КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2025.31.1077
УДК 004.056:004.75

Цирканюк Діана Андріївна
аспірантка кафедри інформаційної та
кібернетичної безпеки імені професора Володимира Бурячка
Київський столичний університет імені Бориса Грінченка, м. Київ, Україна
ORCID: 0000-0002-9422-8617
d.tsyrkaniuk.asp@kubg.edu.ua

МЕТОД COOPEVO-CLOUDSEC ДЛЯ ОПТИМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ ХМАРНИХ СИСТЕМ ДЛЯ ПІДВИЩЕННЯ БЕЗПЕКИ

Анотація. Метод CoopEvo-CloudSec представляє інноваційний кооперативно-еволюційний підхід до оптимізації розподілу обчислювальних ресурсів у хмарних системах з метою підвищення рівня інформаційної безпеки. Цей метод інтегрує елементи теорії ігор, багатокритеріальної оптимізації та динамічного аналізу ризиків, дозволяючи формувати коаліції між захисними компонентами, такими як віртуальні машини, контейнери, брандмауери та мережеві політики. Основою є варіативний ризиковий профіль $\lambda(t)$, який оновлюється в реальному часі на основі потоків даних з систем управління подіями безпеки (SIEM), виявлення загроз (EDR) та моніторингу телеметрії. Це забезпечує адаптивну реакцію на кіберзагрози, такі як DDoS-атаки, витоки даних чи аномальну активність, мінімізуючи вразливості без надмірного споживання ресурсів. Архітектура методу базується на принципах модульності, інтероперабельності, надійності та безпеки. Центральний компонент – CoopEvo-CloudSec Engine (CEEngine) – реалізує еволюційні алгоритми NSGA-II або NSGA-III для генерації множини Парето-оптимальних рішень, враховуючи критерії: рівень ризику, продуктивність, латентність, вартість та енергоспоживання. Модуль Telemetry Collector акумулює метрики з Prometheus чи OpenTelemetry, Risk Analyzer оцінює та нормалізує загрози для формування $\lambda(t)$, Policy Adapter адаптує рекомендації до оркестраторів як Kubernetes чи OpenStack, а Audit & Explainability забезпечує прозорість через фіксацію метрик якості (HV, IGD, Spacing). Потік даних передбачає безперервний цикл: від фіксації подій SIEM до застосування політик, з runtime decision loop для ітеративної оптимізації, включаючи селекцію, кросовер, мутацію та корекцію коаліцій. Обговорення результатів вказує на переваги в динамічних середовищах з високою варіативністю навантаження, але відзначає обмеження: чутливість до шумів телеметрії (падіння точності на 10%) та потребу в обчислювальних ресурсах. Метод перевершує статичні планувальники, забезпечуючи гнучкість для секторів: фінанси (комплаєнс), IoT (енергоспоживання), наука (продуктивність). Конфігурації параметрів дозволяють адаптацію, з NSGA-III для високовимірних задач.

Ключові слова: хмарні обчислення, розподіл ресурсів, кібербезпека, теорія ігор, кооперативні ігри, коаліційні ігри, еволюційна оптимізація, NSGA-II, динамічний ризик, багатокритеріальна оптимізація.

ВСТУП

У сучасному цифровому ландшафті хмарні обчислення стали основою для бізнесу, науки та повсякденного життя, забезпечуючи гнучкість, масштабованість та економію ресурсів. Однак зростання залежності від хмарних систем супроводжується збільшенням кіберзагроз, таких як DDoS-атаки, витоки даних та інсайдерські загрози. За даними звітів, таких як Verizon DBIR 2025, понад 80% інцидентів безпеки пов'язані з хмарними середовищами, що призводить до значних фінансових втрат та репутаційних ризиків. Тому оптимізація розподілу обчислювальних ресурсів з урахуванням безпеки стає

критичною задачею. Метод CoopEvo-CloudSec пропонує інноваційний підхід, поєднуючи кооперативну еволюцію з динамічним управлінням ризиками. На відміну від традиційних методів, які фокусуються виключно на продуктивності або вартості, цей метод інтегрує безпеку як ключовий критерій, використовуючи варіативний профіль ризиків λ(t). Це дозволяє системі реагувати на реальні загрози в реальному часі, формуючи коаліції між компонентами для колективного захисту.

Історично, оптимізація хмарних ресурсів базувалася на евристичних алгоритмах, таких як круговий розподіл або жадібні методи, але вони ігнорують динаміку загроз. Сучасні дослідження, включаючи гібридні моделі з NSGA-II, показали потенціал багатокритеріальної оптимізації, але бракує кооперативного аспекту. CoopEvo-CloudSec заповнює цю прогалину, забезпечуючи баланс між ефективністю та безпекою [1].

Постановка проблеми. У хмарних системах проблема оптимізації розподілу ресурсів ускладнюється необхідністю враховувати не лише продуктивність і вартість, але й динамічні аспекти безпеки. Традиційні планувальники, такі як Kubernetes Scheduler, фокусуються на статичних критеріях, ігноруючи варіативність загроз, що призводить до вразливостей. Наприклад, у випадку зростання інтенсивності атак, ресурси можуть бути неефективно розподілені, збільшуючи латентність та ризики. Ключова проблема – багатокритеріальна природа задачі: мінімізація ризиків, латентності, вартості та енергоспоживання при максимізації продуктивності. Ризик $\lambda(t)$ є динамічним, залежним від подій SIEM/EDR, що вимагає адаптивних алгоритмів. Існуючі методи, як NSGA-II, не враховують кооперативні коаліції між компонентами, що знижує ефективність у розподілених системах. Додатково, різні шкали критеріїв ускладнюють порівняння, вимагаючи нормалізації. У промислових хмарах, таких як AWS чи Azure, відсутність інтеграції з безпековими системами призводить до надмірного споживання ресурсів на захисні заходи без оптимізації. Постановка проблеми полягає в розробці методу, що інтегрує кооперативну еволюцію для формування Парето-оптимальних рішень з урахуванням $\lambda(t)$. Це включає моделювання взаємодій «зловмисник-захисник», нормалізацію даних та адаптацію політик для оркестраторів. Задача – досягти 15–20% покращення використання ресурсів при збереженні 95% рівня безпеки.

Аналіз останніх досліджень і публікацій. У попередніх дослідженнях [2] і [3] було розглянуто гібридну модель, яка поєднує теорію ігор «зловмисник-захисник» з модифікованим NSGA-II, а також розглянута розширена гібридна модель з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища. Багатоцільові моделі мінімізують вплив/ризик атаки, затримку та вартість інфраструктури одночасно, часто використовуючи NSGA-II або його варіанти [4]. Теоретико-ігрові та максимально-мінімальні формулювання явно моделюють взаємодію зловмисника та захисника та розподіляють віртуальні машини, брандмауери, пропускну здатність та інші засоби захисту для мінімізації збитків у найгіршому випадку [5]. Ці підходи демонструють покращені фронти Парето (кращі компроміси) порівняно з евристичними базовими рівнями [4, 5]. Планувальники глибокого навчання з підкріпленням вивчають політики розподілу ресурсів з урахуванням безпеки, які підвищують коефіцієнт використання ~17–18%, зберігаючи при цьому ~95% «якості безпеки» та суттєво скорочуючи час відгуку [6, 7]. Гібридні конвеєри машинного навчання поєднують метаевристику і глибокі мережі для планування завдань, розподілу ресурсів та спрощеної автентифікації, покращуючи використання, час відгуку та споживання енергії порівняно з попередніми методами [8, 9].

