Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____
(рівень вищої освіти)

Секвенсори аналізу даних в метриці подібності-відмінності
(тема)

Виконав: студент II курсу групи СКСм-20-1

_____Семьонов А.В._____

Спеціальність
_____*123 – Комп'ютерна інженерія*_____
(код і повна назва спеціальності)

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

Освітня програма
Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник проф. Чумаченко С.В.

Допускається до захисту

Зав. каф. АПОТ  Чумаченко С.В.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ:
Зав. кафедри АПОТ
_____Чу
маченко С.В.
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Семьонову Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Секвенсори аналізу даних в метриці подібності/відмінності

затверджена наказом по університету від 04 листопада 2021 р. № 1625 Ст
2. Термін подання студентом роботи до екзаменаційної комісії _17_ грудня 2021 р.
3. Вихідні дані до роботи (проекту)
Мова програмування C#
Комп'ютинг розпізнавання, моделі, цифрова схема, арітектура, моделювання
Мова програмування C++

Перелік питань, що потрібно опрацювати у роботі
Аналіз проблемної галузі (стан питання)
Інтерфейс редагування об'єкта
Програмно-апаратний модуль обчислення подібності/відмінності об'єктів

Аналіз і тестування отриманої структури секвенсора

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

слайди презентації -25

6. Консультанти розділів роботи (проекту)

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

7. Дата видачі завдання 01.09.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.09.2021 − 05.09.2021	
2	Аналіз проблемної області	06.09.2021 − 21.09.2021	
3	Аналіз джерел з проблемної галузі	22.09.2021 - 05.10.2021	
3	Опис інтерфейсу редагування об'єкта	06.10.2021 - 19.10.2021	
4	Опис програмно-апаратного модуля обчислення подібності-відмінності об'єктів	20.10.2021 − 01.11.2021	
5	Аналіз і тестування отриманої структури секвенсора	02.11.2021 − 20.11.2021	
6	Інфраструктура подібності-відмінності для діагностичного комп'ютингу	21.11.2021 – 01.12.2021	
7	Оформлення пояснювальної записки	02.12.2021 − 10.12.2021	
8	Оформлення графічного матеріалу	11.12.2021 - 15.12.2021	
9	Перевірка виконаного проекту керівником	16.12.2021 - 17.12.2021	

Студент _____ Семьонов А.В.
(підпис)

Керівник роботи _____ зав. каф. АПОТ Чумаченко С.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 64 с., 3 таблиці, 4 лістинги, 7 рис., 22 джерел.

СЕКВЕНСОР, МЕТРИКА ПОДІБНОСТІ/ВІДМІННОСТІ, ВЕКТОРНІ СТРУКТУРИ ДАНИХ, АНАЛІЗ ВЕЛИКИХ ДАНИХ, КІБЕРФІЗИЧНИЙ КОМП’ЮТИНГ, МЕТОД ДІАГНОСТИКИ ОБЧИСЛЕНЬ, ВІДСТАНЬ ЛЕВЕНШТАЙНА.

Пропонуються комп’ютингові рішення для аналізу даних і діагностування технічного стану цифрових пристроїв на основі метрики подібності/відмінності. Аналізується стан питання. Пропонуються методи і алгоритми з їх кодовими реалізаціями для аналізу процесів і явищ, на основі векторних або табличних структур даних: 1) Інтерфейс введення даних. 2) Обчислення подібності/відмінності об'єктів. 3) Similarity-Difference method for Diagnosis Computing.

ABSTRACT

The explanatory note contains: 64 pages, 3 tables, 4 listing, 7 figures, 22 sources according to the list of links.

SEQUENCER, SIMILARITY-DIFFERENCE METRICS, VECTOR DATA STRUCTURES, BIG DATA ANALYTICS, CYBER PHYSICAL COMPUTING, METHOD FOR DIAGNOSIS COMPUTING, LEVENSTEIN DISTANCE.

We offer computer solutions for data analysis and diagnosing the technical condition of digital devices based on similarity / difference metrics. The state of the issue is analyzed. Methods and algorithms with their code implementations for the analysis of processes and the phenomena, on the basis of vector or tabular data structures are offered: 1) Data entry interface. 2) Calculation of similarities / differences of objects. 3) Similarity-Difference method for Diagnosis Computing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASIC – спеціалізований обчислювач;

ML – машинне навчання;

AI-engineering – ШІ-інженерія (інженерія штучного інтелекту);

LS-метрика – L-відповідність, подібність S;

GUI – graphical user interface (графічний інтерфейс користувача);

IDE – Integrated Development Environment (інтегроване середовище розробки);

IoT – Internet of Things, (Інтернет речей);

SoC – System of Chip (Система на Кристалі).

ВСТУП

У роботі розглядаються питання, пов'язані з розробкою комп’ютингових рішень для аналізу даних і діагностування технічного стану цифрових пристроїв на основі метрики подібності/відмінності.

Мета дослідження – суттєве зменшення обчислювальної складності алгоритмів комп'ютингу розпізнавання кіберфізичних об'єктів та діагностування шляхом розробки моделей, методів, алгоритмів, архітектур з використанням метрики подібності-відмінності.

Задачі дослідження орієнтовані на удосконалення та створення моделей, методів і архітектур комп'ютингу розпізнавання/діагностування кіберфізичних об'єктів. Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати сучасні технологічні тенденції;
- розширити моделі, методи, архітектури комп'ютингу розпізнавання/діагностування кіберфізичних об'єктів.

Розробляються методи і алгоритми та їх програмні реалізації для аналізу кіберфізичних об'єктів на основі векторних або табличних структур даних, а саме:

1) Інтерфейс введення даних.

2) Обчислення подібності/відмінності об'єктів.

3) Similarity-Difference method for Diagnosis Computing.

Спільність теорії аналізу моделей на основі метрики подібності/відмінності об'єднує всі наведені підходи.

Об'єкт дослідження – кіберсоціальні/кіберфізичні процеси і явища, автоматично керовані комп'ютинговими сервісами на основі використання метрики подібності/відмінності.

Предмет дослідження – структури даних, методи і алгоритми паралельного розподілу та об'єднання векторів по метриці подібності/відмінності для синтезу архітектури кіберфізичного комп'ютингу розпізнавання та діагностування технічного стану цифрових пристроїв.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Структури даних і алгоритми

Таблиця істинності є ідеальною формою структур даних для організації спеціалізованих обчислювачів (ASIC). Таблицю істинності можна умовно розглядати як цифровий автомат з одним станом, тригер – як елементарний автомат (пам'ять) з двома станами. Таблиця переходів за формою не відрізняється від таблиці істинності. Обидві реалізують сукупність умов (if - then) для пошуку або визначення рішення. По суті, таблиця переходів є скорочена або неповністю визначена таблиця істинності. Об'єднання умов if - then створює метрику істотних параметрів для форматування всіх рядків-умов у межах таблиці істинності. Можна автоматизувати процес синтезу таблиці істинності: якщо параметр вхідної умови відсутній в метриці, то він додається до вектору змінних. При цьому всі інші рядки таблиці істинності, де даного параметра не було, міститимуть на його координаті символ невизначеності або інваріантності. Така процедура ефективно буде працювати при створенні ML-структур даних на основі єдиної таблиці істинності. Умовою перевірки коректності таблиці є перетин всіх пар рядків між собою дорівнений порожній множині.

Приклад синтезу умов: if a, then 1; if b then 0; if ab then 2; if c then 3; if bcd then 4; if ae then 1. Таблиця істинності для згаданих умов має вигляд:

1	2	3	4	5	y
a					1
	b				0
a	b				2
		c			3
	b	c	d		4
a				e	1

Рішення по таблиці істинності визначається як об'єднання виходів для тих рядків, які мають непорожній результат перетину вхідної умови, записаного в форматі метрики параметрів.

Страви \ Ціна	1	2	3	4	5	6	7	8	9
Борщ	100	200	50	60	70	40	80	90	50
Суп харчо	100	150	90	90	99	98	97	86	55
Салат	33	43	44	55	44	33	44	33	55
Компот	12	12	22	32	22	11	23	33	33
Пиріг	23	32	23	32	33	44	33	34	32
Кисіль	17	34	23	22	33	44	55	22	22
Молоко	11	22	22	11	23	44	33	22	43

Будь-який як завгодно складний комп'ютинг можна подати таблицею істинності з примітивним автоматом управління, в один стан, що реалізує синхронізацію. З'являється метрика комп'ютингу, яка задається добутком простір-час n = ST, що має взаємно однозначну відповідність з імплементацією даних параметрів в аналогічну пару логіка-пам'ять n = LM. Замість однієї великої таблиці істинності створюється кінцеве число маленьких таблиць, обробка яких синхронізована в часі. Це стає можливим за рахунок введення вже нетривіального автомата управління (алгоритму), що координує роботу маленьких таблиць істинності або мікрооперацій. Алгоритм завжди є перетворювач -трансформатор простору таблиці істинності в часовий інтервал обчислення, необхідний для отримання результату: SAT. Чим складніше алгоритм, тим більше час його обробки. Простір і час створюють методологічний добуток n=ST, який має враховуватися експертами при розробці обчислювальних пристроїв. Зменшення простору спричиняє зниження швидкодії і навпаки. Наприклад, три точки на кривій (рис. 1.1) формують однаковий добуток n = ST = 8 = 4= 2.

