КИЇВСЬКИЙ СТОЛИЧНИЙ УНІВЕРСИТЕТ ІМЕНІ БОРИСА ГРІНЧЕНКА

Кваліфікаційна наукова
праця на правах рукопису

ЦИРКАНЮК ДІАНА АНДРІЇВНА

УДК 004.056:004.75

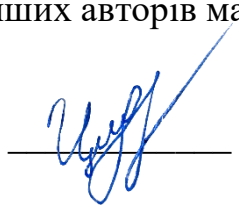
ДИСЕРТАЦІЯ

МЕТОДИ ТА МОДЕЛІ ОПТИМІЗАЦІЇ РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ ХМАРНИХ СИСТЕМ ДЛЯ ПІДВИЩЕННЯ БЕЗПЕКИ

Спеціальність 125 Кібербезпека
Галузь знань 12 Інформаційні технології

Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.



_____ Д. А. Цирканюк

Науковий керівник:
Соколов Володимир Юрійович
кандидат технічних наук, доцент

Київ – 2026

АНОТАЦІЯ

Цирканюк Д. А. Методи та моделі оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 125 Кібербезпека. – Київський столичний університет імені Бориса Грінченка, Київ, 2026.

Дисертаційна робота присвячена вирішенню актуального наукового завдання, сутність якого полягає в підвищенні ефективності оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення кібербезпеки завдяки гібридизації багатокритеріальних еволюційних алгоритмів (NSGA-II і NSGA-III) та ігрових моделей безпеки, з урахуванням варіативного ризикового профілю, продуктивності, економічної ефективності та коаліційної вигоди в хмарних середовищах, а також впровадженню кооперативно-еволюційного методу CoopEvo-CloudSec на підприємствах та в організаціях.

Методологія оптимізації ресурсів хмарних систем є потужним інструментом, який має значний вплив на безпеку держави та роботу комерційних організацій через автоматизацію процесів розподілу задач, оцінки ризиків та кооперативного захисту в реальному часі, чому сприяють декілька факторів, які змушують звернути увагу на методології, на актуальність їх удосконалення, а саме:

1. Зміна ландшафту кіберзагроз. Із появою розподілених атак (DDoS, EDoS) та збільшенням обчислювальних можливостей традиційні моделі розподілу ресурсів, які покладаються на статичні дані, перестають адекватно виявляти та реагувати на компрометацію вузлів. Тому актуальними стають задачі по виявленню, реєстрації та реагуванню на нові виклики, а також швидкий розвиток даної галузі.

2. Перехід хмарних систем від локальних до гібридних середовищ. При використанні традиційних інфраструктур (IaaS, PaaS, SaaS) їх контроль обмежувався провайдером, тому масштабованість та ізоляція були меншими та

піддавалися ручному керуванню. Із переходом до гібридних хмарних середовищ вартість ресурсів зменшилася, а розповсюдження віртуалізації створило уяву безпечності середовища, то користувачі стали розгортати більш складні та довготривалі задачі, що стало особливо актуальним в епоху віддаленої роботи. Також через збільшення об'єму обчислювальних задач необхідно швидше зі сторони держави опрацьовувати їх для вчасного виявлення, до прикладу, терористичних загроз, а зі сторони приватних підприємств – для виявлення витоку конфіденційних даних.

3. Порушення даних і зовнішні загрози. Компрометація гіпервізорів та введення спотворень в розподіл ресурсів створюють загрози для перенасичення хмарної системи запитами. Виявлення та протидія атакам при аналізі ризиків, в тому числі, генерації фейкових навантажень, призводять до перенавантаження вузлів та обмеженню ресурсів реагування, що створює загрозу недоотримання уваги легітимними задачами.

4. Розширення ролі хмарних служб. Оскільки підприємства та організації все частіше використовують хмарні послуги для зберігання конфіденційних даних, то виникає потреба в додатковій обробці, в тому числі, ізоляції та видалення вразливих компонентів з ресурсів.

5. Вимоги відповідності. До ресурсів хмарних середовищ висуваються вимоги щодо їхньої конфіденційності в межах державних стандартів (GDPR, HIPAA), комерційних (PCI DSS) та/або етичних обмежень. В свою чергу, хмарні дані є важким видом інформації для структурованого пошуку та аналізу стосовно висунутих вимог та обмежень.

6. Безперервний моніторинг і адаптивна безпека. Обробка ресурсів може проводитися як архівних, так і в режимі реального часу, але вузьким місцем хмарних середовищ є потокова оптимізація. Тому реагування на інциденти може проводитися у два способи: невідкладні дії та розслідування інцидентів, але обидва підходи мають свій набір невирішених завдань.

7. Реагування на інциденти та виявлення загроз. Системи оптимізації ресурсів не мають в своєму складі механізмів щодо реагування на інциденти, тому повинні

сигналізувати іншим системам в режимі реального часу. Інтеграція із зовнішніми хмарними середовищами для забезпеченні безпеки має обмеження на швидкодію та затримки на час обробки запитів, але все одно зменшує потенційну шкоду. Також слід зазначити, що актуальність реагування різко зменшується з плином часу.

Таким чином, дослідження щодо вдосконалення оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки є актуальним через його узгодження з поточним ландшафтом кібербезпеки, вирішення проблем, пов'язаних із гібридними середовищами, забезпеченням конфіденційності даних користувачів та еволюцією природи кіберзагроз. Воно забезпечує адаптивний підхід до безпеки, необхідний для оцінки ризиків та загроз ресурсів, які циркулюють в хмарних середовищах, віртуальних мережах та розподілених системах.

Для досягнення мети в підвищенні ефективності та безпеки функціонування хмарних систем шляхом розробки методу кооперативно-еволюційного розподілу обчислювальних ресурсів було вирішено наступні задачі:

1. Вперше запропонований та математично обґрунтований метод кооперативно-еволюційного розподілу обчислювальних ресурсів (CoopEvo-CloudSec) у хмарних системах з урахуванням ризиків для їхньої безпеки, та якій вирізняється від наявних, інтеграцією критерію коаліційної взаємодії захисників безпосередньо у еволюційний процес пошуку рішень на базі алгоритмів багатокритеріальної оптимізації NSGA-II/NSGA-III, залежно від часових обмежень, що дозволяє сформувати узгоджені стратегії захисту та відповідну проводити реконфігурацію ресурсів ХМС.

2. Вперше запропонована багатокритеріальна оптимізаційна модель розподілу ресурсів ХМС, яка, на відміну від відомих, враховує чотири суперечливі критерії – ризик, продуктивність, вартість та, вперше, коаліційну вигоду від взаємодії захисників, що дозволило математично формалізувати задачу пошуку множини Парето-оптимальних рішень в умовах мультиагентної протидії кіберзагрозам;

3. Вдосконалена теоретико-ігрова модель конфліктної взаємодії «атакуючий–захисник», яка, на відміну від чинних, містить параметр агресивності $\lambda(t)$, який

описує варіативність інтенсивності атак у часі, що дозволило врахувати стохастичну або еволюційну природу ризику та застосувати механізм розрахунку виграшу коаліції на основі вектора Шеплі для формування узгоджених стратегій захисту ХМС.

У вступі обґрунтовується важливість й актуальність теми дисертаційного дослідження, сформульовано мету та задачі роботи, визначено основні положення, наукову та практичну цінність отриманих результатів роботи та наведено особистий внесок автора.

У першому розділі здійснено аналіз методів і моделей оптимізації ресурсів хмарних систем для підвищення кібербезпеки. Показана специфіка хмарних систем як об’єкта дослідження, проведено огляди архітектурних моделей (IaaS, PaaS, SaaS тощо) та методів розподілу ресурсів, також порівняльний аналіз підходів до управління ресурсами з урахуванням безпеки. Визначено роль безпеки в оптимізації ресурсів хмарних систем, проаналізований поточний стан досліджень у сфері кібербезпеки хмарних середовищ, визначено основні аспекти, підходи та принципи до багатокритеріальної оптимізації з інтеграцією ігрових моделей. Сформульовано актуальне наукове завдання, яке полягає в подальшому розвитку методів оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки та продуктивності, зокрема з урахуванням багатокритеріальних критеріїв та кооперативних стратегій захисту. Тому для його вирішення визначено мету роботи, яка полягає в підвищенні ефективності оптимізації ресурсів хмарних систем завдяки гібридизації багатокритеріальних еволюційних алгоритмів та ігрових моделей безпеки в процесі управління ресурсами та оцінки ризиків.

У другому розділі визначено основні підходи до моделювання розподілу обчислювальних ресурсів у хмарних системах з урахуванням ризику. Вперше запропоновано гібридну модель кооперативно-еволюційного розподілу ресурсів, що поєднує теорію ігор, багатокритеріальну оптимізацію NSGA-II та адаптивну оцінку ризику, що дало змогу запропонувати способи балансування безпеки, продуктивності та вартості в хмарному середовищі. Також зазначені обмеження та

теоретична складність моделі при масштабуванні коаліційних взаємодій у системах кібербезпеки. З урахуванням отриманих в поточному розділі результатів щодо ігрової оцінки ризику та кооперативних стратегій захисту в наступному розділі приділено увагу розширенню даних теоретичних обґрунтувань у вигляді алгоритмічної реалізації та практичної перевірки результатів.

У третьому розділі визначено методику проведення експериментального дослідження з використанням синтетичних і реальних даних для симуляції хмарних систем. Запропоновано вдосконалений підхід до моделювання варіативного ризикового профілю $\lambda(t)$ та кооперативних стратегій захисту за допомогою алгоритмів NSGA-II і NSGA-III. Сформульовано проблеми та вибір підходів до оцінки ефективності методу CoopEvo-CloudSec, верифіковано метрики якості (Hypervolume, IGD, Spacing) та набори даних, а також проведене їхнє експериментальне дослідження. Вдосконалено метод кооперативно-еволюційного розподілу ресурсів шляхом інтеграції коаліційної вигоди та динамічної оцінки ризиків за допомогою підбору експериментальних сценаріїв, побудови алгоритму симуляції та експериментальної установки, а також проведення експерименту та верифікації його результатів із балансування безпеки, продуктивності й вартості. Вперше запропоновано матрицю рекомендованих конфігурацій параметрів методу для різних класів задач хмарних систем. Для перевірки функціонування методу розроблено архітектуру симуляційного середовища, підібрано набори даних, визначено сценарії з високою варіативністю загроз, проведено тренінг та верифікацію результатів експерименту із порівнянням версій методу (V1–V4), а також проведено оцінку ефективності результатів для різних платформ і секторів економіки за запропонованим методом.

Дисертація виконувалась в Київському столичному університеті імені Бориса Грінченка.

Результати наукових досліджень були використані на кафедрі інформаційної та кібернетичної безпеки імені професора Володимира Бурячка факультету інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка в рамках науково-дослідної роботи: «Методи та моделі

забезпечення кібербезпеки інформаційних систем переробки інформації та функціональної безпеки програмно-технічних комплексів управління критичної інфраструктури» (№ 0122U200483, КСУБГ, м. Київ).

Також результати наукових досліджень прийняті до впровадження в діяльність Київського столичного університету імені Бориса Грінченка (акт від 09.12.2025 року) та Інституті програмних систем Національної академії наук України (акт від 09.12.2025 року).

Ключові слова: хмарна система, кібербезпека, інформаційна безпека, оптимізація ресурсів, багатокритеріальна оптимізація, теорія ігор, оцінка ризику, модель управління, ігрова модель, захисник, атакуючий, симуляційне моделювання, продуктивність, економічна ефективність.

ANNOTATION

Tsyrkaniuk D. A. Methods and Models for Optimizing the Distribution of Computing Resources of Cloud Systems to Improve Security. – Qualification of scientific work on the rights of a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 125 Cybersecurity. – Borys Grinchenko Kyiv Metropolitan University, Kyiv, 2026.

The dissertation is devoted to solving a relevant scientific problem, the essence of which is to increase the efficiency of optimizing the allocation of computing resources of cloud systems to improve cybersecurity through the hybridization of multi-criteria evolutionary algorithms (NSGA-II and NSGA-III) and game security models, taking into account the variable risk profile, productivity, economic efficiency and coalitional benefit in cloud environments, as well as the implementation of the cooperative-evolutionary method CoopEvo-CloudSec at enterprises and organizations.

The methodology for optimizing cloud system resources is a powerful tool that has a significant impact on state security and the work of commercial organizations through the automation of task allocation processes, risk assessment, and cooperative protection in real time, which is facilitated by several factors that force us to pay attention to methodologies and the relevance of their improvement, namely:

1. Changing the landscape of cyber threats. With the advent of distributed attacks (DDoS, EDoS) and the increase in computing capabilities, traditional resource allocation models that rely on static data become inadequate for detecting and responding to node compromise. Therefore, the tasks of detecting, registering, and responding to new challenges, as well as the rapid development of this industry, have become relevant.

2. The transition of cloud systems from local to hybrid environments. When using traditional infrastructures (IaaS, PaaS, SaaS), control was limited to the provider, resulting in less scalability and isolation that were subject to manual management. With the transition to hybrid cloud environments, the cost of resources decreased, and the spread of virtualization created the illusion of a secure environment. As a result, users

began to deploy more complex and long-term tasks, which became especially relevant in the era of remote work. Additionally, due to the increase in the volume of computing tasks, the state must process them more efficiently for timely detection, such as terrorist threats, and for private enterprises to identify leaks of confidential data.

3. Data breaches and external threats. Compromises of hypervisors and the introduction of distortions in resource distribution create threats to oversaturate the cloud system with requests. Detection and counteraction of attacks during risk analysis, including the generation of fake loads, can lead to node overload and a limitation of response resources, creating a threat to not receiving attention from legitimate tasks.

4. Expanding the role of cloud services. As enterprises and organizations increasingly utilize cloud services to store confidential data, there is a growing need for additional processing, including the isolation and removal of vulnerable components from resources.

5. Compliance requirements. Cloud resources are subject to confidentiality requirements within government standards (e.g., GDPR, HIPAA), commercial standards (e.g., PCI DSS), and/or ethical constraints. In turn, cloud data is a complex type of information that requires structured search and analysis in relation to specific requirements and limitations.

