НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

До захисту допущено:

Завідувач кафедри

_____ Лариса ГЛОБА

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Застосування нечітких логічних правил для аналізу та структуризації великих даних»

Виконав:
студент IV курсу, групи ТІ- 62
Савчук Захар Романович _____

Керівник:
асистент кафедри ІТМ ІТС
Бугаєнко Юрій Михайлович _____

Консультант :
Професор кафедри ІТМ ІТС, д.т.н., професор
Глоба Лариса Сергіївна _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім’я, по батькові _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

ЗАВДАННЯ
на дипломну роботу студенту
Савчуку Захару Романовичу

1. Тема роботи «Застосування нечітких логічних правил для аналізу та структуризації великих даних», керівник роботи асистент кафедри інформаційно-телекомунікаційних мереж Бугаєнко Юрій Михайлович, затверджені наказом по університету від «30» березня 2020 р. № 924-с

2. Термін подання студентом роботи 8 червня 2020 р.

3. Вихідні дані до роботи

Теорія обробки інформації за допомогою нечіткої логіки, теорія методів візуалізії великих даних

4. Зміст роботи

1. Проаналізувати методи побудови баз знань.
2. Проаналізувати можливість подання баз знань, використовуючи теорію нечіткої логіки для подальшої обробки даних у форматі правил у форматі Якщо… То…

3. Проаналізувати методи візуалізації бази знань за допомогою графів з перевіркою їх на коректність.
4. Модифікувати алгоритм узгодження та перевірки на коректність правил бази нечітких знань та її візуалізація за допомогою апарату теорії метаграфа
5. Провести порівняльну характеристику розробленого методу з існуючими за такими параметрами, як швидкість обробки даних та швидкість візуалізації метаграфу.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Тема, актуальність, мета, задачі.
2. Використання нечітких логічних правил для побудови бази знань.
3. Візуалізація метаграфу, як спосіб представлення бази нечітких знань.
4. Модифікований метод очищення бази нечітких правил та подання її за допомогою апарату теорії метаграфу.
5. Порівняння запропонованого методу з існуючими методами обробки статистичних даних.
6. Загальні висновки.

6. Дата видачі завдання 15 вересня 2019

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Аналіз методів побудови баз знань.	15.09.2019-06.10.2019	Виконано
2	Огляд існуючих методів щодо подання бази нечітких знань за допомогою нечіткої логіки.	07.10.2019-27.10.2019	Виконано

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
3	Аналіз методів зменшення кількості нечітких логічних правил бази знань	28.09.2019-17.11.2019	Виконано
4	Підготовка статті (XIV Міжнародна науково-технічна конференція "Перспективи телекомунікацій 2020")	02.03.2020-29.03.2020	Виконано
5	Створення модифікованого алгоритму узгодження та перевірки на коректність правил бази нечітких знань	17.11.2019-19.12.2019	Виконано
6	Подання нечітких знань за допомогою апарату теорії метаграфу	20.01.2020-01.03.2020	Виконано
7	Підготовка концепції на конкурс інноваційних ідей DigIn-Net	13.04.2020-04.05.2020	Виконано
8	Отримання і аналіз результатів роботи запропонованого методу структуризації великих даних.	27.04.2020-24.05.2020	Виконано
9	Підготовка тексту диплому	04.05.2020-07.06.2020	Виконано

Студент Захар САВЧУК

Керівник Юрій БУГАЄНКО

РЕФЕРАТ

Робота містить 78 сторінок, 37 рисунків, 6 таблиць. Було використано 38 джерел.

Мета роботи: підвищити ефективність роботи телекомунікаційної системи за рахунок побудови бази нечіких знань з статистичних наборів даних у вигляді логічних структурованих правил, а також проводити їх перевірку на достовірність за допомогою апарату теорії метаграфу.

Завдання даного підходу полягає в структуризації бази нечітких знань у форматі логічних правил, враховуючи перетворення числових значень у терми лінгвістичних змінних, а також при формуванні правил запобігти наявності конфліктуючих і дублюючих правил, краще зрозуміти зміст їх логічних зв'язків. Ці правила можуть покрити увесь простір можливих станів телекомунікаційної системи, а також сервісів, які вона надає, і разом з тим не мати високої обчислювальної складності під час застування у вузлах обчислювальних мереж.

Ключові слова: статистичні дані, нечітка логіка, база нечітких знань, нечіткі логічні правила, апарат теорії метаграфу.

ABSTRACT

The work consists of 78 pages, 6 tables and 37 figures. Used 38 references.

Goal of diploma: to increase the efficiency of the telecommunications system by building a database of fuzzy knowledge from statistical data sets in the form of logical structured rules, as well as to check their validity using the metagraph theory apparatus.

The main purpose of this method is to structure the fuzzy knowledge base in the format of logical rules, taking into account the transformation of numerical values into terms of linguistic variables, as well as to understand the content of their logical connections when formulating rules to prevent the presence of conflicting and duplicate rules. These rules can cover the entire range of possible states of a telecommunication systems and, at the same time, not have high computational complexity when stalled at the nodes of the computing networks.

Key words: statistical data, fuzzy logic, fuzzy knowledge base, fuzzy logic rules, metagraph theory apparatus.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ACE	Algebraic Multigrid Method
AKO	A-Kind-Of
AUC	Area Under Curve
FMS	Fast Multi-scale Method
GVA	Grid-Variant Algorithm
HDE	High-Dimensional Embedding
MAML	Model-Agnostic Meta-Learning
БЗ	База знань
БНЗ	База нечітких знань
ЕОМ	Електронно-обчислювальна машина
ЕС	Експертна система
НЛП	Нечіткі логічні правила
ПО	Предметна область

ВСТУП

Актуальність: в сучасних телекомунікаційних системах широко застосовують різноманітні програмні комплекси, призначені для збору, збереження та обробки значної кількості інформації.

Обробка інформації в таких системах виконується із залученням досвіду фахівців, які базуючись на власному досвіді та за допомогою спрощених математичних методів проводять оцінку загального стану системи, якості та оперативності надання телекомунікаційних послуг кінцевому користувачу. Для оцінювання таких систем часто використовують математичні методи нечітких множин, які пропонують нам гнучкість для міркувань, де можна розглянути усі невизначеності.

Системи нечіткого виводу призначені для перетворення значень вхідних змінних процесу керування у вихідні змінні на основі використання нечітких правил продукцій. Для цього системи нечіткого виводу повинні містити базу правил нечітких продукцій і реалізувати нечіткий висновок на основі умов, представлених у формі нечітких лінгвістичних висловлювань.

Об’єкт роботи: процеси, які відбуваються в телекомунікаційиних системах під час надання сервісів та послуг кінцевим користувачам.

Предмет роботи: математичні методи та моделі побудови нечітких лоігчних правил, теорії алгебри, логіки, методи підтримки прийняття рішень і методи нечіткої логіки, теорія метаграфів.

Мета роботи: підвищити ефективність роботи телекомунікаційної системи за рахунок побудови бази нечіких знань з статистичних наборів даних у вигляді логічних структурованих правил, а також проводити їх перевірку на достовірність за допомогою апарату теорії метаграфу.

Для досягнення мети дослідження було поставлено та вирішено такі **основні задачі:**

1. Проаналізувати методи побудови баз знань.

2. Проаналізувати можливість подання баз знань, використовуючи теорію нечіткої логіки для подальшої обробки даних у форматі правил у форматі Якщо… То…

3. Проаналізувати методи візуалізації бази знань за допомогою графів з перевіркою їх на коректність.

4. Модифікувати алгоритм узгодження та перевірки на коректність правил бази нечітких знань та її візуалізація за допомогою апарату теорії метаграфа

5. Провести порівняльну характеристику розробленого методу з існуючими за такими параметрами, як швидкість обробки даних та швидкість візуалізації метаграфу.

Теоретичний результат дослідження:

1. Аналіз методів подання бази нечітких знань із статистичних наборів даних у вигляді логічних структурованих правил.

2. Модифікований алгоритм узгодження та перевірки на коректність правил бази нечітких знань та її візуалізація за допомогою апарату теорії метаграфа.

Практичний результат роботи:

На основі представленого методу реалізовано програмне забезпечення, яке проводить очищення правил та візуалізовує базу нечітких знань за допомогою метаграфу.

РОЗДІЛ 1.

АНАЛІЗ МЕТОДІВ ПОБУДОВИ БАЗИ ЗНАНЬ НА ОСНОВІ СТАТИСТИЧНОЇ ІНФОРМАЦІЇ В ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ

1.1 Огляд методів побудови бази знань

Принципова відмінність систем штучного інтелекту полягає в тому, що для такого роду систем програміст не готує конкретні програми для виконання. Людина лише дає машині потрібне завдання, а програму, що виконує це завдання, система повинна побудувати сама. Для цього потрібні знання як про предметну області, до якої відноситься завдання, так і про те, як будуються програми. Всі ці знання зберігаються в інтелектуальних системах в спеціальному блоці, званому *базою знань*. База знань основа будь-якої інтелектуальної системи.

Знання, що зберігаються в базі знань, записуються в спеціальній формалізованій формі. У базі знань можуть реалізуватися процедури узагальнення та коригування збережених знань, а також процедури, що створюють нові знання на підставі тих, які вже там є. Знання, збережені в базі, відрізняються від даних, що зберігаються в базі даних, декількома особливостями. По-перше, структура знань набагато складніше структури даних. По-друге, знання мають властивість внутрішньої активності. Щоб пояснити цю принципову властивість знань, згадаємо, що при звичайній роботі з комп'ютером дані завжди грають пасивну роль.

База знань – це сукупність програмних засобів, що забезпечують пошук, зберігання, перетворення і запис в пам'яті ЕОМ складно структурованих інформаційних одиниць – *знань*.[1]

Замкнута база знань – це база знань, вміст якої в процесі функціонування не змінюється. Логічний висновок в такій базі еквівалентний висновку в формальній системі і має властивість монотонності, тобто раніше

виведені твердження залишаються вірними на весь період функціонування бази знань.

Відкрита база знань – це база знань, що дозволяє в процесі її функціонування поповнювати вміст бази і прибирати знання з бази. Властивість відкритості призводить до того, що висновок в такій базі є немонотонним, тобто істинність виведених в ній тверджень може змінюватися в процесі роботи системи з такою базою.

Вимоги до моделей знань

Представлення знань – важлива проблема, тому що вона впливає на властивості і характеристики інформаційно-обчислювальних систем і являє собою формалізацію знань для їх введення в базу знань.

Дії над знаннями здійснюються програмним шляхом, тому знання повинні бути представлені формальними моделями, до яких пред'являються дві основні вимоги:

а) Однорідність – дає можливість спростити механізм керування логічним висновком і знанням;

б) Зрозумілість для експертів і користувачів. Не інакше як ускладнюється процес отримання знань і оцінка їхньої якості.

Основні типи моделей представлення знань

Логічна модель являє собою формальну систему – деяке логічне обчислення предикатів першого порядку відбувається тоді, коли предметна область або завдання описується у вигляді набору аксіом [1].

Всі знання про предметну область описуються у вигляді формул цього обчислення або правил виведення. Опис в вигляді формул дає можливість представити декларативні знання, а правила виведення – процедурні знання.

Предикат в обчисленні має спеціальний знак, який відображає певне відношення між кінцевою множиною сутностей – аргументів. Предикат першого порядку має два стани – істина і неістина.

Розглянемо як приклад знання: "Коли загруженність мережі досягає 60% і пройшло менше 50 хвилин з моменту запуску, тиск не може

перевершувати критичний. Якщо з моменту запуску мережі пройшло більше 50 хвилин, то необхідно відкрити підмережу №3".

Логічна модель представлення цього знання примає вигляд:

$$P(p = 60)\ T\ (t < 50)\ \rightarrow (\ D <\ D_{\text{кр}})$$

$$P(p = 60)\ T\ (t > 50)\ =>\ F(\text{№}3\ )$$

У цьому записі використані наступні позначення:

$P(p = 60)$ – предикат, що стає істинним, коли загруженість досягає 60%;

$T\ (t < 50)$ – предикат, що залишається істинним протягом 50 хвилин з початку процесу;

$T\ (t > 50)$ – предикат, що стає істинним після закінчення 50 хвилин з початку процесу;

$D <\ D_{\text{кр}}$ – твердження про те, що тиск нижче критичного;

$F(\text{№}3\ )$ – команда відкрити підмережу №3.

Перший рядок у записі представляє декларативні знання, а друга – процедурні. Обчислення предикатів 1-го порядку в промислових експертних системах практично не використовується. Ця логічна модель застосовується в основному в дослідницьких системах, так як пред'являє дуже високі вимоги та обмеження до предметної області.

Продукційна модель - це модель, заснована на правилах, в якій знання представлені в вигляді пропозицій типу «якщо (умова), то (дія)» [1].

Під «умовою» (антецедентом) розуміється деяка пропозиція – зразок, за яким здійснюється пошук в базі знань, а під «дією» (консеквентом) – дії, виконувані при успішному результаті пошуку (вони можуть бути проміжними, термінальними або цільовими).

Основу моделі складають системи продукцій. Кожна продукція в найбільш загальному вигляді записується як стандартний вираз наступного виду:

"Ім'я продукції":

Ім'я сфери;

Передумова;

Умова для ядра;

Ядро: «Якщо А, то В»;

Ім'я продукції може виражатися у вигляді номера або слова (словосполучення). Служить для визначення розташування в системі продукцій.

Ім'я сфери вказує ту предметну область, до якої відносяться знання, зафіксовані в даній продукції. В інтелектуальній системі може зберігатися сукупність знань (її називають базою знань), що відносяться до різних областям (наприклад, знання про різні захворювання людини або знання з різних розділів математики). Ясно, що якщо в даний момент вирішується завдання з області фізики твердого тіла або з геометрії трикутника, то треба використовувати знання, що відносяться саме до цієї область. Сфери і виділяють такі підобласті знань.

Передумова визначає необхідні передумови застосування умови для ядра продукції. Передумови може і не бути зовсім.

Умова для ядра визначає ті ситуації, при яких потрібно перевіряти наявність або істинність А в ядрі продукції.

Ядро – основна частина продукції. Ядро має вигляд: ""Якщо А, то В", де А і В можуть мати різні значення. Решта елементів, що утворюють продукцію, носять допоміжний характер.

Основні переваги даного методу:

- Простота створення і розуміння, окремих правил;
- Простота розуміння і модифікування знань;
- Простота програмної реалізації механізму логічного висновок.

Слабкі сторони продукційних систем:

- Неясність взаємовідносин правил;
- Складність оцінки цілісного способу знань;
- Дуже низька ефективність обробки;

- Істотні відмінності від людських систем знань;
- Відсутність гнучкості в логічному висновку.

Дана модель широко застосовується в експертних системах, у тому числі промислового (комерційного) рівня. Є велика кількість програмних засобів, що реалізують продукційний підхід (мова OPS 5, EC-EXSYSProfessional, Kappa, ЕКО та інші).

Семантичне представлення – це модель, в якій структура знань предметної області формалізується у вигляді орієнтованого графа вершини якого – *поняття*, а дуги – *відносини* між ними. В якості *понять* зазвичай виступають абстрактні або конкретні об'єкти, події, властивості, операції.

Термін «семантична» означає «смислова», тому семантика – це наука, що встановлює відносини між символами і об'єктами, які вони позначають, тобто наука, що визначає сенс знаків.

Відносини – це зв'язки різного типу [1]. Існують *відносини* різних типів:

а) логічні (диз'юнкція, кон'юнкція, заперечення, імплікація);
б) теоретико - множинні (частина – ціле, множина – підмножина, клас – елемент класу);
в) функціональні (кількісні, тимчасові, просторові та інші характеристики: об'єкт – властивість, властивість – значення);
г) квантифікаційні (логічні квантори спільності та існування, нелогічні квантори, наприклад, багато, декілька);

Найбільш часто в семантичних мережах використовуються наступні відносини:

- зв'язки типу «частина – ціле» («клас – підклас», «елемент – множина»). Відносини «являється» і «має частину» визначають ієрархічну структуру, в якої властивості «вищих» понять автоматично переносяться на «нижчі» поняття. Це дозволяє уникнути дублювання інформації в мережі;

- функціональні зв'язки (визначаються зазвичай дієсловами «виробляє», «впливає»);
- кількісні (більше, менше, дорівнює);
- просторові (далеко від, близько від, за, під, над);
- тимчасові (раніше, пізніше, протягом);
- атрибутивні зв'язки (мати властивість, мати значення).

Проблема пошуку рішення в базі знань типу семантичної моделі зводиться до задачі пошуку фрагмента мережі, відповідного деякої підмережі, що відображає поставлений запит до бази.

Основною перевагою семантичної моделі є те, що вона порівняно з іншими моделями відповідає сучасним уявленням про організацію довготривалої пам'яті людини. Недоліком цієї моделі є складність організації процедури пошуку виводу в семантичній мережі.

Для реалізації семантичних моделей представлення знань існують спеціальні мережеві мови, наприклад .NET, мова реалізації систем SIMER + MIR та інші широко відомі експертні системи, що використовують семантичні мережі як мову представлення знань – PROSPECTOR, CASNET, TORUS.

Фреймова модель представлення знань

Фрейм – це структура для представлення знань, яка при заповненні відповідними значеннями перетворюється в опис конкретного факту, процесу, події або ситуації.

Фрейм має майже однорідну структуру і складається із стандартних одиниць, так званих слотами. Кожна така одиниця містить назву і своє значення.

Фрейм можна представити:

а) у вигляді ланцюга:

Фрейм: <слот 1>, <слот 2>, <…>, <слот N>.

Або більш детальніше, як список властивостей:

Назва фрейму: <назва 1-го слоту: значення 1-го слоту>, <назва 2-го слоту: значення 2-го слоту>, <...>, <назва N-го слоту: значення N-го слоту>;

б) у вигляді таблиці (див. рисунок 1.1):

Назва фрейму	
Назва слоту 1	Значення слоту 1
Назва слоту 2	Значення слоту 2
...	...
Назва слоту N	Значення слоту N

Рис. 1.1 Представлення фрейму у вигляді таблиці

Модель фрейма є досить універсальною, оскільки до складу фрейму можуть входити слоти з іменами дій, тому фрейми можуть також використовуватись для представлення як декларативних, так і процедурних знань. Фрейм є простим, якщо він не містить в собі інших фреймів. Складний (складений) фрейм містить в собі два і більше фрейма, і по суті представляє мережу фреймів.

Фрейм дозволяє відобразити все різноманіття знань через:

- фрейми-структури, що використовуються для позначення об'єктів і понять;
- фрейми-ролі (менеджер, касир, клієнт);
- фрейми-сценарії (банкрутство, збори акціонерів);
- фрейми-ситуації (тривога, аварія, робочий режим).

