Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Зменшення розмірності мережевих систем на базі агрегації структурної моделі

(тема)

Виконав:
студент II курсу, групи СПм-18-3
Пономаренко О.Є.
(прізвище, ініціали)

Спеціальність
123 – Комп’ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: проф. Горбачов В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Пономаренко Ользі Євгеніївні
(прізвище, ім’я, по батькові)

1. Тема роботи Зменшення розмірності мережевих систем на базі агрегації структурної моделі

затверджена наказом по університету від “ 30 ” березня 2020 р. № 478Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2020 р.

3. Вхідні дані до роботи 1) теорія графів; 2) опис агрегації структурної моделі; 3) мова програмування Java.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз літератури та постановка завдання
2) Моделі структур складних систем
3) Зменшення розмірності мережевих систем
4) Оцінка ефективності зменшення розмірності мережевих систем

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайди презентації

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	31.03.20-13.04.20	
2	Здійснення агрегації структурної моделі	14.04.20-21.04.20	
3	Реалізація програмної платформи	22.04.20-27.04.20	
4	Оцінка ефективності зменшення розмірності	28.04.20-05.05.20	
	мережевих систем		
5	Оформлення матеріалів атестаційної роботи	06.05.20-11.05.20	
6	Подання атестаційної роботи керівникові та її	12.05.20-13.05.20	
	попередній захист		
7	Подання атестаційної роботи на рецензування	14.05.20-15.05.20	

Дата видачі завдання 30 березня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Горбачов В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 107 с., 26 рис., 10 табл., 2 дод., 21 джерело.

СКЛАДНІ СИСТЕМИ, АГРЕГАЦІЯ СТРУКТУРНОЇ МОДЕЛІ, ПРОГРАМНА ПЛАТФОРМА, ГЕНЕРАТОР МЕРЕЖЕВИХ СТРУКТУР, ПОШУК МАКСИМАЛЬНОГО ПОТОКУ.

Метою атестаційної роботи є аналіз методів зменшення розмірності мережевих систем та застосування агрегації структурної моделі для зменшення розмірності мережевих систем.

У ході виконання атестаційної роботи був здійснений формальний опис агрегації структурної моделі, була розроблена програмна платформа, що здійснює зменшення розмірності мережевої системи, застосовуючи агрегацію структурної моделі, та була здійснена оцінка ефективності розглянутого методу.

У результаті роботи була розроблена програмна платформа, що генерує мережеві структури та здійснює агрегацію структурної моделі, для оцінки ефективності була вирішена задача пошуку максимального потоку. В результаті агрегації структурної моделі зменшується розмірність системи, обчислювальна складність та час вирішення проблеми.

ABSTRACT

Master’s thesis: 107 pages, 26 figures, 10 tables, 2 appendices, 21 sources.

COMPLEX SYSTEMS, STRUCTURE MODEL AGGREGATION, SOFTWARE PLATFORM, NETWORK STRUCTURES GENERATOR, MAXIMUM FLOW SEARCHING

The major goal of this thesis is an analysis of methods of dimension reduction of network systems and application of structure model aggregation for dimension reduction of network systems.

In order to fulfil the work, the following tasks were resolved: formal description of structure model aggregation, implementation of the software platform which accomplishes dimension reduction of network system using structure model aggregation and evaluation of the applied method effectiveness.

As a result, the software platform that generates network structures and performs structural model aggregation was implemented. Maximum flow searching problem was solved for effectiveness evaluation of applied method. System dimension was reduced and computational complexity and problem solving time decreased as a consequence of structural model aggregation.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HCS – ієрархічні складні системи (англ., Hierarchical Complex Systems)

HG – ієрархічний граф (англ., Hierarchical graph)

MAS – мультиагентні системи (англ., Multi-Agent Systems)

SFN – безмасштабні мережі (англ., Scale-free Networks)

SMA – агрегація структурної моделі (англ., Structure Model Aggregation)

ВСТУП

Складні системи зустрічаються в багатьох сферах, вони характеризуються великою розмірністю та обчислювальною складністю. Типовими прикладами є Інтернет, соціальні, економічні та біологічні мережі.

Виділяють такі аспекти складних систем, як характер елементів системи, характер зв’язків між елементами системи та схема зв’язків між елементами системи. Третій аспект відіграє велику роль у поведінці системи, тому що, якщо структура системи не відома, то неможливо повністю визначити, як вона працює. Схема зв’язків у системі може бути представлена у вигляді мережі, де елементи системи представлені вершинами мережі, а зв’язки представлені дугами.

Збільшення розмірів мереж призводить до розробки нових аналітичних підходів для їх представлення та оцінки ефективності. Вчені в різних сферах розробили математичні, обчислювальні та статистичні інструменти для аналізу та моделювання мереж. Існують інструменти, що представляють мережу як множину вершин і дуг, та завдяки результатам обчислення можна отримати інформацію про мережу. Також існують інструменти, що використовують мережеву модель та здійснюють математичні прогнози щодо процесів, які відбуваються в мережах.

Складна система містить велику кількість елементів та зв’язків між ними, та для вирішення ряду завдань може бути корисним об’єднати елементи у групи та отримати опис на більш високому рівні системи. Такий опис не повинен зберігати всю інформацію нижчого рівня, але функціонування на більш високому рівні дає такий саме ефект, що і спостереження агрегованих одиниць для нижчого рівня системи.

Існує багато сфер, де може бути застосована агрегація структурної моделі. Агрегація може застосовуватись для вирішення задачі пошуку найкоротших шляхів у графі. Це може бути застосовано в розробці

супутникової навігації, щоб мати можливість реагувати в режимі реального часу на оновлення трафіку. Ієрархічне представлення може полегшити роботу з такими оновленнями, тому що тільки ті частини, що містять змінені дороги, потребують оновлення. Також агрегація може бути застосована в обчисленні потоків дорожнього руху, коли потрібно дізнатися про можливі наслідки пропонованих змін у дорожній мережі.

Метою роботи є аналіз методів зменшення розмірності мережевих систем, формальний опис агрегації структурної моделі та розробка програмної платформи для оцінки ефективності агрегації структурної моделі. Програмна платформа складається з наступних модулів: модуль аналізу характеристик реальної системи, модуль генератора мережевих структур, модуль агрегації структурної моделі та модуль пошуку максимального потоку. Для оцінки запропонованих рішень науковці використовують згенеровані мережі, тому що створення реальної системи для експериментів майже неможливе через дуже великий розмір сучасних систем. Метою модуля генерації мережевих структур є створення великої кількості структур систем та застосування агрегації для згенерованих систем. Метою модулю агрегації структурної моделі є створення ієрархічної системи та зменшення розмірності системи зі збереженням коректності її параметрів. Модуль пошуку максимального потоку використовується для оцінки ефективності агрегації структурної моделі. Суть модулю полягає у знаходженні значення максимального потоку для усіх рівнів ієрархічної системи.

Актуальність роботи полягає у тому, що деякі проблеми можуть бути вирішені на більш високому рівні ієрархії системи. Для цього здійснюється агрегація структурної моделі, яка полягає у об’єднанні елементів системи у підсистеми, в результаті чого формується ієрархічна система. Перевагою цього підходу є зменшення кількості елементів системи та зв’язків між ними. Також в результаті агрегації структурної моделі зменшується розмірність системи, обчислювальна складність та час вирішення проблеми.

1 АНАЛІЗ ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Складні системи

Мережі використовуються як загальна модель різноманітних складних систем. В даний час широко використовується наступна класифікація складних мереж [1, 2]: технологічні, біологічні, екологічні та соціальні мережі. Складні системи характеризуються великою розмірністю та обчислювальною складністю [3]. За останній час кількість робіт, присвячених вивченню складних мереж або складних систем з мережевою топологією, зросла. Велика увага приділяється теорії мереж, мережевому моделюванню.

Відомо, що математична модель складної системи складається з опису її елементів та структури системи. Формальна техніка, що описує взаємозв’язки між елементами структури, добре вивчена і включає мережі Петрі, алгебру процесів та теорію множин. Ряд проблем системного аналізу потребує дослідження структурної моделі складної системи. При цьому деякі з них вирішуються лише шляхом перетворення існуючої структури у такий вид, який дозволяє досягти вирішення завдань, поставлених перед дослідженням.

Композиція складної мережі застосовується з метою зменшення її розмірності та зменшення часу моделювання. Основний зміст композиції полягає у поділі мережі на дві частини: основні вузли, що цікавлять, та вузли, що залишилися. Також демонструється, як метрики структурної моделі складної мережі можна використовувати для створення випадкових мереж [4]. Розроблені випадкові мережі оцінюють типи відмов мережі та пов’язані з ними наслідки.

Також агрегація може застосовуватись для вирішення задачі пошуку найкоротших шляхів у графі [5]. Це може бути застосовано в розробці

супутникової навігації, щоб мати можливість реагувати в режимі реального часу на оновлення трафіку

Контекстом, в якому можуть бути запропоновані схеми агрегації, також є марковські процеси [6]. Підхід полягає в тому, щоб об’єднати стани марковского процесу в групи та запропонувати марковский процес на множині груп, що має агреговану стаціонарну ймовірність. Потенційні переваги полягають у ефективному обчисленні, що включає перерахунок для врахування місцевих змін

Завдання топологічного аналізу систем являють собою комплекс складних задач, вирішення яких потребує великих обчислювальних ресурсів та розробки математичних методів. Їх можна об’єднати в наступні групи.

1. Розробка структурного опису складної системи. Завдання цієї групи складають топологічну структуру на основі оригінальної специфікації складної системи, тобто визначають елементи та підсистеми складної системи та зв’язки між ними.

2. Визначення характеристик складної системи із заданою топологічною структурою. Наприклад, визначення найкоротших шляхів та циклів. Крім того, модель структури системи використовується для аналізу показників якості структури. Наприклад, кількість посилань між підсистемами.

3. Проектування та еквівалентні перетворення топологічної структури складної системи.

Більшість завдань третьої групи – це завдання підвищеної складності. Проектування структури мережевих систем полягає у виборі оптимальної схеми з’єднання вузлів, вибору пропускної здатності каналів та оптимальної маршрутизації. Вибір топологічної структури здійснюється з використанням різних критеріїв і враховує обмеження затримки в часі та надійність передачі інформації.

Проблема еквівалентних перетворень належить до третьої групи завдань топологічного аналізу. Еквівалентні перетворення можуть бути

застосовані у наступних випадках:

- перерозподіл зв’язків та схем взаємодії в рамках початкової структури;
- агрегація та декомпозиція компонентів системи;
- зменшення розмірності системи, коли час оцінки метрик має велике значення у вирішенні завдань управління в реальному часі, а також коли час моделювання системи є критичним параметром.

Більшість проектів з розробки системи складається з аналізу архітектури системи [7]. Передбачалося, що існує правильний спосіб декомпозиції системи. Зазвичай процес розкладання визначається сприйняттям і знаннями дослідника та вмінням розкласти систему на багаторівневу ієрархію взаємодіючих компонентів. Передбачалося, що люди, які знайомі з системою, будуть розділяти систему ідентичним чином. Але це дуже малоймовірно. Більш ймовірно, що дві різні команди будуть використовувати різні підходи для розподілу системи на підсистеми. Необхідно розробити загальне керівництво з розділення системи, особливо складної системи, що складається з тисяч компонентів.

При декомпозиції системи треба вирішити процес декомпозиції складності і розміру підсистем. Ці рішення важливі, оскільки вони безпосередньо впливають на результати архітектурного аналізу. Співвідношення цілих і складових частин повинно бути постійним протягом усього поділу системи.

Приклад застосування агрегації розглядається у роботі, метою якої є розробка алгоритму зменшення розмірів великих мереж для більш ефективного моделювання мережі [8]. Метою дослідження є моделювання системи повітряного простору, яка містить велику кількість аеропортів. Моделювання реальної системи може зайняти багато часу, тому необхідні методи для моделювання великих систем з відповідним балансом точності та швидкості.

У багатьох мережах черг показники продуктивності цікавлять лише для

невеликої кількості вузлів у мережі. Інші вузли цікавлять лише з того боку, як вони впливають на показники продуктивності в основних вузлах. Підхід роботи полягає в розподілі мережі черги на дві підмережі: C та R.

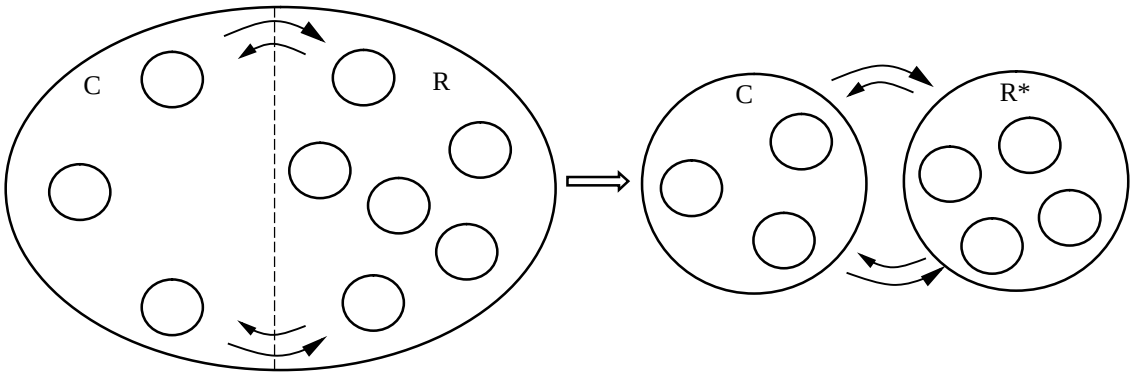

Рисунок 1.1 – Зменшення мережі

Підмережа C містить основні вузли, а підмережа R містить інші вузли. Мета полягає в перетворенні підмережі R до меншої підмережі R* таким чином, щоб моделювання C з R* наближалося до моделювання C з R. Показники ефективності C повинні бути однаковими в обох мережах, але показники продуктивності в R* та R не обов’язково повинні співпадати. У перетворенні мережі агрегація полягає в тому, що R зводиться до меншої підмережі R*. Таким чином, перетворена мережа менша, ніж вихідна мережа.

1.2 Засоби представлення систем

Існує багато систем, що складаються з окремих частин або компонентів, які певним чином пов’язані між собою [9]. Приклади включають Інтернет та людські суспільства, що представляють собою сукупність людей, які пов’язані між собою соціальними взаємодіями.

Існує декілька аспектів цих систем. Першим аспектом є характер окремих компонентів, наприклад, як працює комп’ютер, або як діє людина. Другий аспект – це характер зв’язків та взаємодій між компонентами,

наприклад, протоколи зв’язку, що використовуються в Інтернеті, або динаміка людських дружніх стосунків. Третій аспект цих систем, який має вирішальне значення для поведінки системи, – це схема зв’язків між компонентами. Схема зв’язків у системі може бути представлена у вигляді мережі, де компоненти системи представлені вершинами мережі, а зв’язки представлені дугами.

Одним з найвідоміших і найбільш широко вивчених прикладів мережі є Інтернет – комп’ютерна мережа даних, у якій вершинами є комп’ютери, а дугами представлені фізичні з’єднання між ними, наприклад, оптоволоконні кабелі.

Другим прикладом є соціальні мережі. Соціальна мережа – це мережа людей або груп людей, наприклад, компаній. Люди або групи представлені вершинами мережі, а дуги представляють зв’язки між ними, наприклад, знайомства або ділові відносини між компаніями.

Третьою сферою, в якій мережі стали важливими за останні роки, є біологія, де мережі зустрічаються в багатьох випадках. Зустрічаються конкретні фізичні мережі, наприклад, нейронні мережі – мережі зв’язків між нейронами в мозку. Також зустрічаються більш абстрактні мережі, наприклад, харчова мережа – це екологічна мережа, в якій біологічні види в екосистемі представлені вершинами, а дуги представляють зв’язки між об’єктами живлення.

Структура мережі, а саме конкретна схема взаємодій, може мати великий вплив на поведінку системи. Наприклад, схема з’єднання між комп’ютерами в Інтернеті впливає на маршрути, якими передаються дані, та на ефективність транспортування даних. Якщо не відома структура мережі, то неможливо повністю дізнатися, як вона працює.

Мережа є спрощеним представленням системи, яке перетворює систему до абстрактної структури, що охоплює лише основні схеми зв’язків. Вершини та дуги в мережі можуть бути позначені додатковою інформацією, щоб показати більше деталей системи, але частина інформації втрачається в

процесі перетворення системи до мережі. Це має свої недоліки та переваги.

