ISSN 3083-6573 Print
ISSN 3083-6581 Online

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

IT&S

INFORMATION TECHNOLOGIES & SYSTEMS

2(2)
2025

TOPICS

- PROSPECTS DEVELOPMENT OF MACHINE LEARNING
- AUTHORSHIP IDENTIFICATION OF PROGRAM CODE
- CLASSIFICATION OF UKRAINIAN TEXTS BY STYLES

Editor-in-Chief: O.M. KHIMICH, (Kyiv, Ukraine)

Deputies Editor-in-Chief: O.Ye. VOLKOV, (Kyiv, Ukraine),
Ye.A. SAVCHENKO-SYNIAKOVA, (Kyiv, Ukraine)

Editorial Board: Ali Abbasov Mammad oglu (Baku, Azerbaijan); O.Yu. Azarkhov (Dnipro, Ukraine); I.Ye. Andrushchak (Lutsk, Ukraine); A.V. Anisimov (Kyiv, Ukraine); I. Vlahavas (Thessaloniki, Greece); W. Wojcik (Lublin, Poland); D.O. Volosheniuk (Kyiv, Ukraine); A.M. Hlibovets (Kyiv, Ukraine), O. Gorbunovs (Riga, Latvia); V.F. Gubarev (Kyiv, Ukraine); L.F. Gulyanytskyi (Kyiv, Ukraine); A.M. Gupal (Kyiv, Ukraine); V.V. Zosimov (Odesa, Ukraine); P.I. Kohut (Dnipro, Ukraine); L.M. Kozak (Kyiv, Ukraine); S.L. Kryvyy (Kyiv, Ukraine); V.I. Lytvynenko (Kherson, Ukraine); R. Martínez Béjar (Murcia, Spain); S.V. Pavlova, (Kyiv, Ukraine); O.V. Palagin (Kyiv, Ukraine); S.L. Pogorilyy (Kyiv, Ukraine); N. Prokofyeva (Riga, Latvia); B. Savchynskyy (Heidelberg, Germany); A.-B. M. Salem (Cairo, Egypt); K.M. Synytsa (Kyiv, Ukraine); V.S. Stepashko (Kyiv, Ukraine); I.V. Surovtsev (Kyiv, Ukraine); L.S. Fainzilberg (Kyiv, Ukraine); Chen Huafeng (Zhejiang, China); A.O. Chykriy (Kyiv, Ukraine); M.I. Schlesinger (Kyiv, Ukraine)

Responsible Executor: H.O. Pezentsali
Editors: N.A. Charchiyan, A.Yu. Vitchenko, O.O. Lysenko
Computer Group: O.V. Tupalskiy, N.S. Stashkova

Media ID R30-05899
Editorial address: Institute of Information Technologies and Systems
of the National Academy of Sciences of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187
phone: +380 (44) 526-00-09, e-mail: its.journal.ua@gmail.com,
https://nasu-periodicals.org.ua/index.php/its

Головний редактор: О.М. ХІМІЧ (Київ, Україна)

Заступники головного редактора: О.Є. ВОЛКОВ (Київ, Україна),
Є.А. САВЧЕНКО-СИНЯКОВА (Київ, Україна)

Редакційна колегія: Алі Аббасов Маммед огли (Баку, Азербайджан); О.Ю. Азархов (Дніпро, Україна); І.Є. Андрущак (Луцьк, Україна); А.В. Анісімов (Київ, Україна); І. Влахавас (Салоніки, Греція); В. Войчик (Люблін, Польща); Д.О. Волошенюк (Київ, Україна); А.М. Глібовець (Київ, Україна); О. Горбуновс (Рига, Латвія); В.Ф. Губарєв (Київ, Україна); Л.Ф. Гуляницький (Київ, Україна); А.М. Гупал (Київ, Україна); В.В. Зосімов (Одеса, Україна); П.І. Когут (Дніпро, Україна); Л.М. Козак (Київ, Україна); С.Л. Кривий (Київ, Україна); В.І. Литвиненко (Херсон, Україна); Р. Мартінес Бежар (Мурсія, Іспанія); С.В. Павлова (Київ, Україна); О.В. Палагін (Київ, Україна); С.Л. Погорілий (Київ, Україна); Н. Прокоф'єва (Рига, Латвія); Б. Савчинський (Гейдельберг, Німеччина); А.-Б. М. Салем (Каїр, Єгипет); К.М. Синиця (Київ, Україна); В.С. Степашко (Київ, Україна); І.В. Суровцев (Київ, Україна); Л.С. Файнзільберг (Київ, Україна); Чень Хуафен (Чжецзян, Китай); А.О. Чикрій (Київ, Україна); М.І. Шлезінгер (Київ, Україна)

Відповідальний виконавець: Г.О. Пезенцалі
Редактори: Н.А. Чарчіян, А.Ю. Вітченко, О.О. Лисенко
Комп'ютерна група: О.В. Тупальський, Н.С. Сташкова

Ідентифікатор медіа: R30-05899
Адреса: Інститут інформаційних технологій та систем НАН України, м. Київ,
Просп. Акад. Глушкова, 40, 03187
Телефон: 526-00-09, e-mail: its.journal.ua@gmail.com,
Сайт: https://nasu-periodicals.org.ua/index.php/its/

Підп. до друку 17.07.2025 р. Формат 70 × 108/16. Гарн. Book Antiqua.
Ум. друк. арк. 8,75. Обл.-вид. арк. 8,88. Тираж 84 пр. Зам. № 7727

Видавець і виготовлювач ВД «Академперіодика» НАН України
01024, Київ, вул. Терещенківська, 4

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи серії ДК № 544 від 27.07.2001

NATIONAL ACADEMY OF SCIENCES OF UKRAINE
INSTITUTE OF INFORMATION TECHNOLOGIES AND SYSTEMS
OF THE NAS OF UKRAINE

INFORMATION TECHNOLOGIES & SYSTEMS

2 (2) 2025

ACADEMIC AND RESEARCH JOURNAL
FOUNDED IN JANYARY 2025
PUBLISHED 6 TIMES PER YEAR
KYIV

CONTENTS

© Institute of Information Technologies and Systems of the NAS of Ukraine, 2025
© Publisher PH «Akademperiodyka» of the NAS of Ukraine, 2025

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА СИСТЕМ
НАН УКРАЇНИ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ & СИСТЕМИ

2 (2) 2025

НАУКОВИЙ ЖУРНАЛ
ЗАСНОВАНО В СІЧНІ 2025
ВИХОДИТЬ 6 РАЗІВ НА РІК
КИЇВ

ЗМІСТ

© Інститут інформаційних технологій та систем НАН України, 2025
© Видавець ВД «Академперіодика» НАН України, 2025

THEORY OF INFORMATION TECHNOLOGIES AND SYSTEMS CONSTRUCTION

ТЕОРІЯ ПОБУДОВИ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА СИСТЕМ

https://doi.org/10.15407/intechsys.2025.02.003
UDC 004.8 + 004.032.26

О.А. УРСАТЬЄВ, канд. техн. наук, старш. наук. співроб., провідн. наук. співроб.,
Інститут інформаційних технологій та систем НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://org/0009-0009-8323-0525
aleksei@irtc.org.ua

О.Є. ВОЛКОВ, канд. техн. наук., старш. дослідник, директор,
Інститут інформаційних технологій та систем НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0000-0002-5418-6723
alexvolk@ukr.net

В.Г. ТКАЛЯ, аспірант,
Інститут інформаційних технологій та систем НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0009-0004-7870-3256
advv0207@gmail.com

АВТОМАТИЗОВАНЕ МАШИННЕ НАВЧАННЯ. СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ

Розглянуто автоматизоване машинне навчання як рішення на основі штучного інтелекту для потреби автоматизації наскрізного процесу застосування машинного навчання, тобто проєктування конвеєрів машинного навчання – послідовності кроків, які перетворюють необроблені дані на машинну модель, прийнятну для розгортання у практичному використанні. Присутність людини у цьому циклі має бути значно скорочена або її бажано зовсім виключити. Розглянуто напрям подальшого розвитку штучного інтелекту і автоматизованого машинного навчання та тенденції його розвитку.

Ключові слова: *автоматизоване машинне навчання, демократизація штучного інтелекту, керований даними штучний інтелект, глибоке посилене навчання, трансферне навчання.*

Cite: Урсатьєв О.А., Волков О.Є., Ткаля В.Г. Автоматизоване машинне навчання. Стан та перспективи розвитку. *Information Technologies and Systems*, Київ, 2025, Том 2 (2), 03—33. https://doi.org/ 10.15407/intechsys.2025.02.003

© Publisher PH «Akademperiodyka» of the NAS of Ukraine, 2025. The article is published under an open access license CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

Вступ

На сучасному етапі дані перетворюються на капітал, який безпосередньо бере участь у формуванні та керуванні різними сферами діяльності людини. Яскравим прикладом цього може бути альтернативна цінність даних [1], зосереджена в необмеженому повторному використанні їх. Абсолютна цінність даних може набагато перевищувати ту, яку вдається отримати під час первинного використання. Інноваційні компанії можуть отримати приховану цінність і, потенційно, також величезні переваги, тобто, цінність даних треба розглядати як можливості їхнього подальшого використання, а не лише нинішнього.

Статтю присвячено огляду інформаційних послуг з автоматизації наскрізного процесу застосування та підвищення ефективності методів машинного навчання — сучасні підходи, методи, загалом технології у цій сфері, що конкурують внаслідок розвитку інтелектуальних засобів оброблення інформації, а іноді навіть перевершують експертів з машинного навчання.

Її метою є ознайомити фахівців, що потребують навичок роботи з даними, із завданнями, пов'язаними з обробленням даних, досконалим математичним апаратом та інструментарієм світового рівня для отримання з них інформації та знань.

Машинне навчання як один з інструментів штучного інтелекту

Сучасні платформи оброблення даних розширюють свої можливості, зокрема *Data Science* — на потребу прогностичної або, інакше, предикативної (*Predictive Analytics*) і рекомендаційної (*Prescriptive Analytics*) аналітики [1]. *Data Science* — це міждисциплінарна область, у якій застосовують наукові методи, процеси, алгоритми для отримання знань (*knowledge*) із різноманітних даних та для розуміння їх (*insights*). Застосовуючи методи та теорії в контексті математики, комп'ютерних наук та інформатики із залученням машинного навчання, вона пропонує концепцію об'єднання їх, для аналізування даних та розуміння реальних явищ, що їх представляють. Ця концепція відображає всі аспекти роботи з даними: здатність збирати та обробляти їх, вміти розуміти, отримувати цінність, візуалізувати. Отже, завдання, вирішуване тими, хто займається *Data Science*, полягає в видобуванні знань із даних з метою розуміння того, що містять у собі ці дані, а самі дані не є для *Data Science* предметом цієї науки. Інтелектуальний аналіз даних та Великі Дані є розділами *Data Science*.

В області *Data Science* машинне навчання (*machine-learning*, *ML*), як один з інструментів дослідження даних, охоплює автоматизовані форми статистичних алгоритмів, самонавчальних на даних. *Data science* — це ключова дисципліна в розвитку штучного інтелекту (*AI*),

а *ML* — це основний інструмент *AI*. Поняття «штучний інтелект» охоплює багато різних технологій і ніколи не належав до конкретного типу. Це, радше, «соціотехнічна конструкція» [2] — термін, який застосовують для позначення можливостей машин, що вирішують складні завдання, які донедавна могли розв'язувати лише люди. Нинішній інтерес до *AI*, ажіотаж навколо нього, пов'язаний насамперед із машинним навчанням, що дає алгоритмам змогу навчатися самостійно, не будучи явно запрограмованими, та поглиблювати свій досвід. Машинне навчання в бізнесі спрямоване на створення та навчання моделей. *AI* використовує ці машинні моделі, щоб коригувати ситуацію за певних обставин. Він перебуває на іншому рівні агрегації, ніж *Data Science* та *ML* — на рівні застосунку. Створена прогнозна математична модель процесу і *ML*-моделі мають бути об'днані для інших можливостей спільної роботи, таких як інтерфейс користувача та керування робочим процесом для створення програми *AI*. Структури *AI* дають змогу розширити аналітичні можливості поза традиційну *Data Science* і *ML* (рис.1). Ці рамки охоплюють потужні, сфокусовані інструменти великої аналітики, такі як самонавчання та посилене навчання (в зарубіжній літературі — неконтрольоване навчання (*Unsupervised Learning, UL*) глибоке[1] (*Deep Learning, DL*) та навчання з підкріпленням (*Reinforcement Learning, RL*). Це особливий тип машинного навчання, в якому шари нейронних мереж імітують роботу нашого мозку, і де останнім часом було досягнуто значного прогресу. Технологія була застосована в області маркування зображень, транскрипції голосу, автоматичного перекладу, безпілотних автомобілів і показала дуже хороші результати в порівнянні зі звичайним *ML* [2].

Когнітивні обчислення — це ще один термін, який зазвичай використовується для позначення найскладніших типів технологій штучного інтелекту, які намагаються імітувати людське мислення. Але його точне значення неясне, і багато аналітиків і постачальників уникають цього терміну, оскільки він дуже тісно пов'язаний з *IBM* [2].

[1] Термін «глибоке навчання» виник у 1980-х роках. Цей підхід до створення *AI* передбачає використання моделі нейронних мереж, які сприймають інформацію, що надходить, шарами. Кожен із таких шарів нейронів, об'єднаних в єдину мережу, оцінює отриману інформацію за якоюсь однією ознакою. Отримані дані сумуються мережею видачі кінцевого результату оцінки. Лауреатами премії імені Алана Тьюринґа за 2018 рік стали «хрещені батьки» штучного інтелекту — чені Джеффрі Хінтон, Янн ЛеКун та Йошуа Бенжіо. Зі слів Асоціації обчислювальної техніки, усі троє розробили концептуальні основи та інженерні рішення, які зробили глибинні нейронні мережі наріжним компонентом в обчислювальній техніці, і які сприяли розвитку технологій штучного інтелекту. Саме глибоке машинне навчання з використанням нейромереж стало передовим напрямом у галузі створення *AI*. Премію Тьюринґа отримали розробники штучного інтелекту з *Google* та *Facebook*. Березень, 2019, ФОКУС — *https://focus.ua/technologies/424630-premiyu-tyuringa-poluchili-razrabotchiki-iskusstvennogo-intellekta-iz-google-i-facebook*

Рис. 1. Крива зрілості інноваційних технологій або цикл очікувань (*The Gartner hype cycle*[2])

Автоматизоване машинне навчання (Automated Machine Learning). Машинне навчання досягло значних успіхів, і дедалі більше дисциплін покладаються на нього. Однак ці успіхи визначаються насамперед внеском науковців у розробку теоретичних питань, наприклад [3], і значною мірою залежать від фахівців з *ML*, які обирають парадигми машинного навчання, відповідні функції, робочі процеси, алгоритми та гіперпараметри. Автоматизоване машинне навчання (*AutoML*) — це сфера *ML*, яка спрямована на розроблення рішень, що дають змогу різнобічним спеціалістам у науці, експертам і фахівцям, добре обізнаним з предметною областю ПрО (*citizen data scientist*) та з моделями, генерувати останні, застосовуючи прогнозну чи рекомендаційну аналітику (але основна частина роботи при цьому перебуває поза сферою статистики та аналітики), ефективно створювати точні прогностичні моделі з мінімальним втручанням людського інтелекту. Видобування знань із даних охоплює кілька етапів, таких як: вибір підмножини даних, очищення їх і перетворення, вибір функцій чи проєктування їх, а також застосування відповідних методів аналізу даних для вибору шаблонів (*pattern*) [1], налаштування алгоритму та оптимізацію гіперпараметрів моделі, оцінювання та інтерпретацію результатів і розгортання моделей машинного навчання. *AutoML* підвищує ефективність машинного навчання та прискорює дослідження у цій сфері. Алгоритм *AutoML* подано на рис. 2 [4], а успіх його реалізації вирішальною мірою залежить від експертів

[2] «*The Gartner hype cycle*» — «Цикл хайпа *Gartner*» або «Цикл очікувань» — крива входження або зрілості інноваційних технологій. Цикл очікувань *Gartner* — це графічна презентація, розроблена, використана та брендована американською дослідницькою, консультативною та інформаційною компанією *Gartner*, щоб показати зрілість, впровадження та соціальне застосування конкретних технологій. Цикл очікувань стверджує, що забезпечує графічне та концептуальне подання зрілості нових технологій у п'ять етапів. Модель піддавалася критиці з різних причин, у тому числі через ненаукову точність і використання суб'єктивної термінології — *https://en.wikipedia.org/wiki/Gartner_hype_cycle*

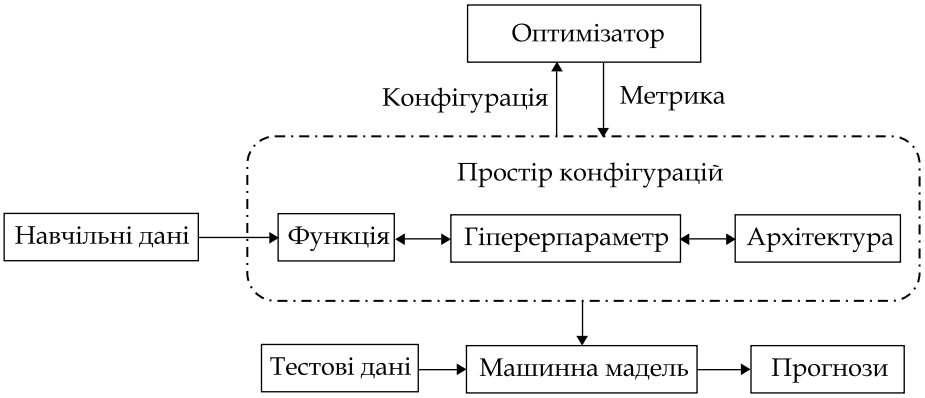

Рис. 2. Алгоритм автоматизованого машинного навчання

з *ML*, здатних виконати наведені завдання, із залученням глибокої експертизі фахівців з ПрО.

Історія створення *AutoML* сягає початку 2010-х років, коли фахівці в області *ML* почали вивчати способи автоматизації процесу розроблення моделей машинного навчання. У минулому десятилітті сфера досліджень, спрямована на завдання прогресивної автоматизації *AutoML*, була сформульована в [5, 6] як вирішення проблеми автоматизації ефективного виявлення об'ктів (*Object Detection, OD*) та проєктування конвеєрів *ML* – послідовності кроків, які перетворюють необроблені дані на машинну модель, прийнятну для виробництва. Фахівці з даних (*data scientists and researchers; citizen data scientists* — науковці та дослідники даних; фахівці, які добре обізнані з предметною областю) мали знайти правильний спосіб перетворення даних, щоб подавати їх у конвеєр *ML*. Будь-яка реальність (статичний чи динамічний розвиток подій) передбачає підготовку даних або їх попереднє оброблення [1]. Було досягнуто значного прогресу в наскрізному автоматичному *ML* для вирішення складніших завдань, таких як оброблення тексту, зображень, відео та мовлення з використанням методів самонавчання та посиленого навчання. Однак і ці методи мають багато варіантів моделювання та гіперпараметрів [5, 6].

Методи оптимізації гіперпараметрів

Гіперпараметри моделі можуть охоплювати не лише змінні, що відповідають перемиканню між альтернативними моделями, а й варіанти моделювання, такі як параметри попереднього оброблення, топологію та розмір нейронної мережі або параметри регуляризації алгоритму навчання — долучення деяких обмежень з метою запобігти перенавчанню, наприклад, деяких апріорних розподілів на параметри моделі. Оптимізація гіперпараметрів (ОГП, англ. *HPO*) дає змогу суттєво поліпшити продуктивність алгоритмів *ML*, адаптуючи

їх до цієї проблеми. Машинна модель має такий вигляд:

$$y = f(x; \theta) = f(x; a(\theta), \theta).$$

Тут вектор параметра моделі a є неявною функцією вектора гіперпараметра, отриманого для фіксованого значення θ, і навчальних даних *Dtrain*.

Спочатку проблему *AutoML* розв'язували за допомоги оптимізації гіперпараметрів. Це рішення передбачало, що є обраний алгоритм машинного навчання A, який має гіперпараметри $\theta_1, \ldots, \theta_n$ з відповідними областями визначення $\theta_1, \ldots, \theta_n$ і сумісний простір у вигляді багатомірного вектора $\theta = \theta 1 \times \ldots \times \theta 1_n$. Для кожного набору гіперпараметрів $\theta_i \in \Theta_i$ застосовується символ A_θ для позначення алгоритму навчання A із цим налаштуванням. Для позначення втрат під час перевірки (наприклад, рівня помилкової класифікації), якого досягає A_θ на даних *Dvalid* в разі навчання на *Dtrain*, користуються виразом $l(\theta) = L(A_\theta; Dtrain; Dvalid)$. Завдання оптимізації гіперпараметрів полягає в тому, аби знайти $\theta \in \Theta$, що мінімізує $l(\theta)$ [7]. Існує інше визначення, яке для багатьох алгоритмів машинного навчання можна сформулювати як мінімізацію критерію навчання, що включає гіперпараметр [8]. Цей гіперпараметр зазвичай обирається методом спроб та помилок з використанням критерію вибору моделі.

Стандартом де-факто оптимізації гіперпараметрів у машинному навчанні був простий пошук у сітці (*grid-search*), тобто вичерпний пошук серед усіх можливих комбінацій дискредитованої сукупності параметрів, який імовірно, буде обчислювально забороненим у великих просторах пошуку або з великими наборами даних його вельми проблематично використовувати. Для гіперпараметричної оптимізації, особливо в багатовимірних просторах, вважаються ефективнішими випадково вибрані випробування (*random search*), де простір досліджується протягом обмеженого часу, порівняно з пошуком по сітці. Його здійснюють за заздалегідь заданим розподілом імовірностей у просторі пошуку, і він є найпростішим, але досить ефективним на практиці підходом [9]. Є також інші підходи, такі як градієнтний пошук. У [8] наведено методологію оптимізації декількох гіперпараметрів, засновану на обчисленні градієнта критерію вибору моделі стосовно гіперпараметрів. Тут показано, що випадковий пошук є ефективнішим для оптимізації гіперпараметрів, ніж пошук по сітці. У порівнянні з нейронними мережами, налаштованими на чистий пошук по сітці, виявили, що випадковий пошук у тій же області дає змогу знаходити моделі, які є кращими за часом обчислень. Випадковий пошук при тому самому обчислювальному бюджеті знаходить кращі моделі за допомоги ефективного пошуку у більш-менш перспективному конфігураційному просторі. Є ще одне важливе зауваження: для наборів даних лише кілька гіперпараметрів справді

мають значення, але різні гіперпараметри є важливими для різних наборів даних. Це явище робить пошук у мережі поганим вибором для налаштування алгоритмів під нові набори даних [8–10]. Також є інші адаптивні (послідовні) методи, наприклад, еволюційний пошук і баєсівська оптимізація. Хоча ці адаптивні підходи відрізняються тим, як вони визначають, які моделі оцінювати, всі вони намагаються змістити пошук убік конфігурацій, які із більшою імовірністю працюватимуть добре.

Через базові припущення, зроблені завдяки різним методам пошуку, вибір відповідного методу може залежати від простору пошуку. Баєсівські підходи (БП) за допомоги гаусівських процесів для моделювання характеристик ефективності узагальнення, наприклад [11], і градієнтні підходи [12] зазвичай можуть бути застосованими до просторів безперервного пошуку. Баєсівські методи оптимізації часто поєднують з іншими методами, наприклад, такими, як ансамблеві методи, щоб отримати перевагу в деяких ситуаціях з обмеженням за часом [5]. Деякі з цих методів спільно розглядають дворівневу оптимізацію і приймають витрати часу як важливу установку для пошуку гіперпараметрів. БП на основі дерев [13], еволюційні стратегії [14] і випадковий пошук є гнучкішими, їх можна застосовувати до будь-якого простору пошуку. Застосування посиленого навчання до загальних завдань оптимізації гіперпараметрів є обмеженим через складність вивчення політики поведінки над великими просторами безперервної дії [5, 15].

БП, наприклад [13], що моделюють умовну імовірність продуктивності $p(y \mid \theta)$ за метрикою оцінки Y (тобто, точністю тесту), з огляду на конфігурацію набору гіперпараметрів Θ, спрямовані на швидше визначення перспективних змін. Вони забезпечують статистично стійкі (надійні) та виграшні конфігурації гіперпараметра моделі. Налаштування гіперпараметра виконується послідовно та концентрується на перспективних областях простору пошуку. БП вирішують принципово складне завдання одночасного підбору та оптимізації багатомірного простору, зокрема неопуклих функцій, і, можливо, оцінки шуму. Для вирішення цієї проблеми методи БП застосовують евристику для моделювання цільової функції або пришвидшення підпрограм ресурсів. Однак адаптивні методи вибору конфігурації за своєю суттю є послідовними, і їх важко розпаралелювати, що потрібно для пришвидшення обчислень.

Оскільки значення гіперпараметрів можуть ефективно впливати на кінцеву продуктивність робочого процесу, проблема полягає не лише в розмірі пошукового простору, а й у керуванні часом, необхідним для оцінювання кожного можливого рішення. Ця проблема містить і методи пошуку передбачуваних змін властивостей у просторі пошуку і оцінювання *ML*-моделей, тобто визначення виграшної (*scoring*) конфігурації гіперпараметра з набору можливих значень.

Простір пошуку містить цей набір конфігурацій і може вміщувати безперервні або дискретні гіперпараметри.

Байєсівська оптимізація (Bayesian Optimization). Автоматичний вибір робочого процесу було спочатку формалізовано в *Auto-Weka*[3] як завдання вибору комбінованого алгоритму та оптимізації гіперпараметрів (*CASH Combined algorithm selection and hyperparameter optimization*). Оптимізацію вибору моделі виконано за дворівневою програмою із зазначенням нижньої мети $J1$ для навчання параметрів моделі та верхньої $J2$ – для навчання гіперпараметрів θ. Обидва оптимізуються одночасно. Для визначення простору гіперпараметрів використано структуру дерев [16]. Баєсівські процедури оптимізації [5, 9, 11, 15–18]: послідовний алгоритм налаштування моделюванням (*SMAC – Sequential Model-based Algorithm Configuration*), оцінювач Парзена з деревоподібною структурою (*TPE, But Tree-structured Parzen Estimator*) і *Spearmint* (*TPE* не є стандартним баєсівським алгоритмом оптимізації [9]). *SMAC* використовує випадкові ліси для моделювання $p(y|\theta)$ як розподіл Гауса, а потім *Spearmint*, застосовуючи гаусові процеси (*GP*) для моделювання $p(y|\theta)$, виконує вибірку зрізів за гіперпараметрами *GPs*. Було виявлено [9, 19], що деревоподібний метод оцінювання Парзена, який моделює $p(x|y)$ і $p(y)$, а не $p(y|x)$ безпосередньо, перевершує методи баєсівської оптимізації на основі гаусівських процесів для задач структурованої оптимізації з багатьма гіперпараметрами, зокрема дискретні. Основна ідея цих методів полягає в тому, щоб підігнати $J2(\theta)$ до гладкої функції у спробі зменшити дисперсію та оцінити її в тих областях гіперпараметрічного простору, які недостатньо відібрано для направлення пошуку в області високої дисперсії.

Алгоритм налаштування моделюванням *SMAC*, який допоміг на кілька порядків пришвидшити і локальний пошук, і пошуки у дереві на певних розподілах примірників, виявився ефективним для гіпероптимізації параметрів алгоритмів машинного навчання, забезпечив краще масштабування до великих розмірів і дискретних вхідних вимірів. Пошук в об'єднаному просторі алгоритмів і гіперпараметрів дав ефективніші моделі, ніж вибір стандартних алгоритмів і гіперпараметра.

SMAC засновано на випадкових лісах та *Tree Parzen Estimator*, які підходять для оптимізації багатовимірних гіперпараметрів у глибоких нейронних мережах [19]. Мета байєсівської оптимізації – скласти імовірнісну модель і потім використовувати її для ухвалення рішення про те, як слід оцінювати наступну функцію. Принцип полягає у застосуванні інформації, отриманої в результаті попередньої

[3] *Weka* – популярна бібліотека машинного навчання, одна з застосовуваних бібліотек *ML*, яка містить множину методів попереднього оброблення та алгоритмів *ML*, які перевершують інші бібліотеки.

оцінки, що дає змогу знайти мінімум неопуклої складної функції для оптимізації гіперпараметрів [20].

В результаті цих розробок було створено інструментарій для автоматизації конвеєрів машинного навчання методом баєсівської оптимізації, і введено велику бібліотеку тестів для оптимізації гіперпараметрів під назвою *HPOlib*.

В [9, 17] показано, що баєсівські методи оптимізації емпірично перевершують випадковий пошук на кількох тестових завданнях. Однак для задач великої розмірності стандартні баєсівські методи оптимізації працюють аналогічно до випадкового пошуку. Методи, спеціально розроблені для задач великої розмірності, передбачають нижчу ефективну розмірність проблеми[4] або деревоподібне розкладання для цільової функції.

Інші рішення *AutoML* також спрямовано на пошук конвеєрів *ML* з хорошою продуктивністю. Розробки *Auto-Weka*, *Auto-Sklearn*, *PoSH Auto-Sklearn*, *TPOT* та інші вирішують, принаймні, проблему *CASH*.

Байєсівська оптимізація з послідовним поділом наявних ресурсів навпіл (Bayesian Optimization with SH Algorithm). PoSH Auto-Sklearn[5] поєднує в собі попередньо обраний портфель машинних навчальних конвеєрів, побудову ансамблю для досягнення стійкої продуктивності та баєсівську оптимізацію з послідовним поділом наявних ресурсів навпіл (*Bayesian Optimization with Successive Halving*) [21]. Спочатку запускається одна ітерація алгоритму *SHA* (*Successive Halving Algorithm*) із портфоліо машинних навчальних конвеєрів, а потім використовується баєсівська оптимізація для отримання нових конфігурацій гіперпараметра. Ідея, покладена в основу *SHA*, випливає безпосередньо з його назви: рівномірно розподілити бюджет по набору конфігурацій гіперпараметрів, оцінити продуктивність будь-яких змін, відкинути найгіршу половину й повторювати доти, поки не залишиться одна конфігурація. Таким чином, стратегія полягає в усуненні найгіршої половини конфігурацій гіперпараметрів і у збільшенні кількості часу на ітерації процесу, що залишилися. При цьому витрачається значно менше часу, наскільки це можливо, на претендентів (кандидатів) і відкидання непридатних, що дає змогу приділити більше часу перспективним. Алгоритм *SH* виділяє експоненціально більше ресурсів для перспективних конфігурацій. У машинному навчанні таке завдання — вибір із ряду грального автоматів типу «однорукий бандит» того з них, який забезпечить виграш за мінімальної витрати ресурсу (обмеження в грошах і часі) — отримало назву «бандитського» алгоритму або багаторуких бандитів (*multi-ar-*

[4] *Wang Y. et. al., Automatic reconstruction of fault networks from seismicity catalogs including location uncertainty,* 2013, 36 p. — *https://arxiv.org/ftp/arxiv/papers/1304/1304.6912.pdf*

[5] Це ім'я є абревіатурою *PortfolioSuccessiveHalving*, об'єднаної з бібліотекою *ML Auto-sklearn*.

med bandits, *MAB*). Його постановка називається завданням про багаторукого бандита. Останнім часом *SHA* було успішно застосовано до оптимізації гіперпараметрів, демонструючи високу продуктивність за невеликих бюджетів.

SHA дає змогу вказати мінімальний і максимальний бюджет для кожної конфігурації (наприклад, під кутом зору кількості навчальних ітерацій або розміру набору даних). Він починає з мінімального бюджету й ітеративно збільшує бюджет для конвеєрів, які працюють із поточним бюджетом.

Простір конфігурації є підпростором *Auto-sklearncon-figuration*, що дозволяє працювати з бюджетами *SH*: попередні оброблення набору даних (масштабування об'єктів, обчислення пропущених значень, обробка категоріальних значень) і один із чотирьох класифікаторів: метод опорних векторів (*SVM*), випадковий ліс, лінійна класифікація (за допомоги стохастичного градієнтного спуску) або *XGBoost*, що призводить до 37 гіперпараметрів.

Successive Halving покращує оптимізацію гіперпараметра у випадковому пошуку за допомоги ділення та вибору випадково згенерованих конфігурацій гіперпараметрів ефективніше, ніж *Random Search*, не роблячи припущень щодо природи простору конфігурацій. Ефективніший розподіл ресурсу означає, що, застосовуючи той самий набір конфігурацій, *Successive Halving* знайде оптимальну конфігурацію швидше.

Оптимізація гіперпараметрів (Hyperband). *Hyperband* — це алгоритм адаптивного розподілу ресурсів із принципового раннього зупинення оптимізації гіперпараметрів [15, 22]. Другий приклад алгоритмів, що оптимізують розподіл ресурсів — це *Hyperband* і асинхронний *Successive Halving*. На жаль, оригінальний алгоритм *SH* потребує в ролі вхідних даних кількість n конфігурацій. З огляду на деякий кінцевий бюджет B (наприклад, час навчання для вибору конфігурації гіперпараметра), ресурси B/n розподіляються в середньому за конфігураціями. Проте, для фіксованого B, не зрозуміло, чи має апріорі розглядатися (*a*) безліч конфігурацій (велике n) із невеликим середнім часом навчання; або (*b*) невелика кількість конфігурацій (мале n) з тривалішим середнім часом навчання. Інакше кажучи, алгоритму *SH* властива дилема «n проти B/n». Відсутність цієї додаткової інформації перешкоджає застосуванню методу оцінювання конфігурації, тим самим мотивуючи нестохастичну установку для оптимізації гіперпараметрів [21]. У літературі про *MAB* для визначення кумулятивної помилки (*regret*, регрет) алгоритму провісника використовуються алгоритми та здійснюються дослідження і для нестохастичних, і для стохастичних параметрів, тоді як оптимальну схему ідентифікації розглянуто лише в стохастичної постановці. Нестохастичне завдання ідентифікації найкращої якості не здійснювали [21].

Потрібен алгоритм, який за будь-яких значень параметрів забезпечив би теоретично обґрунтоване мінімальне значення *regret*, яке визначають відстанню від оптимального рішення, за максимально короткий час. Це в чистому вигляді є завданням дослідження *MAB*, яке могло би забезпечити теоретичні гарантії та обнадійливі емпіричні результати в налаштуванні гіперпараметра.