6. Continuous monitoring and adaptive security. Resource processing can be carried out in both archival and real-time modes, but the bottleneck in cloud environments is streaming optimization. Therefore, incident response can be carried out in two ways: immediate actions and incident investigation, but both approaches have their own set of unresolved tasks.

7. Incident response and threat detection. Resource optimization systems lack built-in mechanisms for responding to incidents, so they must signal other systems in real-time. Integration with external cloud environments for security has limitations in terms of speed and latency, but it still reduces the potential for harm. It is also worth noting that the urgency of response decreases significantly over time.

Thus, research on improving the optimization of cloud computing resource allocation for security is relevant because it aligns with the current cybersecurity

landscape, addressing issues related to hybrid environments, ensuring user data privacy, and the evolving nature of cyber threats. It provides an adaptive approach to security that is necessary to assess the risks and threats to resources circulating in cloud environments, virtual networks, and distributed systems.

To achieve the goal of increasing the efficiency and security of cloud systems by developing a method of cooperative-evolutionary allocation of computing resources, the following tasks were solved:

1. For the first time, a method of cooperative-evolutionary allocation of computing resources (CoopEvo-CloudSec) in cloud systems, taking into account the risks to their security, has been proposed and mathematically substantiated, which differs from existing ones by integrating the criterion of coalitional interaction of defenders directly into the evolutionary process of finding solutions based on multi-criteria optimization algorithms NSGA-II/NSGA-III, depending on time constraints, which allows for the formation of coordinated protection strategies and the appropriate reconfiguration of HMS resources.

2. For the first time, a multi-criteria optimization model of HMS resource allocation has been proposed, which, unlike the known ones, takes into account four contradictory criteria – risk, productivity, cost and, for the first time, coalitional benefit from the interaction of defenders, which allowed mathematically formalizing the problem of finding a set of Pareto-optimal solutions in the conditions of multi-agent counteraction to cyber threats;

3. An improved game-theoretic model of the “attacker–defender” conflict interaction, which, unlike the current ones, contains the aggressiveness parameter $\lambda(t)$, which describes the variability of the intensity of attacks over time, which made it possible to take into account the stochastic or evolutionary nature of the risk and apply the mechanism for calculating the coalition gain based on the Shapley vector to form coordinated strategies for protecting the HMS.

The introduction justifies the importance and relevance of the topic of the dissertation research, formulates the goal and objectives of the work, identifies the main provisions, scientific and practical value of the results obtained, and gives the author’s personal contribution.

The first section analyzes methods and models for optimizing cloud system resources to improve cybersecurity. The specifics of cloud systems as an object of research are presented, including reviews of architectural models (IaaS, PaaS, SaaS, etc.) and resource allocation methods, as well as a comparative analysis of approaches to resource management with security in mind. The role of security in optimizing cloud system resources is established, the current state of research in the field of cybersecurity for cloud environments is analyzed, and the main aspects, approaches, and principles of multi-criteria optimization with the integration of game models are identified. A relevant scientific task is formulated, which involves further developing methods for optimizing the allocation of computing resources in cloud systems to enhance security and productivity, particularly by considering multi-criteria criteria and cooperative protection strategies. Therefore, the goal of this work is defined as increasing the efficiency of optimizing cloud system resources through the hybridization of multi-criteria evolutionary algorithms and game security models in the process of resource management and risk assessment.

The second section defines the main approaches to modeling the allocation of computing resources in cloud systems, taking into account risk. For the first time, a hybrid model of cooperative-evolutionary resource allocation is proposed, combining game theory, multi-criteria optimization, NSGA-II, and adaptive risk assessment, which enables the proposal of ways to balance security, performance, and cost in a cloud environment. The limitations and theoretical complexity of the model, particularly when scaling coalition interactions in cybersecurity systems, are also highlighted. Building on the results obtained in the current section on game risk assessment and cooperative protection strategies, the following section focuses on expanding the theoretical justifications through algorithmic implementation and practical verification of the results.

The third section outlines the methodology for conducting an experimental study, utilizing both synthetic and real data to simulate cloud systems. An improved approach to modeling the variable risk profile $\lambda(t)$ and cooperative protection strategies is proposed, utilizing the NSGA-II and NSGA-III algorithms. The problems and approaches to assessing the effectiveness of the CoopEvo-CloudSec method were formulated, quality

metrics (Hypervolume, IGD, Spacing) and datasets were verified, and an experimental study was conducted. The method of cooperative-evolutionary resource allocation was enhanced by integrating coalitional benefit and dynamic risk assessment through the selection of experimental scenarios, the development of a simulation algorithm and experimental setup, and the conduct of an experiment to verify its results in balancing security, performance, and cost. For the first time, a matrix of recommended configurations for the method parameters across different classes of cloud system tasks was proposed. To verify the functioning of the method, the architecture of the simulation environment was developed, datasets were selected, scenarios with high threat variability were identified, training and verification of the experimental results were conducted with a comparison of the method versions (V1–V4), and an assessment of the effectiveness of the results for different platforms and sectors of the economy using the proposed method was carried out.

The dissertation was conducted at Borys Grinchenko Kyiv Metropolitan University.

The results of scientific research were used at the Department of Information and Cybersecurity named after Professor Volodymyr Buriachok of the Faculty of Information Technologies and Mathematics of Borys Metropolitan Grinchenko Kyiv University within the framework of research work: “Methods and Models for Ensuring Cybersecurity of Information Systems, Information Processing and Functional Security of Software and Hardware Complexes for Critical Infrastructure Management” (No. 0122U200483, BGKMU, Kyiv).

Also, the results of scientific research have been accepted for implementation in the activities of Borys Grinchenko Kyiv Metropolitan University (12/09/2025) and Institute of Software Systems of the National Academy of Sciences of Ukraine (12/09/2025).

Keywords: cloud system, cybersecurity, information security, resource optimization, multi-criteria optimization, game theory, risk assessment, management model, game model, defender, attacker, simulation modeling, productivity, cost-effectiveness.

Наукові статті, опубліковані у наукових виданнях, включених на дату опублікування до переліку наукових фахових видань України:

1. **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

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. **Цирканюк, Д.** (2025). Метод багатокритеріальної оптимізації безпеки хмарних обчислень на основі модифікованого алгоритму NSGA-II. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(30)*, 692–714. https://doi.org/10.28925/2663-4023.2025.30.983

5. **Цирканюк, Д.** (2025). Метод CoopEvo-CloudSe для оптимізації обчислювальних ресурсів хмарних систем для підвищення безпеки. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(31)*, 872–887. https://doi.org/10.28925/2663-4023.2025.31.1077

Наукові публікації, у яких додатково висвітлено результати дисертації:

1. Shevchenko, S., Zhdanova, Y., Dreis, Y., Kyrychok, R., & **Tsyrkaniuk, D.** (2023). Protection of Information in Telecommunication Medical Systems based on a Risk-Oriented Approach. In *Cybersecurity Providing in Information and Telecommunication Systems* (Vol. 3421, pp. 158–167). (Scopus).

2. Гулак Г. М., Скітер І. С., Гулак Є. Г., & **Цирканюк Д. А.** (2023). Базові засади побудови центру кібербезпеки об’єктів ядерної енергетики. In *14-а Всеукраїнська науково-практична конференція «Актуальні проблеми управління інформаційною безпекою держави»* (pp. 262–266).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВМ – віртуальна машина ‘virtual machine’

ІБ – інформаційна безпека ‘information security’

ІТ – інформаційні технології

ХМС – хмарні системи ‘cloud systems’

ХО – хмарні обчислення ‘cloud computing’

APT – Advanced Persistent Threat ‘цілеспрямовані вторгнення’

CaaS – Container as a Service ‘контейнер як послуга’

CPU/GPU – Central/Graphics Processing Unit ‘центральний/графічний процесор’

CSPM – Cloud Security Posture Management ‘управління безпекою хмарних технологій’

DDoS – Distributed Denial of Service ‘розподілена відмова в обслуговуванні’

DLP – Data Loss Prevention ‘запобігання втраті даних’

EaaS – Edge as a Service / Edge-oriented as a Service ‘периферія як послуга’

EDoS – Economic Denial of Sustainability ‘економічне заперечення сталого розвитку’

EDR – Endpoint Detection and Response ‘виявлення та реагування на кінцеві точки’

FaaS – Function as a Service ‘функція як послуга’

HV – HyperVolume ‘гіпероб’єм’ – метрика якості фронту Парето

IaaS – Infrastructure as a Service ‘інфраструктура як послуга’

IAM – Identity and Access Management ‘управління ідентифікацією та доступом’

IGD – Inverted Generational Distance ‘зворотна генераційна відстань’

MOEA/D – Multi-Objective Evolutionary Algorithm based on Decomposition ‘багатоцільовий еволюційний алгоритм, заснований на декомпозиції’

NSGA-II/III – Non-dominated Sorting Genetic Algorithm II/III ‘недомінований генетичний алгоритм сортування II/III’

PaaS – Platform as a Service ‘платформа як послуга’

QoS – Quality of Service ‘якість обслуговування’

RBAC – Role-based Access Control ‘контроль доступу на основі ролей’

SaaS – Software as a Service ‘програмне забезпечення як послуга’

SDN – Software-Defined Networking ‘програмно-визначені мережі’

SIEM – Security Information and Event Management ‘інформація про безпеку та управління подіями’

SLA – Service Level Agreement ‘угода про рівень обслуговування’

Spacing – метрика рівномірності розподілу (spread)

XaaS – Everything as a Service / Anything as a Service ‘будь-що як послуга’

ВСТУП

Обґрунтування вибору теми дослідження. Протягом останніх десятиліть відбулися значні зміни обчислювальних парадигм. Хмарні обчислення (ХО) віддзеркалили найбільш помітні з цих змін. Завдяки розгортанню хмарних технологій, хмарні центри обробки даних сьогодні керують аналізом, зберіганням даних та прийняттям рішень в більшості бізнес-процесів. ХО, які надали користувачам масштабовані ресурси за моделями IaaS, PaaS та SaaS, заклали фундамент розгортання важливих для бізнесу хмарних сервісів, проте їхня багатокористувацька природа та розподілена архітектура створили нові вектори кіберзагроз. В умовах зростання інтенсивності атак, зокрема розподілена відмова в обслуговуванні ‘Distributed Denial of Service’ (DDoS), економічне заперечення сталого розвитку ‘Economic Denial of Sustainability’ (EDoS) та цілеспрямованих вторгнень (APT), наявні методи, моделі та інформаційні технології (IT) в завданнях управління ресурсами хмарних систем (ХМС), виявилися недостатньо ефективними. Що більше, світова статистика інцидентів засвідчила, що фінансові та репутаційні втрати від простою ХМС зросли експоненційно, перетворив завдання управління ресурсами з суто технічної на вирішальну умову виживання бізнесу. Зокрема, за даними звітів 2023–2024 рр., понад 60–80 % організацій зафіксували інциденти інформаційної безпеки (ІБ), пов’язані з ХО, серед причин яких, домінували неправильні конфігурації та помилки в управлінні доступом (IAM). Звідсіля виникла об’єктивна суперечність між необхідністю забезпечення високої якості обслуговування (QoS) та вимогою гарантування надійності й конфіденційності даних в ХМС. Це, відповідно, потребує синтезу нових, гнучких та ефективних методів розподілу обчислювальних потужностей із вбудованими механізмами безпеки ХМС. Дослідження в напрямі підвищення ефективності та захищеності ХМС активно ведуться вже декілька десятиріч.

Вагомий внесок у розбудову теоретичних та практичних засад ХО, а також методів розподілу ресурсів ХМС зробили численні закордонні вчені. Зокрема, фундаментальні питання класифікації моделей розподілу ресурсів ХМС, забезпечення енергоефективності та масштабованості висвітлено у роботах Saidi

K., Hioual O., Siam A., Senthilkumar G., Tamilarasi K., Velmurugan N. та Periasamy J. K. Проблематику планування задач та автономного управління навантаженням досліджували Rabaaoui S., Hachicha H., Zagrouba E., Jabber S. A., Hashem S. H. та Jafer S. H., закцентував переважно увагу на економічних метриках та QoS. Систематизації стратегій алокації ресурсів та аналізу програмно-конфігурованих мереж присвячені роботи Kareem Awad W., Mohamed A., Vergara J., Botero J. та Fletscher L. Окремий пласт досліджень, спрямований на використання методів штучного інтелекту в завдання прогнозування навантаження та управління ХМС, представлено у доробку Lekkala C., Ashawa M. та Manzoor M. F. Завдання багатозмінного управління та оптимізації SLA розглядали Gong S., Madni S. H. H., Li C. та Li L. Y., які сформували базис для розуміння економічної ефективності ХМС. Специфіку забезпечення безпеки у ХМС та інтеграцію захисних механізмів у процеси управління ресурсами досліджували такі науковці, як Parast F. K., Sindhav C., Saxena D., Singh A. K., Han J., Zang W. та Liu L. Значний вплив на розвиток теоретико-ігрових підходів до кібербезпеки та моделювання взаємодії «атакуючий–порушник» мають роботи Wilczyński A., Jebalia M., Xu X., Yu H., Ait Temghart A., а також концептуальні розробки архітектури самозахисту (SECURE) авторства Gill S. S. та Buyya R. Різні аспекти контейнерної безпеки та захисту даних аналізували Jarkas O., Feng D., Aldhyani T. H. та Alkahtani H. Серед вітчизняних науковців, які зробили суттєвий внесок у розв’язання задач оптимізації ресурсів та захисту інформації в комп’ютерних системах, зокрема ХМС, відзначимо Дорогого Я. Ю., який розглядав розподіл ресурсів критичної інфраструктури, та Волка М. О., Курочкіна В. С. і Запорожченка А. П., які запропонували гібридні методи управління ХМС. Питаннями первинного виділення ресурсів та застосуванням методу аналізу ієрархій займалися Гребенюк Д. С. та Давидов В. В. Також серед українських науковців вагоме місце займають праці Петровської І. Ю., Кучука Г. А., Панченка В. І. та Філоненко А. М., присвячені методам розподілу ресурсів при наданні хмарних послуг та забезпеченню збалансованості навантаження ХМС. Проблематика кібербезпеки хмарних обчислень, аналіз загроз та розробка засобів захисту знайшли відображення у роботах Ковалюк О. А., Лучика С. Д., Шевченка

В. В., Бобренка В. В. та Гуди А. І. Критерії вибору та оцінювання якості хмарних сервісів досліджували Закаляк Р. Ф., Андрощук О., Голобородько М., Кондратенко Ю. та Литовченко Г. Окремі аспекти захисту інформації, застосування ігрових моделей та стохастичного моделювання ризиків розглядалися у працях Михайліва В. І., Кобевка А. Т., Тимченка О. В., Шабали Є., Корнійчука Б., Стефурака О. Р., Тихонова Ю. О., Лаптєва О. А., Зозулі С. А., Павлова І. М. та Толюпи С. В.