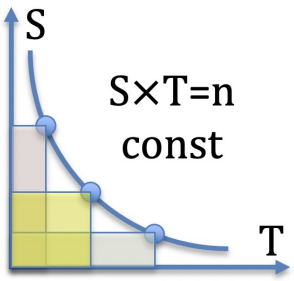

Рисунок 1.1 – Просторово-часова залежність комп'ютингу

Тут з'являється пам'ять, необхідна для зберігання проміжних результатів розрахунків і станів алгоритму. Алгоритм, пам'ять і час є метрикою складності обчислювального пристрою, або платою за зменшення простору таблиці істинності. За фактом можна сказати, що ускладнення структур даних не спрощує їх обробку. Чим вони простіше, тим швидше логіка їх обробки, тим вища продуктивність комп'ютингу. Таблиця істинності є кращою і простішою формою опису комбінаторних структур даних для людини і машини. За таким табличним шляхом успішно розвивалися технології машинного навчання і нейромереж. Потім спостерігалося ускладнення структур, підключення пам'яті і часу, за якими розвивався детермінований комп'ютинг. Комбінаційна цифрова схема, як сукупність безадресних логічних елементів, є аналог квантового комп'ютера. Його сутність полягає в тому, як паралельно обробити таблицю істинності для отримання результату при вирішенні комбінаторних завдань. Звичайно ж вона вирішена в рамках дворівневої комбінаційної схеми класичного комп'ютингу шляхом паралельного поширення аналогових сигналів від входів до виходів.

Таким чином, ускладнення структур даних від таблиці істинності у бік її розбиття призводить до появи складних алгоритмів, що використовують пам'ять, і тягне за собою істотне збільшення часу обчислювального процесу для отримання результату. Маючи на увазі, що пам'яті на сьогодні досить для реалізації таблиць істинності великої розмірності, необхідне повернення архітектури обчислювача до найпростіших структур даних - таблиць і

примітивним алгоритмам управління на основі логіки. До того ж слід мати на увазі, що досить зберігати не всю таблицю, а тільки вектор її вихідних станів, що ще більш спонукає спеціалізований комп'ютинг у бік векторних структур даних і примітивної логіки алгоритмів для їх обробки.

1.2 Gartner's Priorities 2021-2022

У прогнозах Gartner 2021 [21] мова йде про комбінаторику інновацій, які створюють нові властивості кіберфізичних систем (рис. 1.2).

Першим трендом є глибока орієнтованість на потреби людей, що включає розвиток інтернету етичної поведінки, узагальнення досвіду в різних сферах і впровадження в управління людиною, підвищення конфіденційності даних при обробці.

Рисунок 1.2 – Gartner's Priorities 2021 [21]

Другий тренд представлений розподіленими хмарними сервісами, операційними діями, інваріантними до геолокації суб'єктів, масштабованими мережами кібербезпеки процесів і явищ.

Третій тренд - стійкі поставки в умовах світової нестабільності формується інтелектуальним композитним і гнучким бізнесом, надійними засобами AI-engineering, гіперавтоматізацією всього, що можна автоматизувати в організації, має бути зроблено [21].

Інтегрально дослідження присвячені наступними технологіям і напрямками:

1) Відображення фізичних і соціальних процесів в кіберпросторі.

2) Аналітика великих даних для прийняття рішення людиною.

3) Застосування моделей і методів штучного інтелекту для прийняття рішення експертами.

4) Надання найпростіших хмарних сервісів громадянам з боку держави і компаній.

5) Моніторинг соціальних мереж, як реакції на актюаторні впливи владних структур.

6) Розпізнавання деструктивних контентів для подальшого блокування сайтів і медіаканалів.

7) Експертні системи прийняття рішення в кіберпросторі на основі аналізу великих даних.

8) Зелені технології соціального комп'ютингу для морального управління поведінкою громадян.

9) Застосування Cloud-Fog-Edge Computing для вичерпного моніторингу соціальних процесів.

10) Створення соціальних роботів для online обслуговування громадян в медицині, побуті, подорожах.

11) Квантові структури і алгоритми для ефективного вирішення проблем аналізу великих даних, вичерпного моніторингу і цифрового управління соціальними групами і громадянами.

Основні прогнози Gartner на 2022 рік та наступний період відображають три основні теми епохи пандемії: орієнтованість на людину, стійкість та здатність перевершити очікування. Їх можна використовувати як припущення для сценаріїв та стратегічних планів на наступні три-п'ять років (рис. 1.3) [22].

Рисунок 1.3 – Gartner Priorities 2022 [22]

Прогноз № 1: Синтетичні дані для більшої конфіденційності. До 2025 синтетичні дані скоротять збір особистих даних про клієнтів, що дозволить уникнути 70% санкцій за порушення конфіденційності.

У міру зростання популярності синтетичних даних, що генеруються за допомогою методів штучного інтелекту, вони можуть бути проксі для реальних даних, знижуючи або усуваючи ризики розкриття приватної інформації споживачів або конфіденційних даних. Оскільки це несправжні дані, вони знімають занепокоєння з боку нормативних вимог і фактично можуть надати більш точну інформацію, оскільки штучний інтелект може краще моделювати

непередбачувану поведінку клієнтів. Синтетичні дані можна використовувати для навчання та тестування моделей штучного інтелекту, щоб справлятися з незапланованими збоями, несподіваними подіями та плануванням сценаріїв, що зрештою створює більш стійку організацію.

Прогноз № 2: Споживачі борються із збором даних. До 2024 року 40% споживачів обманюватимуть показники відстеження поведінки, щоб навмисно знецінювати зібрані про них персональні дані, що ускладнює їхню монетизацію.

Споживачі дуже обізнані про кількість даних, які збирають організації, і, щоб не стати «продуктом», вони активно намагаються знецінити дані та маніпулювати ними. В результаті споживачі все частіше використовують такі тактики, як VPN, неправдиву інформацію або взагалі відмовляються від збору даних.

Прогноз № 3: Видобуток даних мозку. До 2027 року чверть компаній зі списку Fortune 20 будуть витіснені компаніями, які масштабують та впливають на підсвідому поведінку.

Нейромайнінг (застосування поведінкового інтелекту та пов'язаних технологій для аналізу, розуміння людської поведінки та впливу на нього в масштабі) дозволить організаціям глибше зрозуміти споживачів. Це покращить відносини з клієнтами та збільшить залучення співробітників. У міру розвитку нейробіології ця техніка ставатиме все більш складною та точною.

Прогноз № 4: Agile витіснить менеджерів. До 2024 року 30% корпоративних команд залишаться без начальника через гнучкий та гібридний характер роботи.

При гнучкому запровадження 30% і гібридної роботі 50% організації близько третини команд можуть працювати без традиційної ролі менеджера. Пандемія призвела до зростання потреби в організаційній стійкості, що спричинило більшу гнучкість всередині бізнесу. За визначенням, Agile вимагає атмосфери довіри, що не піддається традиційним ієрархіям. Збільшення гібридної робочої сили показує, що у значної частини менеджерів відсутній

набір навичок, необхідних для управління співробітниками у віддалених чи гібридних ситуаціях, і лише 47% працівників вважають, що керівник може призвести команду до успіху у майбутньому.

Прогноз № 5: Африка – наступний великий центр стартапів. До 2026 року 30% збільшення талантів розробників по всій Африці допоможе перетворити її на провідну у світі екосистему стартапів, яка конкурує з Азією щодо зростання венчурних фондів.

За останнє десятиліття венчурний капітал, що надходить до Африки, продовжував зростати як з боку місцевих, так і іноземних інвесторів. Процвітаючу технологічну арену Кенії називають «Силіконовою саванною» у Східній Африці, і тут знаходяться найпередовіші стартапи на континенті. У той же час зростання кількості каналів інформаційної освіти для розробників робить розробку програмного забезпечення більш доступним у широкому масштабі. Окрім зростання можливостей для стартапів, інші країни починають використовувати ресурси талантів, доступні в Африці.

Прогноз № 6: Модульний бізнес чи крах. До 2024 року 80% опитаних IT-директорів вкажуть модульну модернізацію бізнесу за рахунок можливості компонування як одну з п'яти основних причин підвищення ефективності бізнесу.

Руйнування – це нова норма. Навіть перед пандемією організації мали справу з торговими війнами, Brexit, зміною клімату та сукупністю інших внутрішніх та зовнішніх проблем. Нині 74% організацій впроваджують ті чи інші модульні або компонентні технології. Як правило, організації будуть зосереджені на гнучкості та адаптованості, а не на стабільності, щоб протистояти волатильності ринку.