Гіперпараметрічну оптимізацію формулюють як проблему нестохастичного бандита з нескінченним озброєнням (*non-stochastic infinite-armed bandit problem*), де визначений ресурс, такий як ітерації для ітераційних алгоритмів, вибірки даних або функції, виділяють для випадково обраних конфігурацій. *Hyperband* розв'язує проблему «*n* проти *B*/*n*», розглядаючи кілька можливих значень *n* для фіксованого *B*. По суті, виконується пошук по сітці за допустимим значенням *n*. З кожним значенням *n* пов'язано мінімальний ресурс *r*, який виділяється всім конфігураціям до того, як деякі з них відкидаються; більше значення *n* відповідає меншому *r* і, отже, агресивнішому ранньому зупиненню. *Hyperband* розширює алгоритм послідовного розподілу навпіл *SH* [21], застосовуючи його як підпрограму організації внутрішнього циклу *Hyperband* (внутрішній цикл викликає *Successive Halving* для фіксованих значень *n* і *r*) з автоматизацією вибору швидкості раннього зупинення за допомоги запускання різних варіантів *SHA*.

Як зазначено в працях [15, 22], *Hyperband* надає надійний, теоретично обґрунтований алгоритм адаптивного розподілу ресурсів і принципового раннього зупинення оптимізації гіперпараметрів. Він спрямований на пришвидшення випадкового пошуку [9], розроблений для нестохастичного настроювання, та автоматично адаптується до невідомого *F* (незалежно від рівня гладкості або структури функції *F*, що оптимізується) без будь-яких параметричних припущень. Випадковий пошук буде асимптотично сходитися до оптимальної конфігурації за будь-якого *F* за допомоги простого аргументу покриття. Швидкість збіжності для випадкового пошуку залежить від гладкості та є експоненційною за кількістю вимірів у просторі пошуку. Це є так само вірним і для баєсівських методів оптимізації без додаткових структурних припущень. Отже, створено новий підхід [15] до оцінювання конфігурації як вирішення проблеми адаптивного розподілу ресурсів випадково обраних конфігурацій гіперпараметрів. Процедура *Hyperband* спирається на принципову стратегію раннього зупинення для розподілу ресурсів, даючи змогу оцінювати на порядки більше конфігурацій, ніж процедури «чорного ящика», такі як баєсівські методи оптимізації. Порівнюється робота цього алгоритму з популярними методами баєсівської оптимізації на наборі завдань оптимізації гіперпараметрів. Спостерігається, що *Hyperband* може забезпечити швидкодію вищою на порядок, порівняно з конкурентами, на множині проблем посиленого навчання та навчання на

основі ядра. Так, *Hyperband* працює у 5—30 разів швидше, ніж баєсівські алгоритми оптимізації. Однак він не годиться для паралельного настроювання під час роботи з великими даними з огляду на схильність відставати в обробленні даних та з тієї ж причини пропущені завдання [15].

Асинхронне послідовне подвійне скорочення (Asynchronous Successive Halving). Сучасні моделі навчання характеризуються великими просторами гіперпараметрів і тривалим часом навчання. Ці властивості зумовлюють необхідність розпаралелювання обчислень і, тим самим, мотивують необхідність розроблення досконалих функцій оптимізації гіперпараметрів в умовах розподілених обчислень, щоб мати можливість знайти добру конфігурацію за розумний час. Користуватися випадковим пошуком і пошуком по сітці за паралельного налаштування, оскільки ці два методи є тривіальними для розпаралелювання та легко масштабуються на будь-яку кількість машин, не є доцільним, оскільки обидва методи є підходами «грубої сили» і погано масштабуються з кількістю гіперпараметрів. За послідовного налаштування застосовують *SH*, добре відомий алгоритм багаторукого бандита (*SHA*). Тоді можна оцінити на кілька порядків більше конфігурацій гіперпараметрів, ніж в разі випадкового пошуку, внаслідок адаптивного розподілу ресурсів для перспективних конфігурацій. Цим досягається швидше дослідження простору пошуку та стандартно висока якість для великих наборів даних. Однак розпаралелити процес обчислень важко, тому що алгоритм приймає набір конфігурацій як вхідні дані, й очікує завершення будь-яких змін у ланцюжку, перш ніж відбудеться перехід до наступної ланки ланцюжка. Щоб усунути це вузьке місце, що створюється синхронними просуваннями, автори [15, 23] налаштовують алгоритм *SH* так, щоб просувати вгору конфігурації наскільки це можливо, замість того, щоб починати з широкого набору конфігурацій і звужувати їх. Такий алгоритм назвали асинхронним алгоритмом послідовного розподілу навпіл (*ASHA*). Автори стверджують, що налаштування обчислювально важких моделей із використанням масового паралелізму є новою парадигмою оптимізації гіперпараметрів.

ASHA використовує паралелізм і агресивне раннє зупинення для вирішення великомасштабних завдань оптимізації гіперпараметрів. Чималі емпіричні результати показали, що *ASHA* перевершує наявні сучасні методи оптимізації гіперпараметрів, масштабується лінійно з кількістю робочих вузлів у розподілених налаштуваннях, і годиться для масового паралелізму.

Еволюційна оптимізація. Інші методи пошуку застосовують еволюційні алгоритми. Ці підходи враховують сукупність налаштувань гіперпараметрів, модифікують і усувають безперспективні налаштування відповідно до їхньої оцінки перехресної перевірки. Але по-

при разючі результати, вони потребують величезної кількості обчислювальних ресурсів, і їх важко масштабувати.

TPOT (*Tree-Based Pipeline Optimization Tool*) [14] — деревоподібний інструментарій проєктування й оптимізації конвеєрів *ML*, працює як і *Auto-Sklearn*, у тандемі з бібліотекою *scikit-learn*, описуючи себе як її оболонку на *Python*. У його основі лежить платформа *DEAP*[6], модульна структура (*a modular framework*), яка допомагає реалізовувати процеси, описані еволюційними алгоритмами (*EA*). Реалізація застосовує генетичне програмування (*GP*) — еволюційну техніку обчислень для автоматичного конструювання комп'ютерних програм — із залученням *ES* (*Evolution strategy*)[7] стратегії.

Успішні розробки у сфері еволюційної багатоцільової оптимізації (*Multi-Objective MO*) і застосування концепції парето-оптимальності до машинного навчання підвищують продуктивність також і традиційних методів одноцільового машинного навчання, створюючи різноманітні множинні парето-оптимальні моделі під час побудови ансамблів моделей. Ці моделі відповідають теоретичним вимогам хороших ансамблів. Даючи змогу користувачеві генерувати ансамблі і вибирати моделі відповідно до потреб, багатоцільовий підхід тим самим прагне поліпшити алгоритми машинного навчання, долаючи спільні проблеми балансу компромісу між точністю та різноманітністю. У цьому сенсі *MO* має перевагу перед одноцільовими методами, оскільки під кутом зору багатоцільової оптимізації не має єдиної моделі навчання, яка могла б одночасно вирішувати різні завдання. Багатоцільова оптимізація на основі принципа Парето — єдиний спосіб впоратися з конфліктними цілями [24, 25].

---

[6] *DEAP* (*Distributed Evolutionary Algorithms в Python*) — це еволюційне обчислювальне середовище для швидкого прототипування та тестування ідей. Вільно доступне, з великою документацією (http://deap.gel.ulaval.ca), *DEAP* є проєктом з відкритим вихідним кодом під ліцензією *LGPL*. Еволюційні обчислення (*EC*) — складна область із дуже різноманітними методами та механізмами, де навіть добре спроєктовані структури можуть стати доволі складними, щоб розглянути можливість внесення змін. Мова програмування *Python* забезпечує необхідний зв'язок для збирання складних *EC*-систем, надаючи практичні інструменти для швидкого створення прототипів для користувача еволюційних алгоритмів, де кожен крок процесу є настільки явним (як псевдокод), легко прочитуваним і зрозумілим, наскільки це можливо. Це також надає великого значення і компактності, і зрозумілості коду — [*Fortin F.-A., Rainville F.-M. D., Gardner M.-A., Parizeau M. and Gagn C., Jul. 2012, DEAP: Evolutionary Algorithms Made Easy, Journal of Machine Learning Research, vol.* 13, *pp.* 2171–2175].

[7] *ES* — евристичний метод оптимізації в розділі еволюційних алгоритмів, базований на адаптації та еволюції. Розроблено в 1964 році німецьким ученим І. Рехенберге і розвинено в подальшому Швефелом Х.-П. та ін — [*Schwefel Hans-Paul. Cybernetic Evolution as Strategy for Experimental Research in Fluid Mechanics (in German). Diploma Thesis. Hermann Föttinger-Institute for Fluid Mechanics, Technical University of Berlin, March 1965. H.-G. Beyer and H.-P. Schwefel, Evolution strategies. A comprehensive introduction, Natural Computing, vol. 1, no. 1, pp.* 352, *Mar.* 2002].

Об'єднання *GP* і оптимізації за Парето дає змогу автоматично створювати високоточні та компактні *ML*-конвеєри внаслідок того, що еволюційні алгоритми на основі цього принципа успішно застосовують для оцінювання багатьох рішень, тобто послідовності конвеєрів. Ця особливість призводить до повільної оптимізації, тому що навчання є трудомістким процесом. У робочому процесі *AutoML* є попереднє оброблення функцій (*Feature Preprocessing*) — це очікуваний крок у конвеєрах машинного навчання, бо з'являється можливість змінювати функції[8] так, щоб спростити набір даних і забезпечити продуктивність моделі машинного навчання. *TPOT* оптимізує конвеєри, зокрема низку препроцесорів функцій і моделей *ML*, з метою вибору найбільш відповідних і ефективніших конвеєрів у контрольованій задачі класифікації. Побудова з них деревоподібних конвеєрів — дерев *GP* — відбувається за допомоги об'єднання операторів машинного навчання, що їх використовують як примітиви генетичного програмування. Кожен оператор конвеєра *ML* (примітив *GP*) у ньому відповідає алгоритму машинного навчання. Усі реалізації алгоритмів машинного навчання взято з *scikit-learn* (крім *XGBoost*).

У генетичному програмуванні кожне рішення (або кандидат) подано як дерево, яке складається з вузлів і листя. Кожен внутрішній вузол називається примітивом, а кожен листок — терміналом. Примітив розглядають як функцію, термінал — як аргумент або константу. Примітиви та термінали визначають залежно від розв'язуваної проблеми. У разі *TPOT* примітиви, алгоритми *ML* або методи та термінали попереднього оброблення є гіперпараметрами. Кореневий вузол дерева — обов'язково алгоритм *ML*, який використовують як остаточну оцінку. Інші вузли діють як оператори, що беруть дані на вхід і надають на виході перетворені дані. Дерева можуть складатися з різних гілок, де дві гілки об'єднуються через спеціальний примітив, поданий *TPOT*. Злиття об'єднує висновок із попередніх вузлів.

Отже, шлях вирішення проблеми оптимізації конвеєра машинного навчання — оператори машинного навчання, які використовують як примітиви генетичного програмування, що ґрунтуються на дереві конвеєрів, використовують для об'єднання примітивів в робочі навчальні конвеєри машин і алгоритм *GP*, який використовують для модифікації деревоподібних конвеєрів.

Під час роботи з великими наборами даних *TPOT* є дуже повільним, тому що розмір навчальної вибірки і кількість оцінюваних

[8] *Scikit-feature*: репозиторій вибору об'єктів, зокрема функцій, з відкритим вихідним кодом на *Python*. Ґрунтується на пакеті машинного навчання *scikit-learn* і двох наукових обчислювальних пакетах *Numpy* і *Scipy*. *Scikit-feature* містить приблизно 40 популярних алгоритмів вибору функцій, зокрема традиційні алгоритми та деякі алгоритми структурних і потокових функцій — *https://www.kdnuggets.com/2016/03/scikit-feature-open-source-feature-selection-python.html*

кандидатів є суттєвими. Еволюційні алгоритми зазвичай повільно сходяться, що робить *TPOT* нездатним масштабуватися до великих наборів даних. Відома робота [26], в якій для пришвидшення дослідження простору пошуку та поліпшення продуктивності за великих наборів даних застосовують алгоритм *Successive Halving* послідовного поділу навпіл, використовують у задачах *Multi-Armed Bandit*. При цьому *TPOT-SH* — швидше рішення *AutoML* — скорочує час, якого потребують еволюційні алгоритми, щоб знайти конвеєри *ML* для великих наборів даних.

Сфера застосовності AutoML

З викладеного випливає, що *AutoML* — це потреба автоматизації наскрізного процесу застосування машинного навчання до реальних проблем. Крім складності автоматизації багатьох завдань з оброблення даних, суть *AutoML* полягає в допомозі спеціалістам з даних та звільненні їх від тягаря повторюваних і менш вимогливих завдань, щоб вони могли витрачати свій час на складніші, творчі та важкі для автоматизації завдання. Фахівці *data scientists* через брак *AutoML* мають виконати налаштування алгоритму та оптимізацію гіперпараметрів, щоб максимізувати прогнозовану продуктивність своєї остаточної моделі машинного навчання. Оскільки багато із цих кроків часто виходять за рамки можливостей *Citizen Data Scientist* — експертів у ПрО, *AutoML* для дедалі більших потреб застосування *ML* було запропоновано як рішення на основі штучного інтелекту. Нині системи *AutoML* можуть швидко створювати прогнозні моделі, що забезпечують оптимальну продуктивність. Однак їх охоплення, як і раніше, є вузьким, а їхній справжній потенціал досі ще не використовується. Є три обмеження наявних систем *AutoML*: складні типи даних (1), знання предметної області (2) і самонавчання та посилене навчання (3) [27].

На рис. 3 показано типовий командний робочий процес фахівців з даних *TDSP* (*Team Data Science Process*) — це гнучка ітеративна методологія оброблення даних, яка дає змогу ефективно надавати рішення прогнозної аналітики та інтелектуальних програм [28]. Вона акцентує обмеженість сфер, у яких нині використовується *AutoML* [27].

1. *Складні типи даних.* Інструменти *AutoML* можна значною мірою класифікувати за сценарієм застосування або форматом навчальних даних. Більшість із них було спроєктовано для роботи з найпоширенішим і визнаним типом даних, (згідно з опитуванням *Kaggle*[9])

[9] *Kaggle* — головний організатор конкурсів з дослідження даних, а також соціальна мережа спеціалістів з обробки даних і машинного навчання. *KDnuggets* (*nugget* — самородок) — провідний сайт з науки про дані, машинне навчання, штучний інтелект й аналітику було засновано у квітні 2010 р. Григорієм П'ятецьким-Шапіро. *KD* (*Knowledge Discovery*) — означає «Знаходження (відкриття) знань» — *https://www.kdnuggets.com/about/index.htm*

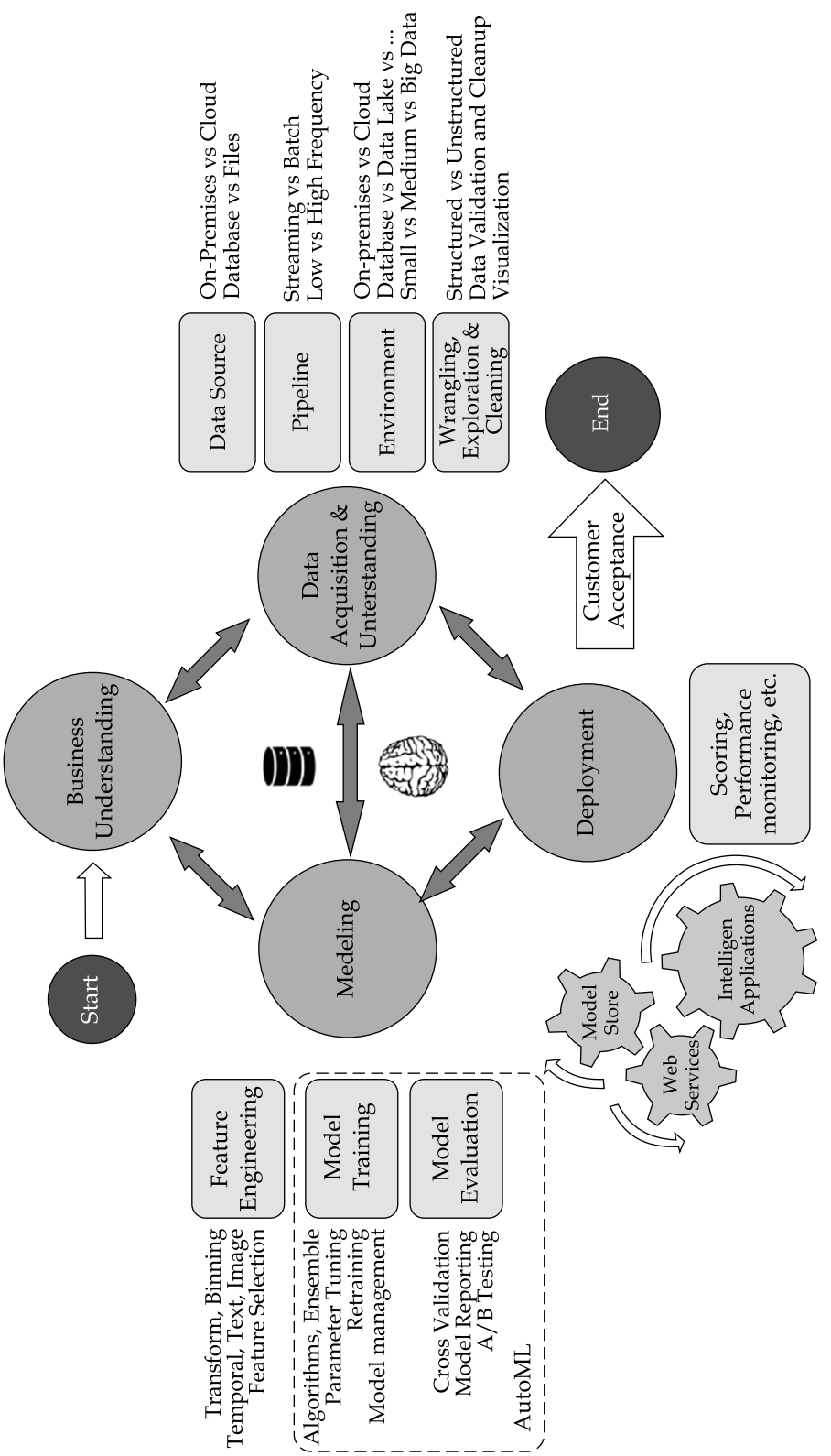

Рис. 3. Позиція *AutoML* у робочому процесі за даними *TDSP* та основні функційні вузли цього процесу

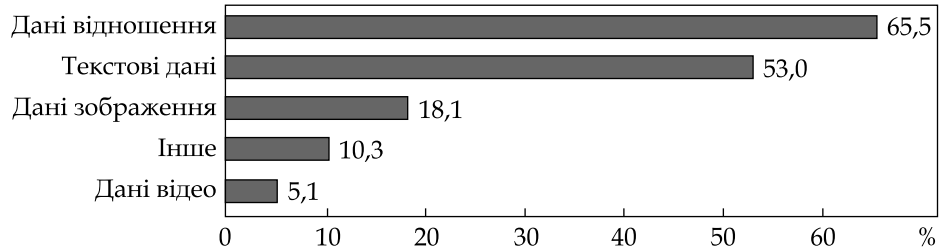

Рис. 4. Найпоширеніші типи даних систем *AutoML*

ML and Data Science Survey 2017 року [27], реляційні структуровані табличні дані — *IID* (таблиці з рядками та стовпцями), подані числами або категоріальним змінними (номінальні дані, наприклад, стать, колір шкіри, і порядкові дані, наприклад, рівень освіти). Іншим традиційним типом даних, переважно у фінансовій та роздрібній торгівлі, є дані з тимчасової залежністю. Флагманський продукт *H2O Driverless AI*[10] компанії *H2O.ai*[11] містить інструменти для автоматичної побудови прогнозних моделей часових рядів. Його, як і його попередника *H2O AutoML*, зорієнтовано на табличні дані *IID*, він підтримує дані часових рядів і необроблений текст. Деякі системи *AutoML* також було розширено для оброблення неструктурованих даних, а саме тексту та зображень (наприклад, *Google Cloud AutoML Natural Language, Google Cloud AutoML Vision, AutoKeras, DataRobot*), оскільки цей тип даних є вельми поширеним (рис. 4).

У разі текстових даних може бути застосовано інструмент загального призначення, такий як *H2O Driverless AI*, або спеціально розроблений для тексту, такий як *Google Cloud Natural Language*. Для вирішення з відкритим вихідним кодом, застосовують алгоритм *H2O Word2Vec* (або будь-який інший інструмент оброблення тексту з відкритим вихідним кодом), щоб перетворити текст на числовий формат, який може бути застосовано *H2O AutoML* (безпосередня підтримка текстових даних у *H2O AutoML* — його дорожня карта).

Мережеві та веб-дані, з іншого боку, є двома складнішими типами даних. Ці типи даних й досі не є предметом розгляду *AutoML*, що обмежує тип проблем, які розв'язують за допомоги *AutoML*. Однак, імовірно найближчим часом, автоматичні засоби оброблення цих типів даних відображатимуть міру розвитку інструментарію систем *AutoML*.

2. *Знання предметної області.* Системи *AutoML* вважають синонімом вибору моделі та налаштування гіперпараметра, і є лише невеликою частиною пазла у видобуванні знань у базах даних (*the knowledge*

[10] *H2O Driverless AI*, *https://www.h2o.ai/products/h2o-driverless-ai/*

[11] *H2O*.ai — компанія-розробник програмного забезпечення з Силіконової долини, яка вивела на ринок нові технології для просування *AI*, є творцем *H2O* — провідної платформи з відкритим вихідним кодом для дослідження даних і машинного навчання. Рішення *H2O* підтримує швидкі алгоритми *ML*, розподілені в оперативній пам'яті, і забезпечує самонавчання та посилене навчання.

discovery in databases, KDD) підприємства. Ці два етапи найлегше автоматизувати, враховуючи об'єктивність і послідовність їхніх кроків у завданнях *ML*-навчання. Проте, один із ключових компонентів для створення адекватних моделей *ML* часто ігнорувався в системах *AutoML* — це розроблення функцій. Проєктування функцій (це більше, ніж наука/знання), імовірно, це можливість, яка пропонує найблагодатніший ґрунт для розквіту людської творчості. Створення функцій фахівцями, кожен із яких намагається моделювати з метою розкриття значущих аспектів процесу, потребує уяви, творчості та досвіду в предметній області. Отже, продуктивність може варіюватися, якщо розроблення функцій виконують різнопрофільні фахівці. Розроблення функцій фахівцем також залежить від проблем і типу об'єктів, які можна відтворити, часто обмежене вхідним набором даних. Як наслідок, це один із важких, тривалих і трудомістких етапів будь-якого проєкту *data science*, поряд із очищенням і попереднім обробленням даних. Проте, достатньо хороші моделі може бути побудовано за допомоги прийняття загальнішої структури для створення ознак, що не залежать від набору даних (наприклад, *Deep Feature Synthesis* (*DFS*) — синтез посилених особливостей) [27].

Однак деякі платформи *AutoML* пропонують автоматичне проєктування функцій (наприклад, *DataRobot, H2O Driverless AI*). Проте жодна з них не може автоматично долучати знання предметної області до процесу *ML*, і це залишається виняткова людською функцією. Цікавим прикладом розв'язання цього питання є підхід, реалізований у *KNIME* [29, 30], так звана *Guided Analytics* — керована аналітика для автоматизації машинного навчання. Завдання її полягає в тому, щоб дозволити фахівцям, які працюють з даними, створювати інтерактивні системи, інтерактивно допомагаючи бізнес-аналітикам у їхньому прагненні знайти нове розуміння даних та передбачити майбутні результати.

Попри все це, заслуговують на увагу спроби автоматизувати складну та трудомістку задачу розроблення функцій, де секретним компонентом отримання високоякісних моделей у багатьох реальних задачах як і раніше залишається знання предметної області. Майбутній розвиток *AutoML* має бути зосереджений на створенні складніших методів для долучення певних знань в автоматично створювані функції. В ідеалі має бути розроблено складніші методи для долучення специфічних для предметної області знань в автоматично створювані функції із застосуванням закономірностей і залученням міждисциплінарної команди до розроблення продуктів *AutoML*. Системи *AutoML* також мають пропонувати можливість комбінування функцій, що генеруються автоматично, з функціями, створеними фахівцями, це свідчило б про їхню гнучкість [27].

3. *Самонавчання* (*UL*) *та посилене навчання* (*RL*). Глибокі нейронні мережі (*DNN*) демонструють дуже високу продуктивність під час

вирішення багатьох завдань машинного навчання, але вони дуже чутливі до налаштування своїх гіперпараметрів. При навчанні нейронної мережі необхідно налаштувати множину гіперпараметрів (швидкість навчання, розмір пакета, швидкість відсіву тощо), проте одним із найзначніших чинників, що впливають на продуктивність моделі, є мережева архітектура. В області завдань класифікації зображень (див. рис. 4) найчастіше використовують метод *AutoML*, який називають «пошуком нейронної архітектури» (*neural architecture search, NAS*). Є кілька інструментів із відкритим вихідним кодом для пошуку нейронної архітектури, зокрема реалізація ефективного пошуку нейронної архітектури (*efficient neural architecture search, ENAS*) — наприклад, *TensorFlow* і *Auto-Keras*, яка виконує ефективний *NAS* із застосуванням байєсівської оптимізації гіперпараметрів. *SaaS Google Cloud* пропонує *AutoML Cloud Vision* і *Cloud Natural Language*. Вони виконують пошук *NAS* для класифікації зображень і тексту відповідно. Цим інструментам потрібні необроблені графічні або текстові дані безпосередньо, і тому вони не надаються для типових числових чи категоріальних табличних даних. Двома областями, в яких *Deep Neural Networks* поліпшили продуктивність порівняно з традиційними методами машинного навчання, є класифікація зображень і оброблення природної мови (*natural language processing, NLP*) [27, 31].

Сучасний стан візуального розпізнавання об'єктів характеризується застосуванням згорткових нейронних мереж (*CNN*). Їх ємністю можна керувати, варіюючи глибину та ширину мережі, у порівнянні зі стандартними нейронними мережами прямого поширення з шарами однакового розміру. Також вони роблять задовільні припущення про природу зображень (а саме, про стаціонарність статистики та локальність піксельних залежностей) [32]. *CNN* мають набагато менше зв'язків і параметрів, тому їх легше навчати, а їхні теоретично найкращі результати продуктивності, ймовірно, є не гіршими. Кількість гіперпараметрів, що налаштовуються, зменшується. У [33] наведено, як приклад, гіперпараметри діапазону архітектур, зокрема у повністю пов'язаних мережах (*Fully Connected Networks*), великих згорткових (*Large Convolutional*) і маленьких згорткових (*Small Convolutional*) нейронних мережах *CNN*. Простір гіперпараметрів змодельовано за зразком архітектури в [32].

Попри привабливі якості *CNN* та відносну ефективність локальної архітектури, застосування їх у великих масштабах до зображень із високою роздільною здатністю, є занадто витратним. Сучасні графічні процесори у поєднанні з високо-оптимізованою реалізацією 2*D*-згортки є достатньо потужними, щоб полегшити навчання дуже великих *CNN*, а набори даних, такі як *ImageNet*, містять достатньо розмічених прикладів для навчання таких моделей без серйозного перенавчання. Ідеться про велику глибоку згорткову нейронну мережу, яка має 60 мільйонів параметрів і 500 000 нейронів, складається

з п'яти згорткових шарів, за деякими з яких ідуть шари максимального пулу, і двох глобально пов'язаних шарів із фінальним *softmax* із 1000 шляхів. Для прискорення навчання застосовувалися ненасичені нейрони та ефективна реалізація згорткових мереж на графічному процесорі. Щоб зменшити перенавчання у глобально пов'язаних шарах, застосували оригінальний метод регуляризації [32].

Традиційний *AutoML,* концентруючись на контрольованих задачах машинного навчання, які потребують маркованих, розмічених даних (*labelled data*) як вхідних, не передбачає складніші випадки навчання: самонавчання та посиленого *RL*-навчання. Навчання без учителя спрямовано на відкриття закономірностей з урахуванням даних, коли вірогідна інформація є недоступною. Немає чіткої міри успіху, яку можна застосовувати для оцінювання якості результатів самонавчання, оскільки немає строго обґрунтованого факту (немає точки відліку), з яким було би можливо порівнювати (наприклад, чи є невелике зображення на рентгенограмі пухлиною тощо). В результаті складніше судити про ефективність різних підходів, оскільки немає прямого способу порівняти їх. Ця суб'єктивність у визначенні і важлива роль експертних знань під час процесу є двома ймовірними причинами, з яких наявні системи *AutoML* не охоплюють цей підхід. Проте, з огляду на те, що у світі більшість даних не мають маркування, системи *AutoML* стали б іще кориснішими, якби їх застосування було розширено внаслідок автоматичного коригування машинних моделей. Посилене навчання, коли програмні агенти навчаються виконувати завдання методом спроб та помилок, отримують зворотний зв'язок від власних дій. Якщо дія є кроком до досягнення мети, то агент отримує винагороду. Інакше це карається. Таким чином, агент навчається на своїх помилках і вдосконалюється з досвідом. Подібно до навчання з учителем, при посиленому навчанні існує певна міра успіху, яка робить це завдання автоматизації *ML* реальним. Однак не було запропоновано жодної системи *AutoML* для автоматизації процесу самонавчання [27].

Подальший розвиток штучного інтелекту та *AutoML*. Тенденції

Наразі штучний інтелект застосовують у більшості областей знань. Він швидко розвивається та поширюється на кілька нових сфер, таких як мультимодальні застосунки та мобільні пристрої, що призводить до зростання технологічного розвитку небаченими раніше темпами. До 2021 року турбота про прозору й гідну довіру до *AI* призвела до того, що концепція відповідального та зрозумілого *AI* почала набувати дедалі більшого значення, допомагаючи зрозуміти, як працює *AI*. Ще одна сфера, в якій спостерігається сильне зростання — це гіперавтоматизація з використанням множини взаємозалежних тех-

нологій, таких як штучний інтелект та машинне навчання. Еволюція інтелектуальної автоматизації процесів та її впровадження забезпечить компаніям більшу гнучкість, принісши більше користі, ніж просто економія часу, підвищивши операційну ефективність і допомагаючи спростити процеси. Останнім часом напрямок розвитку *AI* полягає в прагненні покращити інтерпретованість моделей. Отже, дослідження дедалі більше рухатимуться вбік гібридних моделей між символічним *AI* з високим рівнем розуміння для людей та реальними статистичними глибокими нейронними мережами — рішення, яке забезпечить довіру учасників до надійного запровадження відповідальних систем [34].

В умовах певного успіху сучасних технологій і швидкого зростання досліджень та застосунків у галузі штучного інтелекту виникають і безпрецедентні можливості, і обґрунтовані побоювання з приводу його потенційного неправильного використання. Цю заяву зробила відомий вчений *Isabelle Guyon*[12] у рамках проєкту *HUMANIA*[13] [35]. Це занепокоєння є спільним. У роботі [36] авторами «вноситься певна помірність в уявлення, що виникло останніми роками, що проблему розпізнавання образів, яку традиційно вважають одним з розділів штучного інтелекту, цілковито вичерпано проблемою машинного навчання. Наука про штучний інтелект схильна зараз до серйозного випробування внаслідок надміру захопленої реклами наукових досягнень, адресованої переважно за межі наукової спільноти. Не можна вирішувати будь-яке складне завдання без будь-яких інтелектуальних зусиль щодо його дослідження» [3]. Два погляди на одну проблему, спільна занепокоєність — не ввести в оману, не нашкодити собі та суспільству — але шляхи її подолання є різними. Висновок другої роботи не потребує пояснень, а у першій він формулюється так: нагальна потреба зробити *AI* доступнішим для використання ширшим шаром населення. У контексті опублікованого документа «*HUMANIA. Artificial Intelligence for All*» [35] висловлено прагнення допомогти розв'язати це питання. Це має стати важливим фактором економічного зростання та сприяти зміцненню демократизації

[12] *Isabelle Guyon* — Директор, науковий співробітник *Google Research* (з жовтня 2022 р.). Завідувач кафедри штучного інтелекту (*PR EX1*) та дослідник *INRIA*, машинного навчання та оптимізації (команда *TAU*), Міждисциплінарна лабораторія цифрових наук (*LISN*) Університету Париж-Саклі, Франція (з 2015 року — професор). Співголова програми *NeurIPS* 2016 та генеральний голова *NeurIPS* 2017; потім член правління *NeurIPS*. З 2003 року організатор змагань з машинного навчання. Використання завдань як напряму досліджень у таких галузях, як причиново-наслідковий зв'язок, комп'ютерний зір, автоматичне машинне навчання та фізика високих енергій.

[13] This is the *ANR-19-CHIA*-0022 *project, a* 4*-year chair for the democratization of AI, started in September* 2020. *Publications on HAL. It is funded by L'agence Nationale de la Recherche* (*ANR*). По суті, це короткий виклад проєкту *ANR-19-CHIA*-0022, його мета, концепція, основні напрями наукових досліджень.

штучного інтелекту. Значення *AutoML* у сфері штучного інтелекту є неоціненним. Він демократизував процес створення та впровадження моделей машинного навчання, зробивши його доступним для ширшої аудиторії з різним рівнем знань у галузі машинного навчання. Автоматизуючи складні та трудомісткі завдання, *AutoML* значно знижує вхідний бар'єр для непрофесіоналів, які бажають використовувати можливості *AI* у своїх застосунках [37].

Дослідження [35] спрямовано на зниження потреби у фаховому досвіді під час реалізації алгоритмів розпізнавання образів та моделювання, включно з посиленим навчанням, у різних галузях застосування (медицині, інженерії, соціальних науках, енергомережах тощо) із застосуванням кількох модальностей (зображень, відео, тексту, часових рядів тощо). Цьому сприятиме організація наукових конкурсів проєктів (або змагань) з автоматизованого машинного навчання. Кульмінацією цих зусиль стане конкурс *AutoRL* (посиленого автоматичного навчання — *Automated Reinforcement Learning*). Проєкти готуватимуть спільноту до складніших та різноманітніших умов, постійно зменшуючи необхідність втручання людини у процес моделювання. «...Залучаючи наукове співтовариство як ціле у розв'язання складних завдань, ми ефективно помножимо на важливий чинник наше державне фінансування для вирішення таких складних задач *AutoML...*» [35].