Попри значний обсяг наукових напрацювань, більшість релевантних підходів розглядають питання продуктивності ХМС та безпеки ізольовано або застосовують статичні моделі ризиків, що не дозволяє ефективно протидіяти варіативним загрозам ХМС. Відсутність комплексної моделі, яка б поєднувала багатокритеріальну оптимізацію, гнучку оцінку ризиків та апарат кооперативної взаємодії захисників, обумовила потребу проведення даного дослідження та підтвердило його актуальність для подальшого розвитку технологій захисту ХМС.

Зв'язок роботи з науковими програмами, планами, темами. Напрям дисертаційного дослідження безпосередньо пов'язаний з реалізацією доктрини інформаційної безпеки України, Стратегії інформаційної безпеки та Стратегії кібербезпеки України. Дисертаційна робота виконана відповідно до планів наукової і науково-технічної діяльності Київського столичного університету імені Бориса Грінченка в рамках науково-дослідної роботи: «Методи та моделі забезпечення кібербезпеки інформаційних систем переробки інформації та функціональної безпеки програмно-технічних комплексів управління критичної інфраструктури» (№0122U200483, КСУБГ, м. Київ).

Мета і завдання дослідження. *Мета* дисертаційного дослідження полягає в підвищенні ефективності та безпеки функціонування хмарних систем шляхом розробки методу кооперативно-еволюційного розподілу обчислювальних ресурсів.

У відповідності до поставленої мети для вирішення наукового завдання в роботі визначено та розв'язано такі *часткові* ***завдання***:

– проаналізувати попередні дослідження з проблематики дисертації, методи та моделі управління ресурсами в хмарних системах, виявити їхні обмеження щодо

врахування варіативних ризиків безпеки та обґрунтувати необхідність розробки нового методу розподілу ресурсів ХМС;

– розробити багатокритеріальну оптимізаційну модель розподілу ресурсів ХМС яка дозволить знаходити множину Парето-оптимальних рішень;

– удосконалити теоретико-ігрову модель взаємодії «порушник-захисник» в ХМС;

– розробити метод кооперативно-еволюційного розподілу обчислювальних ресурсів, який поєднає інструментарій еволюційної оптимізації (NSGA-II / NSGA-III) з апаратом кооперативних ігор для формування узгоджених стратегій захисту та отримання синергетичного ефекту у вигляді коаліційної вигоди від взаємодії захисників;

– експериментально дослідити ефективність запропонованого методу шляхом імітаційного моделювання, оцінити якість отриманих рішень за метриками Hypervolume (HV), зворотна генераційна відстань ‘Inverted Generational Distance’ (IGD), Spacing та розробити рекомендації щодо його практичного впровадження в платформи оркестрації ХМС.

Об’єктом дослідження є процеси розподілу обчислювальних ресурсів у хмарних системах.

Предметом дослідження є методи та моделі оптимізації розподілу ресурсів хмарних систем з урахуванням критеріїв безпеки, продуктивності, вартості та коаліційної взаємодії захисників.

Методи дослідження. Теоретико-методологічну основу дисертаційної роботи склав системний підхід, який поєднав методи системного аналізу, математичного моделювання та штучного інтелекту. Для розв’язання поставлених завдань використано: методи системного аналізу та узагальнення – для дослідження специфіки архітектурних моделей хмарних систем (IaaS, PaaS, SaaS тощо), систематизації існуючих методів та моделей в завданні управління ресурсами ХМС та виявлення їхніх обмежень щодо врахування релевантних кіберзагроз; апарат теорії ігор, зокрема, антагоністичні та кооперативні ігри – для формалізації конфліктної взаємодії між підсистемою захисту та порушником

політики інформаційної безпеки, а також для моделювання коаліційної співпраці захисників із використанням вектору Шеплі для розподілу вигоди; методи теорії ймовірностей та стохастичного моделювання – для математичного опису параметра агресивності атак $\lambda(t)$ та розрахунку функції ризику для вузлів ХМС; методи багатокритеріальної оптимізації та еволюційні алгоритми – для синтезу методу CoopEvo-CloudSec, зокрема, алгоритми NSGA-II та NSGA-III, застосовано для пошуку множини Парето-оптимальних рішень у просторі чотирьох критеріїв – безпека, продуктивність, вартість, коаліційна вигода; методи імітаційного моделювання та методи математичної статистики – для проведення обчислювальних експериментів, оцінювання якості отриманих рішень за метриками HV, IGD, Spacing та підтвердження статистичної значущості переваг розробленого методу CoopEvo-CloudSec порівняно з чинними аналогами.

Наукова новизна одержаних результатів полягає в подальшому розвитку теоретичних і практичних методів та методів оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки:

1. Вперше запропонований та математично обґрунтований метод кооперативно-еволюційного розподілу обчислювальних ресурсів (CoopEvo-CloudSec) у хмарних системах з урахуванням ризиків для їхньої безпеки, та якій вирізняється від наявних, інтеграцією критерію коаліційної взаємодії захисників безпосередньо у еволюційний процес пошуку рішень на базі алгоритмів багатокритеріальної оптимізації NSGA-II/NSGA-III, залежно від часових обмежень, що дозволяє сформувати узгоджені стратегії захисту та відповідну проводити реконфігурацію ресурсів ХМС.

2. Вперше запропонована багатокритеріальна оптимізаційна модель розподілу ресурсів ХМС, яка, на відміну від відомих, враховує чотири суперечливі критерії – ризик, продуктивність, вартість та, вперше, коаліційну вигоду від взаємодії захисників, що дозволило математично формалізувати задачу пошуку множини Парето-оптимальних рішень в умовах мультиагентної протидії кіберзагрозам.

3. Вдосконалена теоретико-ігрова модель конфліктної взаємодії «атакуючий-захисник», яка, на відміну від чинних, містить параметр агресивності $\lambda(t)$, який

описує варіативність інтенсивності атак у часі, що дозволило врахувати стохастичну або еволюційну природу ризику та застосувати механізм розрахунку виграшу коаліції на основі вектора Шеплі для формування узгоджених стратегій захисту ХМС.

Практичне значення одержаних результатів полягає у розробці та програмній реалізації на мові програмування Python (IDE PyCharm) з використанням бібліотек Pymoo методу CoopEvo-CloudSec та відповідного програмного комплексу, який дозволяє підвищити стійкість хмарних систем (ХМС) до релевантних кіберзагроз без втрати продуктивності системи. Розроблений програмний модуль CoopEvo-CloudSec Engine на відміну від стандартних планувальників, дозволяє знаходити компромісні рішення між безпекою, вартістю та часом виконання завдань в ХМС. Експериментально підтверджено, що використання модифікованого алгоритму на базі алгоритму NSGA-II) дозволяє покращити якість фронту Парето за показником HV на 25,2% порівняно з базовим алгоритмом NSGA-II (значення 1,11 проти 0,89). Це забезпечує адміністраторам ХМС ширший та якісніший вибір стратегій захисту. Доведено експериментально зростання різноманітності рішень (метрика Diversity) до рівня 1,23 (проти 0,268 у базових методах), що дає можливість обрати оптимальні конфігурації підсистеми захисту ХМС. Розроблено архітектурну схему інтеграції методу CoopEvo-CloudSec, включаючи модулі Risk Analyzer та Policy Adapter, яка сумісна з платформами Kubernetes та OpenStack, що дозволило під час впровадження (акти впровадження наведено в Додатках В і Г) використати розроблений метод CoopEvo-CloudSec як алгоритмічно-програмну надбудову над наявними оркестраторами без необхідності зміни їх ядра, забезпечив підтримку прийняття рішень на основі даних телеметрії SIEM/EDR систем.

Апробація результатів дисертації. Основні теоретичні та практичні результати були представлені та обговорені в ході ряду наукових конференцій:

1. Workshop on Cybersecurity Providing in Information and Telecommunication Systems (CPITS), 2023.

2. 14-ї Всеукраїнської науково-практичної конференції «Актуальні проблеми управління інформаційною безпекою держави», 2023.

Публікації. Основні результати дисертації висвітлено у 7 наукових публікаціях, із них 4 – одноосібні, 3 – у співавторстві: 5 статей (з них 1 у співавторстві) у наукових виданнях, включених на дату опублікування до переліку наукових фахових видань України; 1 стаття (у співавторстві) у періодичному науковому виданні, проіндексованому в наукометричній базі даних Scopus. Наукові результати дисертації повною мірою висвітлено у наукових публікаціях.

Особистий внесок здобувача. Дисертація є самостійною науковою працею, в якій висвітлено власні ідеї і розробки автора, що дозволили вирішити поставлені завдання. Робота містить теоретичні та методичні положення і висновки, сформульовані здобувачем особисто. Використані в дисертації ідеї, положення чи гіпотези інших авторів мають відповідні посилання і використані лише для підкріплення ідей здобувача. Безпосередньо автором розроблено новий метод (CoopEvo-CloudSec) кооперативно-еволюційного розподілу ОбР у ХМС з урахуванням безпекового ризику, багатокритеріальну оптимізаційну модель розподілу ресурсів ХМС, удосконалену теоретико-ігрову модель конфліктної взаємодії «атакуючий–захисник», програмні модулі та архітектурну схему інтеграції методу CoopEvo-CloudSec, включаючи модулі Risk Analyzer та Policy Adapter.

У статті «Методика розслідування інцидентів інформаційної безпеки» опублікованій у співавторстві, внесок Цирканюк Д. А. полягає у зборі статистичних даних, систематизації інцидентів інформаційної безпеки та їхня класифікація при розслідуванні, що загалом складає 60% тексту статті.

У статті «Модель розподілу обчислювальних задач у хмарній інфраструктурі з урахуванням продуктивності, вартості та безпеки» опублікованій одноосібно, внесок Цирканюк Д. А. полягає у розробці моделі розподілу обчислювальних задач у хмарній інфраструктурі, що загалом складає 100% тексту статті.

У статті «Розширена гібридна модель з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища» опублікованій одноосібно, внесок

Цирканюк Д. А. полягає у розробці гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища, що загалом складає 100% тексту статті.

У статті «Метод багатокритеріальної оптимізації безпеки хмарних обчислень на основі модифікованого алгоритму NSGA-II» опублікованій одноосібно, внесок Цирканюк Д. А. полягає у розробці Метод багатокритеріальної оптимізації безпеки хмарних обчислень та проведені експериментального дослідження, що загалом складає 100% тексту статті.

У статті «Метод CoopEvo-CloudSe для оптимізації обчислювальних ресурсів хмарних систем для підвищення безпеки» опублікованій одноосібно, внесок Цирканюк Д. А. полягає у розробці методу CoopEvo-CloudSe та валідації його меж застосування, що загалом складає 100% тексту статті.

У статті «Protection of Information in Telecommunication Medical Systems based on a Risk-Oriented Approach» опублікованій у співавторстві, внесок Цирканюк Д. А. полягає у розробці моделі захисту інформації в інформаційно-телекомунікаційних медичних системах, що загалом складає 30% тексту статті.

У статті «Базові засади побудови центру кібербезпеки об’єктів ядерної енергетики» опублікованій у співавторстві, внесок Цирканюк Д. А. полягає у розробці моделі реалізації загроз інформаційній системі об’єкту ядерної енергетики, що загалом складає 30% тексту статті.

Структура та обсяг дисертаційної роботи. Дисертація складається зі вступу, трьох розділів, висновків, списку використаних джерел із 156 найменування на 18 сторінках і4 додатків. Загальний обсяг роботи становить 242 сторінки серед яких 149 сторінок основного тексту, 22 рисунків і 31 таблиця.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ І МОДЕЛЕЙ ОПТИМІЗАЦІЇ РЕСУРСІВ ХМАРНИХ СИСТЕМ ДЛЯ ПІДВИЩЕННЯ КІБЕРБЕЗПЕКИ

1.1. Хмарні системи як об’єкт дослідження розподілу обчислювальних ресурсів в системах кібербезпеки

Хмарні обчислення (ХО) як парадигму надання обчислювальних послуг відзначають наступними властивостями:

еластичність;

масштабованість;

розподіленість ресурсів.

Ці та інші властивості створили як нові перспективи для оптимального використання ІТ-інфраструктури, так і низку специфічних проблем у сфері інформаційної безпеки (ІБ).

Архітектурні моделі або сервісні рівні Infrastructure as a Service (IaaS), Platform as a Service (PaaS) та Software as a Service (SaaS) розрізняють за ступенем контролю користувача над інфраструктурою, рівнем абстракції та відповідальностей між провайдером і споживачем послуг. А, отже, це безпосередньо вплине на підходи до планування і розподілу ресурсів. Так саме й на заходи щодо захисту інформації. У IaaS споживач отримує найбільш гнучкий контроль над обчислювальними ресурсами. Останні – це віртуальні машини (ВМ), мережеві об’єкти, дискові масиви тощо. Відтак завдання оптимізації охоплює низку технічних параметрів. Зокрема, йдеться про віртуалізацію, розміщення ВМ, маркування ресурсів за важливістю та політиками ізоляції [1].