Прогноз № 7: Війна з кібератаками. До 2024 року кібератака завдасть такої шкоди критично важливій інфраструктурі, що член G20 у відповідь оголосить фізичну атаку.

У той час як нещодавні великомасштабні атаки кібербезпеки на лікарні, уряди та енергетичні об'єкти вважаються злочинами, дедалі більше руйнівні

наслідки означають, що вони будуть вважатися актами війни. Посилення регулювання на національному рівні наголосить, що підприємства не можуть нести одноосібну відповідальність за обмеження розміру збитків. Кібератаки недержавних суб'єктів на критично важливу інфраструктуру з часом зменшуватимуться за частотою та інтенсивністю у міру того, як кіберзлочинці пристосовуються до реальності кінетичних репресій з боку військових та спецслужб.

Прогноз № 8: Розрив із поганими клієнтами. До 2025 року 75% компаній «розійдуться» з невідповідними споживачами, оскільки витрати на їх утримання затьмарюють витрати на залучення клієнтів.

Організації часто приймають погано пристосованих клієнтів у конвеєрі продажів і намагаються утримати їх, не враховуючи належним чином, наскільки ці клієнти обходяться організації дорого. Задоволення невідповідних клієнтів вимагає часу та грошей: наприклад, адаптувати пропозиції чи дизайнерські рішення, які їм підходять. У міру того, як керівники краще розуміють ці витрати, їм буде зручніше відпускати непродуктивних клієнтів з огляду на витік бренду та довгостроковий прибуток, не кажучи вже про емоційний тягар для співробітників.

Прогноз № 9: Гіперфіксація гіпертокенізації. До 2026 року невзаємозамінна гейміфікація токенів виведе підприємство до десятки найцінніших компаній.

Компанії швидко усвідомлюють потенціал невзаємозамінних токенів (NFT) для розвитку бізнес-моделей та залучення нових потоків доходів. Публічні блокчейни експоненційно розширюють можливості цифрового зв'язку цінності, представленої токенами, приносячи ще одну хвилю, здавалося б, безмежних можливостей у цифровому світі. Взаємозамінні та невзаємозамінні токени є рушійною силою цього нового явища, яке називається «гіпертокенізація».

Прогноз № 10: Більше інтернету за менші гроші. До 2027 року низькоорбітальні супутники розширять охоплення Інтернету ще на мільярд найбідніших людей світу, що допоможе 50% з них вирватися зі злиднів.

Створення баз стільникового зв'язку у сільській місцевості дорого та неефективно. Супутники на низькій орбіті пропонують більш дешеве та гнучке рішення. Вони можуть надати острівці підключення, тому мережі можуть розгортатися безпосередньо там, де є клієнти, а не лінійно по округу. Крім того, для задоволення збільшених потреб можуть бути запущені нові супутники. У міру того, як Інтернет стає все більш доступним, до нього додаються мільярди нових користувачів мережі, що підключилися до неї, які змінюють Інтернет з точки зору контенту і культури.

1.3 Висновки до розділу 1

Бібліографія актуальної теми Cyber Computing в частині аналізу даних представлена 3380 джерелами в бібліотеці IEEE Xplore, а також сотнями книг, виданих у видавництві Springer. Найбільш цікавими є монографії [1-6]. Проте, дослідження, які стосуються цифрового управління соціальними процесами і явищами в online режимі на основі метричного моніторингу, представлені досить слабо. До найбільш значущих публікацій слід віднести [7-17].

Далі представлені методи і алгоритми з їх кодовими реалізаціями для аналізу процесів і явищ, представлених векторними або табличними структурами даних: 1) Інтерфейс введення даних. 2) Обчислення подібності/відмінності об'єктів. 3) Similarity-Difference method for Diagnosis Computing.

2 ІНТЕРФЕЙС РЕДАГУВАННЯ ОБ'ЄКТА

2.1 Постановка завдання

Мета – істотне зменшення часу пошуку метричної відстані між процесами і явищами за рахунок розробки технології вибору оптимального маршруту редагування шляхом вставки порожнього символу, що дає можливість отримати уніфіковані структури даних для перетворення одного об'єкта в інший і привести метрику Левенштайна до простого підсумовування відмінності/подібності, але вже між компонентами двох векторів однакової розмірності.

Сутність – розробка процесора структурного трансформування одного об'єкта в інший шляхом вставки порожнього елемента за рахунок зсуву вектора, що дає можливість: завжди отримувати об'єкти однієї уніфікованої розмірності для обчислення відстані по Левенштайну шляхом підсумовування координат відмінності/подібності, а також оптимальний маршрут перетворення.

Завдання:

1) Визначення координат-букв, які формують кортеж подібності двох слів, шляхом вставки порожніх символів, що дозволяють ідентифікувати збіг літер на однакових номерах позицій в уніфікованої для обох слів системі координат.

2) Формування-запис чисто трансферної оцінки Левенштайна для випадку, коли критерій після уніфікації двох слів має відстань більше максимальної довжини одного з них.

3) Формування відносних оцінок подібності-відмінності, наведених до уніфікованого формату, а також до довжини початкового і кінцевого слова.

4) Синтез секвенсора операції зсуву вправо даних для вставки порожнього символу з метою формування уніфікованого формату пари слів, що має однакові літери за однойменними координатами.

5) Програмно-апаратна реалізація і тестування процесора структурного перетворення одного об'єкта в інший.

2.2 Сутність алгоритму

Обчислювальна складність запропонованої технології пошуку відстані за Левенштайном визначається двома операціями: зсуву вправо даних R в одному з регістрів і запису порожніх символів Z в місця, що звільнилися:

Z),

n – кількість збігів в побудованому кортежі подібності, які супроводжуються зрушенням даних. Якщо реалізувати дві згадані операції паралельно, що цілком здійсненно в апаратному виконанні, то оцінка обчислювальної складності матиме вигляд:

,

завдяки тому факту, що кожна з n операцій зсуву виконується в регістрі за 1 автоматний такт, так само як і запис пустого символу у осередки, що звільнилися, здійснюється в паралельному режимі також за один такт.

Сутність алгоритму:

1) Пошук першої пари однакових символів і визначення відстані між ними n = i-j для виконання зсуву вправо з метою отримання рівності символів по стовпцю.

2) Якщо n>0, то виконується зрушення вмісту другого регістра-слова вправо на n символів з подальшим занесенням порожніх символів в координати

.

Якщо n<0, то виконується зрушення вмісту першого регістра-слова вправо на n символів з подальшим занесенням порожніх символів в координати

.

Якщо n = 0, то ніяких дій робити не слід.

3) Підрахунок відстані в метриці Левенштайна шляхом вирахування з уніфікованої довжини пари слів кількості стовпців подібності

У загальну довжину входять стовпці, де хоча б одна з двох координат має значущий символ :.

Довжина L-відповідності є кількість упорядкованих пар або стовпців координат, отриманих в результаті n-зрушень і вставок порожніх символів, необхідних для пошуку відстані за Левенштайном

$$cardN = cardS + cardD.$$

Подібність S і відмінність D в структурах даних по L-відповідності формує повну множину подій N=SD, яка дорівнює числу координат-стовпців, що відповідають умові: .

Тому норми по Левенштайну визначаються відношенням потужності подібності/відмінності до довжини L-відповідності:

=; =).

2.3 Реалізація алгоритму та тестування програми

Алгоритм реалізовано на мові C++, код наведено у лістингу 2.1. Тестування програми виконано на 20 різних парах слів (таблиця 2.1).

Таблиця 2.1 – Тестування програми

Рядок 1	**Рядок 2**	**Схожість (%)**	**Різниця**
Loans and Accounts	Loans Accounts	78	4
loans and accounts	loan and account	89	2
loans and accounts	accounts and loans	44	10
fishing, "camping"; and 'forest	fishing camping and forest	84	5
Loan Account and Dealing	LOAN ACCOUNTS DEALINGS	17	20
LoanAccountDealing	Load, Account, Dealing	77	5
abcdef	azced	50	3
kitten	sitten	83	1
INTENTION	EXECUTION	44	5
live	leave	60	2
lives	leave	40	3
Fun	Funny	60	2
Bored	Boring	50	3
Tired	Tiring	50	3

bare	bear	50	2
cell	sell	75	1
fairy	ferry	60	2
flour	flower	67	2
hear	here	50	2

Лістинг 2.1 – Код алгоритму на мові програмування С ++

```
double levenshteinDistance( const std::string& str1, const std::string& str2 ) {
  const int len1 = str1.size();
  const int len2 = str2.size();
  std::vector< std::vector< int > > dp( len1 + 1, std::vector< int >( len2 + 1 ) );

  for( int i = 1; i <= len1; ++i )
    dp[ i ][ 0 ] = i;

  for( int i = 1; i <= len2; ++i )
    dp[ 0 ][ i ] = i;

  for( int i = 1; i <= len1; ++i ) {
    for( int j = 1; j <= len2; ++j ) {
      if( str1[ i - 1 ] == str2[ j - 1 ] )
        dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ];
      else
        dp[ i ][ j ] = 1 + std::min( { dp[ i - 1 ][ j - 1 ], dp[ i - 1 ][ j ], dp[ i ][ j -
1 ] } );
    }
  }

   return dp[ len1 ][ len2 ] / static_cast< double >( std::max( len1, len2 ) ) *
100;
}
```

2.4 Висновки до розділу 2

Запропоновано алгоритм знаходження відмінностей між двома послідовностями символів. Виконана програмна реалізація алгоритму, яка протестована на 20 різних парах слів.