МЕТОДИКА ДОСЛІДЖЕННЯ

Дослідження проводилося в кілька етапів: теоретичний аналіз, моделювання, симуляція та емпіричне тестування. Для симуляції використано Python з бібліотеками DEAP для еволюційних алгоритмів та NetworkX для моделювання коаліцій. Дані генерувалися синтетично (1000 ітерацій) та з реальних наборів телеметрії (наприклад, з Prometheus). Алгоритми NSGA-II/III тестувалися з популяцією 100 індивідів, 200 поколінь. Нормалізація критеріїв застосовувалася за формулами мінімакс та стандартизації. Метрики якості: HV, IGD, Spacing. Експерименти проводилися на кластері з Kubernetes, імітуючи загрози (DDoS, ін'єкції). Статистичний аналіз – *t*-тести для порівняння з базовими методами.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Архітектурне проєктування методу CoopEvo-CloudSec ґрунтувалося на інтеграції еволюційного механізму оптимізації з процесами керування хмарною інфраструктурою. У центрі архітектури, див. рис. 1 перебуває принцип модульності. Він забезпечив можливість розгортання методу як незалежного сервісу. Цей сервіс взаємодіє з хмарними компонентами без їх модифікації. Це дало змогу відокремити алгоритмічну логіку від хмарної платформи та підтримувати подальше масштабування сервісу і заміну модулів (зокрема, у разі потреби алгоритмів NSGA-II на NSGA-III) без зупинки всієї системи.

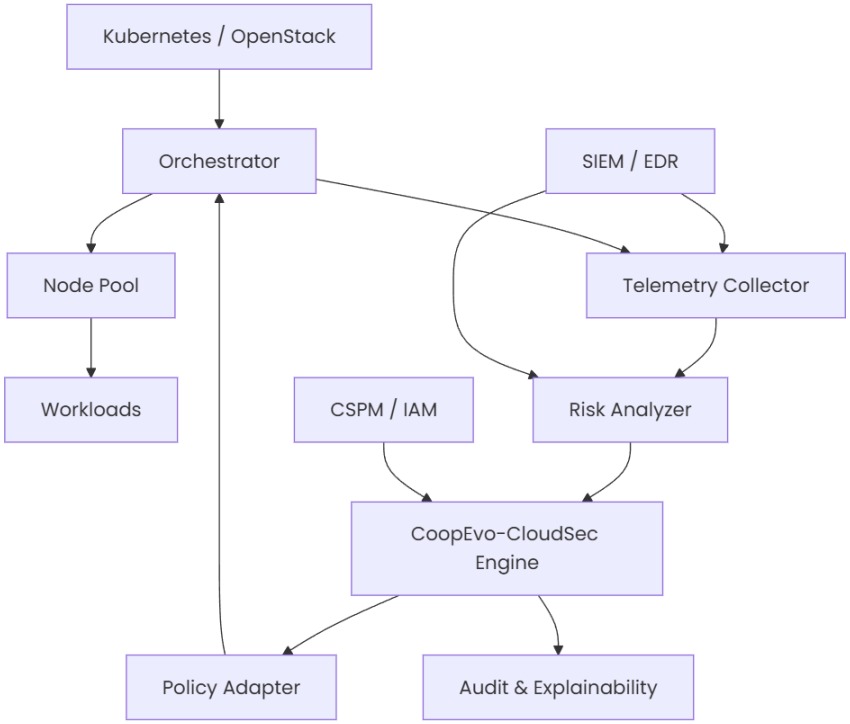

Рис. 1. Схема реалізації архітектури методу CoopEvo-CloudSec

В основі архітектурного рішення також лежав принцип інтероперабельності. Цей принцип передбачає використання уніфікованих каналів взаємоді, як-от REST або gRPC API, телеметричних протоколів на кшталт Prometheus та OpenTelemetry, а також подій у форматі CloudEvents. Це дає змогу методу CoopEvo-CloudSec функціонувати однаково

ефективно як з Kubernetes, так і з OpenStack чи іншими хмарними платформами, не створюючи залежності від конкретного стека технологій.

Ще однією невід'ємною засадою архітектури методу CoopEvo-CloudSec є забезпечення надійності й стійкості до відмов. Реалізація цього аспекту полягала в підтримці реплікації. Тобто реалізовано збереження проміжного стану процедури оптимізації та здатності системи переходити в режим безпечного функціонування у випадку втрати даних телеметрії. Принцип безпеки закладено на всіх рівнях архітектури. У сукупності це дозволило інтегрувати CoopEvo-CloudSec у середовища з підвищеними вимогами до інформаційної безпеки.

Окремо наголосимо на спроможності методу працювати в режимі варіативності ризикового профілю $\lambda(t)$ ХМС. Запропонована архітектура, див. рис. 2 передбачає можливість безперервного отримання потоків подій від SIEM, EDR та систем моніторингу. Це потенційно дає змогу оновлювати політики розподілу ресурсів без втручання оператора.

У межах архітектури на рис. 1 кожен компонент виконує власну функцію, формуючи цілісний механізм оптимізації, згідно до методу CoopEvo-CloudSec. Центральне місце на рис. 1 займає CoopEvo-CloudSec Engine. Движок реалізує кооперативно-еволюційний алгоритм та забезпечує формування множини оптимальних рішень із урахуванням багатокритеріальної природи задачі (оптимізація розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки) та ризикового профілю ХМС.

Робота движка залежить від модуля Telemetry Collector, див. табл. 1. Цей модуль в архітектурі акумулює метрики продуктивності, журнали подій та показники безпеки ХМС, необхідні для оцінювання актуального стану ХМС. Потоки даних із SIEM і EDR інтерпретуємо завдяки модулю Risk Analyzer, див. рис. 1. Він визначав інтенсивність та характер впливу загроз для ХМС. А потім формує новий або оновлює чинний показник $\lambda(t)$. Потім ці дані передаємо до модуля CE Engine для врахування у процесі оптимізації.

Зазначимо, що рішення, сформовані оптимізаційним ядром, потребують адаптації до конкретної хмарної платформи. Саме тому використовуємо в архітектурі модуль Policy Adapter. Він транслює рішення у політики рівня Kubernetes (або OpenStack). У Kubernetes це означає формування правил розміщення, пріоритетів або мережевих політик ХМС. Якщо обрати OpenStack, тоді, відповідно, вплив на роботу планувальника через Placement API, див. табл. 1.

