювання тексту статті;

– Хоменко О.: розробка концепції, алгоритму, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті.

9. Gavrilenko O., Khomenko O., Zhurakovska O., Kogan A., Matviichuk R., Piskun A., Khavikova Y. Establishing the grouping principle of public services based on the analysis of similarity coefficients. *Eastern-European Journal of Enterprise Technologies*. 2023 Vol. 3, No. 3 (123). P. 22–29. https://doi.org/10.15587/1729-4061.2023.280218.

Особистий внесок кожного автора:

– Gavrilenko O.: розробка алгоритму, визначення завдань в статті, редагування статті;

– Khomenko O.: розробка алгоритму, написання комп'ютерного коду алгоритму, формування набору даних, проведення експериментів, формування результатів, редагування статті;

– Zhurakovska O.: розробка алгоритму, формалізація проблематики, редагування статті;

– Kogan A.: перевірка, виправлення та доопрацювання тексту статті;

– Matviichuk R.: перевірка, виправлення та доопрацювання тексту статті;

– Piskun A.: перевірка, виправлення та доопрацювання тексту статті;

– Khavikova Y.: перевірка, виправлення та доопрацювання тексту статті.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

10. Коваль О. В., Хоменко О. М. Вплив каскадних ефектів на роботу критичної інфраструктури. *Матеріали І Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 14 грудня 2023. С. 36–38.

Особистий внесок кожного автора: Коваль О. В. – перевірка та редагування тексту; Хоменко О. М. – розробка концепції публікації, редагування тексту.

11. Хоменко О. М., Сенченко В. Р., Коваль О. В. Програмні засоби аналізу каскадних ефектів в критичній інфраструктурі. *Матеріали XXI Міжнародної науково-практичної конференції молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики»*, Київ, 23–26 квітня 2024. С. 190–192.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення аналізу роботи критичної інфраструктури, редагування тексту; Сенченко В. Р. – перевірка тексту; Коваль О. В. – перевірка тексту.

12. Хоменко О. М., Коваль О. В. Підходи до аналізу каскадних ефектів в роботі електромереж. *Матеріали II Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 13 листопада 2024. С. 71–73.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення аналізу, систематизація інформації, редагування тексту; Коваль О. В. – перевірка тексту.

13. Хоменко О. М., Коваль О. В. Створення онтологічної моделі для опису моделювання електромережі. *Матеріали VI Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»*, Київ, 24 квітня 2025. С. 469–472.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, розробка онтологічної моделі для опису моделювання електромережі, оформлення результатів; Коваль О. В. – перевірка тексту.

14. Хоменко О. М., Коваль О. В. Генерація даних для аналізу роботи пов'язаних критичних інфраструктур. Abstracts of XXIV International Scientific and Practical Conference «Latest theories and technologies for the development of scientific research», Zaragoza, 16–18 June 2025. С. 256–259.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, опис підходів для генерації даних про роботу критичної інфраструктури, оформлення результатів; Коваль О. В. – перевірка тексту.

15. Хоменко О. М., Коваль О. В. Використання онтологічної моделі та машинного навчання для аналізу каскадних ефектів в критичній інфраструктурі. *Матеріали III Міжнародної науково-практичної конференції «Сучасні аспекти інженерії програмного забезпечення»*, Київ, 13 листопада 2025. С. 61–63.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, проведення експериментів, оформлення результатів; Коваль О. В. – перевірка тексту.

16. Гавриленко О. В., Хоменко О. М., Аналіз роботи алгоритмів для формування портфелів публічних (адміністративних) послуг на основі асоціативних правил. *Матеріали IX Міжнародної науково-технічної Internet-конференції «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами»*, Київ, 25 листопада 2022. С. 89–90.

Особистий внесок кожного автора: Гавриленко О. В. – розробка концепції публікації, проведення експериментів, редагування тексту; Хоменко О. М. – розробка концепції публікації, проведення експериментів, редагування тексту.

17. Хоменко О. М., Гавриленко О. В. Концепція інформаційної системи для формування портфелів публічних (адміністративних) послуг. *Матеріали IV Міжнародної науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)»*, Київ, 9–11 травня 2023. С. 34–37.

Особистий внесок кожного автора: Хоменко О. М. – розробка концепції публікації, редагування тексту; Гавриленко О. В. – розробка концепції публікації, редагування тексту.