AutoML змінив підхід до штучного інтелекту, демократизуючи його розроблення та прискорюючи темпи інновацій. Його еволюція, значення, реальне застосування підкреслюють ключову роль *AutoML* у формуванні майбутнього *ML* та *AI*. Цей підхід ґрунтується на низці фундаментальних робіт [35], в одній із яких запропоновано розробити систематичну й уніфіковану методологію для організації та використання наукових завдань у викладанні та дослідженнях у галузі *ML*, а саме штучного інтелекту, керованому даними (*data-driven Artificial Intelligence*). На сьогодні завдання стають дедалі популярнішими як засіб просування нових досягнень завдяки залученню вчених різного віку і в академічних колах, і за їхніми межами. Це може бути формою *citizen science*[14] громадянської науки. Є надія, що такий внесок спільноти в науку посприяє відтворенню досліджень та демократизації штучного інтелекту. Проте, є небезпека, що коли дані (або показники, що використовуються для визначення завдань), містять упередженість або артефакти, результат таких зусиль може бути в кращому випадку марним, а в гіршому — шкідливим для репутації громадянської науки [35].

[14] Експерт в ПрО у *Gartner* «*citizen data scientists*» — це дослівно цивільно-громадянські спеціалісти з даних. *Gartner* визначає *Citizen Data Scientist* як «особу, яка створює або генерує моделі, що використовують прогнозну або рекомендаційну аналітику, але чия основна робоча функція перебуває поза цариною статистики та аналітики» — 20 квітня 2018 р [1].

Робота має створити формальні основи для організації викликів і надання спільноті корисних інструкцій. У поєднанні з інструментами викликів, які буде розроблено, це обіцяє стати корисним внеском до здобутків наукової спільноти. Теоретичні фундаментальні внески мають спиратися на експериментальний дизайн і теорію ігор, а також на практичні емпіричні результати, що будуть отримані в результаті аналізу тестування альтернативних протоколів виклику в реальних змаганнях [35].

Орієнтоване на дані автоматизоване самонавчання (*DUL*)

Попри нещодавні успіхи глибокого навчання у застосунках [38], проєктування надійних глибоких нейронних мереж залишається складним завданням і вимагає практичного досвіду та знань, а головне, великої кількості даних [39–41]. В останні кілька років багато зусиль було спрямовано на вибір архітектур і гіперпараметрів, а також на навчання складних глибоких мереж, що відбувалося через множину циклів проб і помилок з використанням багатьох евристик. Пришвидшений попит на вирішення завдань глибинного навчання породжує потребу вдосконалити автоматизацію проєктування цих рішень [42]. Зазвичай, грубе оброблення завдань шляхом подачі до нейронних мереж ще більших наборів даних є привабливим рішенням, чому сприяють успіхи, досягнуті великими компаніями, такими як *Google*[15]. Отже, стає дедалі очевиднішим, що необхідний зсув парадигми в машинному навчанні, а не зосередження уваги на модельно-орієнтованій автоматизації машинного навчання: наприклад пошук нейронної архітектури та оптимізація гіперпараметрів, а також вирішення проблеми отримання якісніших даних. Однак збирання, оброблення, очищення, усунення спотворень та попереднє оброблення даних [1] можуть фактично стати вузьким місцем *AutoML* з інтенсивним застосуванням людської праці. Робота спрямована на вирішення проблеми ефективнішого використання (в автоматичному режимі) вже зібраних даних для отримання значних якісних та кількісних переваг машинного навчання, орієнтованого на дані [35], оскільки глибинні нейронні мережі, як відомо, прагнуть даних.

У зв'язку з цим було визначено кілька напрямів досліджень під алгоритмічним, теоретичним та практичним кутом зору. Зокрема

[15] Директор *Google* з досліджень штучного інтелекту *Peter Norvig* заявив: «У нас немає алгоритмів краще, ніж у когось іншого» і ще «у нас просто більше даних». Однак після скандалів, що викривають необачні рішення, що приймаються машинами, що навчаються, той же *Peter Norvig* переглянув свою думку і сказав: «Більше даних краще, ніж розумні алгоритми, але якісніші дані краще, ніж більше даних».

пропонується оцінити можливість об'єднання даних завдяки включенню кількох наборів даних з різних областей або з однієї області. Це уможливить входження у проблему машинного навчання, орієнтованого на дані, та дасть змогу вирішувати фундаментальні питання, такі як подолання різних типів помилок (випадкових, епістемічних, алеаторичних[16], систематичних тощо) у даних в автоматизованому режимі, що виникають через поганий збір та підготовку даних для аналізу. Окрім того, з метою обґрунтування теоретичних методів фільтрації, повторного балансування або корекції даних, необхідно провести аналіз джерел появи помилок/шуму/упередженості[17] в даних, та здійснити модифікацію поточних концепцій випадкових та епістемічних помилок через відсутність навчальних даних (зовсім відсутніх, або вичерпаних у конкретних областях простору даних), алеаторних помилок (помилка між прогнозованим та фактичним значенням) через внутрішні обмеження даних (наприклад, погане подання даних або погана попередня обробка; погане внесення пропущених значень), систематичних помилок тощо.

Зважаючи на результати окремих праць, є можливість імпортувати методи підвищення якості у глибоке навчання та об'єднати їх зі збільшенням даних [35, 43]. Однак необхідно проаналізувати покращення меж узагальнення[18], внаслідок використання інформації, отриманої за допомоги збільшення даних або застосування інших алгоритмів, орієнтованих на дані (наприклад, використання симетрії в даних, а також двоїстості між вибором моделі та вибором даних), з точки зору залежних від даних меж продуктивності [44], які сприяють поширеному в машинному навчанні спрощеному припущенню, що навчальні та перевірочні або тестові набори мають дотримуватися того ж розподілу.

Під кутом зору практики такі напрями можуть охоплювати: поєднання відбору даних і вибору моделі, вивчення питань, що стосуються упередженості в даних, зрозумілість модельних рішень щодо вибору репрезентативних вибірок даних, *treating multi-view* або *multimodal data* та прилаштування методів до суперкомп'ютерів, щоб надати вченим-прикладникам ефективні комплексні алгоритми навчання, орієнтовані на дані.

Ми очікуємо, що цей напрям досліджень матиме вплив у багатьох сферах застосування, де глибоке навчання довело свою ефективність.

[16] Системні науки та інформатика: збірник доповідей II науково-практичної конференції «Системні науки та інформатика», 4–8 грудня 2023 року, Київ. К., НН ІПСА КПІ ім. Ігоря Сікорського, 2023. 416 с. *http://mmsa.kpi.ua/sites/default/files/systemni_nauky_ta_informatyka_2023.pd*f

[17] Загалом, це схильність сприймати інформацію, що відповідає переконанням дослідників, та ігнорувати протилежну.

[18] Помилка узагальнення оцінює якість машинної моделі, тобто, наскільки добре модель узагальнює дані, яких раніше не було.

Це дасть змогу використовувати набори даних обмеженого розміру або низької якості для ефективного навчання таких моделей або ж об'єднувати дані з багатьох джерел, які збираються в різних умовах і потенційно є погано відкаліброваними. Враховуючи поточний дисбаланс зусиль дослідницької спільноти між підходами, орієнтованими на моделі та дані, ми сприятимемо зміні парадигм і, сподіваємося, зробимо плідний внесок у цій сфері, яка перебуває зараз у зародковому стані [35].

У сфері машинного навчання, зокрема самонавчання та трансферне навчання

Сучасна тенденція у сфері штучного інтелекту значною мірою покладається на системи, здатні навчатися на прикладах, таких як моделі глибокого навчання (*DL*), які є сучасним втіленням штучних нейронних мереж. Попри те, що за останні роки на ринок вийшли численні програми (зокрема, безпілотні автомобілі, автоматизовані помічники, служби бронювання та чат-боти, вдосконалення пошукових систем, рекомендації та реклама, а також застосунки для охорони здоров'я тощо), моделі *DL* все ще важко розгорнути в нових програмах. Зокрема, потрібна величезна кількість навчальних прикладів, години навчання *GPU* та висококваліфіковані інженери для ручного налаштування їхніх архітектур. Цей напрям дослідження сприятиме зменшенню бар'єрів входу до використання моделей *DL* для нових програм, що є кроком до «демократизації *AI*».

Дослідження [35] полягає в розробці нових підходів до трансферного навчання (*TL*), заснованих на модульних архітектурах *DL*. Трансферне навчання охоплює всі методи пришвидшення навчання завдяки використанню попередніх подібних завдань. Наприклад, використання попередньо навчених мереж, цілковито або частково (модульність), є ключовою тактикою *TL*.

У цьому контексті важливим є наступне. Під технічним кутом зору поточні обмеження попереднього навчання охоплюють: (T1) у багатьох доменах немає доступних попередньо навчених мереж через брак великих наборів даних у пов'язаних доменах; (T2) нові архітектури мереж, такі як «графові нейронні мережі» (*GNN*), не легко піддаються попередньому навчанню; (T3) крім простого перенавчання останнього рівня та тонкого налаштування внутрішніх рівнів, засоби повторного використання попередньо навчених мереж у нових контекстах розроблено недостатньо. Ці три проблеми ускладнюють дослідницькі можливості для ефективного використання попередньо набутих знань, симуляторів даних та/або доповнення даних, а також розробки нових алгоритмів і архітектур, які навчаються модульним способом для повторного використання. Наприклад, з точки зору фундаментальних досліджень, модульність і успадкування попере-

дньо підготовлених навчальних модулів у біологічно інспірованих системах навчання є гострою темою *AI* [35].

У цьому контексті пропонується дослідити новий підхід до перенесення навчання, який ми називаємо «Глибоке модульне навчання» [35]. Треба вирішувати проблему навчання великих штучних нейронних мереж, архітектура яких є модульною, та чиї модулі в кінцевому підсумку можна використовувати повторно. Можливим підходом до проблеми буде застосування багаторівневих алгоритмів оптимізації, спрямованих на оптимізацію всієї системи (досягнення мети вищого рівня) за умови, що модулі досягають мети нижчого рівня (повторне використання). Одна з наукових цілей полягатиме в тому, щоб поставити під сумнів гіпотезу про те, що модульність є важливою для систем навчання, оскільки вона прискорює навчання, роблячи можливою ефективну форму передачі навчання, центральну функційність *AI*. Декілька припущень можуть керувати цим дослідженням, зокрема такі:

(*P1*) Принцип економності або «бритва Оккама», втілений у сучасній теорії навчання як «регулярізація», яка стверджує, що «з двох теорій, рівноцінно потужних у відтворенні спостереження, слід віддавати перевагу простішій». Насправді модульні архітектури, що мають ідентичні підмодулі, мають менше настроюваних параметрів, і тому можуть вважатися менш складними, ніж, наприклад, повністю підключені усі модулі.

(*P*2) Гіпотеза вродженості: здатність розв'язувати завдання є комбінацією вроджених і набутих навичок. Чи властиво інтелектуальним системам більшою мірою покладатися на набуті навички, наприклад мову, а не успадковувати їх? Чи правда, що мову можна повністю вивчити «з нуля»?

(*P*3) Індукція, дедукція, концептуалізація та причиновість: чи ґрунтуються інтелектуальні системи навчання на модульності для концептуалізації, засвоєння мови та причиново-наслідкового висновку? Практичне застосування цієї структури на практиці є можливим у таких областях як біомедицина (наприклад, молекулярна токсичність або ефективність), екологія, економетрика, розпізнавання мови, обробка природної мови, обробка зображень або відео тощо [35].

Висновки

На початку нинішнього століття турбота про прозорі рішення та гідну довіру до штучного інтелекту стала причиною того, що концепція відповідального та зрозумілого *AI* почала набувати дедалі більшого значення, допомагаючи зрозуміти, як працює *AI*. Останнім часом напрям розвитку *AI* полягав у прагненні покращити інтерпретованість моделей. Тому дослідження дедалі інтенсивніше рухатимуться в бік гібридних моделей між символічним *AI* з високим рівнем розуміння,

та реальними статистичними глибинними нейронними мережами — рішення, яке забезпечить довіру учасників до надійного запровадження відповідальних систем.

Якісне вирішення зазначених у цьому огляді проблем роботи з даними з сучасним аналітичним інструментарієм їх дослідження із застосуванням множини взаємозалежних технологій, таких як штучний інтелект та машинне навчання, успішне проникнення в їхню суть на рівні отримання знань при високій автоматизації процесу як цілого.

Значення *AutoML* у сфері штучного інтелекту є неоціненним. Він демократизував процес створення та впровадження моделей машинного навчання, зробивши його доступним для ширшої аудиторії з різним рівнем знань у галузі машинного навчання. Автоматизуючи складні та трудомісткі завдання, *AutoML* значно знижує вхідний бар'єр для непрофесіоналів, які бажають використовувати можливості *AI* у своїх застосунках.

AutoML змінив підходи до штучного інтелекту. Його еволюція від традиційного *AutoML* до *AutoDL* та *AutoRL,* його значення, реальне застосування наголошують на ключовій ролі *AutoML* у формуванні майбутнього *AI* та *ML*. Однак надмірна залежність від автоматизації без фундаментального розуміння концепцій машинного навчання може призвести до неоптимального вибору моделювання та обмеженого розуміння даних.

І останнє. Не слід нехтувати застереженням [35, 36] щодо науки про штучний інтелект, яка стоїть зараз перед серйозним викликом внаслідок надміру захопленої реклами наукових досягнень, адресованої переважно за межі наукової спільноти. Неможна вирішувати будь-яке складне завдання без будь-яких інтелектуальних зусиль щодо його дослідження. Це стосується також автоматизації підбору функцій та моделі, що вимагають залучати знання та обмеження предметної галузі у процес їх розроблення, гарантуючи, що остаточна модель відповідатиме вимогам ПрО.

ЛІТЕРАТУРА

1. Oursatyev O. Data Research in Industrial Data Mining Projects in the Big Data Generation Era. *Control Systems and Computers*, 2023, Issue 3, 33–54. [In Ukrainian: Дослідження даних у промислових data-mining-проєктах в епоху генерації великих даних] https://doi.org/10.15407/csc.2023.03.033
2. Elliott T. Intelligence Called?! — Innovation Evangelism. URL: https://timoelliott.com/blog/2017/06/what-is-artificial-intelligence-called.html [Accessed: 11 June 2019].
3. Schlesinger M., Hlavac V. Ten Lectures on Statistical and Structural Pattern Recognition. *Computational Imaging and Vision*. Kluwer Academic Publishers, Dordrecht, Boston London, 2002, Vol. 24, 520 p. https://doi.org/10.1007/978-94-017-3217-8
4. GitHab. Awesome-AutoML-Papers. What is AutoML? URL: https://github.com/hibayesian/awesome-automl-papers [Accessed: 3 March 2024]

5. Guyon I., et al. Design of the 2015 ChaLearn AutoML Challenge. URL: http://haralick.org/ML/automl_ijcnn15.pdf [Accessed: 01 Apr. 2024]
6. Guyon I. et al., Analysis of the AutoML Challenge Series 2015–2018. URL: https://link.springer.com/chapter/10.1007/978-3-030-05318-5_10 [Accessed: 01 Apr. 2024]
7. Domhan T., Springenberg J.T., Hutter F. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. *24th International Conference on Artificial Intelligence IJCAI 2015*, Buenos Aires, Argentina, 3460–3468.
8. Bengio, Y. Gradient-Based Optimization of Hyperparameters. *Neural Computation,* 2000, Vol. 12 (8), 1889–1900. https://doi.org/10.1162/089976600300015187
9. Bergstra, J., Bengio, Y., Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research,* 2012, Vol. 13, 281–305. URL: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf
10. Budjac R., Nikmon M., Schreiber P., Zahradníková B., Janáčová, D. Automated machine learning overview. *Research Papers Faculty of Materials Science and Technology Slovak University of Technology*, 2019, Vol. 27 (45), 107–112. https://doi.org/10.2478/rput-2019-0033
11. Snoek J., Larochelle H., Adams Ryan P. Practical Bayesian Optimization of Machine Learning Algorithms**.** *ArXiv*, 2012, 206.2944, 1–9. https://doi.org/10.48550/arXiv.1206.2944
12. Bengio Yo., Ducharme R., Vincent P., Janvin C. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 2003, Vol. 3, 1137–1155. URL: https://dl.acm.org/ doi/10.5555/944919.944966
13. Hutter F., et al. Sequential Model-Based Optimization for General Algorithm Configuration. *Learning and intelligent optimization: 5th international conference*, LION 5, Rome, Italy, 2011, 1–15. URL: https://ml.informatik.uni-freiburg.de/papers/11-LION5SMAC.pdf
14. Olson R., Moore J. TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. *JMLR: Workshop and Conference Proceedings AutoML Workshop (ICML2016),* 2016, 66–74. URL: http://proceedings.mlr.press/v64/olson_tpot_2016.pdf
15. Li Liam. *Towards Efficient Automated Machine Learning*. 2020, 184 p. URL: https://www.ml.cmu.edu/research/phd-dissertation-pdfs/thesis_li_liam.pdf
16. Thornton C., Hutter F. et al. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. *19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD-2013), 2013,* 847–855. https://doi.org/ 10.1145/2487575.2487629
17. Thornton C., Hutter F. et al. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13), 2013, 847-855. URL: https://www.cs.ubc.ca/~hoos/Publ/ThoEtAl13.pdf [Accessed: 01 Apr. 2024]
18. Feurer M. et al. Practical Automated Machine Learningfor the AutoML Challenge 2018, *ICML 2018 AutoML Workshop*. URL: https://ml.informatik.uni-freiburg.de/papers/18-AUTOML-AutoChallenge.pdf
19. Bergstra J., et al. Algorithms for Hyper-Parameter Optimization. *Part of Advances in Neural Information Processing Systems (NIPS 2011), 2011, Vol. 24, 1-9*. URL: https://papers.nips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
20. Sun L., et al. Automatic Neural Network Search Method for Open Set Recognition. *2019 IEEE Int. Conf. Image Process,* Beijing, China Fujitsu Laboratories Ltd., Kawasaki, Japan, 2019. https://doi.org/10.1109/ICIP.2019.8803605
21. Jamieson K., Talwalkar A. Non-stochastic Best Arm Identification and Hyperparameter Optimization, Feb 2015, 1–13. URL: https://arxiv.org/pdf/1502.07943.pdf

22. Li L., Jamieson K., Rostamizadeh A., Talwalkar A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research,* 2018б Vol. 18, 1-52. URL: https://arxiv.org/pdf/1603.06560.pdf
23. Li L., Jamieson K., Rostamizadeh A., Gonina E., et al. A System for Massively Parallel Hyperparameter Tuning. *3-rd MLSys Conference*, Austin, TX, USA, 2020, 1–17. URL: https://arxiv.org/pdf/1810.05934.pdf
24. Jin Y. et al. *Multi-Objective Machine Learning.* Berlin Heidelberg: Springer, 2006. Vol. 16., 657 p. URL: https://link.springer.com/content/pdf/ bfm%3A978-3-540-33019-6%2F1.pdf
25. Jin Y., Sendhoff B. Pareto-Based Multiobjective Machine Learning: An Overview and Case Studies. *IEEE transactions on systems, man, and cybernetics – part C: applications and reviews,* 2008, Vol. 38 (3), 397–415. https://doi.org/10.1109/TSMCC.2008.919172
26. Parmentier L., Nicol O., Jourdan L., Kessaci M. TPOT-SH: A Faster Optimization Algorithm to Solve the AutoML Problem on Large Datasets. *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 2019, 471–478. https://doi.org/10.1109/ICTAI.2019.00072
27. Oliveira M. 3 Reasons Why AutoML Won't Replace Data Scientists Yet. 2019. URL: https://www.kdnuggets.com/2019/03/why-automl-wont-replace-data-scientists.html
28. Adopting a data science process framework. URL: https://microsoft.github.io/azureml-ops-accelerator/1-MLOpsFoundation/2-SkillsRolesAndResponsibilities/1-AdoptingDSProcess.html [Accessed 6 Jun. 2025]
29. Berthold M. Principles of Guided Analytics. 2018. URL: https://www.knime.com/blog/principles-of-guided-analytics
30. Tamagnini P., Schmid S., Dietz C. How to Automate Machine Learning. 2019. URL: https://www.knime.com/blog/how-to-automate-machine-learning
31. LeDell E. The different flavors of AutoML. 2018. URL: https://www.h2o.ai/blog/the-different-flavors-of-automl/
32. Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks. *NIPS,* 2012, 1097–1105.
33. Domhan T., Springenberg J.T., Hutter F. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. *24th International Conference on Artificial Intelligence [IJCAI 2015]*, 2015, 3460–3468.
34. Piatetsky Gregory, KDnuggets on December 10, 2021 in 2022 Predictions. Main 2021 Developments and Key 2022 Trends in AI, Data Science, Machine Learning Technology – https://www.kdnuggets.com/2021/12/trends-ai-data-science-ml-technology.html
35. Guyon I. HUMANIA. Artificial Intelligence for All – https://guyon.chalearn.org/projects/humania
36. Gritsenko V.I., Schlesinger M.I. Interaction of pattern recognition machine thinking and learning problems. International Scientific Technical Journal «Problems of Control and Informatics», 3, 108-136. URL: http://jnas.nbuv.gov.ua/article/UJRN-0001259001 [In Russian].
37. AI Glossary. Automated Machine Learning Automl, December 24, 2023. URL: https://www.larksuite.com/en_us/topics/ai-glossary/automated-machine-learning-automl
38. LeCun Y., BengioY., Hinton G. Deep learning. *Nature*, 2015, Vol. 521 (7553), 436-444. https://doi.org/10.1038/nature14539
39. Alom M. Z. et al. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 2019, Vol. 8 (3), 292-357. https://doi.org/10.3390/electronics8030292
40. Deng J. et al. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

41. Sun C. et al. Revisiting unreasonable effectiveness of data in deep learning era. *IEEE international conference on computer vision*, 2017, 843–852. https://doi.org/10.1109/ICCV.2017.97
42. Assunçao F. et al. Evolving the topology of large scale deep neural networks. *European Conference on Genetic Programming,* Springer, 2018, 19–34. https://doi.org/10.1007/978-3-319-77553-1_2
43. Cubuk, Ekin D. et al. Autoaugment: Learning augmentation policies from data. *CoRR*, 2018, Vol. 1, 1-14. https://doi.org/10.48550/arXiv.1805.09501
44. Vapnik V. *Statistical Learning Theory.* Wiley-Interscience publication, Wiley, 1998, 736 p. URL: https://books.google.fr/books?id=GowoAQAAMAAJ [Accessed: 01 Apr. 2024]

Отримано 10.06.2024

O.A. Oursatyev, PhD (Engineering), Senior Research Associate,
Institute of Information Technologies and Systems of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/ 0009-0009-8323-0525
aleksei@irtc.org.ua

O.Ye. Volkov, PhD (Engsneering), Senior Researcher, Director,
Institute of Information Technologies and Systems of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0000-0002-5418-6723
alexvolk@ukr.net

V.H. Tkalya, PhD Student,
Institute of Information Technologies and Systems of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0009-0004-7870-3256
advv0207@gmail.com

AUTOMATED MACHINE LEARNING.
STATE AND PROSPECTS DEVELOPMENT

Introduction. The task of automated machine learning is considered as one of the tasks of artificial intelligence for the automation of the end-to-end process of designing machine learning pipelines. In this case, human presence should be significantly reduced or, preferably, completely excluded. The article provides an overview of information services for the automation of the end-to-end process of applying and improving the efficiency of machine learning methods — modern approaches, methods, technologies in this area, competing due to the development of intelligent means of information processing, and sometimes even surpassing machine learning experts.

The purpose of the paper is to familiarize specialists in need of data skills with tasks related to data processing, advanced mathematical apparatus and world-class tools for obtaining information and knowledge.

Methods. Machine learning methods are studied. machine learning is considered as an artificial intelligence-based solution for automating the end-to-end process of applying machine learning, i.e. designing machine learning pipelines — a sequence of steps that transform raw data into a machine model suitable for deployment in practical use. The direction of further development of artificial intelligence and automated machine learning and its development trends are considered.

Results. Automated machine learning is considered as an AI-based solution for the needs of automating the end-to-end process of applying machine learning to design machine learning pipelines. This is a sequence of steps that transform raw data into a machine model adopted for deployment in practical use. The direction

of further development of artificial intelligence and automated machine learning, as well as its development trends, are considered.

Conclusion. The conducted research has shown the importance of automated machine learning in the field of artificial intelligence. This approach has democratized the process of creating and implementing machine learning models, making it accessible to a wider audience with different levels of knowledge in the field of machine learning. By automating complex and time-consuming tasks, automated machine learning significantly reduces the entry barrier for non-professionals who want to use the capabilities of artificial intelligence in their applications.

***Keywords**: AutoML, Artificial Intelligence for All, data-driven Artificial Intelligence, Deep Learning, DL, Reinforcement Learning, RL. Transfer Learning, TL.*

https://doi.org/10.15407/intechsys.2025.02.034
UDC 004.8 + 004.032.26

B.YE. RYTSAR, DSc (Engineering), Professor,
Department of Radioelectronic Technologies of Information Systems,
Institute of Information and Communication Technologies and Electronic Engineering,
L'viv Polytechnic National University
12, Bandera str., L'viv, 79013, Ukraine
http://orcid.org/0000-0002-2929-2954
bohdanrytsar@gmail.com

NEW METHOD FOR GENERATING TEST CODES TO DETECT MULTIPLE STUCK-AT-FAULTS IN COMBINATIONAL CIRCUITS[1]. PART 1

The article is devoted to a new method of generating test codes to detect multiple damages in digital combinational circuits, which is based on the artificial introduction of nonessential variables and the application of the procedure of q-partition of minterms of a given function. Due to the use of a numerical set-theoretic approach, the proposed method differs from the known ones in a relatively simpler practical implementation to detect stuck-at-faults (0/1) type both at one point and at several points simultaneously of the circuit under study.

Keywords: *ccombinational circuit, single and multiple stuck-at-faults (0/1) type damage, q-partition of minterms, nonessential variables, vector of test codes.*

Introduction

In [1, 2], a method for detecting stuck-at-faults (0/1) at any single point of a combinational circuit is proposed by determining (generating) vectors of test codes. This method is based on a set-theoretical approach, using simple procedures and operations on binary minterms of a given function [1] or system minterms of a given system of functions [2] describing the operation of the studied circuit.

[1] This article is a development of a topic published in the author's previous articles [1, 2].

Cite: Rytsar B.Ye. New Method for Generating Test Codes to Detect Multiple Stuck-at-faults in Combinational Circuits. Part 1. *Information Technologies and Systems*, Київ, 2025, Том 2 (2), 34–54. https://doi.org/10.15407/intechsys.2025.02.034

© Видавець ВД «Академперіодика» НАН України, 2025. Стаття опублікована на умовах відкритого доступу за ліцензією CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

In the practice of microcircuit design, there are often situations where stuck-at-faults (0/1) type damages can occur not only at a single point in the circuit, but also at several different points in the circuit simultaneously, which may be interconnected. Such damage is more difficult to detect due to the random nature of their occurrence, and in addition, their binary values may differ. Determining them by pointwise search is not a convincing proof of the reliability of the obtained result, which consequently reduces the reliability of the design process.

Known methods for detecting multiple stuck-at-faults (0/1) at several points in a combinational circuit [3–12] are mainly based on modeling single errors to find the minimum number of test vector sets. However, this approach is quite cumbersome, as it requires the use of additional procedures, since the problem of detecting such faults is NP-complete with a complexity of at least $O(2^n)$ for n errors. To solve this complexity, in [6] it was proposed to use the SAT Solver algorithm to generate test sets in combinational circuits with several errors, that also complicates the problem.

Among the approaches to the diagnostics of digital circuit design for detecting multiple errors, in addition to modeling, symbolic methods are also known [3, 4]. However, they do not rely on test vectors, but usually use ordered binary solution diagrams (OBBD) to characterize the necessary and sufficient conditions of a potential error source as a Boolean function. Although symbolic approaches are more accurate than modeling approaches, the strong increase in the complexity of their implementation imposes practical limitations for detecting the above-mentioned errors during the design of digital circuits.

This article is devoted to a new method of generating test code vectors to detect stuck-at-faults (0/1) type damages both at one point and at several different points of a combinational circuit simultaneously. The method is based on the idea of artificially introducing into the Boolean space a completely specified function $f(x_1, x_2, ..., x_i)$, which describes the operation of the studied circuit, a certain number (but not n) of a nonessential variables and applying the q-partition procedure of the minterms of the perfect STF Y^1 function f [12, 13]. Unlike the known methods of generating test codes, the proposed approach is relatively simpler in terms of practical implementation.

Theoretical Part

It is generally known that a Boolean function $f(x_1, x_2, ..., x_i)$ significantly depends on the variable x_i if there are such values $\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n$ of the variables $x_1, ..., x_{i-1}, x_{i+1}, ..., x_n$ that

$$f(\sigma_1, ..., \sigma_{i-1}, 0, \sigma_{i+1}, ..., \sigma_n) \neq f(\sigma_1, ..., \sigma_{i-1}, 1, \sigma_{i+1}, ..., \sigma_n). \quad (1)$$

If condition (1) is not met, then the variable x_i is nonessential for the function f.

Instead, to transform an essential variable x_i of the function f into a nonessential one, which is assumed by the proposed method, it is necessary to replace the value σ_i with the opposite one in the domain of definition of the function f, that is, in the set 2^n of sets $\mathbf{E}_2^n = \{\langle \sigma_1,...,\sigma_i,...,\sigma_n \rangle : \sigma_i \in \{0,1\}\}$ of its variables. Then, the Boolean space of the function f will be divided into two equivalent subspaces, and the newly created function will differ from the original one in values on certain sets of variables. Accordingly, if r "nonessential" variables are simultaneously introduced, the Boolean space will be divided into 2^r equivalent subspaces, and those sets of variables, on which the values of the function f have changed, are precisely the desired vectors of test codes, which will allow to recognize the type and the location of stuck-at-faults (0/1) damage in the studied scheme.

Table 1 illustrates the changes in the values $f_0, f_1, ..., f_7$ (3rd column) of the conditional function $f(x_1, x_2, x_3)$ in various intervention situations: when one "nonessential" variable is introduced, for example x_1 (4th and 5th columns) and x_2 (6th and 7th columns), and when the same variables are introduced simultaneously as "nonessentials" ones (8th, 9th, 10th and 11th columns).

In Table 2, this case is illustrated by an example of a function $f(x_1, x_2, x_3)$ that has a perfect STF $Y^1 = \{(001),(101),(110),(111)\}^1$ (*see* Example 1 in [1]). For the same "nonessential" variables x_1 and x_2, the elements of the pseudo-perfect STF differ from the elements of the perfect STF Y^1, which are highlighted here in bold. For example (*see* the 4th column), a function $f_{x_1/0}$ corrupted by an "nonessential" variable $x_1/0$ has a pseudoperfect STF $Y^1_{x_1/0} = \{(001),(101)\}^1$, where two elements on the sets of variables (110) and (111), which are vectors of test codes, are highlighted in bold.

In Table 2, the numbers in the 9th (log. 0) and 11th (log. 1) columns, highlighted in square frames, show that these test code vectors cannot be determined separately for the "nonessential" variables x_1 and x_2. Since such vectors are not present in the "damaged" functions $f_{x_1/\sim}$ and $f_{x_2/\sim}$,

Table 1

"10"	$x_1\,x_2\,x_3$	f	$f_{x_1/0}$	$f_{x_1/1}$	$f_{x_2/0}$	$f_{x_2/1}$	$f_{x_1x_2/00}$	$f_{x_1x_2/01}$	$f_{x_1x_2/10}$	$f_{x_1x_2/11}$
0	0 0 0	f_0	f_0	f_4	f_0	f_2	f_0	f_2	f_4	f_6
1	0 0 1	f_1	f_1	f_5	f_1	f_3	f_1	f_3	f_5	f_7
2	0 1 0	f_2	f_2	f_6	f_0	f_2	f_0	f_2	f_4	f_6
3	0 1 1	f_3	f_3	f_7	f_1	f_3	f_1	f_3	f_5	f_7
4	1 0 0	f_4	f_0	f_4	f_4	f_6	f_0	f_2	f_4	f_6
5	1 0 1	f_5	f_1	f_5	f_5	f_7	f_1	f_3	f_5	f_7
6	1 1 0	f_6	f_2	f_6	f_4	f_6	f_0	f_2	f_4	f_6
7	1 1 1	f_7	f_3	f_7	f_5	f_7	f_1	f_3	f_5	f_7

this confirms the need to search for possible multiple stuck-at-faults (0/1) in the studied circuit, which is important for diagnostics.

To determine (generate) vectors of test codes that will detect stuck-at-faults (0/1) type damage both at one point and at several different points of the combinational circuit simultaneously, it is convenient to apply the *q*-partition procedure of minterm of the perfect STF Y^1 of the Boolean function *f*, the essence of which is as follows.

As noted in [12, 13], the *q*-partition procedure (operator $\overset{P^q}{\Rightarrow}$) of an *n*-position binary minterm $m_i = (\sigma_1\sigma_2\cdots\sigma_n)$, $\sigma_i \in \{0,1\}$ of the function $f(x_1, x_2, ..., x_n)$ is a sampling of *q* positions ($q \in \{1,2,...,(n-1)\}$, $q < n$) from the minterm m_i. This procedure is performed by applying of a mask of literals $\{l_{\alpha_1} l_{\alpha_2} \cdots l_{\alpha_{n-q}} | l_{\beta_1} l_{\beta_2} \cdots l_{\beta_q}\}$ to the minterm m_i, formed by partitioning into two classes — the $(n-q)$-class and the *q*-class — the conjunction of literals $l_1 l_2 \cdots l_n$, $l_i \in \{\overline{x}_i, x_i\}$, where the dash | is the symbol of the *q*-partition.

The *q*-partition procedure of minterms $m_1, m_2, ..., m_k$ of a perfect STF Y^1 of an arbitrary completely specified function *f* forms a set of partitioned minterms:

$$Y^1 = \{m_1, m_2, ..., m_k\}^1 \overset{P^q}{\Rightarrow} \{l_{\alpha_1} l_{\alpha_2} \cdots l_{\alpha_{n-q}} | l_{\beta_1} l_{\beta_2} \cdots l_{\beta_q}\} =$$
$$= \{m_{\alpha_1}^{n-q} | m_{\beta_1}^q, m_{\alpha_2}^{n-q} | m_{\beta_2}^q, ..., m_{\alpha_k}^{n-q} | m_{\beta_k}^q\}^1 \Rightarrow, \quad (2)$$

where $m_i^{n-q} | m_i^q$ is the partitioned *i*-th minterm consisting of two subminterms: $(n-q)$-class — m_i^{n-q} and *q*-class — m_i^q; in binary format — $m_i^{n-q} | m_i^q = (\sigma_{\alpha_i} \cdots \sigma_{\alpha_{n-q}}) | (\sigma_{\beta_i} \cdots \sigma_{\beta_q})$. For example, the 2-partition of a binary minterm (10010) for mask of literal $\{l_2 l_3 l_4 | l_1 l_5\}$ looks like $(10010) \overset{P^2}{\Rightarrow} \{l_2 l_3 l_4 | l_1 l_5\} = (001|10)$.