Фреймова модель представлення знань – це модель, в якої структура знань предметної області формалізується у вигляді сукупності взаємопов'язаних фреймів, що описують об'єкти, а властивості цих об'єктів і факти, що відносяться до них, описуються в структурних елементах фрейму.

В якості значення слоту може виступати новий фрейм, що дозволяє на безлічі фреймів здійснювати ієрархічну класифікацію. Це дуже зручна властивість фреймів, так як людські знання, як правило, впорядковані.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж – успадкування властивостей. У фреймах спадкування відбувається по АКО-зв'язкам (A-Kind-Of), які пов'язують фрейми з фреймами, що знаходяться на рівень вище в ієрархії, звідки неявно успадковуються (переносяться) значення слотів. Слот АКО вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

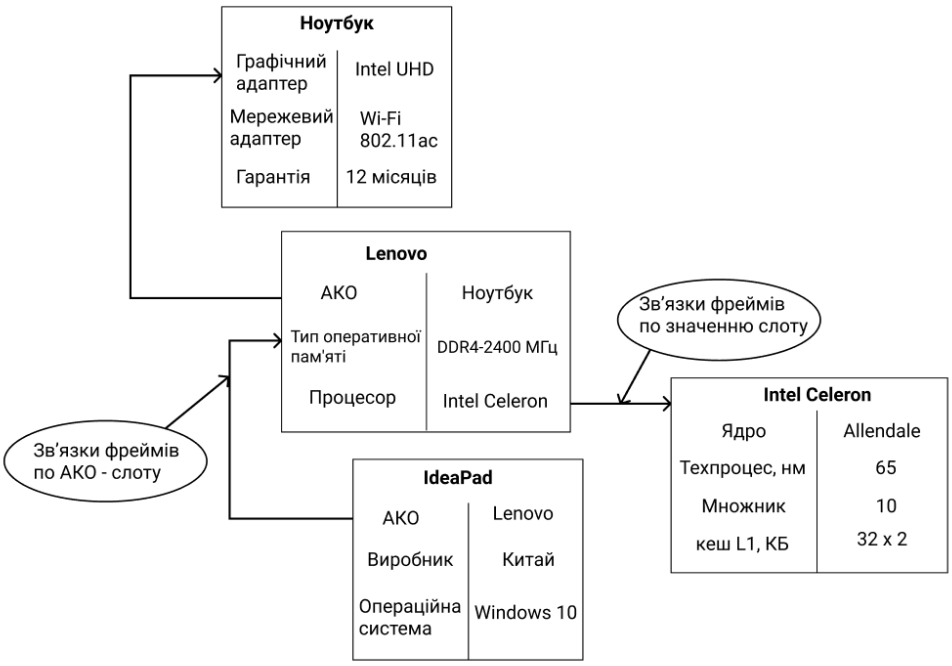


Рис. 1.2 Фрагмент мережі фреймів

Експертна модель представлення знань

Експертна система (ЕС) - це складний програмний комплекс, що акумулює знання фахівців у конкретних предметних областях і тиражуючий цей емпіричний досвід для консультацій менш кваліфікованих користувачів. Узагальнена структура експертної системи представлена на рисунку 1.3.

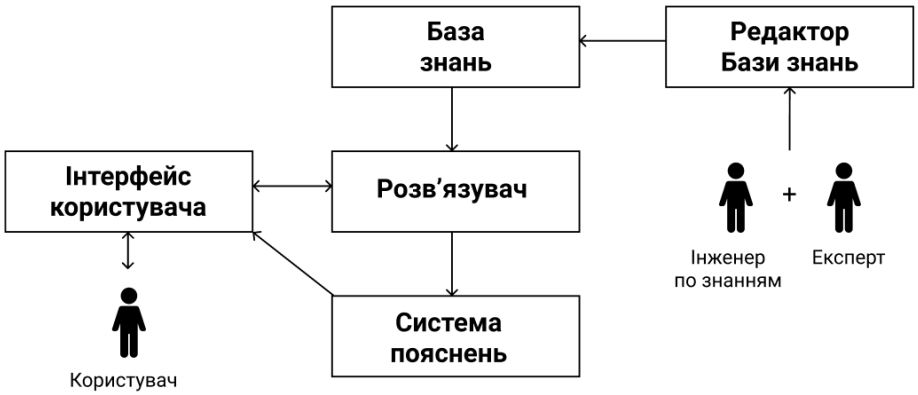


Рис. 1.3 Загальна структура експертної системи

Слід врахувати, що реальні експертні системи можуть мати більш складну структуру, однак блоки, зображені на малюнку, неодмінно присутні в будь-якій дійсній експертній системі, оскільки являють собою негласний канон на структуру сучасної експертної системи.

Інтерфейс користувача – комплекс програм, реалізують діалог Користувача з ЕС як на стадії введення інформації та отримання результатів («пояснення» рішення).

База знань (БЗ) – ядро ЕС, сукупність формалізованих знань предметної області, записана на машинному носії у формі, зрозумілій експерту та інженеру за знаннями. Паралельно такому поданню існує БЗ у внутрішньому «машинному» поданні.

Розв'язувач (машина виводу, інтерпретатор) – програма, що моделює хід міркувань експерта на підставі формалізованих знань, наявних в БЗ і вихідних даних (фактах), одержуваних від користувача.

Підсистема пояснень – програма, що протоколює роботу розв'язувача у вигляді «ланцюга логічних висновків». Вона дозволяє користувачу отримати відповіді на питання: «Як була отримана та чи інша рекомендація?» та «Чому система прийняла таке рішення?». Відповідь на питання «Як» - це трасування всього процесу отримання рішення із зазначенням використаних фрагментів БЗ, тобто всіх кроків ланцюга логічних висновків. Відповідь на питання «Чому» - посилання на умовивід, безпосередньо передувало отриманому рішенню, тобто відхід на один крок назад.

Редактор БЗ – програма, що представляє інженеру по знанням можливість створювати і поповнювати БЗ в діалоговому режимі. Здійснює введення формалізованих знань. Наприклад, правил продукційної моделі подання знання. Включає в себе систему вкладених меню, шаблонів мови представлення знань, підказок ("help" - режим) та інших сервісних засобів, що полегшують роботу з базою знань.

У розробці та експлуатації ЕС беруть участь як мінімум чотири людини:

а) експерт;

б) інженер по знаннях;

в) програміст (на схемі не показаний);

г) користувач.

Дослідники в області ЕС для назви своєї дисципліни часто використовують також термін «інженерія знань», введений Е.Фейгенбаумом як «приведення принципів та інструментарію досліджень з області штучного інтелекту в рішення важких прикладних проблем, що вимагають знань експертів».

Важливість експертних систем полягає в наступному:

- технологія експертних систем істотно розширює коло практично значущих завдань, що вирішуються на комп'ютерах, вирішення яких приносить значний економічний ефект;
- технологія ЕС є найважливішим засобом в вирішенні глобальних проблем традиційного програмування: тривалості і високої вартості розробки складних додатків;
- висока вартість супроводу складних систем, яка в кілька разів перевершує вартість їх розробки;
- об'єднання технології ЕС з технологією традиційного програмування додає нові якість програмним продуктам за рахунок: забезпечення динамічної модифікації додатків користувачем, а не програмістом; більшої «прозорості» застосунків (наприклад, знання зберігаються на

обмеженому, що не вимагає коментарів до знань, спрощує навчання і супровід); графіки; інтерфейсу і взаємодії.

Традиційно знання існують у двох видах – колективні знання, якими володіють більшість людей, і особисті знання, якими володіють фахівці (експерт).

Якщо більша частина знань в предметній області представлена у вигляді колективного досвіду (наприклад, вища математика), ця предметна область не потребує експертних системах (рис. 1.4а).

Якщо в предметній області велика частина знань є особистим досвідом фахівців високого рівня (експертів), якщо ці знання з яких-небудь причин слабо структуровані, така предметна область швидше за все потребує експертної системи (рис. 1.4б).

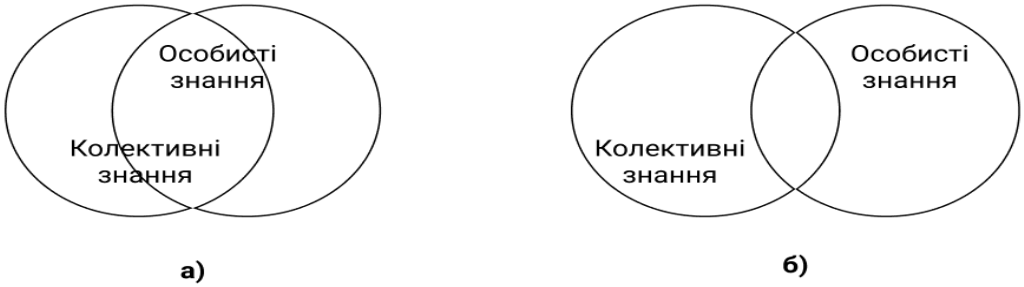


Рис. 1.4 Співвідношення колективних і особистих знань в предметній області

Необхідно відзначити, що в даний час технологія експертних систем використовується для вирішення різних типів завдань (інтерпретація, передбачення, діагностика, планування, конструювання, контроль, налагодження, інструктаж, управління) в найрізноманітніших проблемних областях, таких, як фінанси, нафтова і газова промисловість, енергетика, транспорт, фармацевтичне виробництво, космос, хімія, освіта, телекомунікації та зв'язок та ін. Однак ступінь охоплення вирішуваних завдань істотно відрізняється для різних предметних областей. Тому, що слід враховувати, що для вирішення всіх можливих завдань у предметній області

необхідно нескінченне число фактів та правил. Застосування ЕС реально тільки для вузьких предметних областей, а якщо бути більш точним, то для вирішення конкретного завдання в предметній області.

Дана модель представлення знань об'єднує в собі кілька тисяч різних програмних комплексів, які можна класифікувати за різними ознаками (див. рисунок 1.5).

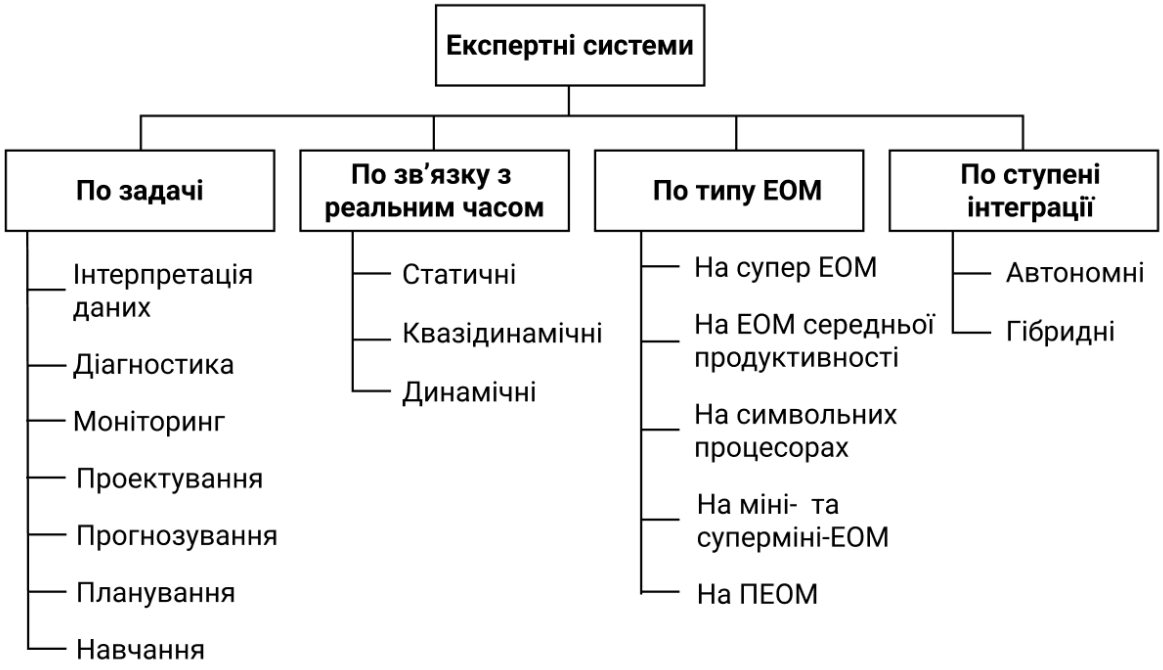


Рис. 1.5 Класифікація експертних систем

На думку провідних фахівців, ЕС знайдуть наступне застосування:

- ЕС відіграватимуть провідну роль у всіх фазах проектування, розробки, виробництва, розподілу, продажу, підтримки та надання послуг;
- технологія ЕС забезпечить революційний прорив в інтеграції застосунків із готових інтелектуально-взаємодіючих модулів;
- рішення, так званих, неформалізованих і слабоструктурованих задач.

У нашому випадку використовується *продукційний* підхід до організації обчислювального процесу тому, що на це впливають такі фактори:

а) переважна частина людських знань може бути представлена у вигляді продукцій.

б) системи продукцій є модульними. За невеликим винятком видалення або додавання продукцій в базу знань не призводить до змін в інших продукціях.

в) при необхідності системи продукцій можуть реалізовувати будь-які алгоритми і, отже, здатні відображати будь-яке процедурне знання, доступне ЕОМ.

г) наявність в продукціях вказівників (направлених зв’язків) на сферу застосування продукції дозволяє ефективно організовувати пам'ять, скоротивши час пошуку в ній необхідної інформації.

д) при об'єднанні систем продукцій і представлень у вигляді мереж створюються засоби, що володіють значною обчислювальною потужністю. Засоби побудовані за допомогою систем продукцій дозволяють отримувати нові знання, застосовуючи обробку декларативних знань та зв’язків між ними представлених у вигляді семантичної мережі.

е) природний паралелізм в системі продукцій, асинхронність їх реалізації роблять продукційні системи зручною моделлю обчислень для ЕОМ паралельної архітектури.

1.2 Основні поняття нечіткої логіки

Проблеми прийняття рішення в складних умовах займають в даний час особливе місце в інформаційних технологіях. Математичні методи широко застосовуються для опису та аналізу складних економічних, соціальних та інших систем. Теорія оптимізації створила сукупність методів, які допомагають, при використанні ЕОМ, ефективно приймати рішення при відомих та фіксованих параметрах. Системи прийняття рішення мають успіхи і в тому випадку, коли параметрами є випадкові величини з відомими законами розподілу. Однак, основні труднощі виникають тоді, коли параметри являються невизначеними і в той же час вони сильно впливають на результати рішення. Спеціалісти часто стикаються з необхідністю

розрахунків при наявності в рівняннях нечітко заданих параметрів або нечітко технологічної інформації. Виникаючі при цьому порушення рівності, балансових співвідношень і т.д. призводять до необхідності варіювати деякими параметрами для точного задоволення заданих рівнянь і отримання прийнятного результату. Такого роду ситуації можуть виникати як наслідок недостатньої вивченості об'єктів, так і через участь в управлінні людини або групи осіб. Особливість подібних систем полягає в тому, що значна частина інформації, необхідної для їх математичного опису, існує у формі уявлень або побажань експертів. Але в мові традиційної математики немає об'єктів, за допомогою яких можна було б досить точно відобразити нечіткість уявлень експертів.

Інший підхід спирається на передумову про те, що елементами мислення людини не є числа, а елементи деяких нечітких множин або класів об'єктів, для яких перехід від "приналежності до класу" до "неналежність» не скачкоподібний, а безперервний. Традиційні методи недостатньо придатні для аналізу подібних систем саме тому, що вони не в змозі охопити нечіткість людського мислення і поведінки. Це твердження наводить на думку про те, що для моделей процесів управління більше підійшли б нечіткі математичні методи, ніж класичні.

Теорія нечітких множин [4] була вперше запропонована американським математиком Лотфі Заде в 1965р і призначалася для подолання труднощів уявлення неточних понять, аналізу і моделювання систем, в яких бере участь людина.

Підхід на основі теорії нечітких множин є, по суті справи, альтернативою загальноприйнятим кількісним методам аналізу систем. Він має три основні відмінні риси:

а) замість або на додаток до числових змінним використовуються нечіткі величини – «лінгвістичні» змінні;
б) зв’язки між змінними описуються за допомогою нечітких висловлювань;

в) складні зв’язки описуються нечіткими алгоритмами.

Такий підхід дає наближені, але в той же час ефективні способи опису поведінки систем, настільки складних і погано певних, що вони не піддаються точному математичному аналізу.[2]

Теорія нечітких множин, введена Л. Заде [5] є математичним інструментом для перекладу абстрактних понять, що знаходяться в природній мові, в обчислювальну сутність. Нечіткі множини представляють розпливчасті описи або властивості об'єктів, тобто високий, середній, малий тощо. Теорія нечітких множин пропонує спосіб поводження з неточною інформацією, щоб імітувати людські міркування та сприйняття. Нечітка логіка стосується поняття часткової істини; значення істини варіюються від 0,0 - повністю помилкові до 1,0 - повністю правдиві. Тоді нечіткий набір А визначається як множина, функція приналежності μ_A має значення в діапазоні [0,0, 1,0]. Значення $\mu_A(x) = 0{,}0$ і $\mu_A(x) = 1{,}0$ стоять, відповідно, для нульового і повного приналежності від x до A, тоді як усі значення $\mu_A(x)$ від 0,0 до 1,0 вказують на часткову приналежність x до A. Математично нечіткий набір А представлений функцією приналежності, визначеною на домені X , що називається областю дискурсу:

$$\mu_A : X \to [\,0{,}1\,]$$

де А - нечітка мітка або лінгвістична змінна, що описує змінну x. Як розширення на булеву логіку, $\mu_A(x)$ являє собою ступінь приналежності x, що належить нечіткому набору А. Нечіткі множини можуть використовуватися для визначення значень нечітких змінних.

Розглянемо нечітку змінну якості сигналу, яку можна описати різними лінгвістичними термами, кожен з яких має свій нечіткий набір. Область дискурсу знаходиться в межах 0 - 100, показаний на рис. 3.1, де визначені нечіткі набори низького, середнього та високого стану якості. Чіткий стан 35 має ступінь приналежності 0,5 як для низьких, так і для середніх нечітких наборів. Сила нечітких змінних полягає в тому, що вони полегшують

поступовий перехід між атрибутами і, отже, мають природну здатність виражати та вирішувати невизначеності.