3 ПРОГРАМНО-АПАРАТНИЙ МОДУЛЬ ОБЧИСЛЕННЯ ПОДІБНОСТІ/ВІДМІННОСТІ ОБ'ЄКТІВ

3.1 Постановка завдання

Мета – визначення якісної і кількісної метричної взаємодії між процесами і явищами шляхом використання паралельних регістрових логічних операцій.

Завдання:

1) Синтез логічної структури для обчислення метричної взаємодії між процесами і явищами, заданими у двійковому коді.

2) Кодування структур даних і алгоритму на одній з мов опису апаратури.

3) Аналіз і тестування отриманої структури секвенсора на представницької вибірці тестових вхідних векторів.

3.2 Логічна схема обчислення оцінок подібності об’єктів

Скористаємось теоретичними положеннями, викладеними у роботах [18, 19].

На відміну від абсолютних оцінок відмінності і подібності, при обчисленні взаємодії між множинами переваги завдають нормовані оцінки подібності-відмінності, що є більш інформативними. Вони наведені до

знаменника у вигляді суми істотних координат двох векторів. При цьому виключають тільки нульові по однойменним координатам значення .

Для отримання нормованих оцінок знаменник обчислюється шляхом підсумовування всіх одиничних координат після виконання операції диз'юнкції

N=.

При цьому чисельником може виступати відстань за Хемінгом, що визначає абсолютну оцінку відмінності між двійковими векторами. Проте при подальших розрахунках ураховується модифікація, що пов'язана з розподілом подібності або відмінності на кількість одиничних координат, які можуть бути здобуті після логічного додавання двійкових векторів . Таке доповнення додає значущості нормованій оцінці відмінності/подібності за рахунок зменшення знаменника.

Таким чином, мають місце дві нормовані оцінки:

=

Такий підхід розрахунку відрізняється від метрик за Хемінгом та Левенштайном шляхом видаленні виключно нульових однойменних координат зі структури даних для оцінювання подібності/відмінності.

Наприклад, дано два вектора, які описують об'єкти:

A	1	0	1	1	0	0	0	1	0	1
B	0	1	1	1	0	0	0	0	1	1

Вектори можна трансформувати до наступної мінімальної структури за рахунок вилучення нульових (незначущих) стовпців та збереження тільки значущих стовпців, яки містять хоча б одну одиницю:

A	1	0	1	1	1	0	1
B	0	1	1	1	0	1	1

Далі до векторів застосовується процедура обчислення подібності-відмінності за наведеними вище формулами:

.

Природно, що .

Можна підрахувати похідну між нормами відмінності-подібності

.

Звідси видно, яка оцінка більше – подібність чи відмінність. Проте в даній метриці краще уникнути появи негативних чисел, які потребують додаткової інтерпретації.

Отже, процедура підрахунку наведених оцінок подібності/відмінності зводиться до виконання трьох векторних паралельних операцій () з подальшим підрахунком одиниць в отриманих результуючих векторах.

Цифрова логічна схема підрахунку подібності-відмінності представлена на рис. 3.1.

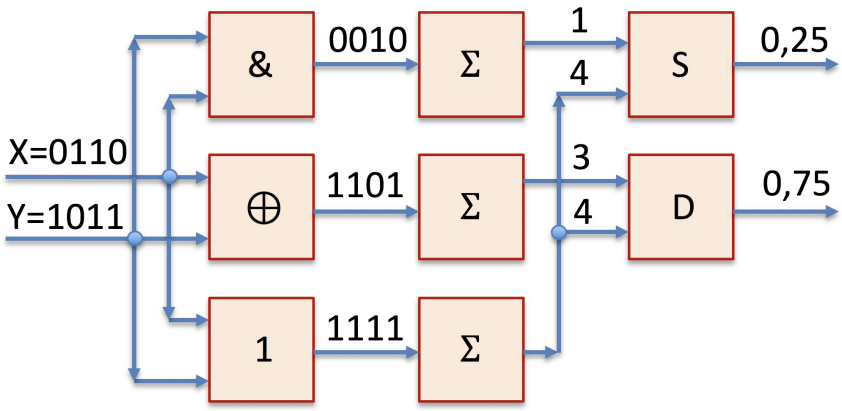

Рисунок 3.1 – Логічна схема обчислення оцінок подібності об’єктів

Недоліком наведеної метрики визначення подібності є нерозрізнення відношення включення одного рядка до іншого, усунути який можна шляхом xor-порівняння вихідних векторів з результатами їх перетину:

Відношення включення виконується, якщо значення дробу в виразі дорівнює нулю:

Якщо на обох виходах одночасно з’являються нулі, то це свідчить про еквівалентність векторів .

На рис. 3.2 наведена модифікована логічна схема обчислення оцінок подібності векторів, що дозволяє визначати відношення включення.

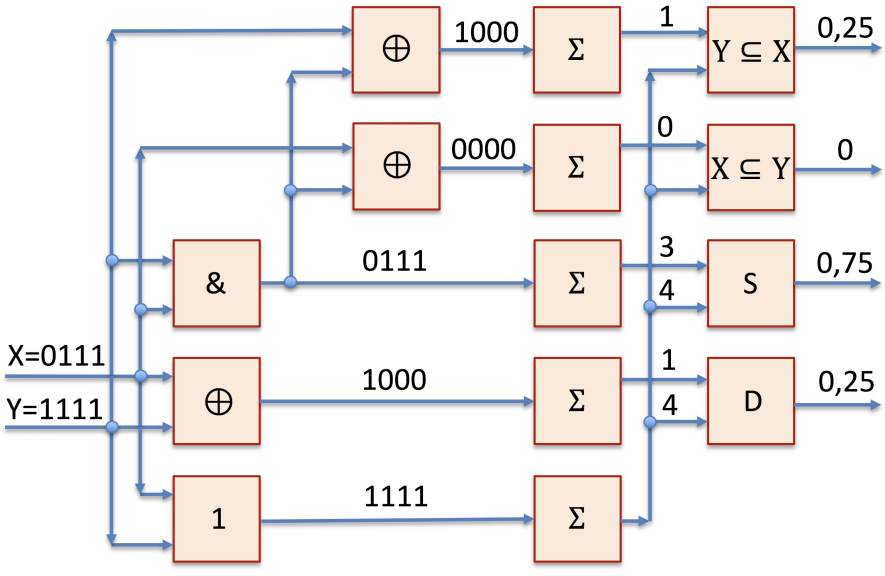

Рисунок 3.2 – Схема визначення норм подібності і відношення включення

0-сигналом

3.3 Кодування структур даних і алгоритму, аналіз і тестування отриманої структури секвенсора

Результати (рис. 3.3) представлені: 1) кодом (лістинг 3.1) і апаратною реалізацією цифрової структури, 2) тестуванням (таблиця 3.1) і висновками по швидкодії і витратам пам'яті.