Вчені в різних сферах розробили широкий набір математичних, обчислювальних та статистичних інструментів для аналізу, моделювання та розуміння мереж. Багато інструментів представляють мережу як набір вершин і дуг, та результати розрахунків дають інформацію про мережу, наприклад, у якої вершини найбільша кількість зв’язків або яка довжина шляху від однієї вершини до іншої.

Інші інструменти використовують мережеву модель та можуть робити математичні прогнози щодо процесів, які відбуваються в мережах, наприклад, як буде проходити трафік в Інтернеті. Оскільки такі інструменти працюють з мережами в їх абстрактній формі, ці інструменти теоретично можна застосувати до будь-якої системи, що представлена у вигляді мережі. Таким чином, мережі є загальним, але потужним засобом представлення схем зв’язків та взаємодій між частинами системи.

1.3 Характеристики та моделі мереж

Виділяють наступні характеристики мереж.

Міра центральності – це характеристика, що визначає, наскільки важливі вершини або ребра в мережевій системі. Існує велика кількість математичних засобів визначення міри центральності вершини, що базуються на різних поняттях того, що означає бути центральним у мережі. Одним з прикладів є ступінь. Ступінь вершини в мережі – це кількість ребер, що з’єднані з нею. Наприклад, в Інтернеті ступенем є кількість з’єднань, що має комп’ютер, маршрутизатор чи інший пристрій. У спрямованих мережах вершини мають як вхідну ступінь, так вихідну ступінь. У багатьох випадках вершини з найвищим ступенем в мережі – ті, що мають найбільше з’єднань – відіграють важливу роль у функціонуванні системи, а отже, ступінь може бути показником для зосередження уваги на найважливіших елементах мережі.

Геодезична відстань – це відстань між двома вершинами в мережі, що визначається як мінімальна кількість дуг, які потрібно пройти, щоб дістатися від однієї вершини до іншої.

Також в мережах виділяють таке поняття як кластери або спільноти. Прикладом кластерів є розділення соціальних мереж до підспільнот – тісно зв’язаних груп людей у більшій мережі. Також у мережі ділових відносин між компаніями можна виділити кластери, які формуються з множини компаній, що працюють у певних галузях економіки. Зв’язки можуть бути сильнішими, наприклад, між парою комп’ютерних компаній або парою біотехнологічних компаній, ніж між комп’ютерною компанією та біотехнологічною компанією. Спосіб розподілення мережі на кластери може показати рівні та концепції організації мережі, які нелегко визначити без мережевих даних, а також може допомогти зрозуміти структуру системи.

Виділяють наступні моделі мереж: випадковий граф та безмасштабні мережі.

Випадковий граф – це мережа, в якій певний набір параметрів приймає фіксовані значення, але, з іншої сторони, мережа є випадковою. Одним з прикладів випадкового графа є мережа, в якій задається лише кількість вершин n та кількість дуг m. Тобто m дуг розміщуються між n вершинами випадково. З усіх можливих пар вершин випадково вибирається m пар вершин і з’єднуються дугою. Цей граф визначається як G (n, m).

Безмасштабні мережі – це мережі, що містять важливі вузли або хаби, які з’єднані з дуже великою кількістю інших вузлів [10]. Такі мережі стійкі до випадкових відмов, але уразливі до керованих атак. Наявність хабів підтверджує, що складні мережі мають певну архітектуру і працюють за правилами, які застосовуються до клітин, комп’ютерів, мов і соціуму [11].

Раніше складні мережі розглядалися як випадкові мережі. Так як вузли у випадкових мереж з’єднаються один з одним випадковим чином, тому більшість вузлів має приблизно однакове число зв’язків. З’єднання вузлів здійснюється згідно з розподілом Пуассона. У таких мережах дуже рідко

можна зустріти вузол, число зв’язків якого набагато менше або набагато більше ніж середнє значення.

Рисунок 1.2 – З’єднання вузлів у випадкових мережах

На відміну від випадкових мереж безмасштабні мережі містять хаби – вузли з дуже великою кількістю зв’язків. З’єднання вузлів здійснюється за степеневим законом. Таким чином, більшість вузлів мають невелику кількість з’єднань, а мала частина вузлів має дуже велику кількість з’єднань, тому система «не має розміру». SFN зустрічаються в багатьох сферах, наприклад, мережа наукових публікацій, зв’язки компаній в бізнесі.

Теорія випадкових графів не пояснює існування хабів. Дана модель передбачає, що число вершин відомо заздалегідь, до з’єднання вузлів дугами. Але більшість мереж поступово збільшується. Також всі вузли не однакові. Існує такий процес як «переважне приєднання», який полягає в тому, що якщо у вузла є багато зв’язків, то швидше за все до нього буде приєднаний новий вузол. Наприклад, найбільш цитовані статті в науковій літературі читає і цитує все більша кількість дослідників.

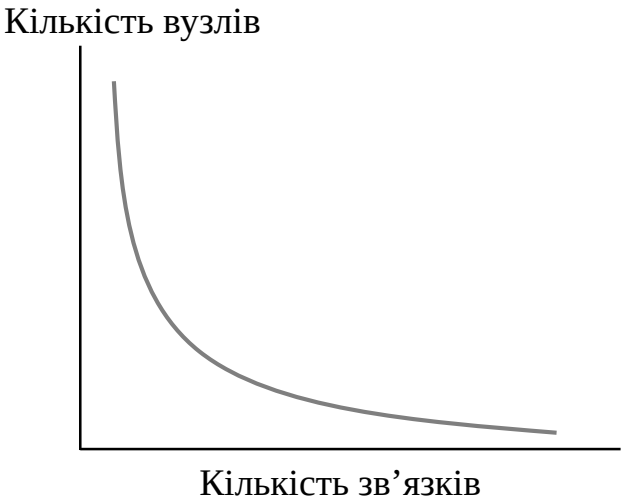

Рисунок 1.3 – З’єднання вузлів у безмасштабних мережах

Таким чином, виділяють два механізми, які пояснюють існування хабів: зростання і переважне приєднання. Їх робота полягає в наступному: новий вузол, що з’являється в системі, має тенденцію з’єднуватися з вузлом, який має найбільшу кількість зв’язків. В результаті чого з часом такий вузол або хаб набуває все більше зв’язків на відміну від інших вузлів.

1.4 Постановка завдання

Складні системи містять велику кількість компонентів та зв’язків між ними, через це такі системи мають велику розмірність та обчислювальну складність. Для зменшення розмірності системи здійснюють еквівалентні перетворення топологічної структури системи. Завдяки зменшенню розмірності системи зменшується математична складність системи, час аналізу системи та час вирішення задач. Для перетворення структури системи застосовується агрегація структур систем, яка полягає в об’єднанні елементів системи у підсистеми, у результаті чого створюється ієрархічна система.

Завданням атестаційної роботи є аналіз агрегації структур систем, розробка програмної платформи та оцінка ефективності програмної платформи та ефективності агрегації структур систем. Треба оцінити ефективність агрегації, перевірити коректність параметрів на різних рівнях системи та розглянути аспект втрати інформації під час застосування агрегації.

Програмна платформа включає наступні модулі: модуль аналізу характеристик реальної системи, модуль генератора мережевих структур, модуль агрегації структурної моделі та модуль пошуку максимального потоку. Треба розробити генератор мережевих структур для застосування агрегації на великій кількості систем з метою оцінки ефективності агрегації та перевірки коректності параметрів на різних рівнях системи. Модуль агрегації структурної моделі здійснює агрегацію структури системи з метою зменшення розмірності системи. Модуль пошуку максимального потоку використовується для оцінки ефективності агрегації структур систем.

2 МОДЕЛІ СТРУКТУР СКЛАДНИХ СИСТЕМ

2.1 Моделювання складних систем

Складна система складається з множини компонентів, між якими відбувається взаємодія. Компоненти системи також містять в собі множину компонентів. Важливою характеристикою складних систем є те, що такі системи ієрархічні і складаються з множини рівнів [12].

Існують наступні техніки моделювання складних систем.

Декомпозиція – це поділ основної проблеми на декілька менших проблем. Завдяки декомпозиції складна система може бути розділена на множину підсистем на іншому рівні структурної організації. Кожна підсистема може розглядатися як незалежна.

Абстракція – це визначення спрощеної моделі системи для того, щоб отримати систему, яка підкреслює певні деталі та властивості та одночасно приховує інші.

Організація – це процес визначення та контролю взаємодії між різними компонентами системи. Це дозволяє згрупувати певну кількість основних компонентів і працювати з ними як з підсистемою більш високого рівня.

Застосування цих технік дає можливість використовувати ієрархічний граф для представлення складної системи. Це дозволяє представляти структуру системи та взаємодію компонентів всередині неї, отже, це дає можливість прогнозувати її поведінку в часі. Використовуючи даний метод моделювання, результати дослідження складних систем можуть застосовуватися в біології, екології та інших областях.

Для моделювання систем використовують ієрархічний граф і мультиагентні системи. Ієрархічний граф – це граф, який містить два типи вузлів: елементарні та складні. Елементарний вузол – це вузол, у якого немає внутрішнього стану. Складний вузол – це вузол, у якого є внутрішній стан –

інший ієрархічний граф. Складний вузол містить ієрархічний граф, який може містити інші складні вузли. Іншими словами, будь-який складний вузол в ієрархічному графі – це компонент іншого складного вузла на більш високому рівні системи. Таким чином, складний вузол може використовуватися для представлення складної системи, де кожен компонент системи також є складним вузлом. Отже, вся система є найвищим рівнем ієрархії.

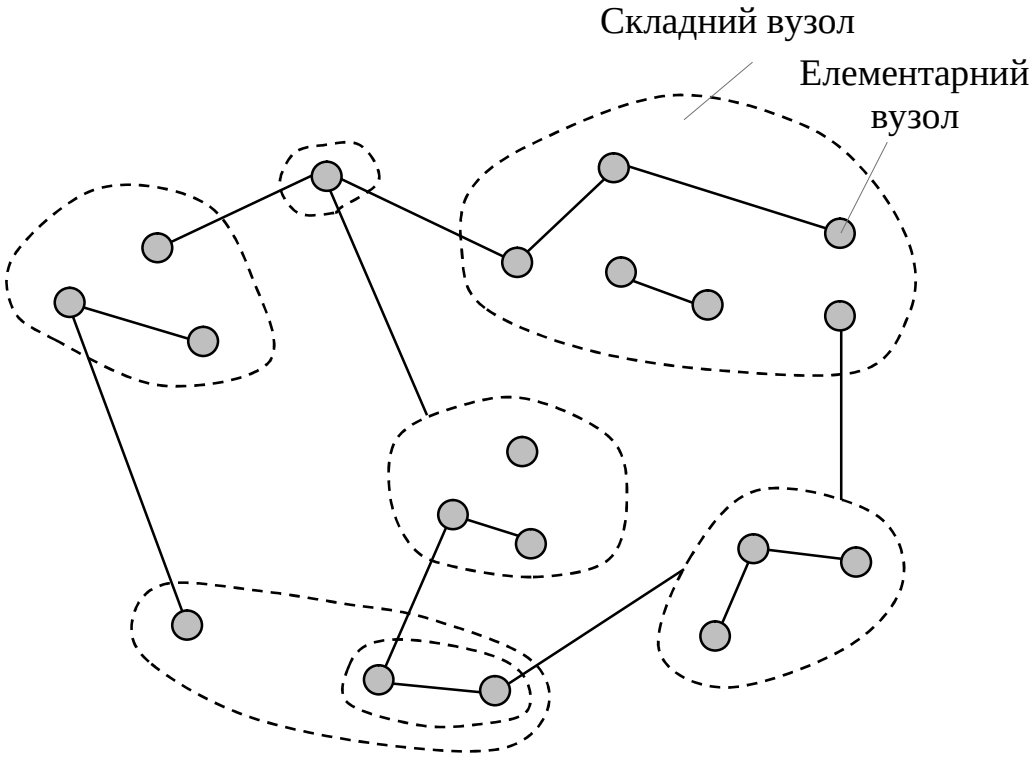

Рисунок 2.1 – Ієрархічний граф

Ієрархічний граф задається наступним співвідношенням:

$$G(N,E), \tag{2.1}$$

де E – множина дуг;

N – множина елементарних та складних вузлів.

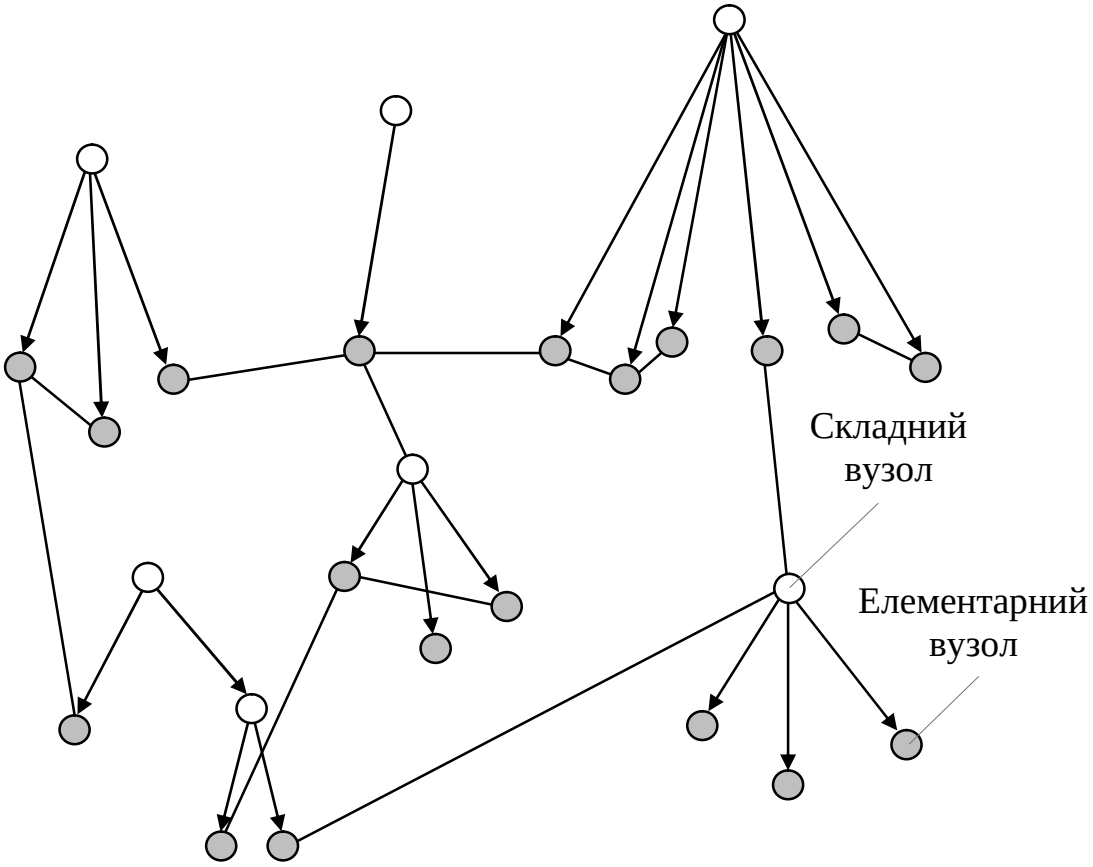

Рисунок 2.2 – Ієрархічний граф

Опис елементарного вузла задається наступним співвідношенням:

$$AN = < ID, L >, \tag{2.2}$$

де ID – ідентифікатор вузла;

L – позначення вузла.

Опис складного вузла задається наступним співвідношенням:

$$CN = < ID, L, C >, \tag{2.3}$$

де ID – ідентифікатор вузла;

L – позначення вузла;

C – внутрішній стан вузла, тобто вузол є складним або елементарним.

Використання HG дозволяє управляти ієрархічними аспектами складних систем, наприклад, рівнем структурної організації. Граф використовується для моделювання різних ситуацій, також він є проміжним з'єднанням між фізичною системою та математичною моделлю.

Мультиагентні системи використовуються в моделюванні для того, щоб зрозуміти процеси, які відбуваються в складних системах, наприклад, в біологічних системах. MAS в основному складаються з компонентів, які можна представити у вигляді колекції взаємодіючих об'єктів, що функціонують без будь-якого зовнішнього впливу. У таких системах розглядається поведінка окремого агента.

Агент – це комп'ютерна система, що знаходиться в певному середовищі, яка здатна до гнучких автономних дій, які допомагають досягти поставленої мети. Агент визначається як активний об'єкт, який:

- сприйнятливий – агент повинен сприймати навколишнє середовище і своєчасно реагувати на зміни, що в ній відбуваються;
- проактивний – агент не повинен просто реагувати на навколишнє середовище, він повинен вміти пристосовуватися, діяти відповідно до поставлених цілей та проявляти ініціативу, де це доречно;
- соціальний – агент повинен вміти взаємодіяти з іншими агентами для досягнення поставлених цілей.