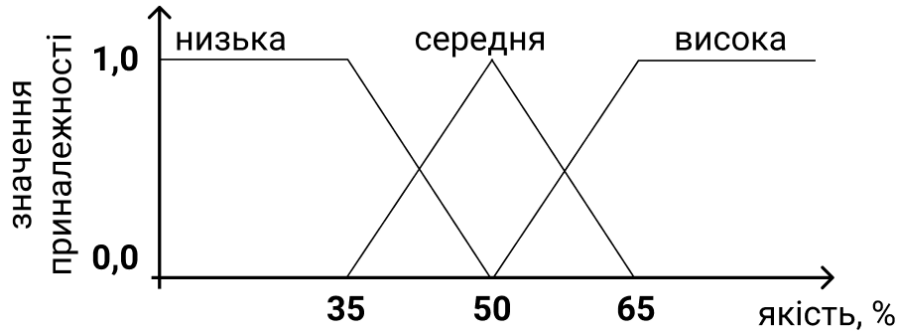


Рис. 1.6 Приклад нечітких множин, визначених для змінної якості

Властивості нечітких наборів відіграють важливу роль у моделюванні можливостей нечіткої системи, і щоб модель була справді прозорою, ці множини повинні чітко представляти терми, що описують вхідні та вихідні змінні. Експерт системи повинен визначити форму нечітких наборів. Однак у більшості випадків семантика, що охоплюється нечіткими множинами, не надто чутлива до варіації форми; отже, зручно використовувати прості функції приналежності. Найпоширенішими функціями приналежності є трикутна, трапецієподібна та функція Гауса, зображені на рисунку 1.7. Такі, нечіткі набори являють собою потужний підхід не тільки для боротьби з неповними або неточними даними, але й для розробки моделей даних, які забезпечують розумніші та плавніші характеристики, ніж традиційні методи моделювання.

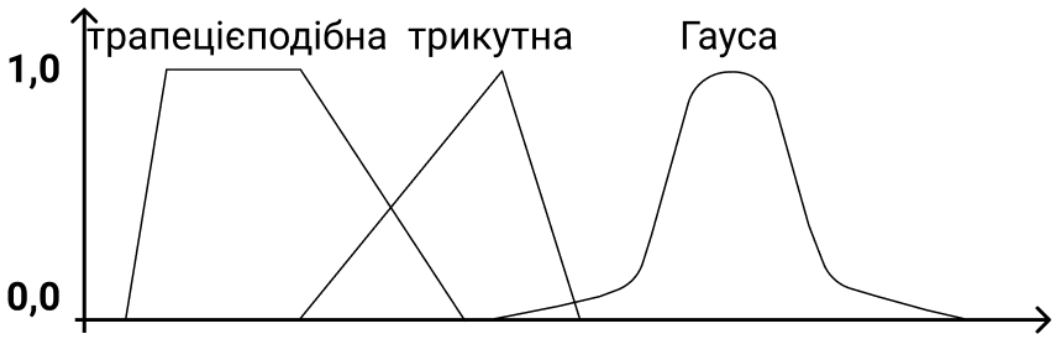


Рис. 1.7 Основні типи функцій приналежності

Популярність і практичність нечітких систем випливає з їх здатності виражати складні відносини з точки зору мовних правил. Таким чином, нечіткі системи мають переваги відмінних можливостей описати задане відображення вхід-вихід. Друга важлива властивість, яку слід розглядати, стосується наближення функції: нечіткі системи були доведені як універсальні апроксиматори, тобто вони здатні рівномірно наближати безперервні функції до будь-якої міри точності на закритих та обмежених множинах. Більше того, на відміну від інших універсальних апроксиматорів (наприклад, нейронних мереж) нечіткі системи однозначно підходять для включення мовної інформації природним та систематичним способами. Тому, нечіткі системи успішно використовуються для побудови моделей, які можуть пояснити складні процеси, такі як біологія, хімія, медицина або економіка, переростаючи в різні галузі застосування.

Нечіткі системи, основані на правилах

Схема роботи нечіткої системи заснована на конкретному процесі виводу де задіяні змінні моделюються за допомогою нечітких наборів.

Використовуючи нечіткі набори, нечітка система може представляти неточність, що характеризує проблеми реального світу, використовуючи правила ЯКЩО … ТО … виражені природною мовою. Колекція таких правил, що називаються базою знань, використовується для опису відображення наближення вводу/виводу, а також застосовується належний процес висновку. Звідси, нечіткою системою, основана на правилах, наближає невідоме відображення вводу/виводу за умовиводу з набору нечітких правил, що є зрозумілими для людини твердженнями, такими як опис типового співвідношення між рівнем шумів та бажаним виходом якості сигналу.

R1: Якщо *рівень шумів* є *низька* То *якість сигналу* є *висока*

R2: Якщо *рівень шумів* є *середня* То *якість сигналу* є *середня*

Такі правила використовують лінгвістичні терми, такі як "низький" та "середній", які виражають розпливчасті поняття, визначені вхідною змінною "рівень шумів". Такі розпливчасті поняття моделюються як нечіткі набори.

Основна структура нечіткої системи, описана Мамдані і Ассіліаном [6], показано на рисунку 1.8. Нечітка система обробляє чіткі дані на вході і виробляє чіткі дані на виході за допомогою виведення з бази нечітких правил, яка являє собою базу знань системи. На початку даної системи для перетворення чітких даних в нечіткі змінні використовується фазифікатор, а на виході системи-дефазифікатор для перетворення нечітких множин в чіткі значення.

Рис. 1.8 Загальна схема системи нечіткого виведення

Механізм нечіткого виведення об'єднує правила в базі правил відповідно до наближеної теорії міркувань для отримання відображення від нечітких множин у вхідному просторі до нечітких множин у вихідному просторі. Таким чином, нечітка система надає обчислювальну схему, яка описує, як правила повинні бути оцінені і об'єднані для обчислення чіткого вихідного значення (вектора) для будь-якого вхідного чіткого значення. Тому можна розглядати нечітку систему просто як параметричну функцію, яка зіставляє реальні вектори з реальними векторами.

Відповідно до форми нечітких правил для визначення системи, заснованої на нечітких правилах, можуть бути використані різні моделі. У нечіткій системі типу Мамдані наслідком кожного правила є нечітка множина, визначена для лінгвістичної змінної. Висновок з правил слідує моделі modus ponens, розширеної до нечітких множин. А саме, враховуючи

вхідне значення, висновок правила полягає в зменшенні нечіткої множини наслідків на величину спрацьовування правила, яка визначається ступенем приналежності вхідного значення до передумов. Визначення оператора ТО, також називається оператором імплікації, дає вихідну нечітку множину для цього правила.

Через використання нечітких множин в передумові правил, кілька нечітких правил можуть бути запущені одночасно. Наприклад, з нечіткими множинами, визначеними на рис. 3.1, обидва ці правила R1 і R2 спрацьовують при якості сигналу 35. Отже, вихідна команда обчислюється шляхом об'єднання нечітких множин, отриманих в результаті виведення обох правил R1 і R2. Зазвичай для виконання агрегації правил, тобто для об'єднання часткових результатів, що надаються окремими правилами, використовуються оператори максимуму або суми.

Однак для більшості керуючих додатків в якості вихідних даних потрібні чіткі значення, тому якісна інформація у вигляді нечітких множин не підходить для вихідної змінної. Тому фаза дефазифікації необхідна для отримання вихідного чіткого значення з нечіткої множини. Існує кілька методів дефазифікації, деякі з яких засновані на центроїді результатів, а інші - на максимальних значеннях, заданих функціями приналежності.

Фазифікація

Фазифікація – це процес, який перетворює чіткі вхідні дані в нечіткі множини, визначені на вхідному просторі. Зазвичай компонент системи, що виконує цей процес, називається фазифікатором. На цьому етапі вводиться функція фазифікації або кожна вхідна змінна для вираження пов'язаної з нею невизначеності вимірювань. Мета функції фазифікації полягає в тому, щоб інтерпретувати вимірювання вхідних змінних, кожна з яких виражається реальним числом, як більш реалістичною нечіткою апроксимацією відповідних реальних чисел. Більш розпливчасті моделі невизначеностей виявляються на вході і отже це згладжує відповідь системи, що робить його

менш чутливим до конкретних вхідних значень і, отже, до невизначеності на вході, такий як шум.

У багатьох випадках вхідні змінні не є чіткими. Тобто застосовується одноразова фазифікація, яка передбачає відсутність шуму на чітких входах, тому вимірювання вхідних змінних використовуються безпосередньо в процесі виведення. Одноразова фазифікація зіставляє чітке вхідне значення x_0 нечіткому синглтон, тобто нечіткій множині, так що його підтримка зводиться до x_0. Цей тип фазифікації вимагає низьких обчислювальних витрат, так як розрахунок вихідних даних системи спрощений. Однак одноразова фазифікація не завжди може бути адекватною, особливо в тих випадках, коли вхідні дані пошкоджені шумом. Для врахування невизначеності в даних необхідна не одноразова фаззифікація [7]. У таких випадках функція фазифікації має вигляд:

$$f_e: [-a, a] \rightarrow X$$

де X позначає множину всіх нечітких множин, $f_e(x_0)$ - нечітка апроксимація вимірювання x_0.

База нечітких правил і механізм виведення

Для нечіткої системи з n входами і одним виходом база правил складається з набору K нечітких правил, формально визначених як

$$R_k : \text{Якщо } (x_1 \in A_{1k}) \text{ Та } \ldots \text{ Та } (x_n \in A_{nk}) \text{ То } (y \in B_k) \qquad (1.1)$$

для k = 1, ..., K, де Aik, i = 1, ... N є нечіткі множини, визначені на вхідних змінних, а Bk-це нечітка множина, визначена на вихідних змінних. Антецедент правила можна розглядати як багатовимірний нечіткий набір Ak, отриманий як перетин одновимірних нечітких множин Aik, i = 1, ..., N. Звідси, основна форма нечіткого правила (1.1) може бути записано як

$$R_k : \text{Якщо } (x_1 \in A_k) \text{ То } (y \in B_k) \qquad (1.2)$$

Використовуючи оператор нечіткої імплікації, кожне правило зіставляє попередню нечітку множину на кшталт наступної нечіткої множини Bk. Тоді

кожне правило Rk можна розглядати як нечітку імплікацію Ak → Bk, що характеризується безперервною багатовимірною функцією приналежності.

$$\mu_{Rk}(x,y) = \mu_{Ak} \rightarrow \mathrm{B}_k(x,y) = \tau\big(\mu_{Ak}(x), \mu_{Bk}(y)\big) \qquad (1.3)$$

де τ є T- оператором норми.

Далі оброблені методами логіки невиразності вхідного сигналу A0 = (A10, ..., A0), механізм виведення використовує базу нечітких правил для виведення нечіткого виведення B0 за допомогою композиційного правила виведення. Композиційне правило може бути застосоване локально до кожного правила Rk, і результуючі нечіткі множини агрегуються для отримання виведеної нечіткої множини. Зокрема, механізм виведення спочатку становить A0 (нечіткий вхід) з кожним правилом Rk, виробляючи в якості проміжного результату нечіткий набір Bk0 = A0 ◦ Rk з функцією приналежності

$$\mu_{Bk0}(y) = \sup x[\tau(\mu_{A0}(x), \mu_{Rk}(x,y))] \qquad (1.4)$$

Потім нечіткий вихід виводиться як об'єднання всіх окремих нечітких виходів, тобто $\mathrm{B}_0 = \cup_{k=1}^{k} \mathrm{B}_{k0}$, що є K числом нечітких правил в базі правил.

Дефазифікація

Часто вихід системи, заснованої на нечітких правилах, повинен бути чітким значенням, що є істотною вимогою в багатьох інженерних завданнях, наприклад в програмному забезпеченні, побудованому на базі алгоритмів нечіткого керування. У цих випадках стадія дефазифікації необхідна для отримання чіткого виведення з нечіткого виведення, отриманого в результаті виведення правил. Цей етап виконується дефазифікатором, який відображає вихідний нечіткий набір в єдину чітку точку у вихідному просторі. Існує безліч різних методів дефазифікації, більшість з яких описані в розділі [8-11]. Найбільш вдалим методом є метод центру ваги або центру області, в якому значення чіткості для загальної вихідної змінної y визначається як

$$\mathrm{y}_0 = \frac{\int_y \mu_{B0}(y) y dy}{\int_y \mu_{B0}(y) dy} \qquad (1.5)$$

Як правило, для зниження обчислювальних витрат використовується дискретне представлення наведеної вище формули

$$y_0 = \frac{\sum_{q=1}^{N_q} \mu_{B0}(y_q) y_q}{\sum_{q=1}^{N_q} \mu_{B0}(y_q)} \tag{1.6}$$

де N_q – число кроків квантування, за допомогою яких дискретизується область дискурсу Y. Існує також ряд обчислювально-спрощених методів дефазифікації, які об'єднують агрегацію правил і дефазифікацію в одну фазу. Одним з них є середньозважений метод, який обчислює дефазифікований вихід як

$$y_0 = \frac{\sum_{k=1}^{K} \mu_k(x_0) b_k}{\sum_{k=1}^{K} \mu_k(x_0)} \tag{1.7}$$

де K – кількість правил, $\mu_k(x_0)$ – ступінь активації k-го правила, а b_k – числове значення, пов'язане з послідовністю b_k k-го правила.

Теорія нечітких множин стала важливим направленням в моделюванні та керуванні, оскільки вона дає нові можливості в науково-технічних дослідженнях, а саме:

а) Можливість створення штучного інтелекту, подібного з інтелектом людини, і його застосування в автоматах і роботах. В даний час спостерігається стійка і навіть зростаюча тенденція до отримання в цьому напрямку результатів, які свідчать про те, що для ряду конкретних програм штучний інтелект перевершує людський за обсягом і швидкості обробки інформації.

б) Створення комп'ютерів, програмованих за допомогою природної мови. Застосування таких комп'ютерів в автоматах і роботах робить можливим керування ними і «спілкування» з ними на природній мові з використанням нечітких понять.

в) У разі обробки значних обсягів інформації (Big Data) наявні такі характеристики інформації як недостатньо чітко визначена структуризація інформації, складність її розділення на певні групи або

кластери. Використання інформації будь-якого ступеня гранулювання в задачах моделювання, керування, оптимізації та діагностики. Більш висока ступінь гранулювання може привести до скорочення обсягів інформації, що обробляється і зберігається і до підвищення швидкодії алгоритмів.

г) Зазначені вище особливості Big Data потребують підстроювання рівня гранулювання інформації під необхідну точність моделювання, керування, оптимізації, діагностики, що є перевагою методів нечіткої логіки.

1.3 Аналіз методів зменшення кількості нечітких логічних правил бази знань

Важливим етапом розробки експертної системи (ЕС) є верифікація – процес доведення того, що БЗ експертної системи не містить внутрішніх помилок, таких як надмірність, суперечливість і неповнота [12]. Головним завданням верифікації БЗ є виявлення аномалій, а саме суперечливих або надлишкових правил, циклів. Ґрунтуючись на отриманій в результаті аналізу інформації, в базу вносяться зміни, необхідні для організації коректного висновку. Одним з ключових елементів верифікації продукційної БЗ є виявлення помилок в ланцюгах виведення.

Зі збільшенням розмірів БЗ зростає число розгалужень і взаємозв'язків між ланцюгами виведення, і їх аналіз стає досить складним завданням [13]. Крім того, навіть якщо в результаті виконання верифікації в БЗ відсутні аномалії, висновок в такій базі все ж може давати невірні результати [14]. Причина цього полягає в наявності смислових помилок, характерних для конкретної предметної області (ПО). У свою чергу, виникнення таких смислових помилок в процесі виведення є наслідком того, що машина виведення і стратегії управління продукціями (за винятком, до деякої міри, метапродукцій) виконують дії над правилами механічно, без урахування відносин між поняттями ПО, представленими в цих правилах.

Для того щоб розібратися в причинах виникнення смислових помилок, розглянемо, яким чином експертне знання може бути відчужене і представлено у вигляді, придатному для використання експертною системою. Бази знань ЕС 70-х-початку 80-х рр. ґрунтувалися на використанні будь-якого одного формалізму (часто продукційних правил) для представлення знань про предметну область. Різні аспекти знання (як вирішувати задачу, що являє собою предметна область, чому все відбувається саме так) описувалися на єдиному рівні абстракції і були неявно взаємопов'язані [15]. Така організація знань приводила до різних проблем при придбанні знань, поясненні одержуваних результатів, роботі поза сферою компетенції ЕС і в процесі підтримки ЕС.[16, 17]

На відміну від ранніх робіт, так звані ЕС другого покоління ґрунтуються на використанні різних видів знання і різних уявлень. У цих системах знання поділяється на керуюче (control knowledge) і знання про предметну область (domain knowledge). Крім того, знання може бути структурним і функціональним, якісним і кількісним, причинно-наслідковим (causal) і емпіричним.[15]

Вимоги до «хорошого» способу представлення знання досить широкі і можуть включати конфліктуючі критерії (наприклад, виразність опису і ефективність виведення). Тому замість вибору одного універсального формалізму в ЕС другого покоління нерідко стали використовувати кілька різних уявлень, кожне з яких найбільш підходить для опису певного типу знання, виконання певних типів виводу або навіть описує одне і те ж знання в різній формі.[18]

Більшість стратегій управління продукціями (наприклад, засновані на використанні пріоритетів або вибирають правила з найбільш загальним або найбільш конкретною умовою) не враховують ситуації, що виникає в процесі виведення. Рішення про активацію того чи іншого правила приймається на основі апріорної інформації, яку інженер вносить при заповненні БЗ. Проблемою є те, що кожне правило, що додається в БЗ, може мати смислові

взаємозв'язки з іншими правилами. При цьому більшу їх частину модель не відображає. Для БЗ невеликого розміру інженеру не складає труднощів відстежувати ті відносини між правилами, які можуть вплинути на результати, і вносити необхідні зміни в стратегію управління продукціями або в структуру БЗ.

Інформація, на основі якої виконується вирішення конфлікту, може бути описана в правилах не повністю. При формулюванні правил опускається значна частина інформації, Наприклад, різні другорядні умови активації правил, крайні випадки, аспекти ситуації, що розуміються інженером і експертом як само собою зрозумілі і через це стає проблематичним відстежувати взаємозв'язки між правилами в БЗ, так як частина інформації, на якій ґрунтується управління, не представлена явно. Крім того, цю інформацію необхідно співвідносити з можливими конфліктами, відстежувати виникнення яких стає все важче з ростом БЗ. Для вирішення цих проблем пропонується модифікувати стратегію управління продукціями так, щоб вона враховувала ситуацію, що виникає в процесі виведення.

Реконструювання відносин між правилами

Щоб вирішення конфліктів активації ґрунтувалося на ситуації, що виникає в процесі виведення, необхідно вирішити наступні завдання:

а) Описати необхідну для прийняття рішення інформацію про істотні в по взаємозв'язках між правилами.

б) Зіставити цю інформацію з можливими конфліктами.

