Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем
(назва факультету)
Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
Магістр
(ступінь вищої освіти)

на тему: Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем
за освітньою програмою 12 Інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав студент групи: ПЗ2121 _____ Андрій ТЕРЛЕНКО
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ проф. Владислав СКАЛОЗУБ
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ доц. Світлана ВОЛКОВА
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:
Техніко-економічні розрахунки _____ доц. Микола ГНЕНИЙ
(назва розділу) (підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент Терленко А. П. _____
(підпис)

Дніпро – 2022 рік

Міністерство освіти і науки України
Український державний університет науки і технологій

Комп'ютерних технологій і систем
(назва факультету)
Комп'ютерні інформаційні технології
(повна назва кафедри)

Пояснювальна записка
до кваліфікаційної роботи
Магістр
(ступінь вищої освіти)

на тему: Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем
за освітньою програмою 12 Інформаційні технології
зі спеціальності: 121 Інженерія програмного забезпечення

Виконав студент групи:

	(підпис студента)	Андрій ТЕРЛЕНКО (Ім'я ПРІЗВИЩЕ)
Керівник:	(підпис)	проф. Владислав СКАЛОЗУБ (посада, Ім'я ПРІЗВИЩЕ)
Нормоконтролер:	(підпис)	доц. Світлана ВОЛКОВА (посада, Ім'я ПРІЗВИЩЕ)

Консультанти:

Кошторис на розробку ПЗ (назва розділу)	(підпис)	доц. Микола ГНЕНИЙ (посада, Ім'я ПРІЗВИЩЕ)

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

Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Computer technologies and systems
(faculty)
Computer information technology
(department)

Explanatory Note
to Master's Thesis
Master
(higher education degree)

on the topic: Research of intelligent procedures and software tools for optimization of the service systems order flows
according to educational curriculum 12 Information technologies
in the Speciality: 121 Software engineering

Done by the student of the group:
Andrii TERLENKO
(name, surname)

Scientific Supervisor: professor Vladyslav SKALOZUB
(position, name, surname)

Normative controller : docent Svitlana VOLKOVA
(position, name, surname)

Supervisors
Calculation of the software development costs docent Mykola HNENYI
(Chapter title heading) (position, name, surname)

Dnipro – 2022

Дніпровський інститут інфраструктури та транспорту

Факультет Комп'ютерні технології і системи кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ .
(підпис)
«___»_____2022 р.

ЗАВДАННЯ

до дипломного проекту на здобуття ОС Магістр
(освітньо-кваліфікаційний рівень)
студента групи ПЗ2122 Андрій ТЕРЛЕНКО
(номер групи) (ПІБ)

1 Тема дипломного проекту: Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем»
затверджена наказом по університету від _____.
2 Термін подання студентом закінченого проекту «22» грудня 2022 р.
3 Вихідні дані до дипломного проекту
приклади постановок завдань оптимізації потоків замовлень сервісних систем, процедури та програмні засоби щодо планування виконання потоків замовлень систем обслуговування, моделі та методи формування інтелектуальних процедур систем обслуговування, процедури нейронних мереж для класифікації неповних та збурених даних досліджуваних об'єктів, приклади застосуваня моделей асоціативної пам'яті в завданнях оцінювання характристик потоків замовлень
4 Зміст пояснювальної записки (перелік питань до розробки) огляд досліджень щодо можливостей реалізації завдань оптимізації потоків замовлень сервісних систем з використанням інтелектуальних процедур та відомих програмних засобів, постановки нових завдань сфери оптимізації потоків замовлень сервісних систем, формування завдань класифікації об'єктів з неточно визначеними параметрами на основі нейронних мереж Хеммінга, як інтелектуальних процедур оптимізації потоків замовлень, розробка процедур мережі Хеммінга для нечітких та експертно визначених (коефіцієнти впевненості) характеристик даних, результати чисельних експериментальних досліджень алгоритмів і процедур мережі Хемінга з нечіткими та експертно визначеними (коефіцієнти впевненості) характеристиками, розробка алгоритмів і програмних засобів класифікації об'єктів з не точно визначеними характеристиками, проведення числових експериментів з аналізу ефективності запропонованих алгоритмів, розроблення програмних засобів для модифікованих мереж Хеммінга, результати економічного аналізу програмних засобів, технічна документація.

5 Перелік демонстраційного матеріалу розробка презентації щодо результатів дослідження інтелектуальних процедур завдань оптимізації потоків замовлень сервісних систем, структура та постановки нових завдань оптимізації потоків замовлень в обслуговуючих системах на основі інтелектуальних процедур асоціативної пам'яті, моделі перетворення не точно визначених характеристик даних в показники шаблонів мережі Хеммінга, результати чисельних експериментальних досліджень алгоритмів і процедур мережі Хемінга з нечіткими та експертно визначеними (коефіцієнти упевненості) характеристиками, програмні засоби класифікації об'єктів з не точно визначеними характеристиками, програмний комплекс реалізації інтелектуальних процедур класифікації для сервісних систем, відео-демонстрація функціонування програмного комплексу.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Кошторис на розробку ПЗ	Доцент Гненний М.В.		

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ		
2	Дослідження питань оптимізації потоків замовлень сервісних систем, процедури та програмні засоби з планування виконання потоків замовлень з урахуванням невизначеності даних.		
3	Аналіз інтелектуальних засобів моделей оптимізації потоків замовлень систем обслуговування,		
4	Постановка завдань оптимізації потоків замовлень в обслуговуючих системах на основі інтелектуальних процедур асоціативної пам'яті, розробка технічного завдання		30%
5	Кошторис на розробку ПЗ		

6	Розробка математичних та інформаціних моделей, алгоритмів та програмного забезпечення досліджень		
7	Проведення чисельних досліджень можливостей та ефективності інтелектуальних процедур сервісних систем та програмних засобів		60%
8	Підготовка тез доповідей за результатами роботи		
9	Підготовка сматеріалів до статті у фаховий журнал		
10	Оформлення пояснювальної записки		
11	Розробка демонстраційних матеріалів		100%

Дата видачі завдання «__» _____ 2022 р.

Керівник дипломного проекту _____ /Владислав СКАЛОЗУБ /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ /Андрій ТЕРЛЕНКО /
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є інтелектуальні процедури і програмні засоби оптимізації потоків у сервісних системах при не точно визначених характеристиках даних.

Предметом дослідження є математичні моделі оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур асоціативної пам’яті мережі Хеммінга при не точно визначених характеристиках даних.

Метою роботи є розвиток постановок завдань та удосконалення математичних моделей оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур Хеммінга при не точно визначених характеристиках даних.

Методи дослідження: методи порівняльного аналізу, статистики, класифікації багатопараметритчних об’єктів, методи нечіткого моделювання та експертних систем, методи експериментальних досліджень та комп’ютерне моделювання. Методи програмної інженерії використовувалися для проектування та розробки програми.

Результати та їх новизна: виконано аналіз моделей та інтелектуальних процедур широкого кола завдань оптимізації потоків замовлень в сервісних системах, запропоновано нові постановки завдань сфери оптимізації потоків замовлень сервісних систем, для реалізації яких застосовуються інтелектуальні процедури класифікації за не точно визначеними даними, проведений широкий числовий експеримент підтвердив достовірність та ефективність запропонованих моделей і методів. Отримано програмну реалізацію розроблених моделей класифікації об’єктів з неточно визначеними параметрами.

Розрахунково-пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та додатків.

Вступ – представляє сутність завдань оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур при не точно визначених характеристик даних. Визначає актуальність завдань досліджень та розробок (3 сторінки).

Перший розділ – містить опис матеріалів наукових літературних джерел щодо сутності та результатів завдань оптимізації потоків замовлень сервісних систем, методів і процедур нейронних мереж Хеммінга, особливостей даних необхідних для процедур класифікації (16 сторінок).

Другий розділ – містить результати розвитку постановок завдань щодо оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур при не точно визначених характеристиках даних, моделі представлення не точно визначених характеристик даних для класичних нейронних мереж Хеммінга, нечіткі та експертні форми представлення не точно визначених даних, а також результати чисельних досліджень модифікованих алгоритмів процедури нейронної мережі Хеммінга. (26 сторінок).

Третій розділ – складається зі структури програмного комплексу класифікації за не точно визначеними характеристиками даних, результати проектування і розробки програми (17 сторінок).

Четвертий розділ – містить результати експериментальних досліджень можливостей запропонованих процедур асоціативної пам'яті для не точно визначених характеристик даних, які підтвердили достовірність та ефективність результатів досліджень та розробок. (9 сторінок).

П'ятий розділ – містить результати економічного аналізу процесів створення програми (10 сторінок).

Висновки Складаються з 1 сторінки;

Список літератури – включає в себе список використаної літератури. Складає 5 сторінки;

Додатки – технічне завдання і робочий проект. Таблиць – 6, рисунків – 59 , бібліографія – 70.

Ключові слова: потоки замовлень, сервісні системи, інтелектуальні процедури , нейронні мережі Хеммінга, нечітка модель даних, модель даних MYCIN на основі коефіцієнтів впевненості, багатовимірна класифікація, програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність роботи. Завдання оптимізації потоків замовлень у сервісних, обслуговуючих системах виникає у багатьох технологіях та системах, є змістовним та досить поширеним. При цьому, для характеристики процедур оптимізації виконується врахування все більшого набору властивостей досліджуваних потоків. Складність інформаційних, технологічних та інших процесів, які забезпечуються сервісними системами, часто впливає на можливості отримати достовірні дані та точно визначені дані про характеристики елементів потоків замовлень. Це суттєво впливає на результати оптимізації роботи обслуговуючих систем. Крім того, певні потоки можуть мати велику кількість компонентів, а також обслуговуючих підсистем з різними властивостями, характеристики яких також можуть змінюватися і відомі не точно. Для забезпечення ефективного (найкращого за поточними даними) вибору обслуговуючого «приладу» можливо застосувати методи інтелектуальних систем. В нашому випадку – інтелектуальні процедури класифікації за не точно визначеними даними. Призначення таких процедур – визначити клас для «виконавця замовлення». Аналіз публікацій свідчить, що натепер завдання із розвитку та розробки таких процедур мають значний теоретичний та науковий інтерес, але дослідження інтелектуальних процедур класифікації за не точно визначеними даними досліджені не в повній мірі. Таким чином, тема досліджень представленої роботи є актуальною.

Об’єктом дослідження є інтелектуальні процедури і програмні засоби оптимізації потоків замовлень у сервісних системах при не точно визначених характеристиках даних.

Предметом дослідження є математичні моделі оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур асоціативної пам’яті мережі Хеммінга при не точно визначених характеристиках даних.

Метою роботи є розвиток постановок завдань та удосконалення математичних моделей оптимізації потоків замовлень сервісних систем на

основі інтелектуальних процедур Хеммінга при не точно визначених характеристиках даних.

Методи дослідження: методи порівняльного аналізу, статистики, класифікації багатопараметритчних об'єктів, методи нечіткого моделювання та експертних систем, методи експериментальних досліджень та комп'ютерне моделювання. Методи програмної інженерії використовувалися для проектування та розробки програми.

Завдання. Завдання дослідження складаються з таких:

- виконати аналіз моделей та інтелектуальних процедур завдань оптимізації потоків замовлень в сервісних системах;
- запропонувати нові постановки завдань щодо оптимізації потоків замовлень сервісних систем, для реалізації яких застосовуються інтелектуальні процедури класифікації за не точно визначеними даними;
- виконати вдосконалення математичних моделей оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур Хеммінга при не точно визначених характеристиках даних, які враховують різні моделі вихідних даних;
- розробити програмні засоби із класифікації компонентів послідовностей замовлень на основі модифікованих процедур мережі Хеммінга;
- провести числовий експеримент з метою підтвердити достовірність та ефективність запропонованих моделей і методів;
- отримати рекомендації стосовно застосування процедур класифікації процедур Хеммінга при не точно визначених характеристиках даних, які враховують різні моделі вихідних даних.

Результати дослідження та їх новизна:

- виконано аналіз моделей та інтелектуальних процедур широкого кола завдань оптимізації потоків замовлень в сервісних системах,

- запропоновано нові постановки завдань сфери оптимізації потоків замовлень сервісних систем, для реалізації яких застосовуються інтелектуальні процедури класифікації за не точно визначеними даними,
- удосконалені математичні моделі оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур Хеммінга при не точно визначених характеристиках даних, які враховують різні моделі вихідних даних.
- проведений широкий числовий експеримент підтвердив достовірність та ефективність запропонованих моделей і методів.
- отримано програмну реалізацію розроблених моделей класифікації об’єктів з неточно визначеними параметрами.

Практичне значення мають результати аналізу стану досліджень щодо методів оптимізації послідовностей замовлень у сервісних системах, нові постановки завдань сфери оптимізації потоків замовлень сервісних систем, для реалізації яких застосовуються інтелектуальні процедури класифікації за не точно визначеними даними, удосконалені математичні моделі оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур Хеммінга, які враховують різні моделі невизначеності параметрів вихідних даних, результати числових експериментальних досліджень, що підтверджують достовірність та ефективність запропонованих моделей і методів класифікації при різних моделях невизначеності параметрів вихідних даних, програмна реалізація розроблених моделей класифікації об’єктів з неточно визначеними параметрами.

Апробація результатів дослідження. Результати дослідницької роботи доповідались на семінарах кафедри КІТ 30.11.2022 р. Матеріали роботи доповідалися на 16 міжнародній науково-практичні конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (13 – 14 грудня 2022 р.) м. Дніпро, Український державний університет науки і технологій.

1. ДОСЛІДЖЕННЯ ПИТАНЬ ОПТИМІЗАЦІЇ ПОТОКІВ ЗАМОВЛЕНЬ СЕРВІСНИХ СИСТЕМ, ПРОЦЕДУРИ ТА ПРОГРАМНІ ЗАСОБИ З ПЛАНУВАННЯ ВИКОНАННЯ ПОТОКІВ ЗАМОВЛЕНЬ З ВРАХУВАННЯМ НЕВИЗНАЧЕНОСТІ ДАНИХ.

1.1. Аналіз результатів сфери моделювання недетермінованих процесів з урахуванням невизначеності даних спостережень.

1.1.1. Моделі та методи завдань оптимізації потоків замовлень сервісних систем. Актуальність дослідження.

Завдання з оптимізації потоків замовлень у сервісних, обслуговуючих, системах (ОПЗСеС) виникає у багатьох технологіях і виробництвах, є змістовним та досить поширеним. При цьому для формування процедур оптимізації виконується урахування все більшого набору властивостей досліджуваних потоків. Складність інформаційних, технологічних та інших процесів у сервісних системах (СеС), часто впливає на можливості отримати достовірні та точно визначені дані про характеристики потоків замовлень. Зазначене суттєво впливає на результати оптимізації роботи СеС. Крім того, певні потоки можуть мати велику кількість компонентів з різними властивостями, характеристики яких також відомі неточно. Для забезпечення ефективного вибору обслуговуючого «приладу» можливо застосувати методи інтелектуальних систем, у нас – це процедури класифікації замовлень за неточно визначеними даними. Призначення процедур полягає у визначенні класу, який показує «виконавця». Аналіз публікацій свідчить, що натепер завдання із розвитку та розробки таких процедур класифікації при неповних та неточно визначених даних мають значний теоретичний та науковий інтерес. Разом з цим дослідження застосування подібних інтелектуальних процедур при оптимізації потоків замовлень в обслуговуючих системах проведені не в повній мірі.

Одним із нових і сучасних напрямків щодо розвитку інтелектуальних процедур оптимізації потоків замовлень у СеС при неточно визначених характеристиках даних, являється використання алгоритмів моделі асоціативної пам'яті мережі Хеммінга (МХ). Необхідно зазначити, що загальновідомі, класичні МХ дозволяють виконувати класифікацію об'єктів (замовлень) при збурених даних, якщо властивості елементів оцінюються значеннями з множини {– 1; + 1}. Важливим завданням із розвитку МХ і розширення кола постановок завдань класифікації послідовностей замавлень у СеС являється удосконалення математичних моделей та процедур оптимізації потоків замовлень СеС на основі інтелектуальних процедур мережі МХ при *неточно визначених характеристиках даних*, далі МХН. В нашій дипломній роботі визначалися та досліджувалися можливості використання в якості моделей первинних характеристик портоків замовлень нечітких множин, а також показників достовірності експертних систем, коефіцієнтів впевненості CF(A) з множини [– 1; + 1]. За рахунок запропонованих форм відображення неточно визначених даних про характеристики потоків замовлень суттєво розширюється сфера практичногозастосування результатів розробок.

Для забезпечення обгрунтованого вирішення указаних питань необхідно виконати розробки та провести дослідження, які полягають в наступному: - виконати аналіз моделей та інтелектуальних процедур завдань оптимізації потоків замовлень в сервісних системах (ОПЗСеС); - запропонувати нові постановки завдань ОПЗСеС (система паркінгу авто; призначення фахівців на посаду в ІТ проєктах тощо), для реалізації яких застосовуються процедури класифікації МХН; - удосконалити математичні моделі ОПЗСеС на основі інтелектуальних процедур МХН, які враховують різні моделі вихідних даних; - розробити програмні засоби із класифікації компонентів послідовностей замовлень на основі модифікованих процедур МХН; - провести широкий і всебічний числовий експеримент, який підтвердити достовірність та ефективність запропонованих моделей і методів МНХ; - отримати

рекомендації стосовно застосування МНХ при неточно визначених характеристиках даних, які враховують запропоновані форми моделей вихідних даних (нечіткі множини (НМ), коефіцієнти впевненості CF(A)). При використанні НМ треба запропоновати моделі кодування, за допомогою яких нечіткі величини подавалися як в МХ {– 1; + 1}, а для моделей CF(A) використовувати процедури, що безпосередньо застосовувала схеми МХ.

1.1.2. Аналіз інтелектуальних засобів моделей оптимізації потоків замовлень систем обслуговування.

Моделі і методи, а також програмні засоби щодо завдань планування та реалізації оптимізації потоків замовлень систем обслуговування мають досить широке поширення. Цим питанням та визначенню програмних забобів їх реалізації присвячено багато наукових досліджень. Щоб встановити актуальність, а також відмінність розробок, виконаних у дипломній роботі, зупинемося на деяких з них.

У роботі [1] запропоновано "Нечітко-множинну багатоперіодну модель вибору стратегій взаємодії організації з групами стейкхолдерів на основі детермінованого еквівалента", А в [2, 3] запропоновано Нечітко-множинну модель вибору стратегій взаємодії вузу зі стейкхолдерами. Питання щодо кластерного аналізу на основі нечітких моделей досліджені в [4], а в [5] подано загальну характеристику нечітких моделей прийняття рішень. Завдання Розробка веб-додатка для вирішення задачі вибору методології управління проектом за нечітких вихідних даних представлені у роботі [6]. Приклади застосування Застосування нечітких моделей у завданнях класифікації, які відповідають темі класифікації при неточно визначених даних, наведені в роботі [8] . Проблеми моделювання щодо Моделі з оцінки ефективності бізнес-процесів організації на основі положень теорії нечітких множин розглянуті в [9], а питання щодо нечітко-множинного сценарного аналізу стратегій взаємодії організації зі стейкхолдерами досліджені в статті [10]. Нечіткі моделі

та методи багатокритеріального вибору в інтелектуальних системах підтримки прийняття рішень [11]. Реалізація завдань із "Управління технічним станом складних систем на основі нечіткої моделі" виконується в [13]. Завдання щодо Моделювання метрик Евкліда та Хеммінга в реальному режимі часу наведено в роботі [15]. Моделі, методи та технологій щодо Застосування математичної моделі нейронної мережі Хеммінга для контролю якості та відновлення некоректних атрибутів метаданих із заголовків сейсмічних файлів. Досліджуються в роботі [17], [19] нечіткі моделі даних, що характеризують «складний об'єкт», учня, застосовуються для загального оцінювання. Такі моделі у певній мірі можуть бути застосовані для завдань управління у СеС. Значний інтерес має робота [20] Нечітка модель оцінки якості порталу вишу для ефективного просування освітніх послуг, в якій зроблено нечітку оцінювання солодного програмного утворення. Стаття [21] Метод нечіткої ідентифікації критичних елементів автоматизованих систем з урахуванням фізично неклонируемых функцій. Містить моделі та процедури ідентифікації елементів автоматизованих систем. Питання щодо Дослідження методів інтелектуального аналізу бібліографічних описів та розробка програмної системи для аналізу списку літератури прирведені та обговорюються у [22]. Розробка ПОШУКОВА СИСТЕМА НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ наведена у статті [24]. Біонічна модель нечіткого логічного аналізу параметрів контрольно-технічних оглядів авіаційних систем представлена у роботі [25], де моделюються складні та досить поширені завдання логічного аналізу параметрів контрольно-технічних оглядів авіаційних систем. Важливе значення для завдань має робота [27] НЕЧІТКО-МНОЖИННА МОДЕЛЬ ОЦІНКИ ЕФЕКТИВНОСТІ ВИКОНАННЯ МІЖНАРОДНИХ ПЕРЕВЕЗЕНЬ ВАНТАЖІВ У ПРОЕКТАХ РОЗВИТКУ МІЖНАРОДНИХ ТРАНСПОРТНИХ КОРИДОРІВ наведено та досліджено у статті.

Приведений огляд результатів досліджень та розробок свідчить про широкий спектр додатків, значні можливості застосування моделей та методів на основі нечітких та експертних моделей первинних даних спостережень

складних систем. Всі такі моделі та методи принципово являються інтелектуальними, що відповідає темі дипломної роботи. Все ж їх застосування спирається на аналіз властивостей конкретного завдання, Безпосередньо формування певних інтелектуальних процедур, що дозволяють виконувати нечітку класифікацію за неточними та неповними даними, за рахунок визначення оптимального «керування», не вдалось знайти. Розробка таких уніфікованих інтелектуальних процедур на основі нейронної мережі Хеммінга являється актуальним завданням

Для постановки завдання дослідження дипломної роботи будемо використовувати результати, представлені у статтях [2, 3]. В [2] представлено результати досліджень завдань оптимального планування широкого кола виробничо-технологічних, логістичних та інших сервісних процесів. В основу методів планування покладено нові інтелектуальні процедури упорядкування (ІПУ) послідовностей елементів (замовлень). Призначення процедур - підвищення ефективності отримання упорядкування замовлень з урахуванням складності операцій формування, а також обмежень на ресурси. В статті розглянуті моделі та методи застосування зазначених процедур, що орієнтовані на процеси розформування-формування (РФ) багатогрупових залізничних составів (БГС) на сортувальних станціях. Формально такі процеси представлені новими моделями упорядкування мульти-послідовностей замовлень з урахуванням складності операцій. При пошуку оптимальних рішень використані моделі асоціативної пам’яті Хеммінга , які дозволяють класифікувати поточні ситуації процесів формування рішень. В цих моделях кожному класу визначених станів (з урахуванням неповноти та збурення даних) відповідає один або кілька раціональних операторів дії (керування) із числа можливих. Процедури інтелектуального формування дозволяють зменшувати кількість варіантів аналізу та підвищують чисельну ефективність методу оптимізації мульти-послідовностей замовлень. В статті приведені інтелектуальні процедури класифікації операцій формування-розформування составів на сортувальних станціях на основі моделей нейронних мереж

Хеммінга. При цьому також розроблено удосконалену структуру інформаційної технології РФ з використанням інтелектуальних процедур, наведені приклади їх застосування.

Основним завданням оптимального планування [2] завдання було упорядкування мульти-послідовностей (УМПСЗ) певних замовлень, представлених у вигляді

$$(S \to Q) : \{S_p \to Q_q\}; p, q = 1,2,\ldots,d \qquad (1.1)$$

де: p – число вхідних in-set, q – число вихідних out-seq потоків завдання упорядкування МП$^{(p,q)}$, а d – граничне значення. Для структур моделей потоків $S_1 \to Q_1$, $S_m \to Q_1$, $S_1 \to Q_r$ тощо, будемо позначати: S_1Q_1, S_mQ_1, S_1Q_r. Послідовності S_p і Q_q містять неподільні складові, елементи, $e_i(p)$, $e_j(q)$ (замовлення, операції ін.). Ці елементи відрізняють за номерами i_p (вхідні неупорядковані) та i_q (вихідні упорядковані), що також мають індекси призначення відповідних замовлень pos-ind n_q для i_q. Прийнято що між елементами out-seq потоків $e_k(q)$ та $e_m(q)$ виконується умова порядку за індексами pos-ind n_q: $n_r(q) \le n_m(q), if\ r < m$ при $(r < m)$, де через r и m позначені номери елементів out-seq Q_q.

Суттєва відмінність та новизна роботи [2] полягала в застосуванні інтелектуальних процедур оптимального упорядкування. В раніше пропонованих [2] методах реалізації завдань УМПСЗ вибір локально оптимального на етапі оператора із числа можливих виконувався за рахунок повного перебору цієї множини, а також формування дерева можливих станів. При цьому для вибору оптимального на етапі оператора управління (ОПУ) використовувався показник порядку станів (метрика). Для зменшення перебору варіантів формування при виборі певного оператора перетворення із числа ОПУ використовуються інтелектуальна процедура класифікації (КОПУ) з використанням моделі асоціативної пам’яті Хеммінга [16, 18] Така

процедура мала визначати «класи» поточних ситуацій процесу упорядкування УМПСЗ, При цьому кожному класу станів (з урахуванням неповноти та збурення даних) відповідає один або кілька раціональних операторів із числа ОПУ. За рахунок таких інтелектуальних процедур класифікації зменшується число варіантів аналізу та підвищується чисельна ефективність процедур оптимального формування УМПСО.

В нашій дипломній роботі отримує подальший розвиток ідея застосування нейронних мереж Хопфілда та Хеммінга при виборі оптимальних дій сервісних систем при реалізації функцій виконання потоків замовлень. На відміну від статей [2, 3], в яких характеристики елементів потоку замовлень були детермінованими і повністю відомими, в дипломній роботі розроблені математичні моделі, алгоритмічні та програмні засоби, призначені для потоків з «неточно визначеними характеристиками параметрів елементів», Огляд літературних джерел, а також програмних засобів не дав результатів цього напрямку досліджень та практичних робіт.

Огляд літературних джерел, виконаний в роботі, разом з тим показав значну кількість завдань сфери сервісного обслуговування потоків замовлень, які за своїми властивостями даних являються «розмитими», слабко структурованими, не повністю визначеними, потребують експертної оцінки значень, В роботі нами побудовано змістовні приклади та сформовано моделі оптимізації виконання запитів до таких недетермінованих за даними та станами функціонування сервісних систем. В них передбачено застосування модифікацій нейронних мереж Хеммінга, розроблених в цій роботі.

Загальний підхід до інтелектуальних процедур оптимізації процесів обслуговування завдань – елементів потоку визначається наступним. Для інтелектуальних систем характерним є реалізація деяких загальних завдань інтелектуального змісту (інтерпретація складних даних, узагальнення, прогнозування, зіставлення шаблонів на основі асоціацій та ін. [2]), які виникають при функціонуванні технічної або технологічної системи. Для формування інтелектуального змісту процедур управління може

використовуватися спеціалізований інструментарії - моделі і методи нечіткого моделювання, нейронних мереж, генетичних алгоритмів і т.п. [3]. У нашому випадку використовуються певні можливості застосування нейронних мереж (як засіб реалізації асоціативної пам'яті, АП)), призначені для створення елементів інтелектуальних систем управління. Зміст і роль АП в зазначених процедурах може бути зведена до наступного. Функціонування певної керуючої системи, в тому числі управління у потоках, зводиться до реалізації деяких «шаблонів дій», виходячи з отриманих оцінок параметрів «поточного стану». Тобто має місце продукційна модель (продукційна система - ПС) управління за «шаблонами дій»

$$\text{ПС} = \{< \text{поточний стан} > \rightarrow < \text{дія} >\}. \qquad (1.2)$$

Часто значення характеристик «поточного стану», набору вхідних параметрів, можуть не збігатися ні з одним з можливих шаблонів. У цьому випадку все ж можливо в рамках моделі конкретної системи керування виконувати «дії», наприклад, виконуючи той шаблон, що є «найближчий» до параметрів «поточного стану». Пошук такого шаблону і може виконувати модель асоціативної пам'яті, в тому числі за умов неповноти та збурених даних, що використовуються для вибору купування. Що реалізує НМХ.

Можливо застосувати різні види процедур функціонування ПС, які розрізняються способами оцінки поточного стану, відбору шаблонів, змістом і способами реалізації куруючих дій. У простій формі «дія» може вказувати наступний шаблон, який слід виконати. Управління об'єктом представляється як реалізація послідовностей етапів типу

а) $\{< \text{поточний стан} > \rightarrow < \text{розпізнаний шаблон} > \rightarrow < \text{дії} >\}$, (1.3)

чи виконувати операції узагальнення або ж прогнозування значення характеристик для вибору керуючих впливів:

б) $\{< \text{стан} > \rightarrow < \text{шаблон} > \rightarrow < \text{прогноз} > \rightarrow < \text{дія} >\}$. (1.4)

В рамках процедури (12) – (14) можливо реалізувати завдання оптимального управління розподілом елементів послідовності замовлень між

виконавцями в сервісних системах. В нашій роботі вибір оптимального управління реалізується за (13) за рахунок модифікацій мережі НМХ.

1.1.3. Огляд програмних аналогів щодо планування виконання потоків замовлень з врахуванням невизначеності даних.

На підставі огляду літературних джерел встановити існування прямих загальноприйтятих програмних аналогів щодо планування завдань виконання потоків замовлень з врахуванням невизначеності даних не вдалося. Відомі нам аналоги реалізують окремі функції, які мржуть бути використані для завдань оптимізації планування потоків у СеС. Інша частина засобів потребує виконання завдань проограмування в власному середовищі. Одним із надпотужних засобів являється програмне забезпечення Oracle Crystal Ball. Цей засіб Oracle Crystal Ball являється додатком до Excel для інтелектуального моделювання бізнес-процесів, прогнозування невизначених змінних тощо. Oracle Crystal Ball використовується як додаток до Microsoft Office Excel. Засіб Oracle Crystal Ball дає змогу вирішення широкого спектру завдань моделювання та прогнозування, дозволяє оцінювати важливі характеристики завдань. складових завдань На рис. 1 приведений приклад із представлення графіка прогнозу часового ряду. Разом з тим реалізація завдань щодо інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем в названих системах безпосередньо не передбачено. Це пояснюється їх новизною да досліднгициким характериром розробки.