Таким чином, будь-який компонент у складній системі може бути представлений як агент, тому що він робить автономні дії. Кожен агент має множину станів. У певний час агент знаходиться в стані S. Якщо немає зовнішнього впливу, система залишається в поточному стані. Зовнішні події агент отримує через вхідний контакт. Реакцію агента на зовнішній вплив і відповідні зміни агента можна описати математичної функцією.

Складну систему можна представити як колекцію взаємодіючих об'єктів, які функціонують без зовнішнього впливу для досягнення поставлених цілей. Ієрархічний граф є з'єднанням між реальною складною системою та моделюванням динаміки системи за допомогою взаємодіючих

агентів. Використання ієрархічного графа для моделювання структурної організації складних систем допомагає у визначенні агентів для моделювання. Також взаємодії між підсистемами можуть бути представлені за допомогою зв’язків між агентами.

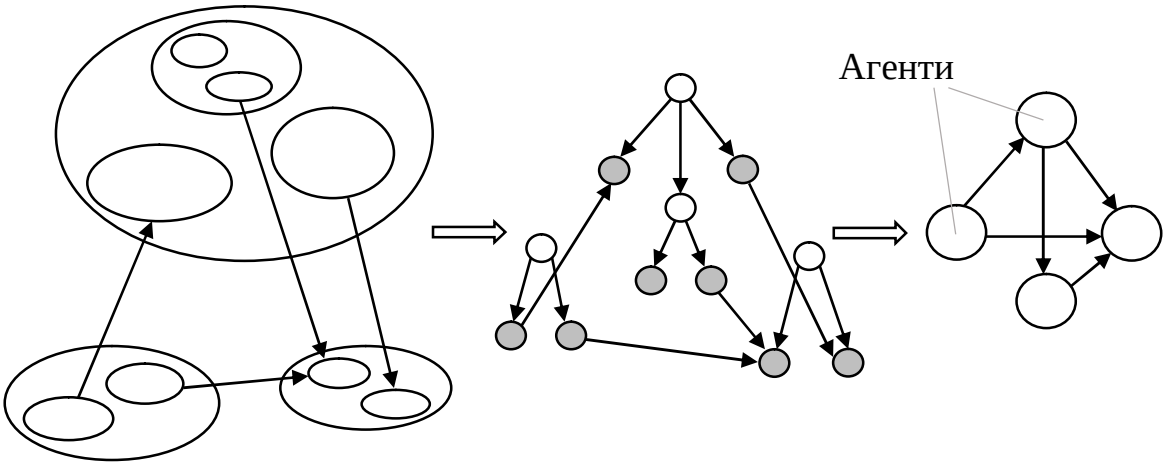

Рисунок 2.3 – Співвідношення між складною системою, ієрархічним графом та агентами

2.2 Визначення агрегації структурної моделі

Складна система – це велика колекція взаємозалежних одиниць. Беручи до уваги опис на рівні окремих одиниць, може бути корисним розділити колекцію на групи одиниць, які можна розглядати як підсистеми і отримати ефективні взаємодії між ними. Таким чином, можна отримати опис на більш високому рівні системи.

Опис на високому рівні не повинен зберігати всю інформацію на низькому рівні, але в ідеалі взаємодії на більш високому рівні виробляють такий самий результат, як і спостереження агрегованих одиниць для вхідної системи. Якщо ведеться запис про процедуру агрегації, то можна вивести опис нижчого рівня з опису більш високого рівня.

Нехай N та $\overline{N}$ – множини елементів [13]:

$$N = \{\alpha_k\}_{k=1,\dots,n} \tag{2.4}$$

$$\overline{N} = \{\beta_i\}_{i=1,\dots,m}, \tag{2.5}$$

де m ≤ n.

Якщо здійснюється мапінг ϕ з мережі N у мережу $\overline{N}$:

$$\phi: N \rightarrow \overline{N}, \tag{2.6}$$

то N – це початкова мережа, $\overline{N}$ – це агрегована мережа, ϕ – це мапінг агрегації.

Мережі N та $\overline{N}$ складаються з елементів:

$$N = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\} \tag{2.7}$$

$$\overline{N} = \{\beta_1, \beta_2, \beta_3\} \tag{2.8}$$

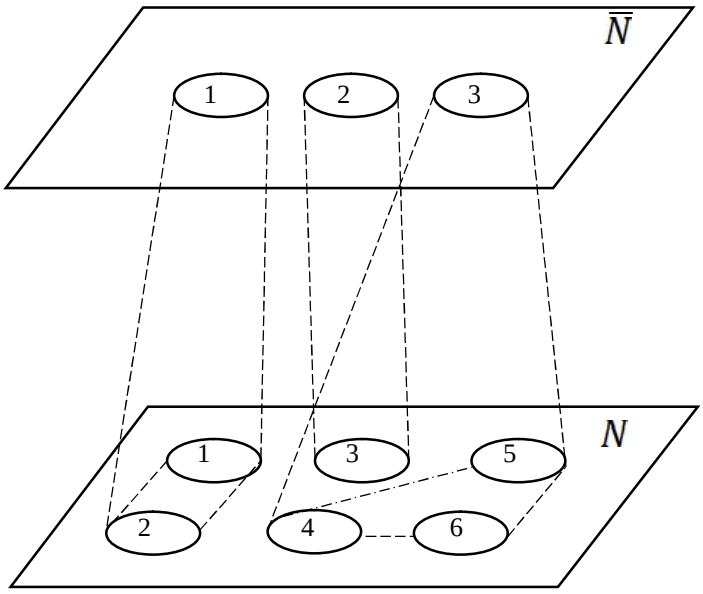

Рисунок 2.4 – Агрегація мережі

SMA представляється матрицею агрегації:

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

У цьому випадку

$$\beta_1 = \alpha_1 + \alpha_2 \tag{2.10}$$

$$\beta_2 = \alpha_3 \tag{2.11}$$

$$\beta_3 = \alpha_4 + \alpha_5 + \alpha_6 \tag{2.12}$$

Так як агрегована мережа $\overline{N}$ початкової мережі N залежить від матриці агрегації S, тому вона може бути представлена як функція від S, $\overline{N}(S)$.

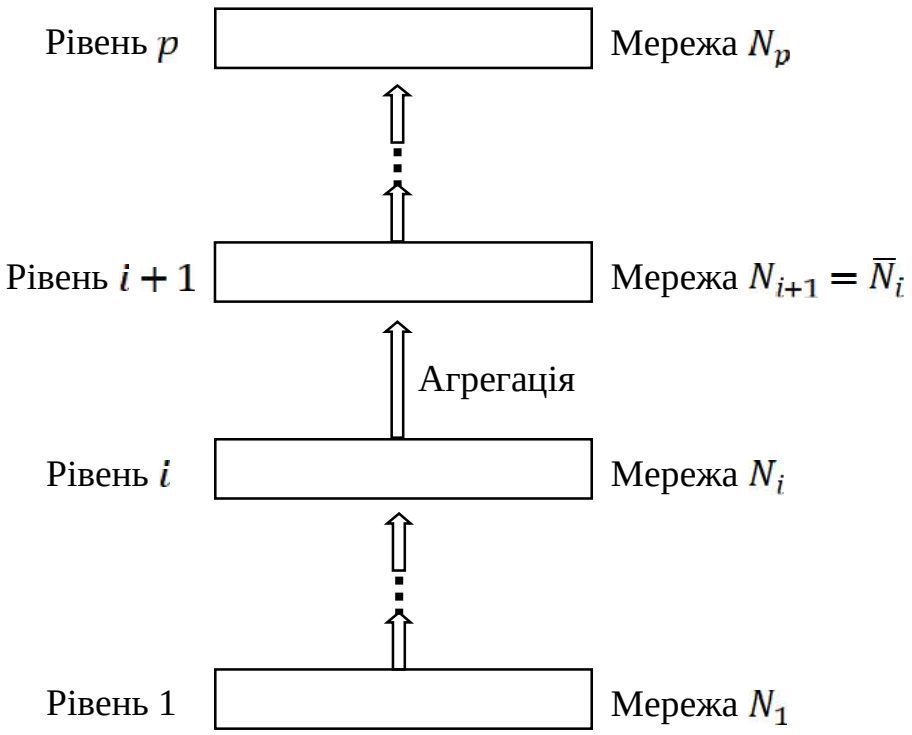

Рисунок 2.5 – Ієрархічна структура мережі

Зазвичай HCS мають декілька рівнів, кожен з яких характеризується різними функціями. Наприклад, в екосистемах можна розглядати подію на рівні клітини, органу, організму, виду та спільноти.

Мережа з p ієрархічними рівнями виражається символом N. Мережа, що представляє i-й ієрархічний рівень, виражається символом N_i. Отже N складається з $\{N_i\}_{i=1,\dots,p}$. У ієрархічній структурі N верхній рівень розглядається як агрегована мережа нижніх рівнів. Тобто мережа, що показує i-й рівень N_i.є агрегованою версією мережі на (i-1) рівні, тобто N_{i-1}.

2.3 Проблема втрати інформації

Під час агрегації мережі виконується наступне співвідношення:

$$M(N) \geq M(\bar{N}), \tag{2.13}$$

де M – це інформація, що міститься у структурі мережі.

Формула (2.7) показує, що інформація не може збільшуватись під час процесу агрегації, тобто різниця $M(N) - M(\bar{N})$ ніколи не буде від’ємною. Втрата інформації може розглядатися як вартість J агрегації:

$$J = M(N) - M(\bar{N}) \tag{2.14}$$

Вартість J агрегації завжди порівнюється з економічною вартістю відсутності агрегації, тобто вартістю збору додаткових даних або іншого аналізу розширеної системи. Часто витрати фіксуються фіскальними обмеженнями та обмеженнями робочої сили, тому проблема зводиться до вибору розміру агрегованої системи m, звертаючи увагу на наявні ресурси, наприклад, робоча сила, фінансування. Процес агрегації продовжується до

створення m підсистем, обраних таким чином, щоб мінімізувати втрати інформації.

Елементи, що складають підсистеми вибирають таким чином, щоб мінімізувати втрату інформації. Тобто вартість агрегації J зменшується шляхом вибору оптимального S^*:

$$J^* = \min_{\{S\}} J = M(N) - M[\bar{N}(S^*)] \quad (2.15)$$

Ієрархічна структура, де кожен рівень є агрегацією більш нижчого рівня системи, є дуже важливим особливим випадком, що має велику кількість застосувань. На загальну інформацію в системі впливає взаємодія між усіма рівнями ієрархії. Структура системи може бути спрощена тільки тоді, коли ієрархія строго вкладена, тобто складається з каскаду мапінгів. Важливо уникати такого способу об’єднання елементів у підсистеми, що призводять до приховування важливих аспектів організації в системі. Це призводить до вибору таких схем агрегації, що мінімізують зменшення спільної інформації у асоційованій мережі.

3 ЗМЕНШЕННЯ РОЗМІРНОСТІ МЕРЕЖЕВИХ СИСТЕМ

3.1 Формальний опис агрегації структурної моделі

Метою агрегації є створення ієрархічної системи та зменшення розмірності системи. Система S складається з елементів $C_1,C_2,...,C_N$, які мають вхідні контакти $X_i^{(j)}$, де i = 1,2, ..., m і вихідні контакти $Y_l^{(j)}$, де l = 1,2,...,r [14, 15]. Елемент C_j містить m_j вхідних контактів; контакт $X_i^{(j)}$ приймає елементарні сигнали $x_i^{(j)}(t)$; i=1,2,...,m_j; j=1,2,...,N. Елемент C_j містить r_j вихідних контактів; контакт $Y_l^{(j)}$ видає елементарні сигнали $y_l^{(j)}(t)$; l = 1,2,...,r_j.

Фіктивним елементом C_0 представляється зовнішнє середовище системи S, вхід якого містить m_0 вхідних контактів $X_i^{(0)}$, а вихід містить r_0 вихідних контактів $Y_l^{(0)}$. Таким чином, кожен C_j елемент системи S характеризується множиною вхідних контактів $X_1^{(j)}$, $X_2^{(j)}$,..., $X_m^{(j)}$ та множиною вихідних контактів $Y_1^{(j)}$, $Y_2^{(j)}$,..., $Y_m^{(j)}$.

Для формального опису з’єднання елемента C_j з іншими елементами та зовнішнім середовищем використовується математична модель, що складається з множин:

$$\left[X_i^{(j)}\right]_1^m, \tag{3.1}$$

$$\left[Y_l^{(j)}\right]_1^r, \tag{3.2}$$

де i – номер вхідного контакту, i=1,2,...,m;

l – номер вихідного контакту, l=1,2,...,r;

j – номер елемента системи, j=1,2,...,N.

До вхідного контакту будь-якого елементу системи підключається не

більше ніж один елементарний канал, а до вихідного контакту може бути підключено будь-яке кінцеве число елементарних каналів, за умови, що до входу одного і того ж елемента системи направляється не більше ніж один зі згаданих елементарних каналів.

Згідно з припущенням, що кожному вхідному контакту $X_i^{(j)}$ відповідає не більше ніж один вихідний контакт $Y_l^{(k)}$, вводиться однозначний оператор:

$$Y_l^{(k)} = R(X_i^{(j)}), \tag{3.3}$$

де область визначення на множині:

$$\cup_{j=0}^{N} \left[X_i^{(j)}\right]_1^m; \tag{3.4}$$

та область значень на множині:

$$\cup_{k=0}^{N} \left[Y_l^{(k)}\right]_1^r. \tag{3.5}$$

Таблиця 3.1 – Сполучення елементів в системі

i\j	1	2	3
0	1,2,54	4,3,23	-,-,-
1	0,1,68	3,1,60	4,1,60
2	3,2,28	4,2,41	-,-,-
3	0,2,45	2,1,23	-,-,-
4	0,2,58	1,1,17	2,1,36

Оператор (3.3) з областю визначення і областю значень називається схемою сполучення елементів в системі S і оператором сполучення або формальною моделлю структури системи.

Оператор сполучення можна задати у вигляді таблиці, в якій на перетині рядків з номерами елементів системи j та стовпців з номерами її вхідних контактів і розташовуються пари чисел (k, l), що вказують номер елемента k та номер вихідного контакту l, з яким з'єднаний контакт $X_i^{(j)}$.

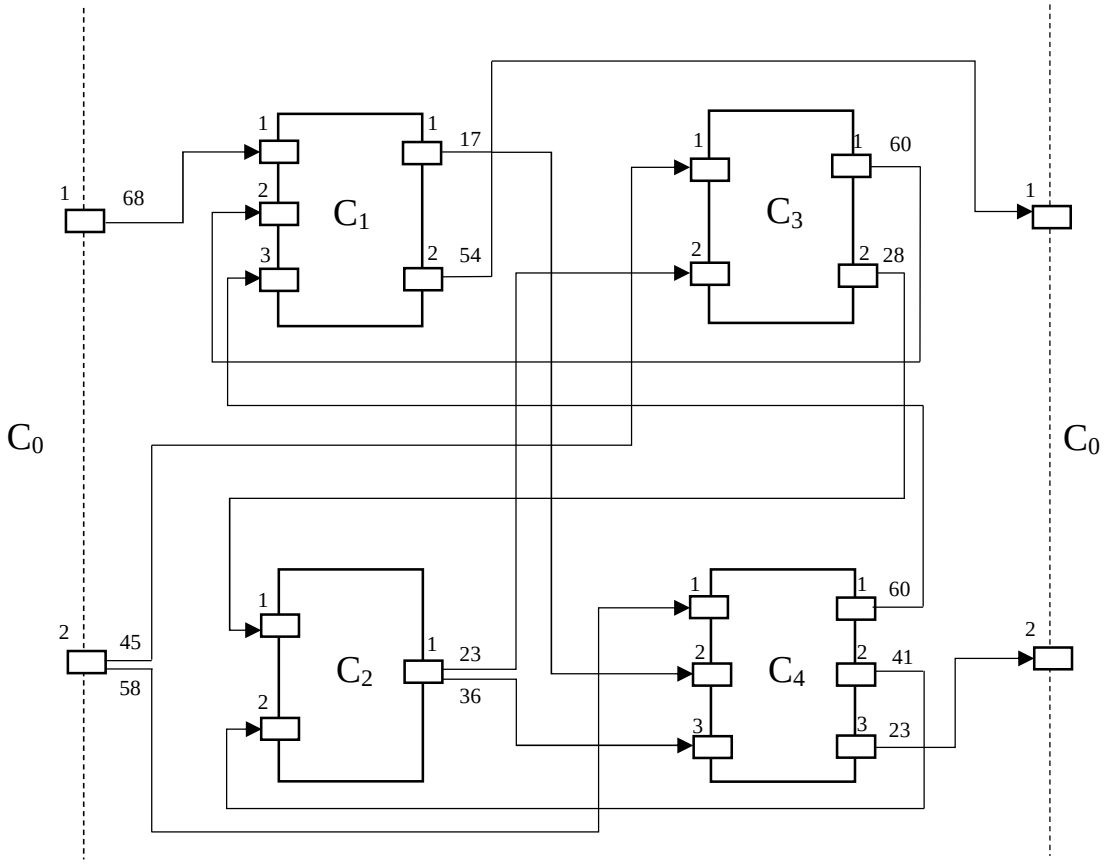

Рисунок 3.1 – Схема взаємодії елементів системи

Відповідність між $Y_l^{(к)}$ і $X_i^{(j)}$, що описується оператором (3.3), не є взаємно однозначною. Таким чином, маємо неоднозначний оператор:

$$X_i^{(J)} = R^{-1}\left(Y_l^k\right), \quad (3.6)$$

де $X_i^{(j)}$ – вхідні контакти елемента;

$Y_l^{(k)}$ – вихідні контакти елемента.