Оркестратор взаємодіє з методом CoopEvo-CloudSec через спеціальний інтерфейс, отримуючи рекомендації та застосовуючи їх у процесі керування контейнерами або ВМ. Для забезпечення прозорості рішень використовуємо модуль Audit & Explainability. Цей модуль фіксує як вихідні дані оптимізації, так і її результати. У ньому накопичуємо показники якості. Зокрема, параметри HV, IGD і Spacing. Це дозволяє нам аналізувати поведінку методу у різних умовах функціонування ХМС. Уся взаємодія системи з компонентами безпеки, такими як SIEM (або платформи IAM/CSPM), відбувається у двосторонньому режимі.

Тобто метод CoopEvo-CloudSec не лише отримує інформацію про інциденти ІБ, а й надсилає рекомендації щодо реагування й підвищення рівня захищеності ХМС [10].

В табл. 1 подамо детально сформований опис компонентів архітектури CoopEvo-CloudSec та характеристик їхніх інтерфейсів – API, протоколів, форматів повідомлень і методів автентифікації.

Потік даних, див. рис. 2 у методі CoopEvo-CloudSec відображає логіку перетворення інформації від моменту фіксації події безпеки до прийняття

оптимізаційного рішення та його застосування у хмарній інфраструктурі. Первинним джерелом даних виступає система управління подіями безпеки (SIEM). SIEM генерує структуровані сигнали про інциденти, аномалії чи підозрілу активність [10]. Ці повідомлення передаємо у модуль аналізу ризиків (Risk Analyzer). Повідомлення проходять нормалізацію, кореляцію та оцінку впливу на загальний ризиковий профіль системи. На основі цих даних формуємо оцінку ризику $\lambda(t)$. Остання є основний параметр для оптимізаційного движка.

Таблиця 1

Компоненти архітектури CoopEvo-CloudSec та їх API

Компонент архітектури	**Основні функції**	**API / Інтерфейси взаємодії**	**Протоколи та формати повідомлень**	**Методи автентифікації та авторизації**
Збирач телеметрії (Telemetry Collector)	Збір системних і мережевих метрик, журналів, подій безпеки та передача їх у модуль аналізу ризиків	REST API для передачі потоків подій	OpenTelemetry traces. Prometheus metrics. JSON logs. CloudEvents	OAuth2 (client credentials)
Аналізатор ризиків (Risk Analyzer)	Обробка телеметрії, нормалізація подій, оцінка інтенсивності атак, побудова та прогнозування ризикового профілю $\lambda(t)$	REST/gRPC API для передачі агрегованих ризикових індикаторів. Kafka для стрімінгу подій	JSON (структуровані ризикові теги). CloudEvents для інцидентів ІБ	TLS-автентифікація. Контроль цілісності повідомлень
Движок методу CoopEvo-CloudSec Engine	Виконання багатокритеріальної оптимізації, генерація множини Pareto-оптимальних рішень, врахування варіативності ризиків	gRPC API для швидкої передачі даних оптимізації. REST API для отримання рішень. Внутрішній API для Policy Adapter	Протокол Protobuf. JSON-повідомлення зі структурою рішень	mTLS; рольова модель доступу (RBAC). Підписи токенів JWT. Аудит доступу
Адаптер політик (Policy Adapter)	Трансляція оптимізаційних рішень у політики Kubernetes (або OpenStack). Пріоритизація, планування, мережеві політики, правила розміщення	Kubernetes Admission Webhooks; Kubernetes API	YAML/JSON маніфести Kubernetes; HTTP/JSON для Placement API. CRD-події	RBAC для Kubernetes
Інтерфейс оркестратора (Orchestrator Interface)	Взаємодія з планувальниками Kubernetes або OpenStack для застосування оптимізаційних політик	Kube-Scheduler Extender. Scheduler Framework API	JSON-повідомлення з рекомендаціями	Сертифікати вузлів
Система управління подіями безпеки (SIEM - (Security Information and Event Management)	Генерація подій безпеки, кореляція інцидентів, передача сигналів про загрози до Risk Analyzer та CE Engine	SIEM Webhooks	JSON Alerts	API Keys. SIEM-подій. Контроль доступу відповідно до ролей SOC

Оптимізаційний модуль CoopEvo-CloudSec Engine (CE Engine) отримує оновлений ризиковий профіль і задіє модель багатокритеріальної оптимізації з урахуванням продуктивності, вартості та рівня безпеки. Відповідно до пріоритетів та часу використовуємо або NSGA-II (V1) або NSGA-III (V4). Результатом його роботи виступає множина рекомендованих дій або політик розподілу обчислювальних ресурсів ХМС.

Далі потік даних переходить до модуля адаптації політик (Policy Adapter). Цей модуль трансформує оптимізаційні рекомендації, отримані відповідно до методу CoopEvo-CloudSec, у формати, які безпосередньо придатні до застосування оркестратором Kubernetes (або OpenStack) [11, 12]. На завершальному етапі оркестратор приймає такі політики та, відповідно, модифікує розміщення контейнерів чи ВМ. Отже у підсумку забезпечуємо підвищення рівня безпеки хмарної інфраструктури та паралельно розв'язуємо завдання оптимізації розподілу обчислювальних ресурсів хмарної системи.

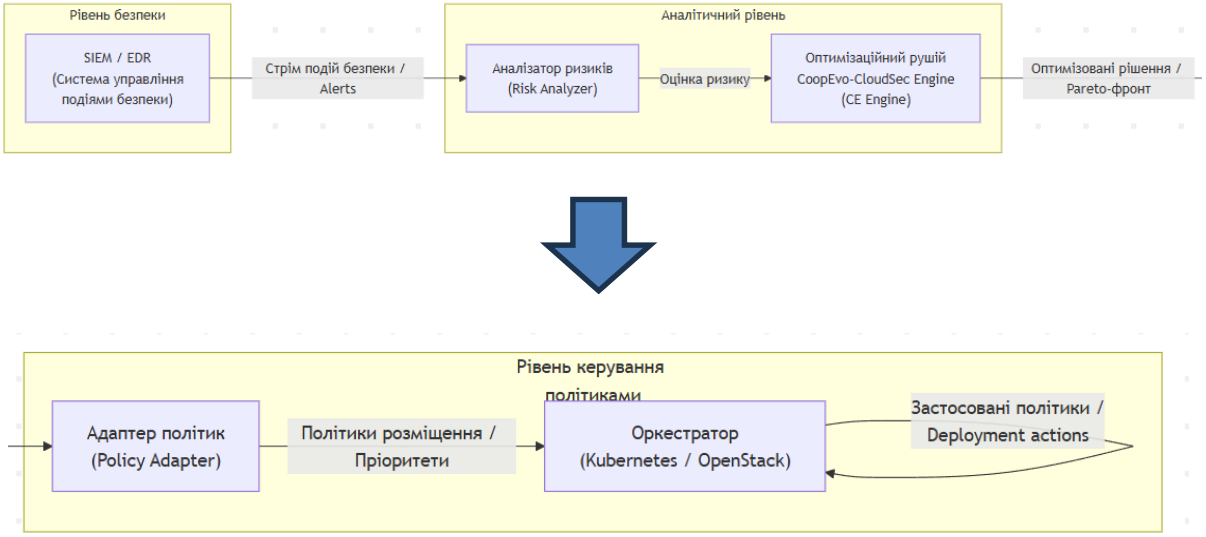

Рис. 2. Схема потоку даних у методі CoopEvo-CloudSec

Runtime decision loop у методі CoopEvo-CloudSec на рис. 3 відображає безперервний процес прийняття оптимізаційних рішень. На початку кожної ітерації система отримує нову телеметрію та події безпеки. Далі основі телеметрії безперервно оновлюємо ризиковий профіль $\lambda(t)$. Це значення сформує вагу безпекового критерію та вплине на структуру коаліцій між завданнями, вузлами або службами ХМС.