На рівні PaaS питання розподілу ресурсів пов’язане більше з розгортанням контейнерних середовищ, управлінням середовищем виконання та балансуванням навантаження на сервіси. А у SaaS модель орієнтована на оптимізацію функціональних запитів користувачів і масштабування сервісних

компонентів. Отже, багатокористувацькість хмарних систем формує додаткові ризики. Мова йде про спільне використання фізичних ресурсів, логічну ізоляцію й питання витоку даних або побічних каналів. З урахуванням останніх обставин суттєвими чинниками при ухваленні з використовування відповідної моделі ХО стає рішення про розміщення задач в хмарі. Зазначимо, що, оптимізація розподілу ресурсів у хмарних середовищах (далі ХМС) нездійснена без одночасного врахування показників (метрик) безпеки. Як показано в [2] це зокрема політики ізоляції, оцінки ризику компрометації вузлів, пріоритизація сервісів та можливостей для оперативної реакції на інциденти. Теоретичне та практичне обґрунтування цього висновку підтверджено релевантними науковими дослідженнями з проблематики безпеки хмарних сервісів, які виділяють багатокритеріальну природу задачі та необхідність інтегрованих рішень [1–20].

Зокрема, у [1, 2] автори систематизували загрози, притаманні різним сервісним моделям. В [2] доведено, що стратегії розподілу ресурсів мають враховувати специфічні вектори атак. Зокрема, для середовищ IaaS значущою є загроза компрометації гіпервізора. Це вимагає впровадження посилених механізмів ізоляції ВМ при їх розміщенні на фізичних хостах. Для PaaS та SaaS основні ризики пов’язані з безпекою API та управлінням ідентифікацією. Це випливає з того що зловмисники можуть експлуатувати слабкості в інтерфейсах керування для несанкціонованого доступу (НСД) до ресурсів інших користувачів. А отже ефективний розподіл ресурсів – не лише завдання технічної оптимізації, а й впровадження проактивних засобів зниження ризиків. Тому політики безпеки ХО зумовлюють допустимі конфігурації та межі взаємодії між компонентами хмарної системи [3].

Як довели автори багатьох досліджень, у ХМС ефективний розподіл обчислювальних ресурсів – це одне із головних завдань. Саме розподіл ресурсів визначає продуктивність, енергоефективність, надійність та якість надання послуг у ХМС. Цей аспект привернув значну увагу багатьох науковців [1–20] в останні роки. Саме тому доцільно розглянути основні результати попередніх

досліджень з цієї проблематики, оскільки ці та інші результати досліджень створили підґрунтя для синтезу наукових положень висвітлених далі у даній дисертаційній роботі.

Зокрема, Saidi K. разом зі співавторами у [4] представили огляд основних методів та моделей розподілу ресурсів у ХО. Автори класифікували методи за критеріями енергоефективності, QoS (тобто – Quality of Service – якість обслуговування), масштабованості та орієнтації на віртуалізацію. Акцент у статті зроблено на необхідності гнучкого управління ХО в умовах змінного навантаження.

Senthilkumar G. інші автори у [5] зосередилися на класифікації наявних алгоритмів розподілу ресурсів за напрямками оптимізації. Автори розглянули всі релевантні методи, починаючи від евристичних до методів та моделей на основі використання штучного інтелекту (ШІ). В статті проаналізовані переваги та обмеження кожного класу методів. Проте гібридні методи розподілу ресурсів у ХО авторами не розглянуті.

У [6] Rabaaoui S., Hachicha H., та Zagrouba E. (2024) запропонували автономний підхід до розподілу ресурсів із застосуванням оптимізованого планування задач. Стаття містила доволі ґрунтовну експериментальну оцінку ефективності алгоритму в реальному ХМС, проте автори не врахували деталі забезпечення захисту. Автори сфокусувалися у статті виключно на економічних та продуктивних метриках розподілу ресурсів у хмарних системах. Зокрема, розглянуто таки показники, як мінімізація вартості ВМ, час виконання, час відгуку та рівень енергоспоживання. Однак, наведена у статті [6] модель не інтегрувала елементи кібербезпеки, зокрема не розглянуто як вплине на розподіл ресурсів DDoS або EDoS-атака чи маніпуляції ресурсами зловмисниками.

У [7] Jabber S. A., Hashem S. H. та Jafer S. H. виконали огляд та порівняльний аналіз методів розподілу ресурсів та планування задач у ХМС. Проте це саме оглядова робота у якій головне питання це проблематика інтеграції цих двох процесів у ХМС для досягнення узгодженої продуктивності та оптимального використання ресурсів. Питання впливу на розподіл ресурсів безпекових в статті

не досліджено. Так само у [8] Kareem Awad W. зі співавторами здійснили систематизований огляд релевантних стратегій розподілу ресурсів та алгоритмів планування задач у ХМС. Проте на відміну від попередньої роботи акцент зроблено на перспективі гібридних підходів розподілу ресурсів та методах багатокритеріальної оптимізації у великих ХМС.

Mohamed A. зі співавторами у [9] проаналізували потенціал використання програмно-конфігурованих мереж (або SDN – Software-defined Networking) для покращення управління ресурсами у ХМС. Автори аналізували механізми централізованого контролю та гнучкого переналаштування мережевих ресурсів залежно до зміни у навантаженні. Проте безпекові питання не розглянуто.

Vergara J., Botero J. та Fletscher L. у [10] відобразили узагальнення підходів до розподілу ресурсів у гібридних середовищах. Для покращення розв'язання задачі розподілу ресурсів ХМС в дослідженні автори рекомендували урахувати просторову близькість пристроїв та їхніх мережевих параметрів.

Mohammad A. та Abbas Y. у [11] вивчали проблематику впровадження механізмів розподілу ресурсів у малих і середніх підприємствах. Автори дослідження наголосили, що на цей процес істотно впливають обмеженість інфраструктури, бюджетні обмеження та нестача кваліфікованих кадрів. Проблематика забезпечення параметрів безпеки такого впровадження у статі не досліджена.

У статті [12] Lekkala C. дослідив застосування моделей прогнозування, заснованих на ШІ, для завдання управління ресурсами в хмарі. В дослідженні представлено систему, що поєднала машинне навчання (МН) з алгоритмами оптимізації. Проте кількість критеріїв для оптимізаційної задачі обмежена економічними показниками та параметрами продуктивності. Lekkala C. переважно акцентував у статті увагу саме на використанні ШІ для синтезу передбачувальних алгоритмів розподілу ресурсів у ХМС, з метою оптимізації часу під змінні навантаження. Однак модель не інтегрувала елементи кібербезпеки.

Ashawa M. зі співавторами у [13] дослідив використання LSTM-алгоритмів для балансування навантаження у ХМС. Проте реальної реалізації представленої LSTM мережі в роботі не наведено. Статтю у підсумку автори відкликали.

У Manzoor M. F. зі співавторами у статті [14] реалізували огляд сучасних методик розподілу ресурсів у ХМС. Дослідники також аналізували перспективні напрямки розподілу навантаження у ХМС. Зокрема застосування самоорганізованих систем й інтеграцію з IoT-платформами. Проте жодних конкретних прикладів використання цих методик в роботі не надано. Автори зосередилися переважно на оглядовому контенті. Також не зачеплені безпекові фактори які вливають на розподіл ресурсів у ХМС.

Gong S. зі співавторами у [15] запропонували модель адаптивного багатозмінного управління для розподілу кількох типів ресурсів у ХМС. Метод передбачав корекцію параметрів контролю відповідно до змін у запитах на обслуговування. Проте як впливають на багатозмінне управління безпекові фактори залишилося поза увагою дослідників, які віддали перевагу економічним показникам ХМС.

У [16] Madni S. H. H. разом із колегами подали систематизований огляд розвитку підходів до розподілу ресурсів у ХМС. Автори виділили перспективні напрямки еволюції алгоритмів розподілу ресурсів для управління SLA (тобто – Service Level Agreement (SLA) офіційна угода між постачальником послуг та клієнтом ХМС) та QoS.

Варто згадати й вже канонічну роботу Li C. та Li L.Y. [17]. Ще у 2012 році дослідники запропонували модель оптимального резервування ресурсів ХМС. Модель враховувала переважно економічні показники ХМС. Та саме на підставі економічних критеріїв дозволяла досягти збалансованого використання інфраструктури ХМС. Але автори також не досліджували безпекові елементи та завдання розподілу ресурсів у ХМС.

Проблематикою розподілу ресурсів ХМС займалися й українські науковці, зокрема, відмітимо роботи [18–25]. Так у статті [18] запропоновано гібридний метод розподілу ресурсів. Метод поєднав переваги централізованих та

децентралізованих моделей управління ХМС. Автори провели моделювання сценаріїв навантаження ХМС, проте безпекові фактори не досліджено. Гребенюк Д.С. та Давидов В.В. у [19] запропонували метод первинного виділення ресурсів на основі аналізу ієрархій. Метод дозволив структурувати прийняття рішень у ХМС, враховуючи пріоритети користувачів. І хоча метод аналізу ієрархій потенційно може врахувати й вимоги безпеки ХО, проте завдання розподілу ресурсів безпосередньо цей метод для великої кількості критеріїв коректно не працює. Але на цьому ми запинимося далі у наступному підрозділі дисертації, який безпосередньо присвячено розгляду та аналізу саме математичних методів та моделей для завдання оптимізації розподілу обчислювальних ресурсів ХМС для підвищення безпеки.

Дисертаційна робота Петровської І. Ю. [20], а також інші її дослідження [21–24] присвячені розробці методів розподілу ресурсів під час надання хмарних інфраструктурних послуг. Авторка разом із колегами запропонувала власну класифікацію підходів і методику оцінювання ефективності на основі багатокритеріального аналізу. Зазначимо, що роботи [21–24] переважно фокусовано на методах розподілу ресурсів у хмарному середовищі IaaS, спрямованих на підвищення ефективності використання ресурсів, балансування навантаження, використовуючи декомпозицію ресурсів. Хоча в одній з публікацій Петровської І. Ю. [22] згадано «security in cloud environment» у завдані розподілу ресурсів для опрацювання даних, це обмежено загальними елементами, такими як базовий захист даних під час розподілу, без глибокого аналізу загроз чи моделювання антагоністичних взаємодій.

Зазначимо, що різні моделі надання хмарних сервісів істотно різняться. Це елементи рівню контролю користувача, об’єктів оптимізації та безпекових ризиків. Це визначає специфіку завдань розподілу обчислювальних ресурсів і зумовлює потребу в окремому аналізі специфіки кожного сервісного рівня ХМС. Тому таки узагальнені характеристики подано далі у табл. 1.1, яку складено на підставі аналізу робіт [1–25].

Таблиця 1.1

Порівняння сервісних моделей у хмарних обчисленнях (складено автором на підставі аналізу робіт [1–25])

Параметр	Модель						
	IaaS	PaaS	SaaS	FaaS	CaaS	XaaS	EaaS
Контроль користувача над інфра-структурою	Високий. ВМ, мережі, сховища тощо	Середній Середовище виконання	Низький. Готові застосунки	Дуже низький. Лише функції, код	Середній. Контейнери та оркестрація	Варіативний. Все як сервіс	Обмежений. Орієнтований на периферійні ресурси
Об’єкти оптимізації розподілу	ВМ, міграція, SLA	Планування сервісів, баланс навантаження	QoS для користувачів, масштабування застосунків	Запуск функцій, мінімізація латентності	Оркестрація контейнерів, масштабування	Інтеграція різних сервісів у єдиний ланцюг	Розподіл завдань між хмарою і вузлами
Ризики безпеки	Атаки на гіпервізор, побічні канали	Вразливості middleware, ізоляція застосунків	НСД до даних	Витоки через події, підміна функцій	Компрометація контейнерів, атаки на систему автоматичного управління контейнерами	Єдині точки відмови, мульти-тенантність	Небезпечні вузли, фізичний доступ
Політики ізоляції	Хост-ізоляція, мережеві сегменти	RBAC, пісочниця	Контроль доступу на рівні застосунку	Часове обмеження функцій, пісочниця	Простори імен, контрольні групи, мережеві політики	SLA, інтеграційні політики	Мобільні профілі, географічне сегментування
Вплив на рішення оптимізації	Необхідно враховувати апаратний рівень ХО	Оптимізація середовища виконання	Оперативне масштабування під навантаження	Орієнтація на обчислення, чутливі до затримок	Оркестрація кластерів, відмовостійкість	Уніфікація багатьох моделей	Баланс між периферійними та хмарними ресурсами
Складнощі забезпечення безпечного розподілу	Забезпечення продуктивності та ізоляції	Захист контейнерних середовищ	Шифрування даних, доступ	Контроль подій, довіра до тригерів	Захист від бічного переміщення у кластерах	Глобальні SLA, сумісність стандартів	Непередбачуваність мобільних вузлів

Як показав аналіз наукових публікацій [1–25], управління ресурсами у ХМС має багатовимірний характер. Завдання управління ХМС залежать не лише від обраної архітектури чи типу сервісної моделі. На них також впливають показники співвідношення продуктивності, надійності та вимоги забезпечити безпеку ХО [26–32]. Отже для моделі IaaS головна задача – це оптимізація віртуалізації, міграція ВМ та управління SLA. При цьому враховуємо загрозу атак на гіпервізор і потребу ізолювати хмарне середовище. У PaaS пріоритет зміщений в бік контейнеризації, масштабування сервісів і захисту API. Відповідно ще створює певне ризики доступу до середовищ виконання. У моделі SaaS зусилля сконцентровано на забезпеченні QoS для кінцевих користувачів. А також на контролі доступу до застосунків. У FaaS головна проблема – це контроль подій у середовищах, які орієнтовані на короткотривале виконання функцій. У CaaS передбачено пріоритетне розв’язання завдань оркестрації контейнерів і захист від атак на механізми автоматичного управління. Концепція XaaS прагне інтегрувати всі моделі в єдину сервісну парадигму. Отже для XaaS проблематику безпеки зведено до узгодження SLA та мінімізації ризиків. І, нарешті, у EaaS акцент перенесено використання периферійних та хмарних ресурсів. Отже тут, визначальним є фактор фізичного доступу та непередбачуваності мобільних вузлів. Зауважимо, що більш детально на питанні факторів безпеки для різних моделей ХМС ми запинимося окремо в межах наступного параграфу роботи. Поточний параграф завершимо узагальненим аналізом проаналізованих досліджень [1–25], оскільки розглянуті роботи засвідчили, що підходи до оптимізації розподілу ресурсів у ХМС істотно різняться як за використаними методами, так і за орієнтацією на різні критерії ефективності.