As a result of the *q*-partition procedure of the binary minterms of the perfect STF Y^1 (2), a set of the partitioned of binary minterms is formed like that:

$$\Rightarrow \{(\sigma_{\alpha_1} \cdots \sigma_{\alpha_{n-q}}) | (\sigma_{\beta_1} \cdots \sigma_{\beta_q}), (\sigma_{\alpha_2} \cdots \sigma_{\alpha_{n-q}}) | (\sigma_{\beta_2} \cdots \sigma_{\beta_q}), ...,$$
$$(\sigma_{\alpha_k} \cdots \sigma_{\alpha_{n-q}}) | (\sigma_{\beta_k} \cdots \sigma_{\beta_q})\}^1 \overset{|\cup}{\Rightarrow}$$

Table 2

«10»	$x_1\, x_2\, x_3$	f	$f_{x_1/0}$	$f_{x_1/1}$	$f_{x_2/0}$	$f_{x_2/1}$	$f_{x_1x_2/00}$	$f_{x_1x_2/01}$	$f_{x_1x_2/10}$	$f_{x_1x_2/11}$
0	0 0 0	0	0	0	0	0	0	0	0	**1**
1	0 0 1	1	1	1	1	**0**	1	**0**	1	1
2	0 1 0	0	0	**1**	0	0	0	0	0	**1**
3	0 1 1	0	0	**1**	**1**	0	**1**	0	**1**	**1**
4	1 0 0	0	0	0	0	**1**	0	0	0	**1**
5	1 0 1	1	1	1	1	1	1	**0**	1	1
6	1 1 0	1	**0**	1	**0**	1	**0**	**0**	**0**	1
7	1 1 1	1	**0**	1	1	1	1	**0**	1	1

and, it can be further simplified first by the "rightwise union" procedure (operator $\overset{|\cup}{\Rightarrow}$), and if necessary, then by the "leftwise union" procedure (operator $\overset{\cup|}{\Rightarrow}$) [14].

Let the binary subminterms of the $(n-q)$-class reflect the values of "nonessential" variables and their number $r=1,2,...,(n-1)$. Then the values of this subminterms in the $(n-q)$-class must form a complete set $\mathbf{E}_2^r$. Thus, the values of the "damaged" function f for each case of a one "nonessential" variable $(r=1)$ or "nonessential" variables $(r=2,3,...,(n-1))$ will form sets of pseudo-perfect STFs, accordingly.

Let us consider the proposed approach in more detail.

Let $q=(n-1)$. This means that one "nonessential" variable x_i is chosen, for the existence of which it is necessary to have in the $(n-q)$-class (here in the 1-class of the partition) two values of the subminterms $\mathbf{E}_2^1=\{0,1\}$, namely:

$$Y^1=\{m_1,m_2,...,m_k\}^1 \overset{P^{n-1}}{\Rightarrow}\{l_{\alpha_i}\,|\,l_{\beta_1}l_{\beta_2}\cdots l_{\beta_{n-1}}\}\overset{|\cup}{\Rightarrow}\{(0\,|\,N_0^{n-1}),(1\,|\,N_1^{n-1})\}^1, \qquad (3)$$

where N_0^{n-1}, N_1^{n-1} are sets of subminterms of the $(n-1)$-class formed by the $(n-1)$-partition procedure and the "rightwise union" operation.

Based on (3), two sets can be formed: a pseudoperfect STF $Y_{x_i/0}^1$ — when a "nonessential" variable x_i caused "damage" s-a-0 $(x_i/0)$ and a pseudoperfect STF $Y_{x_i/1}^1$ — when a "nonessential" variable x_i caused "damage" s-a-1 $(x_i/1)$. To determine the test codes, it is necessary to first perform the concatenation procedure on the partitioned minterms (3) $(0\,|\,N_0^{n-1})$ and $(1\,|\,N_1^{n-1})$, and then, to combine the resulting sets in a polynomial format with the given minterms, i.e. to form $\{Y_{x_i/0}^1,Y^1\}^{\oplus}$ and $\{Y_{x_i/1}^1,Y^1\}^{\oplus}$. As a result, the desired test code vectors are "filtered" into separate sets $C_{x_i/0}^1,C_{x_i/0}^0$ and $C_{x_i/1}^1,C_{x_i/0}^0$ [2]:

$$\begin{Bmatrix} Y_{x_i/0}^1 \\ Y^1 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} C_{x_i/0}^1 \\ C_{x_i/0}^0 \end{Bmatrix}, \qquad (4)$$

$$\begin{Bmatrix} Y_{x_i/1}^1 \\ Y^1 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} C_{x_i/1}^1 \\ C_{x_i/1}^0 \end{Bmatrix}. \qquad (5)$$

Let $q=(n-2)$. This means that two "nonessential" variables x_i and x_j are chosen, for the existence of which it is necessary to have four values of subminterms in the 2-partition class — $\mathbf{E}_2^2=\{00,01,10,11\}$, namely:

$$Y^1=\{m_1,m_2,...,m_k\}^1 \overset{P^{n-2}}{\Rightarrow}\{l_{\alpha_i}l_{\alpha_j}\,|\,l_{\beta_1}l_{\beta_2}\cdots l_{\beta_{n-2}}\}\Rightarrow$$

$$\Rightarrow\{(00\,|\,N_{00}^{n-2}),(01\,|\,N_{01}^{n-2}),(10\,|\,N_{10}^{n-2}),(11\,|\,N_{11}^{n-2})\}^1, \qquad (6)$$

where N_{00}^{n-2}, N_{01}^{n-2}, N_{10}^{n-2}, N_{11}^{n-2} are sets of subminterms of the $(n-2)$-class.

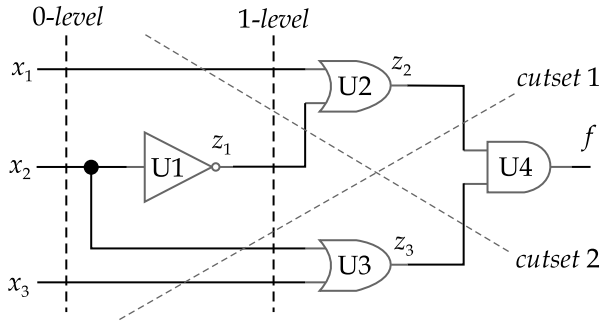

Fig. 1

From the set (6) similarly to (3) it is possible to form four pseudoperfect STFs $Y^1_{x_i x_j/00}$, $Y^1_{x_i x_j/01}$, $Y^1_{x_i x_j/10}$, $Y^1_{x_i x_j/11}$ and on their basis to obtain vectors of test codes for determining possible stuck-at-faults (0/1) type damages simultaneously at two points of the combinational circuit.

It is obvious that by using the same algorithm, it is possible to generate sets of test code vectors for a larger number of introduced "nonessential" variables of the function *f*.

It should be noted, that in practice, when studying (diagnostics) combinational circuits for verification of possible stuck-at-faults (0/1) type damages, a situation may occur when $|Y^1| > |Y^0|$. In this case, due to the less cumbersome (simpler) implementation of the necessary procedures and operations, it is advisable to use a set Y^0 instead Y^1 (*see* example 2 below). Then, after the operation of combining pseudoperfect STFs $Y^0_{x_i/0}$ and $Y^0_{x_i/1}$ in inverse polynomial format (using the example of introducing an "nonessential" variable x_i) with a perfect STF Y^0, the desired sets of test code vectors $C^0_{x_i/0}$, $C^1_{x_i/0}$ and $C^0_{x_i/1}$, $C^1_{x_i/1}$ can be defined as follows:

$$\begin{Bmatrix} Y^0_{x_i/0} \\ Y^0 \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} C^0_{x_i/0} \\ C^1_{x_i/0} \end{Bmatrix}, \tag{7}$$

$$\begin{Bmatrix} Y^0_{x_i/1} \\ Y^0 \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} C^0_{x_i/1} \\ C^1_{x_i/1} \end{Bmatrix}. \tag{8}$$

Practical Part

Let us illustrate the proposed method with examples.

Example 1. Determine the vectors of test codes for detecting multiple stuck-at-faults (0/1) type damages at the 0- and 1-levels and at the intersection points of the cutset 1 and 2 of the logic circuit shown in Fig. 1. (*borrowing from* [1])

Solution. The scheme in Fig. 1 implements the function $f(x_1, x_2, x_3)$ given by Table 2.

Let us first define the vectors of test codes for the case of introducing one "nonessential" variable x_1.

At the 0-level of the scheme, as a result of the 2-partition of the minterms of the perfect STF Y^1 of the function f, we obtain sets of partitioned minterms, which after the "rightwise union" procedure will simplify to the following form:

$$Y^1 = \{(001),(101),(110),(111)\}^1 \overset{P^2}{\Rightarrow} \begin{Bmatrix} l_1 \,|\, l_2 l_3 \\ l_2 \,|\, l_1 l_3 \\ l_3 \,|\, l_1 l_2 \end{Bmatrix} = \begin{cases} \{(0\,|\,01),(1\,|\,01,10,11)\}^1 \\ \{(0\,|\,01,11),(1\,|\,10,11)\}^1 \\ \{(0\,|\,11),(1\,|\,00,10,11)\}^1 \end{cases}.$$

For each mask of literals from the two formed subsets, we obtain pseudoperfect STFs $Y^1_{x_i/0}$ and $Y^1_{x_i/1}$ after the concatenation operation (operator $\overset{con}{\Rightarrow}$). Each of these sets, similarly to ([1] see table 2), after combining in polynomial format their minterms with the given minterms of the perfect STF Y^1 of the function f and simplifying by eliminating identical pairs, will form the desired vectors of test codes (here and further we will highlight the value of the "nonessential" variable in bold):

$$Y^1_{x_1/0} = \{(\mathbf{0},1)\,|\,01\}^1 \overset{con}{\Rightarrow} \{(001),(101)\}^1 \Rightarrow \begin{Bmatrix} \cancel{001,101} \\ \cancel{001,101},110,111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{110,111\}^0 \end{cases},$$

$$Y^1_{x_1/1} = \{(0,\mathbf{1})\,|\,01,10,11\}^1 \overset{con}{\Rightarrow} \{(001),(010),(011),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \cancel{001},010,011,\cancel{101,110,111} \\ \cancel{001,101,110,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{010,011\}^1 \\ \{\varnothing\}^0 \end{cases};$$

$$Y^1_{x_2/0} = \{(\mathbf{0},1)\,|\,(01,11)\}^1 \overset{con}{\Rightarrow} \{(001),(011),(101),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \cancel{001},011,\cancel{101,111} \\ \cancel{001,101},110,\cancel{111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{011\}^1 \\ \{110\}^0 \end{cases},$$

$$Y^1_{x_2/1} = \{(0,\mathbf{1})\,|\,10,11\}^1 \overset{con}{\Rightarrow} \{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 100,\cancel{101,110,111} \\ 001,\cancel{101,110,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{100\}^1 \\ \{001\}^0 \end{cases}.$$

In Table 2, columns 4–7 correspond to the pseudoperfect STFs $Y^1_{x_1/0}$, $Y^1_{x_1/1}$, $Y^1_{x_2/0}$ and $Y^1_{x_2/1}$.

$$Y^1_{x_3/0} = \{(\mathbf{0},1)\,|\,(11)\}^1 \overset{con}{\Rightarrow} \{(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \cancel{110,111} \\ 001,101,\cancel{110,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{001,101\}^0 \end{cases},$$

$$Y^1_{x_3/1} = \{(0,\mathbf{1})\,|\,00,10,11\}^1 \overset{con}{\Rightarrow} \{(000),(001),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix} 000,\cancel{001},100,\cancel{101,110,111} \\ \cancel{001,101,110,111} \end{matrix}\right\}^{\oplus} \Rightarrow \left\{\begin{matrix} \{000,100\}^1 \\ \{\varnothing\}^0 \end{matrix}\right. .$$

Note that the resulting test code vectors are the same as in the article [1].

Let us apply the 1-partition of the given minterms to determine the test codes for the case of simultaneous stuck-at-faults (0/1) damage at two points on the 0-level of the circuit:

$$Y^1 = \{(001),(101),(110),(111)\}^1 \overset{P^1}{\Rightarrow} \left\{\begin{matrix} l_1 l_2 \,|\, l_3 \\ l_1 l_3 \,|\, l_2 \\ l_2 l_3 \,|\, l_1 \end{matrix}\right\} =$$

$$= \left\{\begin{matrix} \{(00,10\,|\,1),(01\,|\,\varnothing),(11\,|\,0,1)\}^1 \\ \{(00\,|\,\varnothing),(01\,|\,0),(10\,|\,1),(11\,|\,0,1)\}^1 \\ \{(00\,|\,\varnothing),(01\,|\,0,1),(10,11\,|\,1)\}^1 \end{matrix}\right. ,$$

$$Y^1_{x_1x_2/00,10} = \{(\mathbf{00},01,\mathbf{10},11)\,|\,1\}^1 \overset{con}{\Rightarrow} \{(001),(011),(101),(111)\}^1 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix} \cancel{001},011,\cancel{101,111} \\ \cancel{001,101},110,\cancel{111} \end{matrix}\right\}^{\oplus} \Rightarrow \left\{\begin{matrix} \{011\}^1 \\ \{110\}^0 \end{matrix}\right. ,$$

$$Y^1_{x_1x_2/01} = \{(00,\mathbf{01},10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow} \{\varnothing\}^1 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix} \varnothing \\ 001,101,110,111 \end{matrix}\right\}^{\oplus} \Rightarrow \left\{\begin{matrix} \{\varnothing\}^1 \\ \{001,101,110,111\}^0 \end{matrix}\right. ,$$

$$Y^1_{x_1x_2/11} = \{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow} \{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix} 000,001,010,011,100,101,110,111 \\ \cancel{001,101,110,111} \end{matrix}\right\}^{\oplus} \Rightarrow \left\{\begin{matrix} \{000,010,011,100\}^1 \\ \{\varnothing\}^0 \end{matrix}\right. .$$

In Table 2, columns 8–11 correspond to pseudoperfect STFs $Y^1_{x_1x_2/00}$, $Y^1_{x_1x_2/01}$, $Y^1_{x_1x_2/10}$ and $Y^1_{x_1x_2/11}$, and the numbers, highlighted in bold, are vectors of test codes for cases of simultaneous stuck-at-faults (0/1) damage at the two specified points of the circuit.

For other pairs of "nonessential" variables at the 0-level of the circuit, we obtain the following result:

$$Y^1_{x_1x_3/00}=\{(\mathbf{00},01,10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}\varnothing\\ 001,101,110,111\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{\varnothing\}^1\\ \{001,101,110,111\}^0\end{Bmatrix},$$

$$Y^1_{x_1x_3/01}=\{(00,\mathbf{01},10,11)\,|\,0\}^1 \overset{con}{\Rightarrow}\{(000),(001),(100),(101)\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}000,\cancel{001},100,\cancel{101}\\ \cancel{001,101},110,111\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{000,100\}^1\\ \{110,111\}^0\end{Bmatrix},$$

$$Y^1_{x_1x_3/10}=\{(00,01,\mathbf{10},11)\,|\,1\}^1 \overset{con}{\Rightarrow}\{(010),(011),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}010,011,\cancel{110,111}\\ 001,101,\cancel{110,111}\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{010,011\}^1\\ \{001,101\}^0\end{Bmatrix},$$

$$Y^1_{x_1x_3/11}=\{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$
$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}000,001,010,011,100,101,110,111\\ \cancel{001,101,110,111}\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{000,010,011,100\}^1\\ \{\varnothing\}^0\end{Bmatrix};$$

$$Y^1_{x_2x_3/00}=\{(\mathbf{00},01,10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}\varnothing\\ 001,101,110,111\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{\varnothing\}^1\\ \{001,101,110,111\}^0\end{Bmatrix},$$

$$Y^1_{x_2x_3/01}=\{(00,\mathbf{01},10,11)\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$
$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}000,001,010,011,100,101,110,111\\ \cancel{001,101,110,111}\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{000,010,011,100\}^1\\ \{\varnothing\}^0\end{Bmatrix},$$

$$Y^1_{x_2x_3/10,11}=\{(00,01,\mathbf{10},\mathbf{11})\,|\,1\}^1 \overset{con}{\Rightarrow}\{(100),(101),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow\begin{Bmatrix}100,\cancel{101,110,111}\\ 001,\cancel{101,110,111}\end{Bmatrix}^{\oplus}\Rightarrow\begin{Bmatrix}\{100\}^1\\ \{001\}^0\end{Bmatrix}.$$

At the 1-level of the scheme we have a function $f(x_1,z_1,x_3)$, which has a perfect STF $Y^1=\{3,4,5,7\}^1$. After performing the 2-partition procedure over its minterms, we obtain the same test code vectors as in ([1], *see* table 4):

$$Y^1=\{(011),(100),(101),(111)\}^1 \overset{p^2}{\Rightarrow}\begin{Bmatrix} l_1\,|\,l_2l_3 \\ l_2\,|\,l_1l_3 \\ l_3\,|\,l_1l_2 \end{Bmatrix}=\begin{Bmatrix} \{(0\,|\,11),(1\,|\,00,01,11)\}^1 \\ \{(0\,|\,10,11),(1\,|\,01,11)\}^1 \\ \{(0\,|\,10),(1\,|\,01,10,11)\}^1 \end{Bmatrix},$$

$$Y^1_{x_1/0}=\{(\mathbf{0},1)\,|\,11\}^1 \overset{con}{\Rightarrow}\{(011),(111)\}^1 \Rightarrow \begin{Bmatrix} \sout{011,111} \\ \sout{011},100,101,\sout{111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{100,101\}^0 \end{Bmatrix},$$

$$Y^1_{x_1/1}=\{(0,\mathbf{1})\,|\,00,01,11\}^1 \overset{con}{\Rightarrow}\{(000),(001),(011),(100),(101),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix} 000,001,011,100,101,111 \\ \sout{011,100,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001\}^1 \\ \{\varnothing\}^0 \end{Bmatrix};$$

$$Y^1_{z_1/0} \Rightarrow \begin{Bmatrix} \{110\}^1 \\ \{011\}^0 \end{Bmatrix},\ Y^1_{z_1/1} \Rightarrow \begin{Bmatrix} \{001\}^1 \\ \{100\}^0 \end{Bmatrix},\ Y^1_{x_3/0} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{011,111\}^0 \end{Bmatrix},\ Y^1_{x_3/1} \Rightarrow \begin{Bmatrix} \{010,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix}.$$

In case of simultaneous stuck-at-faults (0/1) damage at two points on the 1-level of the circuit, after performing the 1-partition procedure of the minterms of the function $f(x_1,z_1,x_3)$, we obtain the following result:

$$Y^1=\{(011),(100),(101),(111)\}^1 \overset{p^1}{\Rightarrow}\begin{Bmatrix} l_1l_2\,|\,l_3 \\ l_1l_3\,|\,l_2 \\ l_2l_3\,|\,l_1 \end{Bmatrix}=\begin{Bmatrix} \{(00\,|\,\varnothing),(01,11\,|\,1),(10\,|\,0,1)\}^1 \\ \{(00\,|\,\varnothing),(01\,|\,1),(10\,|\,0),(11\,|\,0,1)\}^1; \\ \{(00,01\,|\,1),(10\,|\,\varnothing),(11\,|\,0,1)\}^1 \end{Bmatrix}$$

$$Y^1_{x_1z_1/00}=\{\varnothing\}^1 \Rightarrow \begin{Bmatrix} \varnothing \\ 011,100,101,111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{011,100,101,111\}^0 \end{Bmatrix},$$

$$Y^1_{x_1z_1/01,11}=(00,\mathbf{01},10,\mathbf{11})\,|\,1\}^1 \overset{con}{\Rightarrow}\{(001),(011),(101),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix} 001,\sout{011,101,111} \\ \sout{011},100,\sout{101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{001\}^1 \\ \{100\}^0 \end{Bmatrix},$$

$$Y^1_{x_1z_1/10}=\{(00,01,\mathbf{10},11)\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$
$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix} 000,001,010,011,100,101,110,111 \\ \sout{011,100,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001,010,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix};$$

$$Y^1_{x_1x_3/00}=\{\varnothing\}^1 \Rightarrow \begin{Bmatrix} \varnothing \\ 011,100,101,111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{011,100,101,111\}^0 \end{Bmatrix},$$

$$Y^1_{x_1x_3/01} = (00,\mathbf{01},10,11)\,|\,1\}^1 \overset{con}{\Rightarrow} \{(010),(011),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 010,\cancel{011},110,\cancel{111} \\ \cancel{011},100,101,\cancel{111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{010,110\}^1 \\ \{100,101\}^0 \end{Bmatrix}$$

$$Y^1_{x_1x_3/10} = (00,01,\mathbf{10},11)\,|\,0\}^1 \overset{con}{\Rightarrow} \{(000),(001),(100),(101)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 000,001,\cancel{100,101} \\ 011,\cancel{100,101},111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001\}^1 \\ \{011,111\}^0 \end{Bmatrix},$$

$$Y^1_{x_1x_3/11} = \{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow} \{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 000,001,010,\cancel{011,100,101},110,\cancel{111} \\ \cancel{011,100,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001,010,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix};$$

$$Y^1_{z_1x_3/00,01} = (\mathbf{00},\mathbf{01},10,11)\,|\,1\}^1 \overset{con}{\Rightarrow} \{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \cancel{100,101},110,\cancel{111} \\ 011,\cancel{100,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{110\}^1 \\ \{011\}^0 \end{Bmatrix},$$

$$Y^1_{z_1x_3/10} = (00,01,\mathbf{10},11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow} \{\varnothing\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \varnothing \\ 011,100,101,111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{011,100,101,111\}^0 \end{Bmatrix},$$

$$Y^1_{z_1x_3/11} = \{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow} \{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 000,001,010,\cancel{011,100,101},110,\cancel{111} \\ \cancel{011,100,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001,010,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix}.$$

Table 3

2-partition	0-level			1-level		
Stuck-at-fault	$x_1/\sim$	$x_2/\sim$	$x_3/\sim$	$x_1/\sim$	$z_1\sim$	$x_3\sim$
s-a-0	$\begin{pmatrix}110\\111\end{pmatrix}^0$	$\begin{matrix}(011)^1\\(110)^0\end{matrix}$	$\begin{pmatrix}001\\101\end{pmatrix}^0$	$\begin{pmatrix}100\\101\end{pmatrix}^0$	$\begin{matrix}(110)^1\\(011)^0\end{matrix}$	$\begin{pmatrix}011\\111\end{pmatrix}^0$
s-a-1	$\begin{pmatrix}010\\011\end{pmatrix}^1$	$\begin{matrix}(100)^1\\(001)^0\end{matrix}$	$\begin{pmatrix}000\\100\end{pmatrix}^1$	$\begin{pmatrix}000\\001\end{pmatrix}^1$	$\begin{matrix}(001)^1\\(100)^0\end{matrix}$	$\begin{pmatrix}010\\110\end{pmatrix}^1$

Table 3 contains vectors of test codes at 0- and 1-levels of the circuit for the case of stuck-at-faults (0/1) type damage at one point, and Table 4 for the case of multiple damage at two points of the circuit.

For the cutset 2 we have the function $f(x_1, z_1, z_3) = x_1 z_3 \vee z_1 z_3 \Rightarrow$ $\Rightarrow \{(1-1),(-11)\}^1$ that has a perfect STF $Y^1 = \{3,5,7\}^1$ ($Y^0 = \{0,1,2,4,6\}^0$). After performing a 2-partition of its minterms, we define the vectors of test codes:

$$Y^1 = \{(011),(101),(111)\}^1 \overset{P^2}{\Rightarrow} \begin{Bmatrix} l_1 \mid l_2 l_3 \\ l_2 \mid l_1 l_3 \\ l_3 \mid l_1 l_2 \end{Bmatrix} = \begin{cases} \{(0 \mid 11),(1 \mid 01,11)\}^1 \\ \{(0 \mid 11),(1 \mid 01,11)\}^1 \\ \{(0 \mid \varnothing),(1 \mid 01,10,11)\}^1 \end{cases};$$

$$Y^1_{x_1/0} = \{(\mathbf{0},1) \mid 11\}^1 \overset{con}{\Rightarrow} \{(011),(111)\}^1 \Rightarrow \begin{Bmatrix} \cancel{011},\cancel{111} \\ \cancel{011},101,\cancel{111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{101\}^0 \end{cases}$$

$$Y^1_{x_1/1} = \{(0,\mathbf{1}) \mid 01,11\}^1 \overset{con}{\Rightarrow} \{(001),(011),(101),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 001,\cancel{011},\cancel{101},\cancel{111} \\ \cancel{011},\cancel{101},\cancel{111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{cases} \{001\}^1 \\ \{\varnothing\}^0 \end{cases},$$

Table 4

1-partition	0-level			1-level		
Stuck-at-fault	$x_1x_2/\sim\sim$	$x_1x_3/\sim\sim$	$x_2x_3/\sim\sim$	$x_1z_1/\sim\sim$	$x_1x_3/\sim\sim$	$z_1x_3/\sim\sim$
s-a-00	$(011)^1$ $(110)^0$	$\begin{pmatrix}001\\101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}001\\101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\100\\101\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\100\\101\\111\end{pmatrix}^0$	$(110)^1$ $(011)^0$
s-a-01	$\begin{pmatrix}001\\101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}000\\100\end{pmatrix}^1$ $\begin{pmatrix}110\\111\end{pmatrix}^0$	$\begin{pmatrix}000\\010\\011\\100\end{pmatrix}^1$	$(001)^1$ $(100)^0$	$\begin{pmatrix}010\\110\end{pmatrix}^1$ $\begin{pmatrix}100\\101\end{pmatrix}^0$	$(110)^1$ $(011)^0$
s-a-10	$(011)^1$ $(110)^0$	$\begin{pmatrix}010\\011\end{pmatrix}^1$ $\begin{pmatrix}001\\101\end{pmatrix}^0$	$(100)^1$ $(001)^0$	$\begin{pmatrix}000\\001\\010\\110\end{pmatrix}^1$	$\begin{pmatrix}000\\001\end{pmatrix}^1$ $\begin{pmatrix}011\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\100\\101\\111\end{pmatrix}^0$
s-a-11	$\begin{pmatrix}000\\010\\011\\100\end{pmatrix}^1$	$\begin{pmatrix}000\\010\\011\\100\end{pmatrix}^1$	$(100)^1$ $(001)^0$	$(001)^1$ $(100)^0$	$\begin{pmatrix}000\\001\\010\\110\end{pmatrix}^1$	$\begin{pmatrix}000\\001\\010\\110\end{pmatrix}^1$

$$Y^1_{z_1/0}=\{(\mathbf{0},1)\,|\,11\}^1 \overset{con}{\Rightarrow}\{(101),(111)\}^1 \Rightarrow \begin{Bmatrix}\sout{101,111}\\ 011,\sout{101,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{\varnothing\}^1\\ \{011\}^0\end{Bmatrix},$$

$$Y^1_{z_1/1}=\{(0,\mathbf{1})\,|\,01,11\}^1 \overset{con}{\Rightarrow}\{(001),(011),(101),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix}001,\sout{011,101,111}\\ \sout{011,101,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{001\}^1\\ \{\varnothing\}^0\end{Bmatrix},$$

$$Y^1_{z_3/0}=\{(\mathbf{0},1)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\ 011,101,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{\varnothing\}^1\\ \{011,101,111\}^0\end{Bmatrix};$$

$$Y^1_{z_3/1}=\{(0,\mathbf{1})\,|\,01,10,11\}^1 \overset{con}{\Rightarrow}\{(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix}010,\sout{011},100,\sout{101},110,\sout{111}\\ \sout{011,101,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{010,100,110\}^1\\ \{\varnothing\}^0\end{Bmatrix}.$$

The vectors of test codes with simultaneous stuck-at-faults (0/1) damage at two points of the circuit can be obtained after performing the 1-partition of the minterms of the function $f(x_1,z_1,z_3)$:

$$Y^1=\{(011),(101),(111)\}^1 \overset{P^1}{\Rightarrow}\begin{Bmatrix}l_1l_2\,|\,l_3\\ l_1l_3\,|\,l_2\\ l_2l_3\,|\,l_1\end{Bmatrix}=\begin{Bmatrix}\{(00\,|\,\varnothing),(01,10,11\,|\,1)\}^1\\ \{(00,10\,|\,\varnothing),(01\,|\,1),(11\,|\,0,1)\}^1\\ \{(00,10\,|\,\varnothing),(01\,|\,1),(11\,|\,0,1)\}^1\end{Bmatrix};$$

$$Y^1_{x_1z_1/00}=\{(\mathbf{00},01,10,11)\,|\,\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\ 011,101,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{\varnothing\}^1\\ \{011,101,111\}^0\end{Bmatrix},$$

$$Y^1_{x_1z_1/01,10,11}=\{(00,\mathbf{01},\mathbf{10},\mathbf{11})\,|\,1\}^1 \overset{con}{\Rightarrow}\{(001),(011),(101),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix}001,\sout{011,101,111}\\ \sout{011,101,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{001\}^1\\ \{\varnothing\}^0\end{Bmatrix};$$

$$Y^1_{x_1z_3/00,10}=\{(\mathbf{00},01,\mathbf{10},11)\,|\,\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\ 011,101,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{\varnothing\}^1\\ \{011,101,111\}^0\end{Bmatrix};$$

$$Y^1_{x_1z_3/01}=\{(00,\mathbf{01},10,11)\,|\,1\}^1 \overset{con}{\Rightarrow}\{(010),(011),(110),(111)\}^1 \Rightarrow$$
$$\Rightarrow \begin{Bmatrix}010,\sout{011},110,\sout{111}\\ \sout{011},101,\sout{111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix}\{010,110\}^1\\ \{101\}^0\end{Bmatrix},$$

$$Y^1_{x_1z_3/11} = \{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 000,001,010,\cancel{011},100,\cancel{101},110,\cancel{111} \\ \cancel{011,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001,010,100,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix},$$

$$Y^1_{z_2z_3/00,10} = \{(\mathbf{00},01,\mathbf{10},11)\,|\,\varnothing\}^1 \Rightarrow \begin{Bmatrix} \varnothing \\ 011,101,111 \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{\varnothing\}^1 \\ \{011,101,111\}^0 \end{Bmatrix};$$

$$Y^1_{z_2z_3/01} = \{(00,\mathbf{01},10,11)\,|\,1\}^1 \overset{con}{\Rightarrow}\{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 100,\cancel{101},110,\cancel{111} \\ 011,\cancel{101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{100,110\}^1 \\ \{011\}^0 \end{Bmatrix},$$

$$Y^1_{z_2z_3/11} = \{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 000,001,010,\cancel{011},100,\cancel{101},110,\cancel{111} \\ \cancel{011,101,111} \end{Bmatrix}^{\oplus} \Rightarrow \begin{Bmatrix} \{000,001,010,100,110\}^1 \\ \{\varnothing\}^0 \end{Bmatrix}.$$

The test code vectors obtained for the cutset 2 for damage at one point of the circuit are placed in Table 5, and for simultaneous damage at two points of the circuit — in Table 6.