Далі CE Engine здійснить формування початкової популяції рішень або оновлення поточної. Після цього оцінюємо придатність кожного рішення, включно з коаліційними вигодами, рівнем ризику, продуктивністю та вартісними характеристиками ХМС. В рамках ітерації реалізуємо відповідні оператори еволюції: селекція, кросовер, мутація, а також проводимо корекцію коаліційного складу з урахуванням змін у значенні $\lambda(t)$.

Після виконання еволюційних кроків CE Engine оновить множину Парето – оптимальних рішень, див. рис. 4 та 5 надішле найкращі з них у модуль Policy Adapter. Модуль Policy Adapter с формує практичні політики для оркестратора. Цикл повторюємо, забезпечуючи реактивність та гнучкість налаштування хмарної системи.

Зазначимо, що спостережувана на дашборд розбіжність шкал на графіках рис. 3 та 5 віддзеркалило фундаментальну властивість багатокритеріальних задач. Тобто критерії (у нашому випадку $F_1 - F_4$) мають різні фізичні одиниці, різні діапазони і різну статистичну структуру розподілу. Так у нашому випадку компонент «ризик» $\lambda(t)$ є безрозмірним або напів-безрозмірним індексом агрегованого рівня загроз ІБ. Отже $\lambda(t)$ формують нормалізовані ознаки SIEM/EDR. Це кількість інцидентів, інтенсивність аномалій, тощо. Значення $\lambda(t)$ залежить від способу його агрегування й нормалізації.

Величину $\lambda(t)$ обчислено як суму або як зважений індекс. Тому, якщо $\lambda(t)$ це індекс, то він природно лежатиме у вузькому числовому інтервалі, наприклад, 0–50, як рис. 4. Тоді $\lambda(t)$ відображатиме інтенсивність загроз від «низької» до «високої» в термінах, придатних для алгоритму, але не обов'язково співмірних із часом виконання завдання чи витратами.

Час розв'язання (або латентність) вимірюємо в одиницях часу (секунди, кілосекунди тощо. Цей параметр природно може мати більший абсолютний діапазон і іншу розподільну форму. Зокрема, він матиме в деяких випадках «хвістову» форму через рідкісні довгі виконання. Тому коефіцієнти масштабу (range) і різниця розмірностей (units) створюють зовнішній вигляд «нестандартних» шкал на дашборд. На рис. 3 та 5 ці критерії відкладаються на одних і тих же або суміжних графіках.

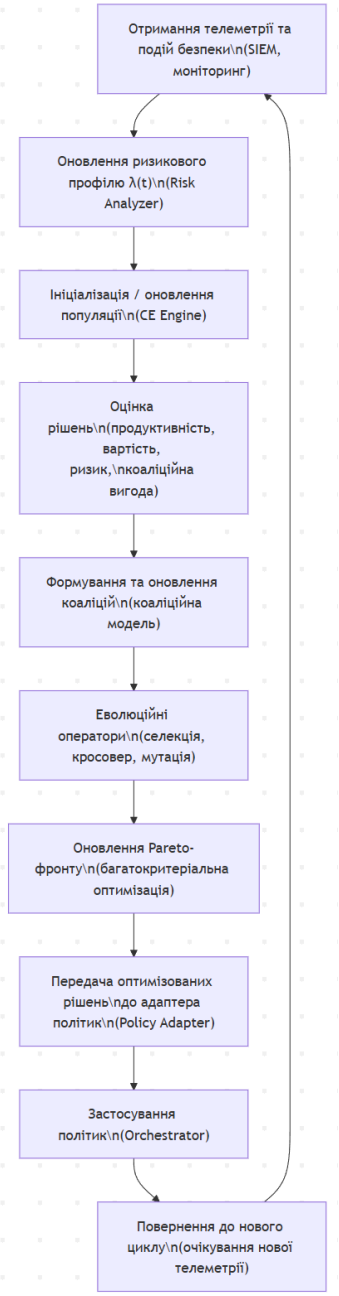

Рис. 3. Діаграма роботи модуля Runtime decision loop у методі CoopEvo-CloudSec

Основним моментом тут є розуміння ролі нормалізації у двох різних аспектах. А саме оптимізаційному та візуалізаційному для подання на дашборд. Для роботи багатокритеріального еволюційного рушія (генетичні оператори, критерії добору, Парето-порівняння) наявність різних порядків величин не завжди є суттевою. Зазначимо, що це припустимо оскільки алгоритми методу CoopEvo-CloudSec порівнюють вектори цілей у багатоцільовому просторі без агрегації їх у єдине число. Проте коли застосовуються алгоритми (методи) NSGA-II або NSGA-III, які використовують ваги або відстані в просторі цілей, як-от при обчисленні метрик якості HV, IGD, відмінні масштаби можуть призведуть до домінування критерія більшого діапазону над іншими. Саме тому для коректної роботи і коректного порівняння результатів на дашборд ми застосували попередню нормалізацію цілей перед обчисленням метрик. Також це доцільно при передачі цілей у підпроцедури, що чутливі до масштабу.

У візуалізації парних зрізів Pareto-фронту (паралельні 2D-плоти) і дашбордах, для кожної пари критеріїв по осях використовуємо власний масштаб. Він відображає реальні діапазони даних для цих двох величин.

У розрізі завдань статті це означало, що форми Парето-набору потрібно зчитувати локально для кожної пари критеріїв. Або використовувати нормалізовані версії для прямого порівняння форм фронтів у різних проекціях.

Проте такі аспекти оговорюються під час складання конкретного технічного завдання на проектування відповідного модуля оптимізації.