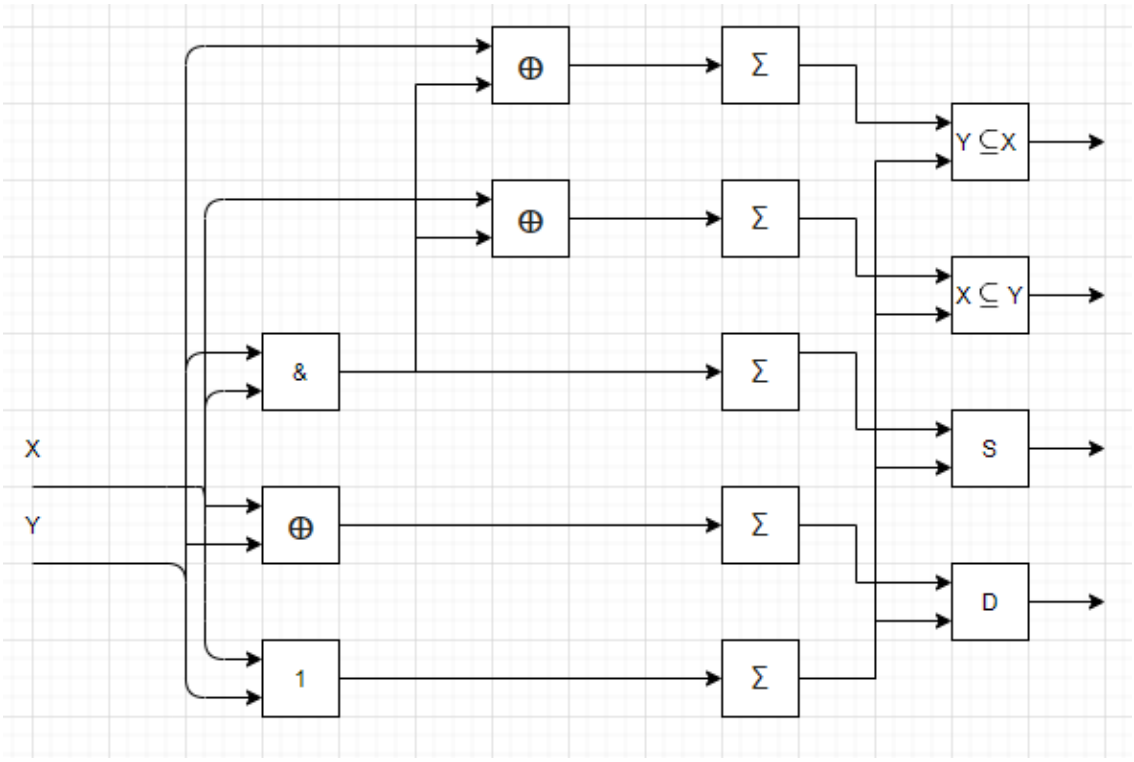

Рисунок 3.3 – Схема визначення подібності та відношення включення

Лістинг 3.1 – Код для обчислення подібності/відмінності об'єктів на мові C #

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
```

```
public class Program
{
   const Int32 ONE_CODE = 49;
   const Int32 ZERO_CODE = 48;

   public static void Main()
   {
      while(true)
      {
         Sequencer();
      }
   }
   private static void Sequencer()
   {
      String firstVector = Console.ReadLine();
      String secondVector = Console.ReadLine();
      if (ValidateOnIncorrectSymbol(firstVector, secondVector))
      {
         Console.WriteLine("Vector can\'t contain digits from decimal system");
         return;
      }

      var lists = CreateByteLists(firstVector, secondVector);

      IList<Byte> xVector = lists.Item1;
      IList<Byte> yVector = lists.Item2;

      Byte[] xAndYVector = AND(xVector, yVector);
      Byte[] xOrYVector = OR(xVector, yVector);
      Byte[] xXorYVector = XOR(xVector, yVector);

      Byte[] xAndYVectorXorX = XOR(xAndYVector, xVector);
      Byte[] xAndYVectorXorY = XOR(xAndYVector, yVector);

      Console.WriteLine($"Similarity - {GetRelationResult(xAndYVector,
xOrYVector)}");
      Console.WriteLine($"Difference - {GetRelationResult(xXorYVector,
xOrYVector)}");
      Console.WriteLine($"Inclusion X to Y -
{GetRelationResult(xAndYVectorXorX, xOrYVector)}");
      Console.WriteLine($"Inclusion Y to X -
{GetRelationResult(xAndYVectorXorY, xOrYVector)}");
   }
```

```
        private static Double GetRelationResult(IEnumerable<Byte> firstVector, IEnumerable<Byte> secondVector)
        {
            Int32 firstResultMetric = firstVector.Count(IsOne);
            Int32 secondResultMetric = secondVector.Count(IsOne);
            return (Double)firstResultMetric / (Double)secondResultMetric;
        }

        private static Boolean IsOne(Byte b) => b == 1;

            private static Byte[] AND(IList<Byte> firstVector, IList<Byte> secondVector) => MakeBytesOperation(firstVector, secondVector, (f, s) => (byte)(f & s));
        private static Byte[] OR(IList<Byte> firstVector, IList<Byte> secondVector) => MakeBytesOperation(firstVector, secondVector, (f, s) => (byte)(f | s));
            private static Byte[] XOR(IList<Byte> firstVector, IList<Byte> secondVector) => MakeBytesOperation(firstVector, secondVector, (f, s) => (byte)(f ^ s));

            private static Tuple<List<byte>, List<byte>> CreateByteLists(String firstVectorString, String secondVectorString)
        {
            List<byte> firstVector = GetByteArray(firstVectorString).ToList();
            List<byte> secondVector = GetByteArray(secondVectorString).ToList();

            Int32 lengthDifferent = firstVector.Count - secondVector.Count;

            if (lengthDifferent != 0)
            {
                    List<byte> lowerVector = lengthDifferent > 0 ? secondVector : firstVector;
                        lowerVector.AddRange(CreateByteArray(lengthDifferent, Convert.ToByte(ZERO_CODE)));
            }

            for (Int32 i = firstVector.Count - 1; i >= 0; --i)
            {
               if ((firstVector[i] | secondVector[i]) == 0)
               {
                  firstVector.RemoveAt(i);
                  secondVector.RemoveAt(i);
               }
            }
```

```
            return new Tuple<List<byte>, List<byte>>(firstVector, secondVector);
        }

        private static Boolean ValidateOnIncorrectSymbol(String firstVectorString,
String secondVectorString)
        {
            String[] vectors = { firstVectorString, secondVectorString };
            const String regExp = "^[0-1]*$";
            Regex regex = new Regex(regExp);

            return vectors.Any(v => !regex.IsMatch(v));
        }

        private static Byte[] CreateByteArray(Int32 lengthOfDifferent, Byte value)
        {
            Byte[] array = new Byte[lengthOfDifferent];

            for (Int32 i = 0; i < lengthOfDifferent; ++i)
            {
                array[i] = value;
            }

            return array;
        }

            private static Byte[] MakeBytesOperation(IList<Byte> firstBytes,
IList<Byte> secondBytes, Func<Byte, Byte, Byte> func)
        {
                    return firstBytes.Select((firstElement, i) => func(firstElement,
secondBytes[i])).ToArray();
        }