For the cutset 1 of the scheme in Fig. 1, we have the function $f(z_2, x_2, x_3) = z_2x_2 \vee z_2x_3 \Rightarrow \{(11-),(1-1)\}^1$ that has

Table 5

Stuck-at-fault	2-partition		
	$x_1/\sim$	$z_1/\sim$	$z_3/\sim$
s-a-0	$(101)^0$	$(101)^0$	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$
s-a-1	$(001)^1$	$(001)^1$	$\begin{pmatrix}010\\100\\110\end{pmatrix}^1$

Table 6

Stuck-at-fault	1-partition		
	$x_1z_1/\sim\sim$	$x_1z_3/\sim\sim$	$z_1z_3/\sim\sim$
s-a-00	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$
s-a-01	$(001)^1$	$\begin{pmatrix}010\\110\end{pmatrix}^1$ $(101)^0$	$\begin{pmatrix}100\\110\end{pmatrix}^1$ $(011)^0$
s-a-10	$(001)^1$	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$	$\begin{pmatrix}011\\101\\111\end{pmatrix}^0$
s-a-11	$(001)^1$	$\begin{pmatrix}000\\001\\010\\100\\110\end{pmatrix}^1$	$\begin{pmatrix}000\\001\\010\\100\\110\end{pmatrix}^1$

a perfect STF $Y^1=\{5,6,7\}^1$ ($Y^0=\{0,1,2,3,4\}^0$). After performing the 2-partition procedure of minterms, we determine the vectors of test codes for the case of damage at the intersection points of section 1:

$$Y^1=\{(101),(110),(111)\}^1 \overset{P^2}{\Rightarrow} \begin{bmatrix} l_1 \mid l_2 l_3 \\ l_2 \mid l_1 l_3 \\ l_3 \mid l_1 l_2 \end{bmatrix} = \begin{cases} \{(0\mid\varnothing),(1\mid 01,10,11)\}^1 \\ \{(0\mid 11),(1\mid 10,11)\}^1 \\ \{(0\mid 11),(1\mid 10,11)\}^1 \end{cases};$$

$$Y^1_{z_2/0}=\{(\mathbf{0},1)\mid\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow \begin{bmatrix} \varnothing \\ 101,110,111 \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{101,110,111\}^{0'} \end{cases}$$

$$Y^1_{z_2/1}=\{(0,\mathbf{1})\mid(01,10,11)\}^1 \overset{con}{\Rightarrow}\{(001),(010),(011),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{bmatrix} 001,010,011,\sout{101,110,111} \\ \sout{101,110,111} \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{001,010,011\}^1 \\ \{\varnothing\}^0 \end{cases};$$

$$Y^1_{x_2/0}=\{(\mathbf{0},1)\mid 11\}^1 \overset{con}{\Rightarrow}\{(101),(111)\}^1 \Rightarrow \begin{bmatrix} \sout{101,111} \\ \sout{101},110,\sout{111} \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{110\}^{0'} \end{cases}$$

$$Y^1_{x_2/1}=\{(0,\mathbf{1})\mid(10,11)\}^1 \overset{con}{\Rightarrow}\{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{bmatrix} 100,\sout{101,110,111} \\ \sout{101,110,111} \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{100\}^1 \\ \{\varnothing\}^0 \end{cases};$$

$$Y^1_{x_3/0}=\{(\mathbf{0},1)\mid 11\}^1 \overset{con}{\Rightarrow}\{(110),(111)\}^1 \Rightarrow \begin{bmatrix} \sout{110,111} \\ 101,\sout{110,111} \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{\varnothing\}^1 \\ \{101\}^{0'} \end{cases}$$

$$Y^1_{x_3/1}=\{(0,\mathbf{1})\mid(10,11)\}^1 \overset{con}{\Rightarrow}\{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{bmatrix} 100,\sout{101,110,111} \\ \sout{101,110,111} \end{bmatrix}^{\oplus} \Rightarrow \begin{cases} \{100\}^1 \\ \{\varnothing\}^0 \end{cases}.$$

After performing 1-partition of the minterms of the function $f(z_2,x_2,x_3)$, we obtain vectors of test codes for detecting damage at two points of the circuit:

$$Y^1=\{(101),(110),(111)\}^1 \overset{P^1}{\Rightarrow} \begin{bmatrix} l_1 l_2 \mid l_3 \\ l_1 l_3 \mid l_2 \\ l_2 l_3 \mid l_1 \end{bmatrix} = \begin{cases} \{(00,01\mid\varnothing),(10\mid 1),(11\mid 0,1)\}^1 \\ \{(00,01\mid\varnothing),(10\mid 1),(11\mid 0,1)\}^1 \\ \{(00\mid\varnothing),(01,10,11\mid 1)\}^1 \end{cases};$$

$$Y^1_{z_2x_2/00,01}=\{(\mathbf{00},\mathbf{01},10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\101,110,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{\varnothing\}^1\\\{101,110,111\}^0\end{matrix},$$

$$Y^1_{z_2x_2/10}=\{(00,01,\mathbf{10},11)\,|\,1\}^1 \overset{con}{\Rightarrow}\{(001),(011),(101),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix}001,011,\cancel{101,111}\\\cancel{101},110,\cancel{111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{001,011\}^1\\\{110\}^0\end{matrix},$$

$$Y^1_{z_2x_2/11}=\{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix}000,001,010,011,100,\cancel{101,110,111}\\\cancel{101,110,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{000,001,010,011,100\}^1\\\{\varnothing\}^0\end{matrix};$$

$$Y^1_{z_2x_3/00,01}=\{(\mathbf{00},\mathbf{01},10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\101,110,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{\varnothing\}^1\\\{101,110,111\}^0\end{matrix},$$

$$Y^1_{z_2x_3/10}=\{(00,01,\mathbf{10},11)\,|\,1\}^1 \overset{con}{\Rightarrow}\{(010),(011),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix}010,011,\cancel{110,111}\\101,\cancel{110},\cancel{111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{010,011\}^1\\\{101\}^0\end{matrix},$$

$$Y^1_{z_2x_3/11}=\{(00,01,10,\mathbf{11})\,|\,0,1\}^1 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(000),(001),(010),(011),(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix}000,001,010,011,100,\cancel{101,110,111}\\\cancel{101,110,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{000,001,010,011,100\}^1\\\{\varnothing\}^0\end{matrix};$$

$$Y^1_{x_2x_3/00}=\{(\mathbf{00},01,10,11)\,|\,\varnothing\}^1 \overset{con}{\Rightarrow}\{\varnothing\}^1 \Rightarrow \begin{Bmatrix}\varnothing\\101,110,111\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{\varnothing\}^1\\\{101,110,111\}^0\end{matrix},$$

$$Y^1_{x_2x_3/11}=\{(00,\mathbf{01},\mathbf{10},\mathbf{11})\,|\,1\}^1 \overset{con}{\Rightarrow}\{(100),(101),(110),(111)\}^1 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix}100,\cancel{101,110,111}\\\cancel{101,110,111}\end{Bmatrix}^{\oplus} \Rightarrow \begin{matrix}\{100\}^1\\\{\varnothing\}^0\end{matrix}.$$

The obtained vectors of test codes for the cutset 1 for damage at one point of the circuit are placed in Table 7, and for simultaneous damage at two points of the circuit — in Table 8.

Example 2. The proposed method is to determine the vectors of test codes to detect single and simultaneous stuck-at-fault (0/1) damage at points *B* and *C* of the circuit in Fig. 2 (*borrowed from* [4], p. 594, Fig. 3).

Solution. The given circuit is described by the function $f(a,b,c,d)=$, $=\overline{\overline{abc}\;\overline{bcd}}$ which has a perfect STF $Y^1=\{2,3,4,5,6,7,10,11,12,13,14\}^1$ or $Y^0=\{0,1,8,9,15\}^0$. Since $|Y^1|>|Y^0|$, then we will solve the example taking into account the perfect STF Y^0.

We will determine the vectors of test codes for single damage at points *B* and *C* of the circuit, applying the 3-partition of the minterms of the perfect STF Y^0 with respect to the variables *b* and *c*:

Table 7

Stuck-at-fault	2-partition		
	$z_2/\sim$	$x_2/\sim$	$x_3/\sim$
s-a-0	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$	$(110)^0$	$(101)^0$
s-a-1	$\begin{pmatrix}001\\010\\011\end{pmatrix}^1$	$(100)^1$	$(100)^1$

Table 8

Stuck-at-fault	1-partition		
	$z_2x_2/\sim\sim 101$	$z_2x_3/\sim\sim$	$x_2x_3/\sim\sim$
s-a-00	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$
s-a-01	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$	$\begin{pmatrix}101\\110\\111\end{pmatrix}^0$	$(100)^1$
s-a-10	$\begin{pmatrix}001\\011\end{pmatrix}^1, (110)^0$	$\begin{pmatrix}010\\011\end{pmatrix}^1, (101)^0$	$(100)^1$
s-a-11	$\begin{pmatrix}000\\001\\010\\011\\100\end{pmatrix}^1$	$\begin{pmatrix}000\\001\\010\\011\\100\end{pmatrix}^1$	$(100)^1$

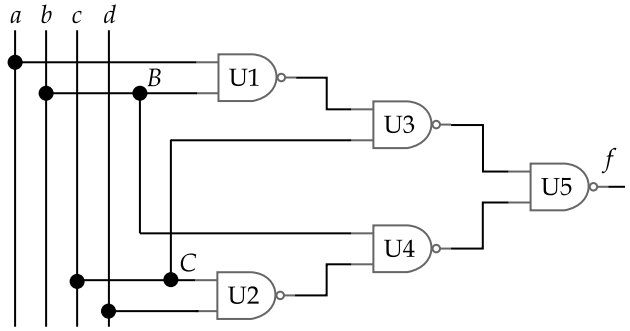

Fig. 2

$$Y^0 = \{(0000),(0001),(1000),(1001),(1111)\}^0 \overset{P^3}{\Rightarrow} \begin{Bmatrix} l_2 \,|\, l_1 l_3 l_4 \\ l_3 \,|\, l_1 l_2 l_4 \end{Bmatrix} =$$

$$= \begin{Bmatrix} \{(0\,|\,000),(0\,|\,001),(0\,|\,100),(0\,|\,101),(1\,|\,111)\}^0 \\ \{(0\,|\,000),(0\,|\,001),(0\,|\,100),(0\,|\,101),(1\,|\,111)\}^0 \end{Bmatrix} \overset{|\cup}{\Rightarrow}$$

$$\overset{|\cup}{\Rightarrow} \begin{Bmatrix} \{0\,|\,(000,001,100,101),1\,|\,111\}^0 \\ \{0\,|\,(000,001,100,101),1\,|\,111\}^0 \end{Bmatrix}.$$

Considering these sets and procedures (7) and (8), we obtain the desired test codes:

$$Y^0_{b/0} = \{(\mathbf{0},1)\,|\,(000,001,100,101)\}^0 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(0000),(0001),(0100),(0101),(1000),(1001),(1100),(1101)\}^0 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \sout{0000,0001},0100,0101,\sout{1000,1001},1100,1101 \\ \sout{0000,0001,1000,1001},1111 \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} (0100,0101,1100,1101)^0 \\ (1111)^1 \end{Bmatrix},$$

$$Y^0_{b/1} = \{(0,\mathbf{1})\,|\,111\}^0 \Rightarrow \{(1011),(1111)\}^0 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 1011,\sout{1111} \\ 0000,0001,1000,1001,\sout{1111} \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} (1011)^0 \\ (0000,0001,1000,1001)^1 \end{Bmatrix};$$

$$Y^0_{c/0} = \{(\mathbf{0},1)\,|\,(000,001,100,101)\}^0 \overset{con}{\Rightarrow}$$

$$\overset{con}{\Rightarrow}\{(0000),(0001),(0010),(0011),(1000),(1001),(1010),(1011)\}^0 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} \sout{0000,0001},0010,0011,\sout{1000,1001},1010,1011 \\ \sout{0000,0001,1000,1001},1111 \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} (0010,0011,1010,1011)^0 \\ (1111)^1 \end{Bmatrix},$$

$$Y^0_{c/1} = \{(0,\mathbf{1})\,|\,111\}^0 \Rightarrow \{(1101),(1111)\}^0 \Rightarrow$$

$$\Rightarrow \begin{Bmatrix} 1101,\sout{1111} \\ 0000,0001,1000,1001,\sout{1111} \end{Bmatrix}^{\bar{\oplus}} \Rightarrow \begin{Bmatrix} (1101)^0 \\ (0000,0001,1000,1001)^1 \end{Bmatrix}.$$

The test codes in the case of simultaneous stuck-at-fault (0/1) damage at points *B* and *C* of the circuit are obtained after applying the 2-partition of minterms and procedures (7) and (8):

$$Y^0 = \{(0000),(0001),(1000),(1001),(1111)\}^0 \overset{P^2}{\Rightarrow} \{l_2 l_3 \,|\, l_1 l_4\} =$$

$$= \{(00\,|\,00),(00\,|\,01),(00\,|\,10),(00\,|\,11),(11\,|\,11)\}^0 \overset{|\cup}{\Rightarrow}$$

$$\overset{|\cup}{\Rightarrow} \{(00\,|\,(00,01,10,11)),(01\,|\,\varnothing),(10\,|\,\varnothing),(11\,|\,11)\}^0 \overset{\cup|}{\Rightarrow}$$

$$\overset{\cup|}{\Rightarrow} \{(00\,|\,(00,01,10,11),((01,10)\,|\,\varnothing),(11\,|\,11)\}^0$$

$$Y^0_{bc/00} = \{(\mathbf{00},01,10,11)\,|\,(00,01,10,11)\}^0 \Rightarrow \left\{\begin{matrix}\mathbf{E}_2^4 \\ 0000,0001,1000,1001,1111\end{matrix}\right\}^{\bar{\oplus}} \Rightarrow$$

$$\Rightarrow \begin{cases}(0010,0011,0100,0101,0110,0111,1010,1011,1100,1101,1110)^0 \\ \{\varnothing\}^1\end{cases};$$

$$Y^0_{bc/01,10} = \{(00,\mathbf{01},\mathbf{10},11)\,|\,\varnothing\}^0 \Rightarrow \{\varnothing\}^0 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix}\varnothing \\ 0000,0001,1000,1001,1111\end{matrix}\right\}^{\bar{\oplus}} \Rightarrow \begin{cases}\{\varnothing\}^0 \\ (0000,0001,1000,1001,1111)^{1'}\end{cases}$$

$$Y^0_{bc/11} = \{(00,01,10,\mathbf{11})\,|\,11\}^0 \overset{con}{\Rightarrow} \{1001,1011,1101,1111\}^0 \Rightarrow$$

$$\Rightarrow \left\{\begin{matrix}0000,0001,1000,\sout{1001,1111} \\ \sout{1001},1011,1101,\sout{1111}\end{matrix}\right\}^{\bar{\oplus}} \Rightarrow \begin{cases}(1011,1101)^0 \\ (0000,0001,1000)^1\end{cases}.$$

Table 9

«10»	*a b c d*	*f*	$f_{b/\sim}$		$f_{c/\sim}$		$f_{bc/\sim}$			
			b/0	*b*/1	*c*/0	*c*/1	*bc*/00	*bc*/01	*bc*/10	*bc*/11
0	0 0 0 0	0	0	**1**	0	**1**	0	**1**	**1**	**1**
1	0 0 0 1	0	0	**1**	0	**1**	0	**1**	**1**	**1**
2	0 0 1 0	1	1	1	**0**	1	**0**	1	1	1
3	0 0 1 1	1	1	1	**0**	1	**0**	1	1	1
4	0 1 0 0	1	**0**	1	1	1	**0**	1	1	1
5	0 1 0 1	1	**0**	1	1	1	**0**	1	1	1
6	0 1 1 0	1	1	1	1	1	**0**	1	1	1
7	0 1 1 1	1	1	1	1	1	**0**	1	1	1
8	1 0 0 0	0	0	**1**	0	**1**	0	**1**	**1**	**1**
9	1 0 0 1	0	0	**1**	0	**1**	0	**1**	**1**	0
10	1 0 1 0	1	1	1	**0**	1	**0**	1	1	1
11	1 0 1 1	1	1	**0**	**0**	1	**0**	1	1	**0**
12	1 1 0 0	1	**0**	1	1	1	**0**	1	1	1
13	1 1 0 1	1	**0**	1	1	**0**	**0**	1	1	**0**
14	1 1 1 0	1	1	1	1	1	**0**	1	1	1
15	1 1 1 1	0	**1**	0	**1**	0	0	**1**	**1**	0

The obtained test codes for single and simultaneously both "nonessential" variables b and c of the function $f(a,b,c,d)$, which is implemented by the scheme in Fig. 2, are shown in Table 9 (*see* the bold numbers in the columns for $f_{b/\sim}$, $f_{c/\sim}$ and $f_{bc/\sim\sim}$).

Conclusion

A new method for generating test codes for detecting multiple stuck-at-faults in digital combinational circuits is proposed, which is based on the artificial introduction of nonessential variables and the application of the procedure of q-partition of minterms of a given function describing the operation of the studied circuit. Due to the application of a numerical set-theoretic approach to the execution of operations and procedures, the proposed method, compared to the known ones, is characterized by a relatively simpler practical implementation of detecting the mentioned faults both at any point and at several points of the studied circuit simultaneously.

REFERENCES

1. Rytsar B. Ye. A Simple Stuck-at-faults Detection Method in Digital Combinational Circuits. *Control Systems and Computers*, 2023, Vol 1 (301), 5–17. https://doi.org/10.15407/csc.2023.01.005
2. Rytsar B. Ye. A Simple Stuck-at-faults Detection Method in Digital Combinational Circuits. II. *Control Systems and Computers*, 2024, Vol 1 (301), 3–17. https://doi.org/10.15407/csc.2024.01.003
3. Azam Beg. A framework for finding minimal test vectors for stuck-at-faults. *3rd Inter. Conf. ICICT'2009 – Aug 2009*,Karachi, Pakistan, 259–262.
4. Jong Chang Kim, Vishvani D. Agraval, Kewal K. Saluja. Multiple Faults: Modeling, Simulation and Test. *7th ASPDAC and 15th Int'l Conf. on VLSI Design*, 2002, pp. 592–597.
5. Jutman A., Ubar R. Design error diagnostic in digital circuits with stuck-at-fault model. *Microelectronics Reliability*, 28 Febr. 2000, Vol. 40 (2), 307–320.
6. Koundinya P., Reddy S., Deepak V., Rutwesh K., Deshpande A.. Test Set Generation for Multiple Faults in Boolean Systems using SAT Solver. *12th Inter. Conf. ICCCNT* – 06-08 July 2021, 329–340.
7. Parag K. Lala. *An Introduction to Logic Circuit Testing*. Morgan & Claypool, 2009, 111 p.
8. Kohavi Z., Jha *N*. Switching and Finite Automata Theory. *Cambridge University Press*, 2010, 206—250.
9. Leila Malihi, Razieh Malihi. Single stuck-at-faults detection using test generation vector and deep stacked-sparse-auto-encoder. *SN Applied Sciences*, 2020, Vol. 2 (1715). https://doi.org/10.1007/s42452-020-03460-0
10. Fujiwara H. Logic testing and design for testability. In *Comp. Syst. Series. Cambridge*, MA: Mass. Inst. Tech, 1986.
11. Karkouri Y, Aboulhamid. Multiple Stuck-at Fault in Logic Circuits. URL: http://www.iro.umontreal.ca/~aboulham/pdfs_sources/KCCVLSI.pdf
12. Rytsar B.Ye. Dekompozytsija bulovykh funktsij metodom *q*-rozbyttja. *UsiM*, 1999, Issue 6, 29–42. [In Ukrainian: Рицар Б.Є. Декомпозиція булових функцій методом *q*-розбиття. 1]
13. Rytsar B.Ye. *Teoretyko-mnozhynni optymizatsijni metody lohikovoho syntezu kombinatsijnykh merezh*. Dissertation DSc (Engineering), Lviv, 2004, 138–142. [In Ukrainian: Рицар Б.Є. Теоретико-множинні оптимізаційні методи логікового синтезу комбінаційних мереж: дис. доктора техн.наук. Львів, 2004,138–142.]

14. Rytsar B., Romanowski P., Shvay A. Set-theoretical Constructions of Boolean Functions and theirs Applications in Logic Synthesis. *Fundamenta Informatica*, 2010, Vol. 99, №3, 339–354.

Received 08.03.2025

Б.Є. Рицар, д-р техн. наук, професор,
Національний університет «Львівська політехніка»,
Інститут інформаційно-комунікаційних технологій та електронної техніки,
вул. Степана Бандери, 12, Львів, 79013, Україна
https://orcid.org/0000-0002-2929-2954
bohdanrytsar@gmail.com

НОВИЙ МЕТОД ГЕНЕРУВАННЯ ТЕСТОВИХ КОДІВ
ДЛЯ ВИЯВЛЕННЯ МНОЖИННИХ ПОШКОДЖЕНЬ *STUCK-AT-FAULTS*
У КОМБІНАЦІЙНИХ СХЕМАХ. ЧАСТИНА 1

Вступ. Важливим розділом логікового проєктування цифрових пристроїв є технічна діагностика, в межах якої розробляються методи перевірки технічного стану пристроїв для забезпечення надійності їх роботи. Виявити несправність у схемі пристрою можна послідовністю контрольних тестів (генеруванням векторів тестових кодів) на її входах та спостереження результатів на її виходах. На практиці проєктування мікросхем часто трапляються ситуації, коли пошкодження типу *stuck-at-faults* (0/1) можуть виникати як в одній точці схеми, так і в кількох різних взаємопов'язаних точках схеми одночасно, які складно виявляти. Відомі методи діагностики множинних пошкоджень такого типу, які ґрунтуються на моделюванні одиночних помилок та символьних методах, не забезпечують переконливі докази достовірності результату, що знижує надійність процесу проєктування.

Мета статті. Запропонувати метод генерування векторів тестових кодів для виявлення як одиночних, так і множинних пошкоджень типу *stuck-at-faults* (0/1) у комбінаційних пристроях, який порівняно з відомими методами може забезпечувати достовірні результати з допомогою реалізації простих операцій і процедур.

Методи. Запропонований метод генерування тестових кодів ґрунтується на числовому теоретико-множинному підході до реалізації всіх операцій і процедур, а саме: штучного впровадження у буловий простір повної функції $f(x_1, x_2, ..., x_i)$, що описує роботу схеми досліджуваного комбінаційного пристрою, одної або більше (до $n-1$) неістотних змінних та застосуванні процедури q-розбиття мінтермів досконалої ТМФ Y^1 функції f.

Результати. За допомогою згенерованих запропонованим методом векторів тестових кодів можна визначити в схемі пристрою як місце пошкодження, так і тип одиночного та множинного *stuck-at-faults* (0/1) пошкодження. Показано застосування процедури q-розбиття двійкових мінтермів, на основі якої реалізується впровадження одної неістотної змінної та формування псевдодосконалої ТМФ $Y^1_{x_i/\sim}$ функції f для визначення одиночних пошкоджень, а також більше (від двох до $n-1$) неістотних змінних та формування відповідних псевдодосконалих ТМФ функції f для визначення множинних пошкоджень.

Висновки. Завдяки застосуванню числового теоретико-множинного підходу для виконання операцій і процедур пропонований метод, порівняно з відомими, відрізняється відносно простішою практичною реалізацією виявлення згаданих несправностей як в будь якій одній точці, так і в одночасно кількох точках досліджуваної схеми. Зазначені переваги методу ілюструють наведені в статті приклади визначення можливих пошкоджень у реальних схемах комбінаційних пристроїв.

Ключові слова: *комбінаційна схема, одиничне та множинне пошкодження stuck-at-faults (0/1), q-розбиття мінтермів, неістотні змінні, вектор тестових кодів.*

https://doi.org/10.15407/intechsys.2025.02.055
УДК 004.7

О.Б. ГОДЛЕВСЬКИЙ, канд. фіз.-мат. наук, старш. наук. співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0009-0005-8067-077X
a.godl49@gmail.com

М.К. МОРОХОВЕЦЬ, канд. фіз.-мат. наук, старш. наук. співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0009-0008-2191-8677
marina.morokhovets@gmail.com

Н.М. ЩОГОЛЕВА, науковий співроб.,
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Акад. Глушкова, 40, м. Київ, 03187, Україна
https://orcid.org/0009-0004-4700-7426
natashch2904@gmail.com

АНАЛІЗ ПОШИРЕНИХ ТИПІВ МЕРЕЖЕВИХ АТАК ТА ЧИННИКИ, ЩО УМОЖЛИВЛЮЮТЬ ЇХ УСПІШНЕ ЗДІЙСНЕННЯ

Подано огляд поширених типів мережевих атак. Для кожного типу атак описано джерела та об'єкти атаки, мету та результати атаки, дії, що виконуються для досягнення мети атаки. Для кожного типу атак зазначено, що уможливлює здійснення атак. На базі знайдених відомостей зібрано та класифіковано чинники, що уможливлюють успішне здійснення атак. Окреслено напрями посилення стійкості до мережевих атак.

Ключові слова: *комп'ютерна мережа, кібератака, інформаційна безпека, протоколи інтернету.*

Вступ

Об'єкти мережевого середовища кіберпростору — дані, що передаються в мережі та зберігаються у сховищах, під'єднаних до мережі, прикладні програми, сервери та інші пристрої — загрожені через

Cite: Годлевський О.Б., Мороховець М.К., Щоголева Н.М. Аналіз поширених типів мережевих атак та чинники, що уможливлюють їх успішне здійснення. *Information Technologies and Systems*, Київ, 2025, Том 2 (2), 55–80. https://doi.org/10.15407/intechsys.2025.02.055

© Видавець ВД «Академперіодика» НАН України, 2025. Стаття опублікована на умовах відкритого доступу за ліцензією CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

зловмисні дії нападників (атаки), які націлені на викрадення й спотворення даних у мережевому просторі, внесення шкідливого вмісту у інформаційні ресурси, що доступні користувачам мережі, та перешкоджання нормальній роботі мережі.

У даній роботі подано огляд поширених типів мережевих атак. Мета роботи — виявити чинники та їх комбінації, що уможливлюють успішне здійснення атак зловмисниками на їх цільові об'єкти. Розробникам та користувачам програм та сховищ даних, що під'єднані до комп'ютерної мережі, знати ці чинники важливо, щоб мати змогу посилювати стійкість до мережевих атак та враховувати те, з чого користаються нападники. Для того, щоб виявити чинники, на базі загальнодоступних джерел проаналізовано ці типи мережевих атак. З урахуванням результатів аналізу окреслено напрями посилення стійкості до мережевих атак.

1. Основні та допоміжні терміни

Визначення атаки на інформаційну систему, кібератаки. Було запропоновано такі визначення атаки на інформаційну систему:

— «напад на систему безпеки, що походить від інтелектуальної загрози, тобто розумової дії, яка є умисною спробою (особливо у сенсі методу або способу) обійти служби безпеки та порушити безпеку системи» [1] (2000 р., Інженерна робоча група Інтернету *(Internet Engineering Task Force)*);

— «будь-яка зловмисна діяльність у намаганнях збирати, порушувати, не давати використовувати, псувати або руйнувати ресурси інформаційної системи чи саму інформацію» [2] (2010 р., Комітет з систем національної безпеки (США) *(Committee on National Security Systems)*).

Було також запропоновано термін «кібератака»:

«атака через кіберпростір, спрямована ініціатором на використання кіберпростору задля порушення, виведення з ладу чи зловмисного контролювання комп'ютерного середовища/інфраструктури або руйнування цілісності даних чи викрадення контрольованої інформації» [2] (2010 р., Комітет з систем національної безпеки (США) *(Committee on National Security Systems)*).

Інформаційна безпека. Основні якості інформаційної безпеки — це так звана КЦД-тріада (*CIA: confidentiality, integrity, availability*) [3]:

К — конфіденційність (щоб *читати* дані та переписку (тобто те, що передається каналами зв'язку), потрібен дозвіл);

Ц — цілісність (щоб *діяти* з даними та каналами зв'язку, потрібен дозвіл));

Д — доступність (користувач *має* доступ до даних та каналів зв'язку).

Мережева атака. У даній роботі йдеться про мережеві атаки. Мережевою атакою називаємо кібератаку, що здійснюється через комп'ютерну мережу.

Під семантикою мережевої атаки розуміємо опис перебігу та результатів атаки. Складниками опису є відомості про: джерело атаки, об'єкт (цільову жертву) атаки, мету атаки, дії, що виконуються для досягнення мети й допоміжні засоби здійснення атаки та її носії, результат атаки.

За наслідками мережева атака призводить до порушення однієї чи кількох складників КЦД-тріади.

Метою нападників, що здійснюють мережеві атаки, є:

— контроль роботи мережі або її складників, а також комп'ютерних служб, під'єднаних до мережі;

— порушення дієздатності мережі або її складників, а також комп'ютерних служб, під'єднаних до мережі.

Допоміжні терміни. Адреса протоколу інтернету, або ІП-адреса (*IP address*, скорочення від *Internet Protocol address*) — числова позначка пристрою, приєднаного до такої комп'ютерної мережі, що використовує для зв'язку набір мережевих протоколів, відомий під назвою *TCP/IP*. Адреса контролю доступу до середовища, або КДС-адреса (*MAC address*, скорочення від *medium access control address*) — унікальний ідентифікатор, приписаний контролеру мережевого інтерфейсу (*NIC*), використовується як мережева адреса для зв'язку в межах сегмента мережі. Сегмент мережі — це ділянка комп'ютерної мережі. Тип та обсяг сегмента залежать від типу мережі та пристрою чи пристроїв, що застосовано для з'єднання кінцевих вузлів. Гост (*host*) — вузол інтернету, який має ІП-адресу.

2. Атака «відмова в обслуговуванні»

Атака «відмова в обслуговуванні» (ВвО, *denial-of-service attack, DoS attack*) [4, 5] — поширений тип мережевих атак, порушує доступність та конфіденційність.

Семантика атаки «відмова в обслуговуванні»

Джерелом атаки є зловмисна особа, яка провадить атаку власноруч за допомогою обчислювальної машини чи іншого технічного пристрою з певною адресою в інтернеті (ІП-адресою) або за допомогою засобів, що уможливлюють проведення атаки автоматизовано.

Об'єктом атаки є або обчислювальна машина чи мережевий ресурс, зокрема, сервер чи вебсторінка, під'єднані до мережі, (атака рівня мережі), або процес (атака рівня застосування).

Метою атаки є порушення чи руйнування дієздатності певної служби, що під'єднана до мережі.

*Дії, що виконуються для досягнення мети атаки***.** Різновиди дій нападника такі:

• захаращення ресурсів жертви:

— нападник надсилає цільовій жертві багато запитів, що потребують відповіді, завантажуючи жертву своїми завданнями до такої міри, що обслуговування нею інших користувачів уповільнюється або блокується;

— нападник надсилає цільовій жертві великі обсяги даних або такі запити, відповіді на які потребують пересилання великих обсягів даних від жертви, через що перевантажується пропускна здатність каналу зв'язку жертви, а це спричиняє уповільнення обслуговування або нездатність обслуговувати інших користувачів;

— нападник ініціює багато з'єднань з цільовою жертвою та підтримує їх напіввідкритими, через що вичерпуються резерви з'єднань з жертвою й вона не може обслуговувати правомірних користувачів;

• сповільнення руху даних:

— нападник надсилає цільовій жертві правомірний запит на пересилання (нападнику) даних великого обсягу, але здійснює читання відповіді вкрай повільно, через що надовго зменшується можливість жертви обслуговувати інших користувачів;

— нападник надсилає цільовій жертві правомірний запит на пересилання (жертві) даних великого обсягу, але пересилає дані вкрай повільно (в обох цих випадках затягування часу руху даних надовго зменшується можливість жертви обслуговувати інших користувачів);

• нападник віддалено адмініструє обладнання цільової жертви (маршрутизатори, принтери, інше обладнання), пошкоджуючи систему жертви до такої міри, що її треба замінити або переустановити обладнання.

Результат атаки:

• для жертви:

— збитки та втрати різного типу (матеріальні, репутаційні тощо) через порушення у роботі служби, що зазнала нападу, або її руйнування;

— отримання стимулів до виявлення джерела нападу та виявлення причин успіху нападу, а також до удосконалення організації служби, що зазнала атаки, та безпекових заходів;

• для нападника:

— роботу певної служби, що під'єднана до мережі, порушено або припинено;

— нападник може далі використати результат атаки як можливість шантажувати жертву, вимагаючи викуп за усунення наслідків нападу.

Перша в історії атака ВвО сталася 6 вересня 1996 р. [6].

Різновидами атаки ВвО є розподілена атака ВвО (distributed denial-of-service attack, *DDoS attack*) та розширена постійна атака ВвО (an advanced persistent *DoS attack*).

2.1. Розподілена атака «відмова в обслуговуванні». Якщо атака ВвО здійснюється одночасно з кількох джерел, то така атака називається розподіленою атакою ВвО [5]. Напад може здійснюватися на кілька об'єктів одночасно. Атака ВвО вважається розподіленою атакою ВвО, коли у ній задіяно більше, ніж 3–5 вузлів різних мереж [7].

Розподілені атаки ВвО здійснюються, зокрема, на хмарні сервіси [8].

Розподілена атака ВвО є більш небезпечною, ніж атака ВвО, позаяк захиститися від неї важче. Масштаби розподілених атак ВвО останнім часом збільшилися й перевищили обсяг 2.3 Тб/сек [9, 10], а також досягли показника 71 млн. запитів на секунду [11].

Розподілені атаки ВвО бувають рівня мережі та рівня застосування (рівня прикладних програм (ПП), доступ до яких забезпечується через мережу). Атаки рівня мережі потужні, але частота їх зменшується [12]. Атаки рівня ПП не мають тенденції до зменшення [12]. Вони потребують менше ресурсів, ніж атаки рівня мережі, але часто супроводжують їх.

Семантика розподіленої атаки ВвО

Об'єкти атаки. Розподілена атака ВвО може мати один чи кілька об'єктів нападу.

Джерелом атаки є зловмисна особа, що провадить атаку за допомогою засобів, що уможливлюють здійснення атаки автоматизовано, або група зловмисників, що здійснюють атаку власноруч чи використовують спеціальне програмне забезпечення (ПЗ).

Способи формування множини джерел атаки:

— формується група людей, учасники якої за домовленістю одночасно починають напад (атаку ВвО) з різних джерел на певну цільову жертву або на сукупність цільових жертв;

— формується група людей, що надають зловмиснику згоду використовувати їхнє обладнання для здійснення атаки;

— формується мережа ботів, тобто група пристроїв, під'єднаних до інтернету, на кожному з яких працює один або кілька ботів (веброботів, тобто прикладних програм, які у великих масштабах запускають автоматизовані завдання через інтернет).

Мета атаки. Мета розподіленої атаки ВвО рівня мережі та сама, що й мета атаки ВвО. Розподілені атаки ВвО рівня ПП здійснюються здебільшого з метою розривання транзакцій та отримання доступу до баз даних. Також метою розподілених атак ВвО може бути спричинення додаткових витрат на боці оператора ПП, який використовує ресурси на основі хмарних обчислень; це робиться, щоб завдати фінансових втрат власнику ПП або зробити його менш конкурентоздатним [8, 13, 14].

*Дії, що виконуються для досягнення мети атаки***.** Це дії, що лежать в основі атак ВвО.

Результат атаки. Розподілені атаки ВвО рівня мережі спричиняють такі самі наслідки для цільових жертв, як й атаки ВвО, але у більших масштабах.

Атаки рівня ПП можуть порушити роботу служб повернення інформації (за запитом замовника) або функції пошуку на вебсайті.

2.2. Розширена постійна атака «відмова в обслуговуванні». Атака цього різновиду (розширена постійна атака ВвО) [15] становить розширену постійну загрозу протягом тривалого періоду. Найдовша відома атака цього типу тривала 38 днів [16].

Семантика розширеної постійної атаки ВвО

Об'єкти атаки. Кілька цільових жертв: одна справжня, основна, решта — другорядні, ними користаються, аби відвернути увагу від нападу на основну.

Джерело атаки. Таке саме, як у розподіленій атаці ВвО.

Мета атаки. Така сама, як й атаки ВвО, розподіленої атаки ВвО.

*Дії, що виконуються для досягнення мети атаки***.**

Розширене зондування (збирання та аналізування перед атакою даних, віднайдених у відкритих джерелах; посилене сканування задля уведення в оману, аби протягом тривалих періодів уникати виявлення).

Здійснення нападу на основну та другорядні жертви, тактичне перемикання між кількома цілями, аби відвернути увагу й ухилитися від захисних заходів.

Здійснення одночасних багатопотокових атак з використанням складних інструментів, що працюють на рівнях 3–7 моделі взаємозв'язку відкритих систем (*Open Systems Interconnection, OSI*) [5], тобто на рівнях від перенесення пакетів даних у мережевому середовищі до контактування з користувачем.

Узгоджене використання засобів протягом усього періоду нападу.

Результат атаки. Ураження основної жертви, порушення у роботі другорядних жертв.

Зазначимо, що відмова в обслуговуванні може виникнути не лише у результаті зловмисних спланованих дій, як-от атаки ВвО чи розподілені атаки ВвО рівня мережі, а також через те, що занадто багато звичайних незалежних один від одного правомірних користувачів починають майже одночасно звертатися до певної служби, перевантажуючи її ресурси.

2.3. Способи здійснення атак ВвО

Атаки ВвО здійснюються:

— навальним нападом, тобто швидко завалюючи об'єкт атаки запитами на обслуговування (діючи «грубою силою»);

— з використанням знань про роботу ПП жертви та мережевих протоколів, аби скласти завдання для жертви, яке буде схоже на правомірне, але виснажуватиме ресурси жертви;

— з використанням шкідливого ПЗ та завдяки користанню з недоліків та вразливостей на боці жертви.

Аби уникнути виявлення системами безпеки з боку жертви нападники вдаються до підроблення адрес відправників у інтернеті (IP-адрес).