Узагальнений аналіз досліджень [1–32] засвідчив, що підходи до оптимізації розподілу ресурсів у ХМС істотно відрізнялися, як за використаними методами, так і за орієнтацією на різні критерії ефективності.

В табл. 1.2 частина робіт сфокусована переважно на економічних показниках і продуктивності, зокрема [16, 17]. В інших, зокрема [4, 8, 15, 20–25], акцент дослідники роблять на масштабованості, QoS чи енергоефективності.

Таблиця 1.2

Методи оптимізації ресурсів у хмарних системах за критеріями

(складено автором на підставі аналізу робіт [1–25])

Автори та джерело	Метод	Критерії оптимізації	Специфіка та обмеження
Saidi K. та ін. [4]	Класифікація методів розподілу ресурсів	Енергоефективність, QoS, масштабованість	Акцент на гнучкому управлінні; безпекові фактори не враховані
Senthilkumar G. та ін. [5]	Систематизація алгоритмів (евристичні, ШІ)	Продуктивність, вартість, масштабованість	Не розглянуті гібридні методи
Rabaaoui S. та ін. [6]	Автономне планування задач ХМС	Вартість ВМ, час виконання, енергоспоживання	Не враховано фактори захисту від DDoS, EDoS атак
Jabber S. A. та ін. [7]	Огляд методів планування і розподілу ресурсів ХМС	Продуктивність, узгодженість процесів	Не інтегровано фактори кібербезпеки
Kareem Awad W. та ін. [8]	Гібридні підходи та багатокритеріальна оптимізація ХМС	Продуктивність, QoS, масштабованість	Зроблено акцент на великих ХМС; безпека розглянута поверхово
Mohamed A. та ін. [9]	Використання SDN	Гнучке управління мережевими ресурсами	Безпекові питання не розглянуті
Vergara J. та ін. [10]	Просторово-мережевий підхід	Близькість пристроїв, мережеві характеристики	Обмежений аналіз безпеки
Mohammad A., Abbas Y. [11]	Методи для малих і середніх підприємств в завданні впровадження ХМС	Економічні та інфраструктурні обмеження ХМС	Акцент лише на ресурсних і бюджетних факторах
Lekkala C. [12]	Прогнозування на основі ШІ	Час виконання, вартість, продуктивність ХМС	Не враховані ризики атак та інцидентів
Manzoor M. F. та ін. [14]	Самоорганізовані системи, інтеграція з IoT	Масштабованість, узгодженість SLA	Теоретичний огляд; без практичних прикладів безпеки
Gong S. та ін. [15]	Адаптивне багатозмінне управління ХМС	Кілька типів ресурсів, QoS	Орієнтація на економіку; безпека не розглянута
Madni S. H. H. та ін. [16]	Систематизація SLA/QoS підходів ХМС	Виконання SLA, QoS	Аналіз еволюції методів; обмежений розгляд загроз
Li C., Li L. Y. [17]	Модель оптимального резервування ХМС	Економічні критерії	Не враховані кіберризики
Волк М.О. та ін. [18]	Гібридний метод управляння ХМС	Балансування наванта-ження, ефективність	Без аналізу безпеки
Гребенюк Д.С., Давидов В.В. [19]	Метод аналізу ієрархій	Пріоритизація користувачів	Складно масштабувати для багатьох критеріїв
Петровська І.Ю. та ін. [20–25]	Багатокритеріальні підходи (IaaS)	Балансування ресурсів, SLA, QoS.	Загальний розгляд security, без глибокого моделювання загроз

Проте ці роботи детально не розглядають безпекові критерії застосування ХМС. Також ці критерії не включені в перелік критеріїв для багатокритеріальної оптимізації.

Проведений в поточному параграфі аналіз свідчить, що завдання оптимізації розподілу ресурсів у ХМС розглядався на протязі останніх років науковою спільнотою з різних позицій. Проте більшість чинних підходів орієнтована переважно на досягнення економічної ефективності, підвищення продуктивності чи забезпечення масштабованості ХМС. Однак, відсутній аналіз ризиків для ХМС у більшості досліджень. Він розглядався лише як допоміжній чинник, як зокрема у роботах Петровської І. Ю. [20–25]. Така ситуація створила очевидний науковий розрив. Адже ХМС за своєю природою є багатокористувацькими та розподіленими. До речі, це зумовлює їхню підвищену вразливість до кіберзагроз. Відповідно, ефективність будь-якого методу оптимізації у ХМС має оцінюватися не лише з точки зору продуктивності та витрат. На подібну оптимізацію потрібно подивитися з позиції здатності ХМС протидіяти атакам і забезпечувати належний рівень довіри до обчислювального середовища.

Тому, з огляду на це, наступний параграф присвячено саме аналізу стану і досліджень саме у сфері безпеки ХМС.

1.2. Сучасний стан і тенденції досліджень у сфері безпеки хмарних систем. Аналіз попередніх досліджень

У релевантних дослідженнях останніх 10–15 років безпека ХМС розглядалася фахівцями як невіддільна (частина/ознака) частина процесу управління обчислювальними ресурсами. Зокрема, міжнародні стандарти серії ISO/IEC 27000 [33–37] та рекомендації NIST [38–42], визначили базові принципи, політики та механізми кіберзахисту. Вони обов’язково мають бути врахованими при проєктуванні та експлуатації ХМС. Саме на основі ISO/IEC NIST [33–42] сформовано методологію інтеграції заходів безпеки у процес розподілу ресурсів.

Розвиток хмарних технологій (ХТ), що пройшли шлях від віртуалізації окремих серверів до складних розподілених систем (моделі IaaS, PaaS, SaaS, FaaS, CaaS), суттєво трансформувала релевантні кіберзагрози для ХТ. В останні роки стан безпеки в хмарі, згідно даних [43–54] визначають протистоянням між ускладненням архітектури ХМС та варіативністю атакуючих стратегій та тактик зловмисників. Останні прилаштували свої методи атак під специфіку кожної сервісної моделі - IaaS, PaaS, SaaS, FaaS, CaaS. Отже для об'єктивної оцінки стану захищеності ХМС проаналізуємо актуальні статистичні дані міжнародних досліджень. Аналіз звітів провідних компаній у сфері кібербезпеки, таких як Cloud Security Alliance (CSA), Verizon DBIR, IBM X-Force та Palo Alto Networks Unit 42 [43–54], засвідчили, що основним джерелом ризиків сьогодні є не стільки зовнішні атаки нульового дня, скільки помилки конфігурації та людський фактор. Зокрема, згідно з дослідженням Flexera (2024) [47], 81% респондентів вказали на безпеку як на головну проблему при роботі з ХМС. Ця тенденція знайшла своє відображення в детальному розподілі основних загроз за типами хмарних сервісів. Для наочності на основі агрегованих даних з джерел [43–54], сформовано узагальнену статистику, яка представлено в табл. 1.3–1.5.

Таблиця 1.3

Частота хмарних інцидентів і основні причини у 2023-2024 роках
(сформовано автором на основі аналізу статистичних звітів [43–54])

Показник	Значення	Джерела
Частка організацій, які мали хоча б один хмарний інцидент у 2024	83 %	[51–54]
Частка організацій, які мали інциденти в публічному хмарному середовищі	60 %	[51–53]
Частка порушень, в яких була задіяна хмара (у 2023)	82 %	[51–54]
Частка «людської помилки» як причини витоку даних	88 %	[51–54]
Середній час реагування на хмарні інциденти	145 год.	[51–55]
Частка невірних конфігурацій та налаштувань ХМС, як джерело вразливостей	68 %	[43–54]

Як свідчать дані табл. 1.3, більшість організацій у 2023–2024 рр. стикалися з інцидентами в хмарних середовищах. Визначальними чинниками залишилися неправильні налаштування та людський фактор. Доволі значний показник

середнього часу реагування на інциденти вказує не негайну потребу вдосконалити механізми моніторингу та автоматизації процесів ліквідації наслідків атак на ХМС.

Таблиця 1.4

Загальні загрози в хмарі (класифікація CSA 2024) – частки звернень за оцінками експертів (сформовано автором на основі аналізу статистичних звітів [43–54])

Загроза / клас	Кількість згадок у статистичних звітах	Коментар Cloud Security Alliance
Неправильні налаштування та неналежний контроль змін.	Високо в рейтингу	CSA називає це топ-3 загрозу
Управління ідентифікацією та доступом (IAM).	Високо	Експерти часто виділяють IAM як одну з ключових проблем
Небезпечні інтерфейси та API.	Високо	Інтерфейси / API – це частий вектор атак
Небезпечні сторонні ресурси.	Згадують в звітах	Ризик від зовнішніх компонентів / постачальників
Обмежена видимість та спостережуваність інцидентів.	Згадують в звітах	Брак видимості в хмарі – це розповсюджена проблема в оцінках CSA

У табл. 1.4 наведено класифікацію загроз за оцінками експертів CSA. Аналіз статистичних звітів показав, що домінуючими є помилки конфігурації та управління доступом. Інші загрози, як-от небезпечні API чи сторонні ресурси, хоч і згадували у статистичних звітах [43–54] рідше, проте створюють додатковий ризик у ХМС.

Диференціація загроз за моделями обслуговування (див. табл. 1.5) засвідчила, що специфіка ризиків суттєво різниться для моделей IaaS, PaaS, SaaS, FaaS та CaaS. Зокрема, для IaaS притаманними є проблеми з конфігураціями та внутрішні загрози. А для SaaS згідно статистичних звітів провідними є атаки на облікові дані та втрати даних. Отже аналіз підтвердив, що забезпечення безпеки ХМС неможливе без урахування конкретної моделі сервісу.

Таблиця 1.5

Основні загрози безпеки та частота їх прояву в різних хмарних моделях обслуговування (2023–2024 рр.) (сформовано автором на основі аналізу статистичних звітів [43–54])

Модель	Загрози			Коментар та джерела	Дані, %
	№1	№2	№3		
IaaS	Небезпечна конфігурація	Компрометація облікових записів	Внутрішні загрози	За даними [43–54] компаній мають як мінімум одну небезпечну конфігурацію в середовищі IaaS	80
PaaS	Небезпечні програми та APIs	Небезпечна конфігурація середовища	Нелегітимне використання сервісів	CSA виділяє AP як головну загрозу для PaaS [43, 44]	—
SaaS	Небезпечна конфігурація	Компрометація облікових записів	Втрата даних	За даними [46] 68% порушень даних у SaaS пов'язані з людським фактором – помилки, фішинг	68
FaaS	Небезпечні функції	Надмірні дозволи	Події ін'єкції.	Звіт [49] показав що понад 70% організацій використовували FaaS-функції з надмірними дозволами	70
CaaS	Небезпечні образи контейнерів	Небезпечна оркестрація	Компрометація реєстру	За даними [43–54] 56% організацій використовували образи контейнерів з певними вразливостями	56
XaaS	Людський фактор (помилки конфігурації)	Фішинг та компрометація облікових записів	Внутрішні загрози	За даними [47, 48] у 2024 році 81% респондентів назвали безпеку головною проблемою ХМС	81

Згідно з табл. 1.6 та рис. 1.2, у відповідь на зростання інцидентів компанії акцентували свою увагу на підвищенні інвестицій у SaaS-безпеку, розвиток платформ управління CSPM та інтеграцію принципів Zero-Trust.

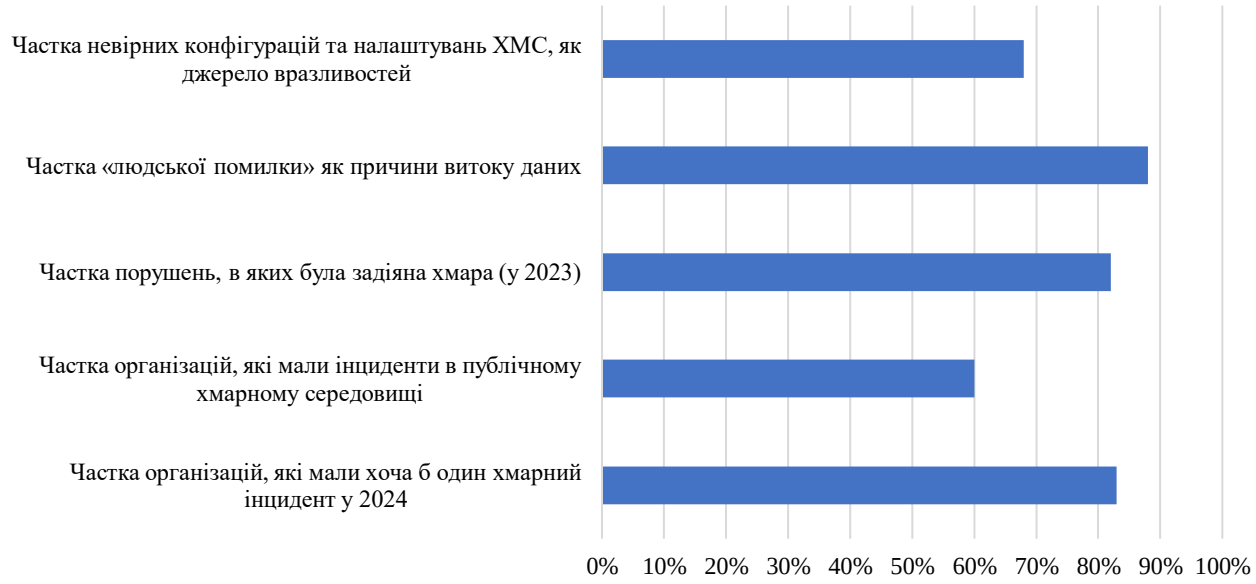


Рис. 1.1. Частота хмарних інцидентів і основні причини у 2023–2024 роках (Сформовано автором на основі табл. 1.3 та аналізу статистичних звітів [43–54])

Аналіз тенденцій захисту засвідчив тенденцію переходу від реактивних до проактивних стратегій захисту. Така стратегія, зокрема, передбачає постійне вдосконалення системи управління доступом. А також вживання, як провайдером, так і споживачем хмарних послуг певних заходів, виявлення помилок конфігурації та інтеграцію засобів безпеки у життєвий цикл ХМС.