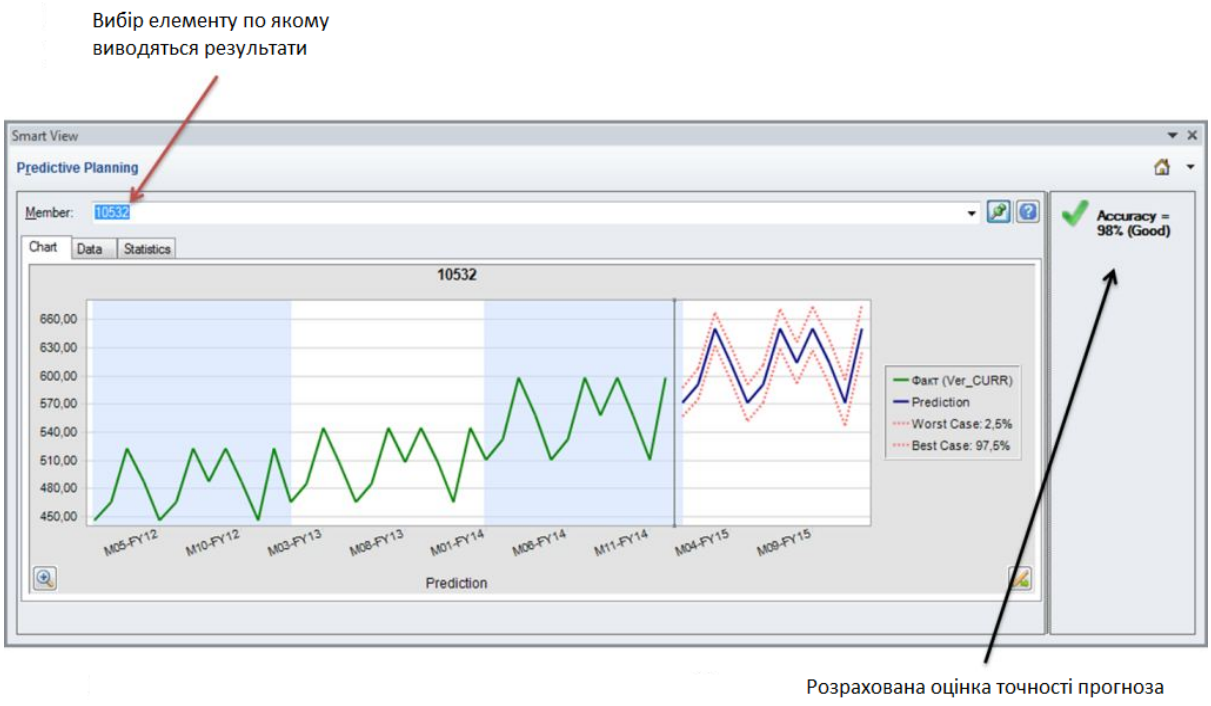


Рисунок 1.1 – Представлення графіка прогнозу часового ряду в Crystal Ball

1.2. Характеристика сфери застосування результатів розробок.

На основі проведення досліджень та виконананя порівняльного аналізу щодо моделей. алгоритмів і процедур оптимізації потоків замовлень у СеС зазначається наступне.

1. Завдання оптимізації потоків замовлень у сервісних системах виникає у багатьох технологіях та системах, є змістовним та досить поширеним. При сучасному розвитку комп'ютерного моделювання виникли можливості використовувати для характеристики процедур оптимізації все більший набір властивостей досліджуваних потоків, в тому числі неточно визначених ознак характеристи. .
2. Складність процесів, які забезпечуються сервісними системами, суттєво впливає на можливості отримати достовірних, повних та точно визначених даних про характеристики елементів потоків. Це суттєво впливає на процедури моделювання та результати оптимізації роботи обслуговуючих систем у цілому.

3. Суттєвим фактором являється те, що контрольовані потоки можуть мати велику кількість компонентів, а також обслуговуючих підсистем з різними властивостями, характеристики яких також можуть змінюватися і відомі неточно.
4. Для забезпечення ефективного функціонання СеС модливо сформулювати завдання із вибору кращого обслуговуючого «приладу», для чого застосувати методи інтелектуальних систем. В роботі такі функції виконують інтелектуальні процедури класифікації за неточно визначеними даними.
5. Огляд літературних джерел показав, що завдання із розробки таких уніфікованих та спеціалізованих процедур мають певний науковий інтерес. В теоретичному та практичному аспекті дослідження інтедектуальних процедур СеС є актуальними.
6. Практичне значення мають щодо сфери застосування результатів роботи мають дослідження методів оптимізації послідовностей замовлень у СеС, нові постановки завдань сфери оптимізації потоків замовлень сервісних систем, для реалізації яких застосовуються уніфіковані інтелектуальні процедури класифікації за неточно визначеними даними на основі інтелектуальних процедур Хеммінга, що використовують моделі для достовірного відображення невизначеності параметрів вихідних даних
7. Важливу роль мають результати числових експериментальних досліджень, що підтверджують достовірність та ефективність запропонованих моделей і методів класифікації, а також програмна реалізація розроблених моделей класифікації об’єктів з неточно визначеними параметрами.

1.3. Постановка завдань оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур асоціативної пам'яті та обґрунтування завдань дослідження.

Робота призначена для реалізації процедур класифікації об'єктів, що належать до упорядкованого потоку завдань (замовлень), які характеризуються системою певних властивостей, щодо організації їх оптимального виконання певною сервісною системою (визначається при класифікації).

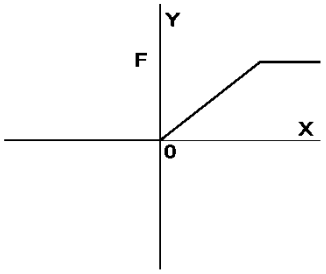
Змістовно створений в роботі програмний комплекс планування отримує потік вхідних даних, що характеризують різні властивості елементів потоку, а також властивості виконавчих вузлів сервісної системи. На основі цих даних обирається найбільш підходящий виконавець з використанням інтелектуальних процедур недетермінованої класифікації.

Вибір оптимального рішення (реалізується на основі нечіткої або іншої недетермінованої класифікації) виконується за допомогою моделі модифікованої нейронної мережі Хеммінга. Для реалізації завдання класифікації за даними, які відображаються нечіткими, або іншими моделями представлення недетермінованих та слабко структурованих даних, у цій роботі були запропоновані, використані та досліджені кілька форм моделей відображення невизначених даних. А саме – у вигляді нечітких величин, μX: $X \rightarrow [0; 1]$, у вигляді коефіцієнтів довіри $CF(X)$: $X \rightarrow [-1, +1]$. Класичні моделі нейродинамічних процесів, які відображають нейронні мережі Хопфілда та Хеммінга, передбачають, що вхідні вектори та шаблони моделі приймають значення із множини $\{-1, +1\}$. Такі властивості процесів застосування класичних нейронних мереж Хопфілда та Хеммінга не відповідають моделям даних, необхідних для представлення слабко структурованих даних, в тому числі нечітких величин (μX) та коефіцієнтів довіри ($CF(X)$).

В роботі для можливості застосування мережі Хеммінга сформовано кілька представлень та перетворень недетермінованих та слабко

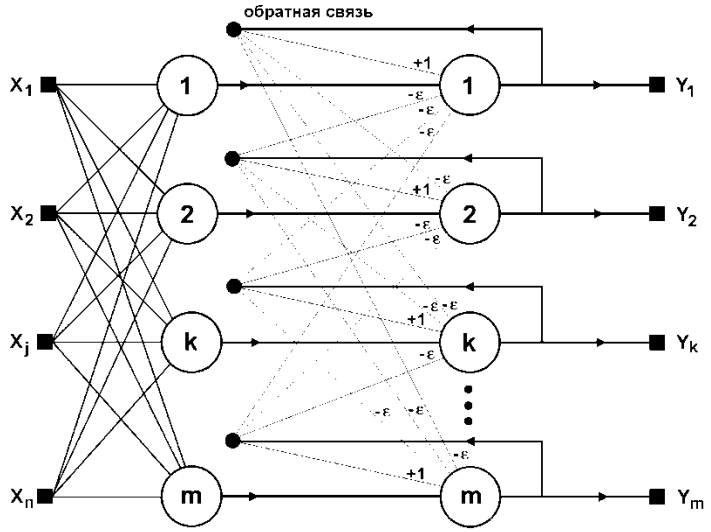
структурованих даних, даних що не є точними, до форми процедур мережі Хеммінга. Для кожного з цих представлень проведено широке коло експериментальних чисельних досліджень процесів збіжності та отриманих результатів класифікації. На основі подальшого аналізу результатів експериментів було визначено кращу модель і сформовано нову модифікацію моделі мережі Хеммінга. Необхідно зазначити, що для запропонованої моделі забезпечуються всі властивості щодо реалізації асоціативної пам'яті. Тобто можливість класифікувати неповні або збурені вхідні дані, відповідно до моделей шаблонів, збережених в пам'яті нейронної мережі.

Розглянемо моделі з зворотними зв'язками та процедури нейронної мережі Хеммінга (НМХ), представленої на рис. 1 та рис. 2. Нейронна мережа НМХ запам'ятовує «k» шаблонів – «n» вимірних векторів {xi (i= 0, 1, n -1) k-ого зразка }, які знаходяться в {-1, +1}.



б)

Рисунок 2 Активаційна функція мержі Хеммінга (б)



Вхід 1 шар 2 шар Вихід

Рисунок 3 Структурна схема мережі Хеммінга

На стадії ініціалізації ваговим коефіцієнтам першого шару і порогу активаційної функції привласнюються наступні значення:

$$w_{ik} = \frac{x_i^k}{2},\ i=0...n-1,\ k=0...m-1 \quad (1)$$

$$T_k = n / 2,\ k = 0...m-1 \quad (2)$$

Тут x_i^k – і-ий елемент k-ого зразка.

Вагові коефіцієнти гальмуючих (зворотних) синапсів у другому шарі беруть рівними деякій величині $0 < \varepsilon < 1/m$. Синапс нейрона, який зв'язаний з його ж аксоном має вагу +1.

Алгоритм функціонування мережі Хеммінга наступний:

1. На входи мережі подається невідомий вектор **X** = $\{x_i: i=0...n-1\}$ з елементами множини {-1, +1}, виходячи з якого розраховуються стани нейронів першого шару (верхній індекс у дужках вказує номер шару мережі):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j,\ j=0...m-1 \quad (3)$$

Після цього отриманими значеннями ініціалізують значення аксонів другого шару:

$$y_j^{(2)} = y_j^{(1)},\ j = 0...m-1$$

2. Обчислити нові стани нейронів другого шару:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0...m-1 \qquad (4)$$

і значення їх аксонів:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], \; j = 0...m-1 \qquad (5)$$

Активаційна функція f [*] має вигляд порога (рис. 2 б), причому величина У=F повинна бути досить великою, щоб будь-які можливі значення аргументу не призводили до «насичення».

3. Перевірити, чи змінилися виходи нейронів другого шару за останню ітерацію. Якщо так - перейди до кроку 2. Інакше - кінець.

В НМХ порівнюється вихід кожного нейрону (*k*-ого зразка) з (майже) середнім значенням всіх інших. За рахунок цього визначається найближчий шаблон від вхідного вектору. Збіжність та стійкість цього послідовного процесу є доведеною (у вигляді достатніх умов) для мережі Хопфілда [***] за вимог що на практиці в більшості випадків реалізуються. Модель НМХ має такі самі властивості щодо умов отримання рішень.

Зазначимо що при цьому структура кодів вхідного вектору та зразків *не враховується* на повторних етапах послідовної процедури (4) – (5). Для коректного застосування процедури НМХ при інших вхідних векторах, ніж $\{x_i: i=0...n-1\}$ з елементами множини {-1, +1}, необхідно визначити формальні моделі переходу від прийнятих форм відображення невизначених даних до наведених ($\{x_i: i=0...n-1\}$, -1, +1}). Представимо такі формальні моделі переходу докладно для та різних типів відображення недетермінованих та слабко структурованих даних, «неточно визначених» даних.

Висновки до розділу 1

Обслуговування потоків замовлень є одним з головних завдань у багатьох сервісних системах (СеС) є змістовним та досить поширеним, воно виникає в багатьох технологіях та системах. При цьому важливим є завдання оптимізації таких процесів. Для забезпечення ефективності процесів обслуговування цього враховується все більший набір властивостей

досліджуваних потоків. Через складність інформаційних, технологічних та інших процесів сервісних систем часто відсутні можливості щодо отримання достовірних та точно визначених даних, які характеризуть елементи потоків замовлень. В розділі показано, що для забезпечення ефективного вибору виконавців замовлень, «обслуговуючого приладу» можливо застосувати методи інтелектуальних систем. В дипломній роботі ними виступають інтелектуальні процедури класифікації за неточно визначеними даними. Призначення процедур класифікації полягає в встановленні відповідного оптимального «виконавця замовлення» для елементів потоку замовлень сервісної системи.

Для реалізації завдань щодо застосування інтелектуальних процедур класифікації для оптимізації роботи СеС у розділі виконано наступне. Проведено аналіз результатів сфери моделювання недетермінованих процесів з урахуванням невизначеності даних спостережень. Виявлені та досліджені основні моделі та методи, які можуть бути застосовані для вирішення завдань оптимізації потоків замовлень сервісних систем. При тому встановлена актуальність таких досліджень та розробок програмних та інструментальних засобів. В розділі проведено та предствалено аналіз можливостей використання інтелектуальних засобів в моделях оптимізації потоків замовлень СеС. Також наведено огляд програмних аналогів, які дають можливості удосконалювати процедури планування та виконання обслуговування потоків замовлень з врахуванням невизначеності даних. На основі таких досліджень були визначені характеристики сфери застосування результатів розробок.

Основним результатом розділу 1 являється постановка завдань з оптимізації потоків замовлень сервісних систем на основі інтелектуальних процедур асоціативної пам’яті, а також визначення та обгрунтування завдань дослідження. При цьому також сформульовані завдання щодо розробок моделей та алгоритмів, потрібних засобів програмного забезпечення.

2. РОЗРОБКА ТА ДОСЛІДЖЕННЯ ЗАВДАНЬ ОПТИМІЗАЦІЇ ПОТОКІВ ЗАМОВЛЕНЬ В ОБСЛУГОВУЮЧИХ СИСТЕМАХ НА ОСНОВІ ІНТЕЛЕКТУАЛЬНИХ ПРОЦЕДУР АСОЦІАТИВНОЇ ПАМ'ЯТІ.

2.1. Розробка та дослідження процедур оптимізації потоків замовлень в сервісних системах на основі нейронних мереж Хеммінга.

2.1.1. Постановки завдань оптимізації потоків замовлень сервісних систем з неточно визначеними даними.

Представимо постановки завдань щодо оптимізації потоків замовлень сервісних систем з неточно визначеними даними. Сутність завдань у тому, що в них процес оптимізації представлений у формі функціонуваня певної ситстеми диспетчерського управління складними системами. На кожному етапі функціонування необхідно на підставі даних про власні характеристики системи, а також про властивості досліджуваного елементу, встановити «прилад» або «виконавця», які описані сукупностями структур, шаблонів, який найбільше відповідає вхідному до СеС елементу. При цьому дані як про властивості СеС, так і про вхідний об'єкт являються неточно визначеними та неповними.

Завдання категорії 1 з оптимізації потоків замовлень сервісної системи, яка виконує призначення на посаду виконавців завдання з тестування програмного забезпечення, з використанням інтелектуальних процедур асоціативної пам'яті, полягає в такому. Розглядається сервісна система, в якій, за нечіткими характеристиками, вирішується завдання щодо вибору тестувальника, кортий є найбільш придатним для виконання певної задачі з тих, що надходять до системи. Задача та «ідеальний» виконавець для неї мають наступні характеристики:

1. Пріоритет задачі

2. Складність задачі
3. Потрібні навички web тестування
4. Потрібні навички API тестування
5. Потрібні навички тестування мобільних додатків
6. Потрібний досвід виконавця
7. Потрібне знання структури проекту
8. Завантаженість виконавця
9. Навички web тестування
10. Навички API тестування
11. Навички тестування мобільних додатків

Були побудовані такі шаблони:

	X1 Приорітет задачі	X2 Складність задачі	X3 Потрібні навички web тестування	X4 Потрібні навички API тестування	X5 Потрібні навички мобільного тестування
K1	1	0.7	1	0.3	0
K2	0.7	0.5	0.8	0.3	0
K3	0.4	0.9	0.4	0	0.8
K4	0.1	0.2	0	0.8	0

	X7 Потрібне знання структури проекту	X8 Завантаженість виконавця	X9 Навички web тестування	X10 Навички API тестування	X11 Навички мобільного тестування
K1	0.85	0.4	0.8	0.7	0
K2	0.4	0.7	0.6	0.3	0
K3	0.6	0.5	0.3	0	0.6
K4	0.4	0.6	0	0.5	0

Опис заданих шаблонів:

K1 - Задача найвищого пріорітету, складністю вище середнього, що точно потребує навичок веб-тестування та може потребувати навичок тестування API. Виконавець повинен мати багатий досвід та високий рівень

знання структури проекту. Також, ідеальний виконавець має бути середньо завантаженим та мати навички веб та API тестування вище середнього.

К2 - Задача високого пріорітету, середньої складності, для якої з високою вірогідністю потрібні навички веб-тестування та можуть бути потрібні навички тестування API. Виконавець може мати середній досвід та рівень знання структури проекту нижче середнього. Виконавець може бути завантаженим на рівень вище середнього, мати середні навички веб тестування, а його рівень API тестування може бути досить низьким.

К3 - Задача середнього пріорітету, але з високою складністю, що з високою вірогідністю потребує навичок мобільного тестування та може потребувати навичок веб-тестування з вірогідністю нижчою за середню. Підходящий виконавець може мати середній та рівень знання структури проекту. Він може бути середньо завантаженим, мати середні навички мобільного тестування. Здатність до веб-тестування може бути досить низькою.

К4 - Задача найнижчого пріорітету, низької складності, для якої з високою вірогідністю потрібні лише навички API тестування. Виконавець може бути недосвідченим та має мати рівень знання структури проекту близький до середнього. Він може бути завантаженим на рівень трохи вище середнього, та мати середні навички API тестування.

Завдання 2 з оптимізації потоків замовлень сервісної системи, яка виконує призначення «місця паркування», з використанням інтелектуальних процедур асоціативної пам’яті, полягає в такому. Розглядається сервісна система, в якій, за нечіткими характеристиками вирішується завдання щодо вибору паркінгу, який найбільш підходить для деякого транспортного засобу з тих, що надходять до системи обслуговування автотранспортних засобів.

Паркінг і транспортні засоби мають наступні нечіткі характеристики:

1. Близькість паркінгу до замовника
2. Ціна паркування

3. Наявність вільних місць
4. Максимальна висота транспортного засобу, що може заїхати до паркінгу
5. Максимальна ширина транспортного засобу, що може заїхати до паркінгу
6. Наявність зарядки для електрокарів
7. Наявність доступу для людей з обмеженими можливостями
8. Висота траспортного засобу, що надходить для паркінгу
9. Ширина траспортного засобу, що надходить для паркінгу
10. Чи є вхідний транспортний засіб електрокаром
11. Чи має водій обмежені можливості

Побудовані наступні шаблони на основі нечітких даних:

	X1 Близькість до замовника	X2 Ціна	X3 Вільних місць	X4 max висота	X5 max ширина
K1	0.8	0.7	0.2	0.4	0.5
K2	0.4	0.3	0.6	0.8	0.9
K3	0.6	0.5	0.1	0.6	0.5
K4	0.2	0.1	0.9	0.7	0.8

	X7 Наявність доступу для інвалідів	X8 Висота ТС	X9 Ширина ТС	X10 Чи є ТС електрокаром?	X11 Чи є водій інвалідом?
K1	1	0.4	0.4	1	1
K2	0	0.7	0.7	1	0
K3	1	0.5	0.5	0	1
K4	0	0.6	0.7	0	0

Опис заданих шаблонів:

К1 - Паркінг найближчий до замовника, з високою ціною та малою кількістю вільних місць. Він має середні максимальну ширину та висоту, має зарядку для електромобілів та доступ для людей з обмеженими можливостями. ТС замовника, що надходить до паркінгу, має середні ширину та висоту, є електрокаром та керується інвалідом.

К2 – Паркінг, що знаходиться на середній дальності від замовника, є досить дешевим та майже наполовину пустим. Має високі максимальну ширину та висоту та має зарядку для електромобілів, але не є доступним для інвалідів. ТС замовника має параметри ширини та висоти вище за середні та є електрокаром.

К3 - Паркінг, що знаходиться на середній дальності від замовника, та має середню ціну. Він майже пустий і дозволяє паркування для транспорту з середньою шириною та висотою. В цьому паркінгу немає зарядки але є доступ для інвалідів. ТС замовника має середню висоту та ширину, а сам водій має обмежені можливості.

К4 - Паркінг знаходиться далеко від замовника, але має низьку ціну і дуже багато вільних місць. Він здатний розмістити широкий транспорт з висотою вищою за середню, але не має зарядки та доступу для інвалідів. ТС замовника має параметри вище за середні.

Одним із методів, який дозволяє перейти від нечіткої форми представленя первинних даних до стандартної множини значень моделі НМХ {-1, +1}, являється перекодування.

Визначимо наступну процедуру перекодування нечітких показників μ_X: $X \rightarrow [0; 1]$ (6) до множини значень моделі НМХ {-1, +1}. На відміну від попередніх моделей (7) – (8) сформуємо нову процедуру представлення нечітких величин для НМХ. По перше, будемо ураховувати елементи метрик (9) – (10), оцінюючи значення функцій приналежності, вважаючи що нечітка величина А’ відповідає змінним зразків (шаблонів фундаментальної пам’яті), а величина В’ – визначає параметри приналежності вхідного «зонду», що

необхідно класифікувати. При цьому будемо визначати значення коду таким індикатором H_r

$$f(|\mu_A(u_r) - \mu_B(u_r)|) =$$

$$H_r = 1, \text{ при } \mu_A(u_r) <= \mu_B(u_r) \quad (11)$$

$$-1, \text{ при } \mu_A(u_r) > \mu_B(u_r). \; r = 1, 2, .., q \text{ (розмірність зразків).}$$

Можливо кодування, коли рахується $H_r = 0$, при $\mu_A(u_r) = \mu_B(u_r)$.

По друге, значення кодів (11) присвоюють зразкам (шаблонам класифікації) кожний раз, коли досліджується нових вхідний «зонд». Таким чином, в процедурі перекодування (11) виконується як для кожного зразка, так і для вхідного «зонду», тобто B'. Саме в цьому перетворенні, зміні представлення, множини зразків {A'} у відповідності до вхідних даних B' полягає особливість запропонованої процедури перекодування до форми НМХ. Зрозуміло, що при таких перетвореннях також залишається можливість однакового представлення кодами різних нечітких величин. Разом з тим такі можливості скорочуються, і залишаються вірними наведені вище твердження стосовно ролі кодування, яке необхідно поєднувати з метою застосування моделі НМХ – виконання класифікації нечітких даних. Необхідно ще раз наголосити, що початкові форми зразків залишаються незмінними і зберігаються в базі даних системи класифікації. Методики отримання таких нечітких представлень вхідних даних різноманітні, вони в цій роботі не розглядаються.

Наведемо приклад застосування процедури (11) для зонду B'та множини шаблонів {A'}

B' =(0.7 0.2 0.4 1 0.5) Коди H_r (B') = **(1 1 1 1 1)**, перша умова (11).

{A'}= { (0.3 0.5 1 0.2 0.6) /(0.7 0.8 0.4 1 0.5) / (0.1 0.4 0.2 0.5 0.9) }.

H_r (B') = {**(1 -1 -1 1 -1)/ (1 -1 1 1 1)/ (1 -1 1 1 -1)**}.

Приклад показує, що початкова нечітка модель класифікації стала *стандартною для мережі НМХ.* Крім того очевидно, що «зразок 2» буде мати найменшу відстань за (9), або за процедурою НМХ (1) – (5).

Інший варіант кодування (11) виконується в два етапи. На першому вхідні дані B’ кодуються відповідно до (7), (8). На наступному за кодами H_r(B’) перетворюються за шаблонами {A’} у відповідності до процедури (11).

Розглядається та досліджується модель кодування та процедура відображення невизначених даних у вигляді коефіцієнтів довіри CF(X): X → [-1, +1]. В ній характеристики неточних знань, оцінок ступеня упевненості, кодуються експертним чином, відповідним функціональним засобом або безпосередньо призначаються, що відповідає широкому класу завдань формування Баз Знань (БЗ) Експертних Систем (ЕС) [***]. Застосування такої моделі (форми представлення знань) зумовлено її широким вживанням і дослідженням можливостей, а також відомими функціональними та програмними засобами реалізації. Також область кодів {-1, 0, +1} входить до [-1, +1].

Блок-схема рис. 2.3 визначає загальну процедуру щодо класифікації засовлень у СеС при недетермінованих характеристиках даних

Б1: Завдання типу кодування нечітких даних

Б2: Отримати дані (нечіткі) шаблонів класифікації

Б3: Отримати зонд-вхідний вектор для класифікації

Б4: Аналіз моделей кодування

Б5: Модель кодування шаблонів відповідно вхідного зонду

Б6: Перша модель кодування шаблонів

Б7: Друга модель кодування шаблонів

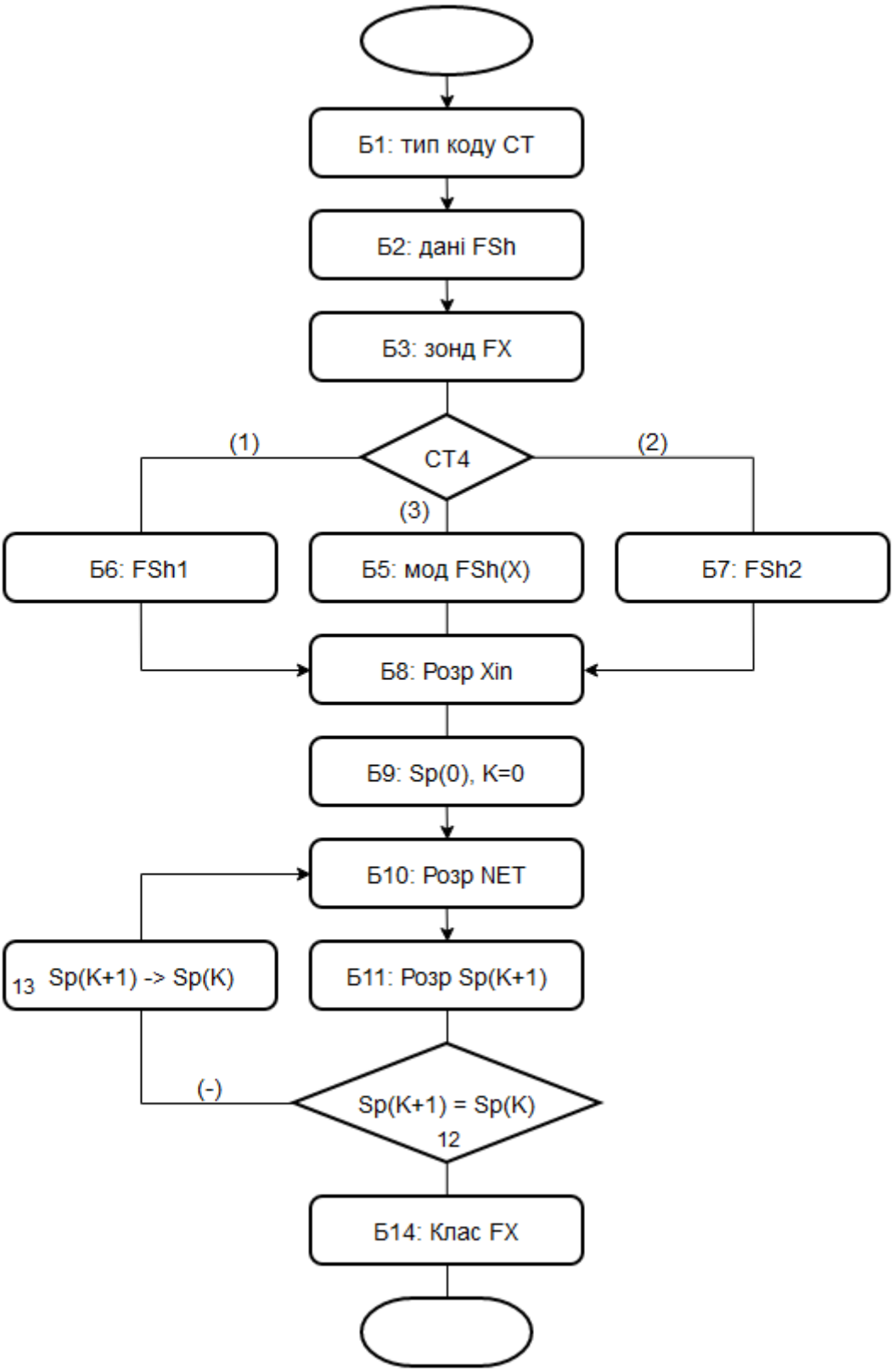


Рисунок 2.3. Блок-схема процедури дослідження

Б8: Розрахунок вхідних кодів зонду для мережі Хеммінга

Б9: Початкові значення станів мережі

Б10: Розрахунок показників NET мережі Хеммінга

Б11: Розрахунок параметрів станів (класів) на новому кроці процесу (К+1)

Б12: Перевірка умови закінчення процедури

Б13: Перезапис параметрів станів

Б14: Повернення результатів класифікації зонда FX

2.1.2. Розробка інтелектуальних процедур класифікації на основі нечітких моделей неточно визначених даних.