Здійснення атаки ВвО грубою силою

Нападник спрямовує на цільову жертву великий за обсягом потік пакетів, захаращуючи ресурси жертви, через що перенасичується пропускна здатність або вичерпуються інші ресурси системи жертви. Напад з багатьох джерел підсилює здатність перенавантажити смугу пропускання. Для атак використовують пакети даних, що перено-

сяться у мережі за допомогою різноманітних протоколів інтернету (*TCP, UDP, ICMP* тощо [5]).

Здійснення атаки ВвО на основі знань про роботу атакованих об'єктів (використання нападником функціональних здатностей жертви чи мережевого обладнання, яким користується жертва)

Користання з автомасштабування

Аби забезпечити гнучкість обслуговування та належну його якість, деякі служби, ПП, що працюють у хмарному середовищі, використовують автомасштабування, тобто механізм збільшення або зменшення застосування (платних) ресурсів залежно від обсягу вхідного потоку даних. Нападник спрямовує на жертву потік даних, поки жертва не збільшить свої ресурси, аби впоратися з цим потоком, після цього нападник призупиняє атаку, а жертва залишається з надмірним ресурсом (який мусить оплачувати). Жертва зменшує ресурс, а нападник відновлює атаку. Хвилі атаки можуть повторюватися. Атакою такого різновиду є атака йо-йо (*уо-уо*) [8, 13, 14]. Жертва потерпає від погіршення якості обслуговування правомірних користувачів у періоди масштабування, а також від фінансових витрат на надлишкові ресурси.

Користання зі способів роботи протоколів інтернету й можливості встановлювати значення параметрів протоколів

Для виснаження ресурсів жертви нападник використовує спосіб роботи протоколу керування передачею (*Transmission Control Protocol*, *TCP* [5]), а саме попередній обмін повідомленнями («рукостискання») у три етапи. Нападник не вимагає завершення процесу рукостискання й намагається виснажити чергу запитів на з'єднання (чергу *SYN*) у пункті призначення або переповнити смугу пропускання сервера. Приклад такої атаки — захаращення *SYN* (*SYN-flooding*) [5]. Нападник створює напіввідкриті з'єднання й таким чином вичерпує доступні з'єднання жертви. Здійснюється атака таким чином. Гост надсилає серверу велику кількість пакетів, часто з підробленою адресою відправника. Кожен з пакетів сервер обробляє як запит на з'єднання, для чого породжує напіввідкрите з'єднання, потім надсилає пакет-відповідь та чекає на відповідь з адреси відправника. Якщо адреса відправника підроблена, відповідь не надходить, а з'єднання залишається відкритим. Зрештою доступні з'єднання сервера вичерпуються, й він не може обслуговувати правомірних користувачів [5].

Щоб вичерпати ресурси жертви, нападник використовує можливість задавати параметри *TCP*. Він надсилає правомірні запити рівня застосування (тобто прикладних програм), але здійснює читання відповідей вельми повільно, через що з'єднання довго залишаються відкритими й резерв з'єднань жертви вичерпується. Щоб затягувати час руху даних, нападник встановлює дуже малий розмір вікна прийому *TCP* й повільно спустошує буфер прийому *TCP*. [17]

Нападник використовує спосіб оброблення запиту *POST* за допомогою мережевого протоколу передавання гіпертекстів *HTTP*. За

запитом *POST* вебсервер має прийняти дані, вміщені у тіло запиту, згідно з обсягом даних, що надано у заголовку запиту. Нападник надсилає правомірний заголовок *HTTP* із запитом *POST*, а далі надсилає тіло повідомлення із вкрай малою швидкістю. Цільовий сервер намагається прийняти повідомлення відповідно до значення параметру «Довжина вмісту» (*Content Length*) у заголовку, що може тривати довго. Нападник ініціює сотні-тисячі таких з'єднань доти, доки усі ресурси для вхідних з'єднань на боці жертви не вичерпаються, через що інші з'єднання унеможливлюються, доки не будуть надіслані усі дані, що почали надсилатися. [18]

Нападник користається з можливості задати тривалість часу існування датаграми (значення поля *TTL* (*time to live*) у заголовку інтернет-пакета). Він надсилає пакети з малим значенням *TTL*. Якщо час *TTL* спливає, пакет відкидається, а ЦП маршрутизатора має сформувати та надіслати відповідь про перевищення часу згідно з мережевим протоколом керування повідомленнями (*Internet Control Message Protocol, ICMP* [5]). Породження великої кількості відповідей може призвести до перевантаження ЦП маршрутизатора [19].

Один з різновидів захаращування та виснажування ресурсів жертви атаки ВвО — це перевищення пропускної здатності каналу зв'язку жертви. Виявилося, що є багато служб (та відповідних протоколів), якими можна скористатися як віддзеркалювачами (тобто засобами, що надсилають відповідь відправнику, навіть тоді, коли він її не потребує) для підсилення атаки на виснаження смуги пропускання. Для низки протоколів обчислено коефіцієнт підсилення (це є спеціальна міра обсягу інформації, що надсилається у відповідь відправнику) [20]. Так, наприклад, у відповіді на запит до сервера *NTP* (*NTP — Network Time Protocol*) з використанням команди *monlist* відправнику надсилаються відомості про сотні гостів, що останнім часом запитували час у сервера *NTP*.

Застосування знань про роботу ПП на боці жертви

Стандартні запити *HTTP* надсилаються цільовому вебсерверу часто, при цьому для оброблення даних запиту потрібні складні процедури, що потребують багато часу, або операції з базами даних, які можуть вичерпати ресурси цільового вебсервера. Прикладом атаки такого типу є атака *CC* [21–23].

Застосування засобів інтернетної телефонії

Засоби інтернетної телефонії (тобто телефонного зв'язку за допомогою протоколу інтернету) дають змогу породжувати (навіть автоматизовано) велику кількість телефонних викликів й використовувати це для атаки ВвО [24, 25].

Здійснення атак ВвО за допомогою шкідливого ПЗ та користання з вразливостей на боці жертви та вразливостей мережевих протоколів

Шкідливе ПЗ у атаках ВвО

Нападник вбудовує засоби здійснення атаки ВвО у шкідливе ПЗ (як-от хробаки, трояни), яке потрапляє у певну систему, а далі атака розпочинається без відома власника цієї системи. Прикладами такого ПЗ є хробак *MyDoom* [26], що запускає атаку у визначений день та час.

Прикладом шкідливого ПЗ, що запускає атаку ВвО, є *Stacheldraht* [27]. Для здійснення атаки ВвО за допомогою *Stacheldraht* зловмисник спочатку уражає шкідливим ПЗ низку систем, що здійснюють оброблення даних, далі використовує клієнтську програму, аби зв'язатися з цими системами, а вони дають команди зомбі-агентам просувати атаку ВвО. У атаці за допомогою *Stacheldraht* нападник користається з вразливостей програм, що обслуговують віддалені з'єднання й працюють на віддалених гостах, що є цільовими жертвами.

Напади рівня ПП (рівня застосування) користаються з дефектів у програмах, що чекають на зв'язок з віддаленими гостами, й використовують програми-пролази (тип шкідливого ПЗ, *exploits*), що можуть змусити ПП, які виконуються на сервері, заповнити дисковий простір чи вичерпати усю доступну пам'ять або час ЦП. Прикладом вірусу, що вичерпує наявні ресурси системи, є *fork-bomb* [28, 29]. Вірус *fork-bomb* містить нескінченний цикл. При роботі вірусу повторно запускається один й той самий процес, що поглинає час ЦП, а також завантажує таблицю процесів операційної системи.

Нападник вдається до автоматизації атакувальних дій. Так, відомий інструмент атаки ВвО *Slowloris* [30, 31]. Атака починається з того, що відкривається з'єднання з цільовою жертвою — вебсервером — та підтримується відкритим якомога довше. Засобом для підтримування відкритого з'єднання є надсилання часткового запиту. *Slowloris* періодично надсилає низки заголовків *HTTP*, не завершуючи запиту. Уражені сервери підтримуватимуть з'єднання відкритими, використовуючи максимально резерв одночасних підключень, через що відмовлятимуть у підключеннях іншим клієнтам.

Заражені шкідливим ПЗ комп'ютери використовуються як пульсуючі зомбі, спрямовані на здійснення періодичних нетривалих захаращень вебсайтів жертви з метою уповільнення їх роботи. Атаки, спрямовані не так на припинення роботи мережевих служб, як на уповільнення роботи мережі або мережевих служб (атаки погіршення обслуговування), важко виявити, вони можуть гальмувати доступ до мережевих служб тривалий час й зрештою значно зашкодити [32].

Атаці з використанням шкідливого ПЗ передує підготовча робота. Спочатку за допомогою шкідливого ПЗ (хробака) нападник інфікує сотні-тисячі пристроїв інтернету речей. Хробак поширюється через мережі та системи, перебираючи контроль над погано захищеними пристроями інтернету речей (як-от термостати, годинники з підтримкою *Wi-Fi*, пральні машини). Власник чи користувач зазвичай не дізнається відразу, що сталося зараження пристрою. Далі заражений пристрій стає складником атаки: коли набирається достатня кількість заражених пристроїв, вони отримують команду

зв'язатися з надавачем послуг інтернету. Тобто спочатку формується мережа ботів (ботнет), а потім цей ботнет атакує жертву. Приклад — шкідливе ПЗ *Mirai* [33], призначене для формування мережі зомбі (ботнету).

Користання з недоліків у проєктуванні систем з метою спричинити самозахаращення жертви

Нападник, аби ініціювати з'єднання, надсилає на відкритий порт цільової жертви пакет *TCP* (пакет, що транспортується за допомогою *TCP*, протоколу керування передачею), що потребує відповіді джерелу, але дані про джерело (ІП-адреса) підроблено таким чином, що вони збігаються з даними пункту призначення, через що цільова жертва (гост) постійно відповідає самій собі. Прикладом такого нападу є атака *LAND* [34, 35]. Здійсненню самозахаращення сприяють недоліки у проєктуванні систем: пристрій має можливість приймати запит на проводові з'єднання, що надходить від цього самого пристрою.

Користання з прорахунків у налаштуванні мережевих пристроїв

Неналежне налаштування мережевих пристроїв використовується у атаці *smurf* [36, 37]: надсилання даних з адресою типу «розсіяна адреса» *(broadcast address)* уможливлює надсилання пакетів усім машинам-гостам певної мережі, а не якійсь конкретній машині. Нападник надсилає багато інтернет-пакетів з підробленою адресою джерела, прописаною як адреса жертви. Більшість пристроїв у мережі за замовчанням відповідатимуть, надсилаючи відповідь на ІП-адресу джерела, через що машина жертви захаращується потоком даних.

Користання з хиб захисних засобів

Нападник користається з недоліків у захисних засобах жертви, через які можливе віддалене адміністрування обладнання жертви (маршрутизаторів, принтерів чи іншого мережевого обладнання). Нападник користається з цих вразливостей, аби змінити прошивку пристрою або пошкодити її. В результаті пристрій стає недієздатним, його потрібно ремонтувати або замінити [38]. Цей прийом застосовується у атаці ВвО. Прикладом ПЗ такого типу є *BrickerBot* [39]; це шкідливе ПЗ спричиняє постійну ВвО пристроїв інтернету речей.

Користання з різноманітних вразливостей на боці жертви та вад мережевих протоколів

За допомогою протоколу користувацьких датаграм (*User Datagram Protocol, UDP*), який не вимагає з'єднання з сервером (це означає, що ІП-адреса джерела не перевіряється, коли запит приймається сервером), нападник може надсилати жертві запити з підроблених ІП-адрес, відповіді на які є значних обсягів. (Про підроблення ІП-адрес див. у розділі 3.)

Для розподіленої атаки ВвО використовувались дефекти у засобах обслуговування однорангових серверів. Зокрема, нападники користались з *DC++* (клієнта однорангового обміну файлами, якого можна застосувати для під'єднання до мережі «Прямого зв'язку»

(Direct Connect) чи до розширеного протоколу прямого з'єднання *(ADC protocol)*), даючи команду клієнтам великих центрів однорангового обміну файлами від'єднатися від їхньої однорангової мережі й натомість приєднатися до вебсайту жертви [40–42].

Користаючись з вразливостей на боці жертви, нападник може спричинити цілочисельне переповнення, маніпулюючи максимальним розміром сегмента (*maximum segment size*) та вибірковим підтвердженням (*selective acknowledgment*) [43, 44].

Користаючись з вади у механізмі короткої перерви під час повторного передавання даних за допомогою протоколу *TCP*, нападник надсилає короткі синхронізовані хвилі потоку даних, аби розірвати *TCP*-з'єднання у каналі зв'язку, яким передаються ці дані [45]. Напад такого зразка названо атакою землерийки *(shrew attack)* [46, 47]: коли клієнт надсилає запити на сервер, зловмисник одночасно надсилає низку запитів на маршрутизатор, змушуючи маршрутизатор призупинити передавання даних; пакети відкидаються під час короткої перерви у передаванні даних *(retransmission timeout, RTO)* (мінімільний рекомендований час перерви 1 сек.); після перерви клієнт повторно передає втрачені пакети, значно сповільнюючи передавання *TCP*. Аби втрутитися у передавання даних клієнта й спровокувати його призупинення, нападник має відслідковувати процес передавання даних клієнта на сервер, тобто діяти як шкідливий посередник (про «шкідливого посередника» див. у розділі 4).

Через помилку у коді збирання фрагментів пакетів *TCP/IP* низки операційних систем сервер неспроможний зібрати фрагментований пакет, якщо нападник надсилає спотворені фрагменти пакету цільовій жертві. Це використано у атаці *teardrop* [48]: нападник надсилає цільовій жертві (серверу) спотворені фрагменти пакету: з перекриттям, з перевищенням корисного навантаження, сервер неспроможний зібрати фрагментований пакет, тому виникає ВвО.

Щоб виснажувати мережі та сервери, нападники використовують вразливості протоколів *UPnP* (*Universal Plug and Play*, сукупність мережевих протоколів, що дають змогу різноманітним пристроям безпосередньо виявляти наявність один одного у мережі та підтримувати мережеве обслуговування [49]). До вразливостей протоколів *UPnP* належать: неправильні стандартні налаштування (налаштування за замовчанням), через які пристрої залишаються відкритими для віддаленого доступу, брак механізму перевірки справжності, вразливості віддаленого виконання коду [50].

Те, що деякі маршрутизатори мають вразливості ПЗ *UPnP*, дає змогу нападнику спрямувати відповіді з порту 1900, через який працює протокол *UPnP*, у напрямку за власним вибором [50–52].

Вразливість протоколу визначення адреси (*Address Resolution Protocol, ARP*) дає можливість нападнику зв'язати свою КДС-адресу з ІП-адресою іншого комп'ютера або пристрою, тобто обманути цей протокол (див. розділ 3).

В результаті призначений справжньому адресату потік даних переспрямовується на адресу зловмисника, через що виникає відмова в обслуговуванні дійсного адресата. У цьому разі відмова в обслуговуванні полягає в тому, що потік даних не доходить до адресата, якому він призначався.

Зазначимо, що інструменти здійснення розподілених атак ВвО виявлено в товарообігу [53].

3. Атаки обману протоколів

Є низка різновидів таких атак, зокрема, обман протоколу визначення адреси (*ARP* spoofing), або «отруєння кешу» *ARP* [54].

Ці атаки порушують конфіденційність.

Протокол виявлення адреси, або ПВА (*Address Resolution Protocol* [55]) призначений для того, щоб дізнатися адресу рівня каналу зв'язку (як-от КДС-адресу), що пов'язана з даною ІП-адресою рівня мережі. Отже, ПВА реалізує відображення множини ІП-адрес у множину КДС-адрес.

Семантика атаки обману ПВА

Джерелом атаки є зловмисна особа, яка провадить атаку в межах деякої локальної мережі, використовуючи або власну обчислювальну машину, або підготовлений до атаки гост з цієї локальної мережі, або за допомогою програмних засобів, що уможливлюють автоматизоване здійснення атаки.

Об'єктами атаки є гости у зоні локальної мережі, зокрема, їхні кеші, де зберігаються повідомлення ПВА, а отже, служби та користувачі, передавання даних між якими здійснюється за підробленими у результаті атаки адресами.

Метою атаки є зв'язування КДС-адреси нападника з ІП-адресою іншого госта, як-от шлюзу, що використовується за умовчанням, аби надалі будь-який потік даних, призначений для цієї ІП-адреси, надсилався нападнику.

Дії, що виконуються для досягнення мети атаки. Нападник надсилає підроблені ПВА-повідомлення в зону локальної мережі.

Результат атаки:

• для жертв:

— витік даних;

• для нападника:

— отримання доступу до потоку даних, що передається каналом зв'язку;

— можливість розпочати атаку «шкідливого посередника», атаку ВвО.

Що уможливлює здійснення атаки обману ПВА.

Атаку можна здійснити лише у мережах, що використовують ПВА, до того у нападника має бути прямий доступ до локального сегменту мережі, на який спрямовано атаку.

Атаці сприяють певні особливості роботи ПВА (вважаються вразливостями ПВА). ПВА застосовується для перетворення адрес шару

мережі (інтернет) на адреси рівня каналу зв'язку. Коли IP-датаграма пересилається від одного госта до іншого у межах локальної мережі, за IP-адресою пункту призначення має бути виявлено КДС-адресу, щоби здійснити передавання даних каналом зв'язку. Коли відома IP-адреса іншого госта й потрібна його КДС-адреса, у локальну мережу у режимі поширення надсилається пакет (*broadcast packet*), так званий ПВА-запит. Машина призначення, IP-адреса якої вказана у цьому ПВА-запиті, відповідає, надсилаючи ПВА-відповідь, що містить КДС-адресу тієї IP-адреси. ПВА є протоколом без збереження стану. Гости у мережі автоматично зберігатимуть у кеші будь-які відповіді ПВА, які вони отримують, незалежно від того, чи були надіслані відповідні запити від цих гостів. Навіть записи ПВА, термін дії яких не вичерпався, будуть затерті, коли надійде новий пакет ПВА-відповіді. У ПВА немає способу, за допомогою якого гост може перевірити справжність однорангового вузла, від якого походить пакет. Отже, обмеженість можливостей ПВА спричиняє вразливість, що уможливлює обманювання ПВА [5, 54, 56].

Нападник може надсилати ПВА-відповідь з IP-адресою деякого пристрою у локальній мережі та підробленою КДС-адресою (як КДС-адресу, відповідну IP-адресі, нападник вказує КДС-адресу свого (контрольованого ним) пристрою), й ця ПВА-відповідь автоматично зберігатиметься у кешах гостів мережі без перевірки справжності відправника.

Зазначимо, що існує низка знарядь, що можуть бути використані для провадження атаки обманювання ПВА (наприклад [57–59]).

4. Атака «шкідливий посередник»

Атаки типу «шкідливий посередник» (атаки ШП, *MITM*, *man-in-the-middle attacks*) [5, 60] порушують конфіденційність та цілісність.

Семантика атаки ШП

Джерелом атаки є зловмисна особа, яка провадить атаку за допомогою обчислювальної машини.

Об'єктами атаки є канал зв'язку (протокол передачі даних), за допомогою якого здійснюється обмін повідомленнями між кореспондентами, а також кореспонденти, що користуються цим каналом для передачі повідомлень.

Метою атаки є несанкціонований доступ до повідомлень, що передаються каналом зв'язку, та перехоплення цих повідомлень.

Дії, що виконуються для досягнення мети атаки [5]**.** Нападник має знати структуру та властивості методу передавання даних, що використовується, а також знати про те, що передавання повідомлення планується. У нападника має бути можливість встановити з'єднання з жертвами. Нападник встановлює з'єднання з кореспондентами (K1, K2), що мають на меті обмінюватися повідомленнями, причому з K1 він з'єднується під виглядом K2, а з K2 — під виглядом K1. Кореспон-

дент К1 надсилає своє повідомлення, вважаючи, що адресатом є кореспондент К2, але насправді повідомлення отримує нападник й чинить з цим повідомленням на свій розсуд: ознайомлюється зі змістом повідомлення, копіює його, вносить зміни у повідомлення. Далі нападник надсилає повідомлення (можливо, модифіковане) кореспонденту К2, який вважає, що відправником є К1.

Результат атаки:

• для жертв:

— витік даних;

— спотворення змісту повідомлень, що їх передають каналом зв'язку;

• для нападника:

– отримання доступу до змісту повідомлень, що передаються каналом зв'язку;

– підміна передаваних повідомлень;

– видалення інформації, що передається каналом зв'язку;

– перенаправлення повідомлення, що передається каналом зв'язку, до пункту, який не передбачався відправником повідомлення;

– зміна параметрів з'єднання між сервером та клієнтом при встановленні між ними з'єднання.

Що уможливлює здійснення атаки ШП.

Наявність можливості контролю інформаційних потоків у системі передавання даних (соціальна мережа, публічний сервіс електронної пошти, система миттєвого обміну повідомленнями належать до публічних засобів комунікації без захисту; власник ресурсу, який забезпечує комунікацію, має контроль над інформацією, якою обмінюються кореспонденти, отже, має потенційну можливість посередництва).

Знаходження нападника у зоні прийому *Wi-Fi* (нападник може втрутитися як ШП у зоні прийому незашифрованої точки доступу безпроводової мережі *Wi-Fi*).

Використання проміжних серверів під час передавання даних (нападник може розбити оригінальне *TCP*-з'єднання на два нових (між собою та клієнтом та між собою та сервером), позаяк не завжди між клієнтом та сервером існує безпосередній зв'язок, у багатьох випадках вони пов'язані через низку проміжних серверів, а деякі з них можуть бути використані зловмисником).

Здійснення атаки ШП на менш безпечні протоколи, наприклад, *SSH*, *IPsec*, *PPTP*.

Обманні дії нападника (нападник намагається уникнути перевіряння справжності кореспондентів, що обмінюються даними (автентифікації); для цього він або користається з недоліків наявних процедур перевіряння справжності, або використовує сертифікат (підтвердження справжності), отриманий шахрайським способом, зокрема, це можливо тоді, коли сам центр сертифікації став жертвою атаки ШП).

Нехтування вимогами інформаційної безпеки з боку користувачів.

Зв'язок між атакою ШП та іншими атаками. Атака ШП уможливлює інші атаки, як-от «впорскування *SQL*» (див. розділ 5), «запуск сценарію, що передається через сайт» (див. розділ 7).

5. Атака «впорскування *SQL*»

Атаки «впорскування *SQL*» (*SQL injection attacks*) [61] порушують конфіденційність, цілісність та доступність.

Перші громадські обговорення «впорскування *SQL*» почалися близько 1998 р. [62].

Семантика атаки «впорскування *SQL*»

Джерелом атаки є зловмисна особа, яка провадить атаку за допомогою обчислювальної машини або за допомогою засобів, що уможливлюють автоматизоване здійснення атаки.

Об'єктом атаки є прикладні програми (застосунки), що керуються даними *(data-driven applications)*, як-от вебсайти, бази даних *SQL* будь-якого типу.

Метою атаки є розкриття даних атакованої системи, псування чи руйнування даних атакованої системи.

*Дії, що виконуються для досягнення мети атаки***.** Загальний план дій нападника: у вхідне поле команд при побудові запиту *SQL* вміщуються шкідливі оператори *SQL (malicious SQL statements)*. Більш детально: оператори *SQL* складаються з даних, що використовуються оператором *SQL*, а також команд, що контролюють те, як виконується оператор *SQL*; нападник так заповнює поля запиту, що після формування запиту з вмісту цих полів у запиті зрештою утворюється шкідлива команда; для цього під час підготовки вхідних даних для запиту у поле даних нападник уводить рядок, що не є даними відповідно до призначення цього поля, а натомість містить команду та/або інструкцію, що зрештою спотворює запит, що мав би будуватися за вхідною формою, тобто нападник використовує поля запиту не за призначенням.

Результат атаки:

- для жертви:
  - витік даних;
  - втрата, спотворення, руйнування даних;
- для нападника:
  - отримання несанкціонованого доступу до даних;
  - здійснення витоку даних;
  - підроблення даних;
  - руйнування даних.

Що уможливлює успіх атаки «впорскування *SQL*». «Впорскування *SQL*» мусить користатися з безпекової вразливості прикладного ПЗ. Приклади такої вразливості:

— вхід користувача ПЗ неправильно фільтрується щодо рядків символів керувальних послідовностей, вставлених у в оператори *SQL*;

— вхід користувача ПЗ не є строго типізованим та виконується несподіваним чином.

Крім описаного «впорскування *SQL*» виявлено також «сліпе впорскування *SQL*» [63].

6. Атака віддзеркалення

Атака віддзеркалення *(reflection attack)* [5] — спосіб нападу на систему «виклик-відгук» підтвердження справжності, яка використовує той самий протокол у обох напрямках, тобто кожна сторона застосовує для підтвердження справжності той самий протокол, що й інша сторона. Ця атака порушує конфіденційність та цілісність.

Семантика атаки віддзеркалення

Джерелом атаки є зловмисна особа, яка провадить атаку власноруч за допомогою обчислювальної машини.

Об'єктами атаки є система підтвердження справжності «виклик-відгук», яка використовує той самий протокол в обох напрямках (тобто кожна сторона застосовує для підтвердження справжності той самий протокол, що й інша сторона), а також сторона (далі — *цільова жертва*, *жертва*), що користується цією системою.

Мета атаки — обманом змусити цільову жертву дати відповідь на виклик самої жертви.

*Дії, що виконуються для досягнення мети атаки***.**

Нападник ініціює з'єднання з цільовою жертвою.

Цільова жертва надсилає виклик ініціатору з'єднання (у випадку атаки віддзеркалення — нападнику), аби перевірити справжність ініціатора з'єднання.

Нападник відкриває інше з'єднання з жертвою й надсилає жертві від себе її виклик.

Жертва відповідає на виклик.

Нападник надсилає жертві відповідь самої жертви через початкове з'єднання (а інше, допоміжне з'єднання, скасовує).

Результат атаки:

• для жертви:

— вступає у контакт з об'єктом, справжність якого насправді не перевірена й не підтверджена;

• для нападника:

— має повністю автентифіковане підключення до каналу.

Що уможливлює здійснення атаки.

Нападник підроблюється під цільову жертву, а на стороні цільової жертви немає засобу виявлення підробки.

7. Атака «Запуск сценарію, що передається через сайт»

Атака «запуск сценарію, що передається через сайт» (*cross-site scripting attack,* атака СПчС, *XSS attack*) [64] є випадком впорскування коду. Порушує конфіденційність та цілісність.

Короткі пояснення термінів

Запуск сценарію, що передається через сайт (СПчС, *cross-site scripting, XSS*) — це тип безпекової вразливості (хиба СПчС), що трапляється у деяких прикладних програмах вебу (вебзастосунках). Термін уведено у 2000 р. [65, 66].

Про вразливості СПчС повідомляли й користалися з них з 1990-х років. Вразливість СПчС дає можливість нападнику вмонтовувати клієнтські сценарії у вебсторінки, які бачать інші користувачі [67].

Більшість фахівців виділяють принаймні два різновиди хиб СПчС: непостійні та стійки [68].

Нестійка (або віддзеркалена) вразливість до СПчС: вебклієнт подає дані (зазвичай у параметрах запиту *HTTP*), на боці сервера здійснюється негайне оброблення даних (граматичний аналіз), результати виводяться на сторінку цього клієнта. Носієм атаки може бути вебадреса *URL* у електронній пошті, вебадреса сайту, якому довіряють, атака можлива через сайт, що має вразливість СПчС.

Стійка (збережена) вразливість СПчС: сервер зберігає дані, які постачає нападник, а потім ці дані відображуються на «нормальних» сторінках, що повертаються користувачам під час звичайного переглядання (вмісту сайту), й при цьому не здійснюється належний захист від можливого шкідливого вмісту *HTML*-тексту (екранування *HTML*).

Семантика атаки СПчС

Джерелом атаки є зловмисна особа, яка провадить атаку власноруч за допомогою обчислювальної машини.

Об'єктами атаки є вебсайт, його користувачі та їхні локальні середовища.

Метою атаки є користання з результатів виконання створеного нападником шкідливого сценарію, впорснутого у носій атаки.

Дії, що виконуються для досягнення мети атаки.

Нападник створює шкідливий сценарій (ШС) та добирає носій атаки. Залежно від того, на яку саме вразливість застосунків розраховує нападник, він добиратиме той чи інший носій атаки.

Нападник впорскує ШС у вираз, що є на позір безпечним й може оброблятися вразливими ПП на боці сервера чи клієнта.

Нападник організовує розміщення шкідливого виразу таким чином, щоб його бачили інші користувачі.

Нападник може застосувати засоби соціальної інженерії, аби спонукати користувача (як цільову жертву) активувати носій атаки (наприклад, натиснути на посилання, що містить шкідливий код).

Результат атаки:

• для жертв:

— користувач — активізує носій атаки;

— вебсервер — містить носій атаки;

— локальне середовище користувача (та користувач) — потерпає від наслідків виконання шкідливого сценарію (ШС) (наприклад, відбувається витік даних користувача, зокрема, секретних);

• для нападника:

— у вебсайт впорснуто шкідливий сценарій (у сторінку вебсайту, яку бачать інші користувачі);

— здобуто доступ до секретних даних користувача (або багатьох користувачів) вебу.

7.1. Що уможливлює здійснення атаки СПчС

Атаки СПчС користаються з відомих вразливостей у ПП вебу (прикладних програмах, доступ до яких здійснюється за допомогою пошуковика вебу), їхніх серверах або системах, що підключаються до цих ПП.

Використовуючи одну з таких вразливостей, нападник вкладає шкідливий вміст у дані, що передаються з підданого небезпеці (через наявну вразливість) сайту. Коли такий комбінований вміст надходить у програму-переглядач на боці клієнта, він є доставленим з надійного джерела і тому працює згідно із дозволами, наданими системі на тому сайті.

Знаходячи способи впорскування шкідливих сценаріїв у вебсторінки, нападник може здобути привілеї доступу до секретного вмісту сторінки, файлів *Cookie* сеансу та іншої інформації, яку програма-переглядач зберігає для користувача.

Коли користаються зі стійкої вразливості СПчС шкідливий сценарій передається автоматично; жертву спеціально не обирають, жертвою виявляється кожен користувач, до якого потрапляє заражена сторінка, а також сайт, через який поширюється сценарій (вразливий сайт).

Носієм впорскування можуть стати будь-які дані, що їх може контролювати нападник, й які надійшли до ПП вебу електронною поштою, від електронних журналів тощо.

7.2. Способи здійснення атак СПчС

Атаки СПчС здійснюються:

— з використанням нестійкої вразливості СПчС;

— з використанням стійкої вразливості СПчС.

Здійснення атаки як віддзеркалення з використанням нестійкої вразливості СПчС

Носієм атаки такого типу є зазвичай електронна пошта або нейтральний вебсайт.

Загальний опис перебігу атаки з віддзеркаленням:

— нападник готує приманку у вигляді вебадреси, яка містить носій СПчС;

— нападник забезпечує розміщення приманки там, де її може побачити цільова жертва (користувач), наприклад, на нейтральному вебсайті, або надсиляє приманку жертві електронною поштою;

— жертва бачить безневинну на вигляд вебадресу *URL*, що вказує на надійний сайт (але ця адреса містить носій СПчС, про що жертва не знає);

— якщо цей (що вважається надійним) сайт має вразливість до носія СПчС, то натискання на посилання може призвести до того, що програма-переглядач жертви виконає впорснутий сценарій.

Приклад перебігу атаки з використанням нестійкої вразливості СПчС [69].

Учасники подій:

— вебсайт (ВС);

— користувач (К), що відвідує ВС;

— нападник (Н), який дізнається, про те, що ВС має нестійку вразливість СПчС (віддзеркалення).

К має змогу зареєструватися на ВС, вказавши своє ім'я та пароль. К може зберігати на ВС свої секретні дані, як-от відомості про рахунки. Коли якийсь користувач реєструється на ВС, програма-переглядач зберігає файли *Cookie* авторизації, а обидві машини (клієнт та сервер) мають запис про те, що цей користувач зареєстрований.

Дії Н та їх наслідки.

1) Н відвідує сторінку «Пошук», тобто сторінку сайту ВС для здійснення пошуку, уводить у відповідне поле пошуковий запит й натискає клавішу, аби подати запит.

Якщо результат не знайдено, на сторінці (що повертається користувачу) буде показано введений пошуковий запит, після якого написано фразу «не знайдено», також з'явиться вебадреса *URL* сайту ВС з відомостями про пошуковий запит.

Якщо запит, наприклад, має вигляд «квіти», то на сторінці з'являється такий текст: «квіти не знайдено», а вебадреса *URL* має вигляд: «*http://BC.org/search?q*=квіти».

Якщо запит має вигляд: «<*script*>*alert*('СПчС');</*script*>», то:

а) з'являється віконце тривоги з написом «СПчС»,

б) на сторінці з'являється фраза «не знайдено», а також повідомляється про помилку з текстом 'СПчС',

в) *URL* має вигляд: «*http://BC.org/search?q*=<*script*>*alert*('СПчС');</*script*>», що свідчить про поведінку ПП сервера, якою може скористатись зловмисник.

2) Н конструює такий *URL*, аби скористатися з виявленої вразливості:

http://BC.org/search?q=квіти<*script%20scr="http://H.com/*ВНС.*js*"></*script*>.

Н може закодувати символи <, >, /, ", щоб можливий читач не зміг негайно розібратися, що ця вебадреса шкідлива.

3) Н надсилає електронний лист якомусь користувачу сайту ВС із закликом «Гляньте-но, які гарні квіти!» та підготовленим посиланням.

4) Цей електронний лист отримує користувач К, який любить квіти і переходить за посиланням. В результаті звернення до ВС у відповідь з'являється фраза «квіти не знайдено», а також діє ярлик сценарію (<*script*>) (на екрані цього не видно) й завантажується та працює шкідлива програма (ВНС.*js*, ВНС — «вельми небезпечний сценарій») нападника Н, запускаючи атаку. К забуває про це (про лист та спробу пошуку цікавинки за посиланням).

5) Програма ВНС.*js* діє у програмі-переглядачеві користувача К так, наче її джерелом є сайт ВС. Вона захоплює копію файлу *Cookie* авторизації К й надсилає її на сервер нападника Н, де Н може її переглядати.

6) Н вміщує файл *Cookie* авторизації К у свою програму-переглядач як свій власний, заходить на ВС й вже зареєстрований як К.

7) Н заходить у розділ рахунків сайту ВС, переглядає номер кредитної картки користувача К та захоплює копію. Далі Н змінює пароль, відтоді К не може зареєструватися.

Кінець прикладу.