        private static Byte[] GetByteArray(String str)
        {
            return str.Select(ch => (byte)Char.GetNumericValue(ch)).ToArray();
```

Таблиця 3.1 – Верифікація програми

X	Y	Similarity	Difference	Inclusion X to Y	Inclusion Y to X
0111	1111	0.75	0.25	0	0.25
1011111	0111110	0.571	0.428	0.285	0.142
0111111	1011110	0.571	0.428	0.285	0.142
10001001	11000111	0.33	0.66	0.16	0.5

11100010	0001001	0.2	0.8	0.6	0.2
1111111	1111111	1	0	0	0

3.4 Інфраструктура подібності-відмінності для діагностичного комп’ютингу

Мета – істотне зменшення обчислювальної складності алгоритмів діагностування шляхом розробки ефективної інфраструктури на основі матричних даних для спрощеного пошуку дефектів з використанням метрики подібності-відмінності.

Завдання:

1) Розробка матричної моделі опису інфраструктури діагностування процесів чи явищ.

2) Розробка ефективних обчислювальних алгоритмів (векторного та кубітного) пошуку дефектів шляхом оцінювання подібності-відмінності.

3) Кодування алгоритмів та їх тестування на основі заданих матриць діагностування цифрових систем.

Для діагностування дефектів можна використовувати метод пошуку, заснований на векторному поданні множини несправностей, що перевіряються на тестовому наборі:

;

D= ;

D= .

Тут фігурують два рівняння, які визначають сукупність одиночних чи кратних дефектів у цифровій системі. При цьому в початковій стадії діагностування слід скористатися припущенням про існування кратного дефекту, що є найімовірнішою подією в процесі експлуатації виробу. Якщо такий алгоритм дає порожню множину дефектів, необхідно скористатися другим рівнянням для пошуку кратних несправностей. У процесі виконання діагностичного експерименту беруть участь дві групи векторів, які класифікуються шляхом їх приналежності до одиничної або нульової множини, що формується на основі {1,0}-реакції R цифрового пристрою на тестові дії.

Далі розглядається матричний метод діагностування несправних станів цифрового виробу, який використовує як діагностичну інформацію двійкову матрицю «тест-несправність» M=<T,F>, що має такий вигляд:

M=<T,F>	F1	F2	F3	F4	F5	F6	F7	F8	R
T1	1		1		1				1
T2		1		1		1			0
T3	1		1			1			0
T4							1		1
T5	1		1		1				0
T6		1						1	0
T7			1						0
T8		1		1		1		1	1
	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	0	1	
D=	0	0	0	0	0	0	1	0	

Далі розглядається кубітний метод пошуку дефектів шляхом теоретико-множинної різниці двох векторів, що відповідають одиничному та нульовому значенню станів виходів, як реакцій спостережуваних виходів на вхідний тест перевірки несправностей:

$$F = (\bigcup_{\forall R_i=1} Q_{ij}) \setminus (\bigcup_{\forall R_i=0} Q_{ij}) = (\bigvee_{\exists R_i=1} Q_{ij}) \wedge \overline{(\bigvee_{\exists R_i=0} Q_{ij})}.$$

Структури даних подані таблицею несправностей на декартовому добутку тестових наборів і множині ліній об'єкта діагностування, де кожен осередок є двома бітами: перший з них ідентифікує константну несправність нуля (10), а другий – константну несправність одиниці (01):

$$\begin{aligned}
&Q = \{F, T, L\},\\
&Q = Q_{ij}, i = \overline{1, m}; j = \overline{1, n};\\
&F = (F_1, F_2, ..., F_j, ..., F_n),\\
&F_j = \{10 \equiv 0; 01 \equiv 1; 11 = \{\equiv 0, \equiv 1\}; 00 = \varnothing\};\\
&T = (T_1, T_2, ..., T_i, ..., T_m);\\
&L = (L_1, L_2, ..., L_i, ..., L_n).
\end{aligned}$$

Суперпозиція несправностей (дві одиниці на одній лінії-комірці) дає можливість суттєво мінімізувати структури даних для зберігання інформації з метою подальшого пошуку дефектів під час виконання діагностичного експерименту в режимі online.

Для перевірки методу пошуку дефектів далі пропонується логічна схема, представлена на рис. 3.4, яка має 6 елементів and-not, 11 ліній, 5 входів та два виходи.

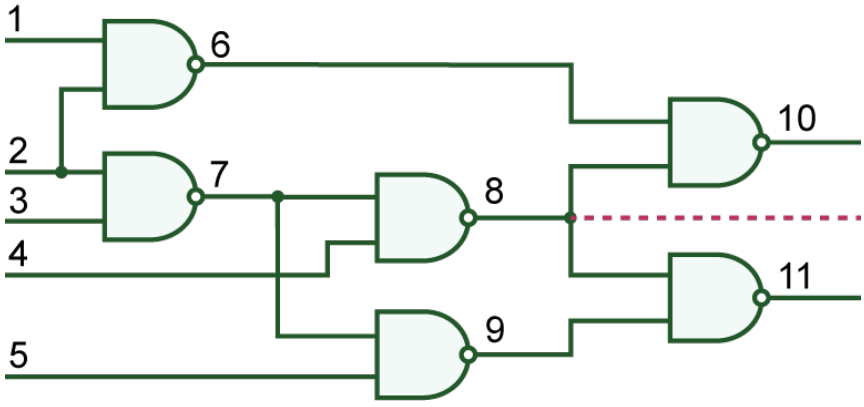

Рисунок 3.4 – ISCAS-схема для верифікації

Наступна таблиця ілюструє виконання діагностичного експерименту для об'єднання множини дефектів, які формують некоректні стани виходів на тестових наборах {T1-R10; T5-R11; T6-R10, R11; T8-R11}:

$Q = Q_{ij}$	1	2	3	4	5	6	7	8	9	10	11	R_{10}	R_{11}
T_1	01	10	01	00	10	00	10	10	00	10	01	1	0
T_2	10	00	10	00	01	10	00	00	10	01	10	0	0
T_3	00	01	01	00	00	01	10	01	01	10	10	0	0
T_4	10	00	01	00	10	00	01	00	10	01	01	0	0
T_5	00	10	00	01	00	01	00	10	00	10	10	0	1
T_6	01	10	00	00	10	00	00	01	10	01	10	1	1
T_7	01	00	00	10	00	00	01	00	10	01	01	0	0
T_8	00	10	10	01	01	10	00	00	00	01	10	0	1
Q_1	01	11	11	01	11	11	10	11	10	11	11	1	1
Q_0	11	01	11	10	11	11	11	01	11	11	11	0	0
F	00	10	00	01	00	00	00	10	00	00	00	1 / 0	1 / 0

Тут диз'юнкція рядків T1, T5, T6, T8 формує вектор Q1, який збирає всі можливі дефекти, що перевіряються на тестових наборах. Вектор Q0 за допомогою рядків T2, T3, T4, T7 поєднує всі неможливі дефекти, що не перевіряються на тестових наборах. Віднімання всіх неможливих з усіх можливих дефектів дає результат у вигляді трьох несправностей, закодованих як F2=10; F4 =01; F8 = 10.

Таким чином, паралельне виконання двох регістрових or-операцій на основі результатів проведеного діагностичного експерименту дозволило визначити три можливі несправності, кожна з яких може мати місце в логічній схемі: $F = \{2^0, 4^1, 8^0\}$.

Більш жорсткою є обмежувальна умова існування в логічній схемі одиночного константного дефекту, використання якого призводить до обчислення дефектів на основі наступного виразу:

$$F = (\bigcap_{\forall R_i=1} Q_{ij}) \setminus (\bigcup_{\forall R_i=0} Q_{ij}) = (\bigwedge_{\exists R_i=1} Q_{ij}) \wedge \overline{(\bigvee_{\exists R_i=0} Q_{ij})}.$$

Застосування даної формули суттєво уточнює результат діагностування та наводить його до вигляду: $F = \{2^0\}$ за рахунок суперечливості кодів дефектів по and-операції у стовпцях 4 і 8. Умова наявності в логічній схемі одиночної константної несправності спирається на наступне твердження.

Твердження. Якщо в стовпці таблиці несправностей існує координата 00 або 01, яка створює на виходах, що спостерігаються, некоректність R=1, пов'язану з несправністю 10 на інших координатах стовпця, то такий одиночний дефект (10) в логічній схемі неможливий.

Доведення. Нехай на n тестових наборах зафіксовано розбіжність на зовнішніх виходах еталонних та реальних значень сигналів. При цьому n-1 координата в стовпці, що розглядається, має значення 10 (01) і лише одна n-координата має значення 01 (10). Якщо припустити, що в логічній схемі є дефект 10, то на n-координаті також має бути дефект 10, який створює некоректний стан виходів. Але за умовами моделювання такого дефекту там немає. Отже, неможливо вважати, що у схемі є дефект 10. Це підтверджується також формальним результатом – порожнім перетином всіх координат стовпця, пов'язаних з некоректними станами виходів схеми:

$$F = (\bigwedge_{\exists R_i=1} Q_{ij}) = \begin{cases} 10 \wedge 10 \wedge \ldots \wedge 10 \wedge 01 = 00; \\ 10 \wedge 10 \wedge \ldots \wedge 10 \wedge 00 = 00. \end{cases}$$

Все сказане відноситься і до стану n-координати, що ідентифікується сигналом порожньої множини 00, взаємодія з яким також унеможливлює присутність у логічній схемі одиночної константної 0-несправності (код 10).

Код програми пошуку дефектів мовою програмування С++ наведено у Лістингу 3.2.

Лістинг 3.2 – Код програми пошуку дефектів

```
#include <iostream>
#include <vector>
#include <bitset>
#include <range/v3/all.hpp>

using namespace std::string_literals;

enum CheckableDefect
{
	_00, _01, _10, _11
};

enum Reaction
{
	_0, _1, _1_0
};

struct Row
{
	std::string m_name;
	std::vector< CheckableDefect > m_checkableDefects;
	std::vector< Reaction > m_reactions;

	bool has_reaction() const
	{
	return
	ranges::any_of(
	m_reactions
	, []( auto _reaction ) { return _reaction; }
	)
	;
	}

	void print() const
	{
	std::cout
```

```
        << m_name
        << ": "
        << (
        m_checkableDefects
        | ranges::views::transform(
        []( auto && _i ){
        return ranges::views::single( std::bitset< 2 >( _i ).to_string() );
        }
        )
        | ranges::views::join( ", " )
        | ranges::actions::join
        )
        << " | "
        << (
        m_reactions
        | ranges::views::transform(
        []( auto && _i ){
        return ranges::views::single( _i == _1_0 ? "1/0" : " " +
std::to_string( int( _i ) ) );
        }
        )
        | ranges::views::join( ", " )
        | ranges::actions::join
        )
        << std::endl;
        ;

        }

        Row subtract( std::string _name, Row _other ) const
        {
        auto defects =
        ranges::views::zip_with(
        []( auto && _this, auto && _oth )
        {
        return CheckableDefect(
        ( std::bitset< 2 >( _this ) & std::bitset< 2 >( _oth ).flip() ).to_ulong() );
        }
        m_checkableDefects
        , _other.m_checkableDefects
        )
        | ranges::to< std::vector >
        ;
```

```
        auto reactions =
        ranges::views::zip_with(
        []( auto && _this, auto && )
        {
        return (_this == _1)? _1_0: _0;
        }
        m_reactions
        , _other.m_reactions
        )
        | ranges::to< std::vector >
        ;

        return Row{ std::move( _name ), defects, reactions };
        }

};


Row fold_rows( std::string _name, ranges::any_view< Row > _rows, bool _or =
true )
{
        if ( ! ranges::distance( _rows ) )
                return Row{ _name, { _00, _00, _00, _00, _00, _00, _00, _00, _00, _00,
_00 }, { _0, _0 } };



        return ranges::accumulate(
        _rows
        , *_rows.begin()
        , [&]( auto && _row1, auto && _row2 )
        {
        auto defects =
        ranges::views::zip_with(
        [&]( auto && _lhs, auto && _rhs )
        {
        return _or? CheckableDefect( _lhs | _rhs ) : CheckableDefect( _lhs & _rhs );
        }
        , _row1.m_checkableDefects
        , _row2.m_checkableDefects
        )
        | ranges::to< std::vector >
        ;
        auto reactions =
```