Першим етапом агрегації є об’єднання елементів у підсистеми. Складна система S розділяється на підсистеми S_μ, де $\mu = 1, 2, ..., M$, які містять не менше ніж по одному елементу. Елемент C_j входить тільки в одну з підсистем S_μ. Якщо елемент C_j, не входить ні в одну з підсистем, то цей елемент самостійно формує підсистему, наприклад, $S_{\mu 0}$.

Здійснемо агрегацію наступним чином: $S_{\mu 0}=\{C_0\}$; $S_{\mu 1}=\{C_1,C_2\}$; $S_{\mu 2}=\{C_3,C_4\}$.

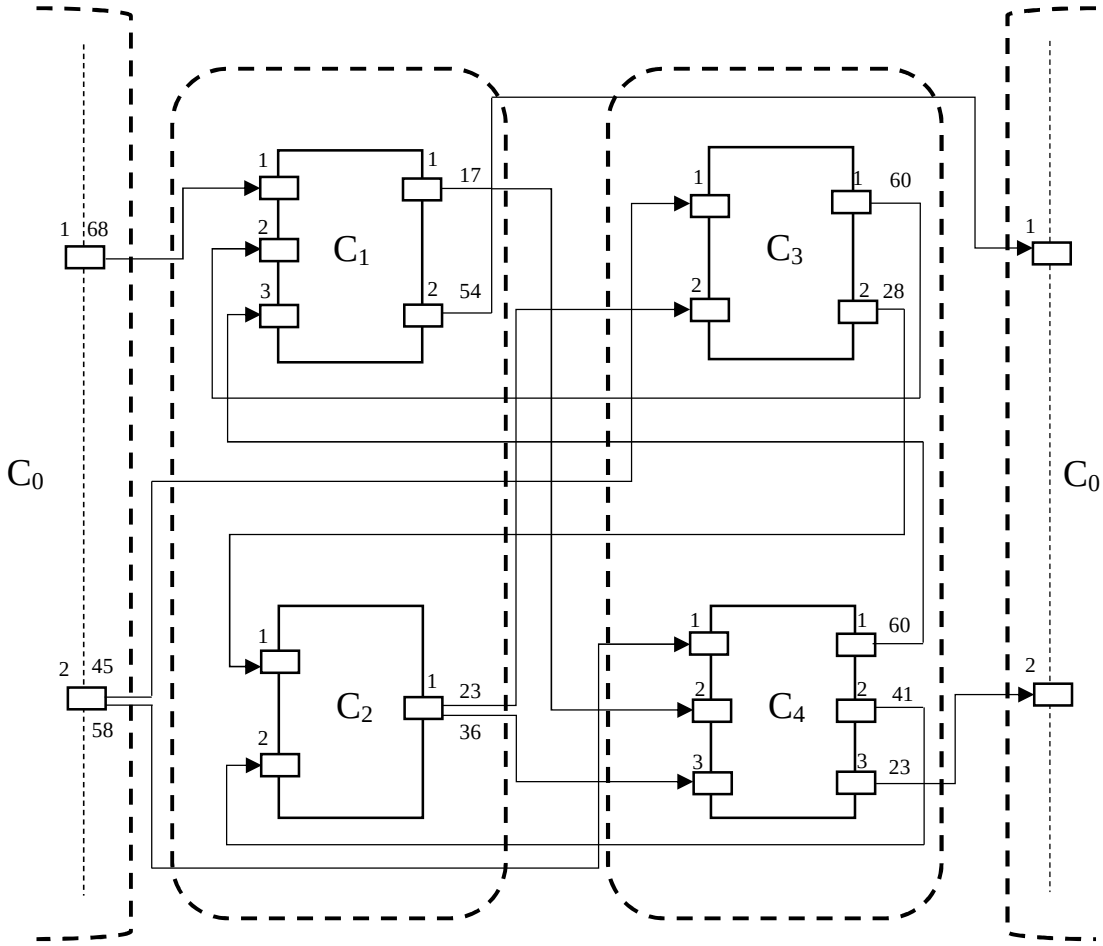

Рисунок 3.2 – Об’єднання елементів у підсистеми

Підсистема S_μ є складною системою, подібно системі S, та одночасно підсистема S_μ є елементом системи S. У першому випадку підсистема S_μ повинна мати зовнішнє середовище – фіктивний елемент, через C_0^μ. В цьому випадку зовнішнє середовище, яке представляється фіктивним елементом

C_0^{μ}, взаємодіє з підсистемою S_μ через свої вхідні контакти $X_i^{(0)\mu}$, які з'єднані з виходами елементів підсистеми S_μ, а також через вихідні контакти $Y_l^{(0)\mu}$, які з'єднані з входами елементів підсистеми S_μ.

Наступним етапом агрегації є визначення фіктивних контактів підсистеми.

Існує дві множини контактів елементів підсистеми S_μ:

- множина вихідних контактів всіх елементів C_j, де $C_j \in S_\mu$, які з'єднані елементарними каналами з вхідними контактами елементів C_k, де $C_k \notin S_\mu$, а також з вхідними контактами фіктивного елемента C_0:

$$\left[Y_l^{(j)}\right]_\mu, \tag{3.7}$$

де μ – номер підсистеми.

- множина вхідних контактів всіх елементів C_j, де $C_j \in S_\mu$, з'єднаних елементарними каналами з вихідними контактами елементів C_k, де $C_k \notin S_\mu$, а також з вихідними контактами фіктивного елемента C_0:

$$\left[X_l^{(j)}\right]_\mu, \tag{3.8}$$

де μ – номер підсистеми.

Для наведених множин потрібно визначити фіктивні контакти, які з'являються при поділі системи S на підсистеми S_μ. Таким чином, буде сформовано дві групи фіктивних контактів: перша група буде характеризувати підсистему S_μ як складну систему, а друга група контактів буде характеризувати підсистему її як елемент системи S.

Розглянемо множину $[Y_l^{(j)}]_\mu$, всіх елементів C_j, де $C_j \in S_\mu$, які з'єднані елементарними каналами з вхідними контактами елементів C_k, де $C_k \notin S_\mu$, а також з вхідними контактами фіктивного елемента C_0.

Кожному $Y_l^{(j)} \in [Y_l^{(j)}]_\mu$ ставиться у відповідність пара операторів:

$$Y_l^{(\mu)} = Q_\mu(Y_l^{(j)}), \tag{3.9}$$

$$X_i^{(0)\mu} = Q'_\mu(Y_l^{(j)}). \tag{3.10}$$

Оператори (3.9), (3.10) називаються операторами нумерації фіктивних контактів підсистеми S_μ, пов'язаних з контактами $Y_l^{(j)}$ всіх елементів C_j, що належать підсистемі S_μ, і з'єднаних елементарними каналами з вхідними контактами елементів C_k, котрі належать до підсистеми S_μ, і з вхідними контактами фіктивного елемента C_0. Оператори Q_μ и Q'_μ задаються таблицею нумерації фіктивних контактів.

Алгоритм формування фіктивних контактів для множини $[Y_l^{(j)}]_\mu$ полягає в наступному. На першому етапі для підсистеми S_μ формується множина $[Y_l^{(j)}]_\mu$. Після визначаються контакти, що належать елементу C_j, де $C_j \in S_\mu$. На другому етапі кожному $Y_l^{(j)}$ з цієї множини ставиться у відповідність пара фіктивних контактів, які після нумеруються з використанням операторів (3.9), (3.10).

Таблиця 3.2 – Таблиця нумерації фіктивних контактів підсистеми $S_{\mu 2}$, підключених до контактів множини $[Y_l^{(j)}]_{\mu 1}$

Оператор	3,1	3,2	4,1	4,2	4,3
$Q_{\mu 1}$	1	2	3	4	5
$Q_{\mu 1}'$	1	2	3	4	5

Розглянемо множину $[X_l^{(j)}]_\mu$ всіх елементів C_j, де $C_j \in S_\mu$, з'єднаних елементарними каналами з вихідними контактами елементів C_k, де $C_k \notin S_\mu$, а також з вихідними контактами фіктивного елемента C_0.

Вводяться оператори нумерації фіктивних контактів:

$$X_i^{(\mu)} = P_\mu(X_i^{(j)}), \quad (3.11)$$

$$Y_l^{(0)\mu} = P'_\mu(X_i^{(j)}). \quad (3.12)$$

Оператори (3.11), (3.12) формують фіктивні контакти підсистеми S_μ, які пов'язані з контактами $X_i^{(j)}$ всіх елементів $C_j \in S_\mu$ і з'єднані елементарними каналами з вихідними контактами елементів $C_k \notin S_\mu$ і з вихідними контактами фіктивного елемента C_0. Розглянуті оператори можуть бути представлені у вигляді таблиць фіктивних контактів.

Алгоритм формування фіктивних контактів для множини $[X_l^{(j)}]_\mu$ полягає в наступному.

На першому кроці для підсистеми S_μ формується множина $[X_l^{(j)}]_\mu$. Потім визначаються контакти, що належать елементу C_k, де $C_k \notin S_\mu$.

На другому кроці кожному $X_l^{(j)}$ ставиться у відповідність пара фіктивних контактів, які після нумеруються з використанням операторів (3.11), (3.12).

Таблиця 3.3 – Таблиця нумерації фіктивних контактів підсистеми $S_{\mu 2}$, підключених до контактів множини $[X_l^{(j)}]_{\mu 1}$

Оператор	3,1	4,1	3,2	4,3	4,2
P_μ	1	1	2	2	3
P'_μ	1	1	2	2	3

Наступним етапом агрегації є визначення зв'яків між елементами в підсистемі. Розглянемо підсистему S_μ як самостійну систему, тобто C_0 і всі інші компоненти системи S представляють для S_μ зовнішнє середовище, яку позначимо як $C_0^{(\mu)}$. При побудові схеми сполучення необхідно враховувати два типи з'єднань. У першому випадку – це внутрішні з'єднання вхідних і вихідних контактів елементів C_j, таких, що $C_j \in S_\mu$. У другому випадку, це

з’єднання вхідних і вихідних контактів елементів C_j, що належать S_μ, з вхідними та вихідними контактами елемента $C_0^{(\mu)}$, що представляє зовнішнє середовище по відношенню до підсистеми S_μ.

У першому випадку, множини (3.1), (3.2) контактів для елементів $C_j \in S_\mu$ задані, а оператор сполучення представляється у вигляді (3.3).

У другому випадку, підсистема S_μ повинна мати контакти $X_i^{(0)\mu}$ и $Y_l^{(0)\mu}$, що з’єднують відповідно входи фіктивного елемента $C_0^{(\mu)}$ з виходами елементів підсистеми S_μ, також виходи фіктивного елемента $C_0^{(\mu)}$ з входами елементів підсистеми S_μ. Контакти $X_i^{(0)\mu}$ та $Y_l^{(0)\mu}$ визначаються за допомогою операторів (3.9), (3.10) і (3.11), (3.12): Q_μ', який визначає вхідні контакти $C_0^{(\mu)}$, та P_μ', який визначає вихідні контакти $C_0^{(\mu)}$.

$$\left[X_i^{(0)\mu}\right]_1^m = Q'_\mu\left(\left[Y_l^{(j)}\right]_\mu\right) \tag{3.13}$$

$$\left[Y_l^{(0)\mu}\right]_1^r = P'_\mu\left(\left[X_i^{(j)}\right]_\mu\right) \tag{3.14}$$

Схема сполучення для підсистеми S_μ як самостійної системи буде визначена за допомогою оператора сполучення R_μ – внутрішнього оператора сполучення елементів в підсистемі S_μ:

$$Y_l^{(k)} = R_\mu(X_i^{(j)}), \tag{3.15}$$

Таблиця 3.4 – Таблиця сполучення елементів підсистеми $R_{\mu 2}$

i\j	1	2	3	4	5
0	3,1	3,2	4,1	4,2	4,3
3	0,1	0,2	-,-	-,-	-,-
4	0,1	0,3	0,2	-,-	-,-

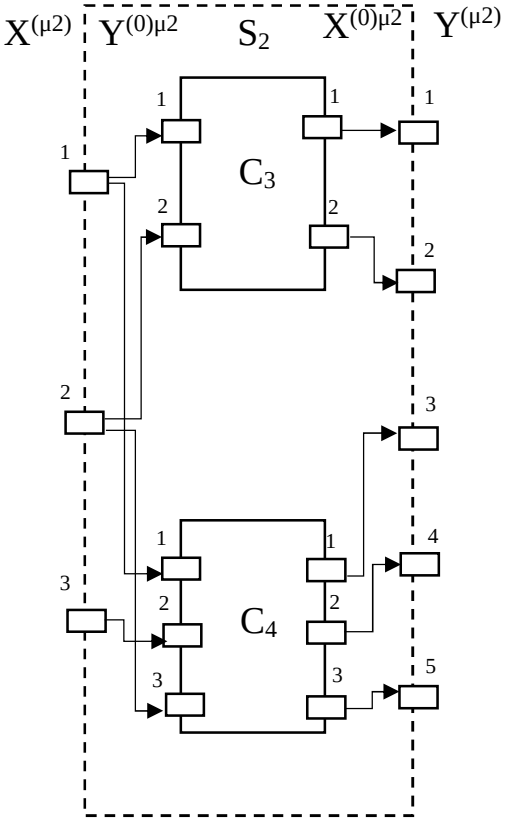

Рисунок 3.3 – Сполучення елементів підсистеми S_2

Наступним етапом агрегації є побудова другого рівня системи. Підсистема S_μ може розглядатися як елемент системи S. Підсистема характеризується:

- множиною $[X_i^{(j)}]_\mu$, яка позначає множину вхідних контактів всіх елементів $C_j \in S_\mu$, з'єднаних елементарними каналами з вихідними контактами елементів $C_k \notin S_\mu$ і вихідними контактами фіктивного елемента C_0;

- множиною $[Y_l^{(j)}]_\mu$, яка позначає множину вихідних контактів $Y_l^{(j)}$ всіх елементів C_j, що належать підсистемі S_μ, які з'єднані елементарними каналами з вхідними контактами елементів $C_k \notin S_\mu$ і вхідними контактами фіктивного елемента C_0;

- множинами фіктивних вхідних і вихідних контактів:

$$\left[X_i^{(\mu)}\right]_1^m, \tag{3.16}$$

де $X_i^{(\mu)}$ – вхідний контакт підсистеми;

$$\left[Y_l^{(\mu)}\right]_1^r, \tag{3.17}$$

де $Y_l^{(\mu)}$ – вихідний контакт підсистеми.

Елементи множин (3.17) і (3.16) формуються операторами (3.9) і (3.11).

$$\left[Y_l^{(\mu)}\right]_1^r = Q_\mu\left(\left[Y_l^{(j)}\right]_\mu\right) \tag{3.18}$$

$$\left[X_i^{(\mu)}\right]_1^m = P_\mu\left(\left[X_i^{(j)}\right]_\mu\right) \tag{3.19}$$

Для побудови схеми сполучення підсистем S_μ, $\mu = (\mu_0,\mu_1,\mu_2,\ldots..\mu_M)$ в системі S вводиться оператор сполучення підсистем:

$$Y_l^{(v)} = R_{II}(X_i^{(\mu)}), \tag{3.20}$$

де область визначення на множині:

$$\cup_{\mu=0}^{M}\left[X_i^{(\mu)}\right]_1^m, \tag{3.21}$$

область значень на множині:

$$\cup_{\mu=0}^{M}\left[Y_l^{(\mu)}\right]_1^r, \tag{3.22}$$

де M – кількість підсистем в системі.