18. Gavrilenko O., Zhurakovska O., Khomenko O. Analysis of the feasibility of implementing service portfolios. *Abstracts of XXVIII International Scientific and Practical Conference «Unusual methods of development of science and thoughts»*, Madrid, 17–19 July 2023. С. 151–154.

Особистий внесок кожного автора: Gavrilenko O. – розробка концепції публікації, перевірка тексту; Zhurakovska O. – розробка концепції публікації,

перевірка тексту; Khomenko O. – розробка концепції публікації, проведення експериментів, формування результатів, редагування тексту.

Відомості про апробацію результатів дисертації

1. IX Міжнародна науково-технічна Internet-конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами», 25 листопада 2022 р., Київ, Україна.

2. IV Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2023)», 9–11 травня 2023 р., Київ, Україна.

3. XXVIII Міжнародна науково-практична конференція «Unusual methods of development of science and thoughts», 17–19 липня 2023 р., Мадрид, Іспанія.

4. I-а Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення», 14 грудня 2023 р., Київ, Україна.

5. XXI Міжнародна науково-практична конференція молодих вчених і студентів «Сучасні проблеми наукового забезпечення енергетики», 23–26 квітня 2024 р., Київ, Україна.

6. 2 Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення», 13 листопада 2024 р., Київ, Україна.

7. VI Всеукраїнська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2025 р., Київ, Україна.

8. XXIV Міжнародна науково-практична конференція «Latest theories and technologies for the development of scientific research», 16–18 червень 2025 р., Сарагоса, Іспанія.

9. III Міжнародна науково-практична конференція «Сучасні аспекти інженерії програмного забезпечення» MoSE-2025, Київ, Україна.