Процедура «прямого» кодування. Для нечіткої моделі представлення вхідних величин будемо використовувати наступну «пряму» процедуру кодування зразків (шаблонів фундаментальної пам'яті) та досліджуваних вхідних векторів. В нечітких моделях «розмитим» X параметрам моделі необхідно зіставити ступені приналежності за рахунок функції μ_X

$$\mu_X: X \rightarrow [0; 1] \qquad (6)$$

Після такої процедури фазифікації дані з діапазону μ_X: [0; 1] необхідно перетворити в елементи множини {-1, +1}. Для цього визначимо наступну функцію кодування

$$x_{ci}: = \{-1, \text{ при } \mu_{Xi}: < 0.5; +1 \text{ при } \mu_{Xi}: >= 0.5\} \qquad (7)$$

або її узагальнення

$$\begin{aligned} x_{ci}: = \{&+1, \text{ при } \mu_{Xi}: >= h; \\ &-1 \text{ при } \mu_{Xi}: < l; \\ &0 \text{ при } l < \mu_{Xi}: < h\}. \end{aligned} \qquad (8)$$

При кодуванні (7) всі елементи нечіткої моделі при класифікації відповідають моделі НМХ, і вона може бути реалізована у вигляді (1) – (5). Кодування (7) являється надто спрощеним, при ньому багато різних «розмитих» даних μ_X: X → [0; 1] отримають однакові коди за (7). Функція

кодування (8) вводить додатковий елемент у простір функціонування моделі НМХ, розширює множину значень до {-1, 0, +1}. Величина коду «0» означає змістовну подібність показників в діапазоні $l < \mu_{Xi} < h$, будь які значення в цій області змістовно «однакові». Уведення «0» в шаблони та в коди вхідних даних x_{ci} не повинно змінити процедуру (8), (1) – (5). При цьому також залишається можливість отримати однакові коди для різних «розмитих» даних.

Кодування (7) і (8) необхідно поєднувати з метою застосування моделі НМХ – виконання класифікації нечітких даних. Через це необхідно зауважити, що у разі співпадіння кодів для шаблонів (зразків даних класів) необхідно просто вилучити однаковий шаблон, Представлення багатьох вхідних даних одним кодом (7) або (8) показує, що всі ці значення векторів «приблизно однакові». А до якого зразка вони відносяться, клас шаблону, визначить процедура (1) – (5), яка залишається змістовною з урахуванням зроблених зауважень.

Нечіткі величини як коди. Наступна форма застосування моделі НМХ - безпосереднє використання нечітких величин в процедурах подібних за змістом до (1) – (5), в яких арифметичні операції замінюються на їх аналоги, які використовуються в теорії нечітких множин. При цьому виникає багато можливостей організації таких процесів поетапного моделювання процесів функціонування мережі, стійкості та збіжності тощо. В роботі пропонується і виконується реалізація цих «нечітких» моделей НМХ (ННМХ) шляхом чисельного експерименту. Прийнятність такої моделі встановлюється засобами аналізу.

В якості даних зразків фундаментальної пам’яті та вхідних даних «зондів» використовуються самі значення величин, тобто показники фаззифікації μ_X: $X \rightarrow [0; 1]$ (6). Зрозуміло, що при цьому багатократно розшириться множина можливих станів моделі ННХ, по відношенню до НМХ, відбудеться зміна інших характеристик процедур функціонування тощо. Для забезпечення можливостей такої реалізації в роботі прийнята найпростіша

модель оперування нечіткими величинами, при цьому моделі (3) – (4) мали наступній вигляд (формальне представлення операцій добутку «*» чисел операцією «min» для нечітких множин, а операцій суми – операцією «max» відповідних нечітких множин). При таких моделях нечітких операцій отримуємо функції наступного вигляду для процедур НМХ

$$Y_j = MAX(MIN(B10;H10);MIN(C10;H11);MIN(D10*H12);$$
$$MIN(E10;H13);MIN(F10;H14);MIN(G10;H15))/2 \quad (9)$$

$$S_j = J19 - C8(\varepsilon)*(MAX(I19(Si); K19(Sk); L19(S_L);\ldots;M19(S_M))). \quad (10)$$

Функція (9) являється змістовним аналогом (3), а функція (10) відображає результат дії функції (4) в термінах зазначених нечітких операцій. В (9) значення змінних B10, C10, …, G10 містять значення ступенів приналежності нечітких величин, які утворюють зразок, а значення змінних H10, H11, …, H15 – визначають параметри приналежності вхідного «зонду». В рівнянні (10) у відповідності до змісту (3) визначається стан «S_j» вихідного шару, що відповідає структурі мережі рис. 2. При цьому змінні I19(Si); K19(Sk); L19(S_L);…;M19(S_M) відповідають вихідному шару мережі, містять вказані параметри НМХ. Змінна C8(ε) – містить константу, відповідно моделі (4).

Модель на основі відстані Хеммінга для нечітких множин. Наведені вище форми (7) – (8) кодування не точно визначених характеристика даних, як «розмитих» множин, не враховують відстаней між показниками приналежності окремих значень.

Знайти необхідні методи та засоби використання нечітких, не точно визначених, даних в НМХ не вдалося. Разом з тим результати наукової статті [Великоіваненко, Г.І.,*] дають певні можливості щодо побудувати математичної моделі, що приводять до застосування мережі Хеммінга для нечітких представлень даних, а також для розробки потрібного для цього програмного забезпечення. В ній, для оцінювання економічної безпеки було

використано поняття нечіткої відстані Хеммінга, яке визначається за формулою:

$$d\left(\underset{\sim}{A},\underset{\sim}{B}\right)=\sum_{r=1}^{q}\left|\mu_{\underset{\sim}{A}}(u_r)-\mu_{\underset{\sim}{B}}(u_r)\right| \qquad (6)$$

Де $\underset{\sim}{A},\underset{\sim}{B}$ - нечіткі множини на скінченній універсальній множині U потужністю q, $u_r \in U$, $\mu_{\underset{\sim}{A}}(u_r),\ \mu_{\underset{\sim}{B}}(u_r) \in [0,1]$,

$r=\overline{1,q}$ і $0 \le d\left(\underset{\sim}{A},\underset{\sim}{B}\right) \le q$.

Іншою формою метрики нечітких величин також, було використано відстань Хеммінга з додатнім відхиленням для нечіткої множини $\underset{\sim}{A}$ між нечіткими множинами $\underset{\sim}{A}$ і $\underset{\sim}{B}$, що знаходиться наступним чином:

$$- \quad d^{+}\left(\underset{\sim}{A},\underset{\sim}{B}\right)=\sum_{r=1}^{q}\alpha_r \cdot\left|\mu_{\underset{\sim}{A}}(u_r)-\mu_{\underset{\sim}{B}}(u_r)\right| \qquad (7)$$

Де $u_r \in U$, $\mu_{\underset{\sim}{A}}(u_r),\ \mu_{\underset{\sim}{B}}(u_r) \in [0,1]$, $r=\overline{1,q}$, α_r – індикатор, що визначається так:

$$\alpha_r=\begin{cases}1, \text{якщо } \mu_{\underset{\sim}{A}}(u_r) \ge \mu_{\underset{\sim}{B}}(u_r)\\ 0, \text{якщо } \mu_{\underset{\sim}{A}}(u_r) < \mu_{\underset{\sim}{B}}(u_r)\end{cases}$$

При цьому на основі вищезгаданих відстаней Хеммінга (6)-(7), наявності множин експертів $X=(x_1,x_2,\dots,x_n)$, визначеної множини індикаторів стану економічної безпеки $Y=(y_1,y_2,\dots,y_n)$, а також множини оцінюваних економічних систем $Z=(z_1,z_2,\dots,z_n)$, у матричному вигляді було побудовано:характеристику «ступінь належності економічної системи Z за індикатором Y до безпечного стану»

$$\underset{\sim}{R_1}=\begin{Vmatrix}\underset{\sim}{\chi_{R_1}}(y_1,z_1) & \underset{\sim}{\chi_{R_1}}(y_1,z_2) & \dots & \underset{\sim}{\chi_{R_1}}(y_1,z_m)\\ \underset{\sim}{\chi_{R_1}}(y_2,z_1) & \underset{\sim}{\chi_{R_1}}(y_2,z_2) & \dots & \underset{\sim}{\chi_{R_1}}(y_2,z_m)\\ \dots & \dots & \dots & \dots\\ \underset{\sim}{\chi_{R_1}}(y_s,z_1) & \underset{\sim}{\chi_{R_1}}(y_s,z_2) & \dots & \underset{\sim}{\chi_{R_1}}(y_s,z_m)\end{Vmatrix}.$$

На основі використання формул оцінки (6) – (7) можливо визначити значення індикатору економічної безпеки.

Скористаємось наведеними вище результатами для вирішення завдання вимірювання близькості нечітких векторів, скориставшись поняттям відстані Хеммінга для нечітких множин. При цьому відстанню Хеммінга для між нечіткими множинами A’ та B’ будемо визначати числом, яке дає формул

$$d(A', B')= \text{sum}(|\mu_A(u_r) - \mu_B(u_r)|), \quad r= 1, 2, .., q. \qquad (9)$$

$$0 <= d(A', B') <= q.$$

Для відстані Хеммінга з додатним відхиленням для нечіткої множини між нечіткими множинами A’ і B’ приймається така формула

$$d^{+}(A', B')= \text{sum}(a_r|\mu_A(u_r) - \mu_B(u_r)|), \quad r= 1, 2, .., q. \qquad (10)$$

в (10) індикатор a_r визначається так

$$a_r = 1, \quad \text{при} \quad \mu_A(u_r) => \mu_B(u_r)$$

$$0, \quad \text{при} \quad \mu_A(u_r) < \mu_B(u_r).$$

Модель (10) дозволяє розрізняти переваги показників $\mu_A(u_r)$ і $\mu_B(u_r)$. За рахунок чого можливо також проводити кодування характеристик нечітких змінних. Розглянемо це питання докладно.

Визначимо наступну процедуру перекодування нечітких показників μ_X: $X \rightarrow [0; 1]$ (6) до множини значені моделі НМХ {-1, +1}. На відміну від попередніх моделей (7) – (8) сформуємо нову процедуру представлення нечітких величин для НМХ. По перше, будемо ураховувати елементи метрик (9) – (10), оцінюючи значення функцій приналежності, вважаючи що нечітка величина A’ відповідає змінним зразків (шаблонів фундаментальної пам’яті), а величина B’ – визначає параметри приналежності вхідного «зонду», що необхідно класифікувати. При цьому будемо визначати значення коду таким індикатором H_r

$$f(|\mu_A(u_r) - \mu_B(u_r)|) =$$

$$H_r = 1, \quad \text{при} \quad \mu_A(u_r) <= \mu_B(u_r) \qquad (11)$$

$$-1, \quad \text{при} \quad \mu_A(u_r) > \mu_B(u_r). \; r= 1, 2,..,q \text{ (розмірність зразків).}$$

Можливо кодування, коли рахується $H_r = 0$, при $\mu_A(u_r) = \mu_B(u_r)$.

По друге, значення кодів (11) присвоюють зразкам (шаблонам класифікації) кожний раз, коли досліджується нових вхідний «зонд». Таким чином, в процедурі перекодування (11) виконується як для кожного зразка, так і для вхідного «зонду», тобто В'. Саме в цьому перетворенні, зміні представлення, множини зразків {А'} у відповідності до вхідних даних В' полягає особливість запропонованої процедури перекодування до форми НМХ. Зрозуміло, що при таких перетвореннях також залишається можливість однакового представлення кодами різних нечітких величин. Разом з тим такі можливості скорочуються, і залишаються вірними наведені вище твердження стосовно ролі кодування, яке необхідно поєднувати з метою застосування моделі НМХ – виконання класифікації нечітких даних. Необхідно ще раз наголосити, що початкові форми зразків залишаються незмінними і зберігаються в базі даних системи класифікації. Методики отримання таких нечітких представлень вхідних даних різноманітні, вони в цій роботі не розглядаються.

Наведемо приклад застосування процедури (11) для зонду В'та множини шаблонів {А'}

В' =(0.7 0.2 0.4 1 0.5) Коди H_r (В') = **(1 1 1 1 1)**, перша умова (11).

{А'}= { (0.3 0.5 1 0.2 0.6) /(0.7 0.8 0.4 1 0.5) / (0.1 0.4 0.2 0.5 0.9) }.

H_r (В') = {**(1 -1 -1 1 -1)/ (1 -1 1 1 1)/ (1 -1 1 1 -1)**}.

Приклад показує, що початкова нечітка модель класифікації стала *стандартною для мережі НМХ*. Крім того очевидно, що «зразок 2» буде мати найменшу відстань за (9), або за процедурою НМХ (1) – (5).

Інший варіант кодування (11) виконується в два етапи. На першому вхідні дані В' кодуються відповідно до (7), (8). На наступному за кодами H_r(В') перетворюються шаблонів {А'} у відповідності до процедури (11).

Процедура прямого застосування даних нечітких величин в НМХ. Як відзначалось раніше, модель НМХ (1) – (5) структура кодів вхідного вектору та зразків *не враховується* на повторних етапах послідовної процедури (4) – (5). В якості бази шаблонів тут використовують вектори оцінок показників для

μ_X (X → [0; 1]) кожного вхідного параметру {A'}, що залишаються незмінними. Вхідні «зонди» також мають вектори оцінок μ_X для всіх відповідних показників. Після закінченні її виконання, збіжності результатів розрахунків, визначається один (кілька нерозподільних моделлю НМХ зразків) зразок, до якого віднесено вхідний «зонд».

2.1.3. Розробка інтелектуальних процедур класифікації нейронних мереж Хеммінга на основі коефіцієнтів впевненості CF(X).

Модель коефіцієнтів довіри CF(X). Розглядається та досліджується модель кодування та процедура відображення невизначених даних у вигляді коефіцієнтів довіри CF(X): X → [-1, +1]. В ній характеристики неточних знань, оцінок ступеня упевненості, кодуються експертним чином, відповідним функціональним засобом або безпосередньо призначаються, що відповідає широкому класу завдань формування Баз Знань (БЗ) Експертних Систем (ЕС) [***]. Застосування такої моделі (форми представлення знань) зумовлено її широким вживанням і дослідженням можливостей, а також відомими функціональними та програмними засобами реалізації. Також область кодів {-1, 0, +1} входить до [-1, +1]. Позначимо модель мережі МНХ з показниками CF(X) як CFMHX. Вимоги та процедури до реалізації відповідають наведеним вище твердження щодо *процедури прямого застосування нечітких кодів даних* для НМХ..

Завдання наступного етапу виконаного дослідження полягало у проведенні чисельного експерименту з метою визначення можливостей представлених вище моделей та процедур відображення невизначених даних щодо виконання класифікації не точно визначених даних НМХ (1) – (5), з урахуванням формул (7) – (11). За результатами такого експерименту визначаються остаточно структура та інтелектуальні процедури програми оцінювання багато параметричних властивостей потоків послідовностей вхідних елементів сервісних систем.

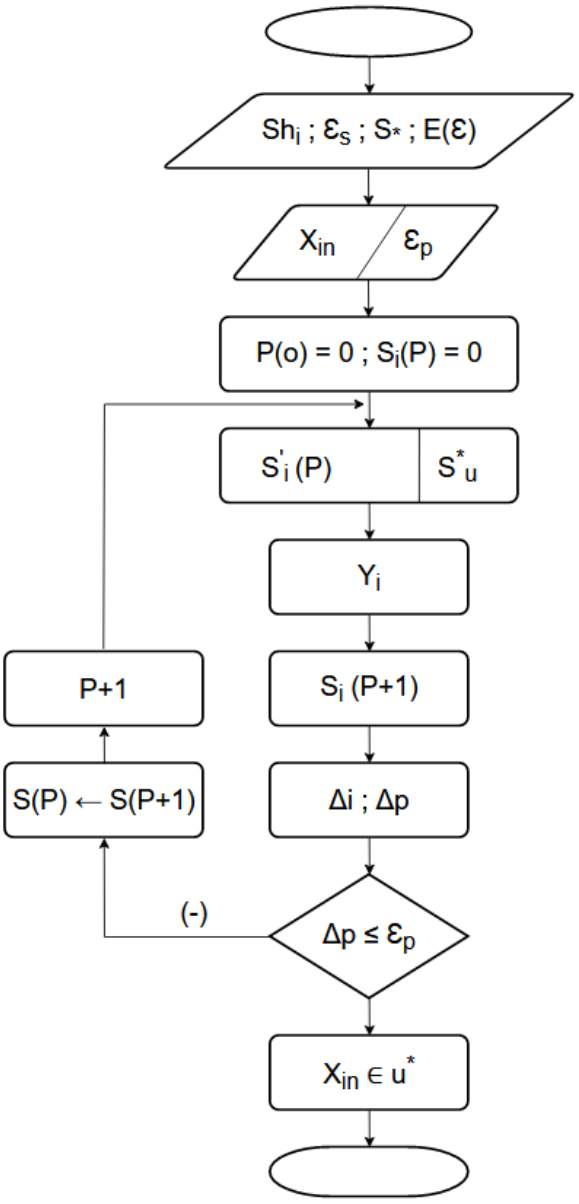


Рис. 2.5 Блок-схема алгоритму модифікованої процедури моделі асоціативної пам'яті для, призначені для моделей НМХ та CFMHX

Наведений вище аналіз завершує постановку завдання розробки та дослідження інтелектуальних засобів оптимального планування потоків за рахунок недетермінованих процедур, які застосовуються до мережі НМХ. Властивість «оптимальності рішення» забезпечується за рахунок «класифікації варіантів управління, шаблонів дії» відповідно вхідних елементів потоків даних. Загальний зміст застосованого у роботі інтелектуального методу оптимізації управління процесами СеС складається

з наступного. Функціонування СеС при обслуговування замовлень представляється у форомі системи типу диспетчерського керування на основі поточного стану. Елементам СеС і замовленням приписані певні ознаки, що характеризують їх загальні властивості, значення яких є неточно визначеними. База знань СеС складається з кінцевої множини схем, класів раціонального функціонування СеС, які відповідають певним значенням ознак, що характеризують раціональне обслуговування. На підставі застосування модифікованих інтелектуальних процедур НМХ та CFMHX для кожного вхідного елементу послідовностей замовлень визначається класс, що однозначно визначає «виконавця».

Для реалізації зазначеної форми оптимального управління у СеС необхідно мати відповідну структуру програмної реалізації, яка забезпечує утворення та функціонування зазначеного процесу обслуговування потоку замовлень.

2.2. Дослідження достовірності та числової ефективності інтелектуальних процедур класифікації за не точно визначеними даними.

Головною метою дослідження являється наступне.

1) Визначити для кожної моделі з не повністю визначеними даними можливості реалізації завдань класифікації, якщо в якості досліджуваного вектора ознак, «зонда», надходять безпосередньо шаблони.
2) Визначити можливості реалізації завдань класифікації, якщо в якості досліджуваних векторів ознак використовуються вектори, які перекривають увесь діапазон зміни значень варіювання.
3) Визначити порівняльну ефективність моделей кодування нечітких вхідних векторів щодо їх запропонованих варіантів.
4) Перевірка класифікації вхідних векторів для граничних значень областей можливих значень.

5) Перевірка кожної моделі з не повністю визначеними даними можливості реалізації завдань класифікації, якщо в якості значень вхідних векторів використовуються «0», тобто невизначені араметри.
6) Визначити порівняльну ефективність моделей коефіцієнтів впевненості та нечітких величин.
7) Систематизувати результати досліджень

Основні дослідження достовірності та числової ефективності інтелектуальних процедур класифікації приведені в розд. 4.

Приведемо приклади реалізації окремих завадань на деяких вибіркових прикладах.

Приклад №1 призначений для дослідження можливостей і оцінки ефективності класичної моделі Хеммінга, але якщо при виборі шаблонів використовуються дані в діапазоні [-1; 1]. Тобто має місце модель коефіцієнтів впевненості. При цьому сукупність шаблонів для класифікації приведена в табл. 1, а вхідний вектор в табл. 2, який також відповідає діапазону варіювання. На вхід, так само, подаються дані від [-1; 1]. Табл. 3 демонструє процес пошуку шаблону. Після кожної ітерації за рахунок функції активації відсіюється шаблони, які не підходять (мають від'ємний результат). Найбільш підходящим шаблоном, результатом класифікації, є той, яких залишається з позитивною оцінкою $K*S > 0$ після кроку ітерації.

Таблиця 1. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	-1	0,5	-0,8	-0,6	0,5	0,8
K2	0,7	0,3	-0,3	0,4	0,9	0,1
K3	1	-0,8	-0,5	-1	0,7	-0,3
K4	-0,8	0,4	0,6	0,9	-0,6	-0,9
K5	0,5	-0,6	1	0,4	-0,9	0,5

Таблиця 2. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
-0,6	0,6	0,8	1	-0,3	-0,7

Таблиця 3. Ітерації для даних №1:

1	S	-0,525	-0,21	-1,24	1,445	0,23
	K*S	-0,58125	-0,1875	-1,475	1,88125	0,3625
2	S	0	0	0	1,88125	0,3625
	K*S	-0,56094	-0,5609375	-0,560938	1,790625	-0,10781

Таблиці табл. 1 – табл. 3 показують, що модель коефіцієнтів впевненості дозволила вірно та ефективно визначити шаблон, результат ітерацій: шаблон №4, очікуваний результат: шаблон №4.

Наступний приклад, табл. 4, табл. 5, демонструють процедуру для даних №2.

Таблиця 4. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
0,7	-0,5	-0,8	-0,6	1	-0,1

Таблиця 5. Ітерації для даних №2:

1	S	0,235	0,615	1,415	-1,145	-0,67
	K*S	0,18125	0,65625	1,65625	-1,54375	-0,95
2	S	0,18125	0,65625	1,65625	0	0
	K*S	-0,39688	0,196875	1,446875	-0,623438	-0,62344
3	S	0	0,196875	1,446875	0	0
	K*S	-0,41094	-0,1648438	1,3976563	-0,410938	-0,41094

Таблиці табл. 4 – табл. 5 також показують, що модель коефіцієнтів впевненості вірно та ефективно визначила шаблон №3, який був очікуваний.

Наступний варіант демонстує роботу процедури при вхідних векторах з багатьма невизначеностями, коли входи мають «0»

Таблиця 6. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
1	0	1	0	1	0

Таблиця 7. Ітерації для даних №3:

1	S	-0,65	0,65	0,6	-0,4	0,3
	K*S	-0,9375	0,6875	0,625	-0,625	0,25
2	S	0	0,6875	0,625	0	0,25
	K*S	-0,39063	0,46875	0,390625	-0,390625	-0,07813
3	S	0	0,46875	0,390625	0	0
	K*S	-0,21484	0,3710938	0,2734375	-0,214844	-0,21484
4	S	0	0,3710938	0,2734375	0	0

	K*S	-0,16113	0,3027344	0,1806641	-0,161133	-0,16113
5	S	0	0,3027344	0,1806641	0	0
	K*S	-0,12085	0,2575684	0,1049805	-0,12085	-0,12085
6	S	0	0,2575684	0,1049805	0	0
	K*S	-0,09064	0,2313233	0,0405884	-0,090637	-0,09064
7	S	0	0,2313233	0,0405884	0	0
	K*S	-0,06798	0,2211762	-0,017242	-0,067978	-0,06798

Таблиці табл. 6 – табл. 7 демонструють значні можливості моделі коефіцієнтів впевненості і при таких нетипових вхідних векторах, коли вірно був визначений шаблон №2, що був очікуваний. Разом з тим процес затребував більше ітерацій, а занчення показника виявилося суттєво меньшим, ніж, наприклад в табл. 5.

Призначення прикладу №2 – теж аналіз поведінки та результативності класичної моделі Хеммінга. В цьому прикладі, при виборі найбільш підходящих шаблонів використовуються такі дані: -1 (позначає відсутність параметра); 1 (параметр присутній). Шаблони класифікації знаходяться в табл. 8, вхідний вектор - в табл. 9. В якості вхідних даних використовуються ті ж числа, що і у шаблонах - -1 та 1. Ітерації пошуку відповідного шаблону видно у табл. 10. Шаблони з від'ємний результат припиняють враховуватися при виборі результату після кожної ітерації. Найпридатнішим шаблоном є той, в якого K*S > 0 після всіх ітерацій.

Таблиця 8. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	-1	1	-1	1	-1	1
K2	1	-1	1	-1	1	-1
K3	1	1	1	-1	-1	-1
K4	1	-1	-1	1	1	-1
K5	-1	-1	1	1	1	-1

Таблиця 9. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
-1	1	-1	1	1	1

Таблиця 10. Ітерації для даних №1:

1	S	2	-2	-2	0	0
	K*S	3	-2	-2	0,5	0,5

З таблиць табл. 8 – табл. 10 видно, що модель правильно визначила шаблон, результат ітерацій: шаблон №1, очікуваний результат: шаблон №1.

Наступний приклад, табл. 11, табл. 12, демонструють процедуру для даних №2.

Таблиця 11. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
1	-1	-1	-1	1	-1

Таблиця 12. Ітерації для даних №2:

1	S	-2	2	0	2	0
	K*S	-3	2	-0,5	2	-0,5

Таблиці табл. 11 – табл. 12 показують, що через лімітовану можливість описати унікальні вхідні дані, модель може визначити декілька шаблонів як правильні, результат ітерацій: шаблон №2 або №4, очікуваний результат: шаблон №2 або №4.

Ще один варіант, табл. 13, табл. 14, показує процедуру для даних №3.

Таблиця 13. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
-1	1	1	1	1	-1

Таблиця 14. Ітерації для даних №3:

1	S	0	0	0	0	2
	K*S	-0,5	-0,5	-0,5	-0,5	2

За даними з таблиць табл. 13 – табл. 14 бачимо, що модель придатна для знаходження правильних шаблонів, результат ітерацій: шаблон №5, очікуваний результат: шаблон №5.

В прикладі №3 показане визначення ефективності нечіткої моделі Хеммінга при знаходженні шаблонів. В якості вхідних даних використовуються числа в діапазоні [0; 1]. Задані шаблони для класифікації приведені в табл. 15, вхідний вектор в табл. 16. Вхідні дані теж знаходяться в діапазоні [0; 1]. Процес пошуку шаблону показано в табл. 17. Шаблони, які мають від'ємний результат відсіюються після кожної ітерації. Шаблоном, найбільш придатним для заданих вхідних даних є той, який має позитивну оцінку K*S > 0 після ітерацій.

Таблиця 15. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	0,8	0,7	0,9	0,9	1	0,7
K2	0,1	1	0	0,5	0,6	1
K3	1	0,1	0,3	0,7	0,4	0,2
K4	0	0,5	0,1	1	0,3	0
K5	0,6	0,3	0,5	0,2	0,5	0,1

Таблиця 16. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
0,6	0,5	0,7	0,7	0,8	0,6

Таблиця 17. Ітерації для даних №1:

1	S	0,4	0,3	0,35	0,35	0,3
	K*S	0,3125	0,2	0,25	0,25	0,2
2	S	0,3125	0,2	0,25	0,25	0,2
	K*S	0,25	0,121875	0,171875	0,171875	0,121875
3	S	0,25	0,121875	0,171875	0,171875	0,121875
	K*S	0,20703125	0,059375	0,109375	0,109375	0,059375
4	S	0,20703125	0,059375	0,109375	0,109375	0,059375
	K*S	0,1796875	0,007617187	0,057617188	0,057617188	0,007617187
5	S	0,1796875	0,007617187	0,057617188	0,057617188	0,007617187
	K*S	0,165283203	-0,037304688	0,012695313	0,012695313	-0,037304688
6	S	0,165283203	0	0,012695313	0,012695313	0
	K*S	0,162109375	-0,041320801	-0,028625488	-0,028625488	-0,041320801

Таблиці табл. 15 – табл. 17 демонструють, що нечітка модель Хеммінга дала можливість вірно визначити шаблон, що найкраще співвідноситься до

вхідних даних, результат ітерацій: шаблон №1, очікуваний результат: шаблон №1.

Табл. 18, табл. 19 ілюструють процедуру для даних №2.

Таблиця 18. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
0,3	0,8	0,3	0,7	0,8	0,8

Таблиця 19. Ітерації для даних №2:

1	S	0,4	0,4	0,35	0,35	0,25
	K*S	0,3	0,3	0,25	0,25	0,15
2	S	0,3	0,3	0,25	0,25	0,15
	K*S	0,225	0,225	0,175	0,175	0,075
3	S	0,225	0,225	0,175	0,175	0,075
	K*S	0,16875	0,16875	0,11875	0,11875	0,01875
4	S	0,16875	0,16875	0,11875	0,11875	0,01875
	K*S	0,1265625	0,1265625	0,0765625	0,0765625	-0,0234375
5	S	0,1265625	0,1265625	0,0765625	0,0765625	0
	K*S	0,094921875	0,094921875	0,044921875	0,044921875	-0,031640625
6	S	0,094921875	0,094921875	0,044921875	0,044921875	0
	K*S	0,071191406	0,071191406	0,021191406	0,021191406	-0,023730469
7	S	0,071191406	0,071191406	0,021191406	0,021191406	0
	K*S	0,053393555	0,053393555	0,003393555	0,003393555	-0,017797852
8	S	0,053393555	0,053393555	0,003393555	0,003393555	0
	K*S	0,040045166	0,040045166	-0,009954834	-0,009954834	-0,013348389

З таблиць табл. 18 – табл. 19 бачимо, що модель може позначити кілька шаблонів як правильні, результат ітерацій: шаблон №1 або №2, очікуваний результат: шаблон №2.

Зразок з даними №3 показаний в таблицях табл. 20, табл. 21.

Таблиця 20. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
0,2	0,3	0,3	0,9	0,6	0,1

Таблиця 21. Ітерації для даних №3:

1	S	0,45	0,3	0,35	0,45	0,25
	K*S	0,3375	0,1875	0,2375	0,3375	0,1375
2	S	0,3375	0,1875	0,2375	0,3375	0,1375
	K*S	0,253125	0,103125	0,153125	0,253125	0,053125
3	S	0,253125	0,103125	0,153125	0,253125	0,053125
	K*S	0,18984375	0,03984375	0,08984375	0,18984375	-0,01015625
4	S	0,18984375	0,03984375	0,08984375	0,18984375	0
	K*S	0,142382813	-0,007617188	0,042382813	0,142382813	-0,047460938
5	S	0,142382813	0	0,042382813	0,142382813	0
	K*S	0,106787109	-0,035595703	0,006787109	0,106787109	-0,035595703
6	S	0,106787109	0	0,006787109	0,106787109	0
	K*S	0,080090332	-0,026696777	-0,019909668	0,080090332	-0,026696777

З таблиць табл. 20 – табл. 21 видно, що модель цілком придатна для знаходження потрібних шаблонів, результат ітерацій: шаблон №1 або №4, очікуваний результат: шаблон №4.