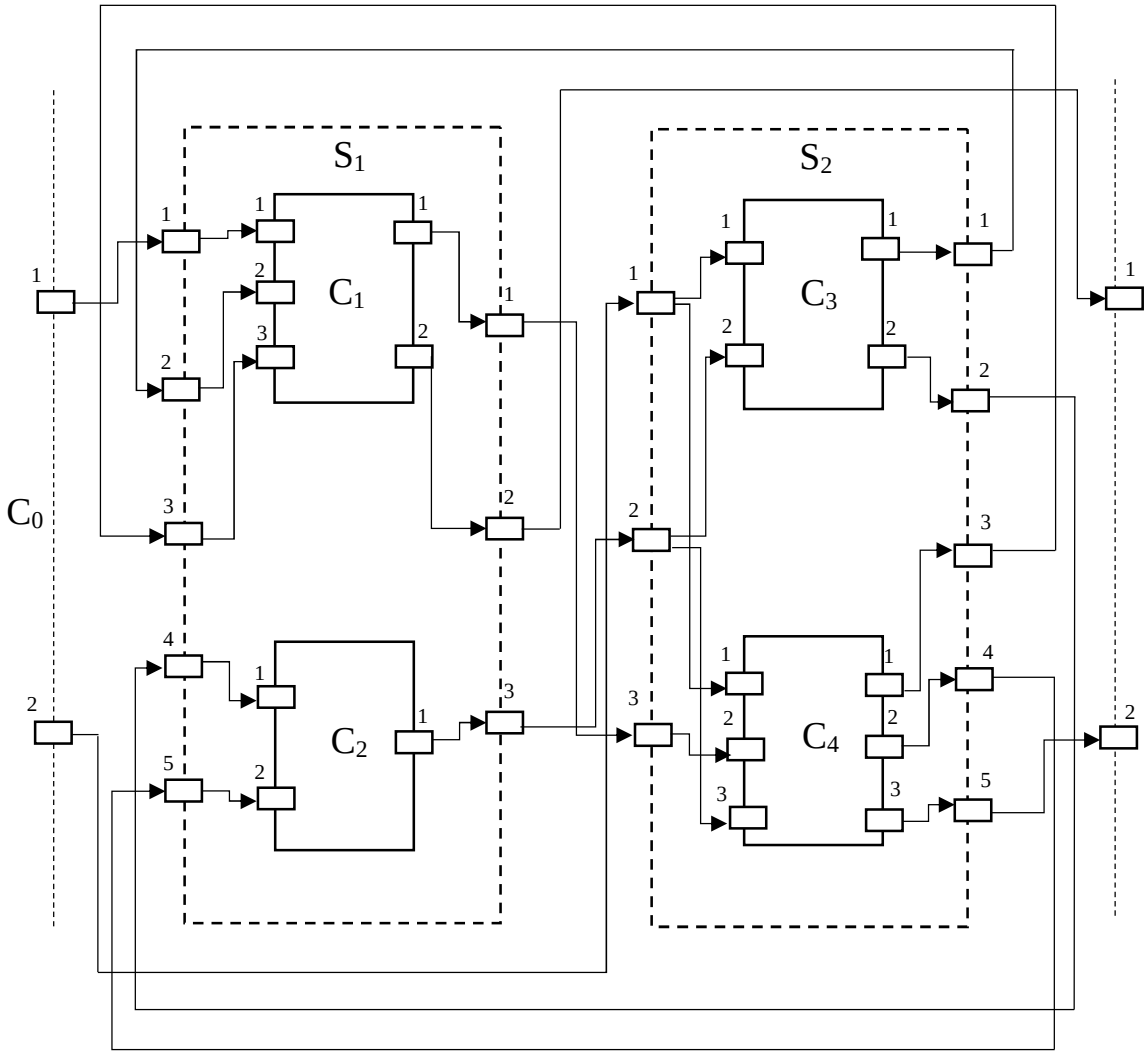

Рисунок 3.4 – Схема другого рівня системи

Оператор (3.20) вхідному фіктивному контакту $X_i^{(\mu)}$ ставить у відповідність вихідний фіктивний контакт $Y_l^{(\nu)}$, з'єднаний з $X_i^{(\mu)}$ елементарним каналом, якщо таке з'єднання в системі S існує. Для фіктивного контакту $X_i^{(\mu)}$ визначається пов'язаний з ним один з контактів:

$$X_i^{(j)} = P_\mu^{-1}(X_i^{(\mu)}), \tag{3.23}$$

де $X_i^{(j)}$ – вхідний контакт елемента C_j, такого, що $C_j \in S_\mu$. Так як $P_\mu^{-1}(X_i^{(\mu)})$ є неоднозначним оператором, в результаті може виявитися більше ніж один контакт $X_i^{(j)}$.

Таблиця 3.5 – Сполучення елементів на другому рівні системи

i\j	1	2	3	4	5
0	1,2,54	2,5,23	-,-,-	-,-,-	-,-,-
1	0,1,68	2,1,60	2,3,60	2,2,28	2,4,41
2	0,2,103	1,3,59	1,1,17	-,-,-	-,-,-

Сукупність схеми сполучення R_{II} підсистем S_μ в системі S і схем сполучення R_μ елементів C_j в підсистемах S_μ називаються дворівневою схемою сполучення системи S.

3.2 Методи проектування генераторів структур складних систем

Дослідникам часто потрібно проводити попередню оцінку нових схем з точки зору ефективності топології мережі, здатності мережі витримувати великі навантаження та залишатися функціональною. Але створити реальну систему з метою експериментального вивчення майже неможливо через великий масштаб сучасних мережевих систем. У цьому випадку оцінюються запропоновані рішення за допомогою створених мереж. У роботі генератор використовується для створення великої кількості структур систем та застосування агрегації для згенерованих структур систем.

Існує велика кількість генераторів. Деякі з них застосовуються для генерації випадкових топологій, інші призначені для генерації ієрархічних властивостей Інтернету. Існують наступні вимоги до генераторів:

- репрезентативність – створені топології повинні бути точними, ґрунтуючись на вхідних аргументах, наприклад, таких як ієрархічна структура та характеристики розподілу ступеня;

- гнучкість – за відсутності загальної моделі генератор повинен включати різні методи та моделі;

- можливість розширення – інструмент повинен дозволяти користувачеві розширювати можливості генератора;

- ефективність – інструмент повинен бути ефективним для створення великих топологій, зберігаючи необхідні характеристики, це може дати можливість тестування реальних сценаріїв.

Генератор структур систем побудований на основі прикладів для агрегації структурної моделі [16]. Вхідними даними є кількість елементів в системі, в результаті утворюється таблиця сполучення елементів в системі.

Алгоритм полягає у наступному: на вхід подається число елементів в системі, після формуються вершини і дуги графа [17]. Шляхом обходу дуг графа формуються елементи системи та їх вхідні та вихідні контакти. Після формується таблиця сполучення елементів в системі.

Алгоритм генератора структур систем включає наступні етапи.

Етап №1 – Формування графа

Задається кількість елементів в системі та після цього здійснюється формування вершин графа. Також створюються дві вершини – джерело та стік, які є зовнішнім середовищем системи.

Так як здійснюється робота з графом, який представляє з’єднання елементів в системі, слід визначити деякі обмеження:

- до вершини джерела не може входити дуга;
- з вершини стоку не може виходити дуга;
- для всіх інших вершин повинна бути хоча б одна вхідна дуга та хоча б одна вихідна дуга;
- з джерела не може йти дуга в стік;
- з вершини n не може йти дуга в вершину n.

Далі формуються дуги графа. Для цього здійснюється пошук максимально можливої кількості дуг з урахуванням введених обмежень.

Одиниця у матриці означає, що з вершини і виходить дуга в вершину j, нуль означає, що зв’язку немає. Прочерк означає те, що не може існувати зв’язку між вершинами.

З урахуванням обмежень спочатку маємо матрицю виду:

$$\begin{matrix} - & 0 & 0 & 0 & 0 & - \\ - & - & 0 & 0 & 0 & 0 \\ - & 0 & - & 0 & 0 & 0 \\ - & 0 & 0 & - & 0 & 0 \\ - & 0 & 0 & 0 & - & 0 \\ - & - & - & - & - & - \end{matrix}, \quad (3.24)$$

У кожному рядку випадковим чином вибирається елемент, заноситься одиниця. Таким чином, тепер з кожної вершини виходить хоча б одна дуга.

$$\begin{matrix} - & 1 & 0 & 0 & 0 & - \\ - & - & 0 & 0 & 0 & 1 \\ - & 0 & - & 1 & 0 & 0 \\ - & 1 & 0 & - & 0 & 0 \\ - & 0 & 0 & 0 & - & 1 \\ - & - & - & - & - & - \end{matrix} \quad (3.25)$$

Аналізуються стовпці: якщо в якомусь стовпці немає жодної одиниці – випадковим чином вибирається елемент стовпця, заноситься одиниця. Таким чином, тепер в кожну вершину входить хоча б одна дуга.

$$\begin{matrix} - & 1 & 0 & 0 & 0 & - \\ - & - & 0 & 0 & 1 & 1 \\ - & 0 & - & 1 & 0 & 0 \\ - & 1 & 1 & - & 0 & 0 \\ - & 0 & 0 & 0 & - & 1 \\ - & - & - & - & - & - \end{matrix} \quad (3.26)$$

Якщо реальне число дуг менше, ніж потрібне, випадковим чином вибирається елемент матриці, заноситься одиниця. Таким чином, отримуємо матрицю:

$$\begin{matrix} - & 1 & 0 & 1 & 1 & - \\ - & - & 0 & 0 & 1 & 1 \\ - & 0 & - & 1 & 1 & 0 \\ - & 1 & 1 & - & 0 & 0 \\ - & 1 & 1 & 0 & - & 1 \\ - & - & - & - & - & - \end{matrix} \quad (3.27)$$

Відповідно до матриці отримуємо наступний граф:

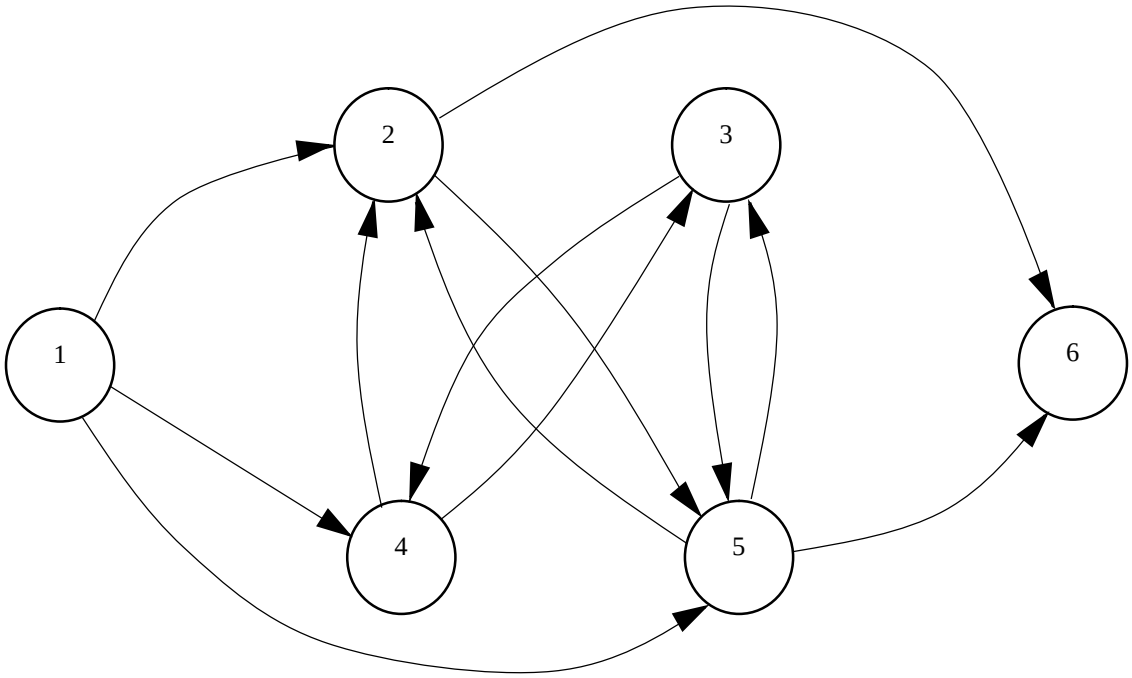

Рисунок 3.5 – Згенерований граф

Тепер потрібно перетворити отриманий граф – змінити номери вершин таким чином, щоб перша вершина була нульовим елементом або джерелом, та щоб остання вершина була також нульовим елементом або стоком. На даному етапі здійснюється заміна номерів вершин графа (рисунок 3.6).

Етап №2 – Формування системи

Після того, як дуги для графа сформовані, потрібно сформувати зв’язки в системі. Алгоритм формування зв’язків системи полягає в наступному. По черзі аналізуються дуги графа.

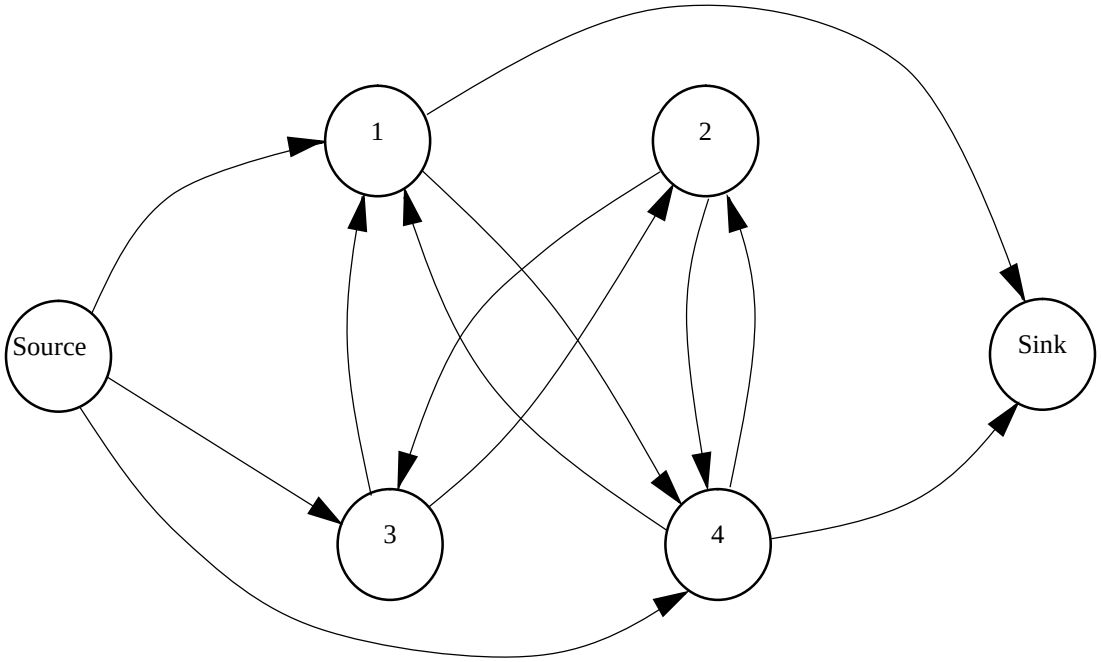

Рисунок 3.6 – Перетворений граф

1) Вершина, з якої виходить дуга.

Якщо не існує елемента в системі з таким же номером, як у вершини графа, то такий елемента створюється. Після цього створюється новий вихідний контакт елемента.

2) Вершина, до якої входить дуга.

Таблиця 3.6 – Сполучення елементів в системі

i\j	1	2	3
0	1,2,54	4,3,23	-,-,-
1	0,1,68	3,1,60	4,1,60
2	3,2,28	4,2,41	-,-,-
3	0,2,45	2,1,23	-,-,-
4	0,2,58	1,1,17	2,1,36

Якщо не існує елемента в системі з таким же номером, як у вершини графа, то такий елемента створюється. Після цього створюється новий

вхідний контакт елемента.

Також для кожного зв’язку генерується пропускна здатність. Після цього шляхом перебору списку зв’язків формується оператор сполучення елементів в системі (таблиця 3.6).