Подання необхідної для прийняття рішення інформації пропонується проводити в два етапи. Спочатку всю істотну в ПО інформацію, як ту, яка була опущена при формулюванні правил, так і ту, яка представлена в них, пропонується систематично описати в онтології ПО. Це дозволить явно уявити використовувані поняття ПО і відносини між ними.

На другому етапі, ґрунтуючись на цій інформації, необхідно виконати опис ситуацій ПО, в яких можуть виникати конфлікти активації. У цьому

описі повинна міститися істотна для ПО інформація, яка була опущена при формулюванні правил. Саме на її основі стратегія управління продукціями буде виконувати вирішення конфлікту.

Для представлення ситуацій пропонується використовувати пам'ять, засновану на епізодичних пакетах (Episodic Memory Organization Packets E-MOPs) [19]. Цей підхід ґрунтується на моделі динамічної пам'яті, запропонованій Шенком [20]. Запам’ятовування в динамічній пам'яті здійснюється шляхом пошуку найбільш схожих елементів у пам'яті та їх адаптації до нових даних [21]. Для цього вміст пам'яті організовано в концептуальні категорії, засновані на смисловій схожості. Ці концептуальні категорії описуються у вигляді епізодичних пакетів (E-MOPs). Епізодичні пакети організовують схожі ситуації відповідно до їх відмінностями, а також будують узагальнення на основі їх подібних рис. Кожен епізодичний пакет містить: норми, що описують узагальнену інформацію, що характеризує збережені ситуації; деревоподібну структуру, що індексує ситуації відповідно до їх відмінностей [19]. При виникненні конфлікту активації необхідно витягти з пам'яті відповідний опис ситуації. Це завдання розбивається на кілька етапів. Спочатку необхідно сформувати запит на основі тієї інформації, яка описана в правилах. Для цього пропонується розібрати правила на поняття ПО, ґрунтуючись на онтології. Отриманий опис буде використовуватися як ключ для пошуку в пам'яті.

Роботу з ситуаціями, представленими в епізодичній пам'яті, пропонується засновувати на міркуванні за прецедентами (case-based reasoning). Прецедентом в нашому випадку буде опис ситуації ПО. Цикл міркування за прецедентами складається з чотирьох етапів [22]:

- витяг прецеденту, найбільш підходящого поточному завданню;
- використання інформації, що міститься в ньому, для вирішення нового завдання;
- перевірка отриманого рішення;

– збереження корисного досвіду.

Ситуація, відповідна сформованому на попередньому етапі запиту, витягується з пам'яті шляхом проходу за індексами епізодичних пакетів [23]. Запит, сформований на основі інформації, що міститься в конфліктуючих правилах, являє собою частковий опис ситуації ПО. В епізодичній пам'яті має міститися повний опис цієї ситуації (тобто разом з усією інформацією з онтології, необхідної для вирішення конфлікту). У цьому випадку використання такого прецеденту полягає в передачі його стратегії управління продуктами, яка використовує його для вирішення конфлікту.

Стратегія управління продукцією групує правила відповідно до отриманого опису. Виконується розбір правил на поняття ПО і їх зіставлення з описом ситуації. В описі ситуації повинні бути представлені всі релевантні взаємозв'язки між поняттями. Конфлікт виникає при наявності взаємовиключних альтернатив. Аналіз взаємозв'язків дозволить визначити, яка з альтернатив краще.

Перевірку побудованого рішення (опису ситуації) виконує інженер. Рішення визнається коректним, якщо на його основі стратегія управління продукціями правильно вирішує конфлікт активації. У цьому випадку рішення зберігається в епізодичній пам'яті. В іншому випадку виконується перевірка наступної гіпотези.

1.4 Аналіз методів візуалізації бази знань з перевіркою їх на коректність

Візуалізація – це процес перетворення великих і складних видів абстрактної інформації в інтуїтивно зрозумілу візуальну форму. Універсальним засобом такого подання структурованої інформації є графи. Графи застосовуються для представлення будь-якої інформації, яку можна промоделювати у вигляді об'єктів і зв'язків між об'єктами [24, 25]. Тому візуалізація графових моделей є ключовою компонентою в багатьох додатках в науці і техніці, а методи візуалізації графів являють собою теоретичну

основу методів візуалізації абстрактної інформації. Методи та засоби візуалізації графів і графових моделей широко використовується в таких областях, як інформаційні системи та програмне забезпечення, комп'ютерне навчання, біологічні науки, штучний інтелект, аналіз фінансової інформації, комп'ютерне моделювання та багато інших.

Зображення (drawing) графа на площині (або в просторі) — це відображення вершин і ребер графа в множину точок площини (або простору). Зрозуміло, що один і той же граф можна візуалізувати різними способами (див. рисунок 1.9), причому якість одного і того ж зображення може по-різному оцінюватися, а різні застосунки можуть вимагати різні способи візуалізації графа. Наприклад, при роботі з різною таксономією часто використовують ортогональне розташування ребер і міток, так як взаємодія із зображенням має полегшувати читання супутньої інформації, а при візуалізації карт доріг потрібно, щоб розташування вершин і ребер відповідали географічним реаліям. Тому головним критерієм оцінки якості методів візуалізації інформації є адекватність зображення графової моделі заданому типу інформації та характеру її використання.

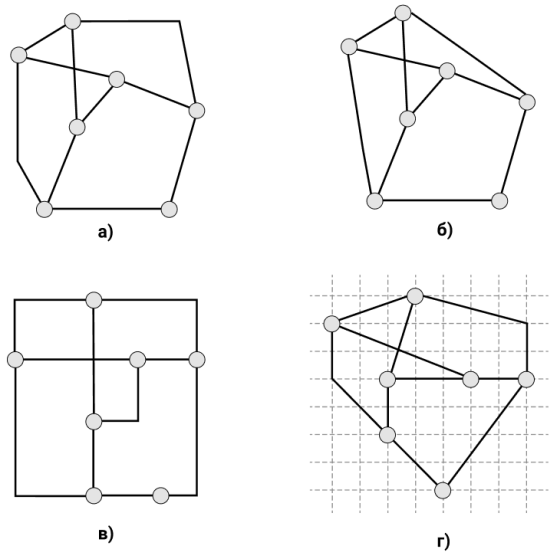


Рис. 1.9 Різні зображення одного графа: а) полі-лінійні;
б) прямолінійні; в) ортогональні; г) сітчасті

Найбільш простим і широко використовуваним для вирішення завдань візуалізації інформації є графи у вигляді дерева. Алгоритми розміщення дерев мають найменшу складність і є найбільш простими для реалізації.

Кореневі дерева часто використовуються для представлення ієрархій, таких як генеалогічні дерева, організаційні схеми або пошукові дерева. До їх зображень часто застосовують додаткові угоди, естетичні критерії та обмеження. Наприклад, при угоді включення (inclusion) вершини кореневого дерева зображуються прямокутниками, а відносини батьківського та дочірнього представляються включенням одного прямокутника в інший, а угода перекидання (tip-over) подібно класичному угоди спадного плоского зображення кореневого дерева, проте допускає, щоб дочірній елемент деяких вершин розташовувалися не горизонтально, а вертикально (див. рисунок 1.10).

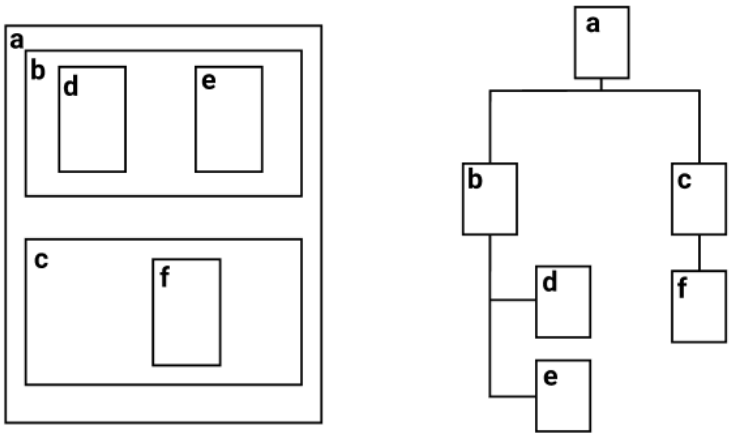


Рис. 1.10 Різні зображення одного дерева: а – включення;
б – перекидання

Простий і ефективний метод побудови спадного плоского зображення кореневого дерева T полягає у використанні рівневого розташування дерева, при якому вершини p глибини i мають y-координату $y(p) = -i$, а x-координати присвоюються таким чином, щоб різниця $x(p)$ $x(p')$ мала той же знак, що і різниця x(батько(p))-x(батько(p')). Деякою варіацією рівневого

зображення дерева є його радіальне розташування, в якому рівні мають вигляд концентричних кіл, а піддерева займають секторні сегменти (див. рисунок 1.11).

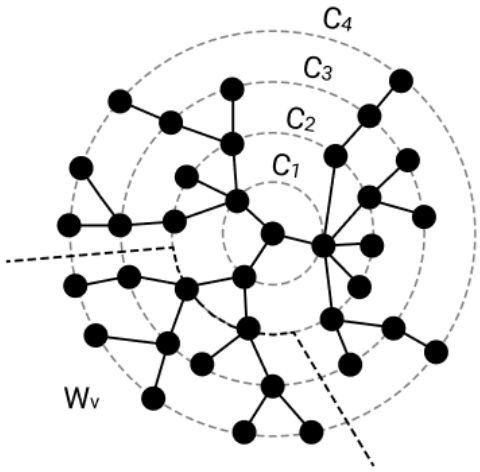


Рис. 1.11 Радіальне зображення дерева

Більшість методологій малювання графа ґрунтується на наступних двох простих спостереженнях: естетичні критерії часто суперечать один одному і таким чином пошуки компромісів неминучі; навіть якщо естетичні критерії не конфліктують, часто алгоритмічно важко задовольнити всім їм одночасно. Є ряд методів, які дозволяють отримати задовільні рішення задач візуалізації графів; основними серед них є наступні.

Планаризация. Плоскі розташування графів, тобто без перетинів ребер, зазвичай більш привабливі, ніж неплоскі. Наприклад, вони дуже важливі в технологіях друкованих плат з точки зору мінімізації розміщення. Деякі планарні графи можна намалювати на площині таким чином, щоб кожна межа грані була опуклим багатокутником. Таке уявлення можливе для графа тільки тоді, коли межі всіх граней є простими циклами. Граф, що не є двосвязним, не має опуклого уявлення, а для будь-якого 3-зв'язкового графа існує опукле уявлення (теорема Татта).

Використання фізичних аналогій. Ці методи інтерпретують граф при побудові його зображення як фізичну систему з силами між вершинами і намагаються мінімізувати енергію системи для отримання хорошого

малюнка. Такого типу алгоритми використовуються для малювання довільних (розріджених) мереж, таких як блок-схеми, графи програмного планування, графи телефонних викликів і т.п. вони також застосовуються для кластерних зображень.

Порівневі або Сугіяма-подібні методи. Найбільш широко використовуваними алгоритмами для малювання ациклічних орграфів є алгоритми, що відносяться до класу, запропонованих Сугіямою. Вони виробляють порівневі (або ієрархічні) зображення, намагаючись також мінімізувати кількість перетинів або розмір області розміщення. Вибір цього підкласу для малювання можна пояснити двома причинами. По-перше, переважна більшість реальних графів, що зустрічаються в програмуванні, є ациклічними, а, по-друге, будь-який орієнтований (і тим більше неорієнтований) граф може бути перетворений до ациклічного орграфу шляхом зміни або завдання орієнтації у частини його ребер.

Потокові методи. Проблема мінімізації числа згинів може ефективно вирішуватися шляхом зведення її до задачі потоку в мережі, принаймні в тих випадках, коли зафіксована топологія розміщення. Ті ж самі методи можуть застосовуватися для максимізаціїї кутів між ребрами.

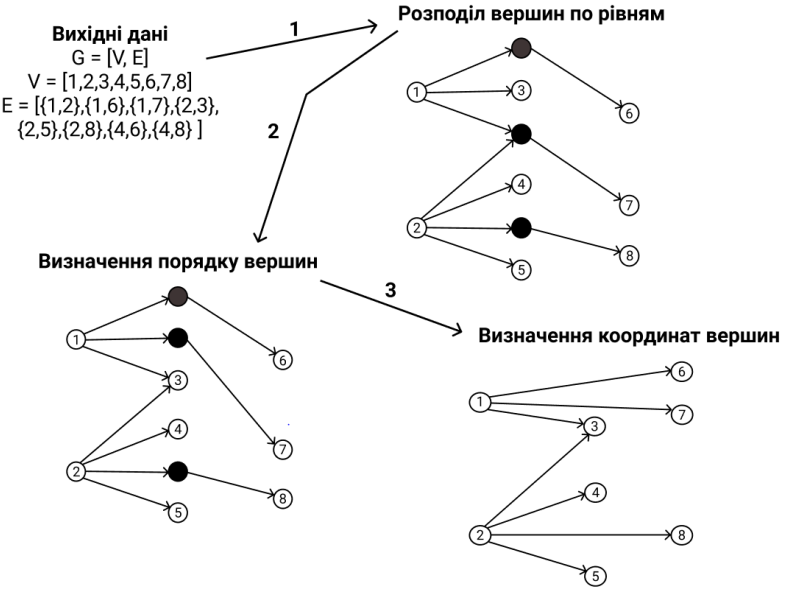


Рис. 1.12 Етапи побудови рівневого зображення

Візуалізація великих графів

Розмір візуалізованого графа є ключовою проблемою при візуалізації графів. Великі графи ставлять ряд складних проблем. Якщо число елементів графа велике, його обробка може виходити за межі продуктивності або навіть досягати граничних можливостей використовуваної для візуалізації платформи. Навіть якщо можливо розмістити і показати всі елементи великого графа, виникають проблеми наочності і зручності, оскільки стає неможливим розрізняти вершини і дуги. По суті, проблема зручності виникає навіть раніше, ніж проблема відмінності елементів. Відомо, що охоплюючий і детальний аналіз даних в графовій структурі найбільш простий, коли розмір демонстрованого графа невеликий.

Взагалі кажучи, немає сенсу перевіряти планарність графа, що складається з декількох сотень вершин, і намагатися мінімізувати кількість перетинів ребер. Часто більш очевидним і практичним рішенням є просте розміщення покриває дерева графа. Як відомо, алгоритми розміщення дерев мають найменшу складність і є найбільш простими для реалізації. При такому підході проблема візуалізації включає завдання знаходження. Він також передбачає побудову зображення графа на основі вже побудованого зображення дерева, яке містить всі вершини даного графа і перед цим було витягнуто з нього. При цьому сам процес побудови зводиться до додавання деяких додаткових ребер до вже наявного зображення дерева. Істотно, що, використовуючи певні зображення покривають дерев для укладання графів, ми можемо також отримувати розміщення графа з певними властивостями.

При роботі з великими графами часто використовуються спеціальні методи, що дозволяють зменшувати кількість деталей, що розміщуються одночасно. Замість статичного розміщення також використовуються різні інтерактивні методи із застосуванням навігації і методів виділення (фокус та контекст), що включають геометричну або семантичну деформацію, кластеризацію, агрегацію та інші техніки.

Кластеризація (clustering) в загальному випадку — це процес розбиття заданої вибірки об'єктів (ситуацій) на підмножини, звані кластерами (clusters), так, щоб кожен кластер складався зі схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Кластеризація дозволяє зменшувати кількість видимих елементів графа зі збереженням при цьому його глобальної структури. Конкретним прикладом, де її застосування досить виправдано, є так звані графи малих світів (small-world graphs) — графи, які мають невеликий середній найкоротший шлях між вершинами, але високий ступінь кластеризації (у порівнянні з усередненим графом відповідного розміру). Ця властивість притаманна багатьом графам реальних додатків, таких як соціальні мережі, нейронні мережі, програмні системи, мережі електропостачання, бази знань з перехресними посиланнями та інтернет.

Якщо кластеризація виконується послідовним застосуванням одного і того ж процесу кластеризації до груп, виявлених на попередньому кроці кластеризації, то говорять про ієрархічну кластеризацію (hierarchical clustering). Результатом такої ієрархічної кластеризації є ієрархія по включенню, і по ній можна здійснювати навігацію як по дереву, в якому кожен кластер представлений у вигляді вершини. Отже, ієрархічна кластеризація може бути використана для породження ієрархій у графах і побудови на їх основі так званих ієрархічних графів (hierarchical graphs), орієнтованих на підтримку візуалізації складних інформаційних моделей [26-31].

Ієрархічний граф H - це пара (G, T), яка складається з графа G і кореневого дерева T, вершини якого відповідають елементам деякої ієрархії фрагментів F графа G, а дуги відображають відношення їх безпосередньої вкладеності. T називається деревом вкладеності, а G-основним графом ієрархічного графа H (див. рисунок 1.13).

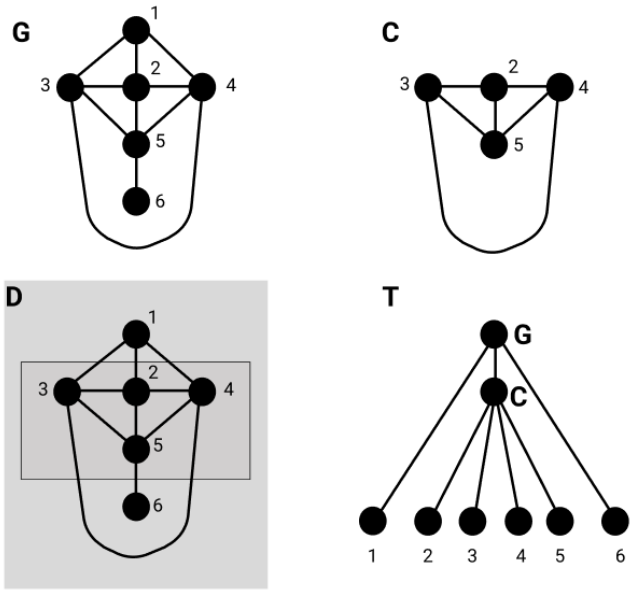


Рис. 1.13 Ієрархічний граф H = (G, T), в якому
C - єдиний нетривіальний фрагмент, відмінний від G і його зображення
D на площині

У ряді випадків тільки за рахунок вибору відповідного способу зображення ребер можна зменшити заплутаність (clutter) зображення, що особливо важливо для тих графів, чиї вершини повинні мати зумовлені позиції, як це, наприклад, має місце в геоінформаційних додатках. Зокрема, заплутаність зображення можна зменшити шляхом заміни полілінійних ліній, що зображують ребра, на гладкі криві. Інші способи-це різні методи злиття зображень ребер або їх вибіркового зображення.