ДОДАТОК В

ЛІСТИНГ КОДУ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

```
class GraphSequenceAutoencoder(nn.Module):

    def __init__(self, num_nodes, num_edges, in_node_features, in_edge_features, hidden_dim, latent_dim, num_heads):
        super().__init__()

        self.num_nodes = num_nodes # Number of nodes
        self.num_edges = num_edges # Number of edges

        # Embedding size
        self.latent_dim = latent_dim

        self.lstm_units = 128

        # Encoder
         self.conv1 = CustomNodeMessagePassing(in_node_features, in_edge_features, hidden_dim)
        self.conv2 = CustomNodeMessagePassing(hidden_dim, hidden_dim, latent_dim)

                self.conv1_edge = CustomEdgeMessagePassing(in_node_features, in_edge_features, hidden_dim)
            self.conv2_edge = CustomEdgeMessagePassing(hidden_dim, hidden_dim, latent_dim)

        self.lstm_layer = nn.LSTM(latent_dim, self.lstm_units, 1, batch_first=True)
```

```
        self.multihead_attn = nn.MultiheadAttention(embed_dim=self.lstm_units, num_heads=num_heads, batch_first=True)

        self.encoder_to_latent = nn.Linear(self.lstm_units, latent_dim)

        #Decoder
        self.latent_to_hidden = nn.Linear(latent_dim, self.lstm_units)

        self.decoder_rnn = nn.LSTM(self.lstm_units, latent_dim, 1, batch_first=True)

        self.unpooling_nodes = nn.Sequential(
            nn.Linear(latent_dim, latent_dim*num_nodes),
            nn.ReLU(),
            nn.Linear(latent_dim*num_nodes, in_node_features*num_nodes),
            nn.ReLU(),
            nn.Linear(in_node_features*num_nodes, latent_dim*num_nodes),
        )

        self.unpooling_edges = nn.Sequential(
            nn.Linear(latent_dim, latent_dim*num_edges),
            nn.ReLU(),
            nn.Linear(latent_dim*num_edges, in_edge_features*num_edges),
            nn.ReLU(),
            nn.Linear(in_edge_features*num_edges, latent_dim*num_edges),
        )

        self.node_decoder_conv1 = CustomNodeMessagePassing(latent_dim, latent_dim, hidden_dim)
        self.node_decoder_conv2 = CustomNodeMessagePassing(hidden_dim, hidden_dim, in_node_features)
```

```
        self.edge_decoder_conv1 = CustomEdgeMessagePassing(latent_dim, latent_dim,
hidden_dim)
                self.edge_decoder_conv2 = CustomEdgeMessagePassing(hidden_dim,
hidden_dim, in_edge_features)

        self.apply(self._init_weights)

    def _init_weights(self, module):
        if isinstance(module, nn.LSTM):
            # Orthogonal initialization for recurrent weights
            for name, param in module.named_parameters():
                if 'weight_hh' in name:
                    nn.init.orthogonal_(param)
                elif 'weight_ih' in name:
                    nn.init.xavier_uniform_(param) # Xavier for input weights
                elif 'bias' in name:
                    # Initialize forget gate bias to 1 to prevent early vanishing
                    param.data.fill_(0)
                    n = param.size(0)
                    param.data[n//4:n//2].fill_(1.0)
        elif isinstance(module, nn.Linear):
            nn.init.kaiming_uniform_(module.weight, nonlinearity='relu')
            if module.bias is not None:
                module.bias.data.fill_(0.01) # Initialize biases to a small constant

    # Encode time series graphs in sequences
    def graph_encode(self, graph_sequences, masks):

        outputs = []
```

```
    for i in range(len(graph_sequences)): # Iterate through time steps (padded length)
        batch_graphs_at_t = graph_sequences[i]
        batch_masks_at_t = masks[:, i]

        # Process valid graphs only (using mask)
        valid_graphs = batch_graphs_at_t[batch_masks_at_t]
        if not valid_graphs:
                                    outputs.append(torch.zeros(len(graph_sequences[0]),
self.conv2.out_channels)) # Placeholder for empty step
            continue

                    # Batch valid graphs for GNN processing (e.g., using
torch_geometric.data.Batch)
        valid_graphs = Batch.from_data_list(valid_graphs)

                    new_x = self.conv1(valid_graphs.x, valid_graphs.edge_index,
valid_graphs.edge_attr, valid_graphs.node_statuses, valid_graphs.edge_statuses)
            new_edge_attr = self.conv1_edge(valid_graphs.x, valid_graphs.edge_index,
valid_graphs.edge_attr, valid_graphs.node_statuses, valid_graphs.edge_statuses)

        new_x = F.relu(new_x)
        new_edge_attr = F.relu(new_edge_attr)

        new_x = F.dropout(new_x, p=0.2, training=self.training)
        new_edge_attr = F.dropout(new_edge_attr, p=0.2, training=self.training)

                res_x = self.conv2(new_x, valid_graphs.edge_index, new_edge_attr,
valid_graphs.node_statuses, valid_graphs.edge_statuses)
```

```
                res_edge_attr = self.conv2_edge(new_x, valid_graphs.edge_index, new_edge_attr, valid_graphs.node_statuses, valid_graphs.edge_statuses)

        res_x = F.relu(res_x)
        res_edge_attr = F.relu(res_edge_attr)

        res_x = F.dropout(res_x, p=0.2, training=self.training)
        res_edge_attr = F.dropout(res_edge_attr, p=0.2, training=self.training)

        # Graph embedding