3.3 Застосування агрегації для вирішення задачі пошуку максимального потоку

Задача про максимальний потік відноситься до множини задач топологічного аналізу, мета яких полягає у розподілі мережевих потоків для досягнення максимальних значень ефективності зв’язку. Суть задачі про максимальний потік полягає в отриманні значення максимального потоку мережі для заданої топології [18]. В такому випадку ефективність використання ресурсів мережі зростає. Задача про максимальний потік зустрічається, наприклад, в логістиці, при пошуку максимальної інтенсивності транспортного потоку між пунктами, представленими графом. Вирішення цієї задачі допомагає виявити ділянки мережі, які можуть утворювати «вузькі місця».

Задача про максимальний потік формулюється наступним чином. Задано пропускні здатності дуг мережі. Необхідно знайти максимальне можливе для даної мережі значення сумарного потоку між джерелом та стоком. Якщо потік не досяг максимального значення, то треба визначити, як його збільшити.

Для вирішення задачі про пошук максимального потоку використовується теорема, що буда доведена Фордом та Фалекрсоном [19]. Згідно з цією теоремою максимально можливе значення сумарного потоку на кінцевих дугах дорівнює мінімальній пропускній здатності обраного розрізу. Під пропускною здатністю розрізу мається на увазі сума пропускних здатностей дуг, що утворюють розріз.

Алгоритм Форда-Фалкерсона включає наступні кроки:

- пошук на графі ланцюга з джерела в стік;
- для кожної дуги ставиться максимальне можливе значення потоку з джерела в стік, значення потоку не може бути більшим ніж вага дуги;
- якщо значення потоку дорівнює вазі дуги, то така дуга є насиченою, та через неї вже не можна пройти під час вибору ланцюгів у графі;
- алгоритм завершується тоді, коли перехід з джерела в стік стане неможливим.

Після завершення дії алгоритму значення максимального потоку дорівнює сумі потоків всіх дуг, інцидентних стоку графа.

Розглянемо граф системи (таблиця 3.1).

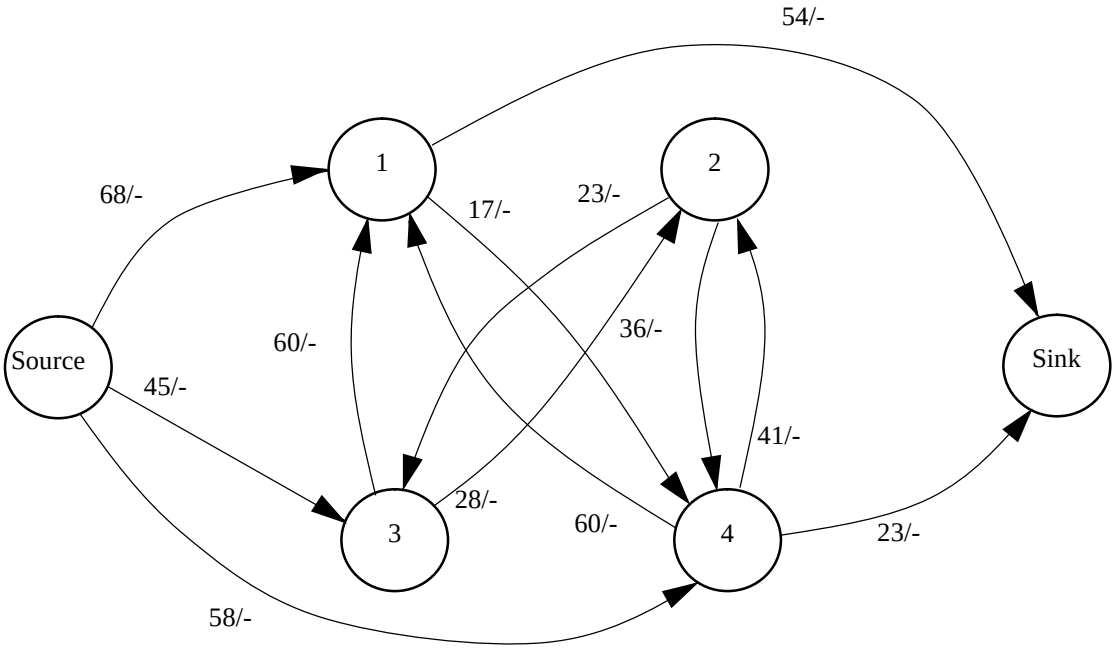

Рисунок 3.7 – Граф системи

Вирішення задачі пошуку максимального потоку для першого рівня системи містить наступні кроки.

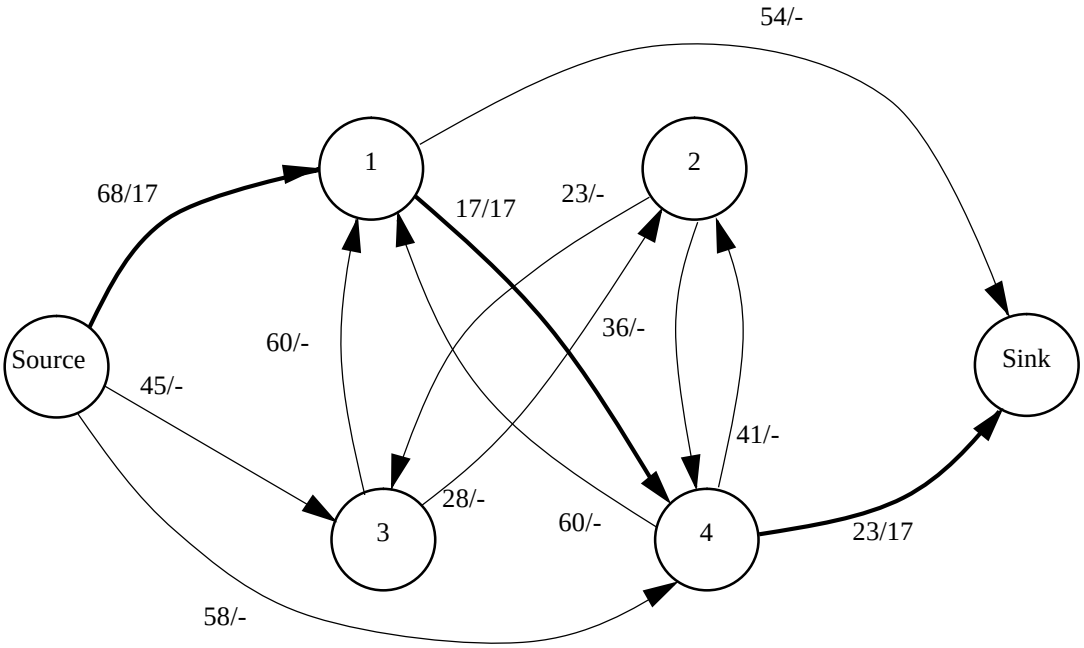

Рисунок 3.8 – Перша ітерація обходу графа

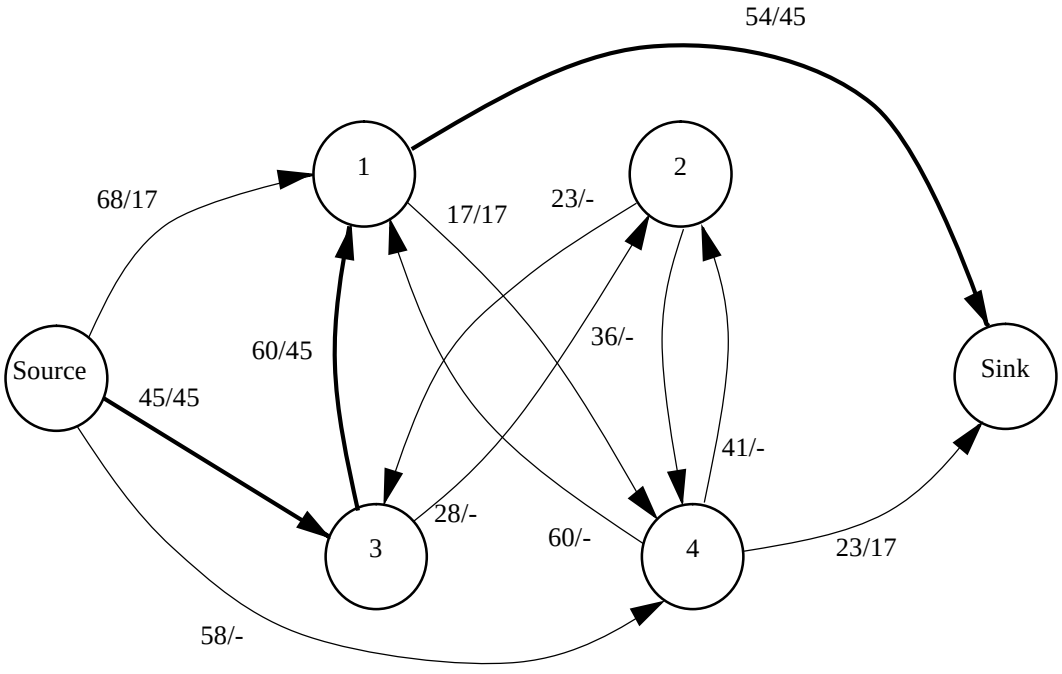

Рисунок 3.9 – Друга ітерація обходу графа

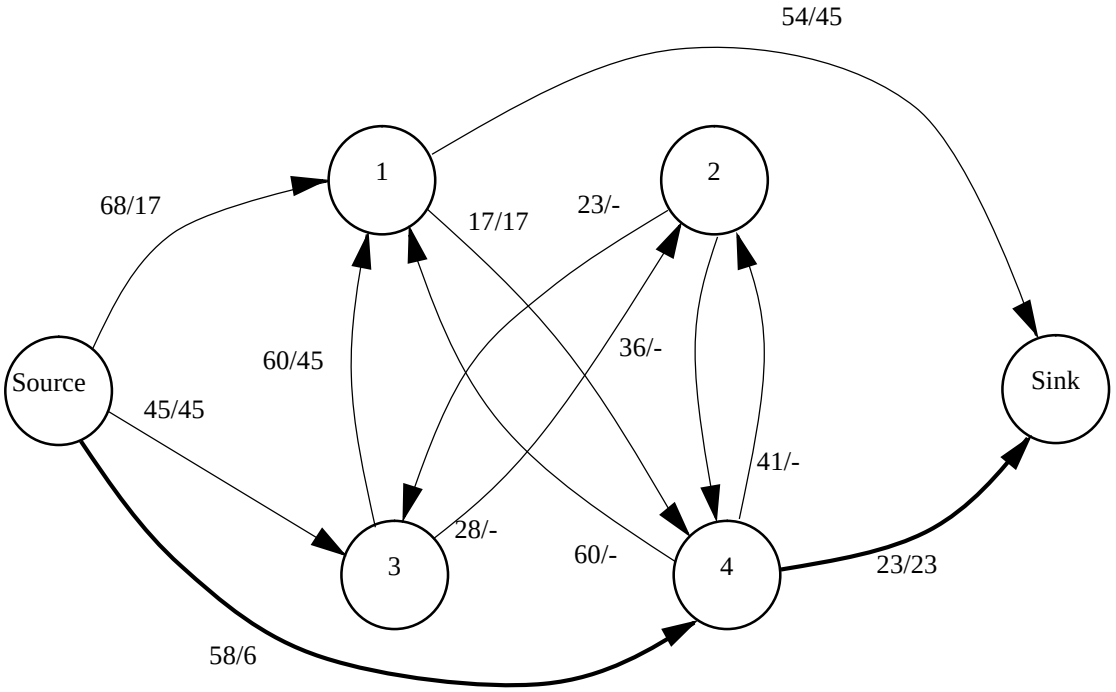

Рисунок 3.10 – Третя ітерація обходу графа

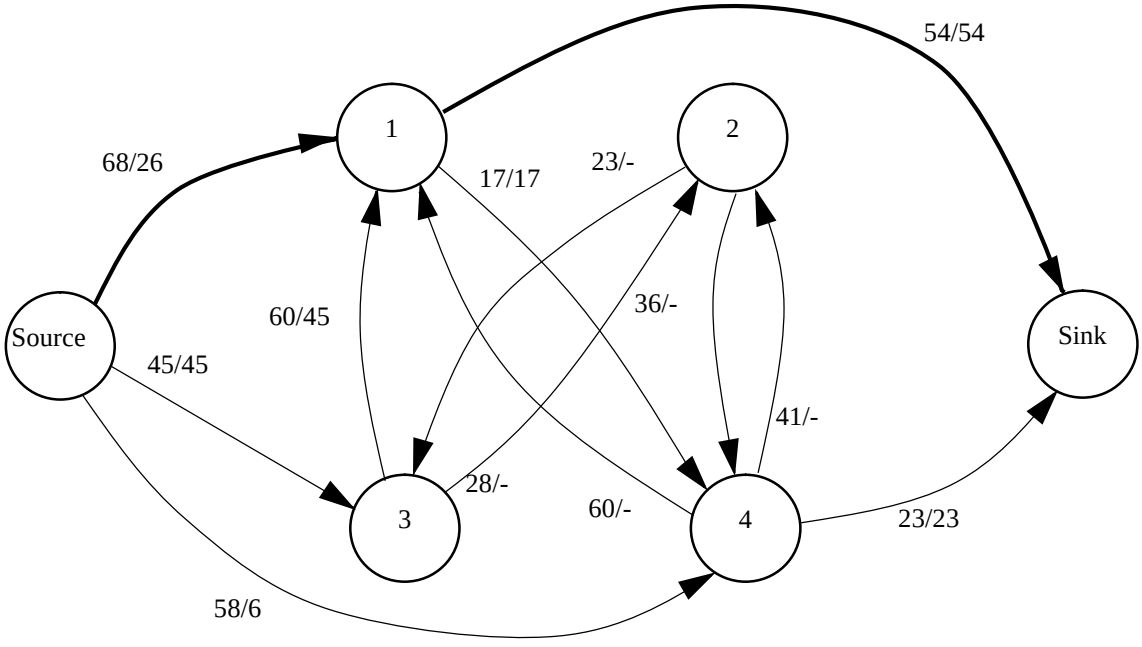

Рисунок 3.11 – Четверта ітерація обходу графа

Таблиця 3.7 – Результат обходу графа

№	Шлях	Значення потоку
1	Source–1–4–Sink	17
2	Source–3–1–Sink	45
3	Source–4–Sink	6
4	Source–1–Sink	9

Максимальний потік мережі визначається за формулою:

$$f = \sum_{i=1}^{n} f_i, \tag{3.28}$$

де n – кількість ребер, інцидентних стоку;

f_i – величина потоку ребра, інцидентного стоку;

За формулою (3.28) був знайдений максимальний потік мережі:

$$f_1 = 17 + 45 + 6 + 9 = 77 \tag{3.29}$$

Після застосування агрегації структурної моделі був отриманий другий рівень системи (таблиця 3.5). Розглянемо граф для другого рівня системи.

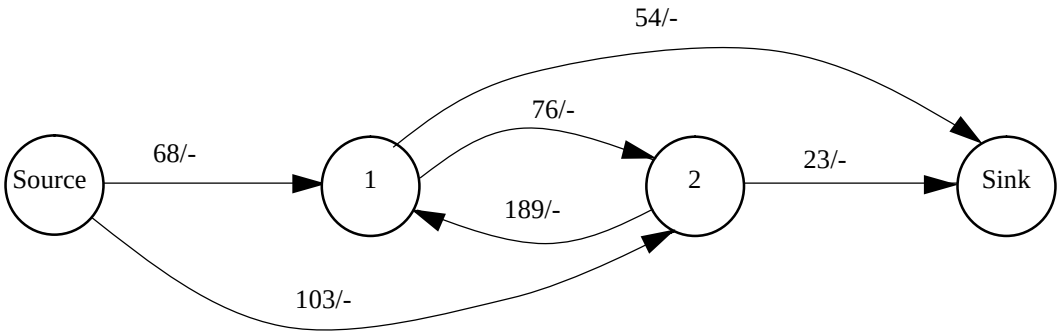

Рисунок 3.12 – Граф другого рівня системи

Вирішення задачі пошуку максимального потоку для другого рівня системи містить наступні кроки.