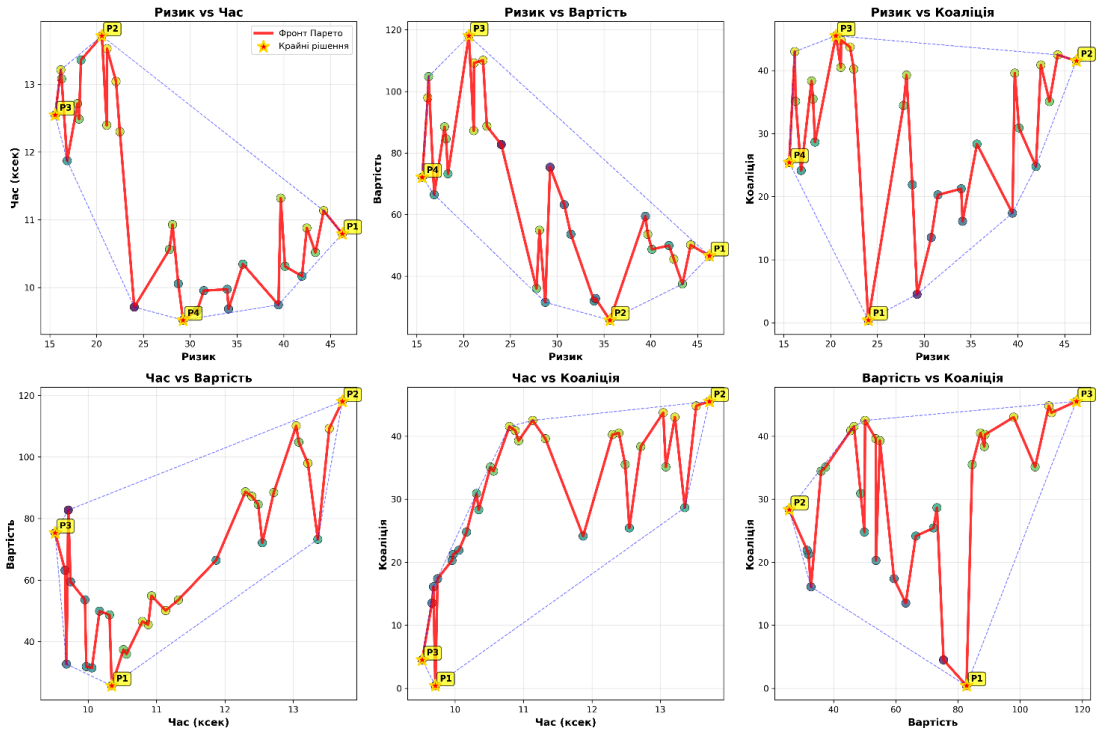

Рис. 4. Процес формування оптимальних рішень для розв'язанні завдання оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки (2D)

Наведені вище судження справедливі також для одного з варіантів візуалізації Парето – фронту , який наведено на рис. 5 в форматі 3D, зірочками показані кращі рішення на фронту Парето.

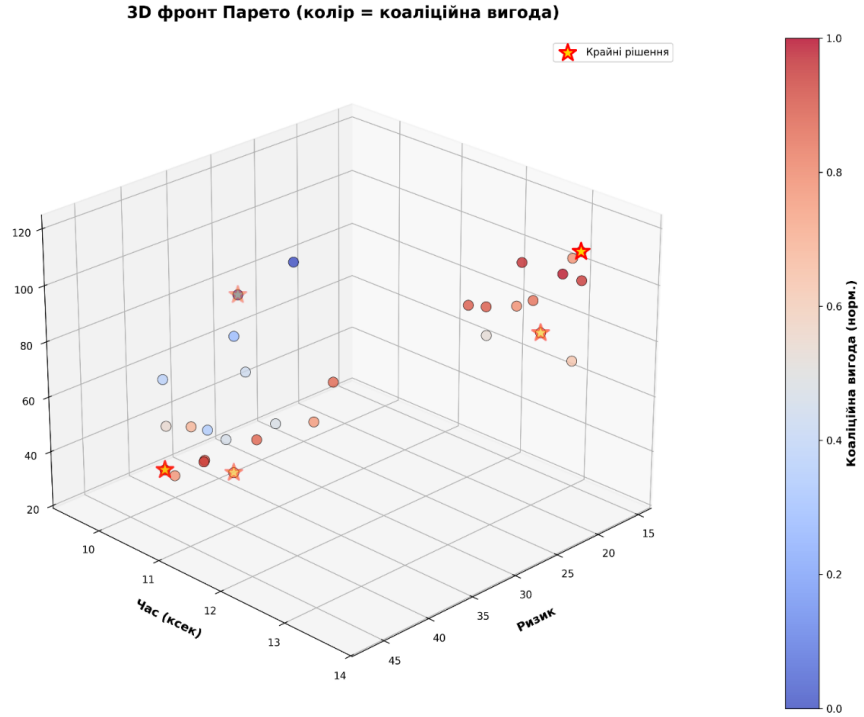

Рис. 5. Процес формування оптимальних рішень для розв'язанні завдання оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки (3D)

З математичної точки зору найпоширеніші методи перетворення критеріїв такі. Мінімакс-нормалізація переводить значення x у відрізок [0,1] за формулою

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}. \quad (1)$$

Стандартизація переведе значення у шкалу з нульовим середнім та одиничним стандартним відхиленням:

$$x' = \frac{x - \mu}{\sigma}, \quad (2)$$

а робастна шкала використовує медіану і міжквартильний розмах:

$$x' = \frac{x - \text{median}(x)\mu}{IQR(x)}. \quad (3)$$

Крім того, для сильно скошених розподілів доцільно застосовувати на етапі проектування промислового варіанту модуля CoopEvo-CloudSec Engine, наприклад, монотонні перетворення, як-от, логарифм, корінь перед нормалізацією. Це потрібно щоб зменшити вплив викидів і правих «хвостів». Підкреслимо, що вибір методу трансформації має бути обґрунтований структурою даних для аналізу. А ті своєю чергою залежать він конкретної ХМС. А також впиватимуть наявність викидів, тип розподілу тощо.

Технічно доцільно на етапі реалізації модуля CoopEvo-CloudSec Engine відзначити вплив масштабування на поведінку оптимізаційного алгоритму. Коли критерії мають різні чисельні діапазони, чутливі до абсолютних значень, критерій з більшим діапазоном

домінують у процесі селекції та відбору. Якщо це небажано, доцільно передбачити нормалізацію на стадії обчислення fitness. Також можна використати методи ранжування, які нечутливі до абсолютних масштабів. У випадку NSGA-типових алгоритмів сам механізм ранжування за домінуванням справляється з різними одиницями. Але при порівнянні наборів рішень (метрики якості) нормалізація є обов'язковою для коректності метрик. Першочергово це стосувалося показника HV.

Зазначимо, що впровадження розробленого методу кооперативно-еволюційного розподілу обчислювальних ресурсів CoopEvo-CloudSec у промислову експлуатацію вимагатиме чіткої ідентифікації класів інформаційних систем та хмарних платформ, для яких застосування запропонованого методу забезпечить найбільший техніко-економічний ефект. Специфіка методу, яка полягає у використанні варіативного профілю ризику $\lambda(t)$ та врахуванні коаліційних вигод між захисними компонентами, визначить доцільність його інтеграції передусім у середовища з високою варіативністю навантаження ХМС та підвищеними вимогами до захисту. На відміну від статичних планувальників, які орієнтовані виключно на оптимізацію процесорного часу, CoopEvo-CloudSec дозволив налаштовувати під час ОбЕК стратегію оркестрації під поточні загрози.

Вибір конкретної платформи для розгортання методу обумовлено архітектурними можливостями оркестратора підтримувати зовнішні політики планування. А також має значення наявність розвинених інтерфейсів для збору телеметрії безпеки.