Таблиця 1.6

Тенденції в захисті хмарних середовищ

(систематизовано автором на основі аналізу статистичних звітів [43–54])

Напрям захисту	Частка, %	Джерела
Частка організацій, що підвищила бюджет на SaaS-Безпеку.	76	[43, 44]
Частка організацій, в яких SaaS безпека є пріоритетом.	86	[43, 44]
Частка організацій, де виправлення неправильних налаштувань – пріоритетний напрям захисту.	>50	[43–54]
Частка організацій, які відчули порушення даних, пов’язані з SaaS / хмарою.	28	[43–54]
Частка організацій, що використовували єдину платформу для управління безпекою хмари (згодні, що це допоможе)	95	[54]
Частка організацій, які очікували збільшення бюджету на безпеку хмари в наступні 12 місяців.	61	[54]

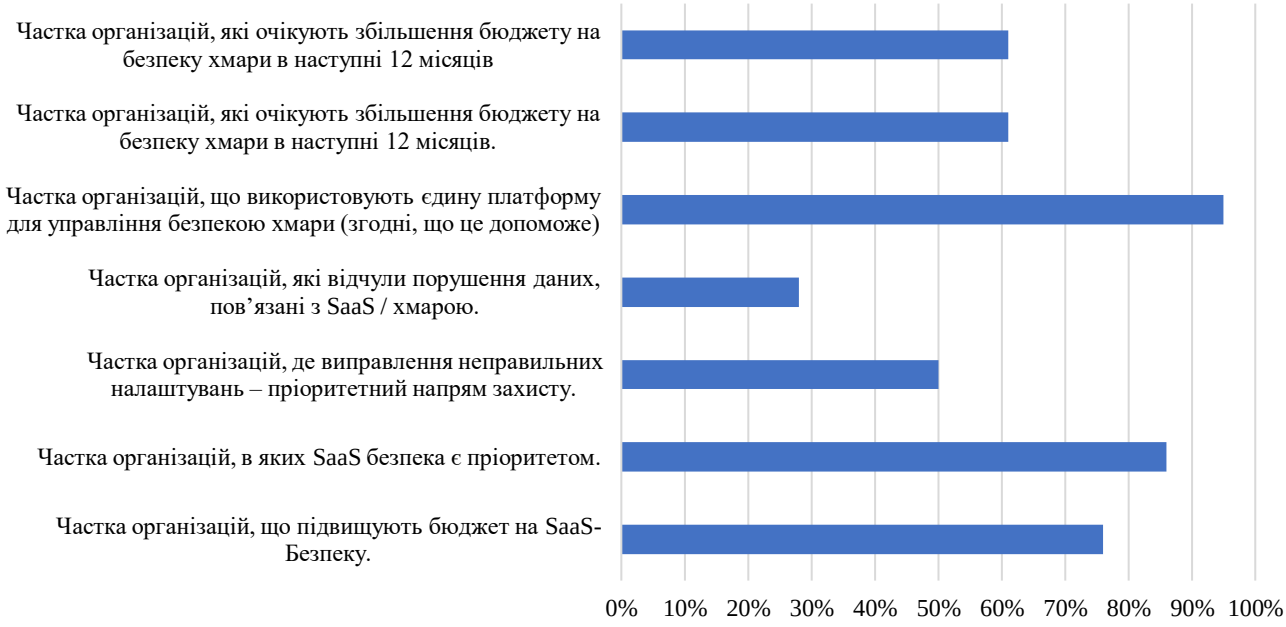


Рис. 1.2. Тенденції в захисті хмарних середовищ (опрацьовано автором на основі аналізу табл. 1.5 і 1.6 та статистичних звітів [43–54])

Порівняння основних векторів атак (див. табл. 1.7, та рис. 1.3) показало, що середовище SaaS найбільше потерпає від фішингу та атак на облікові дані. Моделі IaaS і CaaS є більш вразливими до криптоджекінгу та небезпечних конфігурацій ХМС.

Доволі значна частка (більш ніж 54 %) інцидентів у PaaS та FaaS зумовлена атаками через сторонні залежності. Дані досліджень [43–54] підтвердили, що сучасні атаки вирізняються високою варіативністю. А, отже, реагування на подібні атаки потребує з боку провайдера та споживача хмарних послуг комплексного підходу. Це стосувалося як процесу виявлення атак так й реагування на них.

Як узагальнення аналізу статистичних даних [43–54], у табл. 1.8 наведений розподіл пріоритетів компаній до захисті ХМС. Дані систематизовано за опитуваннями компаній у [43–54].

Таблиця 1.7

Статистика основних векторів атак на хмарні середовища (2023–2024 рр.)

(систематизовано автором на основі аналізу статистичних звітів [43–54])

Вектор атаки	Частота згадування у статистичних звітах	Основна модель, що є потенційно уразливою	Показник, %	Джерела
Фішинг	Дуже висока	Усі	35	[46]
Небезпечна конфігурація	Висока	IaaS, PaaS, SaaS та інші.	70	[45]
Криптоджекінг (нелегітимне використання обчислювальних ресурсів пристрою).	Середня	IaaS, CaaS через вразливі контейнери.	54	[43–54]
Атака через сторонні залежності. Зокрема, за даними [52-54] зловмисники знаходили способи ввести шкідливий код у популярну відкриту бібліотеку. У [52-54] згадані GitHub, PyPI.	Середня	PaaS, FaaS, CaaS.	85	[46]
Атаки на облікові дані.	Висока	IaaS, SaaS.	40	[46]
Вимагання.	Висока	Усі	52	[43–54]

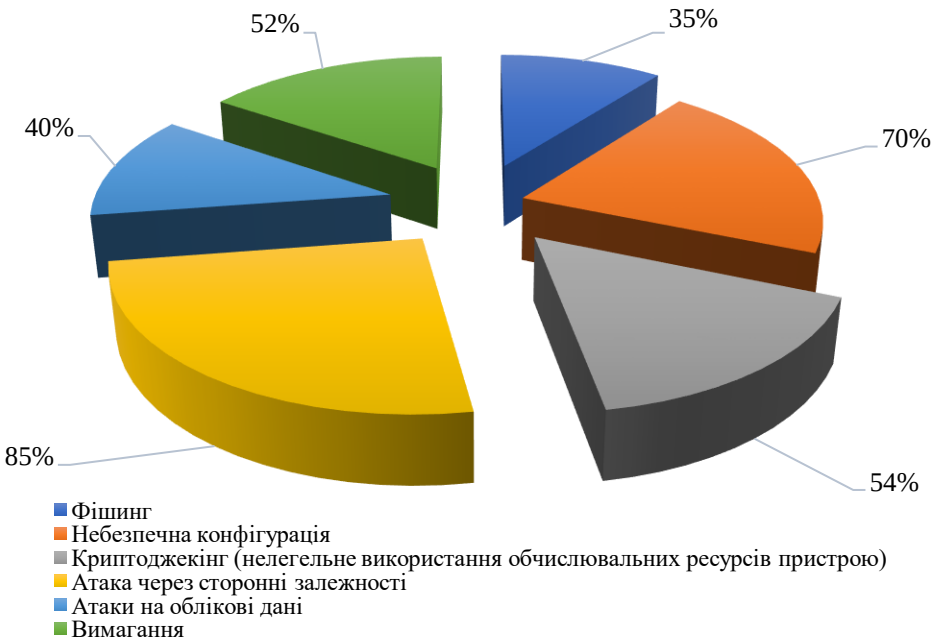


Рис. 1.3. Візуалізація статистики основних векторів атак на хмарні середовища (2023–2024 рр.) (опрацьовано автором на основі аналізу статистичних звітів [43–54])

Отримані під час аналізу звітів [43–54] результати засвідчили, що компанії пріоритетно впроваджували CSPM, IAM, DLP та інші технології, орієнтуючись на автоматизацію й інтеграцію захисту у процеси управління ресурсами. Отже, безпеку ХМС все більше сприймають як невід'ємну складову оптимізації використання обчислювальних ресурсів.

Отже, узагальнений статистичний аналіз підтвердив, що стан безпеки ХМС визначають не лише кількісними показниками інцидентів. Значущою є й структуризація загроз.

Проте, проаналізована статистика дала лише загальну картину ризиків для ХМС, а також тенденцій підвищення захищеності ХО. Для глибшого розуміння методів протидії та ефективних підходів до інтеграції безпеки у процес управління ресурсами ХМС доцільно звернутися до результатів наукових досліджень.

Саме тому у поточному параграфі розглянуті релевантні публікації за останні роки, де автори запропонували різноманітні методи, моделі, методології та алгоритмічні рішення для забезпечення захисту ХМС від загроз, що постійно трансформуються.

Alzoubi Y.I., Mishra A., Topcu A.E. у [55] виконали систематичний огляд застосувань методів машинного та глибинного навчання (ML/DL) у задачах захисту хмарних середовищ. Автори класифікували основні напрямки застосування ML/DL, а також моделі для контролю доступу та виявлення фішингових кампаній.

У [56] Gupta K. та інші співавтори запропонували прикладну архітектуру на базі федеративного навчання для задачі виявлення потенційно зловмисних користувачів в розподілених сценаріях зберігання і доставлення даних. Методологія, яку автори запропонували у статті полягала у федеративному навчанні локальних моделей на вузлах постачальників та споживачів даних. Потім виконувалася агрегація параметрів на центральному сервері для отримання прогнозу зловмисної поведінки. Серед обмежень цього дослідження слід зазначити питання приватності агрегованих оновлень.

Таблиця 1.8

Пріоритети у захисті хмарних середовищ за опитуваннями компаній (систематизовано автором на основі аналізу статистичних звітів [43–54])

Модель	Пріоритет	Рівень пріоритету	Коментар та джерело
IaaS, PaaS, CaaS	CSPM (Cloud Security Posture Management)	Високий	За даними [47] у 2024 році 75 % компаній використовували або планували використовувати CSPM для автоматичного виявлення помилок конфігурації
Усі моделі	Ідентифікація та управління доступом (IAM)	Високий	Впровадження Zero-Trust архітектури для обмеження доступу за принципом найменших привілеїв
SaaS, IaaS, PaaS	Захист даних (Data Loss Prevention - DLP)	Високий	Зростає з поширенням роботи з конфіденційними даними в хмарі
CaaS, FaaS, PaaS	DevSecOps.	Середній, проте зростає	Інтеграція безпеки на ранніх етапах циклу розробки ХМС. Використання SAST/DAST для контейнерів
PaaS, SaaS (для власних додатків)	SWA (Secure Web Application)	Високий	Захист веб-додатків і API від OWASP Top-10 загроз
XaaS	SASE (Secure Access Service Edge)	Зростає	Конвергенція мережевої безпеки та безпеки як сервісу

Dhinakaran D. зі співавторами у [57] запропонував концептуальну гібридну криптографічну схему для безпечного розміщення та обробки даних у хмарі. Схема поєднала класичні схеми забезпечення конфіденційності. Проте саме оптимізаційний компонент розподілу ресурсів на виконання такого завдання в роботі не досліджено.

У [58] Banse C. зі співавторами запропонували власний підхід до побудови графа властивостей хмари. Граф уніфікував дані, отримані зі статичного аналізу коду та конфігураційних файлів хмарної системи. Така уніфікація дозволила формально зв’язувати вразливі ділянки коду з конкретними конфігураційними елементами й шляхами розгортання. Дослідження спростило час на виявлення потенційних наслідків некоректної конфігурації та налаштувань ХМС. Методологічно робота поєднала техніки побудови графових моделей з правилами валідації конфігурацій.

Süß F. разом зі співавторами у [59] довершили огляд релевантних загроз та заходів у сфері безпеки ХМС, Автори оновили класифікацію векторів атак. А також запропонували метрики для пріоритезації ризиків безпеці ХМС. Результати огляду наведені у [59] є актуальними для формування політик розподілу ресурсів. Саме класифікація векторів атак підкреслила необхідність враховувати різні класів інцидентів при прийнятті оптимізаційних рішень для ХМС.

У [60] Saxena D. та інші співавтори описали результати комплексного аналізу підходів та стратегій безпечного управління ресурсами у ХМС. Автори поєднали теоретичну класифікацію проблем з емпіричним оглядом методів оптимізації безпечного управління ресурсами ХМС. Також у дослідженні [60] автори систематизували рекомендації щодо інтеграції заходів безпеки у процеси планування та оркестрації ресурсів ХМС.

Moudni M. E. та Ziyati E. Y. [61] здійснили систематичний огляд проблематики безпеки ХМС. Головний акцент автори зробили на задачі збереження конфіденційності, цілісності й доступності даних у середовищах гібридній хмарі. Автори класифікували релевантні підходи до методів шифрування, політик доступу, контролю над віддаленим зберіганням у гібридній хмарі. Також у роботі визначені головні проблеми те гібридних хмар з погляду на захист від зловмисників. Це висока вартість криптографічних операцій, складність інтеграції протоколів перевірки у масштабовані системи та відсутність єдиних стандартів для мультихмарних архітектур.

У досліджені [62] Chauhan M. та Shiaeles S. проаналізували наявні фреймворки безпеки для ХМС. Автори у статті аналізували обмеження фреймворків й вказали на брак гнучкості у більшості нормативних моделей. В статті автори запропонували інтеграцію традиційних фреймворків з новітніми технологіями. Зокрема, розглянуто перспективи використання AI-driven IDS, Zero-Trust та DevSecOps.

Kumar B. S. A. та Sah B. у статті [63] узагальнили сучасні методи та моделей управління ресурсами з акцентом на безпеку ХМС. Як критерії автори розглянули показники енергоспоживання, SLA-орієнтовану оптимізацію, балансування навантаження, а також інтеграцію безпекових метрик у функції вартості.

Концептуально ця робота наближена до завдань поточної дисертації. Однак, автори не надали в роботі конкретної моделі для розв'язання завдань управління ресурсами ХМС. Робота лише декларативно наголошує на актуальності підходів на основі багатокритеріальної оптимізації та методів штучного інтелекту.