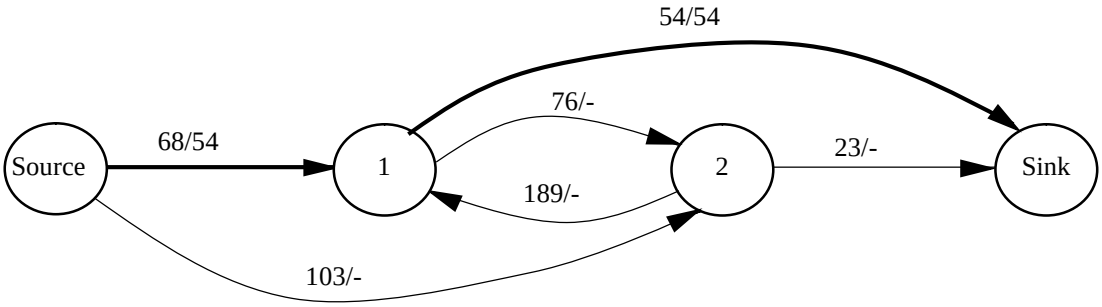

Рисунок 3.13 – Перша ітерація обходу графа

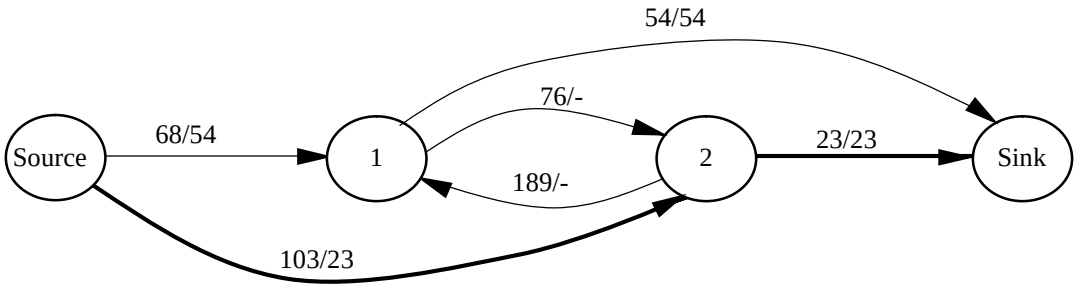

Рисунок 3.14 – Друга ітерація обходу графа

Таблиця 3.8 – Результат обходу графа

№	Шлях	Значення потоку
1	Source–1–Sink	54
2	Source–2–Sink	23

За формулою (3.28) був знайдений максимальний потік мережі:

$$f_2 = 54 + 23 = 77 \tag{3.30}$$

Була вирішена задача пошуку максимального потоку для двох рівнів системи. Для першого рівня було здійснено чотири ітерації, для другого рівня системи було здійснено дві ітерації. На кожній ітерації був знайдений ланцюг з джерела в стік. Після цього для кожної дуги ставилося максимальне можливе значення потоку. Це значення повинно дорівнювати або бути меншим ніж вага дуги. Якщо значення потоку дорівнює вазі дуги, то така дуга стає насиченою. Ітерація завершувалася тоді, коли перехід з джерела в стік був неможливим.

В результаті пошуку максимального потоку на двох рівнях системи можна зробити висновок, що завдяки агрегації структурної моделі зменшується кількість ітерацій, також зменшується довжина ланцюга із джерела до стоку, але значення максимального потоку залишається однаковим для кожного рівня системи.

4 ОЦІНКА ЕФЕКТИВНОСТІ ЗМЕНШЕННЯ РОЗМІРНОСТІ МЕРЕЖЕВИХ СИСТЕМ

4.1 Структура програмної платформи

Програмна платформа використовується для оцінки ефективності різних технологій системного аналізу, а також зменшення розмірності мережевих систем [20]. У роботі програмна платформа використовується для оцінки ефективності агрегації структурної моделі.

Програмна платформа складається з чотирьох модулів: модуль аналізу характеристик реальної системи, модуль генератора мережевих структур, модуль агрегації структурної моделі та модуль пошуку максимального потоку.

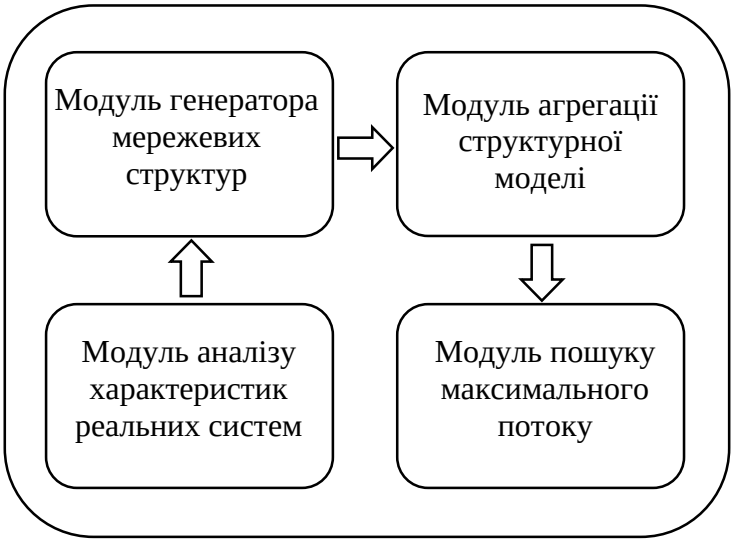

Рисунок 4.1 – Структура програмної платформи

Мета модулю аналізу характеристик реальної системи полягатє в оцінці топологічних параметрів мережі, тому що топологічний аналіз мереж необхідний для планування та управління мережею, розробки алгоритмів

виявлення вузьких місць та відмов, оцінки продуктивності системи. На вхід модуля генератора мережевих структур систем подається кількість елементів системи, модуль генерує структуру системи. В результаті роботи модуля формується таблиця сполучення елементів в системі, яка передається в модуль агрегації структурної моделі. У модулі агрегації структурної моделі елементи системи об’єднуються в підсистеми, і в результаті роботи модуля формується трирівнева система. Таблиці сполучення елементів на трьох рівнях системи передаються в модуль пошуку максимального потоку, в якому вирішується задача пошуку максимального потоку на всіх рівнях системи. Результатом роботи модуля є значення максимального потоку на трьох рівнях системи.

4.2 Реалізація програмної платформи

Генератор структур систем був реалізований на мові програмування Java [21]. Був створений клас Generator, який містить такі поля типу GraphStructure і SystemStructure. Клас GraphStructure представляє граф і містить вершини графа Vertex і дуги Edge. Клас Vertex містить номер вершини та її ступінь, а клас Edge містить номер вершини, до якої входить дуга, і номер вершини, з якої виходить дуга. Також клас містить матрицю matrixOfConnections, в якій здійснюється генерація дуг графа.

Структуру системи представляє клас SystemStructure. Він містить елементи системи Element і зв’язки між елементами Connection. Клас Element містить номер елемента, кількість вхідних і вихідних контактів. Клас Connection містить номера елемента і вхідного контакту, до якого входить зв’язок, номера елемента і вихідного контакту, з якого зв’язок виходить, і пропускну спроможність. На початку роботи алгоритму здійснюється генерація графа. Створюється колекція ArrayList об’єктів Vertex. Після цього здійснюється генерація дуг графа, дані заносяться в матрицю matrixOfConnections. Далі створюється колекція ArrayList об’єктів Edges.

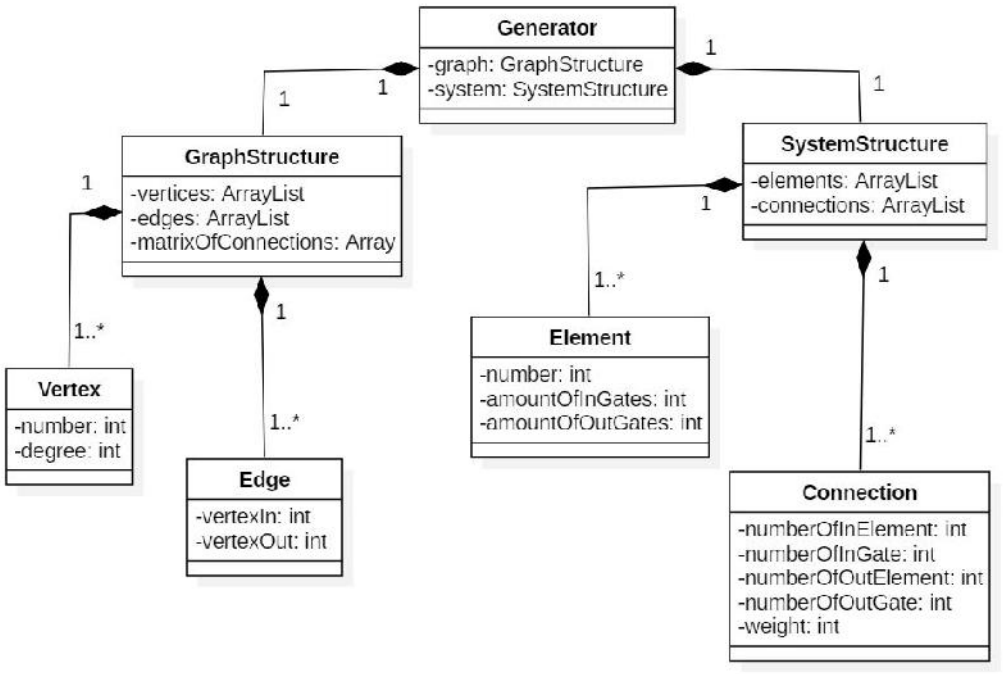

Рисунок 4.2 – UML діаграма генератору структур систем

На наступному етапі створюється структура системи на основі отриманого графа. Здійснюється обхід дуг графа і створюються колекції ArrayList об’єктів Element і Connection. Колекція об’єктів Connection містить усі зв’язки елементів у системі.

На її основі будується таблиця сполучення елементів в системі, яка далі передається в модуль агрегації структурної моделі.

```
i\j          1              2              3
0          1,2,54         4,3,23          -,-,-
1          0,1,68         3,1,60         4,1,60
2          3,2,28         4,2,41          -,-,-
3          0,2,45         2,1,23          -,-,-
4          0,2,58         1,1,17         2,1,36
```

Приклад 4.1 – Результат роботи програми – формування таблиці сполучення елементів системи

Також здійснюється формування json файлу для того, щоб передати згенеровану структуру системи у модуль агрегації структурної моделі.

```
      public  static  void  CreateJSON(ArrayList<Connection>  connectionList,
String filePath)
          {
              JSONArray objectList = new JSONArray();

              for(Connection connection : connectionList)
              {
                  JSONObject currentObj = new JSONObject();

                  JSONObject objInComponent = new JSONObject();

                  objInComponent.put("component",
connection.getNumberOfInElement());
                  objInComponent.put("contact",
connection.getNumberOfInGate());

                  JSONObject objOutComponent = new JSONObject();
                  objOutComponent.put("component",
connection.getNumberOfOutElement());
                  objOutComponent.put("contact",
connection.getNumberOfOutGate());

                  currentObj.put("to", objInComponent);
                  currentObj.put("from", objOutComponent);
                  currentObj.put("weight", connection.getWeight());

                  objectList.add(currentObj);
              }
```

Приклад 4.2 – Формування json файлу

```
      [{"to":{"component":1,"contact":1},"weight":68,"from":{"component":0,"co
ntact":1}},
      {"to":{"component":3,"contact":1},"weight":45,"from":{"component":0,"con
tact":2}},
      {"to":{"component":4,"contact":1},"weight":58,"from":{"component":0,"con
tact":2}},
      {"to":{"component":4,"contact":2},"weight":17,"from":{"component":1,"con
tact":1}},
      {"to":{"component":0,"contact":1},"weight":54,"from":{"component":1,"con
tact":2}},
      {"to":{"component":3,"contact":2},"weight":23,"from":{"component":2,"con
tact":1}},
      {"to":{"component":4,"contact":3},"weight":36,"from":{"component":2,"con
tact":1}},
      {"to":{"component":1,"contact":2},"weight":60,"from":{"component":3,"con
tact":1}},
      {"to":{"component":2,"contact":1},"weight":28,"from":{"component":3,"con
tact":2}},
      {"to":{"component":1,"contact":3},"weight":60,"from":{"component":4,"con
tact":1}},
      {"to":{"component":2,"contact":2},"weight":41,"from":{"component":4,"con
tact":2}},
      {"to":{"component":0,"contact":2},"weight":23,"from":{"component":4,"con
tact":3}}]
```

Приклад 4.3 – Результат роботи програми – формування json файлу

Алгоритм багаторівневої агрегації мережевої системи був реалізований на мові програмування Java. Для практичної реалізації цього завдання необхідно було побудувати класи, які відображають структуру та поведінку системи. Були визначені такі структурні частини: система, що містить компоненти; компонент; контакт компонента; зв’язок між компонентами. На основі цього поділу були побудовані такі класи, як SystemOfComponents, Component, Gate, Connection. Ця система є багаторівневою системою, і для представлення її рівнів був створений клас LevelOfSystem. Для зберігання рівнів системи було створено колекцію в класі SystemOfComponents.

Для створення ієрархічної структури компоненти були об’єднані в підсистеми. Компонент є компонентом системи та незалежною системою одночасно. Для програмної реалізації цього аспекту застосовано механізм наслідування і, як наслідок, клас Component наслідується від класу LevelOfSystem. Для представлення контактів компонента був створений клас Gate. Зв’язок між двома компонентами представляється класом Connection.

На першому етапі треба отримати структуру згенерованої системи. Для цього треба обробити json файл.

```
ArrayList<ConnectionForJson> connectionListFromJson = new ArrayList<>();

for (int i = 0; i < amountOfConnections; i++) {
            JSONObject jsonObject = (JSONObject) jsonArray.toArray()[i];

            JSONObject toObject = (JSONObject) jsonObject.get("to");

            long toComponent = (long) toObject.get("component");
            long toContact = (long) toObject.get("contact");

            long weight = (long) jsonObject.get("weight");

            JSONObject fromObject = (JSONObject) jsonObject.get("from");
            long fromComponent = (long) fromObject.get("component");
            long fromContact = (long) fromObject.get("contact");

            ConnectionForJson connection = new ConnectionForJson((int)
toComponent, (int) toContact,
                    (int) fromComponent, (int) fromContact, (int)
weight);
            connectionListFromJson.add(connection);
        }
```

Приклад 4.4 – Обробка json файлу

Після того, як була отримана структура системи, здійснюється формування другого рівня системи. Для цього елементи першого рівня об’єднуються у підсистеми. Також формуються вхідні та вихідні фіктивні контакти підсистем.

```
      public void CreateSubSystem() {
              systemLevelTwo.addComponent(0, systemComponents, 0);

              int                      amountOfComponents                      =
systemComponents.getAmountOfComponents() - 1;

              if (amountOfComponents % 2 == 0) {
                  for (int i = 1; i <= amountOfComponents / 2; i++) {
                      systemLevelTwo.addComponent(i, systemComponents, i * 2 -
1, i * 2);
                  }
              } else {
                  for (int i = 1; i <= amountOfComponents / 2 - 1; i++) {
                      systemLevelTwo.addComponent(i, systemComponents, i * 2 -
1, i * 2);
                  }
                  int numberOfLastComponent = amountOfComponents / 2;
                  systemLevelTwo.addComponent(numberOfLastComponent,
systemComponents,
                          numberOfLastComponent * 2 - 1,
                          numberOfLastComponent * 2,
                          numberOfLastComponent * 2 + 1);
              }
          }
```

Приклад 4.5 – Формування підсистем

```
Components of subsystem #0
Number of component: 0
Gate In:
1 2
Gate Out:
1 2
Components of subsystem #1
Number of component: 1
Gate In:
1 2 3
Gate Out:
1 2
Number of component: 2
Gate In:
1 2
Gate Out:
1
```

Приклад 4.6 – Результат роботи програми – об’єднання елементів у підсистеми №0 та №1

```
Components of subsystem #2

Number of component: 3

Gate In:
1 2
Gate Out:
1 2

Number of component: 4

Gate In:
1 2 3
Gate Out:
1 2 3
```

Приклад 4.7 – Результат роботи програми – об’єднання елементів у підсистему №2

```
Fictitious contacts of subsystems

Number of component: 0

Gate In:
0,1   0,2
1     2

Gate Out:
0,1   0,2
1     2

Number of component: 1

Gate In:
1,1   1,2   1,3   2,1   2,2
1     2     3     4     5

Gate Out:
1,1   1,2   2,1
1     2     3

Number of component: 2

Gate In:
3,1   4,1   3,2   4,3   4,2
1     1     2     2     3

Gate Out:
3,1   3,2   4,1   4,2   4,3
1     2     3     4     5
```

Приклад 4.8 – Результат роботи програми – формування фіктивних контактів підсистем

На наступному етапі формуються зв’язки між елементами в підсистемах.

```
Connections inside of subsystems

For component 0

Table of connections
#     1     2
0     0μ,1  0μ,2
0μ    0,1   0,2

For component 1

Table of connections
#     1     2     3
0     1,1   1,2   2,1
1     0,1   0,2   0,3
2     0,4   0,5   -,-


For component 2

Table of connections
#     1     2     3     4     5
0     3,1   3,2   4,1   4,2   4,3
3     0,1   0,2   -,-   -,-   -,-
4     0,1   0,3   0,2   -,-   -,-
```