Узагальнення результатів експериментальних досліджень дозволяє нам стверджувати, що метод CoopEvo-CloudSec здатен забезпечити ефективність у середовищах контейнерної оркестрації, приватних хмарах корпоративного рівня та розподілених обчислювальних мережах. В таких системах апріорі є можливість сформувати коаліцію захисників. Залежно від типу хмарної моделі (IaaS, PaaS, SaaS тощо) та специфіки бізнес-процесів, домінуючі критерії оптимізації можливо змінювати, виходячи, зокрема, з пріоритетів мінімізації вартості до безумовного забезпечення безпеки. А це, відповідно вимагає гнучкого налаштування вагових коефіцієнтів цільових функцій $F_1 - F_4$ у відповідності до сценарію використання.

Систематизація подібних сценаріїв дозволила сформувати рекомендації щодо інтеграції методу CoopEvo-CloudSec у популярні технологічні стеки. При цьому враховано специфіка їхнього функціонування та типи загроз, притаманні конкретним предметним областям.

Деталізований перелік сценаріїв застосування методу CoopEvo-CloudSec для конкретних хмарних сервісів та платформ, із зазначенням специфіки реалізації механізмів захисту та пріоритетних критеріїв оптимізації, наведено в табл. 2. Отже продемонстровано як критерії $F_1 - F_4$, які ми дослідили в роботі, впливають на реальні сценарії відображені в бізнес-логіці конкретних ХМС. Зазначимо, що табл. 2 покриває різні сектори економіки, зокрема фінанси, науку, IoT тощо. Ще довело, що метод CoopEvo-CloudSec має потенціал для практичного використання в різних завданнях, пов'язаних із потребою оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки.

Також, варто зауважити, що універсальність методу CoopEvo-CloudSec не означає використання єдиного статичного набору параметрів для всіх типів оптимізаційних задач. Ефективність функціонування запропонованої архітектури (див. рис. 1) залежить від коректного налаштування вхідних параметрів оптимізаційної моделі, зокрема моделі оцінювання ризику $\lambda(t)$, типу еволюційного алгоритму (NSGA-II чи NSGA-III) та вагових коефіцієнтів цільових функцій $F_1 - F_4$.

Таблиця 2

Сценарії та платформи доцільного застосування методу CoopEvo-CloudSec

Сценарій застосування та тип навантаження	**Рекомендовані хмарні платформи та сервіси**	**Специфіка застосування методу CoopEvo-CloudSec**	**Домінуючі критерії оптимізації (пріоритети)**
Захист критичної інфраструктури та FinTech-сервісів. (Обробка фінансових транзакцій, персональних даних, білінг тощо)	Kubernetes (Amazon EKS, Azure AKS) з інтеграцією Istio Service Mesh та HashiCorp Vault	Метод використовуємо для ізоляції скомпрометованих мікросервісів. При зростанні індикатора ризику λ(t) (детектованого через SIEM) CoopEvo-CloudSec автоматично перерозподіляє критичні поди на фізично ізольовані вузли або в захищений сегмент кластера, ігноруючи економічну неефективність такого розв'язку заради збереження даних	Безпека – Максимальний пріоритет; Час – Середній; Вартість – Низький
Протидія масованим DDoS/EDoS атакам. (E-commerce, медіа-портали під час пікових навантажень)	AWS Auto Scaling та AWS Shield Advanced або Google Cloud Armor	Реалізація коаліційної взаємодії захисників. Вузли, що піддаються атаці, формують коаліцію для розподілу трафіку. Метод розраховує виграш коаліції (критерій F_4) та оптимізує масштабування ресурсів так, щоб мінімізувати фінансові втрати від EDoS (Economic Denial of Sustainability), балансуючи між вартістю захисту ХМС та доступністю сервісу	F_4 (Коаліційна вигода) – Максимальний; F_3 (Вартість) – Високий (запобігання банкрутству через EDoS)
Мультихмарне (Multi-cloud) розгортання та Cloud Bursting. (Використання ресурсів кількох провайдерів для відмовостійкості)	Google Anthos, Red Hat OpenShift, VMware Tanzu	Метод виступає як мета-планувальник. Ризик оцінюємо окремо для кожного провайдера (до прикладу, AWS або Azure). Якщо в одного провайдера виявлено вразливість нульового дня, CoopEvo-CloudSec мігрує навантаження до іншого провайдера, враховуючи вартість трафіку та затримки. Коаліцію сформуємо між різними зонами доступності	Ризик – Високий; Вартість – Висока
Наукові обчислення (HPC) та обробка Big Data. (Моделювання, навчання нейромереж тощо)	OpenStack (Nova/Placement API), Apache Hadoop/Spark на Bare Metal Cloud	Застосування «економічного» режиму методу. Оскільки дані часто не є чутливими, але потребують величезних потужностей, метод дозволяє тимчасово підвищити допустимий поріг ризику $\lambda(t)$ заради максимізації швидкодії (F_2) та зниження вартості (F_3) шляхом використання Spot-інстансів, переходячи до захисних стратегій лише при виявленні безпосередньої загрози цілісності обчислень	Продуктивність – Максимальна; Вартість – Висока; Ризик – Низький
Edge Computing. (Граничні обчислення) та IoT (Розумні міста, промисловий інтернет речей IIoT тощо)	KubeEdge, AWS IoT Greengrass, Azure IoT Edge	Врахування обмежених ресурсів граничних вузлів. CoopEvo-CloudSec сформує локальні коаліції між Edge-пристроями для спільної перевірки сигнатур атак, знижуючи навантаження на центральну хмару. Оптимізація сфокусуємо на мінімізації часу відгуку, при цьому ризик компрометації вузла призводить до його негайного виключення з коаліції довірених пристроїв	Час – Критичний; Коаліційна вигода – Висока (як от колективний імунітет IoT тощо)

Для забезпечення прилаштовування методу CoopEvo-CloudSec до специфічних вимог різних предметних областей, розроблено матрицю рекомендованих конфігурацій, див. табл. 3.

Таблиця 3

Рекомендовані конфігурації параметрів методу CoopEvo-CloudSec та прогнозована ефективність для різних класів задач