Гіперграф – це таке узагальнення неорієнтованого графа, коли ребрами можуть служити довільні, а не тільки двовершинні і одновершинні, підмножини заданої множини вершин. Завдання на виділення найменшої системи ребер, що містить всі вершини, або найбільшої системи ребер попарно без загальних вершин, на вибір системи різних вершин представників для всіх ребер і т.п. були відомі раніше і частково вирішувалися в загальному теоретико множині плані, але лише зовсім недавно виявилося, що методи теорії графів можна пристосувати до вирішення таких завдань. Це призвело до розвитку теорії гіперграфів на основі узагальнення ряду найважливіших понять теорії графів.

Перші результати основоположного характеру отримав в 1963р Д.Рей-Чоудхурі, показавши, що метод чергуються ланцюгів, що грає одну з головних ролей в теорії графів і її додатках, переноситься в дещо зміненому вигляді на гіперграфи і може стати інструментом для вирішення згаданих вище завдань (і їх узагальнень) про покриттях. Другий фундаментальний напрямок в теорії гіперграфів пов'язано з перенесенням в неї понять правильної розмальовки вершин і хроматичного числа; апріорі можливо кілька нерівносильних визначень, серед яких найбільш цікаве належить П.Ердешу і А.Хайналу (1966). Як показано на русунку 1.14, при природному визначенні ступеня гіперграфа зберігає силу класична теорема Брукса; інше ж визначення ступеня дозволило І.Томеску (1968) перенести на гіперграфи більш тонку верхню оцінку хроматичного числа через ступені вершин, отриману для графів Д.Уелшем і М.Пауелом.

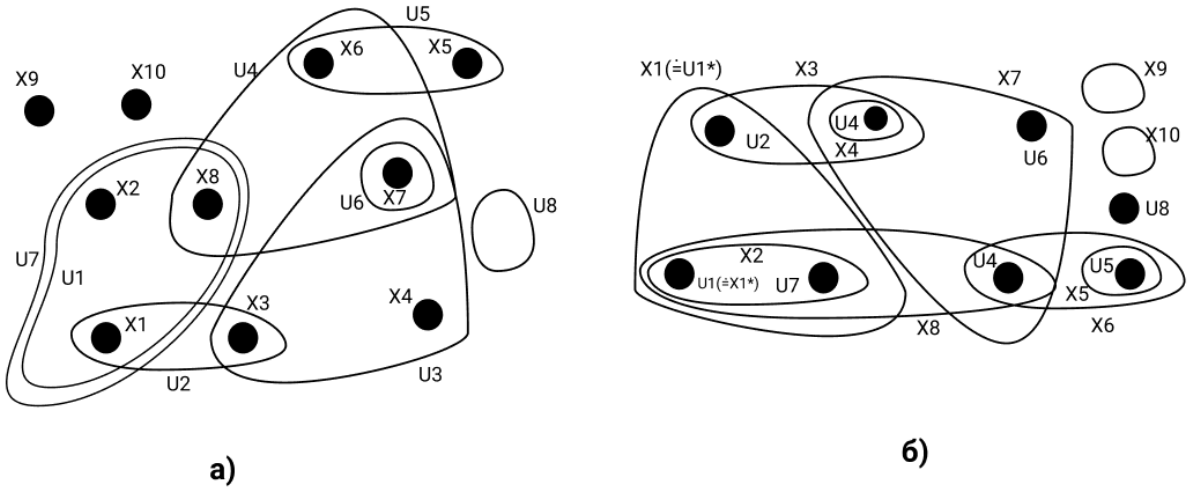


Рис. 1.14 Приклад гіперграфів

Але не всі вищеописані методи відповідають вимогам по швидкості проведення логічного виведення та аналізу на відповідність основним властивостям, виправлення помилок та неточностей. За допомогою простого графа ми не можемо описати та візуалізувати логічні зв'язки взаємозв'язаних множин елементів та їх взаємодію з іншими групами елементів. У такому випадку даний метод генерації зображень графів і графових структур

враховують критерії, які показують, яке зображення вважатиметься хорошим.

На сьогоднішній день, ні один із приведених алгоритмів не використовував можливість подання бази нечітких знань за допомогою метаграфу. Саме тому виникає доречність та актуальність створення автоматичної побудови даного засобу, який буде візуалізовувати складні логічні структури. Це дозволить наочно редагувати логічні структури, визначати та аналізувати її властивості, які важко виявити, використовуючи текстове або формальне подання. Часта зміна інформації у базі нечітких знань призводить до необхідності оновлення зображення метаграфа автоматично, оскільки це складно реалізувати вручну [32].

Метаграф – це направлений граф між наборами елементів, де кожен набір є вузлом і ребро в графі являє собою зв'язок між множинами.

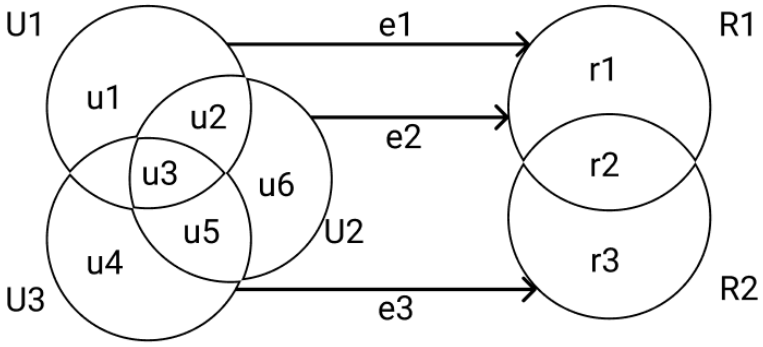


Рис. 1.15 Зображення простого метаграфу

Розглянемо переваги даного методу більш детально:

а) Метаграфи дозволяють виразити залежності:
   1) природно
   2) коротко
   3) без втрати інформації про групування

б) Підтримують математичні оператори для аналізу основних властивостей

в) Меташлях:
   1) визначає досяжність між джерелом і цільовим вузлом

2) являє собою набір ребер

г) Край домінування (edge-dominance):

1) визначає, чи меташлях має зайві ребра

д) Можна використовувати в:

1) системах підтримки прийняття рішень

2) системи управління

3) платформи мережевої конфігурації

В даній роботі пропонується використання візуалізації метаграфа, як нового методу графічного аналізу складних логічних структур. Даний спосіб дозволяє описати та візуалізувати логічні зв'язки взаємозв'язаних множин елементів та їх взаємодію з іншими групами елементів. Крім цього, в подальшій роботі це може стати ефективним інструментом, який дозволить візуалізувати бази нечітких знань, що обумовлює простоту їх сприйняття та можливість наочно виявляти залежності та аномалії.

Висновки

1. Проаналізовано методи побудови бази знань, як основа представлення інформації будь-якої інтелектуальної системи. Із проведеного аналізу, враховуючи переваги кожного з методів, було визначено, що зручною моделю обчислень для ЕОМ паралельної архітектури та найкращим методом представлення бази знань є продукційний підхід.

2. Проведено аналіз можливості подання бази знань за допомогою теорії нечітких множин. Такий підхід дає наближені, але в той же час ефективні способи опису поведінки систем, настільки складних і погано певних, що вони не піддаються точному математичному аналізу.

3. Проведено аналіз існуючих методів верифікації системи – процес доведення того, що БЗ системи не містить внутрішніх помилок, таких як надмірність, суперечливість і неповнота. Головним завданням верифікації БЗ є виявлення аномалій, а саме суперечливих або надлишкових правил, циклів.

4. Запропоновано використання візуалізації метаграфа, як нового методу графічного аналізу складних логічних структур, оскільки даний спосіб дозволяє описати та візуалізувати логічні зв'язки взаємозв'язаних множин елементів та їх взаємодію з іншими групами елементів.

РОЗДІЛ 2.

МОДИФІКОВАНИЙ АЛГОРИТМ УЗГОДЖЕННЯ ТА ПЕРЕВІРКИ НА КОРЕКТНІСТЬ ПРАВИЛ БАЗИ НЕЧІТКИХ ЗНАНЬ

2.1 Побудова БНЗ та визначення впливу дублюючих правил

Механізм формування правил обробляє статистичні дані, тому зручним та найпростішим представленням отриманих результатів можна відобразити у двовимірній таблиці. В даному випадку (див. таблицю 2.1), кожен зразок створює набір логічних правил, представлені у форматі Якщо ... Та ... То. Кожне правило цього типу може бути однозначно ідентифіковано за набором умов Якщо. Тобто в системі не може бути двох правил з однаковим набором умов, так як, припускаючи, що такі правила будуть присутні в системі, вони будуть або дублювати один одного (в разі одного і того ж результату), або перетинатися (в разі різних результатів).

Таблиця. 2.1

Приклад статистичних даних

Зразки	District (%)	Beat (%)	Grid (%)	Latitude (%)	Longitude	KIS	QoS
Зразок 1	0,3	0,096	0,54	0,95739284	0,883	0,093	0,2
Зразок 2	0,45	0,187	0,72	0,82104953	0,316	0,325	0,5
…	…	…	…	…	…	…	…
Зразок N	0,5	0,623	0,93	0,97843972	0,594	0,765	0,92

Для перетворення вибірки даних у формат правил і подальшої роботи над ними використовується поетапний підхід:

а) Перетворення числових значень в терми лінгвістичних змінних;

б) Формування правил;

в) Формування унікального масиву правил.

Для реалізації першого етапу, у роботі [33] запропоновано використовувати кластери отримані на етапі побудови функцій приналежності, для щоб перетворити числові значення у терми лінгвістичних змінних. Тобто кожне числове значення належить деякому кластеру відповідної лінгвістичної змінної. Даний процес має схожість на процес фазифікації, оскільки його основною задачею являється інтерпретація вхідних змінних, які виражаються реальними числами, у більш реалістичну нечітку апроксимацію. Умовно, в даному випадку буде сформована двомірна таблиця, де замість числових значень атрибутів будуть визначенні лінгвістичні змінні (див. таблицю 2.2).

Таблиця. 2.2

Приклад статистичних даних після їх перетворення

Зразки	**District (%)**	**Beat (%)**	**Grid (%)**	**Latitude (%)**	**Longitude**	**KIS**	**QoS**
Зразок 1	middle	low	middle	height	height	low	low
Зразок 2	middle	low	height	height	low	low	middle
…	…	…	...	…	…	…	…
Зразок N	middle	middle	height	height	height	heig ht	height

Переформатування таблиці у вигляд правил:

Якщо ***District (%) – middle*** *Та* ***Beat (%) – low*** *Та* ***Grid – middle*** *Та* ***Latitude – height …*** *То* ***QoS – low***

Якщо ***District (%) – middle*** *Та* ***Beat (%) – low*** *Та* ***Grid – height*** *Та* ***Latitude – height …*** *То* ***QoS – middle***

Якщо ***District (%) – middle*** *Та* ***Beat (%) – middle*** *Та* ***Grid – height*** *Та* ***Latitude – height … …*** *То* ***QoS – height***

Варто зазначити, що при такому підході можуть бути сформовані аномалії у вигляді суперечливих або надлишкових правил. У нечітких моделях, що містяться два або більше ідентичних правил (тобто правила, у яких збігаються умови і висновок), причинами можуть бути:

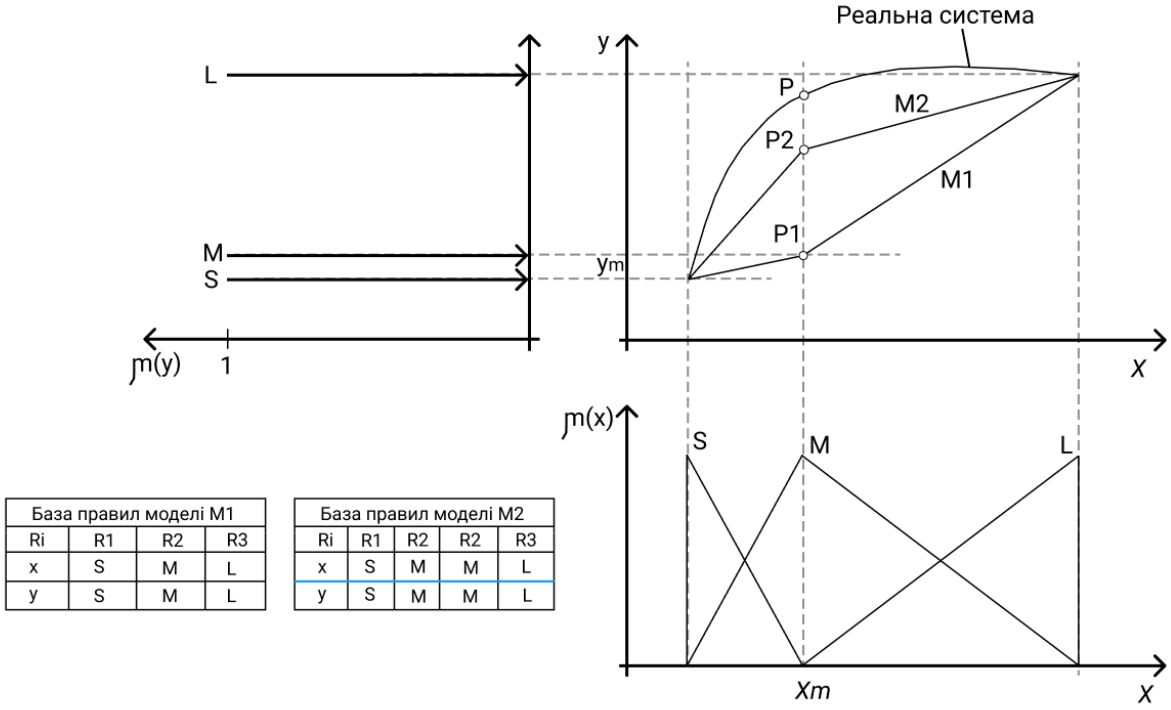
- помилка, допущена при проектуванні бази правил (при великому числі правил);
- в разі самоорганізованої нечіткої моделі, генерація додаткових правил, ідентичних наявним, з метою посилення їх висновків.

Вирішення вищеописаних причин, описано у роботі [34]. У першому, очевидному, випадку надлишкове правило слід виключити. Другий випадок вимагає роз'яснення (див. рисунок 2.1). Поверхня моделі M1 в точці$x = x_m$ значно відрізняється від поверхні реальної системи (точки P1і P). Настільки значна помилка виникла внаслідок неправильного вибору параметра ym вихідної функції приналежності $\mu_M(y)$. Самонавчальна модель в подібній ситуації може сформувати додаткове правило, яке збігається з R2. Два однакових правила R2 можна замінити одним правилом R2 *, висновок якого має вигляд логічної суми:

$$(\text{Якщо } (x = M) \text{ То } (y = M)) \cup (\text{Якщо } (x = M) \text{ То } (y = M)) =$$
$$= \text{Якщо } (x = M) \text{ То } (y = M \cup M)$$

При виконанні логічної суми на основі оператора MAX буде отримана множина

$$M* = M \cup M = M$$



База правил моделі M1			
Ri	R1	R2	R3
x	S	M	L
y	S	M	L

База правил моделі M2				
Ri	R1	R2	R2	R3
x	S	M	M	L
y	S	M	M	L

Рис. 2.1 Порівняння нечітких моделей M1 і M2, відповідно без надлишковості і з надлишковістю бази правил

У разі використання інших операторів, наприклад SUM (сумовування функцій приналежності), результуюча множина $M *$ матиме вигляд $M \cup M \neq M$, що призводить до посилення одержуваного висновку і зменшення помилки моделі (точки P і P2 на рисунку 2.1). Таким чином, кілька співпадаючих правил можна замінити одним правилом, висновок якого відповідним чином посилено.

У роботі [33] запропоновано використовувати підхід очищення правил з метою проведення додаткового аналізу встановлених правил. Механізм очищення правил можна відобразити таким чином:

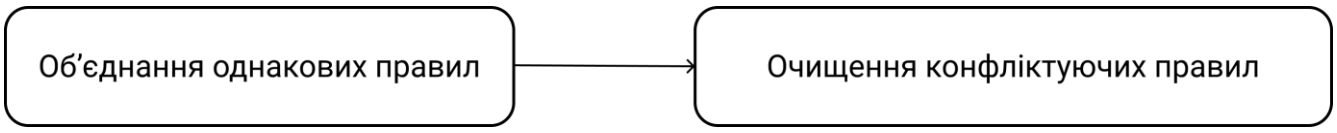


Рис. 2.2 Механізм очищення правил

Формування унікального масиву правил бази нечітких знань починається із об’єднання дублюючих правил - це завдання знаходження та об’єднання, враховуючи важливість впливу правила на роботу системи, в масиві об’єктів груп з однаковим набором Якщо … Та … То.

При проходженні циклу по всьому масиву правил, кожне правило додається у проміжну вибірку із початковою вагою 1.

$$Weight_r = 1, \quad (2.1)$$

де Weight r – вага правила.

Кожне дублююче правило, яке буде зустрічатися у вибірці збільшує вагу дубліката на коефіцієнт росту, який зазвичай дорівнює k=1 і відповідно не вноситься у результуючий масив [33].

$$Weight_{r_{i+1}} = Weight_{r+k}, \quad (2.2)$$

де $Weight_{r_i} -$ вага правила на i-тому році до збільшення ваги, $k -$ коефіцієнт росту ваги правила.

Кожне конфліктуюче правило в даному наборі вводиться в проміжну вибірку як самостійне правило і обробляється аналогічно.

Блок-схема роботи даного механізму зображено на рисунку 2.3, в результаті чого буде отримано масив унікальних правил із зазначенням їхнього впливу на роботу системи або частоту знаходження правила в проміжному наборі. Слід враховувати, що в отриманому проміжному наборі правил існують конфліктуючі правила. Для їх усунення проводиться другий етап, на якому набір правил очищається від конфлікту.

Рис. 2.3 Блок-схема алгоритму очищення дублюючих правил

2.2 Алгоритм очищення конфліктуючих правил

База правил називається несуперечливою (узгодженою), якщо вона не містить несумісні правила, тобто правила, що мають однакові умови, але різні висновки. На рисунку 2.4 наведено приклад моделі, що містить несумісні правила.

Правила R1 і R2 в моделі на (див. рисунок 2.4) мають однакові умови ($x = \text{Малий}$), але різні висновки ($y = \text{Малий}$) і ($y = \text{Дуже великий}$). Оскільки висновки правил висловлюють діаметрально протилежні поняття («малий» і «дуже великий»), то в даній ситуації можна говорити про «сильну» сумісність правил.