2.3. Особливості застосування інтелектуальних процедур оптимізації послідовностей замовлень за не точно визначеними даними.

В розділі приведені результати досліджень, які представляють рекомендації стосовно особливостей застосування інтелектуальних процедур оптимізації послідовностей замовлень за неточно визначеними даними. Рекомендації визначають раціональну область застосуввання розроблених інтелектуальних процедур класифікації для нечітких моделей вихідних даних і моделей на основі коефіцієнтів впевненості CF(X), які використовують алгоритми на основі асоціативної пам’яті.

Числові дослідження форм представлення первинних даних показали певні переваги моделі CF(A), яка *завжди* забезпечувала потрібний результат класифікації, при тому що в моделі з НВ результати класифікації не завжди були однозначними.

Одна із важдивих можливостей щодо формування та розроблення нечітких інтелектуальниїх процедур класифікації замовлень у СеС полягає в використанні моделей перекодування НМ, засобами чого вхідна інформація набуває форому, яку обробляють стандартні нейронні мережі Хеммінга. Виконання у розділі дослідження з метою встановлення порівняльної ефективності запропонованих алгоритмів удосконалення мережі Хеммінга, а також програмна реалізація розроблених моделей класифікації об'єктів з неточно визначеними параметрами МХН, показали високу достовірність та числову ефективність результатів проведених експериментів. Як підсумок, процедура ОПЗСеС на основі модифікованих моделей асоціативної пам'яті МХН може бути широко застосовані в завданнях оптимізації послідовностей замовлень у сервісних системах, які мають неточно визхначені характеристики замовлень.

Висновки до розділу 2

У розділі проведені розробка та дослідження можливостей реалізації завдань щодо оптимізації потоків в сервісніх системах з використанням інтелектуальних процедур асоціативної пам'яті, які базуються на Нейронних мережах Хеммінга. При цьому виконана розробка та дослідження процедур оптимізації, яка складалась із наступного

- постановки нових завдань оптимізації потоків замовлень сервісних систем з неточно визначеними даними,
- розробка інтелектуальних процедур класифікації на основі нечітких моделей неточно визначених даних,
- розробка інтелектуальних процедур класифікації нейронних мереж Хеммінга на основі коефіцієнтів впевненості CF(X).

На підставі результатів розробки проведені дослідження для оцінювання достовірності та числової ефективності інтелектуальних процедур класифікації за неточно визначеними даними, які сформовані у роботі. Результати досліджень повністю підтвердили достовірність ефективність

інтелектуальних процедур класифікації для нечітких моделей вихідних даних, а також для представленя початкової інформації за допомогою коефіцієнтів впевненості CF(X). При цьому було встановлено, що модифіковані моделі нейронних мереж Хеммінга на основі коефіцієнтів впевненості CF(X) мають перевагу перед нечіткими моделями, що запропоновані в дипломній роботі.

В розділі представлені розроблені моделі щодо форм перекодування НМ, а також встановлена їх порівняльна ефективність. Визначено, що запропоновані моделі дозволяють програмну реалізацію розроблених моделей класифікації об'єктів з неточно визначеними параметрами МХН. Важливим висновком являється те, що результати проведених числових експериментів, а також процедура ОПЗСеС на основі модифікованих моделей асоціативної пам'яті МХН, підтвердили практично достовірність результатів щодо нечіткої класифікації на основі моделей Хеммінга. Це твердження сає саме дослідницький рівень обгрунтування. Тому що в класичних НХ неточна форма первинних даних нгавіть не передбачається. .

В розділі також представлені рекомендації стосовно особливостей застосування інтелектуальних процедур оптимізації послідовностей замовлень за неточно визначеними даними, що визначають раціональну область застосуввання розроблених інтелектуальних процедур класифікації для нечітких та моделей вихідних даних на основі асоціативної пам'яті.

3. РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ЗАВДАНЬ ОПТИМІЗАЦІЇ ПОТОКІВ ЗАМОВЛЕНЬ В ОБСЛУГОВУЮЧИХ СИСТЕМАХ НА ОСНОВІ ІНТЕЛЕКТУАЛЬНИХ ПРОЦЕДУР НЕЙРОННОЇ МЕРЕЖІ ХЕММІНГА.

3.1. Постановка та формалізація завдань дослідження інтелектуальних процедур нейронної мережі Хеммінга з неточно визначеними даними.

Формалізація задачі представлена на рівні зовнішнього проектування наведена у вигляді діаграми варіантів використання. Користувач взаємодіє із системою за допомогою варіантів використання. Ці варіанти дають опис можливостей, які програмна система надає акторам.

Діаграма прецедентів визначає варіанти використання рис. 3.1

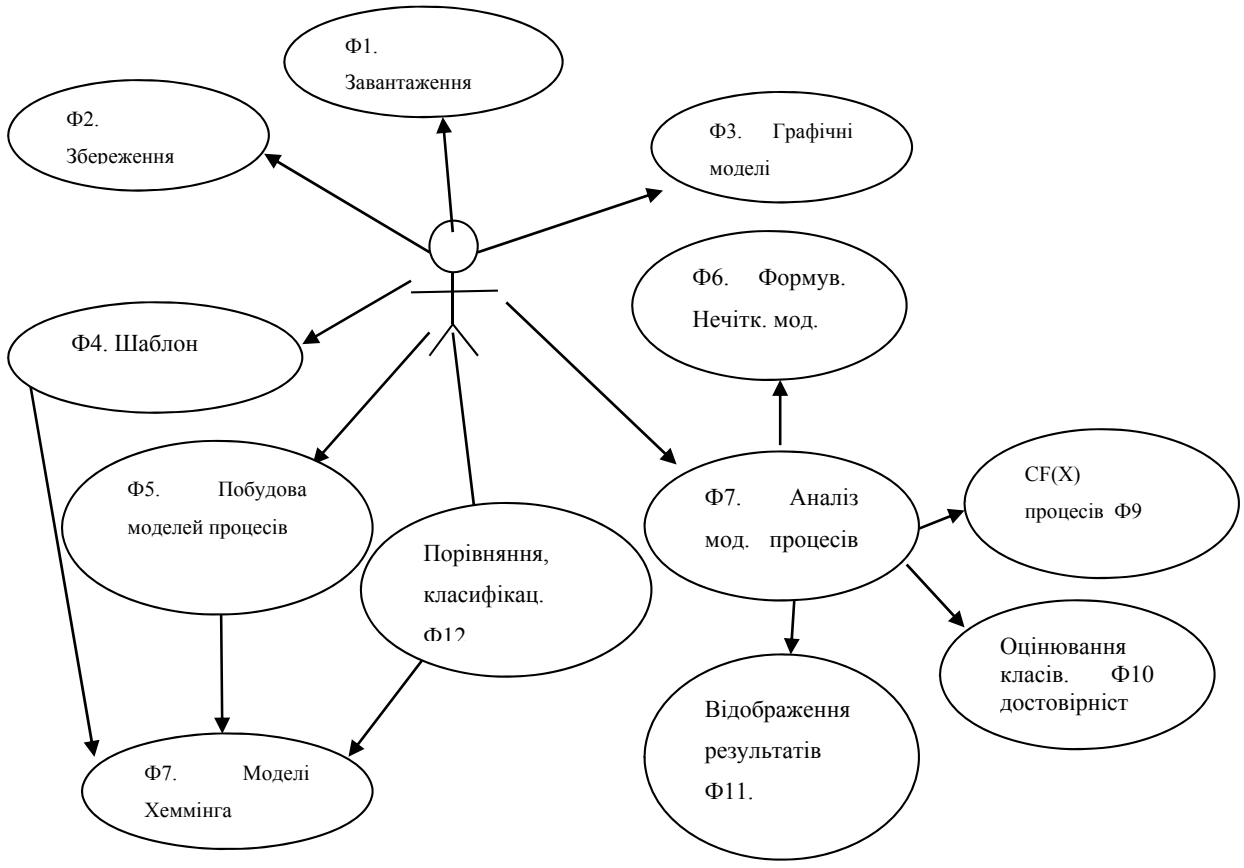


Рисунок 3.1 – Діаграма варіантів використання при дослідженні СеС

На діаграмі представлені наступні функції програми: Ф1 – процедури отриманні вихідних даних для дослідження; Ф2 – функція щодо збереження даних та результатів досліджень; Ф3 – відображення графіків моделей процесів; Ф4 – Формування шаблонів процесів; Ф5 - побудова класів шаблонів використовуючи різні форми невизначеності; Ф6 – формування нечітких модедей процедур Хеммінга; Ф7 – Налаштування моделей, представлених СеС; Ф8 - Аналіз моделей процесів; Ф9 – CF(X) процедури Хеммінга; Ф10- Оцінювання класів шаблонів; Ф11 – Відображення результатів; Ф12 – процедури порівняльного аналізу.

3.2. Розробка структури програми.

В якості базової архітектури програмного комплексу було прийнято рішення використати клієнт-серверний архітектурний підхід. Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів. Така архітектура визначає такі типи компонентів набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них; набір клієнтів, які використовують сервіси, що надаються серверами; мережа, яка забезпечує взаємодію між клієнтами та серверами. Правила взаємодії між клієнтом і сервером називаються протоколом обміну (протоколом взаємодії)

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов’язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій: рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд; прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації; рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дволанкова клієнт-серверна архітектура передбачає взаємодію двох програмних модулів — клієнтського та серверного. В залежності від того, як

між ними розподіляються наведені вище функції, розрізняють: модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення; модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта.

Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін. Трьохланкова клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка застосунку. Програми проміжного рівня можуть функціювати під управлінням спеціальних серверів застосунків, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Дволанкова архітектура простіша, так як всі запити обслуговуються одним сервером, але саме через це вона менш надійна і висуває підвищені вимоги до продуктивності сервера. Триланкова архітектура складніша, але завдяки тому, що функції розподілені між серверами другого і третього рівня, ця архітектура проявляє: високий ступінь гнучкості і масштабованості. високу безпеку (тому що захист можна визначити для кожного сервісу або рівня). високу продуктивність (тому що завдання розподілені між серверами).

Прикладом клієнт-серверної взаємодії є сервіс WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Основна ідея архітектури «клієнт-сервер» полягає в поділі мережевого додатки на кілька компонентів, кожен з яких реалізує специфічний набір сервісів. Компоненти такого додатку можуть виконуватися на різних комп'ютерах, виконуючи серверні і/або клієнтські функції. Це дозволяє підвищити надійність, безпеку і продуктивність мережевих додатків і мережі в цілому.

Серверна частина відповідатиме за обробку та збереження інформації. Оскільки така система може у подальшому виконувати обробку інформації паралельно до виконання інших операцій, треба обирати необхідний стек технологій саме для спрощення подальшого масштабування системи.

Клієнтська частина може бути виконана на будь-якій технології, що підтримую реалізацію інтерфейсу обміну данними з сервером за допомогою RESTful API.

REST — архітектурний стиль взаємодії компонентів розподіленої програми у мережі. Іншими словами, REST - це набір правил того, як програмісту організувати написання коду серверної програми, щоб усі системи легко обмінювалися даними і програму можна було масштабувати

Взаємодію головних елементів між частинами архітектурної моделі, представлено на рис. 3.2

Рисунок 3.2 – Взаємодія головних елементів архітектурної моделі програми «Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем».

3.3. Внутрішнє проектування елементів програми.

3.3.1. Вибір мови програмування.

При виборі мови програмування ураховувалися фактори стосовно можливості програмного середовища, а також можливостей безпосередньо мови програмування. Саме ці два головних критерії ураховувалися при визначенні мови програмування.

Для розробки програмного комплексу було обране програмне середовище WebStorm.

При розробках програми «Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем» використано мову програмування Python з боку серверної частини програми, клієнтська частина реалізована за допомогою технології JavaScript / React.js / Redux.

Ієрархія та взаємодія класів програмної системи

Взаємодія та проектування класів описано UML діаграмою (рис. 3.3.1).

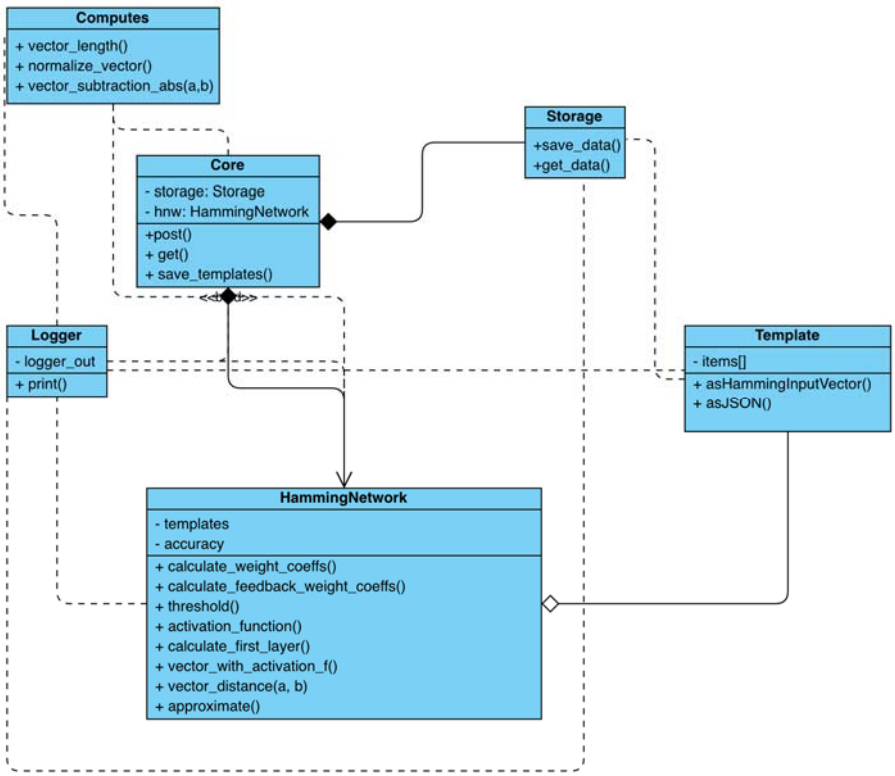


Рисунок 3.3.1 – Діаграмма класів серверної (обчислювальної) частини програмного комплексу

- HammingNetwork – клас, що реалізовує нейронну мережу Хеммінга, описані методи використовуються для отримання апроксимації належності вхідного вектору до одного з шаблонів зі списку templates.
- Template – клас, що описує структуру кожного шаблону, за необхідністю можна трансформувати дані як до типу зберігання, так і до типу, що використовує клас HammingNetwork.
- Logger – клас, що описує логування операцій, виконаних програмою.
- Computes – клас, в якому описані методи роботи з векторами: довжина, нормалізація вектору, операції над векторами.
- Storage – клас, що описує методи роботи з базою знань
- Core – клас, що отримує на вхід дані, використовуючи REST протокол. Зв'язує між собою клієнтські дані та обчислення.

Взаємодія та проектування класів клієнтської частини наведено UML у діаграммі (рис. 3.3.2):

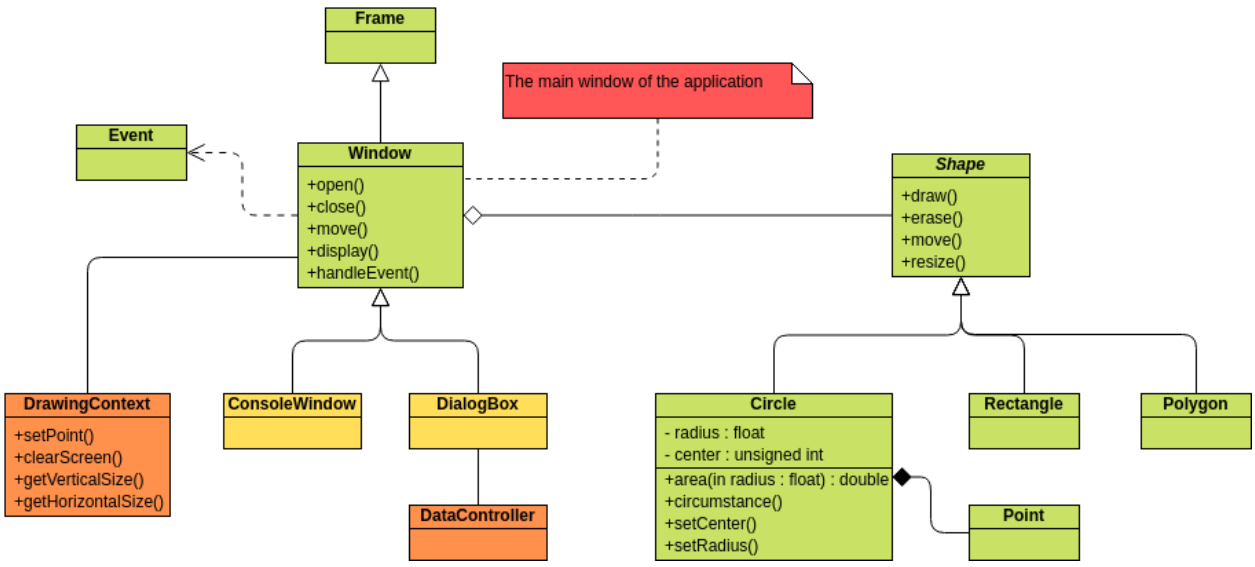


Рисунок 3.3.2. – Діаграмма класів клієнтської (взаємодійної та презентаційної) частини програмного комплексу.

3.3.2. Розробка архітектури взаємодії модулів програмної системи класифікації за не точно визначеними даними

При проектуванні внутрішньої структури системи використовувалися такі загальні принципи об’єктно-орієнтованого проектування:

- забезпечення слабкої зв'язності об'єктів що взаємодіють. За ним чим менше об'єкти знають один про одного, тим гнучкіше створювана програмна система. Кожному компоненту немає необхідності «знати» внутрішній устрій іншого;
- забезпечення взаємодії тільки з «близькими» компонентами. Принцип спрямований на мінімальну інформованість – при проектуванні класу треба звертати увагу на кількість класів, з якими буде відбуватися взаємодія. Чим менше таких класів, тим гнучкіше система;
- голлівудський принцип – не викликайте нас, ми самі вас викличемо. За Фаулеру - це синонім принципу IoC. Згідно ньому компоненти високого рівня (наприклад, інтерфейси) визначають за компоненти низького рівня (реалізації), як і коли їм підключатися до системи. Інші автори Head First Design Patterns допускають, що за цим принципом компоненти низького рівня можуть приймати участь в обчисленнях без формування залежностей з компонентами високого рівня. Саме в цьому полягає відмінність такого проектування від більш жорсткого принципу IoC;
- розділення інтерфейсу, за яким перевага віддається тому проекту, коли у системі використовується велика кількість спеціалізованих інтерфейсів замість кількох універсальних.

На рис. 3.3.2-а приведено структуру програмного комплексу, призначеного для виконання інтелектуальних процедур нечіткої класифікації та упорядкування недетермінованих послідовностей багатопараметричних об’єктів, а також для дослідження таких процедур. На схемі структури програмного комплексу позначено наступне.

Б1 – база даних із управління моделями оцінювання властивостей та упорядкування об’єктів процесів на основі класифікації.

Б1.1 – вибір послідовностей даних, які представляють моделі процесів, перегляд властивостей об’єктів.

Б1.2 – збереження / вилучення моделей в БД,

Б2 – формування моделей аналізу об’єктів.

Б2.1 – формування структури шаблонів (класів оцінювання даних) області аналізу (процесів).

Б2.1.2 – формування та збереження шаблонів детермінованих моделей класифікації в БД.

Б2.1.2.1 – формування та збереження шаблонів нечітких моделей класифікації в БД.

Б2.2 – перевірка повноти та коректності системи моделей класифікації

Б2.3 – відображення шаблонів моделей класифікації даних процесів.

Б3 – оцінювання властивостей об’єктів досліджуваних процесів

Б3.1 – процедури оцінювання показників щодо властивостей об’єктів

Б3.2 – відображення властивостей об’єктів

Б4 – процедури класифікації об’єктів процесів аналізу

Б4.1 – моделі детермінованих шаблонів (-1 / +1), HMX

Б4.1.1 – налаштування, розрахунок класу об’єктів аналізу

Б4.1.2 – оцінювання та інтерпретація результатів класифікації

Б4.1.3 – відображення результатів щодо класу об’єктів аналізу

Б4.2 - моделі нечітких шаблонів [0; 1], FNMX

Б4.2.1 – налаштування, розрахунок класу за нечіткими даними об’єктів аналізу.

Б4.2.2 – оцінювання та інтерпретація результатів нечіткої класифікації

Б4.2.3 – відображення результатів щодо класу нечітких об’єктів аналізу

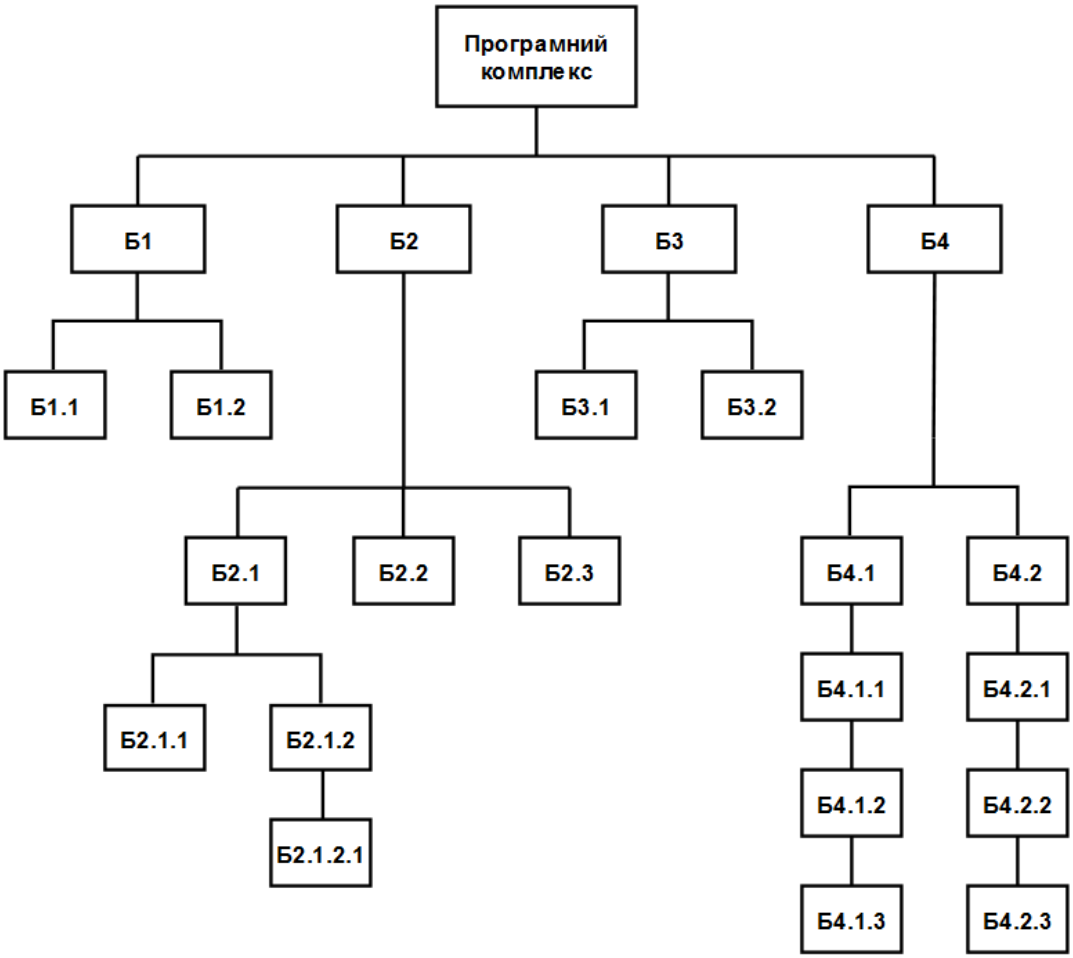


Рисунок 3.3.2-а – Структура програмного комплексу із інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем

3.3.3. Проектування програмної системи.

Проектуючи програмний комплекс, треба передбачити, що інтерфейс веб-серверу буде таким, що інші сервіси зможуть використовувати його. Модульне розширення передбачає, що сам сервіс не буде виконуватись як програма, але його вихідний код буде використовуватись іншим сервісом. Обидва методи мають ряд переваг і недоліків. В модульній взаємодії швидкість взаємодії швидша, так як в сервісній використовується мережа, проте при модульній взаємодії ціла система може працювати повільніше, так сервіси можуть виконуватись на різних ЕВМ.

При сервісній взаємодії контексти процесів не переплітаються, так як модульна взаємодія має загальну пам’ять. При сервісній взаємодії програма може використовувати дані іншої програми, причому якщо програма представлена в скомпільованому вигляді модульна взаємодія з нею неможлива. При модульній взаємодії поведінку модуля можна змінити або перевизначити, що набагато тяжче зробити з сервісом. При сервісній взаємодії сервіси можуть представляти загальний API або один сервіс буде звертатися до іншого. Також при сервісній взаємодії сервіси можуть використовувати загальні ресурси, такі як файлова система, пам’ять, база даних, інший сервіс. Діаграма розгортання при сервісній взаємодії для системи обміну інформацією повинна мати вид, як на рисунку 3.3.4

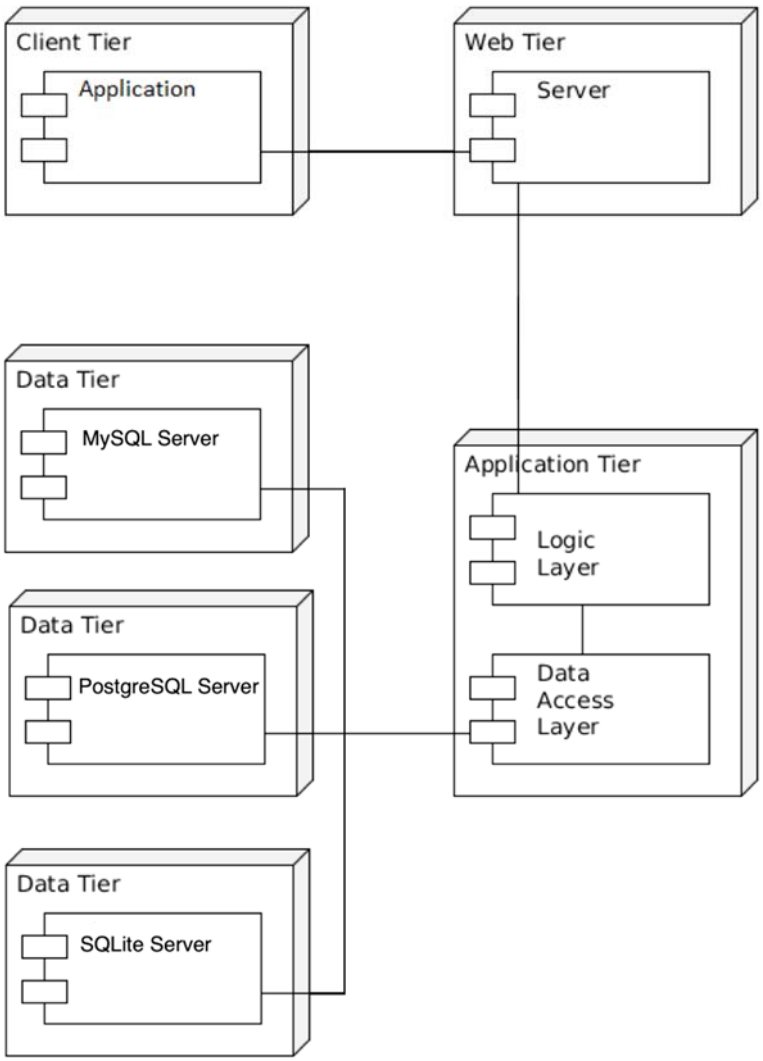


Рисунок 3.3.4 – Діаграма розгортання серверної частини програмного комплексу

3.4.Розробка інтерфейсу користувача програми.

При проектуванні інтерфейсу застосовувався принцип інтуїтивної зрозумілості, при цьому інтерфейс був не нагромаджений великою кількістю елементів. Для цього було створено кілька форм, кожна з яких виконує свою роль. А саме - Головне вікно програмного комплексу дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем , вікно меню, призначене для введення параметрів методів аналізу, вікно для відображення шаблонів класифікації.

3.5. Тестування та налагодження програми.

3.5.1. Опис головних методів тестування та налагодження.

Етапи тестування та відлагодження програм є найбільш трудомісткими в процесах розробки програм. Мета тестування – це виявлення помилок у роботі програмного комплексу. Виявлення синтаксичних помилок в більшості виконується за допомагою компілятора, вони можуть виправляються ще на момент написання. Для виявлення логічних помилок, помолок типізації, помилок узгодженності параметрів плоцндур та ін. можна отримати лише дослідивши роботу алгоритму, шляхом порівння очікуванних результати роботи з спеціально сформованими вихідними даними щодо роботи алгоритму.

При виявленні логічної помилки її необхідно локалізувати та виправити. Для цього на практиці використовують процедури налагодження, коли знають в якому алгоритмі знаходиться локалізована логічна помилка. При відлагодженні перевіряється робота (виконання та дані) кожної команди, та команда алгоритму, яка не відповідає встановленим вимогам, і буде розглядатися як можлива логічна помилка.

Загальними та ефективними методами тестування алгоритму являються метод «чорної скриньки» та метод «білої скриньки».

При застосуванні методу «чорної скриньки» полягають що алгоритм певної функції невідомий, а відомі лише вхідні та очікувані вихідні данні функції, що підходить для виявлення логічної помилки у функції.

Метод «білої скриньки» передбачає що алгоритм реалізації функції відомий, а система вхідних данних будуються так, щоб покрити максимальну кількість гілок виконання алгоритму. Таким чином виявляється ділянка алгоритму, де можливо знаходиться логічна помилка. Піісля виявлення помилки починається етап відлагодження для виправлення помилки, з подальшим повторним тестуванням.

У розділі наведено елементи щодо опису процесу проектування програмного комплексу із дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем.

4. ДОСЛІДЖЕННЯ ДОСТОВІРНОСТІ ТА ЧИСЛОВОЇ ЕФЕКТИВНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОПТИМІЗАЦІЇ ПОСЛІДОВНОСТЕЙ ЗАМОВЛЕНЬ НА ОСНОВІ ПРОЦЕДУР КЛАСИФІКАЦІЇ ЗА НЕТОЧНО ВИЗНАЧЕНИМИ ДАНИМИ.