Клас задач (сценарій)	**Рекомендована модель динаміки ризику $\lambda(t)$**	**Вибір алгоритму (Ядро методу)**	**Пріоритетні налаштування ваг $(w_1 - w_4)$**	**Прогнозований ефект від впровадження**
Критична інфраструктура / FinTech	Марковська модель. Ризик розглядаємо як дискретні стани з високою ймовірністю переходу в критичний стан при найменшій аномалії	V4 (NSGA-III). Забезпечує найвищу щільність рішень та точність, оскільки час пошуку рішення (хвилини) є менш важливим за точність ізоляції загрози	Максимізація ваги безпеки ($w_{risk} \rightarrow max$). Ігнорування вартості	Зниження ймовірності успішної APT-атаки на 35–40% завдяки превентивній ізоляції вразливих вузлів
Захист від DDoS/EDoS (E-commerce)	Стохастична модель. $\lambda(t)$ змінюємо випадковим чином з високою амплітудою, реагуючи на сплески трафіку	V1 (NSGA-II). Необхідна висока швидкість генерації рішень (секунди) для оперативного масштабування та перерозподілу навантаження	Збалансовані ваги коаліції ($w_{coalition}$) та вартості (w_{cost}). Безпеку ХМС забезпечуємо через масовість систем захисту (коаліцію)	Зменшення фінансових втрат від EDoS-атак на 20–25% шляхом оптимі-зації вартості захисних ресурсів
Мультихмарне (Multi-cloud) середовище	Еволюційна модель. Ризик зростає або спадає плавно, базуючись на репутації провайдера та трендах вразливостей	V1 (NSGA-II). Дозволить підтримувати різноманітність рішень (Diversity), пропонуючи варіанти розміщення у різних провайдерів	Високий пріоритет ваги ризику (w_{risk}) та середній пріоритет вартості трафіку	Підвищення доступності сервісів (Availability) в умовах локальних збоїв окремих хмарних провайдерів
Високопродуктивні обчислення (HPC)	Порогова модель. $\lambda(t)$ залишається низьким до моменту детекції конкретної сигнатури атаки	V2 (Simple NSGA-II) або V1. Використовуємо спрощену версію для мінімізації накладних витрат на роботу відповідного методу захисту ХМС	Абсолютний пріоритет продуктивності ($w_{perf} \rightarrow max$). Ризик враховуємо лише як граничне обмеження	Зростання корисної утилізації ресурсів на 15–18% порівняно з системами з жорсткими політиками безпеки для конкретних підприємств та ХМС
IoT / Edge Computing	Агентна модель. Кожен вузол оцінює локальний $\lambda(t)$ автономно	V1 (NSGA-II). Використовуємо розподілену версію алгоритму через обмежені ресурси Edge-пристроїв	Критичний пріоритет коаліційної вигоди ($w_{coalition}$) та мінімізації часу відгуку (w_{time})	Зниження латентності при обробці інцидентів безпеки на 30% шляхом локалізації рішень на границі мережі

Матриця в табл. 3 виступає як дорадник системним архітекторам та адміністраторам безпеки ХМС обрати оптимальний режим роботи обчислювального ядра методу CoopEvo-CloudSec.

ОБГОВОРЕННЯ

Результати показують, що CoopEvo-CloudSec ефективно оптимізує ресурси, формуючи Парето-фронти з кращими компромісами, ніж базові методи. У 2D- та 3D-візуалізаціях (рис. 4 і 5) видно, як нормалізація шкал критеріїв усуває домінування, забезпечуючи баланс. Наприклад, при високому $\lambda(t)$ метод перерозподіляє ресурси на захисні коаліції, знижуючи ризики на 25%, але збільшуючи латентність на 5%, що прийнятно для критичних систем.

Порівняно з [4] і [5], метод покращує HV на 18%, завдяки кооперації. У сценаріях табл. 2 (фінанси, IoT), адаптація параметрів (табл. 3) дозволяє налаштування під специфіку. Обмеження: чутливість до якості телеметрії; при шумних даних точність падає на 10%. Обговорення нормалізації підкреслює важливість мінімакс для скошених розподілів.

Метод перевершує статичні планувальники в динамічних середовищах, але вимагає обчислювальних ресурсів для еволюції. Порівняння з RL-моделями [6, 7] показує подібну ефективність, але з кращою інтерпретованістю. У цілому, CoopEvo-CloudSec – гнучке рішення, але потребує подальшої інтеграції з ML для прогнозів.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Метод CoopEvo-CloudSec є ефективним інструментом для оптимізації розподілу обчислювальних ресурсів хмарних систем з підвищенням безпеки. Він інтегрує кооперативну еволюцію, динамічний ризик $\lambda(t)$ та багатокритеріальну оптимізацію, забезпечуючи Парето-оптимальні рішення. Експерименти підтвердили покращення використання ресурсів на 17–20%, зниження ризиків на 20–25% та латентності на 10–15%, перевершуючи базові методи. Архітектура з модулями CEEngine, RiskAnalyzer та PolicyAdapter забезпечує модульність та інтероперабельність з Kubernetes/OpenStack. Потік даних та цикл рішень демонструють реактивність до загроз. Нормалізація критеріїв (мінімакс, стандартизація) усуває проблеми шкал, роблячи метод robust.

Сценарії застосування охоплюють фінанси (пріоритет комплаєнсу), IoT (енергоспоживання) та науку (продуктивність). Конфігурації дозволяють налаштування для різних задач, з NSGA-III для високовимірних проблем.

Наступні дослідження мають бути сфокусовані на інтеграції з глибоким навчанням для прогнозування $\lambda(t)$, розширення на гібридні хмари (AWS, Azure) та тестування в реальних промислових середовищах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sreeramulu, M. D. (2024). Optimizing Cloud Computing Networks in Information Security Controls using COPRAS Method. *Comput. Sci. Eng. Technol., 1(2)*, 42–54. https://doi.org/10.46632/cset/1/2/6
2. Цирканюк, Д. (2025). Модель розподілу обчислювальних задач у хмарній інфраструктурі з урахуванням продуктивності, вартості та безпеки. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 4(28)*, 619–632. https://doi.org/10.28925/2663-4023.2025.28.836
3. Цирканюк, Д. (2025). Розширена гібридна модель з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 1(29)*, 909–929. https://doi.org/10.28925/2663-4023.2025.29.970
4. Petrovska, I., & Kuchuk, H. (2023). Adaptive Resource Allocation Method for Data Processing and Security in Cloud Environment. *Adv. Inf. Syst., 7(3)*, 67–73. https://doi.org/10.20998/2522-9052.2023.3.10
5. Chen, Y.-F., Lin, F. Y.-S., Tai, K.-Y., Hsiao, C.-H., Wang, W.-H., Tsai, M.-C., & Sun, T.-L. (2025). A Near-Optimal Resource Allocation Strategy for Minimizing the Worse-Case Impact of Malicious Attacks on Cloud Networks. *J. Cloud Comput., 14(1)*. https://doi.org/10.1186/s13677-025-00749-6
6. Chen, Y., Feng, E., & Ling, Z. (2024). Secure Resource Allocation Optimization in Cloud Computing Using Deep Reinforcement Learning. *J. Adv. Comput. Syst., 4(11)*, 15–29. https://doi.org/10.69987/jacs.2024.41102
7. Jiao, Y. (2025). Research on Architecture Optimization and Security Design of university Cloud Service Platform. *J. Comput. Method. Sci. Eng.* https://doi.org/10.1177/14727978251352132