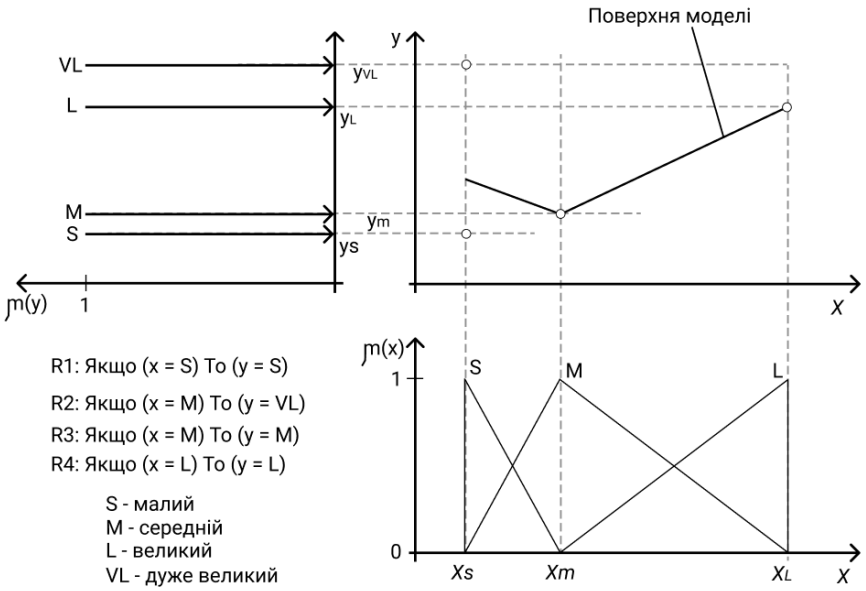


Рис. 2.4 База правил із «сильною» несумісністю правил R1 і R2, а також поверхня відображення X – Y моделі

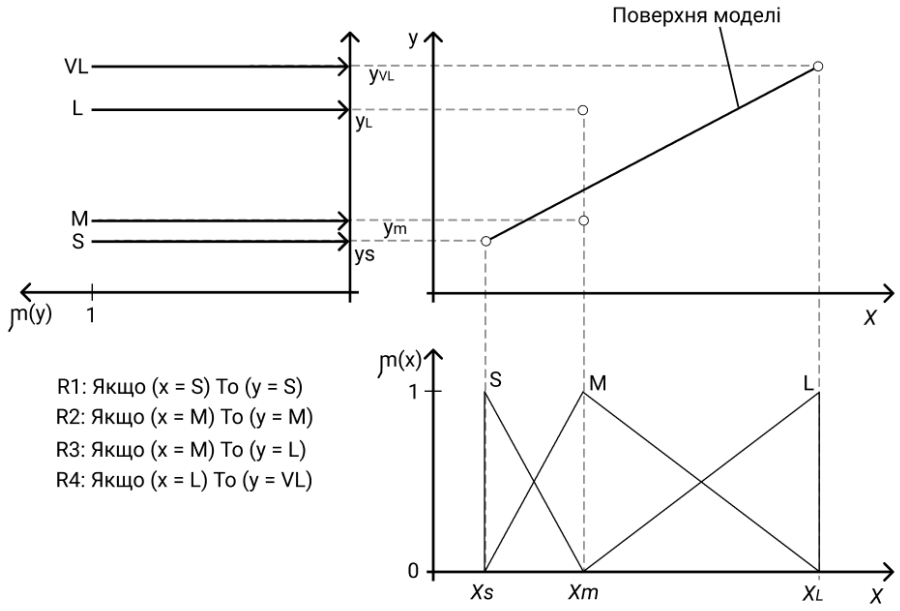


Рис. 2.5 База правил із «слабою» несумісністю правил R1 і R2, а також поверхня відображення X – Y моделі

У моделі на рисунку 2.5 також є несумісні правила. Однак в даному випадку ми можемо говорити, що їх несумісність є «слабкою», оскільки модальні значення несумісних висновків M і L розташовані близько один до одного.

Таким чином, ступінь несумісності правил може бути вище або нижче, в залежності від того, як розташовані один щодо одного модальні значення їх

висновків. Причиною існування несумісних правил в базі нечітких правил може бути викликана, по-перше, помилкою, допущеної в ході формування правил, особливо в разі великого їх числа. Іншою причиною може стати неоднозначність модельованої системи, тобто ситуація, коли вимірювання вхідних і вихідних даних системи не є однозначними, і одному вхідному стану x * можуть відповідати різні вихідні стани (див. рисунок 2.6.).

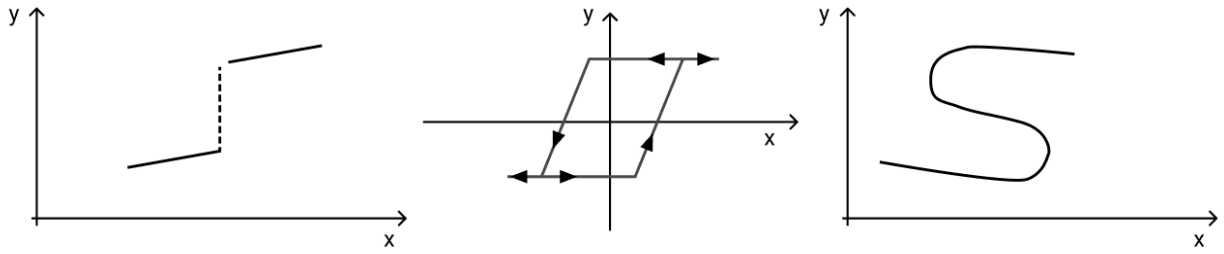


Рис. 2.6 Приклад неоднозначності систем, що приводять до правил з однаковими умовами, але різними висновками

Таким чином, «несумісні» правила не є такими насправді, оскільки вони відображають вірну інформацію про систему. Для того щоб в разі неоднозначності системи (наприклад, при неоднозначній залежності деякої фізичної величини від іншої величини при циклічній зміні останньої) уникнути наявності несумісних правил, необхідно усунути її неоднозначність, що виникає, якщо модель має занадто мало вхідних параметрів. Так, модель гістерезиса (див. рисунок 2.6) стає однозначною при поданні її в тривимірній системі координат з вхідними параметрами $x\ (k)$, $y\ (k\ -\ 1)$ і вихідним параметром $y\ (k)$ [34].

Найпростішим алгоритмом у вирішенні питання конфліктуючих правил було запропоновано у роботі [33]. З набору суперечливих правил (їх може бути більше двох) вибирається правило з найбільшою вагою, це правило залишатиметься в наборі результатів, але його вага стає рівним середньому арифметичному всіх ваг суперечливих правил.

$$Weight_{r_{conf}} = \frac{\sum_{k=1}^{n} Weight_{r_k}}{n}, \qquad (2.3)$$

де n – кількість конфліктуючих правил, $Weight_{r_k}$ - вага k-го конфліктуючого правила, $Weight_{r_{conf}}$ - результуюча вага правила, що замінює всі конфліктуючі правила.

Для швидкого та зручного обчислення та усунення впливу початкової ініціалізації ваги правил та коефіцієнт росту ваги можна додатково провести нормалізацію ваг правил в межах від 0 до 1. Детальніше механізм очищення конфліктуючих правил зображено на рисунку 2.7.

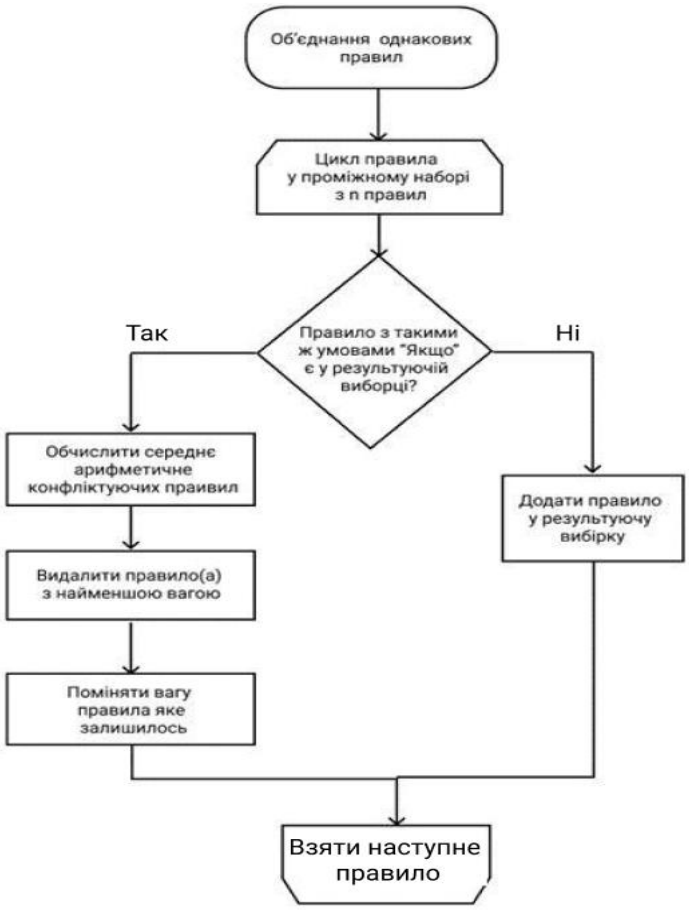


Рис. 2.7 Блок-схема алгоритму очищення конфліктуючих правил

В результаті обробки набору правил було отримано новий набір, у якому відсутні дублікати та суперечливі правила. Цей набір правил можна використовувати в системі підтримки прийняття рішення.

Така база нечітких знань забезпечує можливість досягнення необхідної точності нечіткої моделі. Одночасно з тим, була зменшена обчислювальна складність і модель стала більш «прозорою» (інтуїтивно–зрозумілою),

оскільки кількість правил, що міститься в базі, стала якомога меншою. У роботі [34] вказані властивості нечітких моделей – точність і число правил, які являються взаємно-виключаючі. При великій кількості правил досягнення високої точності моделі потенційно є більш простою задачею, а зменшення кількості правил у системі підтримки прийняття рішення у загальному випадку знижує її точність. Тому при виборі числа правил необхідно враховувати наступні рекомендації:

- число правил збільшується при ущільненні сітки, використовуваної для розбиття простору X входів моделі;
- щільність використовуваної для розбиття сітки слід збільшувати в разі більш рельєфної поверхні відображення X → Y моделі;
- при незмінній щільності сітки (незмінному числі правил) точність моделі може бути підвищена шляхом правильного розміщення опорних точок її поверхні, що задаються правилами.

2.3 Метаграф, як спосіб представлення бази нечітких знань

У результаті обробки набору правил отримано новий набір, у якому відсутні дублюючі та конфліктуючі правила. В даній роботі пропонується використовувати апарат теорії метаграфа для візуалізації бази нечітких знань. Оскільки даний підхід дозволить наочно редагувати логічні структури, визначати та аналізувати її властивості, які важко виявити, використовуючи текстове або формальне подання.

Запропонований об’єднаний метод побудови бази нечітких логічних правил з перевіркою їх на коректність за допомогою теорії метаграфа має такі переваги:

а) спрощена процедура формування правил;
б) зменшена обчислювальна складність під час формування правил;
в) якісна перевірка дозволяє вилучити конфлікти, що підвищує достовірність отриманих правил.

У пропонованому підході, метаграфи використовуються для перевірки сформованих метаописань нечітких баз знань і очищення їх від аномальних даних. Проектування виробничого процесу в реальності є нечітким і тому вимагає використання спеціального математичного апарату для прийняття рішень, близьких до мислення інженера. У цьому випадку необхідно мати експертні правила у формі тверджень Якщо … То …. Для візуалізації такого роду нечітких правил використовуються метаграфи.

Формально метаграф може бути представлений наступним чином [38]:

$V = \{v_r | r = \overline{1N_v}\}$ – множина вершин;

$M = \{m_q | q = \overline{1N_m}\}$ – множина метавершин, де

$m_q = \{v_r | v_r \in V, r = \overline{1N_{m_q}}\}$ – метавершина, яка знаходиться в множині вершин;

$E = \{e_h | h = \overline{1N_E}\}$ – множина ребер;

$mv \in (V \cup M)$ – нода метаграфу.

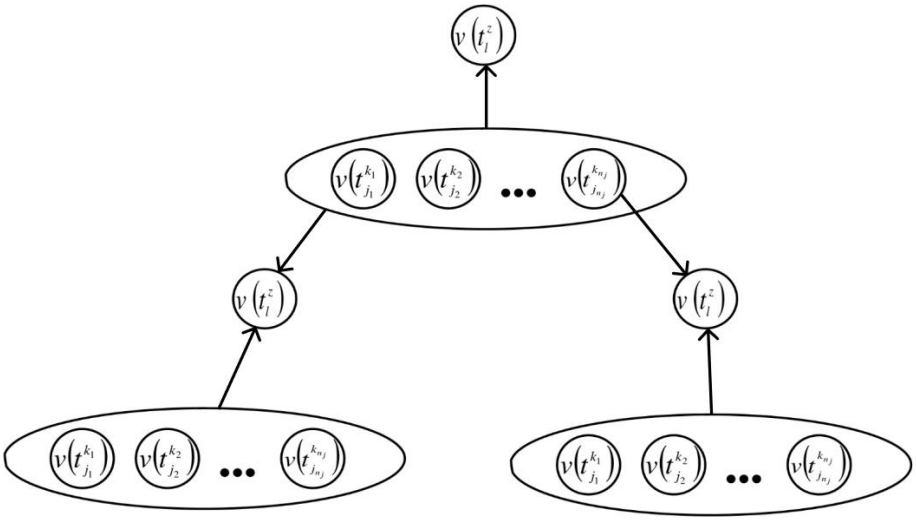


Рис. 2.8 Приклад метаграфу

$V_g = \{v(t_{j1}^{k1}),\ v(t_{j2}^{k2}), \dots, v\left(t_{jn_j}^{kn_j}\right), v(t_l^z)\}$ – вершини, що відповідають термам в лівій частині правил і результуючому терміну;

$M_g = \{\, m_g\}$ – множина метавершин;

$m_g = \{v(t_{j1}^{k1}), v(t_{j2}^{k2}), \dots, v\left(t_{jn_j}^{kn_j}\right)\}$ – метавершини що включають в собі вершини, відповідні термам з лівої частини правила;

$E_g = \{e_g\}$ – множина ребер;

$e_g = \{m_g, v(t_l^z)\}$ – ребро з'єднує метавершину з вершиною, яка відповідає результуючому члену.

За домогою метаграфу можна описувати нечітку базу знань. Правила формату Якщо…То… описуються з боку метаграфа:

$$(P_l^z)_j = \text{Якщо } \left(X_{j1} = t_{j1}^{k1}\right)\text{Та } \left(X_{j2} = t_{j2}^{k2}\right)\text{Та} \dots \text{Та } \left(X_{jn_j} = t_{jn_j}^{kn_j}\right) \text{ То } (X_l = t_l^z)$$

Аномалії, які можна виправити за допомогою метаграфа:

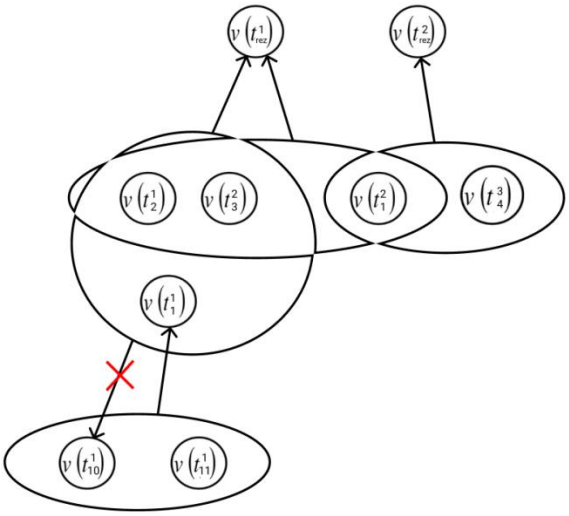


Рис. 2.9 Метаграф не повинен містити циклів

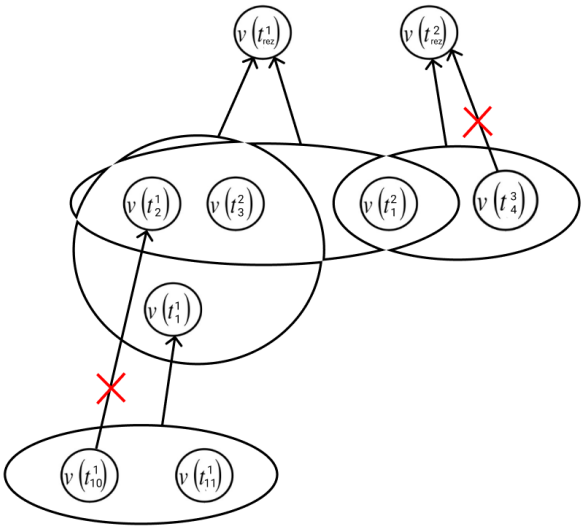


Рис. 2.10 Метаграф не містить ребер, які виходять з вершини, але існують з метавершини

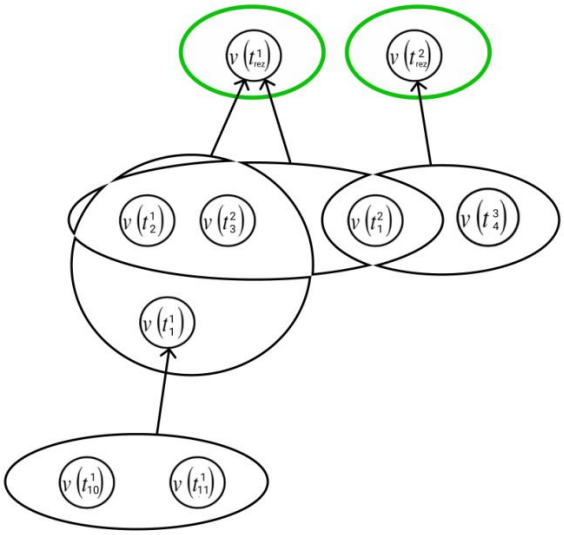


Рис. 2.11 Метаграф містить вершини, які не входять ні в одну з метавершин і повинна бути збурена хоча б один раз як результуюча