         pooling_graph_embedding = global_mean_pool(res_x, valid_graphs.batch) * (global_mean_pool(res_edge_attr,valid_graphs.batch[valid_graphs.edge_index[0]]) + global_mean_pool(res_edge_attr,valid_graphs.batch[valid_graphs.edge_index[1]]))

        # Reconstruct output for the full batch, considering masked elements
                        full_step_output = torch.zeros(len(graph_sequences[0]), self.conv2.out_channels)
        full_step_output[batch_masks_at_t] = pooling_graph_embedding

        #Add timestep data
        # (batch_size, self.conv2.out_channels)
        outputs.append(full_step_output)

       graph_embeddings = torch.stack(outputs, dim=1) # Stack outputs along the sequence dimension

    # (batch_size, seq_length, self.conv2.out_channels)
    return graph_embeddings

  # Encode time series sequences
```

```
def lstm_encode(self, graph_sequences, masks, seq_len):

    # Real length of scenarios in each sequence
    sequence_lengths = masks.sum(dim=1)

    # Pad sequences
    padded_sequences = pad_sequence(graph_sequences, batch_first=True) # Output shape: (batch_size, max_seq_len, features)

    # Pack padded sequences
      packed_input = pack_padded_sequence(padded_sequences, sequence_lengths, batch_first=True, enforce_sorted=False)

    # Forward pass through LSTM
    packed_output, (hidden, cell) = self.lstm_layer(packed_input)

    # Unpack output (optional)
    #Padded to seq_len
        output_padded, output_lengths = pad_packed_sequence(packed_output, batch_first=True, total_length=seq_len)

    # Create key_padding_mask for MultiheadAttention
    # True values indicate positions to be ignored (padded elements)
    max_seq_len = output_padded.size(1)
            key_padding_mask = torch.arange(max_seq_len).unsqueeze(0) >= sequence_lengths.unsqueeze(1)
    # key_padding_mask: (batch_size, seq_len)
       # MultiheadAttention expects query, key, value in (batch_size, seq_len, embed_dim)
    # In self-attention, query, key, and value are all the same (lstm_output)
```

```
        attn_output, _ = self.multihead_attn(query=output_padded, key=output_padded,
value=output_padded,
                                             key_padding_mask=key_padding_mask)
        # attn_output: (batch_size, seq_len, embed_dim)

        last_actual = attn_output[torch.arange(attn_output.size(0)), output_lengths - 1]
        last_actual_padded = torch.zeros(attn_output.size(0),attn_output.size(2))
        last_actual_padded[:] = last_actual

        return last_actual_padded

    # Encode data
    def encode(self, graph_sequences, masks, seq_len):

        # Store outputs for skip connections
        skip_connection_features = []

        graph_encoded_sequence = self.graph_encode(graph_sequences, masks)
          skip_connection_features.append(graph_encoded_sequence) # Save output of
graph conv

        graph_embedding = self.lstm_encode(graph_encoded_sequence, masks, seq_len)
        skip_connection_features.append(graph_embedding) # Save output of lstm

        graph_embedding = self.encoder_to_latent(graph_embedding)

        return graph_embedding, skip_connection_features

    # Decode time series sequences
    def lstm_decode(self, graph_sequences, masks, seq_len, skip_connection_features):
```

```
 decoder_input = graph_sequences + skip_connection_features.pop() # Features from lstm output

        # (batch_size, lstm_hidden) -> (batch_size, seq_length, embedding_hidden)
        decoder_input = decoder_input.unsqueeze(1).repeat(1, seq_len, 1)

        #decoder_input = self.decoder_lstm_layer_norm_input(decoder_input)

        # Real length of scenarios in each sequence
        sequence_lengths = masks.sum(dim=1)

        # Pad sequences
         padded_sequences = pad_sequence(decoder_input, batch_first=True) # Output shape: (batch_size, max_seq_len, features)

        # Pack padded sequences
          packed_input = pack_padded_sequence(padded_sequences, sequence_lengths, batch_first=True, enforce_sorted=False)

        # Forward pass through LSTM
        packed_output, (hidden, cell) = self.decoder_rnn(packed_input)

        # Unpack output (optional)
        #Padded to seq_len
              output_padded, output_lengths = pad_packed_sequence(packed_output, batch_first=True, total_length=seq_len)

        return output_padded
```

```
    # Decode graphs in time series sequences
    def graph_decode(self, input_decoded_rnn_output, masks, edge_index, skip_connection_features, graph_sequences):

                    decoded_rnn_output = input_decoded_rnn_output + skip_connection_features.pop() # Features graph conv output

        #decoded_rnn_output: (batch_size, seq_length, features)
        #masks: (batch_size, seq_length)

        #Info about edges
        #row, node_from: (num_edges)
        #col, node_to: (num_edges)
        row, col = edge_index
        node_outputs = []
        edge_outputs = []
        adj_outputs = []

        for i in range(len(decoded_rnn_output[0])): # Iterate through time steps (padded length)

            batch_graphs_at_t = decoded_rnn_output[:,i,:]
            batch_masks_at_t = masks[:, i]

                batch_node_statuses_at_t = graph_sequences[i].node_statuses.reshape(-1,self.num_nodes, 1)