4.1. Постановка завдань дослідження.

4.1.1. Опис програмно-апаратного середовища.

На рис. 4.1 приведено структуру програмного комплексу, призначеного для виконання інтелектуальних процедур нечіткої класифікації та упорядкування недетермінованих послідовностей багатопараметричних об’єктів, а також для дослідження таких процедур. На схемі структури програмного комплексу позначено наступне.

Б1 – база даних із управління моделями оцінювання властивостей та упорядкування об’єктів процесів на основі класифікації.

Б1.1 – вибір послідовностей даних, які представляють моделі процесів, перегляд властивостей об’єктів.

Б1.2 – збереження / вилучення моделей в БД,

Б2 – формування моделей аналізу об’єктів.

Б2.1 – формування структури шаблонів (класів оцінювання даних) області аналізу (процесів).

Б2.1.2 – формування та збереження шаблонів детермінованих моделей класифікації в БД.

Б2.1.2.1 – формування та збереження шаблонів нечітких моделей класифікації в БД.

Б2.2 – перевірка повноти та коректності системи моделей класифікації

Б2.3 – відображення шаблонів моделей класифікації даних процесів.

Б3 – оцінювання властивостей об’єктів досліджуваних процесів

Б3.1 – процедури оцінювання показників щодо властивостей об’єктів

Б3.2 – відображення властивостей об’єктів

Б4 – процедури класифікації об’єктів процесів аналізу

Б4.1 – моделі детермінованих шаблонів (-1 / +1), НМХ

Б4.1.1 – налаштування, розрахунок класу об’єктів аналізу

Б4.1.2 – оцінювання та інтерпретація результатів класифікації

Б4.1.3 – відображення результатів щодо класу об’єктів аналізу

Б4.2 - моделі нечітких шаблонів [0; 1], FNMX

Б4.2.1 – налаштування, розрахунок класу за нечіткими даними об’єктів аналізу.

Б4.2.2 – оцінювання та інтерпретація результатів нечіткої класифікації

Б4.2.3 – відображення результатів щодо класу нечітких об’єктів аналізу

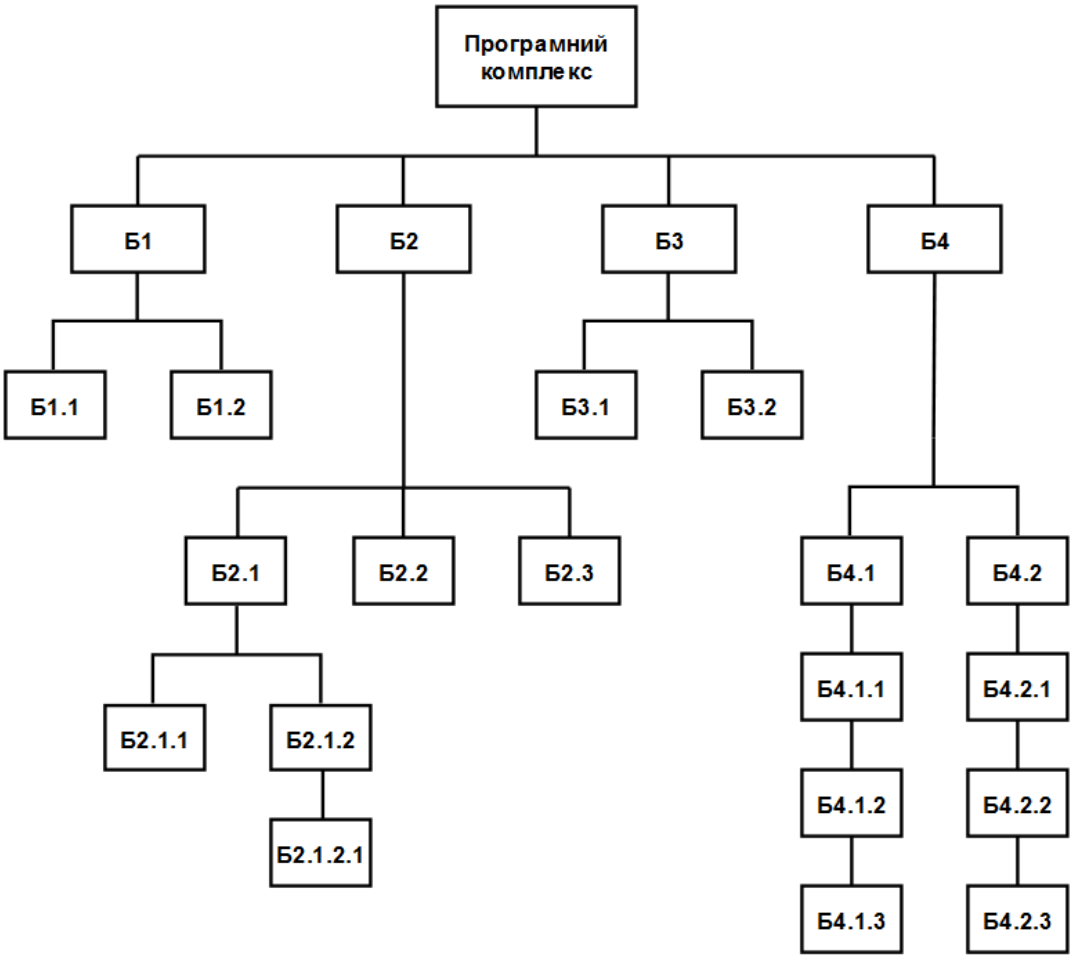


Рисунок 4.1 – Структура програми дослідження інтелектуальних процедур нечіткої класифікації на основі моделі мережі Хеммінга

Програмний комплекс класифікація:

Управління даними, Дослідження проводились з використанням наступного апаратного середовища:

- процесором AMD Ryzen 2200g з тактовою частотою 3.5 ГГц;
- об’єм оперативної пам’яті 16 Гб; об’єм жорсткого диску 1Тб;
- дискретна відео карта Radeon 5700 XT;

4.1.2. Дослідження модулю процедур класифікації на основі нечітких моделей даних об’єктів.

Приклад №1 призначений для дослідження можливостей і оцінки ефективності класичної моделі Хеммінга, але якщо при виборі шаблонів використовуються дані в діапазоні [-1; 1]. Тобто має місце модель коефіцієнтів впевненості. При цьому сукупність шаблонів для класифікації приведена в табл. 1, а вхідний вектор в табл. 2, який також відповідає діапазону варіювання. На вхід, так само, подаються дані від [-1; 1]. Табл. 3 демонструє процес пошуку шаблону. Після кожної ітерації за рахунок функції активації відсіюється шаблони, які не підходять (мають від’ємний результат). Найбільш підходящим шаблоном, результатом класифікації, є той, яких залишається з позитивною оцінкою $K*S > 0$ після кроку ітерації.

Таблиця 1. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	-1	0,5	-0,8	-0,6	0,5	0,8
K2	0,7	0,3	-0,3	0,4	0,9	0,1
K3	1	-0,8	-0,5	-1	0,7	-0,3
K4	-0,8	0,4	0,6	0,9	-0,6	-0,9
K5	0,5	-0,6	1	0,4	-0,9	0,5

Таблиця 2. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
-0,6	0,6	0,8	1	-0,3	-0,7

Таблиця 3. Ітерації для даних №1:

1	S	-0,525	-0,21	-1,24	1,445	0,23

	K*S	-0,58125	-0,1875	-1,475	1,88125	0,3625
2	S	0	0	0	1,88125	0,3625
	K*S	-0,56094	-0,5609375	-0,560938	1,790625	-0,10781

Таблиці табл. 1 – табл. 3 показують, що модель коефіцієнтів впевненості дозволила вірно та ефективно визначити шаблон, результат ітерацій: шаблон №4, очікуваний результат: шаблон №4.

Наступний приклад, табл. 4, табл. 5, демонструють процедуру для даних №2.

Таблиця 4. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
0,7	-0,5	-0,8	-0,6	1	-0,1

Таблиця 5. Ітерації для даних №2:

1	S	0,235	0,615	1,415	-1,145	-0,67
	K*S	0,18125	0,65625	1,65625	-1,54375	-0,95
2	S	0,18125	0,65625	1,65625	0	0
	K*S	-0,39688	0,196875	1,446875	-0,623438	-0,62344
3	S	0	0,196875	1,446875	0	0
	K*S	-0,41094	-0,1648438	1,3976563	-0,410938	-0,41094

Таблиці табл. 4 – табл. 5 також показують, що модель коефіцієнтів впевненості вірно та ефективно визначила шаблон №3, який був очікуваний.

Наступний варіант демонстує роботу процедури при вхідних векторах з багатьма невизначеностями, коли входи мають «0»

Таблиця 6. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
1	0	1	0	1	0

Таблиця 7. Ітерації для даних №3:

1	S	-0,65	0,65	0,6	-0,4	0,3
	K*S	-0,9375	0,6875	0,625	-0,625	0,25
2	S	0	0,6875	0,625	0	0,25
	K*S	-0,39063	0,46875	0,390625	-0,390625	-0,07813
3	S	0	0,46875	0,390625	0	0
	K*S	-0,21484	0,3710938	0,2734375	-0,214844	-0,21484
4	S	0	0,3710938	0,2734375	0	0

	K*S	-0,16113	0,3027344	0,1806641	-0,161133	-0,16113
5	S	0	0,3027344	0,1806641	0	0
	K*S	-0,12085	0,2575684	0,1049805	-0,12085	-0,12085
6	S	0	0,2575684	0,1049805	0	0
	K*S	-0,09064	0,2313233	0,0405884	-0,090637	-0,09064
7	S	0	0,2313233	0,0405884	0	0
	K*S	-0,06798	0,2211762	-0,017242	-0,067978	-0,06798

Таблиці табл. 6 – табл. 7 демонструють значні можливості моделі коефіцієнтів впевненості і при таких нетипових вхідних векторах, коли вірно був визначений шаблон №2, що був очікуваний. Разом з тим процес затребував більше ітерацій, а занчення показника виявилося суттєво меньшим, ніж, наприклад в табл. 5.

Призначення прикладу №2 – теж аналіз поведінки та результативності класичної моделі Хеммінга. В цьому прикладі, при виборі найбільш підходящих шаблонів використовуються такі дані: -1 (позначає відсутність параметра); 1 (параметр присутній). Шаблони класифікації знаходяться в табл. 8, вхідний вектор - в табл. 9. В якості вхідних даних використовуються ті ж числа, що і у шаблонах - -1 та 1. Ітерації пошуку відповідного шаблону видно у табл. 10. Шаблони з від'ємний результат припиняють враховуватися при виборі результату після кожної ітерації. Найпридатнішим шаблоном є той, в якого K*S > 0 після всіх ітерацій.

Таблиця 8. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	-1	1	-1	1	-1	1
K2	1	-1	1	-1	1	-1
K3	1	1	1	-1	-1	-1
K4	1	-1	-1	1	1	-1
K5	-1	-1	1	1	1	-1

Таблиця 9. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
-1	1	-1	1	1	1

Таблиця 10. Ітерації для даних №1:

1	S	2	-2	-2	0	0
	K*S	3	-2	-2	0,5	0,5

З таблиць табл. 8 – табл. 10 видно, що модель правильно визначила шаблон, результат ітерацій: шаблон №1, очікуваний результат: шаблон №1.

Наступний приклад, табл. 11, табл. 12, демонструють процедуру для даних №2.

Таблиця 11. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
1	-1	-1	-1	1	-1

Таблиця 12. Ітерації для даних №2:

1	S	-2	2	0	2	0
	K*S	-3	2	-0,5	2	-0,5

Таблиці табл. 11 – табл. 12 показують, що через лімітовану можливість описати унікальні вхідні дані, модель може визначити декілька шаблонів як правильні, результат ітерацій: шаблон №2 або №4, очікуваний результат: шаблон №2 або №4.

Ще один варіант, табл. 13, табл. 14, показує процедуру для даних №3.

Таблиця 13. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
-1	1	1	1	1	-1

Таблиця 14. Ітерації для даних №3:

1	S	0	0	0	0	2
	K*S	-0,5	-0,5	-0,5	-0,5	2

За даними з таблиць табл. 13 – табл. 14 бачимо, що модель придатна для знаходження правильних шаблонів, результат ітерацій: шаблон №5, очікуваний результат: шаблон №5.

В прикладі №3 показане визначення ефективності нечіткої моделі Хеммінга при знаходженні шаблонів. В якості вхідних даних використовуються числа в діапазоні [0; 1]. Задані шаблони для класифікації приведені в табл. 15, вхідний вектор в табл. 16. Вхідні дані теж знаходяться в діапазоні [0; 1]. Процес пошуку шаблону показано в табл. 17. Шаблони, які мають від'ємний результат відсіюються після кожної ітерації. Шаблоном, найбільш придатним для заданих вхідних даних є той, який має позитивну оцінку K*S > 0 після ітерацій.

Таблиця 15. Задані шаблони:

	X1	X2	X3	X4	X5	X6
K1	0,8	0,7	0,9	0,9	1	0,7
K2	0,1	1	0	0,5	0,6	1
K3	1	0,1	0,3	0,7	0,4	0,2
K4	0	0,5	0,1	1	0,3	0
K5	0,6	0,3	0,5	0,2	0,5	0,1

Таблиця 16. Вхідні дані №1:

X1	X2	X3	X4	X5	X6
0,6	0,5	0,7	0,7	0,8	0,6

Таблиця 17. Ітерації для даних №1:

1	S	0,4	0,3	0,35	0,35	0,3
	K*S	0,3125	0,2	0,25	0,25	0,2
2	S	0,3125	0,2	0,25	0,25	0,2
	K*S	0,25	0,121875	0,171875	0,171875	0,121875
3	S	0,25	0,121875	0,171875	0,171875	0,121875
	K*S	0,20703125	0,059375	0,109375	0,109375	0,059375
4	S	0,20703125	0,059375	0,109375	0,109375	0,059375
	K*S	0,1796875	0,007617187	0,057617188	0,057617188	0,007617187
5	S	0,1796875	0,007617187	0,057617188	0,057617188	0,007617187
	K*S	0,165283203	-0,037304688	0,012695313	0,012695313	-0,037304688
6	S	0,165283203	0	0,012695313	0,012695313	0
	K*S	0,162109375	-0,041320801	-0,028625488	-0,028625488	-0,041320801

Таблиці табл. 15 – табл. 17 демонструють, що нечітка модель Хеммінга дала можливість вірно визначити шаблон, що найкраще співвідноситься до

вхідних даних, результат ітерацій: шаблон №1, очікуваний результат: шаблон №1.

Табл. 18, табл. 19 ілюструють процедуру для даних №2.

Таблиця 18. Вхідні дані №2:

X1	X2	X3	X4	X5	X6
0,3	0,8	0,3	0,7	0,8	0,8

Таблиця 19. Ітерації для даних №2:

1	S	0,4	0,4	0,35	0,35	0,25
	K*S	0,3	0,3	0,25	0,25	0,15
2	S	0,3	0,3	0,25	0,25	0,15
	K*S	0,225	0,225	0,175	0,175	0,075
3	S	0,225	0,225	0,175	0,175	0,075
	K*S	0,16875	0,16875	0,11875	0,11875	0,01875
4	S	0,16875	0,16875	0,11875	0,11875	0,01875
	K*S	0,1265625	0,1265625	0,0765625	0,0765625	-0,0234375
5	S	0,1265625	0,1265625	0,0765625	0,0765625	0
	K*S	0,094921875	0,094921875	0,044921875	0,044921875	-0,031640625
6	S	0,094921875	0,094921875	0,044921875	0,044921875	0
	K*S	0,071191406	0,071191406	0,021191406	0,021191406	-0,023730469
7	S	0,071191406	0,071191406	0,021191406	0,021191406	0
	K*S	0,053393555	0,053393555	0,003393555	0,003393555	-0,017797852
8	S	0,053393555	0,053393555	0,003393555	0,003393555	0
	K*S	0,040045166	0,040045166	-0,009954834	-0,009954834	-0,013348389

З таблиць табл. 18 – табл. 19 бачимо, що модель може позначити кілька шаблонів як правильні, результат ітерацій: шаблон №1 або №2, очікуваний результат: шаблон №2.

Зразок з даними №3 показаний в таблицях табл. 20, табл. 21.

Таблиця 20. Вхідні дані №3:

X1	X2	X3	X4	X5	X6
0,2	0,3	0,3	0,9	0,6	0,1

Таблиця 21. Ітерації для даних №3:

1	S	0,45	0,3	0,35	0,45	0,25
	K*S	0,3375	0,1875	0,2375	0,3375	0,1375
2	S	0,3375	0,1875	0,2375	0,3375	0,1375
	K*S	0,253125	0,103125	0,153125	0,253125	0,053125
3	S	0,253125	0,103125	0,153125	0,253125	0,053125
	K*S	0,18984375	0,03984375	0,08984375	0,18984375	-0,01015625
4	S	0,18984375	0,03984375	0,08984375	0,18984375	0
	K*S	0,142382813	-0,007617188	0,042382813	0,142382813	-0,047460938
5	S	0,142382813	0	0,042382813	0,142382813	0
	K*S	0,106787109	-0,035595703	0,006787109	0,106787109	-0,035595703
6	S	0,106787109	0	0,006787109	0,106787109	0
	K*S	0,080090332	-0,026696777	-0,019909668	0,080090332	-0,026696777

З таблиць табл. 20 – табл. 21 видно, що модель цілком придатна для знаходження потрібних шаблонів, результат ітерацій: шаблон №1 або №4, очікуваний результат: шаблон №4.

4.1.3. Дослідження модулю процедур класифікації на основі показників достовірності CF(X).

При дослідженні варіантів реалізації завдань при класифікації на основі показників достовірності CF(X) за класичною моделлю мережі Хеммінга було встановлено, що така запропонована модель забезпечувала достовірні результати при будь-яких первинних даних, які відповідали коефіцієнтам CF(X).

4.2. Процедури оптимізації потоків замовлень в сервісних системах на основі класифікації.

В експериментальних розрахунках значень показників класифікації, які проводились за допомогою застосування розробленого програмного комплексу, була встановлена перевага моделей на основі коефіцієнтів CF(X)

перед моделями, які використовували моделі нечітких величин при будь яких способах перекодування від нечткого представлення до стандарту нейронних мереж Хеммінга.

Висновки до розділу 4

У розділі досліджено метод моделювання та прогнозування достовірності та числової ефективності програмного забезпечення оптимізації послідовностей замовлень на основі процедур класифікації за неточно визначеними даними.

Отримані результати свідчать про коректність запропонованих розробок математичних моделей, алгоритмів і програмних засобів, що дозволило оптимізувати процеси виконання замовлень сервісних систем на основі застосування інтелектуальних процедур і розроблених програмних засобів.

ВИСНОВКИ

В дипломній роботі виконано розробки та дослідження щодо розвитку інтелектуальних процедур оптимізації потоків замовлень у СеС при неточно визначених характеристиках даних на основі моделі асоціативної пам'яті мережі Хеммінга (МХ). Такі класичні МХ дозволяють виконувати класифікацію об'єктів (замовлень) при збурених даних, якщо властивості елементів оцінюються значеннями з множини {– 1; + 1}. Мета – розвиток і розширення кола постановок завдань, а також удосконалення математичних моделей та процедур оптимізації потоків замовлень СеС на основі інтелектуальних процедур мережі МХ при неточно визначених характеристиках даних, далі МХН. В роботі досліджувалися можливості використання в якості моделей нечітких множин, а також показників достовірності експертних систем, коефіцієнтів впевненості CF(A) з множини [– 1; + 1].

Результати розробок і досліджень полягали в наступному: - виконано аналіз моделей та інтелектуальних процедур завдань оптимізації потоків замовлень в сервісних системах (ОПЗСеС); - запропоновано нові постановки завдань ОПЗСеС (система паркінгу авто; призначення фахівців на посаду в ІТ проєктах тощо), для реалізації яких застосовуються процедури класифікації МХН; - удосконалені математичні моделі ОПЗСеС на основі інтелектуальних процедур МХН, які враховують різні моделі вихідних даних; - розроблено програмні засоби із класифікації компонентів послідовностей замовлень на основі модифікованих процедур МХН; - проведено широкий і всебічний числовий експеримент, який підтвердити достовірність та ефективність запропонованих моделей і методів МНХ; - отримані рекомендації стосовно застосування МНХ при неточно визначених характеристиках даних, які враховують запропоновані форми моделей вихідних даних (нечіткі множини (НМ), коефіцієнти впевненості CF(A)). При використанні НМ були запропоновані моделі кодування, за допомогою яких нечіткі величини подавалися як в МХ {– 1; + 1}, а для моделей CF(A) використовувалися

процедура, що безпосередньо застосовувала схеми МХ. Числові дослідження показали певні переваги моделі CF(A), яка завжди забезпечувала потрібний результат класифікації, при тому що в моделі з НВ результати класифікації не завжди були однозначними.

В роботі представлені розроблені моделі перекодування НМ, а також їх порівняльна ефективність, програмна реалізація розроблених моделей класифікації об’єктів з неточно визначеними параметрами МХН, результати проведених числових експериментів, а також процедура ОПЗСеС на основі модифікованих моделей асоціативної пам’яті МХН.

Реалізовано програмний комплекс, призначений для моделювання та дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем. Програмний комплекс було протестовано та відлагоджено. Результати застосування цієї програмної системи дозволили дослідити та підтвердити достовірність та високу точність запропонованих модифікацій нейронної мережі Хеммінга.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1) Gresko, Aleksandr Aleksandrovich, and Konstantin Sergeevich Soloduhin. "Nechetko-mnozhestvennaya mnogoperiodnaya model vybora strategij vzaimodejstviya organizacii s gruppami stejkholderov na osnove determinirovannogo ekvivalenta." Azimut nauchnyh issledovanij: ekonomika i upravlenie 6.3 (20) (2017): 115-118. https://cyberleninka.ru/article/n/nechetko-mnozhestvennaya-mnogoperiodnaya-model-vybora-strategiy-vzaimodeystviya-organizatsii-s-gruppami-steykholderov-na-osnove
2) Gorbunova, Mariya Vladimirovna, Aleksandr Aleksandrovich Gresko, and Konstantin Sergeevich Soloduhin. "Nechetko-mnozhestvennaya model vybora strategij vzaimodejstviya vuza so stejkholderami." Azimut nauchnyh issledovanij: ekonomika i upravlenie 5.3 (16) (2016): 95-98. https://cyberleninka.ru/article/n/nechetko-mnozhestvennaya-model-vybora-strategiy-vzaimodeystviya-vuza-so-steykholderami
3) Gorbunova, Mariya Vladimirovna, Aleksandr Aleksandrovich Gresko, and Konstantin Sergeevich Soloduhin. "Nechetko-mnozhestvennaya mnogoperiodnaya model vybora strategij vzaimodejstviya organizacii s gruppami zainteresovannyh storon na osnove obobshennogo kriteriya." Vestnik Astrahanskogo gosudarstvennogo tehnicheskogo universiteta. Seriya: Ekonomika 4 (2016): 46-55. https://cyberleninka.ru/article/n/nechetko-mnozhestvennaya-mnogoperiodnaya-model-vybora-strategiy-vzaimodeystviya-organizatsii-s-gruppami-zainteresovannyh-storon-na
4) Romanov, Vadim Nikolaevich. "Klasternyj analiz na osnove nechetkih modelej." Almanah sovremennoj nauki i obrazovaniya 10 (2013): 147-151. http://www.gramota.net/materials/1/2013/10/46.html
5) Romanov, Vadim Nikolaevich. "Nechetkie modeli prinyatiya reshenij." Almanah sovremennoj nauki i obrazovaniya 5 (2013): 144-147. http://www.gramota.net/materials/1/2013/5/46.html
6) Kononenko, Igor Vladimirovich, and Svetlana Yurevna Lucenko. "Razrabotka veb-prilozheniya dlya resheniya zadachi vybora metodologii upravleniya proektom pri nechetkih ishodnyh dannyh." Bulletin of the National Technical University" KhPI". Series: Strategic management, portfolio, program and project management 1 (1326) (2019): 11-17. http://pm.khpi.edu.ua/article/view/160065
7) Gresko, Aleksandr Aleksandrovich, and Konstantin Sergeevich Soloduhin. "Vybor smeshannyh tipov strategij vzaimodejstviya universiteta s inostrannymi studentami na osnove nechetkoj mnogoperiodnoj modeli." Territoriya novyh vozmozhnostej. Vestnik Vladivostokskogo gosudarstvennogo universiteta ekonomiki i servisa 3 (38) (2017): 164-178. https://cyberleninka.ru/article/n/vybor-smeshannyh-tipov-strategiy-vzaimodeystviya-universiteta-s-inostrannymi-studentami-na-osnove-nechetkoy-mnogoperiodnoy-modeli

8) Romanov, Vadim Nikolaevich. "Primenenie nechetkih modelej v zadachah klassifikacii." Almanah sovremennoj nauki i obrazovaniya 5-6 (2014): 108-112. http://www.gramota.net/materials/1/2014/5-6/32.html
9) Chistyakov, A. D., and V. A. Bulanov. "Model ocenki effektivnosti biznes-processov organizacii na osnove polozhenij teorii nechetkih mnozhestv." Vestnik Rostovskogo gosudarstvennogo ekonomicheskogo universiteta (RINH) 4 (48) (2014): 143-149. https://cyberleninka.ru/article/n/model-otsenki-effektivnosti-biznes-protsessov-organizatsii-na-osnove-polozheniy-teorii-nechetkih-mnozhestv
10) Gresko, A. A., and A. V. Petrova. "Nechetko-mnozhestvennyj scenarnyj analiz strategij vzaimodejstviya organizacii so stejkholderami." Problemy sovremennoj ekonomiki 3 (63) (2017): 66-69. https://cyberleninka.ru/article/n/nechetko-mnozhestvennyy-stsenarnyy-analiz-strategiy-vzaimodeystviya-organizatsii-so-steykholderami
11) Glushan, Valentin Mihajlovich, Vladimir Petrovich Karelin, and Olga Leonidovna Kuzmenko. "Nechetkie modeli i metody mnogokriterialnogo vybora v intellektualnyh sistemah podderzhki prinyatiya reshenij." Izvestiya Yuzhnogo federalnogo universiteta. Tehnicheskie nauki 93.4 (2009): 106-113. https://cyberleninka.ru/article/n/nechetkie-modeli-i-metody-mnogokriterialnogo-vybora-v-intellektualnyh-sistemah-podderzhki-prinyatiya-resheniy
12) Anesyanc, Sarkis Artavazdovich, Andrej Dmitrievich Chistyakov, and Safura Shihovna Muradova. "Nechyotkaya model ranzhirovaniya i attestacii personala kafedry." Aspirant 4-1 (2016): 16-18. https://elibrary.ru/item.asp?id=28083587
13) Gavrilyuk, Evgenij Alekseevich, and Sergej Aleksandrovich Mancerov. "Upravlenie tehnicheskim sostoyaniem slozhnyh sistem na osnove nechetkoj modeli." Avtomatizaciya processov upravleniya 1 (2018): 91-98. https://elibrary.ru/item.asp?id=32737057
14) Lihosherst, Elena Nikolaevna, Konstantin Sergeevich Soloduhin, and Andrej Yakovlevich Chen. "Mnogoperiodnaya model vybora strategij vzaimodejstviya organizacii s gruppami zainteresovannyh storon v stejkholderskoj seti s mnozhestvennymi" centrami vlasti"." Azimut nauchnyh issledovanij: ekonomika i upravlenie 7.4 (25) (2018): 287-290. https://cyberleninka.ru/article/n/mnogoperiodnaya-model-vybora-strategiy-vzaimodeystviya-organizatsii-s-gruppami-zainteresovannyh-storon-v-steykholderskoy-seti-s
15) Pesochenko, Svetlana Valerevna, Oleg Nikolaevich Pyavchenko, and Olga Aleksandrovna Usenko. "Modelirovanie metrik Evklida i Hemminga v realnom rezhime vremeni." Izvestiya Yuzhnogo federalnogo universiteta. Tehnicheskie nauki 4 (153) (2014): 49-55. https://cyberleninka.ru/article/n/modelirovanie-metrik-evklida-i-hemminga-v-realnom-rezhime-vremeni

16) Ogneva, Oksana Evgenevna. "Model opredeleniya strategii cenoobrazovaniya promyshlennogo predpriyatiya." Problemi informacijnih tehnologij 2 (2012): 67-72. http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Pit_2012_2_15.pdf
17) Sherbich, Aleksej Yurevich, and Vladimir Nikolaevich Kutrunov. "Primenenie matematicheskoj modeli nejronnoj seti Hemminga dlya kontrolya kachestva i vosstanovleniya nekorrektnyh atributov metadannyh iz zagolovkov sejsmicheskih fajlov." Vestnik Irkutskogo gosudarstvennogo tehnicheskogo universiteta 2 (109) (2016): 50-60. https://cyberleninka.ru/article/n/primenenie-matematicheskoy-modeli-neyronnoy-seti-hemminga-dlya-kontrolya-kachestva-i-vosstanovleniya-nekorrektnyh-atributov
18) Zavgorodnij, V. I., and A. V. Zolotaryuk. "Ocenka effektivnosti informacionnyh bankovskih sistem." Upravlenie razvitiem krupnomasshtabnyh sistem (MLSD'2019). 2019. https://elibrary.ru/item.asp?id=41727514
19) Ratinskaya, E. V. "Primenenie apparata nechetkih mnozhestv k ocenke urovnya sformirovannosti kompetencij obuchayushihsya." Trudy Bratskogo gosudarstvennogo universiteta. Seriya: Estestvennye i inzhenernye nauki 2 (2017): 175-179. https://elibrary.ru/item.asp?id=34957525
20) Shpolyanskaya, I., A. Dolzhenko, and A. Prohorova. Nechetkaya model ocenki kachestva portala vuza dlya effektivnogo prodvizheniya obrazovatelnyh uslug. Litres, 2022. https://books.google.com/books?hl=en&lr=&id=f4-FDwAAQBAJ&oi=fnd&pg=PA47&dq=%D0%BD%D0%B5%D1%87%D0%B5%D1%82%D0%BA%D0%B0%D1%8F+%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C+%D1%85%D0%B5%D0%BC%D0%BC%D0%B8%D0%BD%D0%B3%D0%B0&ots=sRseTZ1rDh&sig=Iknu5FTU5-iTqs4lodXt9ZoR8uo
21) Bukin, Artem Gennadevich, and Gennadij Alekseevich Gordeev. "Metod nechetkoj identifikacii kriticheskih elementov avtomatizirovannyh sistem na osnove fizicheski nekloniruemyh funkcij." Izvestiya Instituta inzhenernoj fiziki 3 (2016): 63-69. https://elibrary.ru/item.asp?id=26586167
22) Melnikov, A. Yu., and K. M. Komissarov. "Issledovanie metodov intellektualnogo analiza bibliograficheskih opisanij i razrabotka programmnoj sistemy dlya analiza spiska literatury." Radioelektronika, informatika, upravlinnya 4 (47) (2018): 121-134. https://cyberleninka.ru/article/n/issledovanie-metodov-intellektualnogo-analiza-bibliograficheskih-opisaniy-i-razrabotka-programmnoy-sistemy-dlya-analiza-spiska