У [64] Ogundapo A. та Ezeaputa V. N. розглянули ризики безпеки у публічних хмарних середовищах. Акцент в роботі автори зробили саме на проблематиці управління доступом та мультиорендності ХМС. Автори запропонували концептуальні підходи до управління ризиками. Зокрема, ці заходи передбачали посилення політик IAM та проведення періодичних аудитів безпеки ХМС. Оптимізаційний компонент управління ХМС в роботі не зачеплений.

У [65] Khan M. A. зі співавторами провели загальний огляд проблем у сфері хмарної безпеки. Автори класифікували вектори атак та описали їхній вплив на ХМС. Дослідження має суто оглядові риси. Питання оптимізації не розглядалося.

Xu G. та Yu W. зі співавторами у [66] представили власний проєкт системи управління кібербезпекою на базі ХО. Автори продемонстрували можливості централізованого збору й аналізу телеметрії з різних вузлів та використання хмарної інфраструктури для масштабованого моніторингу інцидентів.

Робота Takahashi T., Kadobayashi Y. та Fujiwara H. [67] це одна з перших теоретичних спроб формалізувати проблематику безпеки ХМС через онтологічний підхід. У статті автори запропонували модель представлення знань. Модель дозволила формально описати загрози, уразливості й механізми захисту для ХМС.

У [68] Tissir N. разом зі співавторами навів семантичний огляд літератури з проблематики управління безпекою ХМС. Автори розробили концептуальну рамку, яка поєднала технічні та організаційні фактори кіберзахисту ХМС.

Gill S. S. та Buyya R. у [69] порекомендували застосувати архітектуру SECURE для ХМС. Архітектура поєднала механізми моніторингу ресурсів і адаптивного реагування на загрози. Головна мета дослідження – забезпечити самозахист хмарної інфраструктури. Автори інтегрували методи перерозподілу ресурсів з механізмами безпеки ХМС. Головний здобуток роботи полягав у формуванні парадигми «self-protection» у керуванні ресурсами. Ця парадигма є доволі

концептуально близькою до підходів які ми далі розглядаємо у дисертації, оскільки так саме поєднувала оптимізацію продуктивності й безпеки ХМС.

Автори дослідження [70] представили концептуальну модель управління ресурсами у розподілених ХМС. Модель інтегрувала зокрема й критерії безпеки безпосередньо у процес планування та алокації. Автори запропонували фреймворк, що поєднав моніторинг стану ресурсів із системою оцінки ризиків для ХМС. Цей фреймворк дозволив приймати під час експериментальних досліджень рішення про розподіл ресурсів. Рішення враховувало й можливі атаки або вразливості ХМС.

Bhardwaj A. та Goundar S. у [71] простежили взаємозалежність між параметрами безпеки та продуктивністю у ХМС. Автори запропонували модель, яка кількісно відображала компроміси між підвищенням рівня безпеки ХМС та пропускній здатності. Робота [71] обґрунтувала доцільність застосування багатокритеріальних методів оптимізації для задачі управління ресурсами ХМС де авторами одночасно враховано безпекові параметри та ефективність ХМС.

Shri S. J. зі співавторами у [72] сконцентрувалися на практичних методах оптимізації управління ресурсами із урахуванням безпеки даних ХМС. У дослідженні розглянуто алгоритми балансування навантаження. Також автори аналізували методи шифрування та доступу. Проте багатокритеріальна оптимізація в статті не розглядалася.

У [73] Mishra S. D. разом з колегами представили систематизований аналіз проблем кібербезпеки у хмарних платформах. Автори описали типові вектори атак, а також механізми протидії та роль провайдерів у забезпеченні ефективного захисту.

У [74] Jhawar R., Piuri V. та Samarati P. дослідили головні труднощі забезпечення безпеки ХМС. Автори запропонували формалізовану модель, де вимоги безпеки розглянуто поряд з традиційними SLA-параметрами.

У [75] автори дослідили проблему відсутності тісного зв’язку між практиками інженерії ХМС та засобами забезпечення кібербезпеки. Автори запропонували методологічний підхід для інтеграції засобів захисту на всіх етапах життєвого циклу хмарних сервісів.

Jimmy F. N. U. у [76] виконав аналіз сучасних інструментів безпеки хмарних середовищ. Зокрема проаналізовано переваги CSPM, SIEM, EDR та засобів Zero-Trust. Автор оцінив їхню ефективність для усунення вразливостей і зниження ризику атак у ХМС.

Aldhyani T. H. та Alkahtani H. у [77] проаналізували проблематику EDoS-атак у ХМС. Для мінімізації наслідків EDoS-атак автори запропонували застосувати алгоритми ШІ. Автори довели, що методи ML здатні виявляти аномальні шаблони використання ресурсів.

У [78] Feng D., Qin Y. та інші співавтори дослідження навели детальний огляд концепції конфіденційних обчислень. Концепція передбачає використання апаратних довірених середовищ (TEE) для захисту даних у процесі опрацювання. Автори розібрали сучасні рішення, зокрема Intel SGX, AMD SEV, ARM TrustZone.

Jarkas O. зі співавторами у [79] здійснили комплексний огляд проблем безпеки контейнеризованих середовищ (CaaS). Автори класифікували відомі атаки на Docker, Kubernetes та інші системи оркестрації. Також у статті проаналізовані сучасні захисні механізми для ХМС.

У [80] VS D. P., Sethuraman S. C. та Khan M. K. втілили систематичний огляд проблем безпеки контейнерних середовищ. Зокрема у статті проаналізовано вразливості Docker та Kubernetes. Автори класифікували рівні захисту та запропонували відповідні стратегії пом’якшення ризиків для контейнерів.

У звіті [81] аналітики різних провайдерів та компаній у сфері безпеки ХМС репрезентували результати опитування організацій, що використовують мультихмарні середовища у бізнес-процесах. У звіті виділено п’ять проблем. Це відсутність єдиного контролю політик, ускладнене управління ідентичностями, ризики неправильної конфігурацій, зростання витрат на безпеку ХМС та труднощі інтеграції різних інструментів.

Так само у [82] містився технічний аналіз архітектур та механізмів конфіденційних обчислень у ХМС. Звіт аналізував технології Intel SGX, AMD SEV та ARM TrustZone для створення довірених виконуючих середовищ (TEE).

Jordan Smith E. K. у [83] розглянув використання штучного інтелекту для синтезу систем виявлення вторгнень (IDS), які орієнтовані саме на ХМС нового покоління. Автор акцентував увагу на перспективах застосуванні глибинного навчання для ідентифікації аномалій у ХМС. Також у статті розглянуто проблема масштабованості моделей глибинного навчання.

Резюмуючи огляд попередніх досліджень [55–83], відмітимо, що в останні роки у роботах, присвячених ХМС домінували, публікації, орієнтовані на розробку окремих методів протидії атакам. Також багато авторів присвятили свої дослідження фреймворків управління ризиками. Доволі багато публікацій у яких автори досліджували перспективу інтеграції систем ШІ в задачу виявлення загроз для ХМС. При цьому є певна різниця у методах якими користувалися автори, див. табл. 1.9. Одні автори зосередилися конкретних класах атак. Інші розглядали проблематику захисту ХО більш широко, намагаючись розробити універсальні моделі управління доступом та розподілу даних у ХМС. Окремо у параграфі проаналізовано роботи, які присвячено захисту хмарних середовищ, зокрема контейнерів. Для систематизації цих результатів аналізу попередніх досліджень за останні роки у табл. 1.9 наведено узагальнення робіт, у якому вказано об’єкт дослідження, методологічний підхід, внесок у розв’язання завдань захисту ХМС та підвищення їхньої продуктивності.

Узагальнення в табл. 1.9 показало, що попри значну кількість досліджень, переважна більшість із них зосереджувалася або на технічних аспектах безпеки, або на питаннях продуктивності та вартості ХМС. Проте, задача інтеграції критеріїв безпеки безпосередньо у процеси управління та оптимізації ресурсів лишилася недостатньо дослідженою. З огляду на це, у наступному параграфі проаналізовано методи та моделі оптимізації розподілу ресурсів хмарних середовищ з урахуванням критерію безпеки функціонування ХМС.

Таблиця 1.9

Систематизація та узагальнення аналізу публікацій (складено автором)

Автори, рік та джерела	Об’єкт дослідження	Використаний метод, модель	Внесок у сферу безпеки ХМС	Обмеження
Alzoubi Y.I., Mishra A., Topcu A.E., (2024) [55]	Методи МН для безпеки ХМС.	МН та глибоке навчання (або Deep Learning та ML)	Систематизовано методи DL та ML та використання AI у безпеці ХО.	Відсутня задача інтеграції DL та ML з управлінням ресурсами ХМС.
Gupta K. et al. (2024) [56]	Виявлення зловмисних користувачів у ХМС.	Методи федеративного навчання в задачах захисту ХО.	Захист розподілу даних у хмарі.	Робота орієнована виключно на модель FL для захисту ХМС.
Dhinakaran D. et al., (2024) [57]	Конфіденційність даних у ХМС.	Квантовий розподіл ключів (QKD).	Новий підхід до розподілу зберігання з QKD у ХО.	Обчислювальна складність методу.
Süß F., et al. (2024) [59]	Проблеми безпеки ХМС.	Огляд, класифікація методів захисту.	Актуалізовано релевантні загрози ХО.	Концептуальний рівень роботи.
Saxena D. et al. (2025) [60]	Задача розподіл ресурсів з урахуванням безпеки ХО.	Статистичний аналіз.	Формалізовано стратегії керування ресурсами ХМО з урахуванням ІБ.	Відсутні практичні алгоритми розв’язання задачі.
Kumar B., Sah B., (2024) [60]	Управління ресурсами та безпека ХМС.	Багатокритерійна оптимізація.	Систематизовано зв’язок розподілу ресурсів та безпеки ХМС.	Переважно оглядовий тип роботи. Конкретна модель відсутня.
Gill S., Buyya R., (2018) [69]	Самозахист ресурсів у ХМС.	SECURE Framework (набір принципів, політик, рекомендацій і контрольних механізмів, який допомагає організаціям: будувати й експлуатувати хмарну інфраструктуру безпечним способом).	Автоматичне реагування на атаки.	Орієнтовані на окремі сценарії атак на ХМС.
Jarkas O. et al. (2025) [79]	Безпека контейнерів у ХМС.	Огляд та аналіз атак та методів захисту для контейнерів ХМС.	Систематизація загроз для моделей CaaS та FaaS.	Акцент лише на захист контейнерів.
J. Smith E. K. (2025) [83]	IDS нового покоління.	IDS для ХМС на основі систем ШІ.	Адаптивне виявлення атак за допомогою ШІ.	Немає надійних наборів для навчання.

1.3. Аналіз попередніх досліджень із використання методів та моделей в задачі оптимізації розподілу ресурсів хмарних систем

Оптимізація розподілу ресурсів у ХМС з урахуванням метрик безпеки досліджувалася з використанням різних математичних методів та моделей, які умовно поділяють на наступні основні групи:

класичні методи планування і балансування;

математичні моделі багатокритеріальної оптимізації;

еволюційні алгоритми;

ігрові підходи;

використання методів ШІ.

Кожна з цих груп має власні переваги й обмеження. Це визначає їхню ефективність у конкретних сценаріях управління хмарними ресурсами.

Так, зокрема, дослідження Han J. та співавторів [84] присвячено розгляду задачі розміщення ВМ у ХМС крізь призму управління ризиками. Автори описали спосіб розміщення ВМ, який безпосередньо вплинув на рівень уразливості всієї ХМС. Як критерій оптимізації у роботі використано мережеві ризики, які спричинені недоліками гіпервізора. Для розв’язання запропоновано багатокритеріальну модель оптимізації. Рішення знайдено завдяки застосуванню модифікованого еволюційного алгоритму багатокритеріальної оптимізації (або multi-objective GA - SMOOP). Алгоритм SMOOP забезпечив отримання множини Парето-оптимальних альтернатив для розміщення ВМ у ХМС. Автори наголосили, що побудована Парето-фронта дозволила системному адміністратору або планувальнику обрати компромісні конфігурації залежно від пріоритетів. Проте автори відмітили низку обмежень моделі. Зокрема, потрібно попередньо формалізувати вразливості для ВМ у вигляді кількісних показників. А це зумовило потребу ретельно налаштовувати параметри SMOOP.

Так само еволюційний алгоритм дослідники використали у роботі [85]. D. Saxena та інші співавтори представили фреймворк SM-VMP. Фреймворк спрямований на розв’язання проблеми багатокритеріального розміщення ВМ у

хмарних дата-центрах з урахуванням вимог безпеки. Розроблена у [85] модель поєднала мінімізацію числа активних серверів, скорочення мережевих витрат і зниження ризиків. Зокрема, у статті як критерій оптимізації розглянуто ризики пов'язаних із спільним розміщенням ВМ та уразливостями гіпервізора. В роботі використано гібридний еволюційний алгоритм, який поєднував механізми різні варіації генетичних алгоритмів (ГА). А саме алгоритм оптимізації китів (Whale Optimization Algorithm, WOA) та звичайний ГА із застосуванням процедури недомінованого сортування. Це дозволило сформувати Парето-фронт рішень більшої ширини та кращої різнорідності у порівнянні з традиційними методами оптимізації. Експериментальні результати, наведені у статті зафіксували покращення ресурсної ефективності дата центрів при одночасному зниженні ризиків. Проте, залишилася проблема стабільності отриманих рішень у режимах онлайнових змін роботи хмарних дата центрів [86].

Варто, звернути увагу й на роботу J. Chen та співавторів [87]. Дослідники у статті увага зосередилися проблемі розподілу ресурсів у ХМС за умов появи екстрених запитів. Це ситуація коли від системи вимагається забезпечення пріоритетності та мінімального часу відгуку. Автори обґрунтували необхідність відмови від жорстко визначених однокритеріальних моделей на користь багатокритеріального підходу. Математична постановка включала кілька цільових функцій. Зокрема в моделі присутні задача мінімізації кількості активних фізичних серверів і мінімізація відхилень від очікуваного рівня продуктивності. Для пошуку рішень застосовано алгоритми NSGA-II. Однак, у роботі прямо не розглядалися безпекові метрики.