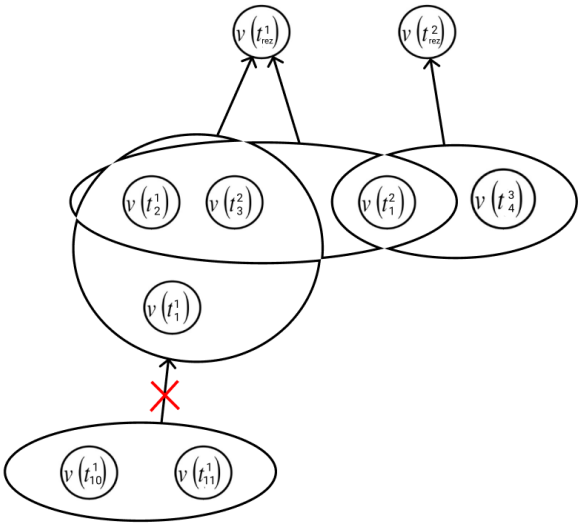


Рис. 2.12 Метаграф не містить ребер, які входять в метавершину

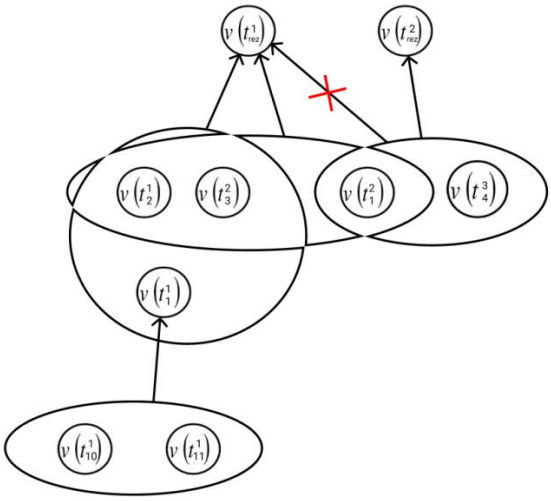


Рис. 2.13 У метаграфі тільки одне ребро повинне вийти з метавершини

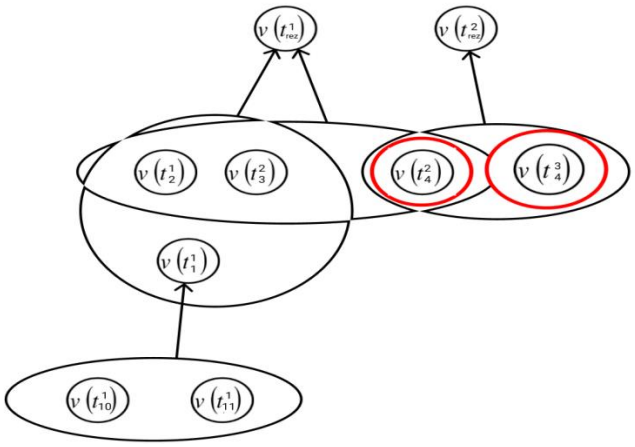


Рис. 2.14 Метавершина не може містити дві або більше вершин, які відповідають термам однієї і тієї ж лінгвістичної змінної

Представленний метод візуалізації метаграфа дозволяє використовувати методи графічного представлення, структуризації та аналізу даних в задачах, що використовують метаграфи. Цей апарат дозволяє виявити аномалії, такі як надмірність, суперечливість і неповнота в моделі предметної області, що важко виявити у текстовому або формальному представленні. Часта зміна інформації у базі нечітких знань призводить до необхідності оновлення зображення метаграфа автоматично, оскільки це складно реалізувати вручну [32]. В результаті цього, актуальність автоматичної побудови для представлення метаграфа відіграє важливу роль у системах підтримки прийняття рішенн.

Висновки

1. Запропоновано використання алгоритму для формування унікального масиву правил бази нечітких знань. Завдання якого заключається в знаходженні та об’єднанні, враховуючи важливість впливу правила на роботу системи, в масиві об’єктів груп з однаковим набором продукцій, а також усунення несумісних правил, що мають однакові умови, але різні висновки.

2. Розроблено критерії щодо візуалізації апарату теорії метаграфу та критерії, які будуть використовуватися для перевірки сформованих метаописань нечітких баз знань та їх коректність.

РОЗДІЛ 3.

ВІЗУАЛІЗАЦІЯ БАЗИ НЕЧІТКИХ ЗНАНЬ ЗА ДОПОМОГОЮ АПАРАТУ ТЕОРІЇ МЕТАГРАФА

3.1 Програмне рішення візуалізації нечітких логічних правил у вигляді метаграфа

Для розробки програмного забезпечення для даного методу, насамперед необхідно розглянути архітектуру, яка виконуватиме функції клієнта так і функції сервера, щоб сприяти обміну інформацією між ними. У сучасному комп'ютерному світі клієнт-серверна система стала популярною, тому що вона використовується практично кожен день для різних додатків. Деякі з стандартизованих протоколів, які клієнт і сервер використовують для зв'язку з самими собою, включають в себе: протокол передачі файлів (FTP), простий протокол передачі пошти (SMTP) і протокол передачі гіпертексту (HTTP).

Таким чином, клієнт-серверну систему можна визначити як програмну архітектуру для об’єднаного методу побудови бази нечітких логічних правил з перевіркою їх на коректність за допомогою теорії метаграфа. Клієнт-серверна архітектура (див. рисунок 3.1) забезпечує міжпроцесну комунікацію, оскільки вона включає в себе обмін даними як від клієнта, так і від сервера, в результаті чого кожен з них виконує різні функції.

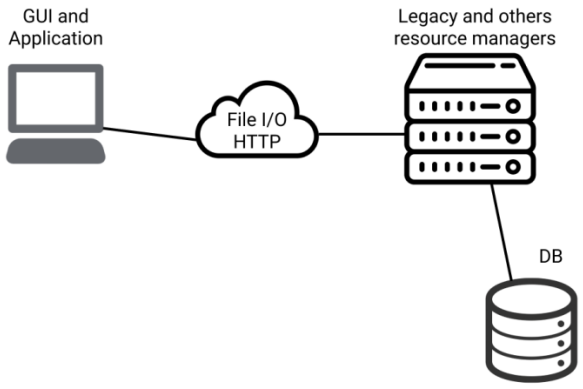


Рис. 3.1 Клієнт – серверна архітетктура

Запропонована архітектура складається з трьох компонентів розподілених в два шари: клієнт (замовник послуг) і сервер (постачальник послуг).

Ці три компоненти є:

1. Користувальницький інтерфейс (наприклад, служби керування сеансами, введенням тексту, відображення побудованого графу у нашому випадку)
2. Керування процесами (наприклад, розробка процесів, впровадження процесів, моніторинг процесів та послуги з управління ресурсами процесів)
3. Керування базами даних (наприклад, даними і файловими сервісами)

Наступним завданням після визначення архітектури, потрібно визначитись із технологіями, за допомогою яких буде відбуватися реалізація запропонованого підходу. Для втілення моделюючого комплексу вибір зупинився на таких технологіях:

а) NodeJS, який використовує ExpressJS – фреймворк веб-програми для архітектури Model-View-Controller;
б) D3.js – бібліотека для відображення графіків з використанням векторної графіки;
в) Jade – ефективний шаблонний механізм;
г) Stylus – препроцесор CSS;
д) JavaScript – мультипарадигменна мова програмування;
е) Зовнішні бібліотеки JavaScript: JQuery, JQueryUI, Bootstrap 3, FontAwesome.

Результати опрацювання механізму очищення НЛП

Інтелектуальна система, яка базується на правилах бази нечітких знань міститься інформація про залежність між вхідними та вихідними змінними у вигляді нечітких логічних правил. Алгоритми очищення дублюючих та конфліктуючих правил буде здійснюватися на крос-платформному середовищі Node.js, оскільки всі обчислення будуть відбуватися на серверній стороні. На етапі очищення, вхідні правила повинні передаватися у JSON форматі. За рахунок своєї лаконічності в порівнянні з XML та CSV формат JSON може бути більш підходящим для серіалізації складних структур.

Нечіткі логічні правила подаються як колекція пар двомірного масиву об’єктів у форматі «ім’я: значення» (див. таблицю 3.1). На першому етапі механізму очищення правил відбувається видалення дублікатів і додавання нової властивості *weight* (вага), яка відображає частоту знаходження правила в проміжному наборі. Результат виконаного алгоритму відображено в таблиці 3.1.

Таблиця. 3.1

Масив об’єктів до та після видалення дублікатів

{“rules ”: [{
qos: 'low',
district: 'low',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
}, … {
qos: 'low',
district: 'middle',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
}, … {
qos: 'low',
district: 'middle',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
} …]}
	{“rules ”: [{
	qos: 'low',
	district: 'low',
	beat: 'low',
	grid: 'low',
	latitude: 'low',
	longitude: 'middle',
	weight: '1'
	}, …{
	qos: 'low',
	district: 'middle',
	beat: 'low',
	grid: 'low',
	latitude: 'low',
	longitude: 'middle',
	weight: '2'
	}, …{
	qos: 'high',
	district: 'middle',
	beat: 'high',
	grid: 'low',
	latitude: 'low',
	longitude: 'high',
	weight: '1'
	} …]}

Наступнй етап – очищення колекції об’єктів JSON файлу від конфліктуючих правил. Проводиться перевірка лівої частини правила (Якщо…) на дублікат. Якщо існують два або більше одинакових частин правила у яких значення результату суперечать одне одному (див. таблицю 3.2), то вибирається той об’єкт, у якого значення weight має більше значення і воно заміняється на середнє арифметичному всіх ваг конфліктуючих правил. Кінцевим результатом механізму очищення правил буде унікальний масив об’єктів без аномалій із зазначеною вагою впливу правила на роботу системи підтримки прийняття рішення (див. таблицю 3.2).

Таблиця. 3.2

Масив об’єктів до та після видалення конфліктів

{“rules ”: [{
qos: 'low',
district: 'low',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
weight: '1'
}, … {
qos: 'low',
district: 'middle',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
weight: '2'
}, … {
qos: 'middle',
district: 'middle',
beat: 'low',
grid: 'low',
latitude: 'low',
longitude: 'middle',
weight: '4'
} …]}
	{“rules ”: [{
	qos: 'low',
	district: 'low',
	beat: 'low',
	grid: 'low',
	latitude: 'low',
	longitude: 'middle',
	weight: '1'
	}, …{
	qos: 'middle',
	district: 'middle',
	beat: 'low',
	grid: 'low',
	latitude: 'low',
	longitude: 'middle',
	weight: '3'
	}, …{
	qos: 'low',
	district: 'low',
	beat: 'middle',
	grid: 'low',
	latitude: 'low',
	longitude: 'middle',
	weight: '1'
	} …]}

Клієнтська сторона

Клієнтська частина системи побудована за допомогою D3.js. D3 – це бібліотека JavaScript для візуалізації даних з використанням веб-стандартів. D3 допоможе структуризувати дані за допомогою SVG, Canvas і HTML. Дана бібліотека поєднує в собі потужні методи візуалізації та взаємодії з керованими даними підходами до маніпулювання DOM, надаючи всі можливості сучасних браузерів і свободу розробки правильного візуального інтерфейсу для колекції різних даних. D3 має більше тридцяти модулів і тисячу методів, тому дана бібліотека буде використовуватися для візуалізації метаграфу.

Перед початком виконанням реалізації побудови метаграфу, JSON файл із очищеними правилами потрібно провести додаткову автоматичну процедуру форматування правил по принципу *nodes* та *links* (див. таблицю 3.3)

Таблиця 3.3

Формат правил по принципу nodes та links

{“nodes”: [{“name”: “*District*”, “group”: “low”}, {“name”: “*Beat*”, “group”: “middle”}, {“name”: “*Latitude*”, “group”: “middle”}, … {“name”: “*Grid*”, “group”: “high”},]}	{“links”: [{“source”: *District*, “target”: “low”}, {“source”: *Beat*, “target”: “middle”}, {“source”: “*Latitude*”, “target ”: “middle”}, … {“source”: *Grid*, “target”:”high”},]}

При автоматичному форматуванню *links* обов’язково визначає “source” і “target”, що вказають на напрям залежності об’єктів. На виході системи отримаємо готовий метаграф, за допомогою якого можна наочно редагувати логічні структури, визначати та аналізувати його властивості, які було б важко виявити, використовуючи текстове подання (див. рисунок 3.2).

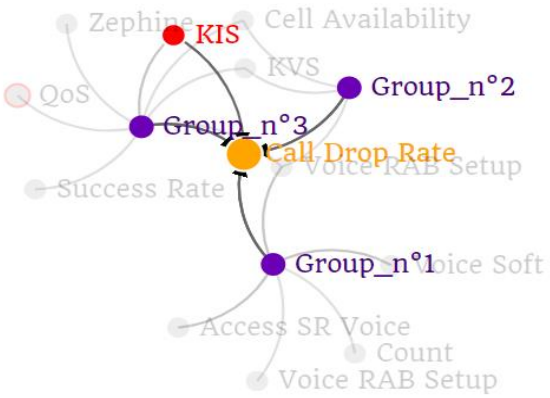


Рис. 3.2 Наявний вигляд залежності вершин від інших вершин метаграфа

3.2 Порівняльний аналіз запропонованого рішення

Порівняльні характеристики алгоритмів візуалізіації графів були проведені у роботах [35, 36, 38]. Експерименти проводились для різних алгоритмів, які генерують прямолінійні креслення загальних великих графів, максимальна кількість вершин сягає до ста тисяч. Досліджувалсь деякі з методів, які засновані на силових або алгебраїчних підходах з точки зору часу виконання і якості малювання на великій різноманітності штучних і реальних графів. Експерименти показують (ТАБЛИЦЯ), що існують значні відмінності в якості малюнка і часу виконання в залежності від класів тестованих графів і алгоритмів, а саме: алгоритм сітки (GVA), метод GRIP, швидкий багатомасштабний метод (FMS), швидкий багатополюсний та багаторівневий метод (FM^3), алгебраїчний багатосітковий метод (ACE) та метод вкладення високої розмірності (HDE).

Таблиця. 3.4

Порівняльні характеристики алгоритмів візуалізації графів

Алгоритм	Складність алгоритму	Максимальна кількість вершин	Можливість зображення метаграфу
GVA	O($\|V\|^2$)	180	-
GRIP	O(|V|(log diam$(G)^2$))	16000	-
Алгоритм	Складність алгоритму	Максимальна кількість вершин	Можливість зображення метаграфу
FMS	O(|V||E|)	16000	-
FM^3	O(|V | log |V | + |E|)	5789	-
ACE	Не визначалась	≤ 10, 000	-
HDE	O(|V | + |E|)	16000	-

Із проведеного вище аналізу, можна зробити висновок, що на сьогоднішній день не існує методу візуалізації метаграфу. Тому виникає актуальність створення системи, яка буде візуалізовувати складні логічні структури у вигляді БНЗ. У даному випадку це дозволить наочно виявити логічні зв’язки між лінгвістичними змінними та аналізувати їхні властивості, які важко виявити, використовуючи формальне подання таких структур.

У роботі [37] були проведені порівняння метаграфу і базових моделей подання даних в двох варіантах, щоб зрозуміти, наскільки добре і наскільки швидко моделі можуть адаптуватися до нових небачених тестових графів. Перевірявся параметр конвергенції, в якому навчали моделі збіжності, і параметр швидкої адаптації, в якому адаптували моделі після виконання тільки 5 оновлень градієнта. Також вимірювалась продуктивність, обчислюючи AUC передбачення зв'язку.

Для налаштування конвергенції виявлено, що метаграф досягає найвищого середнього AUC з відносним поліпшенням на 4,8% в порівнянні з

підходом MAML і поліпшенням на 5,3% в порівнянні з вихідним рівнем Finetune. Продуктивність метаграфа вище, коли використовується тільки 10 відсотків ребер, що демонструє потенціал підходу при роботі з дійсно обмеженими даними.

В умовах швидкої адаптації виявлено, що метаграф перевершує всі базові лінії, крім однієї, із середнім відносним поліпшенням на 9,4% в порівнянні з MAML і на 8% в порівнянні з тонко налаштованим базовим рівнем. Іншими словами, метаграф може не тільки навчатись з обмежених даних, але і швидко приймати нові дані, використовуючи лише невелику кількість градієнтних кроків.

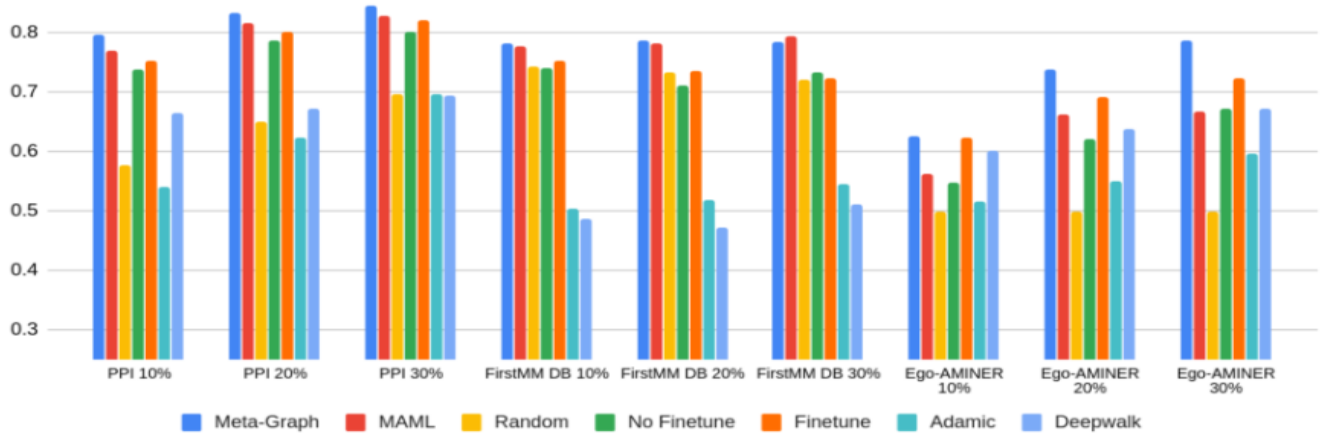


Рис. 3.3 Результати AUC конвергенції для різних фракцій навчальних ребер в різних наборах даних.

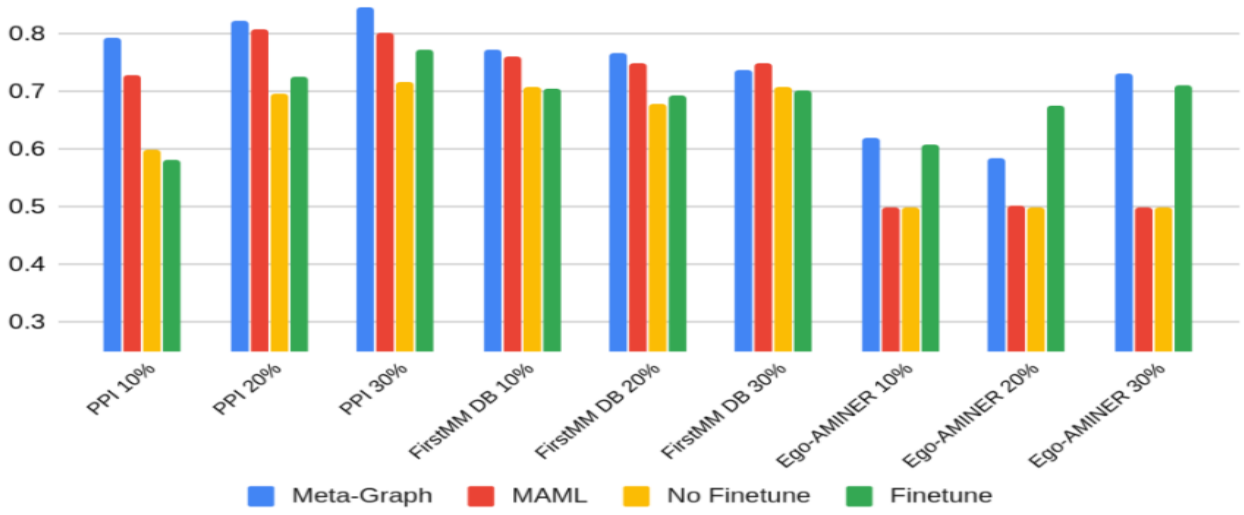


Рис. 3.4 Результати AUC отримані п'ятьма оновленнями градієнта для різних фракцій навчальних ребер в різних наборах даних.