Приклад 4.9 – Результат роботи програми – формування зв’язків між елементами в підсистемах

На наступному етапі формуються зв’язки між підсистемами.

```
Table of subsystem connections
#     1           2           3           4           5
0     1,2,54      2,5,23      -,-,-       -,-,-       -,-,-
1     0,1,68      2,1,60      2,3,60      2,2,28      2,4,41
2     0,2,103     1,3,59      1,1,17      -,-,-       -,-,-
```

Приклад 4.10 – Результат роботи програми – формування зв’язків між підсистемами

Після цього формується json файл, що містить зв’язки між підсистемами – схему сполучення елементів на другому рівні системи, який потім передається в модуль пошуку максимального потоку.

```
      public      static      void      CreateJSON(ArrayList<ConnectionForJson>
connectionList, String filePath)
          {
              JSONArray objectList = new JSONArray();

              for(ConnectionForJson connection : connectionList)
              {
                  JSONObject currentObj = new JSONObject();
                  JSONObject objInComponent = new JSONObject();
                  objInComponent.put("component",
connection.getNumberOfInElement());
                  objInComponent.put("contact",
connection.getNumberOfInGate());
                  JSONObject objOutComponent = new JSONObject();
                  objOutComponent.put("component",
connection.getNumberOfOutElement());
                  objOutComponent.put("contact",
connection.getNumberOfOutGate());
                  currentObj.put("to", objInComponent);
                  currentObj.put("from", objOutComponent);
                  currentObj.put("weight", connection.getWeight());
                  objectList.add(currentObj);
              }

              try (FileWriter file = new FileWriter(filePath))
              {
                  file.write(objectList.toJSONString());
                  file.close();
              }
              catch (IOException e) {
                  e.printStackTrace();
              }
          }
```

Приклад 4.11 – Формування json файлу

```
      [{"to":{"component":0,"contact":1},"weight":54,"from":{"component":1,"co
ntact":2}},
      {"to":{"component":0,"contact":2},"weight":23,"from":{"component":2,"con
tact":5}},
      {"to":{"component":1,"contact":1},"weight":68,"from":{"component":0,"con
tact":1}},
      {"to":{"component":1,"contact":2},"weight":60,"from":{"component":2,"con
tact":1}},
      {"to":{"component":1,"contact":3},"weight":60,"from":{"component":2,"con
tact":3}},
      {"to":{"component":1,"contact":4},"weight":28,"from":{"component":2,"con
tact":2}},
      {"to":{"component":1,"contact":5},"weight":41,"from":{"component":2,"con
tact":4}},
      {"to":{"component":2,"contact":1},"weight":103,"from":{"component":0,"co
ntact":2}},
      {"to":{"component":2,"contact":2},"weight":59,"from":{"component":1,"con
tact":3}},
      {"to":{"component":2,"contact":3},"weight":17,"from":{"component":1,"con
tact":1}}]
```

Приклад 4.12 – Результат роботи програми – формування json файлу

4.3 Програмний експеримент оцінки ефективності зменшення розмірності мережевих систем

Основною метою експерименту є оцінка ефективності мережевих топологічних формальних перетворень, а саме агрегації структурної моделі. Платформа включає набір модулів: модуль аналізу характеристик реальної системи, модуль генератора мережевих структур, модуль агрегації структурної моделі та модуль пошуку максимального потоку. Результати роботи були апробовані в тезах конференцій та статті (додаток Б).

Процедура експерименту полягає в наступному. Модуль генератора мережевих структур генерує мережеві системи. Ці системи мають топологічні характеристики, аналогічні характеристикам мережевої системи, кількість яких дорівнює 12. Експеримент полягав в генерації структур системи для систем різної розмірності. Були згенеровані структури систем з 12, 100, 250, 400 та 500 елементами (таблиця 4.1).

Таблиця 4.1 – Час роботи генератора мережевих структур

Кількість елементів	12	100	250	400	500
Час виконання, с	0.000005	0.002187	0,089079	0.654550	1.260938

Таблиця 4.2 – Загальний час роботи програмної платформи

Кількість елементів	12	100	250	400	500
Час виконання, хв	0.000099	0.003978	0.237939	4.092443	9.953349

На наступному етапі модуль агрегації структурної моделі формує трирівневу систему. На останньому кроці у модулі вирішується проблема

пошуку максимального потоку. Значення максимального потоку є однаковим для всіх трьох рівнів системи, що свідчить про те, що під час агрегації структурної моделі зберігається коректність параметрів системи.

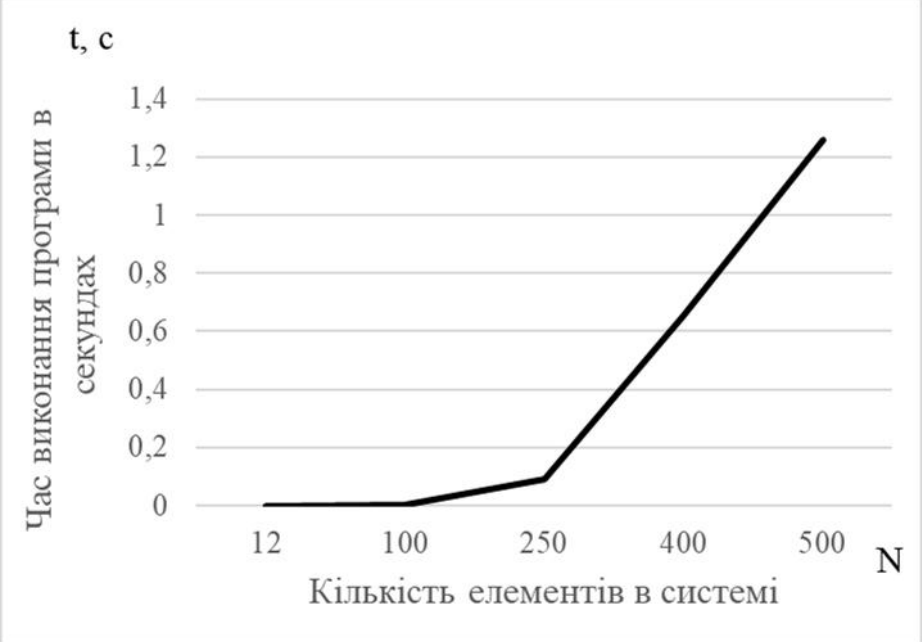

Рисунок 4.3 – Час роботи модуля генератора мережевих структур

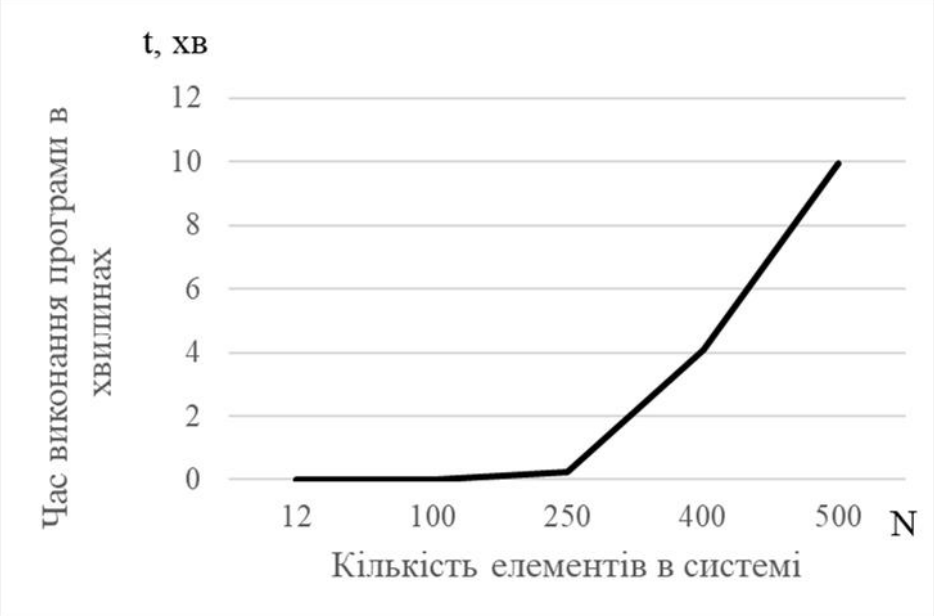

Рисунок 4.4 – Час роботи програмної платформи

Аналіз результатів експериментів полягає у наступному. Час виконання модулів генератора мережевих структур (рисунок 4.3) та пошуку максимального потоку значно менший, ніж загальний час вирішення задачі (рисунок 4.4). Цей час виконання майже дорівнює часу виконання модуля агрегації структурної моделі.

В результаті аналізу графіків часу виконання модулів можна зробити висновок, що зі збільшенням кількості елементів та зв’язків між ними час виконання програми збільшується.

Для створення нового елемента алгоритм генератора структур систем обходить колекцію, що містить дуги графа, та перевіряє, чи існує вже такий елемент. Для цього здійснюється аналіз усіх елементів системи. Якщо кількість елементів системи збільшується, то час роботи генератора також збільшується.

Алгоритм еквівалентних топологічних перетворень полягає у об’єднанні елементів попереднього рівня в підсистеми та формуванні зв’язків всередині цих підсистем та між підсистемами.

Збільшення кількості елементів в системі призводить до значного збільшення розмірності таблиць сполучення елементів, які представляють структуру мережевої системи. Також збільшення кількості елементів в системі призводить до збільшення часу виконання алгоритму на етапі формування фіктивних контактів підсистем. Час формування зв’язків між підсистемами також збільшується через аналіз дуже великої кількості контактів підсистем.

Також була розглянута проблема втрати інформації на прикладі результатів пошуку максимального потоку на першому (таблиця 3.1) та другому (таблиця 3.5) рівнях системи.

Приймемо значення максимального потоку f, що визначається по формулі (3.28) як значення інформації (2.13). В результаті пошуку максимального потоку для першого (3.29) та другого (3.30) рівня системи, отримуємо наступний вираз.

$$f_1 = f_2 \tag{4.1}$$

Тому отримуємо:

$$M(N) = M(\overline{N}), \tag{4.2}$$

де N – початкова мережа;

$\overline{N}$ – агрегована мережа.

По формулі (2.14) знайдемо вартість агрегації J:

$$J = M(N) - M(\overline{N}) = 0 \tag{4.3}$$

Можна зробити висновок, що під час вирішення даної проблеми, в результаті агрегації системи немає втрати інформації.

ВИСНОВКИ

У роботі був проведений аналіз літератури стосовно складних систем, засобів представлення систем, моделей та характеристик мереж. Були розглянуті моделі структур складних систем та техніки моделювання складних систем. У роботі були проаналізовані методи зменшення розмірності мережевих систем та була застосована агрегація структурної моделі для зменшення розмірності мережевих систем.

У роботі був представлений формальний опис агрегації структурної моделі на прикладі створення ієрархічної системи. В результаті багаторівневої агрегації зменшується розмірність системи, обчислювальна складність та час вирішення проблеми. Також були розглянуті генератори структур систем та було розглянуто застосування агрегації структурної моделі для вирішення проблеми пошуку максимального потоку на прикладі ієрархічної системи.

У роботі була розроблена програмна платформа, що складається з таких модулів: модуль аналізу характеристик реальної системи, модуль генератора мережевих структур, модуль агрегації структурної моделі, модуль пошуку максимального потоку. У модулі генератора мережевих структур створюються структури систем для застосування агрегації структурної моделі на великій кількості систем. Згенеровані структури систем передаються до модуля агрегації структурної моделі. У цьому модулі елементи системи об’єднуються у підсистеми та створюється ієрархічна система. Модуль пошуку максимального потоку використовується для оцінки ефективності агрегації структурної моделі. У цьому модулі знаходиться значення максимального потоку для усіх рівнів системи.

У ході оцінки ефективності агрегації структурної моделі був розглянутий аспект втрати інформації. Значення максимального потоку однакове на всіх рівнях системи, тому в результаті агрегації структурної

моделі нема втрати інформації. Можна зробити висновок, що застосовуючи агрегацію структурної моделі, можна здійснити зменшення розмірності мережевої системи зі збереженням коректності її параметрів.

В результаті роботи були виконані такі завдання, як агрегація структурної моделі системи, розробка програмної платформи та оцінка ефективності зменшення розмірності мережевих систем.

Подальша робота полягає в застосуванні агрегації структурної моделі для зменшення розмірності систем з дуже великою кількістю елементів та зв’язків між ними, в наслідок чого може зменшитися їх розмірність, обчислювальна складність і час вирішення задач. Збільшення розмірів мережевих систем призводить до суттєвого збільшення загальної тривалості обчислень. Основна причина полягає у матричних формах, які представляють структуру мережевої системи на всіх етапах вирішення проблеми. Це важливий висновок у розумінні напрямку майбутніх досліджень. Тому майбутні дослідження повинні бути спрямовані на розробку ефективного структурного опису мережевих систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. J. Lu, G. Chen, M. Ogorzalek and L. Trajkovic, “Theory and Applications of Complex Networks: Advances and Challenges”, Proceedings of the IEEE International Symposium on Circuits and Systems, 2013, pp. 2291–2294.

2. P. Landi, O. H. Minoarivelo, Å. Brännström, C. Hui and U. Dieckmann, “Complexity and stability of ecological networks: a review of the theory”, Population Ecology, 2018, pp. 319–345.

3. E. D. Kolaczyk, Statistical Analysis of Network Data: Methods and Models, Springer, New York, 2009.

4. J. Hackl and B. T. Adey, Generation of Spatially Embedded Random Networks to Model Complex Transportation Networks, 14th International Probabilistic Workshop, 2017.

5. R. S. MacKay, “Hierarchical aggregation of complex systems”, Proceedings of the ECCS’11, Vienna, Austria, 2011.

6. R. S. MacKay and J. D. Robinson, “Aggregation of Markov fows I: Theory”, Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences, 2018.

7. N. Chiriac, K. Hölttä-Otto, D. Lysy and E. S. Suh, “Three approaches to complex system decomposition”, Proceedings of the 13th International DSM Conference, 2011.

8. J. F. Shortle, B. L. Mark and D. Gross, “Reduction of closed queueing networks for efficient simulation”, ACM Transactions on Modeling and Computer Simulation, vol. 19, no. 3, Article 10, Jun. 2009.

9. Mark Newman, Networks: An Introduction, Oxford University Press, 2010.

10. A. Barabási and E. Bonabeau, “Scale-Free Networks”, Scientific American, vol. 288, no. 5, 2003, pp. 60-69.

11. A. T. Dumitrescu, V. Oltean, D. Merezeanu and R. Dobrescu,

“Emergence in Hierarchical Complex Systems Structured as Social Networks”, 21st International Conference on Control Systems and Computer Science, pp. 426-431, 2017.

12. C. Lecerf, T. Minh and L. Nguyen, Complex Systems Modeling, 2003.

13. H. Hirata and R. E. Ulanovich, “Information Theoretical Analysis of the Aggregation and Hierarchical Structure of Ecological Networks”, University of Maryland, 1985, pp. 321-341.

14. M. Mesarović, D. Mako and Y. Takahara, Theory of Hierarchical Multilevel Systems, New York: Academic, 1977.

15. V. Gorbachov, D. Sytnikov, O. Ryabov, A. K. Batiaa and O. Ponomarenko, “Dimension Reduction for Network Systems Using Structure Model Aggregation”, International Journal of Design & Nature and Ecodynamics, vol. 15, no. 1, 2020, pp. 13–23.

16. V. Gorbachov, A. K. Batiaa, O. Ponomarenko and Y. Romanenkov, “Formal transformations of stuctural models of complex network systems”, Proceedings of the IEEE 9th International Conference on Dependable Systems, Services and Technologies DESSERT′2018, Kyiv, Ukraine, 2018, pp. 473–477.

17. Хаггарти, Р. Дискретная математика для программистов. – М : Техносфера, 2003. – 320 с.

18. T. Cormen, C. Leiserson, R. Rivest and C. Stein, Introduction to algorithms, 3rd ed. 2009.

19. L. Ford and D. Fulkerson, “Maximal flow through a network”, 1956.

20. O. Ponomarenko, V. Gorbachov, A. K. Batiaa and O. Kotkova, “The Software Platform for Evaluation of Effectiveness of Network Systems Analysis Technologies”, in 2019 IEEE East-West Design & Test Symposium (EWDTS), Conference proceedings, Batumi, Georgia, 2019, pp. 513–516.

21. Шилдт, Г. Java 8. Полное руководство, 9-е изд.: Пер. с англ. – М. : ООО «И.Д.Вильямс», 2015. – 1376 с. : ил. – Парал. тит. англ.