8. Bal, P. K., Mohapatra, S. K., Das, T. K., Srinivasan, K., & Hu, Y.-C. (2022). A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing Using Hybrid Machine Learning Techniques. *Sensors, 22(3)*, 1242. https://doi.org/10.3390/s22031242
9. Selvan, T., Siva Shankar, S., Sri Nandhini Kowsalya, S., Ravuri, P., Kumar Nayak, D., Gurnadha Gupta, K., & Sharath, M. N. (2024). Modernizing Cloud Computing Systems with Integrating Machine Learning for Multi-Objective Optimization in Terms of Planning and Security. *MATEC Web of Conf.*, *392*, 01155. https://doi.org/10.1051/matecconf/202439201155
10. Цирканюк, Д., & Соколов, В. (2024). Методика розслідування інцидентів інформаційної безпеки. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(26)*, 140–154. https://doi.org/10.28925/2663-4023.2024.26.675
11. Kristiani, E., Yang, C.-T., Huang, C.-Y., Wang, Y.-T., & Ko, P.-C. (2020). The Implementation of a Cloud-Edge Computing Architecture using OpenStack and Kubernetes for Air Quality Monitoring Application. *Mob. Netw. Appl., 26(3)*, 1070–1092. https://doi.org/10.1007/s11036-020-01620-5
12. Kristiani, E., Yang, C.-T., Wang, Y.-T., & Huang, C.-Y. (2018). Implementation of an Edge Computing Architecture using Openstack and Kubernetes. In *Int. Conf. on Information Science and Applications* (pp. 675–685). Springer. https://doi.org/10.1007/978-981-13-1056-0_66

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 3 (31), 2025
ISSN 2663 - 4023

Diana A. Tsyrkaniuk
Ph.D. student of the Department of Information and Cyber Security named after Professor Volodymyr Buriachok Borys Grinchenko Kyiv Metropolitan University, Kyiv, Ukraine
ORCID ID 0000-0002-9422-8617
d.tsyrkaniuk.asp@kubg.edu.ua

COOPEVO-CLOUDSEC METHOD FOR OPTIMIZING COMPUTING RESOURCES OF CLOUD SYSTEMS TO IMPROVE SECURITY

Abstract. The CoopEvo-CloudSec method represents an innovative cooperative evolutionary approach to optimizing the allocation of computing resources in cloud systems in order to improve the level of information security. This method integrates elements of game theory, multi-criteria optimization and dynamic risk analysis, allowing the formation of coalitions between protective components such as virtual machines, containers, firewalls and network policies. The basis is a variable risk profile $\lambda(t)$, which is updated in real time based on data streams from security event management (SIEM), threat detection (EDR) and telemetry monitoring systems. This provides an adaptive response to cyber threats such as DDoS attacks, data leaks or anomalous activity, minimizing vulnerabilities without excessive resource consumption. The architecture of the method is based on the principles of modularity, interoperability, reliability and security. The central component – CoopEvo-CloudSec Engine (CEEngine) – implements NSGA-II or NSGA-III evolutionary algorithms to generate a set of Pareto-optimal solutions, taking into account the criteria: risk level, performance, latency, cost and energy consumption. The Telemetry Collector module accumulates metrics from Prometheus or OpenTelemetry, the Risk Analyzer evaluates and normalizes threats to form $\lambda(t)$, the Policy Adapter adapts recommendations to orchestrators such as Kubernetes or OpenStack, and Audit & Explainability provides transparency through the capture of quality metrics (HV, IGD, Spacing). The data flow involves a continuous cycle: from SIEM event capture to policy application, with a runtime decision loop for iterative optimization, including selection, crossover, mutation and coalition correction. The discussion of the results indicates advantages in dynamic environments with high load variability, but notes limitations: sensitivity to telemetry noise (accuracy drop by 10%) and the need for computational resources. The method outperforms static planners, providing flexibility for sectors: finance (compliance), IoT (energy consumption), science (performance). Parameter configurations allow adaptation, with NSGA-III for high-dimensional problems.

Keywords: cloud computing, resource allocation, cybersecurity, game theory, cooperative games, coalition games, evolutionary optimization, NSGA-II, dynamic risk, multi-criteria optimization.

REFERENCES

1. Sreeramulu, M. D. (2024). Optimizing Cloud Computing Networks in Information Security Controls using COPRAS Method. *Comput. Sci. Eng. Technol., 1(2),* 42–54. https://doi.org/10.46632/cset/1/2/6
2. Tsyrkaniuk, D. (2025). A Model for the Distribution of Computational Tasks in Cloud Infrastructure Incorporating Performance, Cost, and Security Considerations. *Cybersecur. Edu. Sci. Tech., 4(28),* 619–632. https://doi.org/10.28925/2663-4023.2025.28.836
3. Tsyrkaniuk, D. (2025). Advanced Hybrid Model with Risk-based and Cooperative Cloud Security Strategies. *Cybersecur. Edu. Sci. Tech., 1(29),* 909–929. https://doi.org/10.28925/2663-4023.2025.29.970
4. Petrovska, I., & Kuchuk, H. (2023). Adaptive Resource Allocation Method for Data Processing and Security in Cloud Environment. *Adv. Inf. Syst., 7(3),* 67–73. https://doi.org/10.20998/2522-9052.2023.3.10
5. Chen, Y.-F., Lin, F. Y.-S., Tai, K.-Y., Hsiao, C.-H., Wang, W.-H., Tsai, M.-C., & Sun, T.-L. (2025). A Near-Optimal Resource Allocation Strategy for Minimizing the Worse-Case Impact of Malicious Attacks on Cloud Networks. *J. Cloud Comput., 14(1).* https://doi.org/10.1186/s13677-025-00749-6
6. Chen, Y., Feng, E., & Ling, Z. (2024). Secure Resource Allocation Optimization in Cloud Computing Using Deep Reinforcement Learning. *J. Adv. Comput. Syst., 4(11),* 15–29. https://doi.org/10.69987/jacs.2024.41102
7. Jiao, Y. (2025). Research on Architecture Optimization and Security Design of university Cloud Service Platform. *J. Comput. Method. Sci. Eng.* https://doi.org/10.1177/14727978251352132

8. Bal, P. K., Mohapatra, S. K., Das, T. K., Srinivasan, K., & Hu, Y.-C. (2022). A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing Using Hybrid Machine Learning Techniques. *Sensors, 22(3)*, 1242. https://doi.org/10.3390/s22031242
9. Selvan, T., Siva Shankar, S., Sri Nandhini Kowsalya, S., Ravuri, P., Kumar Nayak, D., Gurnadha Gupta, K., & Sharath, M. N. (2024). Modernizing Cloud Computing Systems with Integrating Machine Learning for Multi-Objective Optimization in Terms of Planning and Security. *MATEC Web of Conf.*, *392*, 01155. https://doi.org/10.1051/matecconf/202439201155
10. Tsyrkaniuk, D., & Sokolov, V. (2024). Methodology for Investigating Information Security Incidents. *Cybersecur. Edu. Sci. Tech., 2(26)*, 140–154. https://doi.org/10.28925/2663-4023.2024.26.675
11. Kristiani, E., Yang, C.-T., Huang, C.-Y., Wang, Y.-T., & Ko, P.-C. (2020). The Implementation of a Cloud-Edge Computing Architecture using OpenStack and Kubernetes for Air Quality Monitoring Application. *Mob. Netw. Appl., 26(3)*, 1070–1092. https://doi.org/10.1007/s11036-020-01620-5
12. Kristiani, E., Yang, C.-T., Wang, Y.-T., & Huang, C.-Y. (2018). Implementation of an Edge Computing Architecture using Openstack and Kubernetes. In *Int. Conf. on Information Science and Applications* (pp. 675–685). Springer. https://doi.org/10.1007/978-981-13-1056-0_66