Здійснення атаки СПчС з використанням стійкої вразливості СПчС

Загальний опис перебігу атаки з використанням стійкої вразливості СПчС:

— нападник приєднується до сайту С, де користувачі можуть виставляти (розміщувати) повідомлення у форматі *HTML*, аби інші користувачі могли їх прочитати; нападник має свій профіль (короткий біографічний начерк) на цьому сайті;

— нападник пише сценарій, призначений для запуску з програм-переглядачів інших користувачів, коли вони відвідують його профіль на сайті С; сценарій призначено для викрадення персональних даних користувачів, що зберігаються на сервері сайту С.

— коли нападник отримує запит від деякого користувача сайту С, він дає на нього принагідну відповідь, але у кінці своєї відповіді вміщує свій шкідливий сценарій; якщо цей сценарій вміщено у структурний елемент <*script*>, він не з'являтиметься на екрані.

— коли якийсь користувач К, зареєстрований на сайті С, переглядає профіль нападника, у якому міститься відповідь зі шкідливим сценарієм, цей сценарій виконується автоматично програмою-переглядачем користувача К й викрадає копію персональних даних К безпосередньо з машини К й надсилає швидке повідомлення на сервер нападника, що збирає ці дані.

Приклад перебігу атаки з використанням стійкої вразливості СПчС [69].

Учасники подій: нападник (Н), вебсайт (ВС), користувач (К).

1) Н отримує для себе обліковий запис на сайті ВС.

2) Н завважує, що ВС має стійку вразливість СПчС: якщо хтось заходить у розділ новин й розміщує коментар, ВС покаже усе, що введено. Якщо текст коментаря містить ярлики *HTML*, вони будуть додані до того, що було на вебсторінці спочатку, зокрема, будь-які ярлики сценарію запрацюють, коли сторінка завантажується (коли її хтось переглядає).

3) Н читає статтю у розділі новин і вводить такий, наприклад, коментар:

«Мені у цій оповіді сподобалися півники. Вони такі кумедні! <*script scr="http://H.com/*ВНС.*js*">».

4) Коли К (або хтось інший) завантажує сторінку з коментарем, працює ярлик сценарію нападника Н та викрадає файл *Cookie* авторизації К й надсилає його на сайт нападника Н.

5) Н тепер може відібрати у К сеанс та видавати себе за К.

Кінець прикладу.

Різновидами СПчС є видозмінений СПчС [70] та запуск сценарію на основі Об'єктної моделі документа (ОМД, *Document Object Model, DOM*) [71].

Запуски сценаріїв, що передаються через сайт, виконані на вебсайтах, становили приблизно 84 % усіх безпекових вразливостей, засвідчених *Symantec* до 2007 р. [72].

Підсумки та висновки

За допомогою аналізу поширених типів мережевих атак виявлено, що для здійснення цих атак нападники користаються з:

1) навального нападу (атаки ВвО);

2) функційних можливостей ПП жертви чи мережевого обладнання (атаки ВвО);

3) недоліків у проєктуванні мережевого ПЗ та ПП, що працюють на боці клієнта чи сервера, зокрема, з браку перевірок даних, що надходять на вхід ПП (атаки: ВвО, обману ПВА, ШП, впорскування *SQL*, СПчС, віддзеркалення);

4) прорахунків у налаштуванні мережевих пристроїв (атаки ВвО);

5) хиб або браку захисних засобів (атаки: ВвО, ШП);

6) вразливостей мережевих програм, зокрема, помилок у коді (атаки: ВвО, СПчС);

7) наявності проміжних засобів з'єднання (атаки ШП);

8) шкідливого ПЗ (атаки ВвО);

9) нехтування вимогами інформаційної безпеки з боку користувачів (атаки ШП);

10) соціальної інженерії (атаки СПчС).

Отже, в арсеналі нападників є: навальний напад , шкідливе ПЗ, знання про різноманітні вразливості та прорахунки на боці жертви, засоби соціальної інженерії, докладне знання про те, як функціонує мережеве обладнання та ПЗ жертви. Найбільша кількість виявлених чинників пов'язана з атаками типу ВвО, а чинник 3 пов'язаний з усіма розглянутими типами атак. Зазначимо, що завдяки обізнаності про роботу мережевого обладнання та ПП, що під'єднані до мережі, нападники мають можливість здійснювати атаки, поводячись при цьому стосовно протоколів мережі та стосовно атакованих служб як правомірні користувачі й навіть не шукаючи хиб захисних засобів чи вразливостей ПЗ на боці жертви.

У відповідь на напад суб'єкт на боці жертви атаки може здійснити: пошук джерел нападу, виявлення причин успіху нападу, вдосконалення організації служби, що зазнала атаки, та посилення безпекових заходів.

Зазначимо, що попри безпосереднє реагування на кожен конкретний випадок атаки того чи іншого типу розробникам ПЗ, що призначено для роботи в мережі, доцільно уживати перспективних заходів, що сприяли б стійкості до мережевих атак, враховуючи те, з чого користаються нападники.

З огляду на те, що чинник 3 (користання з недоліків у проєктуванні мережевого ПЗ та ПП, під'єднаних до мережі) пов'язаний з усіма розглянутими типами атак, виходить, що винятково важливим напрямом посилення стійкості до мережевих атак, розглянутих в цій роботі, є удосконалення розроблення проєктів ПЗ мереж та ПП, що працюють у мережі, аби запобігти появі вразливостей, якими можуть скористатися зловмисники. Зокрема, на наш погляд, проєктувальнику слід забезпечувати повноту побудувань, як-от:

— визначати ознаки «правильних» та «неправильних» даних, що надходять на вхід програмних складників мережі або є результатом їхньої роботи;

— визначати ознаки нормальних та аварійних ситуацій перебігу процесів у мережі;

— розробляти засоби контролю даних та мережевих процесів;

— розробляти засоби оброблення аварійних ситуацій.

З огляду на роль людського фактору у здійсненні атак важливою для протистояння мережевим атакам є обізнаність користувачів мережі щодо вимог інформаційної безпеки та щодо загроз з боку засобів соціальної інженерії.

Важливий напрям убезпечення від мережевих атак — удосконалення наявних та розроблення новітніх систем забезпечення комп'ютерної безпеки. Зокрема, є потреба у тому, щоб вміти відрізнити розподілену атаку ВвО на службу, під'єднану до мережі, від раптового сплеску звернень правомірних користувачів до цієї служби. Тобто йдеться про те, щоб розпізнати атаку на службу у разі, коли кількість звернень до неї раптово й незвичайно збільшується.

ЛІТЕРАТУРА / REFERENCES

1. Shirey, R. Internet Security Glossary. URL: https://www.rfc-editor.org/info/rfc2828 [Accessed May. 2000]
2. Information Security. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf
3. NIST Special Publication 1800-26A. URL: https://www.nccoe.nist.gov/publication/1800-26/VolA/index.html
4. Understanding Denial-of-Service Attacks. URL: https://www.cisa.gov/news-events/news/understanding-denial-service-attacks
5. Computer networks / Andrew S. Tanenbaum, David J. Wetherall. — 5th ed.. URL: https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Computer%20Networks.pdf
6. Distributed Denial of Service Attacks. *The Internet Protocol Journal,* Vol. 7 (4). URL: https://web.archive.org/web/20190826143507/https:/www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-30/dos-attacks.html

7. DDoS Mitigation using Cumulus Linux. URL: https://www.hyperscalers.com/how-to-implement-DDoS-mitigation-cumulus-linux-bare-metal-switches-BMS-100G-networking
8. Mor Sides, Anat Bremler-Barr, Elisha Rosensweig. Yo-yo attack: vulnerability in auto-scaling mechanism. *SIGCOMM – 15: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 103–104. https://doi.org/10.1145/2785956.2790017
9. Amazon 'thwarts largest ever DDoS cyber-attack' 18 June 2020. *BBC Newsб, Jun 18, 2020. URL:* https://www.bbc.com/news/technology-53093611
10. AWS Security Blog. AWS Shield Threat Landscape report is now available. By Mário Pinho, 29 May. 2020. URL: https://aws.amazon.com/blogs/security/aws-shield-threat-landscape-report-now-available
11. The Cloudflare Blog. Cloudflare mitigates record-breaking 71 million request-per-second DDoS attack 2023-02-13. By Omer Yoachimik, Julien Desgats, Alex Forster. URL: https://blog.cloudflare.com/cloudflare-mitigates-record-breaking-71-million-request-per-second-ddos-attack
12. The Cloudflare Blog. The New DDoS Landscape 2017-11-23 By Junade Ali. URL: https://blog.cloudflare.com/the-new-ddos-landscape
13. Kubernetes Autoscaling: YoYo Attack Vulnerability and Mitigation. Ronen Ben David, Anat Bremler Barr. URL: https://arxiv.org/abs/2105.00542
14. Xiaoqiong Xu, Jin Li, Hongfang Yu, Long Luo, Xuetao Wei, Gang Sun. Digital Towards Yo-Yo attack mitigation in cloud auto-scaling mechanism. Communications andNetworks,2020,Vol. 6 (3),369–376.https://doi.org/10.1016/j.dcan.2019.07.002
15. Video games company hit by 38-day DDoS attack. *Gold, Steve (21 August 2014).SC Magazine UK. URL:* https://web.archive.org/web/20170201181833/https:/www.scmagazineuk.com/video-games-company-hit-by-38-day-ddos-attack/article/541275
16. 38-Day Long DDoS Siege Amounts to Over 50 Petabits in Bad Traffic. URL: https://news.softpedia.com/news/38-Day-Long-DDoS-Siege-Amounts-to-Over-50-Petabits-in-Bad-Traffic-455722.shtml
17. Slow Read DDoS Attacks. URL: https://www.netscout.com/what-is-ddos/slow-read-attacks
18. Slow Post Attacks. URL: https://www.netscout.com/what-is-ddos/slow-post-attacks
19. TTL Expiry Attack Identification and Mitigation. URL: https://sec.cloudapps.cisco.com/security/center/resources/ttl_expiry_attack.html
20. UDP-Based Amplification Attacks. URL: https://www.cisa.gov/news-events/alerts/2014/01/17/udp-based-amplification-attacks
21. What Is a CC Attack? URL: https://support.huaweicloud.com/en-us/antiddos_faq/antiddos_01_0020.html
22. The method of defence CC attack, Apparatus and system. URL: https://patents.google.com/patent/CN106161451A/en
23. CC (Challenge Collapsar) attack protection method and device. URL: https://patents.google.com/patent/CN106330911A/en
24. Voice over IP. URL: https://www.cse.wustl.edu/~jain/cis788-99/h_8voip.htm
25. Voice over IP: Protocols and Standards, Rakesh Arora. URL: https://www.cse.wustl.edu/~jain/cis788-99/ftp/voip_protocols
26. [Review] MyDoom Virus: The Most Destructive & Fastest Email Worm. URL: https://www.minitool.com/backup-tips/mydoom-virus.html?amp
27. Geoffrey Cheng. Analysis on DDOS tool Stacheldraht v1.666. URL: https://www.giac.org/paper/gcih/229/analysis-ddos-tool-stacheldraht-v1666/102150
28. Fork bomb. URL: http://catb.org/~esr/jargon/html/F/fork-bomb.html
29. The Jargon File, Version 4.2.2, 20 Aug 2000. **Editor**: Eric S. Raymond, Guy L. Steele. URL: https://www.gutenberg.org/cache/epub/3008/pg3008-images.html
30. Slowloris DDoS attack. URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris

31. Slowloris attack. URL: https://www.invicti.com/learn/slowloris-attack
32. Degradation of Service Attack. By Editorial Staff. URL: https://www.devx.com/terms/degradation-of-service-attack/#:~:text=A%20Degradation%20of%20Service%20Attack%20is%20a%20type%20of%20cyber,to%20access%20for%20legitimate%20users
33. Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, Jeffrey Voas. DDoS in the IoT: Mirai and Other Botnets. Computer, 2017, Vol. 50 (7), 80–84. https://doi.org/10.1109/MC.2017.201
34. LAND Attacks. URL: https://www.imperva.com/learn/ddos/land-attacks/#:~:text=A%20LAND%20Attack%20is%20a,processed%20by%20the%20TCP%20stack
35. Understanding LAND Attacks: Risks and Mitigation. URL: https://www.indusface.com/learning/land-attacks
36. Tfreak. URL: https://hackepedia.org/?title=Tfreak
37. What is a Smurf DDoS attack? By Martin Pramatarov. URL: https://www.cloudns.net/blog/what-is-smurf-ddos-attack
38. Permanent Denial-of-Service Attack Sabotages Hardware. By Kelly Jackson Higgins. URL: https://web.archive.org/web/20081208002732/http:/www.darkreading.com/security/management/showArticle.jhtml?articleID=211201088
39. "BrickerBot" Results In Permanent Denial-of-Service. URL: https://www.radware.com/security/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service
40. Prolexic Distributed Denial of Service Attack Alert. URL: https://web.archive.org/web/20070803175513/http:/www.prolexic.com/news/20070514-alert.php
41. Peer-to-peer networks co-opted for DOS attacks. *Robert Lemos. URL:* https://www.theregister.com/2007/05/30/p2p_dos_attacks
42. Denying distributed attacks. *Fredrik Ullner. URL:* https://dcpp.wordpress.com/2007/05/22/denying-distributed-attacks
43. SACK Panic and Other TCP Denial of Service Issues. URL: https://web.archive.org/web/20190619100453/https:/wiki.ubuntu.com/SecurityTeam/KnowledgeBase/SACKPanic
44. CVE-2019-11479. URL: https://web.archive.org/web/20190621224631/https:/cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11479
45. Yu Chen, Kai Hwang and Yu-Kwong Kwok, «Filtering of shrew DDoS attacks in frequency domain,» *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, Sydney, NSW, Australia, 2005, 8–793. https://doi.org/10.1109/LCN.2005.70
46. Vinicius de Miranda Rios, Pedro R M Inacio, Damien Magoni, Mario M Freire. Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey. *IEEE Access*, 2022, Vol. 10, 76648–76668. https://doi.org/10.1109/ACCESS.2022.3191430
47. Kuzmanovic, Aleksandar, Knightly, Edward W. Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants. *ACM. Conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM'03*, August 25–29, 2003, Karlsruhe, Germany, *75–86*. https://doi.org/10.1145/863955.863966
48. CERT Advisory CA-1997-28 IP Denial-of-Service Attacks. URL: https://vuls.cert.org/confluence/display/historical/CERT+Advisory+CA-1997-28+IP+Denial-of-Service+Attacks
49. UPnP Forum. UPnP Specifications Named International Standard for Device Interoperability for IP-based Network Devices. URL: https://web.archive.org/web/20140401035712/http://upnp.org/news/documents/UPnPForum_02052009.pdf
50. New DDoS Attack Method Demands a Fresh Approach to Amplification Assault Mitigation by Avishay Zawoznik, Johnathan Azaria, Igal Zeifman. URL: https://www.imperva.com/blog/archive/new-ddos-attack-method-demands-a-fresh-approach-to-amplification-assault-mitigation

51. Stupidly Simple DDoS Protocol (SSDP) generates 100 Gbps DDoS. By Marek Majkowski. URL: https://blog.cloudflare.com/ssdp-100gbps
52. How does a SSDP Attack work? URL: https://www.cloudflare.com/learning/ddos/ssdp-ddos-attack
53. Stress-Testing the Booter Services, Financially. URL: https://krebsonsecurity.com/2015/08/stress-testing-the-booter-services-financially/
54. ARP Cache Poisoning (Gibson Research Corporation). URL: https://www.grc.com/nat/arp.htm
55. David C. Plummer. An Ethernet Address Resolution Protocol — or — Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. URL: https://datatracker.ietf.org/doc/html/rfc826
56. ARP Vulnerabilities: The Complete Documentation. URL: https://web.archive.org/web/20110305160956/http:/www.l0t3k.org/security/tools/arp/
57. APE — The ARP Poisoning Engine. URL: https://web.archive.org/web/20120709235815/http:/www.megapanzer.com/2012/04/11/ape-the-arp-poisoning-engine/
58. Windows 10 ARP Spoofing with Ettercap and Wireshark. URL: https://cybr.com/cybersecurity-fundamentals-archives/windows-10-arp-spoofing-with-ettercap-and-wireshark/
59. Richard Dezso. How to Perform an ARP Poisoning Attack. May 13, 2024. URL: https://www.stationx.net/how-to-perform-an-arp-poisoning-attack/
60. Fact Sheet: Machine-in-the-Middle Attacks . URL: https://www.internetsociety.org/resources/doc/2020/fact-sheet-machine-in-the-middle-attacks/
61. SQL Injection. URL: https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms161953(v=sql.105)?redirectedfrom=MSDN
62. *Michael Kerner*. How Was SQL Injection Discovered? *URL:* https://www.esecurityplanet.com/networks/how-was-sql-injection-discovered/
63. OWASP. *Blind SQL Injection*. URL: https://owasp.org/www-community/attacks/Blind_SQL_Injection
64. Kirsten S. OWASP. Cross Site Scripting (XSS). URL: https://owasp.org/www-community/attacks/xss/
65. Happy 10th birthday Cross-Site Scripting! URL: https://learn.microsoft.com/en-ca/archive/blogs/dross/happy-10th-birthday-cross-site-scripting
66. 2000 CERT Advisories. URL: https://insights.sei.cmu.edu/documents/507/2000_019_001_496188.pdf
67. *Leyden John*. Facebook poked by XSS flaw. *URL:* https://www.theregister.com/2008/05/23/facebook_xss_flaw/
68. Cross Site Scripting. URL: http://projects.webappsec.org/w/page/13246920/Cross%*20Site*%20Scripting
69. XSS Attack Examples (Cross-Site Scripting Attacks) by Lakshmanan Ganapathy on February 16, 2012. URL: https://www.thegeekstuff.com/2012/02/xss-attack-examples/
70. What is Mutation XSS (mXSS)? URL: https://kpmg.co.il/technologyconsulting/blog/what-is-mutation-xss-mxss
71. Types of XSS. URL: https://owasp.org/www-community/Types_of_Cross-Site_Scripting
72. Symantec Internet Security Threat Report Trends for July–December 06 Vol. 11, 2007. URL: https://docs.broadcom.com/doc/istr-07-march-en

Отримано / Received: 19.12.2024

A.B. Godlevsky, PhD (Phys.-Math.), Senior Researcher,
V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0009-0005-8067-077X
a.godl49@gmail.com

M.K. Morokhovets, PhD (Phys.-Math.), Senior Researcher,
V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0009-0008-2191-8677
marina.morokhovets@gmail.com

N.M. Shchogoleva, Researcher,
V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine,
40, Hlushkova Akad. ave., Kyiv, 03187, Ukraine
https://orcid.org/0009-0004-4700-7426
natashch2904@gmail.com

ANALYSIS OF COMMON TYPES OF NETWORK ATTACKS,
AND FACTORS ENABLING THEIR SUCCESSFUL IMPLEMENTATION

Introduction. A survey of common types of network attacks, i.e. cyberattacks carried out through a computer network, is presented. The purpose of the work is to expose factors enabling attacks to be fulfilled successfully.

Methods. To expose the factors, common types of network attacks were analysed on the basis of sources open to general use.

Results. As a summary, it was found out that attackers have in their arsenal: brute force, malicious software, knowledge on various vulnerabilities and flaws on a victim's side, social engineering tools, and detailed knowledge on how both network equipment and victim's software function. It is significantly that due to their knowledge on how network equipment and applications connected to a network function attackers can realize their malicious intentions while behaving themselves as legitimate users regarding the network protocols and regarding the attacked services and without even looking for flaws in the protective resources or software vulnerabilities on the victim's side. It turned out, that such factor as flaws in network software designing relates every type of attack considered.

Conclusions. Taking into account the results of the analysis, the directions of intensifying the resilience to network attacks are outlined: improving the working out of projects of network software and applications that operates in a network in order to prevent arising vulnerabilities which attackers can exploit; improving the existing computer security systems and developing novel ones; network users' conversance with information security requirements and the threats of social engineering means.

Keywords: *computer network, cyberattack, information security, Internet protocols.*

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІ
INTELLECTUAL INFORMATION TECHNOLOGIES

https://doi.org/10.15407/intechsys.2025.02.081
UDC 004.91

A.H. ADAMCHUK, Student,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37, Beresteyskyi ave., Kyiv, 03056, Ukraine
ann.adamchuk2002@gmail.com

V.I. SUSHCHUK-SLUSARENKO, Senior Lecturer,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37, Beresteyskyi ave., Kyiv, 03056, Ukraine
https://orcid.org/0000-0002-6096-3832
Sushchuk.Viktoriia@lll.kpi.ua

A.I. DYCHKA, PhD (Engineering), Assistant,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37, Beresteyskyi ave., Kyiv, 03056, Ukraine
https://orcid.org/0000-0003-0578-2788
andriydychka@gmail.com

AUTOMATED AUTHORSHIP IDENTIFICATION OF PROGRAM CODE BASED ON A METRIC SYSTEM

The paper reviews existing methods for automated program code authorship attribution and then proposes an original method based on a system of metrics. The proposed method uses a metric system grounded in the "fingerprinting" technique. The metrics reflect the individual stylistic features of a programmer, regardless of the programming language.

Keywords: *metrics, attribution, source code, authorship identification, information system..*

Terminology

Metric — a numerical indicator used to measure and analyze the characteristics of a specific object, system, or process.

Author Profile — a set of characteristics (metrics) that define an individual's coding style.

Cite: Adamchuk A.H., Sushchuk-Slusarenko V.I., Dychka A.I. Automated Authorship Identification of Program Code Based on a Metric System. *Information Technologies and Systems*, Київ, 2025, Том 2 (2), 81–89. https://doi.org/10.15407/intechsys.2025.02.081

© Видавець ВД «Академперіодика» НАН України, 2025. Стаття опублікована на умовах відкритого доступу за ліцензією CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

Cyclomatic Complexity — a software metric that evaluates the number of independent paths through a program.

Introduction and Problem Statement

Determining the authorship of program code is an extremely important aspect of detecting unlawful activities in various domains. In particular, it applies to identifying the authorship of malicious code that has infiltrated an information system through unauthorized access, as well as detecting information theft using specially designed software. Given the continuously growing number of cyber threats, the ability to accurately attribute authorship to malicious programs is critically important for ensuring the security of information systems. Additionally, program code authorship identification plays a key role in detecting copyright violations and ensuring academic integrity, which is especially relevant in the digital era.

Currently, automated methods for determining the authorship of texts written in natural languages are considered well-developed. However, attempts to apply these methods to artificial languages, particularly programming languages, do not yield the desired results. This is due to the fact that artificial languages have a significantly smaller vocabulary and much stricter syntactic rules for constructing statements. Thus, developing a new method for automated authorship identification of program code is a relevant challenge today.

The goal of this study is to improve the accuracy of program code authorship identification by developing a method that can be adapted to any programming language and focuses on the individual coding style of different authors. To achieve this goal, it is necessary to review existing solutions for program code authorship attribution, analyze their advantages and disadvantages, and, based on the findings, develop a new method and implement its software realization.

Literature Review

The most common method for determining the authorship of program code is the n-gram attribution method, which analyzes sequences of n elements consisting of adjacent letters, syllables, or other lexical units in the examined text [1]. This method is independent of a specific programming language since it operates on low-level data. However, it has a significant drawback: the core elements of this method are susceptible to subjective manipulation. Specifically, if an attacker knows the frequency of an author's characteristic n-grams, they can easily forge that author's style.

Metric-based methods are also used to determine the authorship of texts written in artificial languages. These methods differ in the number and type of indicators used to construct metrics. For example, one of the earliest code attribution methods, Holsted's metrics [2], relies on only 16 indicators, which is insufficient for accurate authorship identification.

Another method, *IDENTIFIED* [3, 4], developed for the C++ programming language, utilizes cyclomatic complexity and graph-based metrics. Additionally, this method takes into account comments in the code.

A more recently developed method, program "fingerprinting" [5], includes 48 different metrics. This method works specifically for *Java* programs and achieves an accuracy of approximately 73%. The research suggests that source code metrics can indeed help in identifying authorship. However, a major limitation of such metrics is their dependence on the programming language.

To this review, it is worthwhile to add an analysis of contemporary achievements in this field. In particular, study [6] presents an innovative approach that achieves 93–95% accuracy through:

- an expanded set of metrics including: stylistic formatting features, patterns of language construct usage, and abstract syntax tree characteristics;
- a language-agnostic approach that works with various programming languages, unlike most existing methods focused on specific languages (e.g., Java in the "fingerprinting" method);
- deep analysis of structural code features that are more resistant to intentional modification, significantly complicating attempts to forge authorial style.

It is important to note that this method demonstrates substantially higher accuracy (93–95%) compared to the aforementioned "fingerprinting" method, while also having broader applicability due to its language independence. However, like other metric-based approaches, it requires significant computational resources for analyzing large codebases.

Other modern approaches to code authorship attribution should also be analyzed. For instance, Graph-Based methods (e.g., GraphCodeBERT [7]) utilize analysis of code dependency graphs (DFG) combined with textual features, enabling detection of unique function call patterns. A drawback of such methods is the requirement for preliminary graph construction for each code sample, which complicates analysis of large projects.

Another employed solution is Ensemble Learning [8]. This approach involves combining predictions from multiple classifiers (SVM, Random Forest, CNN), improving accuracy to 94%. Its disadvantages include high computational complexity and the need for repeated model training.

Thus, the aforementioned methods cover scenarios where classical approaches are ineffective (binary analysis, small datasets). However, many of them require specialized computational resources or deep ML expertise.

A Metric System for Automated Authorship Identification of Program Code

For further modification in this study, we selected the metric system defined in the program "fingerprinting" method. This system was expanded by adding new indicators, such as the frequency of logical operators, com-

parison operators, increments and decrements, the presence of prefixes and suffixes in variable names, and others. The number of metrics in the expanded system increased to 65.

Next, all metrics were divided into two groups: basic metrics — adaptable to any programming language, and specialized metrics — those that differ depending on the programming language. For convenience, the basic metrics were further categorized into four groups: keywords, operators, delimiters, identifier names.

The specialized metrics include language-specific keywords and operators for Java, Python, C++, and JavaScript. Below, we examine these metric groups in more detail.

Keywords. This category measures the frequency of different keywords, revealing a programmer's specific preferences. For instance, some developers favor *"if"* statements over *"switch/case"* even when both options are interchangeable, or they might prefer *"for"* loops over *"while"* loops. Another example is the preference for *"try + catch"* blocks instead of handling errors with conditional statements. Analyzing such patterns helps track a developer's unique stylistic choices, particularly their inclination toward certain conditional statements and constructs.

Operators. This category includes metrics that reflect the frequency of operators such as "+", "-", "*", "/", "%", and others. It also examines how a developer uses comparison operators ("==", "!=", ">", "<") and logical operators (&&, ||). Additionally, the use of increment ("++") and decrement ("--") operators versus more explicit expressions (e.g., "x = x + 1") is considered. This category helps determine how a programmer structures mathematical operations, conditions, and logical checks in their code.

Delimeters. This category analyzes the frequency and style of using various delimiters, such as spaces, tabs, and newlines, which affect the structure and readability of the code. These symbols help organize text blocks and highlight nested structures (e.g., conditional statements or loops). The placement of spaces around operators is also taken into account. Some developers prefer writing *"x = y + z"*, while others use a more compact style like *"x=y+z"*.

Additionally, this category considers the use of optional symbols in some programming languages, such as semicolons (";"), as well as different types of quotes and commenting styles. Metrics in this category reveal how carefully a programmer formats their code, ensuring readability and adherence to stylistic conventions.

Identifier names. This group includes metrics that analyze variable, class, and method naming styles. First, it considers naming conventions (*camelCase*, *PascalCase*, etc.). Second, some programmers prefer short names (*i, tmp*) in small functions, while others use descriptive names (*indexCounter, temporaryStorage*) for clarity. Third, it accounts for cultural differences, such as using transliterated words from the developer's native language (*knyga* instead of *book*). Finally, the use of standard prefixes and

suffixes is assessed, such as is, get, and others to indicate boolean variables or getter methods (e.g., *isAvailable*, *getName*). The correct use of suffixes for data structures (*list* for arrays/lists, *map* for dictionaries) is also evaluated. These characteristics can reveal a programmer's experience, education, and habits.

Specialized metrics. Specialized metrics account for unique keywords and operators in each programming language. This is crucial because different languages have unique constructs that reflect their design philosophy. For example, *null* in Java has analogous representations in other languages: *None* in Python, *nullptr* in C++, *null* in JavaScript.

Automated Method for Determining Source Code Authorship

The previously described metrics are applied in the developed method for automated authorship identification of source code to establish an author's stylistic characteristics. The method consists of six stages.

Stage 1. Collecting source code from projects

At this stage, software projects with known authorship are selected to form a reference database, along with one project under investigation that needs analysis. The source code of each selected project serves as the basis for calculating metric values, which will be used for further analysis.

Stage 2. Removing auxiliary files from the project

To ensure accurate analysis and prevent bias in the results, all auxiliary files that do not contain core source code are removed. These may include configuration files (XML, JSON, YAML), compiled files (.class or .o), test files, or other files unrelated to the program's primary functionality.

Stage 3. Calculating basic metrics.

For each project, the values of basic metrics are computed.

Stage 4. Calculating special metrics.

If the project contains source code files in C++, Java, Python, or JavaScript, the special metrics are also computed.

Stage 5. Comparing the investigated project's metrics with other projects' metrics

The basic metrics are compared across projects. If both projects contain special metrics, those are also included in the comparison. The deviation of the investigated project's metric values from those of other projects is calculated.

Stage 6. Presenting the results.

The analysis results are presented as a percentage similarity between the investigated project's metric values and those of other projects. The project with the highest similarity percentage should belong to the same author as the investigated project.

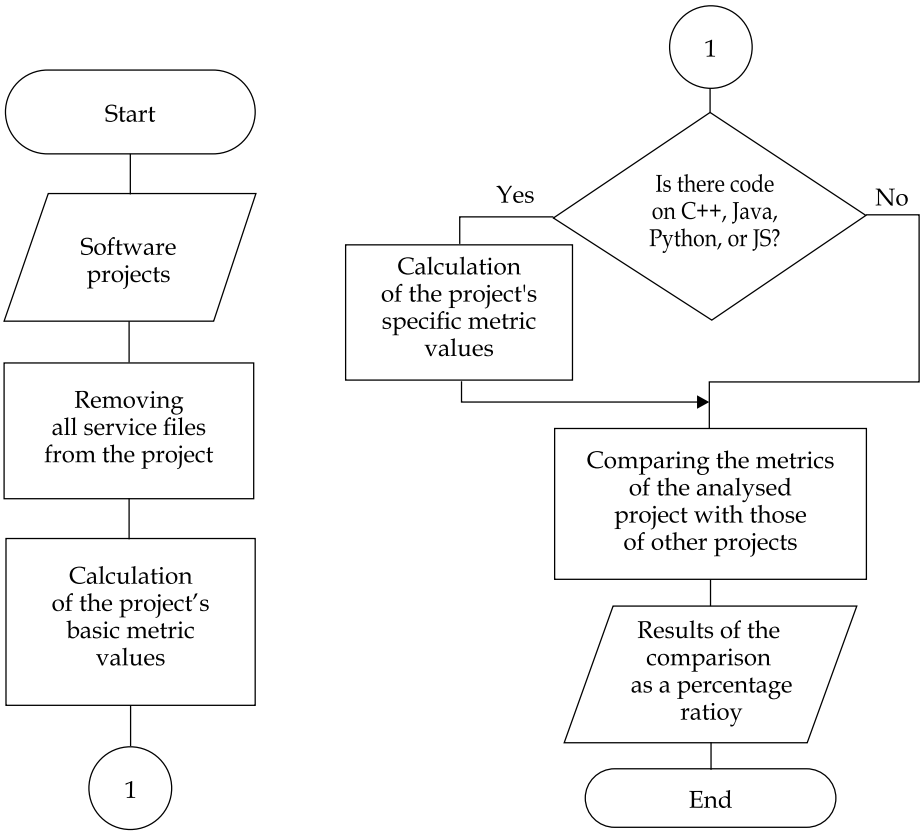

Fig. 1. Sequence of stages of the proposed method

Software Implementation of the Method

In the software implementation of the proposed method, metric values of the investigated project are compared not with the metric values of other projects but with the metric values from author profiles. Initially, for each author's projects — there may be one or more — the method's stages 2 to 4 are applied, meaning auxiliary files are removed, and either basic or both basic and special metrics are calculated. Based on these metric values, an author profile is created: the metric values of all projects by a single author are averaged.

For the development of the software project implementing the proposed method, Python was chosen due to its powerful data processing capabilities. The following libraries were used: NumPy, pandas, and SQLAlchemy. The database management system PostgreSQL is used for database interaction. The server side of the project is built using the Flask framework, which handles request processing, routing, database interaction, and API implementation. The front-end interface is developed using Vue.js, enabling the creation of a dynamic and user-friendly interface.

The web application follows a modular architecture.

The *client-side* includes the following modules:

• API interaction module — responsible for communication between the user interface and the server, sending requests to the server API and receiving responses to be displayed in the graphical interface;

• graphical interface components — implement user interaction features.

The *server-side* includes an API controller module, which processes requests from the client side and calls appropriate modules to perform tasks, along with separate functional modules.

The API controller module contains the following submodules:

• module for working with data about authors, which provides creation, deletion and update of information about project authors;

• module for working with data about projects, which provides addition, deletion of information about projects;

• module for working with metrics, which provides addition, deletion and update of metrics;

• module for working with results, which provides storage of results of determining authorship.

Additional server-side modules:

• a module for calculating the values of basic metrics;

• a module for calculating the values of special metrics;

• a module for calculating the results, in which the deviations of the metric values are calculated and the results are calculated in percentages;

• a module for interacting with the database, which performs the operations of saving, updating and retrieving data from the PostgreSQL database.

The PostgreSQL *database* serves as a storage system for all necessary information, including author data, project data, analysis results, and metric values.

Conclusions

This study reviews existing methods for automated authorship attribution of source code and proposes a new method based on an extended system of metrics.

The proposed method utilizes a metric system based on the "fingerprint» approach to code analysis. These metrics capture the individual stylistic characteristics of a programmer, regardless of the programming language used. By computing metric values for projects, author profiles can be generated, forming the basis for further authorship identification.

As a next step in the project, the authors consider it appropriate to conduct experiments to evaluate the effectiveness of the proposed method using different input data across various programming languages. Additionally, they plan to compare the results obtained with the proposed method against those derived using the Halstead metrics-based approach.