```
        ranges::views::zip_with(
        []( auto && _lhs, auto && _rhs )
        {
        return Reaction( _lhs || _rhs );
        }
        , _row1.m_reactions
        , _row2.m_reactions
        )
        | ranges::to< std::vector >
        ;
        return Row{ _name, defects, reactions };
        }
        );
}

void print_delim( int _size )
{
        std::cout << ( ranges::views::repeat_n( "-"s, _size ) | ranges::actions::join ) <<
std::endl;
}

using Tests = std::vector< Row >;

int main()
{
        auto tests = Tests {
 { "T1", { _01, _10, _01, _00, _10, _00, _10, _10, _00, _10, _01 }, { _1, _0 } }
 , { "T2", { _10, _00, _10, _00, _01, _10, _00, _00, _10, _01, _10 }, { _0, _0 } }
 , { "T3", { _00, _01, _01, _00, _00, _01, _10, _01, _01, _10, _10 }, { _0, _0 } }
 , { "T4", { _10, _00, _01, _00, _10, _00, _01, _00, _10, _01, _01 }, { _0, _0 } }
 , { "T5", { _00, _10, _00, _01, _00, _01, _00, _10, _00, _10, _10 }, { _0, _1 } }
 , { "T6", { _01, _10, _00, _00, _10, _00, _00, _01, _10, _01, _10 }, { _1, _1 } }
 , { "T7", { _01, _00, _00, _10, _00, _00, _01, _00, _10, _01, _01 }, { _0, _0 } }
 , { "T8", { _00, _10, _10, _01, _01, _10, _00, _00, _00, _01, _10 }, { _0, _1 } }
        };

        auto with_reaction =
        tests
        | ranges::views::filter( []( auto && _row ) { return _row.has_reaction(); } )
        ;

        auto without_reaction =
        tests
        | ranges::views::filter( []( auto && _row ) { return ! _row.has_reaction(); } )
```

```
        ;

        auto Q1 = fold_rows("Q1", with_reaction, false);
        auto Q0 = fold_rows("Q0", without_reaction);

        std::cout << " L: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 | R10, R11" << std::endl;

        print_delim(57);

        for ( auto const & test: tests )
        test.print();

        print_delim(57);

        Q1.print();
        Q0.print();

        auto F = Q1.subtract("F", Q0);

        print_delim(57);

        F.print();

        auto results =
        F.m_checkableDefects
        | ranges::views::enumerate
        | ranges::views::filter( []( auto && _pair ){ return _pair.second != _00; } )
        | ranges::views::transform(
        []( auto && _pair )
        {
        return
        "Constant "s + ( _pair.second == _01 ? "1" : "0" )
        + "on line" + std::to_string(_pair.first + 1);
        }
        )
        | ranges::views::common
        ;

        std::cout << std::endl;
        for ( auto const & result: results )
        std::cout << result << std::endl;
}
```

Тестування алгоритму при пошуку одиночних та кратних дефектів.

Тест 1: пошук дефектів за формулою

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | R10, R11
-------------------------------------------------------------
T1: 01, 10, 01, 00, 10, 00, 10, 10, 00, 10, 01 |   1,   0
T2: 10, 00, 10, 00, 01, 10, 00, 00, 10, 01, 10 |   0,   0
T3: 00, 01, 01, 00, 00, 01, 10, 01, 01, 10, 10 |   0,   0
T4: 10, 00, 01, 00, 10, 00, 01, 00, 10, 01, 01 |   0,   0
T5: 00, 10, 00, 01, 00, 01, 00, 10, 00, 10, 10 |   0,   1
T6: 01, 10, 00, 00, 10, 00, 00, 01, 10, 01, 10 |   1,   1
T7: 01, 00, 00, 10, 00, 00, 01, 00, 10, 01, 01 |   0,   0
T8: 00, 10, 10, 01, 01, 10, 00, 00, 00, 01, 10 |   0,   1
-------------------------------------------------------------
Q1: 01, 10, 11, 01, 11, 11, 10, 11, 10, 11, 11 |   1,   1
Q0: 11, 01, 11, 10, 11, 11, 11, 01, 11, 11, 11 |   0,   0
-------------------------------------------------------------
F : 00, 10, 00, 01, 00, 00, 00, 10, 00, 00, 00 | 1/0, 1/0

Constant 0 on line 2
Constant 1 on line 4
Constant 0 on line 8
```

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | R10, R11
-------------------------------------------------------------
T1: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   0,   0
T2: 00, 01, 00, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T3: 01, 00, 01, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T4: 01, 00, 00, 00, 00, 10, 00, 10, 10, 01, 01 |   0,   0
T5: 00, 01, 00, 01, 01, 10, 00, 10, 10, 01, 01 |   1,   1
T6: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   0,   0
T7: 10, 10, 00, 01, 01, 01, 00, 10, 10, 10, 01 |   0,   1
T8: 10, 10, 00, 00, 00, 00, 00, 10, 10, 10, 01 |   0,   0
-------------------------------------------------------------
Q1: 11, 11, 01, 01, 11, 11, 10, 10, 11, 11, 11 |   1,   1
Q0: 11, 10, 00, 10, 00, 10, 10, 11, 10, 11, 11 |   0,   0
-------------------------------------------------------------
F : 00, 01, 01, 01, 11, 01, 00, 00, 01, 00, 00 | 1/0, 1/0

Constant 1 on line 2
Constant 1 on line 3
Constant 1 on line 4
Constant 0 on line 5
Constant 1 on line 6
Constant 1 on line 9
```

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | R10, R1
-------------------------------------------------------------
T1: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   1,
T2: 00, 01, 00, 01, 10, 10, 10, 10, 01, 01, 10 |   1,
T3: 01, 00, 01, 01, 10, 10, 10, 10, 01, 01, 10 |   1,
T4: 01, 00, 00, 00, 00, 10, 00, 10, 10, 01, 01 |   0,
T5: 00, 01, 00, 01, 01, 10, 00, 10, 10, 01, 01 |   0,
T6: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   1,
T7: 10, 10, 00, 01, 01, 01, 00, 10, 10, 10, 01 |   0,
T8: 10, 10, 00, 00, 00, 00, 00, 10, 10, 10, 01 |   0,
-------------------------------------------------------------
Q1: 01, 01, 01, 11, 10, 10, 10, 11, 01, 11, 10 |   1,
Q0: 11, 11, 00, 01, 01, 11, 00, 10, 10, 11, 01 |   0,
-------------------------------------------------------------
F : 00, 00, 01, 10, 10, 00, 10, 01, 01, 00, 10 | 1/0,

Constant 1 on line 3
Constant 0 on line 4
Constant 0 on line 5
Constant 0 on line 7
Constant 1 on line 8
Constant 1 on line 9
Constant 0 on line 11
```

Тест 2: пошук дефектів за формулою

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | R10, R11
-------------------------------------------------------------
T1: 01, 10, 01, 00, 10, 00, 10, 10, 00, 10, 01 |   1,   0
T2: 10, 00, 10, 00, 01, 10, 00, 00, 10, 01, 10 |   0,   0
T3: 00, 01, 01, 00, 00, 01, 10, 01, 01, 10, 10 |   0,   0
T4: 10, 00, 01, 00, 10, 00, 01, 00, 10, 01, 01 |   0,   0
T5: 00, 10, 00, 01, 00, 01, 00, 10, 00, 10, 10 |   0,   1
T6: 01, 10, 00, 00, 10, 00, 00, 01, 10, 01, 10 |   1,   1
T7: 01, 00, 00, 10, 00, 00, 01, 00, 10, 01, 01 |   0,   0
T8: 00, 10, 10, 01, 01, 10, 00, 00, 00, 01, 10 |   0,   1
-------------------------------------------------------------
Q1: 00, 10, 00, 00, 00, 00, 00, 00, 00, 00, 00 |   1,   1
Q0: 11, 01, 11, 10, 11, 11, 11, 01, 11, 11, 11 |   0,   0
-------------------------------------------------------------
F : 00, 10, 00, 00, 00, 00, 00, 00, 00, 00, 00 | 1/0, 1/0