23) Karelin, Vladimir Petrovich, and Oksana Leonidovna Kuzmenko. "Nahozhdenie predstavitelya klassa nechetkih situacij pri postroenii modeli prinyatiya reshenij." Izvestiya vysshih uchebnyh zavedenij. Severo-Kavkazskij region. Tehnicheskie nauki 4 (2008): 50-54. https://cyberleninka.ru/article/n/nahozhdenie-predstavitelya-klassa-nechetkih-situatsiy-pri-postroenii-modeli-prinyatiya-resheniy
24) Куземко, С. М., and С. В. Лужецький. "ПОШУКОВА СИСТЕМА НА ОСНОВІ НЕЧІТКОЇ ЛОГІКИ." *Рецензент: ІВ Кузьмін, д. т. н. ВС Осадчук, д. т. н.* (2006): 389. https://www.researchgate.net/profile/Dubovoy-Vm/publication/245407205_Thirteen_International_Conference_on_Automatic_Control_Avtomatika-2006/links/58502f5e08ae4bc8993b68cb/Thirteen-International-Conference-on-Automatic-Control-Avtomatika-2006.pdf#page=390
25) Горохов, Г. Т. "Біонічна модель нечіткого логічного аналізу параметрів контрольно-технічних оглядів авіаційних систем." *Збірник наукових праць Державного науково-дослідного інституту авіації* 7 (14) (2011): 193-198. http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/Znpdndia_2011_7(14)__33.pdf
26) Великоіваненко, Галина Іванівна, et al. "Оцінювання рівня економічної безпеки на підґрунті відстані Хеммінга." (2018). https://core.ac.uk/download/pdf/197269753.pdf
27) Лебідь, В. В. "UDC 519.86: 658.562 НЕЧІТКО-МНОЖИННА МОДЕЛЬ ОЦІНКИ ЕФЕКТИВНОСТІ ВИКОНАННЯ МІЖНАРОДНИХ ПЕРЕВЕЗЕНЬ ВАНТАЖІВ У ПРОЕКТАХ РОЗВИТКУ МІЖНАРОДНИХ ТРАНСПОРТНИХ КОРИДОРІВ." http://publications.ntu.edu.ua/upravl_progect/2013_12_tech/078.pdf
28) Borisova, Lyudmila, Valery Dimitrov, and Inna Nurutdinova. "Algorithm for assessing quality of fuzzy expert information." 2017 IEEE East-West Design & Test Symposium (EWDTS). IEEE, 2017. https://ieeexplore.ieee.org/abstract/document/8110107/
29) Прогнозування і моделювання (Oracle Crystal Ball) [Електронний ресурс] – Режим доступу: http://ubc-corp.ru/ru/oracle_crystall_ball
30) Прогнозування в Predictive Planning (Oracle Crystal Ball) [Електронний ресурс] – Режим доступу: https://ivan-shamaev.ru/predictive-planning-oracle-crystal-ball/
31) Tahseen A., Aqil S., Burney Cemal A. A New Quantile Based Fuzzy Time Series Forecasting Model [Електронний ресурс] – Режим доступу: https://publications.waset.org/14214/pdf
32) Qiang S., Brad S., Forecasting enrollments with fuzzy time series - Part I [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/016501149390355L

33) Sheng T., Yi-Chung C. Deterministic fuzzy time series model for forecasting enrollments [Електронний ресурс] – Режим доступу: https://www.sciencedirect.com/science/article/pii/S0898122107001630
34) Tahseen A., Jilani S., Aqil B., Ardil C. Multivariate High Order Fuzzy Time Series Forecasting for Car Road Accidents [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/285870449_Multivariate_High_Order_Fuzzy_Time_Series_Forecasting_for_Car_Road_Accidents
35) Q. Song, and B. S. Chissom, "Forecasting enrollments with fuzzy time series — Part I," Fuzzy Sets and Systems, vol. 54, issue 1, 1993a, pp. 1¬9.
36) B. S. Chissom, "Fuzzy time series and its models," Fuzzy Sets and Systems, vol. 54, issue 3, 1993b, pp. 269-277.
37) Q. Song, and B. S. Chissom, "Forecasting enrollments with fuzzy time series — Part II," Fuzzy Sets and Systems, vol. 62, 1994, pp. 1-8.
38) S. M. Chen, "Forecasting enrollments based on fuzzy time series," Fuzzy Sets and Systems, vol. 81, 1996, pp. 311-319.
39) Q. Song, "A note on fuzzy time series model selection with sample autocorrelation functions," Cybernetics and Systems: An International Journal, vol. 34, 2003, pp. 93-107.
40) Q. Song, and R.P. Leland, "Adaptive learning defuzzification techniques and applications," Fuzzy Sets and Systems, vol. 81, 1996, pp. 321-329.
41) J. R. Hwang, S. M. Chen, and C. H. Lee, "Handling forecasting problems using fuzzy time series", Fuzzy Sets and Systems, vol. 100, 1998, pp.217-228.
42) K. Huarng, "Heuristic models of fuzzy time series for forecasting," Fuzzy Sets and Systems, vol. 123, issue 3, 2001a, pp. 369-386
43) K. Huarng, "Effective lengths of intervals to improve forecasting in fuzzy time series," Fuzzy Sets and Systems, vol. 123, issue 3, 2001b, pp. 387-394.
44) S.-M. Chen, "Forecasting enrollments based on high-order fuzzy time series," Cybernetics and Systems: An International Journal, vol. 33, pp. 1-16.
45) О. Мулеса, В. Сниюк та С. Герзанич, «Метод нечіткої класифікації на основі послідовного аналізу вальда,» Automation of technological and business processes, т. №11, pp. 35-42, 2020.
46) ScientificWorldJournal. 2014; 2014: 129483.Published online 2014 Apr 3. doi: 10.1155/2014/129483 PMCID: PMC3997130 PMID: 24982924 A Multistrategy Optimization Improved Artificial Bee Colony Algorithm
47) Шинкаренко В.І., Куроп'ятник О.С, Забула Г.В, Пєтін Д.О., Лукін Є.В. Якість програмного забезпечення та тестування [Текст]: методичні вказівки до лабораторних / уклад.: В.І. Шинкаренко, О.С. Куроп'ятник, Г.В. Забула, Д.О. Пєтін, Є.В. Лукін, Дніпропетр, нац. ун-т. залізн. трасоп. ім. акад. В. Лазаряна. – Д.: Вид-во ПФ «Стандарт-Сервіс», 2018. – 50 с. – 43с.

48) K. Dzh. Kejt Vvedennya v sistemi baz danih Per. s angl. 8-e izd. M.: Izdatelskij dom «Vilyams», 2006 – 1328S.
49) Tomas Konnolli, Karolin Begg. Bazy dannyh. Proektirovanie, realizaciya i soprovozhdenie. Teoriya i praktika – 3-e izd. M.: Izdatelskij dom «Vilyams», 2003, 1436 s.
50) ДСанПІН 3.3.2.007-98 «Державні санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Прийняття від 10.12.1998.
51) ДСН 3.3.6.042-99 «Санітарні норм мікроклімату виробничих приміщень». Прийняття від 01.12.1999.
52) ДБН В.2.5.-28:2021 «Природне і штучне освітлення». Чинні з 28.02.2022.
53) НАПАБ А.01.001-2014 «Правила пожежної безпеки в Україні». Редакція від 03.10.2017.
54) Кодекс цивільного захисту України. Редакція від 28.11.2022.
55) Інструкція для навчальних закладів України «Інструкція з охорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою». Сторінка веб-ресурсу https://osvita-docs.com/node/41. Станом на 02.12.2022.
56) Rick F. van der Lans. SQL for MySQL developers: a comprehensive tutorial and reference. Addison-Wesley, Reading, MA, USA, 2007. ISBN 0-13-149735-9
57) Scott Urman. Oracle8i advanced PL/SQL programming. Osborne/ McGraw-Hill, Berkeley, CA, USA, 2000.
58) George K. (George Kuriakose) Thiruvathukal, John P. Shafaee, and Thomas W. Christopher. Web programming: techniques for integrating Python, Linux, Apache, and MySQL. Prentice-Hall PTR, Upper Saddle River, NJ 07458, USA, 2002
59) Mitchell, Melanie (1996). An Introduction to Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944. С. – 8-9.
60) Holland, J.H. (1975) Adaptation in Natural and Artificial Systems.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор УДУНТ

А.В. Радкевич

«Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем»

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01165-01-ЛЗ

Завідувач кафедри КІТ

_____

Керівник розробки

_____

Виконавець

_____

Нормоконтролер

_____

2022

ЗАТВЕРДЖЕНО

1116130.01165-01-ЛЗ

«Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем»

Технічне завдання

1116130.01165-01

Аркушів 22

2022

АНОТАЦІЯ

Документ 1116130.01165-01 «Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем». Технічне завдання, що входить до складу програмної документації до дипломного проекту.

У даному документі представлене призначення та область застосування програми, основні вимоги, стадії та строки виконання проекту, технічні та техніко-економічні показники, що пред’являються до програми.

ЗМІСТ

ВСТУП

«Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем» – це програмний комплекс для аналізу послідовностей елементів, які надходять до сервісних систем, за рахунок нечіткої класифікації визначається оптимальний пристрій-виконавець, що відповідає властивостям поточного елементу вхідної послідовності.

Різноманітні технологічні, виробничі, інформаційні, лікувальні та інші процеси у складних системах можуть мати значний ступень невизначеності, які можливо представити різними моделями. В цій роботі – моделі нечітких величин та коефіцієнтів достовірності. У якості прикладів такого типу об'єктів розглянуто нові сфери, запропоновані в дипломні роботі. А саме – призначення на посаду працівників при реалізації проєктів програмного забезпечення, а також при паркуванні автотранспорту.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ ректора Українського університету науки і технологій, УДУНТ, проф. Пшінька О. М. № _____ ст від _____. «Про призначення керівників та затвердження тем магістерських робіт» факультету «Комп'ютерні технології і системи» за спеціальністю 121 «Інженерія програмного забезпечення» по кафедрі «Комп'ютерні інформаційні технології».

У відповідності з наказом, тема дипломного проекту – «Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем» Керівник проекту проф. Скалозуб В. В.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Програмний комплекс призначений для аналізування та дослідження математичних властивостей потоків замовлень з неточно визначеними ознаками.

Функціональним призначенням програми є наступне:

- побудова математичної моделі процесів нечіткої класифікації;
- побудова шаблонів баз знань для реалізаціх класифікації;
- розрахунки процесів класифікації на основі мережі Хеммінга.

2.2 Експлуатаційне призначення

Експлуатаційне призначення програмного комплексу полягає в автоматизуванні обробки завдань нечіткої класифікації при дослідженні інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем».

Експлуатаційними параметрами, які визначаються за призначенням програмного комплексу, є:

- визначення класів вхідних елементів на основі їх неточно визначених ознак;
- прогнозування категорії «виконавця» для вхідних елементів.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик програмного комплексу наступні:

- завантаження;
- побудова математичної моделі класифікації;
- побудова варіантів шаблонів для завдань обслуговування;
- побудова таблиць ознак вхідних послідовностей замовлень;
- відображення шаблонів та ознак вхідних елементів що досліджуються;
- відображення на зовнішніх пристроях;
- збереження результатів в файл;

3.1.1 Вхідні дані

Вхідні дані для користувача наступні:

- категорії моделей невизначеності вхідних послідовностей;
- завантаження даних про ознаки елементів;
- параметри розмірності векторів ознак;
- параметри бази шаблонів класифікації.

3.1.2 Вихідні дані

Вихідні дані для користувача наступні:

- відображення баз шаблонів;
- відображення побудованих властивостей вхідних векторів послідовності СеС;
- виведення даних про процес класифікації.

3.2 Вимоги до надійності

Програмний комплекс повинен:

- відповідати специфікації;
- відповідати функціональним характеристикам;
- забезпечити контроль вхідних і вихідних даних. У випадку неправильного вводу даних видати повідомлення про помилку;
- забезпечити надійне збереження даних.

3.3 Умови експлуатації

Для забезпечення надійного функціонування програмного комплексу, користувачеві необхідно дотримуватися таких умов:

- програмний комплекс повинен використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ і мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДСанПіН 3.3.2-007-98 [1];
- для роботи з програмним комплексом, користувач повинен мати базові навички роботи з Microsoft Office Excel, та ознайомлений с керівництвом користувача;
- програмний комплекс повинен використовуватись у приміщеннях з наступними кліматичними умовами: температура навколишнього повітря – 21– 25 ℃, відносна вологість повітря – 40-60 %;

Мінімальна кількість персоналу, необхідного для роботи програми 1 штатна одиниці – 1 користувач програми – користувач.

Користувач програмного комплексу повинен мати практичні навички роботи за ПК.

3.4 Вимоги до складу та параметрів технічних засобів

Для коректного функціювання програми апаратна частина повинна задовольняти наступним умовам:

- процесор AMD Ryzen 2200g; оперативна пам'ять 1 Гб або більше; CD або DVD-привод;
- USB-роз’єм; клавіатура; комп’ютерна «миша».

3.5 Вимоги до інформаційної та програмної сумісності

Програмний комплекс розрахований на операційну систему, що має Microsoft Office Excel.

3.6 Вимоги до маркування та пакування

Упаковка програмного комплексу, включаючи документацію, повинна бути захищена від пошкоджень різного роду (механічних, кліматичних).

На упаковці повинно бути вказано назву програмного комплексу, номер версії, якщо вона змінювалась, мінімальні системні вимоги .

На зворотній стороні упаковки вказується розробник та його юридична адреса. На рис. 3.1 приведений приклад маркування.

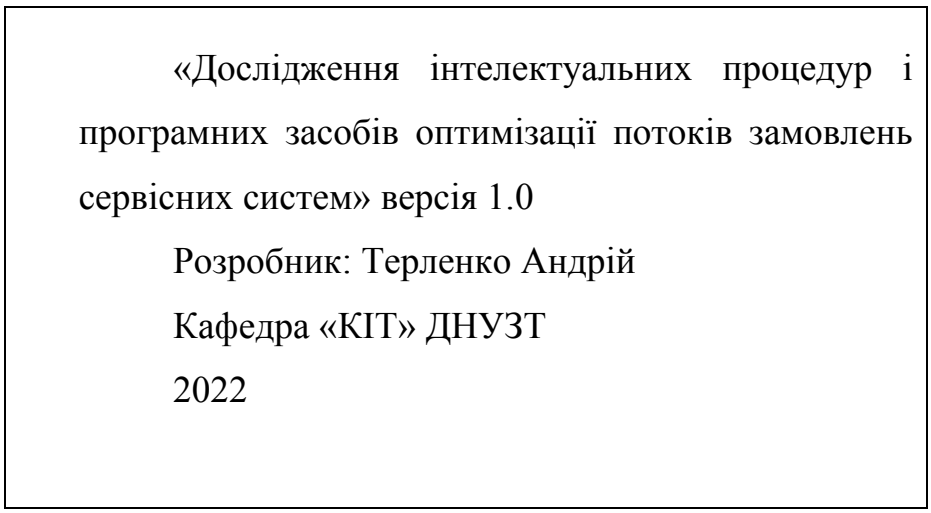


Рисунок 3.1 – Приклад маркування

3.7 Вимоги до транспортування та зберігання

Транспортування повинно проводитися довіреною особою. Воно проводиться в упаковці, яка захищає носії з програмним комплексом від різного роду пошкоджень.

Місце зберігання програмного комплексу повинно бути сухим, з відсутністю пилу та з низьким коефіцієнтом відносної вологості повітря. Строк зберігання програмного комплексу залежить від носія інформації.

Транспортування буде здійснюватися на таких носіях: CD/DVD-RW, флеш, також через глобальну всесвітню систему інформаційного обміну Internet.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації має входити технічне завдання та робочий проект.

До складу робочого проекту мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача. Керівництво з моделювання інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем нерегулярної часової послідовності.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів [3].

5 КОШТОРИС НА РОЗРОБКУ ПЗ

5.1 Загальні положення

Техніко-економічне обґрунтування (ТЕО) — це розрахунок економічної доцільності здійснення проекту, заснований на порівняльній оцінці витрат і результатів ефективності використання, а також строку окупності вкладень. ТЕО — це виваженість кожного вашого кроку в реалізації задуманого.

Техніко-економічне обґрунтування (ТЕО) являє собою комплексний передпроектний документ, базується на аналізах і розрахунках різних показників. На підставі всіх розрахунків і аналітичних даних у ТЕО даються висновки про економічну доцільність реалізації проекту, дається оцінка перспективам впровадження проекту.

На першому етапі для розрахунку величин трудових витрат розробників необхідно оцінити розмір програмного забезпечення. Різниця між методиками для оцінки трудовитрат залежить від типу критерію оцінки якості (кількісний або якісний)[1].

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF,$$

де E– витрати праці на проект (в людино-місяцях);

S^b– розмір коду (в KLOC);

EAF– фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05.

Припустимо, що розмір програмного коду програмного засобу – 907 рядків:

$$E = 2{,}4 \cdot 0{,}907^{1{,}05} \cdot 1 = 2{,}1 \text{ люд./міс.}$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 2,1 людино-місяці.

Нижче наведені розрахунки вартості розробки «Моделювання нечітких часових рядів». Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;
- витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

5.2 Розрахунок основної заробітної плати

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину [2]. Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чоловік	місяців	
	Інженер-програміст	34000			

Описаний в проекті програмний продукт буде розроблений одним програмістом в період з 24.10.22 до 19.12.22, що складає 56 календарних днів або 8 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 196.15 грн/год. Витрати робочого часу за період участі в проекті ($T_{розр}$) визначаються за формулою:

$$T_{розр} = N_{чол} \cdot N_{тиж} \cdot N_{год},$$

де $N_{чол}$ – кількість виконавців, чол.;

$N_{тиж}$ – тривалість розробки в тижнях;

$N_{год}$ – витрати робочого часу за тиждень, год;

$$T_{розр} = 1 \cdot 8 \cdot 40 = 320 \text{ чол./год.}$$

ОЗП визначається за формулою:

$$ОЗП = T_{розробки} \cdot N \cdot K_{кв},$$

де $T_{розробки}$ – витрати праці у чол./год;

N– погодинна ставка;

$K_{кв}$– коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складає:

$$ОЗП = 320 \cdot 196.15 \cdot 0,75 = 47076 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у 22 відсотках від суми мінімальної заробітної плати та становлять 10356,72 грн.

5.3 Розрахунок накладних витрат

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація, обладнання,

зарплату адміністративного персоналу та інше. Кожне підприємство визначає їх індивідуально, та зазвичай вони становлять 30-40% від суми прямих витрат на оплату праці:

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов'язані з її експлуатацією.

Експлуатаційні витрати на персональний комп'ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп'ютеру. В експлуатаційні витрати входять:

- амортизаційні витрати на персональний комп'ютер і програмне забезпечення;
- загальногосподарські витрати (прибирання приміщення, охорона, оренда, комунальні послуги);
- витрати на електроенергію.

5.3.1 Розрахунок витрат на електроенергію

Витрати на електроенергію (Сел) визначаються за формулою:

$$C_{ел} = P \cdot B \cdot T_{розр},$$

де P – потужність комп'ютера та допоміжних споживачів електричної енергії приймається 0,45 кВт/год;

B – вартість 1 кВт/год. Під час розробки диплому у 2022 р. складає 0,93 грн [3];

$T_{розр}$ – вартість роботи з ЕВМ, прийнято рівним загальному робочому часу.

Отже у нашому випадку маємо:

$$C_{ел} = 0,45 \cdot 0,93 \cdot 320 = 133,92 \text{ грн.}$$

Витрати на матеріали протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції (ноутбук Lenovo Legion 5 15ACH6H) приймається 48999 грн. [6], термін експлуатації – 3 роки. Отже, можна визначити ці витрати за період створення програмного засобу:

де $B_{ком}$ – вартість персонального комп'ютеру;

$N_д$– кількість днів розробки програмного продукту;

$N_{експ}$– термін експлуатації персонального комп'ютеру.

Витрати на матеріали визначаються так:

$$C_{вм} = 48999 \cdot \frac{40}{3 \cdot 365} \cdot \frac{10}{100} = 178,99$$

Заробітна плата ремонтника (Срем) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 17500 грн. [4]. Заробітна плата ремонтника визначається так:

За статистикою витрати на комплектуючі вироби для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$СКОМ=СВМ=178,99 \text{ грн.}$$

5.3.2 Розрахунок амортизаційних відрахувань

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АПК}=48999\cdot \frac{40}{3\cdot 365} = 1789{,}91 \text{ грн.}$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння для Windows 5 років, а Microsoft Excel 2010 за 3 роки, то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Windows Home 10, для завантаження та збереження даних використовується Microsoft Excel 2010.

$$\text{АПЗ}_W = 3870 \cdot \frac{2}{5\cdot 12} = 129 \text{ грн.}$$

$$\text{АПЗ}_{Excel} = 2436 \cdot \frac{2}{3\cdot 12} = 135{,}3 \text{ грн.}$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2.

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10 Pro	3870	https://softpro.com.ua/windows-10-pro-fqc-08909-new-oem/	129
Microsoft Excel 2010	2436	https://compsoft.com.ua/ua/p846655257-microsoft-excel-2010.html	135,3
Всього:	6306	-	264,3

Таблиця 5.2 – Використовуване програмне забезпечення

5.3.3 Комунальні послуги

Додаткові витрати : прибирання приміщень, охорона, оренда, комунальні послуги прийняти рівними 8460 гривень на місяць.

Оренду приміщень приймемо рівною 5660 гривень на місяць за 20 м^2 [5].

5.3.4 Зведений розрахунок накладних витрат

Результати розрахунку накладних витрат наведено в табл. 5.3, де враховані основні складові накладних витрат проекту:

$$С_{експ} = С_{ел} + С_{вм} + С_{рем} + АПК + АПЗ + С_{ор} + С_{дод} + С_{Ком};$$

$$С_{експ} = 133,92 + 178,99 + 299 + 1789,91 + 264,3 + 11320 + 5600 + 178,99 =$$

$$19765,11 \text{ грн.}$$

Таблиця 5.3 – Розпис накладних витрат по проекту

№ п/п	Статті видатків	Сума, грн
1	Витрати на електроенергію	133,92
2	Вартість витратних матеріалів	178,99
3	Витрати на ремонт	299
4	Амортизація персонального комп’ютера	1789,91
5	Амортизація програмного забезпечення	264,3
6	Оренда приміщення	11320
7	Додаткові витрати	5600
8	Комплектуючі вироби	178,99
9	Всього накладних витрат	19765,11

5.3.5 Розрахунок витрат на створення програмного продукту

Таким чином, витрати на створення програмного комплексу в рамках виконання проекту складають:

- $С_{розробки} = ОЗП + С_{соц} + С_{накл} + С_{експ}$;
- $С_{розробки} = 47076 + 10356,72 + 19765,11 + 11354,89 = 88552,72$ грн.

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

№ п/п	Найменування витрат	Сума, грн
1	Основна заробітна плата	
2	Відрахування на соціальні потреби	
3	Накладні витрати	,
4	Експлуатаційні витрати	1
5	Всього	

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для моделювання нечітких часових рядів За результатами розрахунків, приблизна вартість розробки складає 88552,72 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Усі стадії та етапи розробки приведені у табл.6.1.

Таблиця 6.1– Етапи розробки та строки

Стаді розробки	Етап розробки	Термін
Технічне завдання (ТЗ)	Постановка задачі, збір початкових матеріалів	15.11.21 – 02.12.21
	Розробка структур вхідних та вихідних даних	02.12.21 – 12.01.22
	Визначення вимог до програми	12.01.21 – 25.01.22
	Узгодження та затвердження технічного завдання	25.01.22 – 19.02.22
Робочий проект	Програмування та відладка програми	19.02.22 – 10.07.22
	Тестування програми	10.07.22 – 01.09.22
Впровадження	Розробка, узгодження та затвердження програмної документації	01.09.22 – 20.12.22

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмі та його специфікації. Контроль виконання роботи забезпечується головним керівником розробки.

Прийом програми здійснюється уповноваженою комісією.

СПИСОК ЛІТЕРАТУРИ

1. Модель оценки стоимости COCOMO [Електронний ресурс] – Режим доступу: https://project.dovidnyk.info/index.php/home/upravlenieproektamiposozdaniyuprogrammnogoobespecheniya/130-modelocenkistoimostisosomo
2. Статистика зарплат програмістів, тестувальників і PM в Україні | DOU [Електронний ресурс] – Режим доступу: https://jobs.dou.ua/salaries/?period=2022-06&position=Junior%20SE&experience=0-2&education=4
3. Тарифи на електроенергію для підприємств в Україні в 2022 році – діючі ціни на світло для бізнесу на сьогодні [Електронний ресурс] – Режим доступу: https://bankchart.com.ua/spravochniki/indikatory_rynka/electric_business_tariff
4. Комп'ютерний майстер: середня зарплата в Україні [Електронний ресурс] – Режим доступу: https://www.work.ua/salary-%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9+%D0%BC%D0%B0%D1%81%D1%82%D0%B5%D1%80/
5. Оренда квартир Дніпро, зняти квартиру на тривалий термін на OLX.ua [Електронний ресурс] – Режим доступу: https://www.olx.ua/uk/nedvizhimost/kvartiry/dolgosrochnaya-arenda-kvartir/dnepr/?search%5Bfilter_float_total_area%3Ato%5D=20&search%5Bfilter_float_total_area%3Afrom%5D=15&search%5Bdistrict_id%5D=119&view=galleryWide
6. Купити Ноутбук ігровий Lenovo Legion5 15ACH6H (82JU00YLRA) Phantom Blue за низькою ціною в Києві, Україні. Найнижча ціна на Legion5 15ACH6H (82JU00YLRA) Phantom Blue в інтернет магазині Comfy (Комфі) ua [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20220316224739/https://comfy.ua/ua/noutbuk-igrovoj-lenovo-legion5-15ach6h-82ju00ylra-phantom-blue.html

ЗАТВЕРДЖЕНО
1116130.01177 13 01-ЛЗ

Система для дослідженя інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем

Опис програми
1116130.01177 13 01

Аркушів 19

2022

АНОТАЦІЯ

Документ 1116130.01177 13-01 «Система для дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем» входить до складу програмної документації до дипломного проекту.

В документі міститься опис програми та її функціональних можливостей. Система реалізована на мовах: JavaScript (React) та Python (Flask) у програмному середовищі Sublime Text 3.

ЗМІСТ

4. Загальні відомості

Розроблена програма для дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем.

Для функціонування програмної системи необхідно мати такі програмні засоби: операційна система Linux 4.54 або вище та середовище.

До системи входять такі складові:

- завантаження;
- побудова математичної моделі;
- розмноження
- побудова варіантів;
- побудова
- відображення;
- відображення на графіку;
- збереження результатів в файл;

Складові реалізації рівнів взаємодії системи у програмному середовищі Sublime Text 3.

5. Функціональне призначення

Програмний комплекс призначений для аналізування та дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем. Процедури інтелектуального формування дозволяють зменшувати кількість варіантів аналізу та підвищують чисельну ефективність методу оптимізації послідовностей замовлень з неточно визначеними ознаками. В роботі приведені інтелектуальні процедури класифікації операцій на основі моделей нейронних мереж Хеммінга. При цьому також розроблено удосконалену структуру інформаційної технології класифікації з використанням інтелектуальних процедур.

При необхідності кожен має змогу написати додаток до даної програми, оскільки продукт позиціонується як Open Source. Користувач даного програмного продукту позиціонується як дослідник, тобто це відносить даний програмний продукт до дослідницького програмного класу.

6. Опис логічної структури

Формалізація задачі на рівні зовнішнього проектування представлена у вигляді діаграми варіантів використання .

Користувач представлені у вигляді актора, що взаємодіє з системою за допомогою варіантів використання. Варіанти використання надають опис можливостей, які система надає акторам.

На діаграмах використані наступні типи відношень між варіантами використання та акторами:

- відношення асоціації – відображає зв’язок між акторами та варіантом використання. Відображається лінією зі стрілкою між акторами і варіантом використання;
- відношення включення – показує, що варіант використання включається в базову послідовність, позначається стрілкою з поміткою «include».

Користувач може виконувати наступні варіанти використання:

- можливість завантаження результатів проведеного експерименту на боці клієнту.
- завантаження;
- побудова математичної моделі;
- побудова варіантів;
- відображення результатів;
- відображення результатів на графіку;
- збереження результатів в файл;

3.1 Алгоритми програми.

Алгоритм програми приведено на рисунку 3.1.

Рисунок 3.1 – Алгоритм програми

3.2 Використані методи

При розробленні системи було використано бібліотеки для виміру часу stopwatch.py, та react.js для створення користувацького інтерфейсу, а також бібліотеку pyplot для роботи з векторами та матрицями за допомогою Python.

3.3 Структура програми

Структура програми зображена на рисунку 3.2.