                batch_edge_statuses_at_t = graph_sequences[i].edge_statuses.reshape(-1,self.num_edges, 1)

            # Process valid graphs only
```

```
        valid_graphs = batch_graphs_at_t[batch_masks_at_t]
        if valid_graphs.numel() == 0:

          node_outputs.append(torch.zeros(len(decoded_rnn_output), self.num_nodes,
self.node_decoder_conv2.out_channels))
          edge_outputs.append(torch.zeros(len(decoded_rnn_output), self.num_edges,
self.edge_decoder_conv2.out_channels))
           adj_outputs.append(torch.zeros(len(decoded_rnn_output), self.num_edges))

           continue

        # Build graphs nodes and edges

        valid_graphs_nodes = self.unpooling_nodes(valid_graphs)
        valid_graphs_edges = self.unpooling_edges(valid_graphs)

              valid_graphs_nodes = valid_graphs_nodes.reshape(-1,self.num_nodes,
self.latent_dim)
               valid_graphs_edges = valid_graphs_edges.reshape(-1,self.num_edges,
self.latent_dim)

        # edges list
         adj_matrix = torch.sum(valid_graphs_nodes[:, row] * valid_graphs_nodes[:,
col] * valid_graphs_edges, dim=2)

        # Batch valid graphs for GNN processing
        #to do through gnn
        data_list = []

        for i in range(len(valid_graphs_nodes)):
```

```
            data_list.append(Data(x=valid_graphs_nodes[i], edge_index=edge_index,
edge_attr=valid_graphs_edges[i],
                                  node_statuses = batch_node_statuses_at_t[i], edge_statuses =
batch_edge_statuses_at_t[i]))

        # Create the DataBatch from the list of Data objects
        databatch = Batch.from_data_list(data_list)

            x_rec = self.node_decoder_conv1(databatch.x, databatch.edge_index,
databatch.edge_attr, databatch.node_statuses, databatch.edge_statuses)
            edge_rec = self.edge_decoder_conv1(databatch.x, databatch.edge_index,
databatch.edge_attr, databatch.node_statuses, databatch.edge_statuses)

        x_rec = F.relu(x_rec)
        edge_rec = F.relu(edge_rec)
        x_rec = F.dropout(x_rec, p=0.2, training=self.training)
        edge_rec = F.dropout(edge_rec, p=0.2, training=self.training)

        node_attrs = self.node_decoder_conv2(x_rec, databatch.edge_index, edge_rec,
databatch.node_statuses, databatch.edge_statuses)
        edge_attrs = self.edge_decoder_conv2(x_rec, databatch.edge_index, edge_rec,
databatch.node_statuses, databatch.edge_statuses)

        # Reshape data
        node_attrs = node_attrs.reshape(-1, self.num_nodes,node_attrs.size(1))
        edge_attrs = edge_attrs.reshape(-1, self.num_edges,edge_attrs.size(1))

        # Reconstruct output for the full batch, considering masked elements
        node_full_step_output = torch.zeros(len(decoded_rnn_output), self.num_nodes,
self.node_decoder_conv2.out_channels)
```

```
        edge_full_step_output = torch.zeros(len(decoded_rnn_output), self.num_edges, self.edge_decoder_conv2.out_channels)
        adj_full_step_output = torch.zeros(len(decoded_rnn_output), self.num_edges)
        # Set graph data using mask
        node_full_step_output[batch_masks_at_t] = node_attrs
        edge_full_step_output[batch_masks_at_t] = edge_attrs
        adj_full_step_output[batch_masks_at_t] = adj_matrix

        #(batch_size, num_nodes, node_features)
        node_outputs.append(node_full_step_output)
        #(batch_size, num_edges, edge_features)
        edge_outputs.append(edge_full_step_output)
        #(batch_size, num_edges)
        adj_outputs.append(adj_full_step_output)
    #(batch_size, seq_length, num_nodes, node_features)
        node_result = torch.stack(node_outputs, dim=1) # Stack outputs along the sequence dimension
    #(batch_size, seq_length, num_edges, edge_features)
        edge_result = torch.stack(edge_outputs, dim=1) # Stack outputs along the sequence dimension
    #(batch_size, seq_length, num_edges)
     adj_result = torch.stack(adj_outputs, dim=1) # Stack outputs along the sequence dimension

    return node_result, edge_result, adj_result

  # Decode data
        def decode(self, graph_embedding, masks, edge_index, seq_len, skip_connection_features, graph_sequences):
```

```
        # Transform data
        graph_embedding = self.latent_to_hidden(graph_embedding)
        graph_embedding = F.relu(graph_embedding)

        # Reconstruct sequences
            decoded_rnn_output = self.lstm_decode(graph_embedding, masks, seq_len,
skip_connection_features)

        # Reconstruct graphs
                                    node_features, edge_features, adj_result =
self.graph_decode(decoded_rnn_output, masks, edge_index,
skip_connection_features, graph_sequences)

        return node_features, edge_features, adj_result

    def forward(self, graph_sequences, masks):

        # edge_index
        edge_index = graph_sequences[0][0].edge_index
        # seq_len
        seq_len = len(graph_sequences)

            graph_embedding, skip_connection_features = self.encode(graph_sequences,
masks, seq_len)
        node_features, edge_features, adj_result = self.decode(graph_embedding, masks,
edge_index, seq_len, skip_connection_features, graph_sequences)

        return node_features, edge_features, adj_result
```