REFERENCES

1. Frantzeskou G., Stamatatos E., Gritzalis S., Chaski C. Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. *International Journal of Digital Evidence,* Trier, Germany, 2007, Vol. 6 (1), 139–148. URL: https://www.researchgate.net/publication/220542545_Identifying_Authorship_by_Byte-Level_N-Grams_The_Source_Code_Author_Profile_SCAP_Method [Accessed 12 Nov. 2024].
2. Frantzeskou G., MacDonell S., Stamatatos E., Georgiou S., Gritzalis S. The significance of user-defined identifiers in Java source code authorship identification *Computer Systems Science and Engineering*. Samos, Greece, 2011.
3. Gray A., Sallis P., MacDonell S. *IDENTIFIED (Integrated Dictionary-based Extraction of Non-language-dependent Token Information for Forensic Identification, Examination, and Discrimination): a dictionary-based system for extracting source code metrics for software forensics.* IEEE Computer Society Press, Dunedin, New Zealand, 1998, 252–259 pp. https://doi.org/10.1109/SEEP.1998.707658
4. Sallis P., Aakjaer A., MacDonell S. *Software Forensics: Old Methods for a New Science.* IEEE Computer Society Press, Dundin, New Zealand, 1998, 367–371 pp. https://doi.org/10.1109/SEEP.1996.534037
5. Ding H., Samadzadeh M.H. Extraction of Java program fingerprints for software authorship identification. *The Journal of Systems and Software,* 2004, Vol. 72 (1), 49–57. https://doi.org/10.1016/S0164-1212(03)00049-9
6. Abuhamad M., AbuHmed T., Mohaisen A., Nyang, D.H. Large-scale and language-oblivious code authorship identification. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security,* 2018, 101–114. https://doi.org/10.1145/3243734.3243738
7. Daya G. et al. GraphCodeBERT: Pre-training Code Representations with Data Flow *Proceedings of the ICLR 2021*. URL: https://www.researchgate.net/publication/344294734_GraphCodeBERT_Pre-training_Code_Representations_with_Data_Flow [Accessed 14 Nov. 2024]
8 Abbasi A., Javed A.R., Iqbal F. et al. Authorship identification using ensemble learning. *Scientific Reports, 2022, Issue 12.* https://doi.org/10.1038/s41598-022-13690-4

Received 06.03.2025

А.Г. Адамчук, студентка,
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»,
Берестейський просп., 37, м. Київ, 03056, Україна
ann.adamchuk2002@gmail.com

В.І. Сущук-Слюсаренко, старш. викладач,
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»,
Берестейський просп. 37, м. Київ, 03056, Україна
https://orcid.org/0000-0002-6096-3832
Sushchuk.Viktoriia@lll.kpi.ua

А.І. Дичка, д-р філософії (техн.), асистент каф.,
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»,
Берестейський просп. 37, м. Київ, 03056, Україна
https://orcid.org/0000-0003-0578-2788
andriydychka@gmail.com

АВТОМАТИЗОВАНЕ ВИЗНАЧЕННЯ АВТОРСТВА ПРОГРАМНОГО КОДУ НА ОСНОВІ СИСТЕМИ МЕТРИК

Вступ. Визначення авторства програмного коду набуває важливого значення в сучасних умовах, коли зростає кількість кіберзагроз, актуалізується захист авторських прав і забезпечення академічної доброчесності. У статті продемонстровано метод автоматизованого визначення авторства програмного забезпечення шляхом аналізу вихідного коду. Метод розширює систему метрик, що застосована в методі «відбитків» програми, що дозволяє працювати з мультимовними проєктами. Ці метрики використовуються для створення «профілів авторів» на основі наявного коду програміста, що дозволяє перевіряти авторство в інших програмних проєктах.

Мета статті. Метою дослідження є підвищення точності визначення авторства програмного коду шляхом розроблення методу, адаптованого під будь-які мови програмування.

Методи. Метод автоматизованого визначення авторства програмного коду розроблено на основі системи метрик, що включають чотири категорії базових метрик: ключові слова, операційні символи, розділові символи та імена ідентифікаторів, а також спеціальні метрики. Для визначення авторства збираються програмні проєкти з достовірною інформацією про авторство, обчислюються значення базових та спеціальних метрик цих проєктів, після чого формуються профілі авторів на основі середніх значень цих метрик. Подальше порівняння значень метрик досліджуваного коду з метриками з профілей авторів дозволяє ідентифікувати автора коду.

Результат. В роботі проведено огляд існуючих методів автоматизованого визначення авторства програмного коду, після чого запропоновано власний метод на основі використання системи метрик. Запропонований метод використовує систему метрик, в основу якої покладено метод «відбитків». Метрики відображають індивідуальні стилістичні особливості програміста незалежно від мови програмування.

Висновки. Матеріали статті будуть корисними при вирішенні задач автоматизованого визначення авторства програмного коду.

Ключові слова: *метрики, атрибуція, вихідний код, визначення авторства, інформаційна система.*

https://doi.org/10.15407/intechsys.2025.02.090
UDC 004.91

M.A. MUZYCHUK, Master's Student,
Computer Systems Software Department of the Applied Mathematics Faculty,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37, Beresteyskyi ave., Kyiv, 03056, Ukraine
maryna.muzychuk@gmail.com
T.M. ZABOLOTNIA, PhD (Engineering), Asociate Professor,
Computer Systems Software Department of the Applied Mathematics Faculty,
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute",
37, Beresteyskyi ave., Kyiv, 03056, Ukraine
https://orcid.org/0000-0001-8570-7571
tetiana.zabolotnia@gmail.com

AUTOMATIC CLASSIFICATION OF UKRAINIAN TEXTS BY FUNCTIONAL STYLES

The proposed multilevel method for classifying Ukrainian texts by functional style combines statistical analysis, keyword analysis, and contextual analysis based on the BERT model, which accounts for semantic and contextual dependencies in the text.

The results support the hypothesis that combining contextual features (generated by BERT) with statistical style parameters yields the highest classification accuracy. This highlights the advantage of the proposed model for tasks requiring high precision and stability in identifying functional text styles.

Keywords: *classification, functional style, stylometry, vectorization, machine learning*

Introduction and Problem Statement

Automatic classification of texts by functional styles is an important task in software engineering, as it enables the automation of text data processing for solving common problems such as information retrieval, document analysis etc. Determining a text's functional style requires analyzing its lexical, grammatical, and stylistic features while considering its context. The main challenge here lies in the significant stylistic diversity of the Ukrainian language. Moreover, within a single text, the boundaries between multiple styles may be blurred. Existing solutions

Cite: Muzychuk M.A., Zabolotnia T.M. Automatic Classification of Ukrainian Texts by Functional Styles. *Information Technologies and Systems*, Київ, 2025, Том 2 (2), 90–97. https://doi.org/10.15407/intechsys.2025.02.090

© Видавець ВД «Академперіодика» НАН України, 2025. Стаття опублікована на умовах відкритого доступу за ліцензією CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)

developed for other languages require substantial adaptation for Ukrainian, posing new challenges for engineers. Therefore, there is a need to develop specialized methods capable of identifying functional styles specifically in Ukrainian texts.

The goal of this study is to improve the accuracy of Ukrainian text classification by functional styles through the development of a method and software for automatic text style identification.

To achieve this goal, the study sets and addresses the following tasks: analyzing existing approaches to automatic text classification; identifying the advantages and disadvantages of current methods in the context of Ukrainian text classification by functional styles; describing the proposed method; outlining the architecture, implementation tools, and key modules of the software developed for the proposed method; evaluating the effectiveness of the developed model and comparing its results with alternative approaches.

Literature Review

Existing approaches to solving the problem of automatic text classification by functional styles can be broadly divided into two groups:

1. Vectorization methods — the purpose of these methods is to represent textual data in a vector format. They include *the keyword method*, *statistical methods, the n-gram method, TF-IDF,* and *BERT (as a vectorization method).* In the context of classifying Ukrainian texts by functional styles, vectorization methods offer **advantages** such as scalability and the ability to capture key textual features, enabling the analysis of stylistic differences. However, their **drawbacks** include sensitivity to the volume and quantity of processed data, as well as the need for adaptation to the Ukrainian language.

2. Machine learning methods — these methods aim to train a model that can later classify input data based on prior training. Among machine learning approaches, notable examples include *the support vector machine (SVM) method, probabilistic methods, ensemble methods and neural network-based techniques*. The **advantage** of these methods for text classification by style lies in their ability to account for complex stylistic structures and their scalability, allowing them to process large datasets without losing efficiency. However, their **disadvantages** include sensitivity to noise, which can lead to model overfitting, as well as the need for significant computational resources [1].

In this study, the authors propose using a pre-trained BERT language model, which has proven to be an effective tool for text analysis due to its ability to consider context at both the word and sentence levels [2]. However, given the specifics of the Ukrainian language and the features of functional styles, standard available BERT models have significant limitations:

• bert-base-ukr-eng-rus-uncased [3], although a specialized Ukrainian model, is trained primarily on publicly available corpora that do not cover a wide variety of functional styles (e.g., literary, academic, journalistic). This limits its ability to accurately recognize stylistic features. Additionally, the model tends to mix stylistically similar styles, such as official-business and academic, due to the lack of clear stylistic segmentation in the data;

• SlavicBERT [4], despite its name, does not support the Ukrainian language. The model is trained mainly on Russian, Bulgarian, Polish, and Czech, making its application to Ukrainian texts not only inefficient but also methodologically incorrect;

• Multilingual BERT (mBERT) [5] — a multilingual model developed by Google to support over 100 languages, including Ukrainian. However, the model is trained on general corpora without accounting for stylistic nuances. As a result, it performs poorly in classification tasks.

Thus, given the identified limitations of existing models, there is a need to adapt BERT for the automatic classification of Ukrainian texts by style. To achieve this, it is necessary to fine-tune the model on specialized corpora covering various functional styles and to incorporate key distinguishing features—such as syntactic patterns, part-of-speech frequency, and specific linguistic expressions. This approach will enable deeper stylistic understanding, improve classification accuracy, and account for both contextual and structural features of each style. Ultimately, it will result in a model capable of effectively processing a wide range of Ukrainian texts while reflecting real-world language practices.

Method for Automatic Classification of Ukrainian Texts by Functional Styles

In the Ukrainian language, four main literary functional styles are distinguished: scientific, artistic, official-business, and journalistic. Currently, there is no universally defined set of features that would allow for an unambiguous determination of a text's style [3]. However, each of these styles has specific linguistic characteristics that can be generalized into three categories: lexicon, syntax, and morphology.

To achieve automatic classification of texts by functional styles, the authors propose using the following key parameters: contextual dependencies within the text, identification of words characteristic of a particular style, and stylometric aspects, i.e., statistical style parameters.

Based on the key aspects described earlier, this study proposes a multi-level method for automatic classification of texts by functional styles, which integrates statistical analysis, keyword analysis, and contextual analysis using neural networks.

Stage 1: Statistical analysis. The first step involves collecting statistical parameters of the text, including: frequency of different parts of speech (nouns, adjectives, verbs, etc.); usage of various verb forms (infinitives, imperative, impersonal forms); sentence structure (simple, complex, asyndetic).

The extracted statistical data is normalized: some features are analyzed relative to the text itself (e.g., the percentage of noun usage compared to the total word count). Others are analyzed in relation to the entire text corpus (e.g., the average sentence length or the number of paragraphs in a text).

Stage 2: Keyword analysis. At this stage, the text undergoes: tokenization, stop-word removal, lemmatization. After preprocessing, the frequency of lemmas characteristic of certain styles is calculated. For journalistic, scientific, and official-business styles, dictionaries of high-frequency words were compiled. For ar-

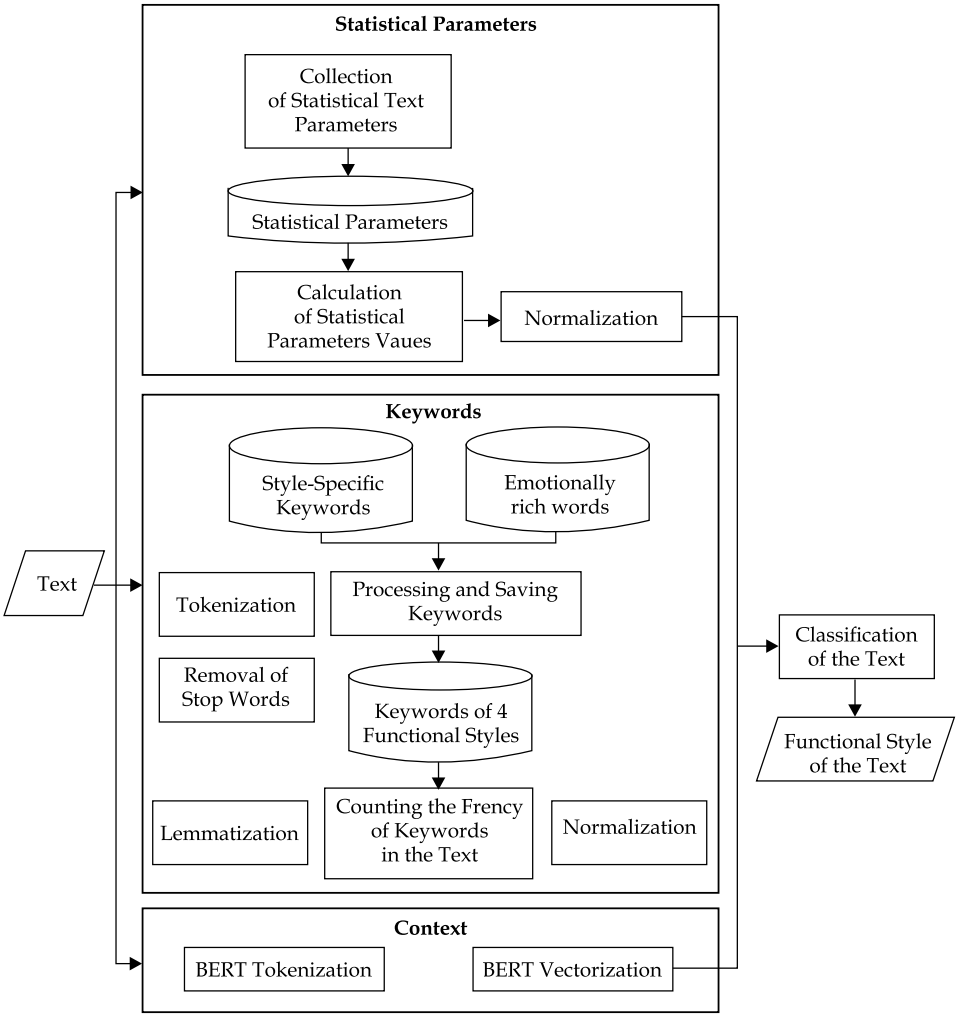

Fig. 1. Schematic representation of the proposed method for automatic text classification by functional styles

tistic style, since it primarily uses general vocabulary, analysis is performed using a set of emotionally charged words, which are distinctive for this style.

Stage 3: Contextual analysis with BERT. The third stage employs a BERT-based model to analyze not only individual words in a text but also relationships between sentences, enabling a deeper understanding of the text's semantics [4]. Through tokenization and vectorization using BERT, a high-dimensional vector representation of the text is obtained, incorporating both semantic and contextual dependencies.

The three main stages of the method can be executed in parallel since the output of each stage does not affect the operation of the other two.

A schematic representation of the proposed method is presented on Figure 1.

As a result of the proposed method, a model is formed that enables the classification of texts by functional style based on the analysis of statistical parameters, lexical features, and context. By combining three stages of analysis, which cover

different aspects of the text, the method ensures adaptability to the specifics of the Ukrainian language.

Implementation of Software and Results

Implementation. The software for automatic classification of Ukrainian texts by functional styles is implemented as a console application. The advantages of a console-based approach include: cross-platform compatibility, allowing it to run on different operating systems without additional configuration; integration capabilities, enabling incorporation into scripts or other systems for automated processing of large text volumes; flexibility in handling textual data.

The following tools were selected for developing the console application:

1. Python as the programming language.
2. Pandas for dataset handling.
3. TensorFlow as the machine learning framework.
4. Pymorphy3 for morphological analysis.
5. NLTK for natural language processing.
6. Redis as an in-memory database management system.
7. SpaCy for text processing and natural language understanding.

The software is designed using a modular architecture, which divides the system into independent modules, each responsible for a specific function. This approach enhances code maintainability and simplifies testing by allowing independent module validation.

The application consists of the following key modules:

1. Root module — the root module, which serves as the entry point for starting the program. This is the primary control module that ensures the coordinated operation of the entire system.

2. Command line processing — a module for processing command-line input, which handles the initial stage of interaction with the user.

3. Statistic parameters analysis — computes statistical characteristics of the text.

4. Key words usage analysis — analyzes the frequency of key terms in the text.

5. Context analysis — generates a vector representation of the text while considering contextual dependencies.

6. Values normalization — normalizes values obtained from statistical and keyword frequency analysis modules.

7. Cache interaction — a module for interacting with the Redis DBMS, which is used for data caching.

8. Filesystem interaction — a module that handles interactions with the file system.

Results. To evaluate the effectiveness of the proposed method for automatic classification of Ukrainian texts by functional styles, the following metrics were used: Accuracy, Precision, Recall, F1-score, and AUC. The choice of these metrics ensures a comprehensive evaluation of the model, as they allow for analysis of classification accuracy, the balance between accuracy and completeness of predictions, as well as the model's ability to reliably rank text styles, contributing to a thorough analysis of the results.

For evaluating the effectiveness of the proposed method, its results were compared with two alternative approaches: the first based on the classical BERT model, and the second using SVM and statistical characteristics of the styles. This comparison allows for testing the hypothesis that integrating contextual features obtained with BERT with statistical style parameters improves the accuracy of text classification by functional styles.

To prepare the textual data required for training the BERT model, this study compiled a large corpus of Ukrainian texts with an even distribution across different functional styles. The source of the corpus was the open-access resource UberText 2.0 [9], which provided over 40,000 text samples — 10,000 per each of the four styles.

For a comprehensive evaluation of the model, an additional dataset with structurally and thematically diverse texts was used, downloaded from an open GitHub repository [10]. Stored in separate .txt files, this corpus was processed to create a validation dataset of 4,000 texts by iterating through the files and sequentially adding their content.

The analysis results indicate that the proposed model significantly outperforms both classical BERT and SVM, which is based on statistical features. The Accuracy metric reached **0.829**, while BERT and SVM showed 0.646 and 0.612, respectively, confirming the higher accuracy in text classification. Similarly, in terms of positive prediction precision (Precision), the proposed model demonstrated **0.780**, surpassing BERT (0.626) and SVM (0.541), indicating a reduction in false positive results. The Recall (0.709) and F1-score (0.729) metrics also exceeded the results of the alternatives, providing a balance between precision and recall.

The AUC score (**0.952**) confirms the model's ability to more accurately recognize text styles, outperforming BERT (0.908) and SVM (0.834). These results prove that the proposed model demonstrates not only high accuracy but also stability in working with different functional styles.

The obtained results confirm the hypothesis that combining contextual features formed with BERT with statistical style parameters provides the highest classification accuracy. The relative improvement in accuracy is **28.39%** compared to BERT and **35.49%** compared to SVM, emphasizing the advantage of the proposed model for tasks that require high accuracy and stability in identifying functional text styles.

Conclusions

The work presents an analysis of existing methods for automatic text classification by functional styles. An overview of the main vectorization methods and machine learning techniques is provided, along with an analysis of their advantages and disadvantages in the context of classifying Ukrainian texts by functional style. A multi-level method is proposed, which combines statistical analysis, keyword analysis, and contextual analysis based on the BERT model, allowing for the consideration of semantic and contextual dependencies within the text. The stages of the proposed method are presented: collecting statistical parameters of the text, identifying characteristic lemmas for styles, and applying contextual analysis to

improve text classification. A console application based on a modular architecture is implemented. The effectiveness of the method is evaluated using Accuracy, Precision, Recall, F1-score, and AUC metrics, demonstrating the superiority of the proposed model over alternative approaches.

REFERENCES

1. What are the advantages and disadvantages of Random Forest? URL: https://aiml.com/what-are-the-advantages-and-disadvantages-of-random-forest/ [Accessed 15 Nov. 2024]
2. Understanding searches better than ever before. URL: https://web.archive.org/web/20210127042834/https://www.blog.google/products/search/search-language-understanding-bert/ [Accessed 15 Nov. 2024]
3. mshamrai/bert-base-ukr-eng-rus-uncased. URL: https://huggingface.co/mshamrai/bert-base-ukr-eng-rus-uncased [Accessed 15 Nov. 2024]
4. Slavic BERT NER. URL: https://github.com/deeppavlov/Slavic-BERT-NER/blob/master/README.md [Accessed 15 Nov. 2024]
5. multilingual.md. URL: https://github.com/google-research/bert/blob/master/multilingual.md [Accessed 15 Nov. 2024]
6. Areshenkov Yu. O. *Stylistics of the Ukrainian language: lecture notes and lesson plans: teaching and methodological manual*. KrDPU, Kryvyy Rih, 2007, 3-th ed., 18p. [In Ukrainian: Арешенков Ю. О. Стилістика української мови: конспект лекцій та плани занять : навч.-метод. посіб.] https://doi.org/10.31812/0564/2140
7. Artistic style as a type of language. Substyles of artistic style. Genres of artistic style. Colors of artistic style. URL: https://studfile.net/preview/5721078/page:36 [Accessed 15 Nov. 2024] [In Ukrainian: Художній стиль як різновид мови. Підстилі художнього стилю. Жанри художнього стилю. Колорити художнього стилю]
8. BERT 101. State Of The Art NLP Model Explained. URL: https://huggingface.co/blog/bert-101 [Accessed 15 Nov. 2024]
9. UberText 2.0. URL: https://lang.org.ua/en/ubertext [Accessed 15 Nov. 2024]
10. Brown corpus of the Ukrainian language. [In Ukrainian: Браунський корпус української мови]

Received 06.03.2025

М.А. Музичук, студентка,
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»,
Берестейський просп. 37, м. Київ, 03056, Україна
maryna.muzychuk@gmail.com

Т.М. Заболотна, канд. техн. наук, доцент,
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»,
Берестейський просп. 37, м. Київ, 03056, Україна
https://orcid.org/0000-0001-8570-7571
tetiana.zabolotnia@gmail.com

АВТОМАТИЧНА КЛАСИФІКАЦІЯ ТЕКСТІВ УКРАЇНСЬКОЮ МОВОЮ ЗА ФУНКЦІЙНИМИ СТИЛЯМИ

Вступ. Автоматична класифікація текстів за функційними стилями є важливим завданням в інженерії програмного забезпечення, оскільки вона дозволяє автоматизувати оброблення текстових даних для ефективного вирішення таких розповсюджених задач, як пошук інформації, аналіз документів тощо. Процес визначення функційного стилю вимагає аналізу лексичних, граматичних та стилістичних особливостей тексту з урахуванням його контексту. Основною складністю тут є те, що українська мова характеризується значною різноманітністю стилістичних варіацій. Наявні рішення,

розроблені для інших мов, потребують суттєвої адаптації для української. У зв'язку з цим постає необхідність розроблення спеціалізованих методів, здатних ідентифікувати функційні стилі саме в українськомовних текстах.

Мета статті. Метою даної роботи є підвищення точності класифікації текстів українською мовою за функційними стилями шляхом розроблення методу та програмного забезпечення для автоматичного визначення стилю тексту.

Методи. Запропонований в статті багаторівневий метод класифікації текстів українською мовою за функційними стилями поєднує статистичний аналіз, аналіз ключових слів та контекстний аналіз на основі моделі *BERT*, що дозволяє враховувати семантичні та контекстуальні залежності в тексті. Етапами запропонованого методу є: збір статистичних параметрів тексту, визначення характерних лем для стилів, а також застосування контекстного аналізу для покращення класифікації текстів. Метод реалізовано в межах консольного застосунку, що базується на модульній архітектурі.

Результат. Запропонована модель значно перевершує як класичну *BERT*, так і *SVM* за метрикою Accuracy: вона досягла 0,829, тоді як *BERT* і *SVM* показали 0,646 і 0,612 відповідно. За точністю позитивних передбачень (Precision) запропонована модель продемонструвала 0,780, випередивши *BERT* (0,626) і *SVM* (0,541), що свідчить про зменшення кількості хибно-позитивних результатів. Показники *Recall* (0,709) і *F*1-score (0,729) також перевищують результати альтернатив, забезпечуючи збалансованість між точністю і повнотою. Показник *AUC* (0,952) підтверджує здатність моделі точніше розпізнавати стилі текстів, перевершуючи значення *BERT* (0,908) і *SVM* (0,834).

Висновки. Отримані результати підтверджують гіпотезу, що поєднання контекстних ознак, сформованих з допомогою *BERT*, із статистичними параметрами стилю забезпечує найвищу точність класифікації. Це підкреслює перевагу запропонованої моделі для задач, які вимагають високої точності та стабільності у визначенні функційних стилів тексту.

Ключові слова: *класифікація, функційний стиль, стилеметрія, векторизація, машинне навчання.*

AUTHOR GUIDELINES

The journal publishes the results of research in the field of computer science, information technologies and systems and their applications in various fields of activity, system analysis, intelligent control, cyber security, biological and medical cybernetics, digital economy and learning in the digital age, etc.

Target audience — scientists, engineers, graduate students and students of higher educational institutions of the relevant specialty.

Requirements for manuscripts

1. 1. The manuscript is accepted in electronic form, if possible — on paper in one copy (language of the article — English (*in priority*) or Ukrainian, manuscript up to 30 pages). The manuscript should contain:

• information about the Authors in English and Ukrainian: Full name, Academic Degree, Academic Title, Position, Affiliations, Postal address of the organization, Direct links to author's ORCID (if necessary, Researcher ID) and E-mail, Author-correspondent and their telephone number (*for contacting the editor*);

• Short Abstract with keywords in paper language, and Extended Abstract with keywords in either Ukrainian for English-language paper or English for Ukrainian-language paper. The text of the Extended Abstract is not less than 1800 characters with spaces, by headings: *Introduction*, *The purpose of the paper*, *Methods, Results, Conclusions, Keywords* (5–8 words);

• *REFERENCES* — a list of sources in English in the order of mention in the text. For Non-English-language sources, citation are translated, the original language is indicated in square brackets, for Ukrainian-language papers information about authors and the title in the original language is given. Examples for design of the References is given below;

• *LITERATURE* — a list of sources in Ukrainian do not use for English-language papers;

• if desired, the authors provide information about the grant or financial support of the research;

• the license agreement is signed automatically when the submission is created in the electronic editorial system.

2. The text of the article should be submitted with the following mandatory headings: *Introduction, Problem Statement / Problem Definition, Objective, Results, and clearly formulated Conclusions*.

Requirements for the text file

File format: *.doc, *.rtf. Applicable styles: Times New Roman font, 12 pt, without a hyphen for a line break, line spacing — 1.5. Paper size: A4, all sides — 2 cm.

Formulas are typed in Formula Editors (preferably Microsoft Equation Editor 3.0. and MathType 6.9b.) Formula Editor options are (10.5; 8.5; 7.5; 14; 10). **The width of formulas is up to 12 cm.**

Figures must be of high quality, they are provided in separate files of appropriate formats (*.png, *.jpg, *.tiff, etc.). **The width of the figures is up to 12 cm.**

Tables are created using a standard text editor built into the Table toolkit. **The width of the table is up to 12 cm.**

КЕРІВНИЦТВО ДЛЯ АВТОРІВ

В журналі друкуємо результати досліджень у сфері інформатики, інформаційних технологій та систем і їх застосувань у різних сферах діяльності, системного аналізу, інтелектуального керування, кібербезпеки, біологічної та медичної кібернетики, цифрової економіки та навчання в цифрову епоху тощо.

Цільова аудиторія — науковці, інженери, аспіранти та студенти вищих навчальних закладів відповідного фаху.

Вимоги до рукописів статей

1. Рукопис приймаємо в електронному виді, за можливості — на папері в одному примірнику (мова статті — англійська (*в пріоритеті*) або українська, рукопис до 30 стор.). Рукопис має містити:

• відомості про авторів англійською та українською мовами: ПІБ, науковий ступінь, вчене звання, посаду, місце роботи, поштову адресу організації, прямі посилання авторських ідентифікаторів ORCID (за потреби Researcher ID) та E-mail, автор-кореспондент із номером телефону (*для зв'язку з редактором*);

• коротку анотацію з ключовими словами мовою статті та розширену анотацію з ключовими словами українською для англомовної статті або ж англійською для українськомовної статті. Текст розширеної анотації не менше 1800 знаків з пробілами, за рубриками: *Introduction / Вступ, The purpose of the paper / Мета роботи, Methods / Методи, Results / Результати, Conclusions / Висновки, Keywords / Ключові слова* (5–8 слів);

• *ЛІТЕРАТУРА* — перелік джерел для українськомовних статей українською мовою оригіналу в порядку згадування в тексті, для неукраїнськомовних джерел посилання даються англійською, в квадратних дужках вказується мова оригіналу;

• *REFERENCES* — перелік джерел англійською мовою в порядку згадування в тексті. Для неангломовних джерел в квадратних дужках вказується мова оригіналу, для українськомовних після двокрапки наводиться інформація про авторів та назва джерела мовою оригіналу. Приклади оформлення переліку посилань наведено далі;

• за бажанням автори надають інформацію про грант або фінансову підтримку дослідження;

• ліцензійний договір підписується автоматично при створені подання в системі електронної редакції.

2. Текст статті подають з обов'язковими рубриками: *Вступ, Постановка завдання / Окреслення проблеми, Мета, Результати,* чітко сформульовані *Висновки*.

Вимоги до текстового файлу

Формат файлу *.doc, *.rtf. Застосовні стилі: шрифт Times New Roman, 12 пт, без переносів, міжрядковий інтервал — 1,5. Формат паперу А4, всі береги — 2 см.

Формули набирають у редакторах формул (бажано Microsoft Equation Editor 3.0. та MathType 6.9b.) Опції редактора формул — (10,5; 8,5; 7,5; 14; 10). **Ширина формул — до 12 см**.

Рисунки мають бути якісними, їх надають окремими файлами відповідних форматів (*.png, *.jpg, *.tiff тощо). **Ширина рисунків — до 12 см**.

Таблиці виконують стандартним вбудованим у інструментарієм «Таблиця» текстового редактора. **Ширина таблиці — до 12 см**.

Examples for design of the References
ЛІТЕРАТУРА* / REFERENCES

The Books / Книги

1. De Vooght E. *Learning to think critically*. Academia Press, Gent , 2025, 168 p. [In Dutch]
2. Ustymenko V.A. *Adaptation of national legislation to EU law: foundations, criteria, sustainability degree.* Akademperiodyka, Kyiv, 2025, 452 p. [In Ukrainian: Устименко В.А. *Адаптація національного законодавства до права Європейського Союзу: основи, критерії, виміри стійкості*]
3. Divan M. J., Johri P., Guim F., Shchemelinin D., Carranza M. *Advances in Image Processing, Reliability, and Artificial Intelligence*. Elsevier, 2025.
4. *Department of Informatics of NAS of Ukraine. Historical and Biographical Directory*. Akademperiodyka, Kyiv, 2017, 286 p. [In Ukrainian: Відділення інформатики НАН України. Історико-біографічний довідник. Академперіодика]

4*. Відділення інформатики НАН України. Історико-біографічний довідник. НАН України. Київ: Академперіодика, 2017, 286 p.

Papers in Periodicals / Статті в періодичних виданнях

5. Zhuoqun Xia, Longfei Huang, Jingjing Tan, Yongbin Yu, Wei Hao, Kejun Long. A lightweight intrusion detection system for connected autonomous vehicles based on ECANet and image encoding. *Journal of Information Security and Applications*, Elsevier, 2025, Vol. 92 (7), Article 104082.
6. ZagorodnyA. G., Khimich O. M., Andon F. I., et al. Implementation of European principles of open science in the National Academy of Sciences of Ukraine. *Visnik Nacionalnoi Academii Nauk Ukrainy*, 2025, Vol. 1, 11–33.

Conferences Materials / Матеріали конференцій

7. Neumannova A. Organizational Culture and Digital Resilience: Competing Values Perspective. *17th IADIS International Conference Information Systems,* Porto, Portugal, 2024, 158-162. URL: https://www.iadisportal.org/digital-,library/organizational-culture-and-digital-resilience-competing-values-perspective [Accessed 20 May. 2025]
8. Husna B. A., Munir R. 3D Traffic Scenes Reconstruction for Autonomous Vehicles Using Gaussian Process Latent Variable Model (GPLVM). *11th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)*, Singapore, Issue 1, 2024, 1-6.

Electronic Sources / Електронні ресурси

9. Information for Authors of Springer Computer Science Proceedings: Instructions for proceedings authors (pdf). URL: https://resource-cms.springernature.com/springer-cms/rest/v1/content/19242230/data/v17 [Accessed Mar. 2025]
10. Cyrillic Gap Analysis. W3C Group Draft Note 02 April 2025. URI: https://www.w3.org/TR/cyrl-gap/ [Accessed 26 Jun. 2025]

УДК 004.6/.9+519+62

Journal *"Information Technologies and Systems"* that is a merger of two academic journals with a long history — Control Systems and Computers journal ISSN (Print) 2706-8145, ISSN (Online) 2706-8153 (published since 1972) and Cybernetics and Computer Engineering journal ISSN (Print) — 2663-2578, ISSN (Online) — 2663-2586 (published since 1965). The organization responsible for publishing the journal is Institute of Information Technologies and Systems of National Academy of Sciences of Ukraine.

The journal publishes original scientific and review papers about fundamental and applied research results of informatics and information technologies, intelligent control and systems, methods and means of information technology support of knowledge, application of the mentioned technologies in various fields of life. Number of Issues is 6 per year. Currently, the journal does not charge any fees to the authors from submission to publication. The papers are an Open Access under the CC BY-NC-ND 4.0 license.

Журнал «Інформаційні технології та системи» є результатом об'єднання двох академічних журналів з багаторічною історією — журналу «Системи керування та обчислювальна техніка» ISSN (друковане видання) 2706-8145, ISSN (онлайн) 2706-8153 (видається з 1972 року) та журналу «Кібернетика та обчислювальна техніка» ISSN (друковане видання) — 2663-2578, ISSN (онлайн) — 2663-2586 (видається з 1965 року). Організацією, відповідальною за видання журналу, є Інститут інформаційних технологій та систем Національної академії наук України.

Журнал публікує оригінальні наукові та оглядові статті про фундаментальні та прикладні результати досліджень інформатики та інформаційних технологій, інтелектуального керування та систем, методів та засобів інформаційно-технологічної підтримки знань, застосування згаданих технологій у різних сферах життя. Кількість випусків — 6 на рік. Наразі журнал не стягує жодних гонорарів з авторів від моменту подання до публікації. Статті знаходяться у відкритому доступі за ліцензією CC BY-NC-ND 4.0.

ISSN 3083-6573 Information Technologies and Systems 2025,№2. 1-100