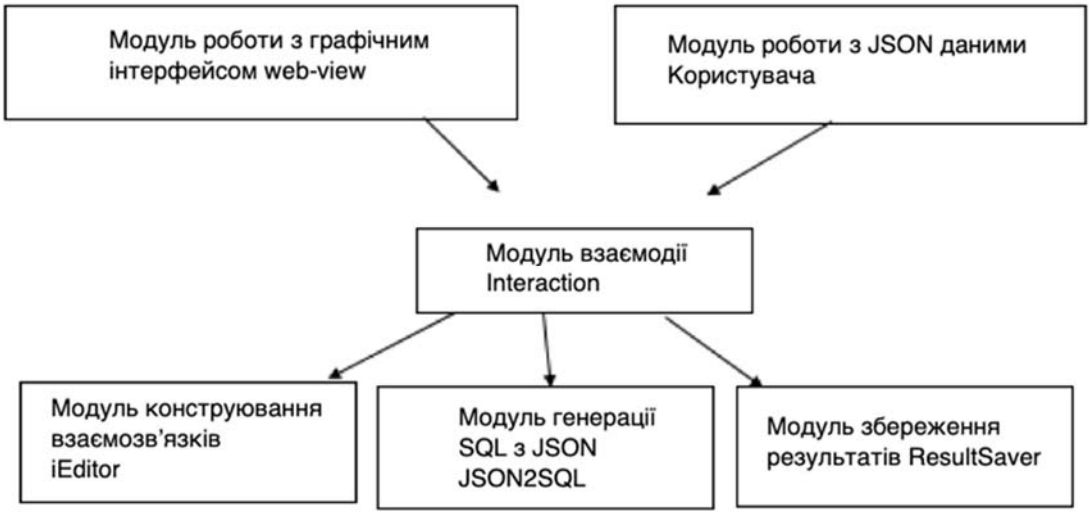


Рисунок 3.2 – Структура програми

У програмному продукті розроблено такі модулі:

- Interaction керує усіма робочими модулями у багатопоточному режимі;
- Web – view отримує дані стосовно користувача-дослідника, відображає користувачу результати у графічному інтерфейсі. Також даний модуль відповідає за інтерфейс сторінки програми;
- Модуль роботи з JSON отримує дані від користувача та генерує відповідну JSON структуру, для подальшої обробки такої структури даною системою. Даний модуль тісно взаємодіє з модулем iEditor;
- JSON2SQL отримує вхідні дані – JSON опис структури

взаємозв’язків БД і виконує генерацію структури у форматі SQL тісно взаємодіє із модулем роботи з JSON;

- ResultSaver відповідає за збереження результатів дослідження на боці клієнту у вигляді .png зображення.

3.4 Зв’язки з іншими програмами

Для використання програми потрібно застосовувати інтернет мережу.

3.5 Структура сервісу та можливість розширення

Сервісне розширення передбачує, що інтерфейс веб-серверу буде таким, що інші сервіси зможуть використовувати його.

Модульне розширення передбачає, що сам сервіс не буде виконуватись як програма, але його вихідний код буде використовуватись іншим сервісом.

Обидва методи мають ряд переваг і недоліків. В модульній взаємодії швидкість взаємодії швидша, так як в сервісній використовується мережа, проте при модульній взаємодії ціла система може працювати повільніше, так сервіси можуть виконуватись на різних ЕВМ. При сервісній взаємодії контексти процесів не переплітаються, так як модульна взаємодія має загальну пам’ять. При сервісній взаємодії програма може використовувати дані іншої програми, причому якщо програма представлена в скомпільованому вигляді модульна взаємодія з нею неможлива. При модульній взаємодії поведінку модуля можна змінити або перевизначити, що набагато тяжче зробити з сервісом.

При сервісній взаємодії сервіси можуть представляти загальний API або один сервіс буде звертатися до іншого. Також при сервісній взаємодії сервіси можуть використовувати загальні ресурси, такі як файлова система, пам’ять, база даних, інший сервіс.

Діаграма розгортання при сервісній взаємодії для системи обміну інформацією повинна мати вид, як на рисунку 3.3

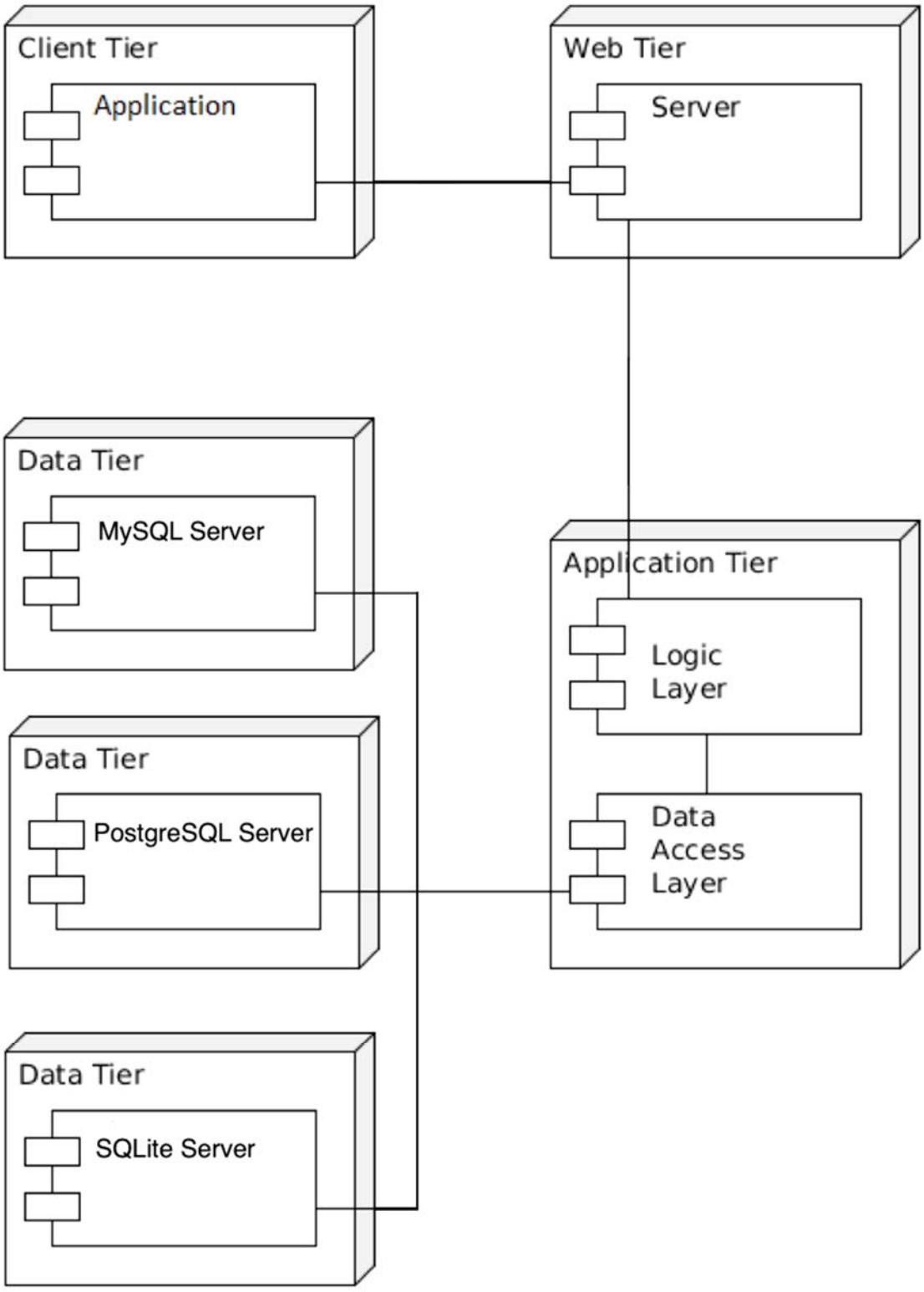


Рисунок 3.3 – Діаграма розгортання

7. Технічні засоби

Для коректного функціонування програмного продукту вимагається наявність ЕОМ, що задовольняє нормальну роботу:

Мінімальна конфігурація ЕОМ для забезпечення роботи програмного продукту:

- процесор з тактовою частотою 2 ГГц;
- оперативна пам’ять 6 Гб;
- 1 Гб вільного місця на жорсткому диску;
- клавіатура;
- миша;
- монітор з мінімальною роздільною здатністю дисплея 1024x720;
- CD дисковод;
- вихід у мережу інтернет;
- відеокарта з доступною відеопам’яттю не менше 512Мб.

Мінімальна конфігурація серверу для забезпечення роботи програмного продукту:

- ОС Linux 4.54;
- процесор Intel Xeon 2,66 GHz або вище;
- оперативна пам’ять 6 Гб;
- вихід у мережу інтернет;
- жорсткий диск: 5 Gb.

8. Виклик і завантаження

Для запуску програмної системи необхідно веб-браузер, бажано Google Chrome, наявність Node.js. Щоб запустити клієнтську частину, необхідно виконати наступні команди розгортування:

npm install -i -g yarn

cd <шлях_до_клієнтської_частини_проекту>

yarn

yarn start

Таким чином відбувається розгортування клієнтської частини, оскільки у якості шаблонізатору було використано React.js.

Для розгортування серверної частини програми необхідно виконати наступні дії:

Розгортування програми відбувається за допомогою спеціального файлу «Dockerfile», що слід передавати в систему Docker. Вміст файлу повинен бути наступним:

FROM python:3

WORKDIR /app

ADD . /app

RUN pip install pipenv

RUN pipenv install --dev

EXPOSE 80

CMD ["python", "app.py"]

Команди для розгортування такі:

- docker build dockerfilepath
- docker images
- docker run --name postgresql96 -e POSTGRES_PASSWORD=
  password -e POSTGRES_USER=user -h localhost -p 5432:5432 IMAGE_ID

9. Вхідні і вихдні дані

Вхідні дані:

- шаблони класів
- похибка у відсотковому відношенні
- кількість проведення досліджень при встановлених параметрах.
- вхідний набір даних для подальшого опрацьвання у вхідний вектор

Вихідними даними є:

- загальний час витрачений на роботу алгоритму;
- відображення .
- відображення побудованих .
- виведення помилок . .

10. Опис призначеного для користувача інтерфейсу

Огляд інтерфейсу користувача

Програмний продукт надає можливість користувачеві обробляти процеси, будь якого виду Для користування програмним комплексом користувач повинен володіти базовими знаннями користування персональним комп'ютером. Для того щоб користувач міг використовувати програмний комплекс у повному обсязі, необхідно ознайомитися з програмною документацію опис програми та керівництво користувача.

Розроблений програмний продукт дозволяє отримати результати та редагувати введені дослідником дані, зберігати введені дослідником дані на боці клієнту, отримувати доступ до своїх проектів у будь-якій точці світу за допомогою мережі «Інтернет». Програмний продукт також має можливість налаштування, за допомогою яких можна регулювати точність та швидкість оцінки. Системи можуть використовувати дану систему повністю самостійно, або впроваджувати сервісно, або модульно (в програмний код). Самостійне рішення даного програмного продукту має інтерфейс користувача, що переставлений на рисунках 7.1-7.3.

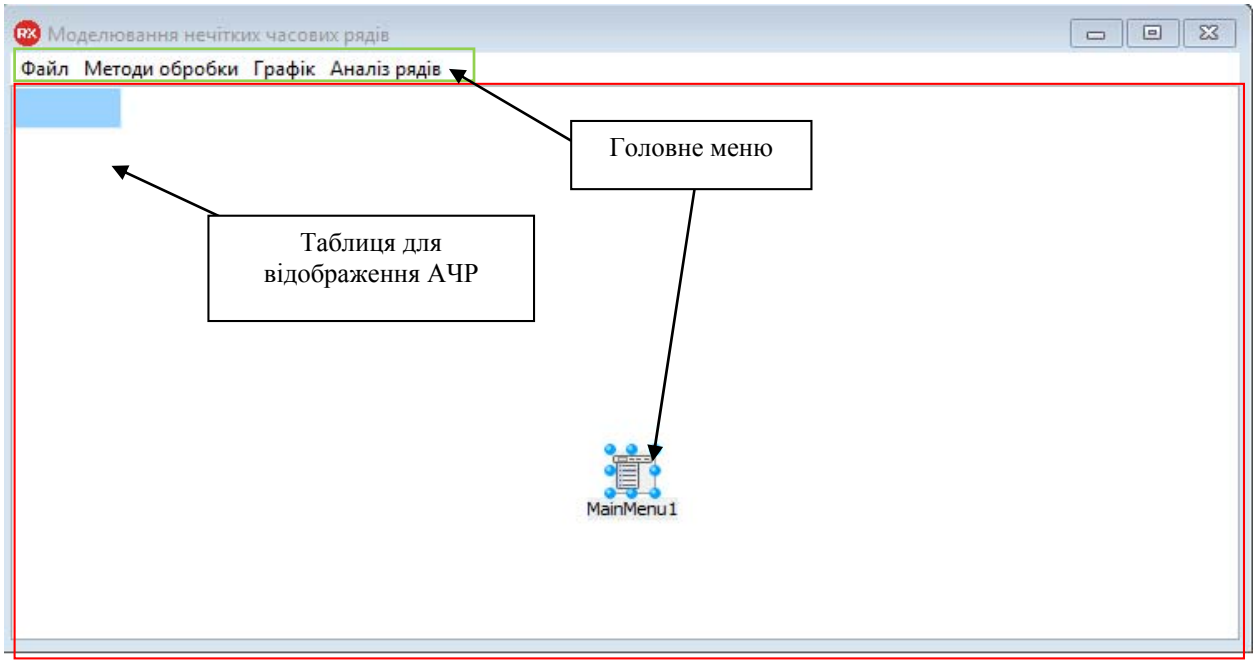


Рисунок 7.1 – Головне вікно програмного комплексу дослідження моделей послідовностей замовлень

Розроблений інтерфейс користувача повністю адаптивний для кожного виду пристроїв. Єдиною вимогою до таких пристроїв є можливість виконувати JavaScript код. Розроблені графіки результатів дослідження доволі інформативні і інтерактивні, що значить, що немає потреби в виконанні додаткового запиту.

Демонстрація REST інтерфейсу

Використання сервісу через інтерфейс користувача не єдиний інтерфейс даної системи. Система представляє зручний REST інтерфейс. Приклади взаємодії за допомогою команди curl по протоколу HTTP можна побачити на рисунках 7.6-7.8

```
$ curl -u miguel:python -i -X GET http://127.0.0.1:5000/login-psw
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 139
Server: Werkzeug/0.9.4 Python/2.7.3
Date: Mo, 28 May 2013 20:04:15 GMT

{
  "token": "eyJhbGciOiJIUzI1NiIsImV4cCI6MTM4NTY2OTY1NSwiaWF0IjoxMzg1NjY5MDU1fQ.eyJpZCI6MX0.XbOEFJkhjH
J5uRINh2JA1BPzXjSohKYDRT472wGOvjc"
}
```

Рисунок 7.4 – Встановлення з’єднання з сервером

```
$ curl -u eyJhbGciOiJIUzI1NiIsImV4cCI6MTM4NTY2OTY1NSwiaWF0IjoxMzg1NjY5MDU1fQ.eyJpZCI6MX0.XbOEFJkhjHJ5
uRINh2JA1BPzXjSohKYDRT472wGOvjc:unused -i -X POST http://127.0.0.1:5000/
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 30
Server: Werkzeug/0.9.4 Python/2.7.3
Date: Mo, 28 May 2013 20:05:08 GMT
```

Рисунок 7.5 – Завантаження файлу з JSON даними

Інтерфейс програмного коду

Ще одним інтерфейсом даного продукту можна назвати програмний код. основними функціями можна виділити наступні:

- upl_jsn – функція, що проводить перевірку і розшифрування json;
- get_editor_specs – дана функція повертає налаштування SQL2JSON модуля;
- check_cnct – функція, що перевіряє підключення користувача;
- make_rs – запускає опрацьовувач результатів;
- drop_data – знищує контейнери з СКРБД на серверах.
- do_calc – функція, що дозволяє виконати розрахунки (апроксимацію за допомогою мережі Хеммінга) у якості end-to-end взаємодії.

11.Повідомлення

У табл. 8.1 наведені повідомлення користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 8.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Error internet connection	Відсутнє інтернет з’єднання	Включити передачу даних
Invalid format image	Невірний формат файлу	Спробувати завантажити обраний файл ще раз або взяти інший

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130. 01165-01 І3 01 ЛЗ

«Дослідження моделей інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем»

Керівництво користувача. Керівництво з моделювання інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем

1116130. 01165-01 І3 01

Аркушів 20

2022

АНОТАЦІЯ

Документ 1116130.01165-01 І3 «Дослідження моделей інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем послідовностей замовлень» Керівництво користувача. Керівництво з моделювання та дослідження моделей інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем послідовностей замовлень.

У даному документі представлене призначення та умови застосування програми, підготовка до роботи, опис операцій, аварійні ситуації, рекомендації щодо засвоєння.

ЗМІСТ

ВСТУП

Програмний комплекс надає можливість користувачеві обробляти процеси, будь якого виду. Для користування програмним комплексом користувач повинен володіти базовими знаннями користування персональним комп’ютером. Для того щоб користувач міг використовувати програмний комплекс у повному обсязі, необхідно ознайомитися з програмною документацію опис програми та керівництво користувача.

1. ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

2. Програмний комплекс призначена для автоматичної обробки отриманих моделей, а також класифікації неточно визначених процесів за допомогою створеного програмного комплексу.

Умови застосування програми

Мінімальна конфігурація ЕОМ для забезпечення роботи програмного продукту:

- процесор з тактовою частотою 2 ГГц;
- оперативна пам’ять 6 Гб;
- 1 Гб вільного місця на жорсткому диску;
- клавіатура;
- миша;
- монітор з мінімальною роздільною здатністю дисплея 1024x720;
- CD дисковод;
- вихід у мережу інтернет;
- відеокарта з доступною відеопам’яттю не менше 2Гб.

3. Підготовка до роботи

Для запуску програмної системи необхідно веб-браузер, бажано Google Chrome, наявність Node.js. Щоб запустити клієнтську частину, необхідно виконати наступні команди розгортування:

```
npm install -i -g yarn
cd <шлях_до_клієнтської_частини_проекту>
yarn
yarn start
```

Таким чином відбувається розгортування клієнтської частини, оскільки у якості шаблонізатору було використано React.js.

Для розгортування серверної частини програми необхідно виконати наступні дії:

Розгортування програми відбувається за допомогою спеціального файлу «Dockerfile», що слід передавати в систему Docker. Вміст файлу повинен бути наступним:

```
FROM python:3
WORKDIR /app
ADD . /app
RUN pip install pipenv
RUN pipenv install --dev
EXPOSE 80
CMD [ "python", "app.py" ]
```

Команди для розгортування такі:

- docker build dockerfilepath
- docker images
- docker run --name postgresql96 -e POSTGRES_PASSWORD=
  password -e POSTGRES_USER=user -h localhost -p 5432:5432 IMAGE_ID

Для запуску програмної системи необхідно веб-браузер, бажано Google Chrome, наявність Node.js. Щоб запустити клієнтську частину, необхідно виконати наступні команди розгортування:

npm install -i -g yarn

cd <шлях_до_клієнтської_частини_проекту>

yarn

yarn start

Таким чином відбувається розгортування клієнтської частини, оскільки у якості шаблонізатору було використано React.js.

Для розгортування серверної частини програми необхідно виконати наступні дії:

Розгортування програми відбувається за допомогою спеціального файлу «Dockerfile», що слід передавати в систему Docker. Вміст файлу повинен бути наступним:

FROM python:3

WORKDIR /app

ADD . /app

RUN pip install pipenv

RUN pipenv install --dev

EXPOSE 80

CMD ["python", "app.py"]

Команди для розгортування такі:

- docker build dockerfilepath
- docker images
- docker run --name postgresql96 -e POSTGRES_PASSWORD=
  password -e POSTGRES_USER=user -h localhost -p 5432:5432 IMAGE_ID

4. ОПИС ОПЕРАЦІЙ

- Після запуску програмного комплексу, з’явиться головне вікно З головного вікна програмного комплексу виконуються функції, описані нижче:

Функція завантаження програмного комплексу зображено на рис. 3.2.

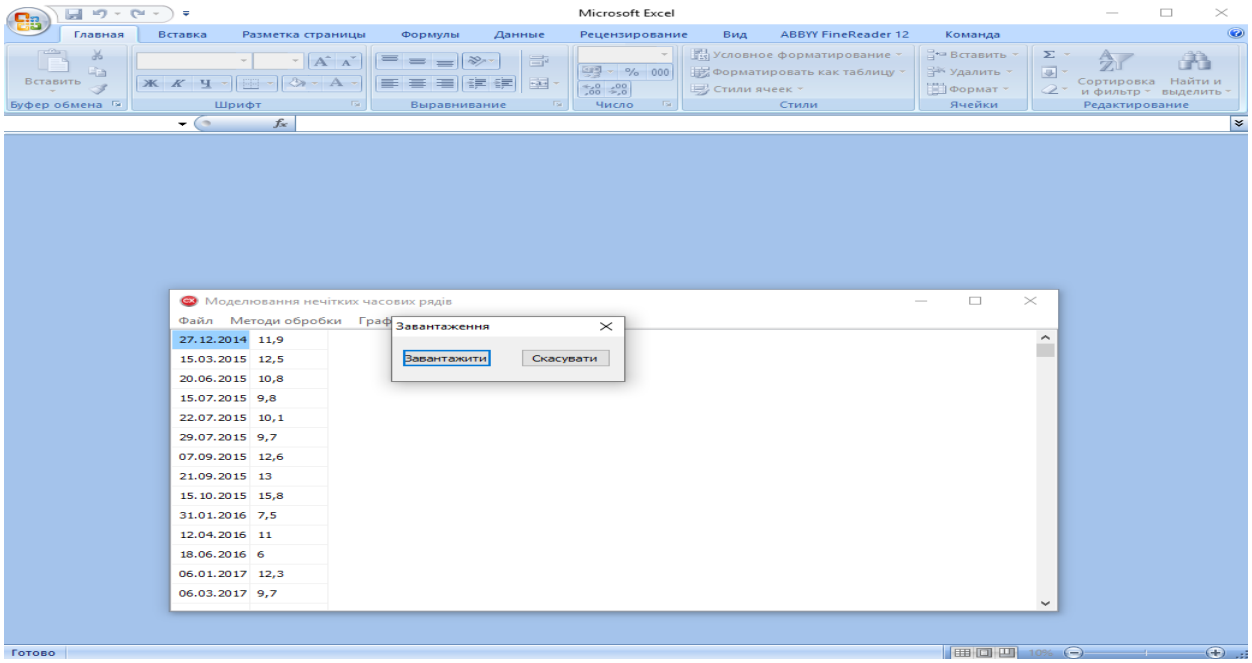


Рисунок 3.2 – Виконання завантаження

5. АВАРІЙНІ СИТУАЦІЇ

У таблиці 4.1 наведені повідомлення користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 4.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Error internet connection	Відсутнє інтернет з’єднання	Включити передачу даних
Invalid format image	Невірний формат файлу	Спробувати завантажити обраний файл ще раз або взяти інший

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖЕНО
1116130. 01165-01 12 01-ЛЗ

«Дослідження моделей інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем»

Текст програми
1116130. 01165-01 12 01
Аркушів 35

2022

АНОТАЦІЯ

Документ 1116130.01165-01 12 «Дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем» Текст програми, що входить до складу програмної документації до дипломного проекту.

У даному документі представлена структура програми, текст програми.

ЗМІСТ

1 СТРУКТУРА ПРОГРАМИ

Для розробленої системи існують наступні залежності:

- NLog.config – файл конфігурації логів системи роботи програмної системи.

Нижче описано структуру програми та основні модулі

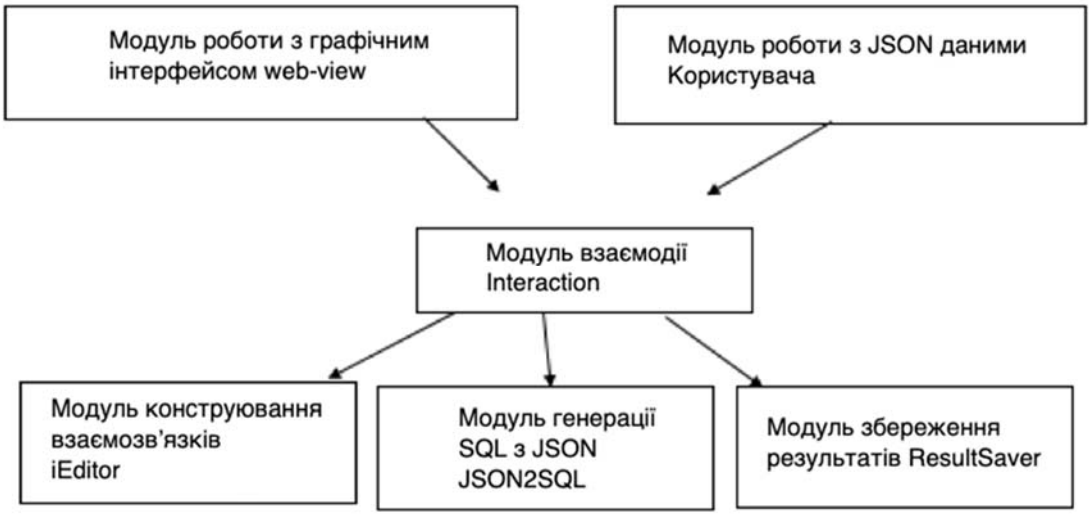


Рисунок 1.1 – Структура програми

У програмному продукті розроблено такі модулі:

- Interaction керує усіма робочими модулями у багатопоточному режимі;
- Web – view отримує дані стосовно користувача-дослідника, структури взаємозв'язків частин системи, та відображає користувачу результати у графічному інтерфейсі. Також даний модуль відповідає за інтерфейс сторінки програми;
- Модуль роботи з JSON отримує дані від користувача та генерує відповідну JSON структуру, для подальшої обробки такої структури даною системою. Даний модуль тісно взаємодіє з модулем iEditor;
- JSON2SQL отримує вхідні дані – JSON опис структури взаємозв'язків БД і виконує генерацію структури у форматі SQL

тісно взаємодіє із модулем роботи з JSON для подальшого зберігання шаблонів классів;

- iEditor – надає можливість внесення даних користувачем-дослідником;
- ResultSaver відповідає за збереження результатів дослідження на боці клієнту у вигляді .png зображення та додавання даних до БД для подальшого опрацювання.

2 ТЕКСТ ПРОГРАМИ

2.1 Текст файлу index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <link
href="https://fonts.googleapis.com/css?family=Lilita+One|Open+Sans|Righteous&display=swap"
rel="stylesheet">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
```

```
    <body>
      <noscript>You need to enable JavaScript to run this app.</noscript>
      <div id="root"></div>
      <!--
        This HTML file is a template.
        If you open it directly in the browser, you will see an empty page.

        You can add webfonts, meta tags, or analytics to this file.
        The build step will place the bundled scripts into the <body> tag.