Запропонований метод у даній роботі був протестований на випадкових метаграфах з числом вершин до 60 і до 120 ребер. Випробування проводилися на ПК з процесором Intel(R) Core (TM) i7-8565U CPU @ 1.80 GHz, 16 Гб RAM. Було виявлено, що в залежності від початкового розташування вузлів результуюче зображення було різним. Час, необхідний для отримання задовільного зображення, залежить від кількості вершин і кількості ребер. Цей час скорочується з поліпшенням початкового розташування вузлів. На рисунку 3.5 показано середній час, необхідний для візуалізації випадкових метаграфів із різною кількістю вершин і ребер.

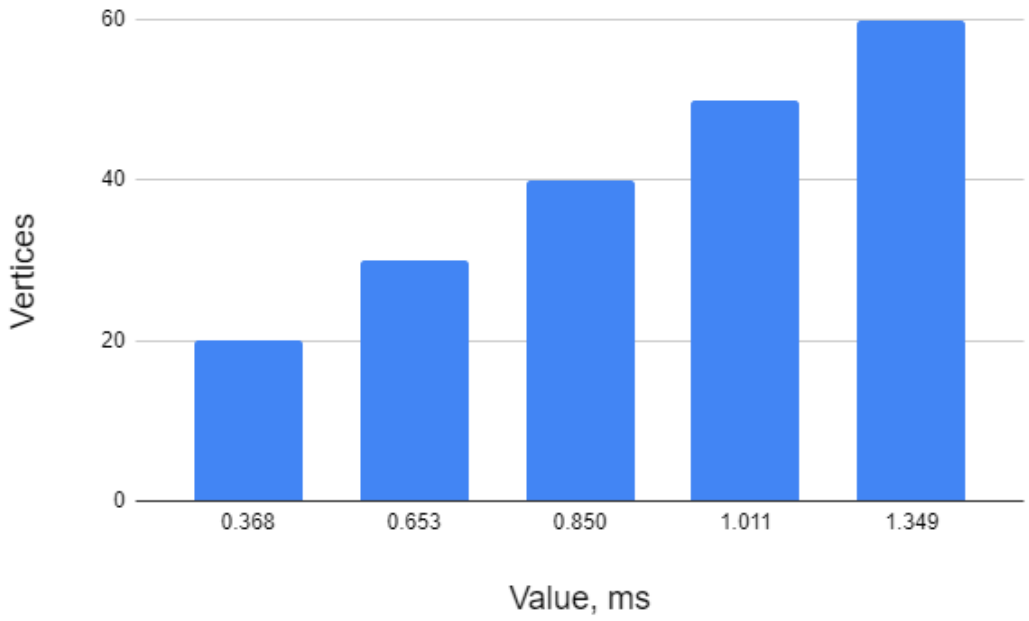


Рис. 3.5 Середній час візуалізації метаграфів

Результати візуалізації випадкових метаграфів представлені на рисунках 3.6 - 3.8.

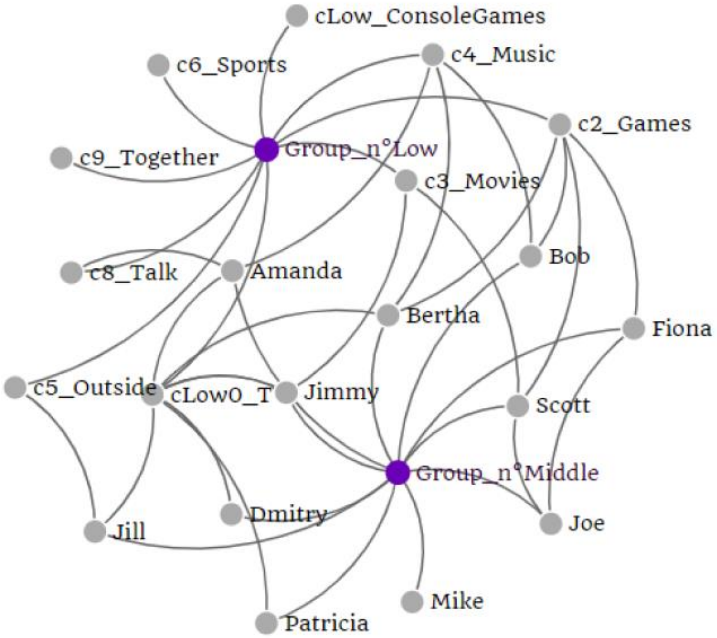


Рис. 3.6 Метаграф із 20 вершинами та 40 ребрами із виконанням по часу 0.368 мс

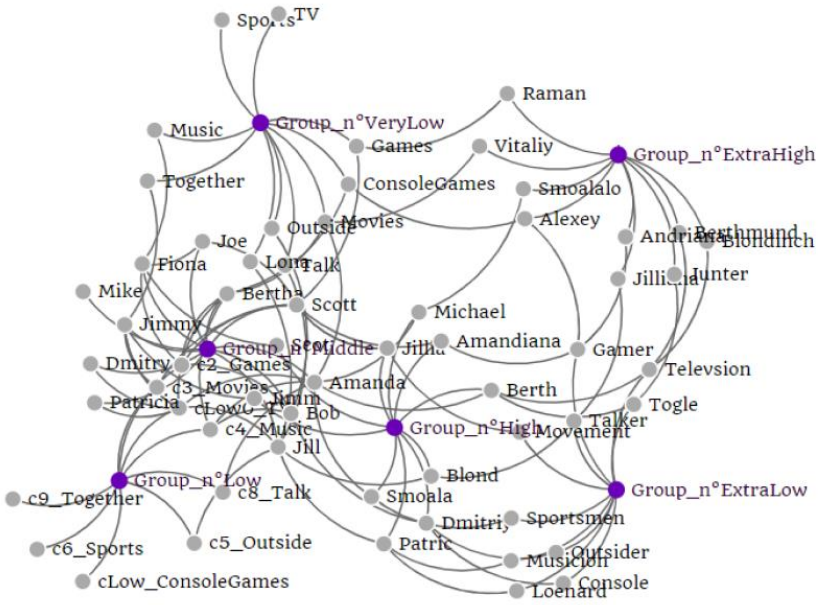


Рис. 3.7 Метаграф із 40 вершинами та 80 ребрами із виконанням по часу 0.850 мс

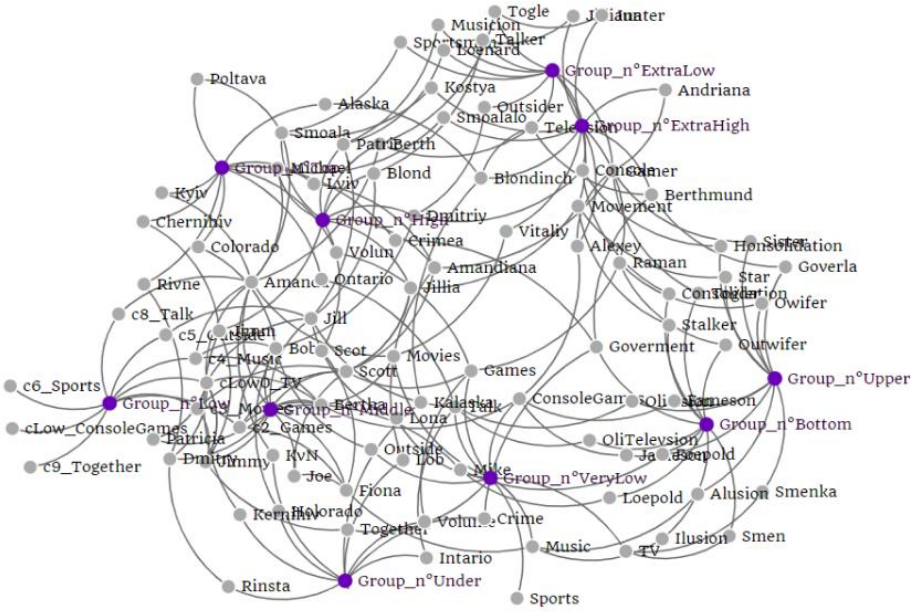


Рис. 3.8 Метаграф із 60 вершинами та 120 ребрами із виконанням по часу 1.349 мс

Метаграф можна вважати, як вирішення проблеми прогнозування середньо- та велико-розмірних зв'язків, в якому мета полягає в точному навчанні моделей, які швидко адаптуються до нових даних. Представлений метод візуалізації в порівнянні з іншими алгоритмами структуризації інформації можна вважати новим продуктивним методом графічного аналізу складних логічних структур, оскільки він дозволяє підвищити якість та достовірність отриманих нечітких логічних правил, що знаходяться у базі нечітких знань. Крім того, представлення бізнес-процесів і робочих процесів у вигляді метаграфів може використовуватися для обміну знаннями про структуру процесів і планування через організаційні межі таким чином, щоб підтримувати не тільки корисну візуалізацію і абстракцію процесів, але і структурний аналіз цих процесів. У подальшому перегляді запропонованого методу планується робота над чітким відображенням метавершин, новою моделлю визначення множинних перетинів метавершин та адаптація методу до навчання.

Висновки

1. Для практичного використання запропонованого методу реалізовано інформаційну систему, обрано популярні підходи та мережні архітектури щодо побудови сучасних систем.

2. Реалізовано запропонований алгоритм формування унікального масиву правил бази нечітких знань. Де відбувається знаходження та об’єднання, враховуючи важливість впливу правила на роботу системи, в масиві об’єктів груп з однаковим набором продукцій, а також усунення конфліктуючих правил, що мають однакові умови, але різні висновки.

3. Проведено аналіз розробленого механізму, який показав що час, необхідний для отримання задовільного зображення, залежить від кількості вершин і кількості ребер. Середній час візуалізації метаграфу із 60-ма вершинами та 120-ма ребрами не перевищує 1,5мс.

4. Виявлено, що в умовах швидкої адаптації метаграф перевершує всі базові моделі подання даних, крім однієї, із середнім відносним поліпшенням на 9,4%. А також при налаштуванні конвергенції виявлено, що метаграф досягає найвищого середнього AUC з відносним поліпшенням на 4,8%.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Проаналізовано методи побудови бази знань, як основа представлення інформації будь-якої інтелектуальної системи. Із проведеного аналізу, враховуючи переваги кожного з методів, було визначено, що зручною моделю обчислень для ЕОМ паралельної архітектури та найкращим методом представлення бази знань є продукційний підхід.
2. Проведено аналіз можливості подання бази знань за допомогою теорії нечітких множин. Такий підхід дає наближені, але в той же час ефективні способи опису поведінки систем, настільки складних і погано певних, що вони не піддаються точному математичному аналізу.
3. Запропоновано використання візуалізації метаграфа, як нового методу графічного аналізу складних логічних структур, оскільки даний спосіб дозволяє описати та візуалізувати логічні зв'язки взаємозв'язаних множин елементів та їх взаємодію з іншими групами елементів.
4. Розроблено критерії щодо візуалізації апарату теорії метаграфу та критерії, які будуть використовуватися для перевірки сформованих метаописань нечітких баз знань та їх коректність.
5. Реалізовано запропонований алгоритм формування унікального масиву правил бази нечітких знань. Де відбувається знаходження та об’єднання, враховуючи важливість впливу правила на роботу системи, в масиві об’єктів груп з однаковим набором продукцій, а також усунення конфліктуючих правил, що мають однакові умови, але різні висновки.
6. Виявлено, що в умовах швидкої адаптації метаграф перевершує всі базові моделі подання даних, крім однієї, із середнім відносним поліпшенням на 9,4%. А також при налаштуванні конвергенції виявлено, що метаграф досягає найвищого середнього AUC з відносним поліпшенням на 4,8%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Козлов А. Н. Интеллектуальные информационные системы / А. Н. Козлов. – Пермь, 2013.
2. Алтунин А. Е. модели и алгоритмы принятия решений в нечетких условиях / А. Е. Алтунин, М. В. Семухин. – 2. Тюменский государственный университет, 2000.
3. Laura Caponetti «Fuzzy Logic for Image Processing» SpringerBriefs in Electrical and Computer Engineering.
4. Zadeh, L.A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst. Man Cybern. SMC-3 28–44, 1973.
5. Zimmermann, H.J.: Fuzzy Set Theory and its Applications. Kluwer, Norwell, 1992.
6. Mamdani, E.H., Assillan, S.: An experiment in linguistic synthesis with a fuzzy logic controller.Int. J. Man-Mach. Stud. 7(1), 1–13, 1975.
7. Mouzouris, G.C., Mendel, M.J.: Dynamic non-singleton fuzzy logic systems for nonlinear modeling. IEEE Trans. Fuzzy Syst. 5(2), 199–208, 1997.
8. Brown, M., Harris, C.J.: Neurofuzzy Adaptive Modelling and Control. Prentice Hall, Hemel Hempstead, 1994.
9. Jang, J.-S.R., Sun, C.-T.: Neuro-fuzzy modelling and control. Proc. IEEE 83, 378–406, 1995.
10. Lee, C.C.: Fuzzy logic in control systems: Fuzzy logic controller - part I and II. IEEE Trans.Syst., Man Cybern. 20(2), 404–435, 1990.
11. Wang, L.: Adaptive Fuzzy Systems and Control. Prentice Hall, Englewood Clis, 1994.
12. Preece A.D., Shinghal R., Batarekh A. Verifying expert systems: a logical framework and a practical tool // Expert systems with applications. 1992. Vol. 5. P. 421–436.
13. Проскуряков Д.П. Поиск противоречий с помощью стратегии управления продукциями на основе онтологии предметной области //

Труды XIX Байкальской Всероссийской конференции «Информационные и математические технологии в науке и управлении». Ч. III. Иркутск: ИСЭМ СО РАН, 2014. С. 166–170

14. Долинина О.Н. Классификация ошибок в базах знаний экспертных систем // Вестник СГТУ. 2010. № 4 (50). Вып. 2. С. 125–130.
15. David J.M., Krivine J.P., Simmons R. Second generation expert systems: a step forward in knowledge engineering // Second Generation Expert Systems.Springer-Verlag, 1993. P. 3–25.
16. Davis R. Expert systems: where are we and where do we go from here // AI Magazine. 1982. №3 (2).
17. Clancey W. The epistemology of a rule-based expert system: a framework for explanation // Artificial Intelligence. 1983. № 20 (3). P. 215–251.
18. Simmons R., Davis. R. The roles of knowledge and representation in problem solving // Second Generation Expert Systems. Springer-Verlag, 1993. P. 27–46.
19. Kolodner J. Maintaining organization in a dynamic long-term memory // Cognitive Science. 1983. Vol. 7. P. 243–280.
20. Shank R. Dynamic memory: a theory of reminding and learning in computers and people. Cambridge: Cambridge University Press, 1982. 234 p.
21. Kolodner J. Case-based reasoning. Morgan Kaufman, 1993. 668 p.
22. Aamodt A., Plaza E. Case-based reasoning: foundational issues, methodological variations, and system approaches // Artificial Intelligence Communications. 1994. 7 (1). P. 39–59.
23. Kolodner J. Reconstructive memory: a computer model // Cognitive Science. 1983. Vol. 7. P. 281–328.
24. Di Battista G., Eades P., Tamassia R., Tollis I.G. Graph Drawing: Algorithms for Vizualization of Graphs. — Prentice Hall, 1999. — 397 p.
25. Drawing Graphs. Methods and Models.— Berlin: Springer, 2001. — 312 p. — (Lect. Notes Comput. Sci.; 2025).

26. Касьянов В. Н. Иерархические графы и графовые модели: вопросы визуальной обработки // Проблемы систем информатики и программирования. - Новосибирск: ИСИ СО РАН, 1999. - С. 7-32.
27. Kasyanov V. N. Hierarchical graphs and visual processing // ICM 1998 International Congress of Mathematicians. Abstracts of Short Communications and Poster Sessions. - Berlin, 1998. - P. 292.
28. Kasyanov V. N., Lisitsyn I. A. Hierarchical graph models and visual processing // Proc. of Intern. Conf. on Software: Theory and Practice (ICS-2000). 16th World Computer Congress IFIP. - Beijing, PHEI, 2000. - P.179-182.
29. Kasyanov V. N. Methods and tools for structural information visualization // WSEAS Transactions on Computers. - 2013. - Vol. 12, Issue 7. - P. 349-359.
30. Kasyanov V. N. Kasyanova E. V. Information visualization based on graph models // Enterprise Information Systems. - 2013. - Vol. 7, N 2. - P. 187-197.
31. Lisitsyn I. A., Kasyanov V. N. Higres - visualization system for clustered graphs and graph algorithms // Lecture Notes in Computer Science. - 1999. - Vol.1731. - P.82-89.
32. Савчук З. Р. Засоби візуалізації складних логічних структур / Захар Романович Савчук. // Збірник матеріалів Міжнародної науково-технічної конференції «Перспективи телекомунікацій». – 2019
33. Захарчук А. Г. Методи нечіткої логіки для обробки даних в мережі Інтернету Речей / Захарчук А. Г. – Київ, 2019.
34. Пегат А. Нечеткое моделирование и управление / А. Пегат. – Москва, 2015. – (3-е издание).
35. Branderburg F. J., Himsolt M., Rohrer C. En Experimental Comparsion of Force-Directed and Randomized Graph Drawing Algorithms // Computational Geometry 7. 1997. Vol. 7. – pp. 303 – 325.
36. Hachul S., Jünger M. An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs // LNCS, Springer. 2006. Vol. 3843. – pp. 235 - 250.

37. Ankit Jain, Piero Molino, Joey Bose, and William Hamilton Meta-Graph: Few-Shot Link Prediction Using Meta-Learning
38. Штогріна О. С. Інформаційна технологія створення та використання баз нечітких знань із застосуванням метаграфів / Штогріна О. С. – Київ, 2016.