Constant 0 on line 2
```

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | R10, R11
-------------------------------------------------------------
T1: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   0,   0
T2: 00, 01, 00, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T3: 01, 00, 01, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T4: 01, 00, 00, 00, 00, 10, 00, 10, 10, 01, 01 |   0,   0
T5: 00, 01, 00, 01, 01, 10, 00, 10, 10, 01, 01 |   1,   1
T6: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   0,   0
T7: 10, 10, 00, 01, 01, 01, 00, 10, 10, 10, 01 |   0,   1
T8: 10, 10, 00, 00, 00, 00, 00, 10, 10, 10, 01 |   0,   0
-------------------------------------------------------------
Q1: 00, 00, 00, 01, 00, 00, 00, 10, 00, 00, 00 |   1,   1
Q0: 11, 10, 00, 10, 00, 10, 10, 11, 10, 11, 11 |   0,   0
-------------------------------------------------------------
F : 00, 00, 00, 01, 00, 00, 00, 00, 00, 00, 00 | 1/0, 1/0

Constant 1 on line 4
```

```
  L: 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11 | K10, K11
-------------------------------------------------------------
T1: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   1,   0
T2: 00, 01, 00, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T3: 01, 00, 01, 01, 10, 10, 10, 10, 01, 01, 10 |   1,   0
T4: 01, 00, 00, 00, 00, 10, 00, 10, 10, 01, 01 |   0,   0
T5: 00, 01, 00, 01, 01, 10, 00, 10, 10, 01, 01 |   0,   0
T6: 00, 00, 00, 10, 00, 00, 10, 01, 00, 10, 10 |   1,   0
T7: 10, 10, 00, 01, 01, 01, 00, 10, 10, 10, 01 |   0,   0
T8: 10, 10, 00, 00, 00, 00, 00, 10, 10, 10, 01 |   0,   0
-------------------------------------------------------------
Q1: 00, 00, 00, 00, 00, 00, 10, 00, 00, 00, 10 |   1,   0
Q0: 11, 11, 00, 01, 01, 11, 00, 10, 10, 11, 01 |   0,   0
-------------------------------------------------------------
F : 00, 00, 00, 00, 00, 00, 10, 00, 00, 00, 10 | 1/0,   0

Constant 0 on line 7
Constant 0 on line 11
```

Отже, реалізовано вихідний код програми, що відповідає векторному та/або кубітному алгоритму пошуку дефектів. Виконано тестування алгоритму на трьох матрицях діагностування при пошуку одиночних та кратних дефектів.

3.5 Висновки до розділу 3

Таким чином, розроблено програму для обчислення подібності/відмінності об'єктів. Виконано тестування і верифікація додатку на декількох двійкових векторах.

Апробоновано кубітний метод пошуку дефектів шляхом теоретико-множинної різниці двох векторів, що відповідають одиничному та нульовому значенню станів виходів, як реакцій спостережуваних виходів на вхідний тест перевірки несправностей.

Розроблено програму визначення дефектів з використанням матриць діагностування. Виконано тестування та верифікацію програми на трьох матрицях діагностування при пошуку одиночних та кратних дефектів.

ВИСНОВКИ

У роботі розглянуто питання, пов'язані з розробкою комп’ютингових рішень для аналізу даних і діагностування технічного стану цифрових пристроїв на основі метрики подібності/відмінності.

Задачі дослідження були орієнтовані на удосконалення та створення моделей, методів і архітектур комп'ютингу розпізнавання/діагностування кіберфізичних об'єктів, а саме:

проаналізовано сучасні технологічні тенденції;

розроблено методи і алгоритми та їх програмні реалізації для аналізу кіберфізичних об'єктів на основі векторних або табличних структур даних;

представлені методи і алгоритми з їх кодовими реалізаціями для аналізу процесів і явищ, на основі векторних або табличних структур даних: 1) Інтерфейс введення даних. 2) Обчислення подібності-відмінності об'єктів. 3) Similarity-Difference method for Diagnosis Computing;

виконано кодування структур даних і алгоритму, проведено аналіз і тестування отриманої структури секвенсора;

апробовано кубітний метод - реалізовано вихідний код програми, що відповідає векторному та/або кубітному алгоритму пошуку дефектів. Виконано тестування алгоритму на трьох матрицях діагностування при пошуку одиночних та кратних дефектів.

Публікація (тези доповіді): Семенов А.В., Чумаченко С.В. Діагностування технічного стану цифрового пристрою на основі метрики подібності-відмінності відмінності // XXVII Міжнародна інтернет-конференція «Innovation and Science». Велика Британія, Ліверпуль. 13-14 грудня 2021 р. 3 с.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Vladimir Hahanov. Cyber Physical Computing for IoT-driven Services New York: Springer 2018. 279 p.

2. Hai Zhuge. Cyber-Physical-Social Intelligence. Springer Singapore. 2020. 351 p. DOI 10.1007/978-981-13-7311-4

3. Ivor Goodson, Michele Knobel, Colin Lankshear, and J. Marshall Mangan. Cyber Spaces/Social Spaces. Culture Clash in Computerized Classrooms. Palgrave Macmillan US. 2002. 172 p. DOI 10.1057/9780230602151

4. Xun Liang. Social Computing with Artificial Intelligence. Springer Singapore. 2020. 289 p. DOI 10.1007/978-981-15-7760-4

5. A. Ant Ozok, Panayiotis Zaphiris. Online Communities and Social Computing. 5th International Conference, OCSC 2013, Las Vegas, NV, USA, Springer-Verlag Berlin Heidelberg. 452 p. DOI 10.1007/978-3-642-39371-6

6. Mariarosaria Taddeo, Ludovica Glorioso. Ethics and Policies for Cyber Operations NATO Cooperative Cyber Defence Centre of Excellence Initiative. Springer International Publishing. 2017. 252 p. DOI 10.1007/978-3-319-45300-2

7. V. Hahanov, S. Chumachenko, E. Litvinova and A. Hahanova, "Cyber-physical social monitoring and governance for the state structures," *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2018, pp. 123-129, doi: 10.1109/DESSERT.2018.8409112.

8. A. Sheth, P. Anantharam and C. Henson, "Physical-Cyber-Social Computing: An Early 21st Century Approach," in *IEEE Intelligent Systems*, vol. 28, no. 1, pp. 78-82, Jan.-Feb. 2013, doi: 10.1109/MIS.2013.20.

9. N. K. Giang, R. Lea and V. C. M. Leung, "Exogenous Coordination for Building Fog-Based Cyber Physical Social Computing and Networking Systems," in IEEE Access, vol. 6, pp. 31740-31749, 2018, doi: 10.1109/ACCESS.2018.2844336.

10. L. -L. Shi et al., "Human-Centric Cyber Social Computing Model for Hot-Event Detection and Propagation," in IEEE Transactions on Computational Social Systems, vol. 6, no. 5, pp. 1042-1050, Oct. 2019, doi: 10.1109/TCSS.2019.2913783.

11. W. Zhang, X. Chen, J. Sun and Q. Xi, "A Distributed DBSCAN Algorithm for Massive Data in Cyber Physical and Social Computing," 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), 2020, pp. 426-433, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00081.

12. K. H. Lee, A. Lippman, A. S. Pentland and P. Maes, "Just-in-Time Social Cloud: Computational Social Platform to Guide People's Just-in-Time Decisions," 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013, pp. 1259-1266, doi: 10.1109/GreenCom-iThings-CPSCom.2013.219.

13. V. Hahanov, S. Chumachenko, E. Litvinova, A. V. Hacimahmud, A. Hahanova and T. Soklakova, "Cyber Social Computing," 2018 IEEE East-West Design & Test Symposium (EWDTS), 2018, pp. 1-8, doi: 10.1109/EWDTS.2018.8524663.

14. A. Sheth, "Computing for human experience: Semantics-empowered sensors, services, and social computing on the ubiquitous Web," in IEEE Internet Computing, vol. 14, no. 1, pp. 88-91, Jan.-Feb. 2010, doi: 10.1109/MIC.2010.4.

15. S. K. Das, "Cyber-physical-social convergence in smart living: Challenges and opportunities," 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), 2016, pp. 1-1, doi: 10.1109/PERCOMW.2016.7457093.

16. X. Zhou, W. Liang, S. Huang and M. Fu, "Social Recommendation With Large-Scale Group Decision-Making for Cyber-Enabled Online Service," in IEEE Transactions on Computational Social Systems, vol. 6, no. 5, pp. 1073-1082, Oct. 2019, doi: 10.1109/TCSS.2019.2932288.

17. J. J. Zhang et al., "Cyber-Physical-Social Systems: The State of the Art and Perspectives," in IEEE Transactions on Computational Social Systems, vol. 5, no. 3, pp. 829-840, Sept. 2018, doi: 10.1109/TCSS.2018.2861224.

18. Чугай В. А. Моделі та методи розпізнавання кіберфізичних об'єктів на основі використання метрики подібності – відмінності : пояснювальна записка до атестаційної роботи здобувача вищої освіти на другому (магістерському) рівні, спеціальність 123–Комп'ютерна інженерія / В. А. Чугай ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків, 2020. – 60 с.

19. Чугай В.А., Чумаченко С.В. Моделі та методи розпізнавання кіберфізичних об'єктів на основі використання метрики подібності-відмінності // LVII Міжнародна інтернет-конференція «Наукові підсумки 2020 року». Вінниця. 17 грудня 2020 р. 3 с.

20. Семенов А.В., Чумаченко С.В. Діагностування технічного стану цифрового пристрою на основі метрики подібності-відмінності відмінності // XXVII Міжнародна інтернет-конференція «Innovation and Science». Велика Британія, Ліверпуль. 13-14 грудня 2021 р. 3 с.

21. Gartner Top Strategic Technology Trends for 2021 – Available at October 20 2020 [https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021/]

22. Gartner Top Strategic Technology Trends for 2022 – Available at December 12, 2021 [https://www.gartner.com/en/information-technology/insights/top-technology-trends].