        To begin the development, run `npm start` or `yarn start`.
        To create a production bundle, use `npm run build` or `yarn build`.
      -->
    </body>
  </html>
```

2.2 Текст файлу App.js

```
import React from 'react';
import { ConnectedRouter } from 'connected-react-router';
import { Provider } from 'react-redux';
import Layout from './Layout';
import history from '../config/history';
import createStore from '../store';
import Routes from '../pages';

const store = createStore({});

class App extends React.Component {

  state = {
    menuOpened: false,
  };

  handleMenuChangeState = () =>
    this.setState(({ menuOpened }) => ({menuOpened: !menuOpened}));
```

```
  render() {
    return (
      <Provider store={store}>
      <ConnectedRouter history={history}>
        <Layout
          menuOpened={this.state.menuOpened}
          handleMenuChangeState={this.handleMenuChangeState}
        >
          <Routes
            menuOpened={this.state.menuOpened}
            handleMenuChangeState={this.handleMenuChangeState}
          />
        </Layout>
      </ConnectedRouter>
      </Provider>
    );
  }
}

export default (App);
```

2.3 Текст файлу Label.js

```
import React from 'react';
import styled, {createGlobalStyle} from 'styled-components';
import clsx from 'clsx';
import { withStyles } from '@material-ui/core/styles';
import CssBaseline from '@material-ui/core/CssBaseline';
import AppBar from '@material-ui/core/AppBar';
import Toolbar from '@material-ui/core/Toolbar';
import IconButton from '@material-ui/core/IconButton';
import MenuIcon from '@material-ui/icons/Menu';
import Header from '../components/Header';

const drawerWidth = 240;
```

```
const ContentWrapper = styled.main`
  // background:red;
`;

const GlobalStyles = createGlobalStyle`
  .highcharts-credits {
    display: none;
  }
  * {
    margin: 0;
    padding: 0;
    font: 1vw 'Open Sans', sans-serif;
  }
  #root {
    display: flex;
    height: 100vh;
  }
  body {
    background-color: #fff;
    background-image: repeating-linear-gradient(45deg,
        rgba(0,0,0,.05) 25px,
        rgba(0,0,0,.05) 26px,
        transparent 26px,
        transparent 51px
      ),
      repeating-linear-gradient(
        -45deg,
        rgba(0,0,0,.05) 25px,
        rgba(0,0,0,.05) 26px,
        transparent 26px,
        transparent 51px
      );
  // background-position: 25px 35px;
```

```
  }
`


const styles = theme => ({
  appBar: {
    background: '#fff',
    zIndex: theme.zIndex.drawer + 1,
    color: '#000',
    transition: theme.transitions.create(['margin', 'width'], {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
  },
  appBarShift: {
    // width: `calc(100% - ${drawerWidth}px)`,
    // marginLeft: drawerWidth,
    transition: theme.transitions.create(['margin', 'width'], {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
  },
  menuButton: {
    marginRight: theme.spacing(2),
  },
  hide: {
    display: 'none',
  },
  drawer: {
    width: drawerWidth,
    flexShrink: 0,
  },
  drawerPaper: {
    width: drawerWidth,
  },
```

```
  drawerHeader: {
    display: 'flex',
    alignItems: 'center',
    padding: theme.spacing(0, 1),
    ...theme.mixins.toolbar,
    justifyContent: 'flex-end',
  },
  content: {
    flexGrow: 1,
    transition: theme.transitions.create('margin', {
      easing: theme.transitions.easing.sharp,
      duration: theme.transitions.duration.leavingScreen,
    }),
    marginLeft: 0,
  },
  contentShift: {
    transition: theme.transitions.create('margin', {
      easing: theme.transitions.easing.easeOut,
      duration: theme.transitions.duration.enteringScreen,
    }),
    marginLeft: drawerWidth,
  },
});

class Layout extends React.Component {

  render() {
    const { classes } = this.props;
    return (
      <React.Fragment>
      <GlobalStyles />
      <CssBaseline />
          <AppBar
            position="fixed"
```

```
          className={clsx(classes.appBar)}
        >
          <Toolbar>
            <IconButton
              color="inherit"
              aria-label="open drawer"
              onClick={this.props.handleMenuChangeState}
              edge="start"
              className={clsx(classes.menuButton)}
            >
              <MenuIcon />
            </IconButton>
            <Header />
          </Toolbar>
        </AppBar>
        <ContentWrapper
        className={clsx(classes.content, {
          [classes.contentShift]: this.props.menuOpened,
        })}
      >
        <div className={classes.drawerHeader} />
        {this.props.children}
      </ContentWrapper></React.Fragment>
    )
  }
}

export default withStyles(styles, { withTheme: true })(Layout);
```

2.4 Текст файлу Constructor.js

```
import React from 'react';
import { connect } from 'react-redux';
import styled from 'styled-components';
import { withStyles } from '@material-ui/core/styles';
import clsx from 'clsx';
```

```
import { bindActionCreators } from 'redux';
import Button from '@material-ui/core/Button';
import TextField from '@material-ui/core/TextField';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogContentText from '@material-ui/core/DialogContentText';
import DialogTitle from '@material-ui/core/DialogTitle';
import { setTableEditorOpened, saveTable, addTable, editTable } from '../modules';
import InputLabel from '@material-ui/core/InputLabel';
import MenuItem from '@material-ui/core/MenuItem';
import FormHelperText from '@material-ui/core/FormHelperText';
import FormControl from '@material-ui/core/FormControl';
import Select from '@material-ui/core/Select';

const Div = styled.div`
  display: flex;
  width: 100%;
  margin: 30px 0;
`;

const dataTypes = [
  'Bit',
  'Tinyint',
  'Smallint',
  'Int',
  'Bigint',
  'Decimal',
  'Numeric',
  'Float',
  'Real',
  'Date',
  'Time',
  'Datetime',
```

```
  'Timestamp',
  'Year',
  'Char',
  'Varchar',
  'Text',
  'Nchar',
  'Nvarchar',
  'Ntext',
  'Binary',
  'Varbinary',
  'XML',
  'JSON',
  'Blob',
];

const tableField = {caption: '', isKey: false, type: null};

const styles = theme => ({
  selectRoot: {
  marginRight: 10,
  width: 130,
  },
  textFieldRoot: {
  width: 250,
  },
});

class TableDialogForm extends React.Component {

  state = {
  table: {
    name: '',
    value: {
    fields: [{...tableField}],
```

```
  geometry: {
    x: 0,
    y: 0,
    w: 350,
    h: 90, // 43 - height, 2 - border
  }
  },
}
}

componentDidMount() {
if (this.props.editData) {
  console.log(this.props.editData)
  this.setState({
  table: JSON.parse(JSON.stringify(this.props.editData)),
  })
}
}

closeTableEditor = () => {
this.props.editTable(null);
this.props.setTableEditorOpened(false)
}

eraseAndClose = () => {
this.setState({
  table: {
    name: '',
    value: {
      fields: [{...tableField}],
      geometry: {
        x: 0,
        y: 0,
        w: 350,
```

```
      h: 90, // 43 - height, 2 - border
    }
   },
  }
});
this.closeTableEditor();
}

handleCancel = () => this.eraseAndClose();

handleSave = () => {
// console.log(JSON.stringify(this.state.table));
this.props.addTable(JSON.parse(JSON.stringify(this.state.table)));
this.eraseAndClose();
}

addField = () => this.setState((prevState) => ({
  table: {
    ...prevState.table,
    value: {
      ...prevState.table.value,
      geometry: {
        ...prevState.table.value.geometry,
        h: prevState.table.value.geometry.h + 45,
      },
      fields: [
        ...prevState.table.value.fields,
        {...tableField},
      ]
    },
  }
}))

textFieldChanged = (i) => (e) => {
```

```
const x = e.target.value;
this.setState(prevState => ({
  table: {
  ...prevState.table,
  value: {
    ...prevState.table.value,
      fields: [
        ...prevState.table.value.fields.slice(0, i),
      {...prevState.table.value.fields[i], caption: x},
      ...prevState.table.value.fields.slice(i + 1),
      ]
  }
  }
}));
}

handleSelectFieldType = (i) => (e) => {
const x = e.target.value;
this.setState(prevState => ({
  table: {
  ...prevState.table,
  value: {
    ...prevState.table.value,
      fields: [
        ...prevState.table.value.fields.slice(0, i),
      {...prevState.table.value.fields[i], type: x},
      ...prevState.table.value.fields.slice(i + 1),
      ]
  }
  }
}));
}

handleNameChanged = (e) => {
```

```
const x = e.target.value;
this.setState(prevState => ({
  table: {
  ...prevState.table,
  name: x,
  }
}))
}
render() {
console.log("REBUILD", this.state)
return (
  <Dialog onClose={this.closeTableEditor} open={this.props.opened} aria-labelledby="form-dialog-title">
  <DialogTitle id="form-dialog-title">Add table</DialogTitle>
  <DialogContent>
    <DialogContentText>
    Fill all requied fields to create a new table
    </DialogContentText>
    <TextField
    onChange={this.handleNameChanged}
    autoFocus
    fullWidth
    placeholder="Table name"
    defaultValue={this.state.table.name}
    type="text"
    />
    {
    this.state.table.value.fields.map((field, i) => (<Div key={i}>
      <Select
        labelId="demo-simple-select-label"
        id="demo-simple-select"
        value={field.type || ""}
        className={clsx(this.props.classes.selectRoot)}
        displayEmpty
```

```
      onChange={this.handleSelectFieldType(i)}
    >
      <MenuItem value={""} disabled>
      Type
      </MenuItem>
      {
      dataTypes.map(type => (
        <MenuItem key={type} value={type}>{type}</MenuItem>
      ))
      }
    </Select>
    <TextField
    key={i}
    onChange={this.textFieldChanged(i)}
    className={clsx(this.props.classes.textFieldRoot)}
    placeholder="Field"
    defaultValue={field.caption}
    type="text"
    />
  </Div>))
  }
  <Button onClick={this.addField} flat>+ Add field</Button>
 </DialogContent>
 <DialogActions>
  <Button onClick={this.handleCancel} color="primary">
  Cancel
  </Button>
  <Button onClick={this.handleSave} color="primary">
  Add
  </Button>
 </DialogActions>
 </Dialog>
);
}
```

```
}

const mapStateToProps = (state) => ({
  opened: state.editor.tableEditorOpened,
  editData: state.editor.editData,
});
const mapDispatchToProps = dispatch => bindActionCreators({
  setTableEditorOpened,
  saveTable,
  addTable,
  editTable,
}, dispatch);

export default connect(mapStateToProps, mapDispatchToProps)(withStyles(styles, { withTheme: true
})(TableDialogForm));
```

2.5 Текст файлу ConstructorMenu.js

```
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import styled from 'styled-components';
import { connect } from 'react-redux';
import { bindActionCreators } from 'redux';
import List from '@material-ui/core/List';
import Divider from '@material-ui/core/Divider';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
import ListItemText from '@material-ui/core/ListItemText';
import AddCircleIcon from '@material-ui/icons/AddCircle';
import RemoveCircleIcon from '@material-ui/icons/RemoveCircle';
import SaveIcon from '@material-ui/icons/Save';
import SaveAltIcon from '@material-ui/icons/SaveAlt';
import Menu from '../../../components/Menu';
import { setTableEditorOpened, setTables } from '../modules';
import history from '../../../config/history';
const Title = styled.span`
```

```
  font-size: .95em;
`;
const Button = styled.button`
  padding: 10px;
  border-radius: 100px;
  font: bold 1em 'Open Sans';
  text-transform: uppercase;
  position: absolute;
  cursor: pointer;
  width: 200px;
  bottom: 20px;
  left: 0;
  right: 0;
  margin: auto;
  outline: none;
  transition: all .2s ease-in-out;
  background: #fff;
  border: 2px solid ${({disabled}) => disabled ? 'grey' : 'rgba(255,20,20,.81)'};
  color: ${({disabled}) => disabled ? 'grey' : 'rgba(255,20,20,.81)'};

  &:hover {
    background: ${({disabled}) => disabled ? 'grey' : 'rgba(255,20,20,.81)'};;
    color: #fff;
    border: 2px solid #fff;
  }
  &:active {
    background: ${({disabled}) => disabled ? 'grey' : 'rgba(255,20,20,.81)'};;
  }
`;
const useStyles = makeStyles(theme => ({
  listItem: {
    paddingTop: 15,
    paddingBottom: 15,
  },
```

```
  menu: {
    position: 'relative',
  }
}));

const ConstructorMenu = ({ menuOpened, handleMenuClose, setTableEditorOpened, tables, setTables })
=> {
  const classes = useStyles();

  const generateBlobData = data => (
    URL.createObjectURL(new Blob(
      [data],
      { type: 'text/plain' },
    ))
  );

  const handleSave = (data) => {
    const blobData = generateBlobData(JSON.stringify(tables));
        const link = document.createElement('a');
        link.setAttribute('href', blobData);
        link.setAttribute('download', 'tables.dbenchmark.json');
        document.body.appendChild(link);
        link.click();
        return document.body.removeChild(link);
  }

  return (
    <Menu opened={menuOpened} handleClose={handleMenuClose} classes={{root: classes.menu}}>
      <List>
        <ListItem onClick={() => setTables(require('../../../tables.dbenchmark.json'))} classes={{root:
classes.listItem}} button key="Load data">
          <ListItemIcon><SaveAltIcon /></ListItemIcon>
          <Title>Load data</Title>
        </ListItem>
```

```
      <ListItem onClick={handleSave} classes={{root: classes.listItem}} button key="Save data">
        <ListItemIcon><SaveIcon /></ListItemIcon>
        <Title>Save data</Title>
      </ListItem>
    </List>
    <Divider />
    <List>
      <ListItem onClick={() => setTableEditorOpened(true)} classes={{root: classes.listItem}} button
key="Add table">
        <ListItemIcon><AddCircleIcon /></ListItemIcon>
        <Title>Add table</Title>
      </ListItem>
      <ListItem classes={{root: classes.listItem}} button key="Remove table">
        <ListItemIcon><RemoveCircleIcon /></ListItemIcon>
        <Title>Remove table</Title>
      </ListItem>
    </List>
    <Divider />
    <Button onClick={() => setTimeout(() => history.push('/results'), Math.floor(Math.random() * 2000)
* Object.keys(tables).length)} disabled={!Object.keys(tables).length}>Start bench</Button>
   </Menu>
  );
};

const mapStateToProps = (state) => !console.log(state.editor.tables) && ({
  tables: state.editor.tables,
});
const mapDispatchToProps = dispatch => bindActionCreators({
  setTableEditorOpened,
  setTables,
}, dispatch);

export default connect(mapStateToProps, mapDispatchToProps)(ConstructorMenu);
```

2.6 Текст файлу Results.js

```
  import React from 'react';
import Menu from '../../../components/Menu';
import List from '@material-ui/core/List';
import Divider from '@material-ui/core/Divider';
import ListItem from '@material-ui/core/ListItem';
import ListItemIcon from '@material-ui/core/ListItemIcon';
import ListItemText from '@material-ui/core/ListItemText';
import InboxIcon from '@material-ui/icons/MoveToInbox';
import MailIcon from '@material-ui/icons/Mail';

const instances = [
 'http://ec2-54-166-128-20.compute-1.amazonaws.com',
 'http://ec2-57-168-117-32.compute-1.amazonaws.com',
 'http://ec2-61-106-107-14.compute-1.amazonaws.com',
];

const ResultsMenu = ({ opened, handleClose, onSave }) => (
 <Menu opened={opened} instances={instances} handleClose={handleClose}>
   <List>
     <ListItem button onClick={onSave}>
       <ListItemIcon>{<InboxIcon />}</ListItemIcon>
       <ListItemText primary={"Save Results"} />
     </ListItem>
   </List>
   <Divider />
   <List>
     <ListItem button>
       <ListItemIcon>{<MailIcon />}</ListItemIcon>
       <ListItemText primary={"Load Results"} />
     </ListItem>
   </List>
 </Menu>
);
```

```
export default ResultsMenu;
```

Main.js

```
  $(document).ready(function() {

  // Attach '.active' to current page's link in navbar
  $("ul.nav li").each(function(i, e) {
      var path = $(e).find('a').attr('href');
      if(path === window.location.pathname) {
          $(e).addClass("active");
      }
  });
});
 function valid(form){
 let fail = false;
 let name = form.name.value;
 let password = form.password.value;
 if(name == "" || name == " ")
  fail = "You didn`t enter your name!";
 else if(password =="" || " ")
  fail = "You didn`t enter your password!";

 if(fail){
  alert(fail);
}
```

2.9 Текст файлу server.py

```
import os
import sqlite3

from flask import Flask, jsonify, make_response, redirect, render_template, request, session, url_for
import settings
app = Flask(__name__)
app.config.from_object(settings)
# Helper functions
def _get_message(id=None):
```

```
    """Return a list of message objects (as dicts)"""
    with sqlite3.connect(app.config['DATABASE']) as conn:
        c = conn.cursor()

        if id:
            id = int(id)  # Ensure that we have a valid id value to query
            q = "SELECT * FROM messages WHERE id=? ORDER BY dt DESC"
            rows = c.execute(q, (id,))
        else:
            q = "SELECT * FROM messages ORDER BY dt DESC"
            rows = c.execute(q)
        return [{'id': r[0], 'dt': r[1], 'message': r[2], 'sender': r[3]} for r in rows]
def _add_message(message, sender):
    with sqlite3.connect(app.config['DATABASE']) as conn:
        c = conn.cursor()
        q = "INSERT INTO messages VALUES (NULL, datetime('now'),?,?)"
        c.execute(q, (message, sender))
        conn.commit()
        return c.lastrowid
def _delete_message(ids):
    with sqlite3.connect(app.config['DATABASE']) as conn:
        c = conn.cursor()
        q = "DELETE FROM messages WHERE id=?"

        # Try/catch in case 'ids' isn't an iterable
        try:
            for i in ids:
                c.execute(q, (int(i),))
        except TypeError:
            c.execute(q, (int(ids),))

        conn.commit()
# Standard routing (server-side rendered pages)
@app.route('/', methods=['GET', 'POST'])
```

```
def home():
    if request.method == 'POST':
        _add_message(request.form['message'], request.form['username'])
        redirect(url_for('home'))

    return render_template('index.html', messages=_get_message())
#stud
@app.route('/student')
def student():
  return render_template('student.html', messages=_get_message())
    # return render_template('student.html')
#/stud

@app.route('/about')
def about():
    return render_template('about.html')
@app.route('/admin', methods=['GET', 'POST'])
def admin():
    if not 'logged_in' in session:
        return redirect(url_for('login'))

    if request.method == 'POST':
        # This little hack is needed for testing due to how Python dictionary keys work
        _delete_message([k[6:] for k in request.form.keys()])
        redirect(url_for('admin'))

    messages = _get_message()
    messages.reverse()
    return render_template('admin.html', messages=messages)

@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
```

```
        if request.form['username'] != app.config[sets] or request.form['password'] != app.config[editor]:
            error = 'Invalid username and/or password'
        else:
            session['logged_in'] = True
            return redirect(url_for('admin'))
    return render_template('login.html', error=error)
@app.route('/logout')
def logout():
    session.pop('logged_in', None)
    return redirect(url_for('home'))
# RESTful routing (serves JSON to provide an external API)
@app.route('/messages/api', methods=['GET'])
@app.route('/messages/api/<int:id>', methods=['GET'])
def get_message_by_id(id=None):
    messages = _get_message(id)
    if not messages:
        return make_response(jsonify({'error': 'Not found'}), 404)
    return jsonify({'messages': messages})
@app.route('/messages/api', methods=['POST'])
def create_message():
    if not request.json or not 'message' in request.json or not 'sender' in request.json:
        return make_response(jsonify({'error': 'Bad request'}), 400)
    id = _add_message(request.json['message'], request.json['sender'])
    return get_message_by_id(id), 201
@app.route('/messages/api/<int:id>', methods=['DELETE'])
def delete_message_by_id(id):
    _delete_message(id)
    return jsonify({'result': True})
if __name__ == '__main__':

    # Test whether the database exists; if not, create it and create the table
    if not os.path.exists(app.config['DATABASE']):
        try:
            conn = sqlite3.connect(app.config['DATABASE'])
```

```
        # Absolute path needed for testing environment
        sql_path = os.path.join(app.config['APP_ROOT'], 'db_init.sql')
        cmd = open(sql_path, 'r').read()
        c = conn.cursor()
        c.execute(cmd)
        conn.commit()
        conn.close()
    except IOError:
        print "Couldn't initialize the database, exiting..."
        raise
    except sqlite3.OperationalError:
        print "Couldn't execute the SQL, exiting..."
        raise
app.run(host='0.0.0.0')
```

messenger_tester.py

```
import json
import os
import sqlite3
import tempfile
import unittest
import messenger
import settings
class MessengerBaseTestCase(unittest.TestCase):
    def setUp(self):
        """Create a temporary database and create the needed table"""
        messenger.app.config.from_object(settings)
        self.db_fd, messenger.app.config['DATABASE'] = tempfile.mkstemp()
        messenger.app.config['TESTING'] = True
        self.app = messenger.app.test_client()
        with sqlite3.connect(messenger.app.config['DATABASE']) as conn:
            sql_path = os.path.join(messenger.app.config['APP_ROOT'], 'db_init.sql')
            cmd = open(sql_path, 'r').read()
            c = conn.cursor()
            c.execute(cmd)
```

```
        conn.commit()
    def tearDown(self):
        os.close(self.db_fd)
        os.unlink(messenger.app.config['DATABASE'])
    # Helper functions for testing login/logout
    def login(self, username, password):
        return self.app.post('/login', data=dict(
            username=username,
            password=password
        ), follow_redirects=True)
    def logout(self):
        return self.app.get('/logout', follow_redirects=True)
class MessengerEmptyTestCase(MessengerBaseTestCase):
    def test_get_empty_db(self):
        """A GET on an empty database should return 404 Not found"""
        rv = self.app.get('/messages/api')
        self.assertEqual(rv.status_code, 404)
        self.assertIn('"error": "Not found"', rv.data)
    def test_post_empty_db(self):
        """POST a message to an empty database"""
        data = json.dumps({'message': 'Test', 'sender': 'jgoney'})
        rv = self.app.post('/messages/api', data=data, content_type='application/json')

        # Check error code
        self.assertEqual(rv.status_code, 201)
        # Decode JSON response and store object id
        resp = json.loads(rv.data)
        self.assertEqual('jgoney', resp['messages'][0]['sender'])
        self.assertEqual('Test', resp['messages'][0]['message'])
    def test_post_bad_request(self):
        """Not setting the Content-Type will return a 400 status code"""
        data = json.dumps({'message': 'Test', 'sender': 'jgoney'})
        rv = self.app.post('/messages/api', data=data)
```

```
        # Check error code
        self.assertEqual(rv.status_code, 400)
class MessengerSingleTestCase(MessengerBaseTestCase):
    def setUp(self):
        """Add a single message to the database for testing"""
        super(MessengerSingleTestCase, self).setUp()
        with sqlite3.connect(messenger.app.config['DATABASE']) as conn:
            cmd = "INSERT INTO messages VALUES (NULL, datetime('now'), 'Test', 'jgoney')"
            c = conn.cursor()
            c.execute(cmd)
            conn.commit()
    def test_get_single(self):
        """Tests fetching a single message"""
        rv = self.app.get('/messages/api/1')
        # Check error code
        self.assertEqual(rv.status_code, 200)
        resp = json.loads(rv.data)
        self.assertEqual('jgoney', resp['messages'][0]['sender'])
        self.assertEqual('Test', resp['messages'][0]['message'])
    def test_delete_single(self):
        """Tests deleting a single message"""
        rv = self.app.delete('/messages/api/1')
        self.assertEqual(rv.status_code, 200)
        self.assertIn('"result": true', rv.data)
    def test_delete_wrong_id(self):
        """Tests deleting a single message with a non-int id"""
        rv = self.app.delete('/messages/api/foo')
        self.assertEqual(rv.status_code, 404)
        self.assertIn('<title>404 Not Found</title>', rv.data)
class MessengerMultipleTestCase(MessengerBaseTestCase):
    def setUp(self):
        """Adds multiple messages to the database for testing"""
        super(MessengerMultipleTestCase, self).setUp()
        with sqlite3.connect(messenger.app.config['DATABASE']) as conn:
```

```
        c = conn.cursor()
        for i in xrange(5):
            cmd = "INSERT INTO messages VALUES (NULL, datetime('now'), ?, ?)"
            m = 'message #%s' % (i,)
            s = 'sender #%s' % (i,)
            c.execute(cmd, (m, s))
        conn.commit()
    def test_delete_multiple(self):
        """Tests deleting multiple messages at once"""
        rv = self.app.delete('/messages/api/1')
        self.assertEqual(rv.status_code, 200)
        self.assertIn('"result": true', rv.data)
    def test_delete_multiple_admin_page(self):
        """Tests deleting multiple messages at once through admin interface"""
        # Set the proper session variable so that login works
        with self.app as c:
            with c.session_transaction() as session:
                session['logged_in'] = True
            # Assert that the setup messages are there
            rv = c.get('/admin', follow_redirects=True)
            self.assertIn('message #0', rv.data)
            self.assertIn('message #1', rv.data)
            self.assertEqual(rv.status_code, 200)
            # Assert that after POST, the chosen messages are deleted
            rv = c.post('/admin', data=dict(
                delete1='on',
                delete2='on'
            ), follow_redirects=True)
            self.assertNotIn('message #0', rv.data)
            self.assertNotIn('message #1', rv.data)
            self.assertEqual(rv.status_code, 200)

class MessengerMiscTestCase(MessengerBaseTestCase):
        # Assert that message has been added
```

```
        self.assertIn('test user', rv.data)
        self.assertIn('test message', rv.data)
        self.assertEqual(rv.status_code, 200)

    def test_admin_page_redirect(self):
        """Tests that the admin page redirects work as intended"""
        rv = self.app.get('/admin', follow_redirects=True)

        self.assertIn('Please login:', rv.data)
        self.assertEqual(rv.status_code, 200)

        with self.app as c:
            with c.session_transaction() as session:
                session['logged_in'] = True
            rv = c.get('/admin', follow_redirects=True)
        self.assertIn('No messages found.', rv.data)
        self.assertEqual(rv.status_code, 200)
if __name__ == '__main__':
    unittest.main()
```

db_init.sql

```
CREATE TABLE IF NOT EXISTS messages (
  id INTEGER PRIMARY KEY,
  dt TEXT NOT NULL,
  message TEXT NOT NULL,
  sender TEXT NOT NULL
);
```

Init.py

```
from settings_common import *
# Conditionally import additional settings depending on whether we're developing or in production
# Can be set, for example by checking for a given environmental variable, or by detecting the hostname
production = False
if production:
    from settings_prod import *
```

```
else:
    from settings_dev import *
"""Common settings are defined here. Only upper case variables are exported."""
import os
APP_ROOT = os.path.dirname(os.path.dirname(__file__))
DB_NAME = 'main.db'
DATABASE = os.path.join(APP_ROOT, DB_NAME)
```

Settingsdev.p

```
"""Development settings are defined here. Only upper case variables are exported."""
DEBUG = True
SECRET_KEY = 'development key'
USERNAME = 'admin'
PASSWORD = '123'
```

2.12 Текст файлу hamming_network.py

```
from computes import Computes

class HammingNetwork(object):

    e = 0.25

    # templates = [
    #     [1, -1, 1, -1, 1, -1, 1, -1, 1],
    #     [-1, 1, -1, 1, 1, 1, -1, 1, -1],
    #     [1, 1, 1, 1, -1, 1, 1, 1, 1],
    #     [1, -1, -1, -1, 1, -1, 1, -1, 1]
    # ]

    templates = [
        [1,1,-1,-1,1,1],
        [-1,1,-1,-1,1,1],
        [1,-1,-1,-1,1,-1],
    ]
```

```
def calculate_weight_coeffs(self):
        weights = []
        for i in self.templates:
                weight_row = []
                for j in i:
                        weight_row.append(j / 2)
                weights.append(weight_row)
        return weights

def calculate_feedback_weight_coeffs(self):
        weights = []
        k = len(self.templates)
        for i in range(k):
                weight_row = []
                for j in range(k):
                        if i == j:
                                weight_row.append(1)
                        else:
                                weight_row.append(-0.00001)
                weights.append(weight_row)
        return weights

def threshold(self):
        return len(self.templates[0]) / 2

def activation_function(self, s):
        t = self.threshold()
        result = s
        if s < 0:
                result = 0
        elif s > t:
                result = t
        return result
```

```
def calculate_first_layer(self):
        weights = self.calculate_weight_coeffs()
        result = []
        for weights_row in weights:
                # sum = self.threshold()
                sum = 0
                for j in range(len(self.input_vector)):
                        sum = sum + self.input_vector[j] * weights_row[j]
                result.append(sum)
        return result


def vector_with_activation_f(self, synapses):
        result = []
        for synapse in synapses:
                result.append(self.activation_function(synapse))
        return result


def vector_distance(self, a, b):
        vectors_substraction = Computes.vector_subtraction_abs(a,b)
        print(f'{a} - {b} = {vectors_substraction}')
        distance = Computes.vector_length(vectors_substraction)
        return distance


def approximate(self):
        start_layer = self.calculate_first_layer()
        feedback_weights = self.calculate_feedback_weight_coeffs()
        start_layer_activated = self.vector_with_activation_f(start_layer)
        current_layer = []
        for feedback_weight_row in feedback_weights:
                sum = 0
                for j in range(len(feedback_weight_row)):
                        sum = sum + feedback_weight_row[j] * start_layer_activated[j]
                current_layer.append(sum)
        current_layer_activated = self.vector_with_activation_f(current_layer)
```

```
        while(self.vector_distance(current_layer_activated, start_layer_activated) > self.e):
                start_layer_activated = current_layer_activated
                current_layer = []
                for feedback_weight_row in feedback_weights:
                        sum = 0
                        for j in range(len(feedback_weight_row)):
                                sum = sum + feedback_weight_row[j] * start_layer_activated[j]
                        current_layer.append(sum)
                current_layer_activated = self.vector_with_activation_f(current_layer)



def __init__(self, input_vector):
        super(HammingNetwork, self).__init__()
        self.input_vector = input_vector
        self.approximate()
```

Моделювання та дослідження інтелектуальних процедур і програмних засобів оптимізації потоків замовлень сервісних систем

Скалозуб В.В., Терлецький І.О., Терленко А.П.
Український державний університет науки і технологій
Скалозуб М. PayPal, Швеція

Завдання оптимізації потоків замовлень у сервісних, обслуговуючих, системах (ОПЗСеС) виникає у багатьох технологіях і виробництвах, є змістовним та досить поширеним. При цьому для формування процедур оптимізації виконується урахування все більшого набору властивостей досліджуваних потоків. Складність інформаційних, технологічних та інших процесів у сервісних системах (СеС), часто впливає на можливості отримати достовірні та точно визначені дані про характеристики потоків замовлень. Зазначене суттєво впливає на результати оптимізації роботи СеС. Крім того, певні потоки можуть мати велику кількість компонентів з різними властивостями, характеристики яких також відомі неточно. Для забезпечення ефективного вибору обслуговуючого «приладу» можливо застосувати методи інтелектуальних систем, у нас – процедури класифікації замовлень за неточно визначеними даними. Призначення процедур полягає у визначенні класу, який показує «виконавця». Аналіз публікацій свідчить, що натепер завдання із розвитку та розробки таких процедур класифікації при неповних та неточно визначених даних мають значний теоретичний та науковий інтерес. Разом з цим дослідження застосування подібних інтелектуальних процедур при оптимізації потоків замовлень в обслуговуючих системах проведені не в повній мірі.

В доповіді приведені результати щодо розвитку інтелектуальних процедур оптимізації потоків замовлень у СеС при неточно визначених характеристиках даних на основі моделі асоціативної пам’яті мережі Хеммінга (МХ). Такі класичні МХ дозволяють виконувати класифікацію об’єктів (замовлень) при збурених даних, якщо властивості елементів оцінюються значеннями з множини {– 1; + 1}. Мета – розвиток і розширення кола постановок завдань, а також удосконалення математичних моделей та процедур оптимізації потоків замовлень СеС на основі інтелектуальних процедур мережі МХ при неточно визначених характеристиках даних, далі МХН. В роботі досліджувалися можливості використання в якості моделей нечітких множин, а також показників достовірності експертних систем, коефіцієнтів впевненості CF(A) з множини [– 1; + 1].

Результати розробок і досліджень полягали в наступному: - виконано аналіз моделей та інтелектуальних процедур завдань оптимізації потоків замовлень в сервісних системах (ОПЗСеС); - запропоновано нові постановки завдань ОПЗСеС (система паркінгу авто; призначення фахівців на посаду в ІТ проєктах тощо), для реалізації яких застосовуються процедури класифікації МХН; - удосконалені математичні моделі ОПЗСеС на основі інтелектуальних процедур МХН, які враховують різні моделі вихідних даних; - розроблено програмні засоби із класифікації компонентів послідовностей замовлень на основі модифікованих процедур МХН; - проведено широкий і всебічний числовий експеримент, який підтвердити достовірність та ефективність запропонованих моделей і методів МНХ; - отримані рекомендації стосовно застосування МНХ при неточно визначених характеристиках даних, які враховують запропоновані форми моделей вихідних даних (нечіткі множини (НМ), коефіцієнти впевненості CF(A)). При використанні НМ були запропоновані моделі кодування, за допомогою яких нечіткі величини подавалися як в МХ {– 1; + 1}, а для моделей CF(A) використовувалися процедура, що безпосередньо застосовувала схеми МХ. Числові дослідження показали певні переваги моделі CF(A), яка завжди забезпечувала потрібний результат класифікації, при тому що в моделі з НВ результати класифікації не завжди були однозначними.

В доповіді представлені розроблені моделі перекодування НМ, а також їх порівняльна ефективність, програмна реалізація розроблених моделей класифікації об’єктів з неточно визначеними параметрами МХН, результати проведених числових експериментів, а також процедура ОПЗСеС на основі модифікованих моделей асоціативної пам’яті МХН.