У публікації Lu Cao [88] разом з колегами запропонували стратегію захисту ХМС від атак типу «спільне розміщення». Як інструментарій розглянуто процес оптимізації розміщення ВМ. Дослідники виходили із того, що проблема має три взаємопов'язані виміри. А саме зменшення ризику спільного розміщення ВМ, зниження енергоспоживання та підтримання балансованого завантаження серверів. Модель передбачала розбиття процесу алокації на часові вікна. В межах кожного вікна виконувалася кластеризація запитів. Далі застосовувався алгоритм колонії

мурах. Однак автори зафіксували складнощі із стабільною роботою гібридного алгоритму.

В дослідженнях розподілу ресурсі ХМС та забезпечення їхньої безпеки також активно використовують апарат теорії ігор. Так у статті [89] A. Wilczyński запропонував інтегрувати апарат теорії ігор із завданням безпечного розподілу обчислювальних ресурсів та планування завдань у ХМС. Автор застосував поліматриці розширені ігри Стекельберга як базу моделі. Тут деякі агенти (тобто гравці) виступили як лідери, які задавали стратегії першими. Інші гравці виступали як послідовники. Останні підлаштовують власні дії під вибір лідерів. Гра застосована до задачі картографування завдань на ВМ ХМС з урахуванням рівня довіри як безпекове обмеження. Модель передбачає, що при плануванні пакетів завдань одночасно створюється множина ВМ. А отже приймається фактично оптимізаційне рішення не лише про те, скільки ВМ активувати, але й про те, які саме ВМ задовольняють вимогам безпеки ХМС. В статті критерії безпеки ХМ формалізовані як умови відповідності між завданнями та ВМ із заданим рівнем довіри. Експерименти автор провів у середовищі OpenStack. Підхід цікавий. Він корелюється з припущенням про доцільність застосування теорії ігор в завданнях оптимізації розподілу навантаження в ХМС. Проте модель спиралася на припущення про наявність і явне розмежування ролей лідерів і послідовників. Також заздалегідь на вхід гри подавались дані про правдивість інформації о рівні довіри й поведінці агентів. Але ці припущення складно застосувати в непередбачуваних сценаріях роботи ХМС.

В оглядовій роботі [90] M. Jebalia та інші співавтори представили огляд прикладів та задач де застосовано коаліційні ігри для розв’язання завдань розподілу ресурсів у ХМС. Автори виходили із припущення, що класичні підходи до задачі оптимальної алокації потужностей у хмарі здебільше орієнтовані на централізоване управління. Однак доволі зазвичай користувачі та постачальники ресурсів діють в умовах багатостороннього розподілу ресурсів ХМС. А це, відповідно, створює конкурентне середовище. Автори припустили, що саме коаліційна теорія ігор дозволить формалізувати взаємодії між різними агентами.

Під агентами у статті розумілися користувачі, ВМ, сервіс-провайдери тощо. Агенти формують об’єднання, де спільне використання ресурсів забезпечує виграш усім сторонам. Проте конкретної моделі стаття не містить. Це скоріш концепція в межах оглядового дослідження. Проте, варто відзначити, що саме апарат коаліційних ігор здатен інтегрувати в моделі не лише технічні, економічні параметри, а й метрики безпеки. А також критерії для задачі балансування навантаження й управління енергоспоживанням ХМС.

Апарат теорії ігор також використали автори роботи [91]. Xu X. та Yu H. дослідили задачу ефективного розподілу ресурсів у ХМС. В статті дослідники виходили із припущення, що хмара є багатокористувацьким середовищем, отже ресурси обмежені. А, користувачі мали різні потреби й пріоритети. Для розв’язання задачі авторами запропонована модель, побудована на засадах теорії кооперативної гри. В межах гри розподіл здійснено з урахуванням корисності кожного учасника. Застосовано концепти ядра гри та значення Шеплі. Це у підсумку дало змогу гравцям досягати за певних умов рівноваги між справедливістю та загальною ефективністю використання ресурсів ХМС.

У дослідженні [92] Ait Temghart разом із колегами подали модель прийняття рішень у сфері кібербезпеки ЗО на основі стекельберзької гри. В такій грі постачальник послуг виступав лідером. А потенційний зловмисник у якості послідовника. Модель припускає формалізацію стратегічної взаємодії між захисником і атакувальником. І головне модель враховувала асиметрію інформації та різні рівні впливу на результат гри. Оптимізація стратегій захисту здійснена в статті шляхом визначення рівноваги Стекельберга. У підсумку модель забезпечила гравців можливість здійснювати раціональний вибір механізмів безпеки ХМС за умов обмежених ресурсів і різних сценаріїв атак. Результати роботи підтвердили, що застосування теорії ігор дозволяє знизити ризики компрометації ХМС. Саме варіативність сценаріїв гри та вибір оптимального сценарію дозволили авторам покращити ефективність розподілу ресурсів ХМС на захисні заходи.

У роботі українських дослідників [93] В.П. Малюкова, А.О. Чикрія, В.А. Лахна на підґрунті апарату теорії ігор розглянуто задачу вибору хмарної платформи в

умовах багатокритеріальних обмежень. Автори запропонували модель, побудовану на апараті диференціальних ігор якості з нечіткою інформацією. Модель описувала взаємодію між постачальниками хмарних сервісів та користувачами як гру, у якій сторони прагнули досягти балансу між якістю послуг, вартістю та рівнем захисту. Використання нечіткої логіки дозволило відобразити нечіткість і неповноту вхідної інформації. Такий сценарій є типовим для вибору інфраструктури.

Поряд із поширеними методами оптимізації (симплекс-метод, графічний метод, метод штучного базису, двоїстий симплекс-метод та інші), еволюційними алгоритмами та підходами на основі теорії ігор, у релевантних дослідженнях, присвячених розв'язанню задачі розподілу ресурсів ХМС дослідниками також застосовуються й інші методи та моделі. Зокрема, за останні роки є чимала кількість публікацій у яких автори використовували Марківські та стохастичні моделі опису процесів у ХМС. Подібні моделі дали змогу врахувати часові параметри загроз і ризиків у ХМС. Зокрема, є декілька релевантних наукових досліджень присвячених застосуванню Марківських процесів (рішень) (MDP), частково спостережуваних MDP (POMDP), а також стохастичних моделей відмов і ризиків для підтримки прийняття рішень у з безпеки ХМС. Такі моделі дозволяли авторам не лише дослідити варіативне навантаження та поведінку користувачів. Але також розглянути зміну станів безпеки хмари й відповідних ресурсів системи. Саме тому далі наведемо стислий аналіз низки робіт, у яких Марківські процеси та пов'язані з ними стохастичні моделі використовувалися авторами як аналітичний апарат для розв'язання задач оптимізації та управління ризиками у ХМС.

Зокрема, у роботі [94] Shi R. та інші співавтори досліджували проблему оптимізації витрат при розподілі ресурсів у середовищах віртуалізації мережевих функцій (NFV). Автори підійшли до розв'язання задачі через використання моделей Марковських процесів прийняття рішень (MDP). Це надало змогу представити процес управління ресурсами як послідовність станів із невизначеними переходами та ймовірнісними наслідками. Для підвищення ефективності та адаптивності у статті запропонована інтеграція MDP з методами МН. У підсумку хмарна система навчалася на основі накопиченого досвіду й

підлаштовувалася під оптимальний розподіл ресурсів. Експериментальна частина дослідження довела ефективність поєднання MDP та МН. Така синергія різних методів забезпечила оптимальні рішення розподілу ресурсі хмари. Проте метод має певні вади. Оскільки реалізація методу вимагала чималих обчислювальних ресурсів.

Chen L., Shen H., та Sapra K. у [95] дослідили задачу забезпечення довготривалої збалансованості навантаження у хмарних дата центрах (ХДЦ). У статті автори запропонували підхід, заснований на використанні скінченних марковських процесів прийняття рішень (MDP). Модель дозволяла фізичним серверам проактивно визначати оптимальні дії для переходу у менш завантажений стан. Запропонований авторами алгоритм застосовано для вибору цільових серверів під час міграції ВМ. Ще дало змогу зменшити кількість порушень угод про рівень обслуговування (SLA). Додатково було підвищено ефективність балансування навантаження. Проте метрики безпеки роботи ХДЦ залишилися поза увагою дослідників.

Близька по концептуальній спрямованості до попередньої роботи й стаття [96]. Kazeminajafabadi A. та Imani M. описали новий підхід до захисту мережевої інфраструктури, який ґрунтувався на частково спостережуваних марковських процесах прийняття рішень (POMDP). На відміну від [95], автори запропонували модель, яка інтегрувала ймовірнісний характер поширення атак на ХМС. У межах запропонованої концепції в статті розроблено дві політики захисту. Перша базувалася на мінімізації середньоквадратичної помилки при оцінці стану мережі. Друга використовувала апостеріорний розподіл компрометацій для прийняття рішень. Обидві політики доповнено оптимізованими стратегіями моніторингу. У підсумку обидві політики забезпечили раціональний розподіл ресурсів ХМС.

Дослідження [97] хоча на пряму не мало відношення до теми дослідження, проте також спиралося на використовування моделі Марківського процесу прийняття рішень (MDP). Автори Malik S. U. та інші у статті подали модель із застосуванням MDP, яка дозволила отримати кількісні оцінки ймовірності компрометації системи під час розвитку атаки. Тобто, потенційно MDP доцільно

розглядати як інструмент аналізу ризиків ХМС та підтримки оптимізаційних рішень.

Слід зауважити, що не дивлячись на переважне використовування дослідниками за останні роки більш просунутих методів розв'язання багатокритеріальних задач, як-от еволюційні алгоритми, методи ШІ (ML, DL), теорії ігор та ін., у останні роки з'явилося декілька робіт, які формулюють задачі безпеко-орієнтованого розподілу ресурсів у хмарі як цілочислові задачі. Автори робіт [98–101] вирішують намагалися розв'язати ці багатокритеріальні задачі застосовуючи стандартних оптимізаційні методи. Ці методи зазвичай реалізовані всередині комерційних застосунків для оптимізація, як-от CPLEX, Gurobi, GLPK тощо [98]. Хоча їх більшої мірою застосовують для одержання точних рішень еталонів для малих і середніх оптимізаційних задач.

Так у [99] Mangalagowri R. та Venkataraman R. дослідили проблему захисту ВМ від шкідливих атак (Cross-VM та VM escape) у ХМС. Для розв'язання задачі автори запропонували рандомізований фреймворк на основі змішаного цілочисельного лінійного програмування (MILP). Основна ідея полягала в пошуку оптимальному розподілі ресурсів безпеки з використанням теоретико-ігрового підходу, а саме рівноваги Штакельберга. В статті досліджено модель взаємодії «захисник-зловмисник». А головна мета - превентивно обрати найкращу стратегію захисту. Для цього потрібно передбачити раціональну реакцію зловмисника. Математичний апарат MILP використовувався для опису задачі та знаходження оптимального вектора покриття безпеки.

У [100] автори подали систематизований огляд прикладів застосування лінійного програмування (LP), цілочисельного лінійного програмування (ILP) та змішаного цілочисельного лінійного програмування (MILP) у задачах оптимізації розподілу ресурсів у мережах п'ятого та наступних поколінь. Автори проаналізували понад сотню публікацій, класифікуючи їх за архітектурами мереж, типами ресурсів та специфікою постановок оптимізаційних задач. Зокрема розглянувши цільові функції та обмеження. В статті акцент зроблено на NP-складності задач ILP та MILP і методам їх розв'язання, які групувалися за

методами. Окрім методів ILP та MILP, у статті розглянуто нові тенденції розв'язання багатокритеріальних оптимізаційних задач. Зокрема, проаналізовано доцільність інтеграції в розв'язок таких задач методів ШІ, МН тощо.

В [101] автори як і роботі [95] розглянули проблему оптимального розміщення ВМ у ХДЦ із позиції багатокритеріальної оптимізації. Але на відміну від [95] у цій статті автори запропонували використати комбінований підхід, який поєднав точне розв'язання на основі багатокритеріального цілочисельного лінійного програмування (MOILP) та наближену евристичну стратегію, реалізовану через алгоритм табу-пошуку. Використання MOILP дозволило отримати оптимальні рішення. Проте, через NP-складність задача швидко стає завеликою для обчислення. Тому табу-пошук розглядався авторами як практична альтернатива. Саме табу-пошук, за задумом авторі здатен забезпечити якісний компроміс між обчислювальною ефективністю й точність оптимізації. Головна перевага роботи це одночасна оптимізація трьох метрик продуктивності. А саме у статті розглянуто у якості метрик – кількість ВМ, рівень нераціонального використання ресурсів та енергоспоживання ХТЦ. Результати експериментів, які подано у статті довели, що табу-пошук забезпечив підвищення ефективності роботи ХТЦ майже 32%. Однак, метрики безпеки автори не дослідили.

Проведений в першому розділі дисертації аналіз наукових джерел показав, що для оптимізації розподілу обчислювальних ресурсів ХМС з урахуванням вимог безпеки науковці застосовують широкий арсенал математичних методів і моделей. Кожен метод має свої сильні сторони та обмеження. З метою систематизації отриманих результатів в табл. 1.10 узагальнено основні характеристики різних методів та моделей.

Узагальнення проведеного аналізу попередніх досліджень, засвідчило, що релевантні підходи до оптимізації розподілу ресурсів у ХМС наразі базуються на різноманітті вибору методів. Також автори застосовують різні рівні абстракцій на яких реалізоване управління ресурсами ХМС. Використання класичних методів математичного програмування хоча і дає змогу формалізувати постановку задач дослідження, проте обмежено масштабованістю критеріїв.

Таблиця 1.10

Систематизація огляду й аналізу методів та моделей, які використовувалися різними авторами для розв’язання завдання оптимізації розподілу обчислювальних ресурсів ХМС для підвищення безпеки (складено автором на підставі аналізу робіт першого розділу дисертації)

№	Автори, джерело	Метод (Модель)	Основні переваги	Основні недоліки
1	2	3	4	5
1	Saidi K. et al. [4]	Класичні методи - планування/черги, балансування тощо	Простота. Відомі аналітичні властивості для QoS-метрик. Підходять малого простору критеріїв	Не враховують складні безпекові метрики. Складно масштабувати
2	Senthilkumar G. et al. [5, 98–100]	Лінійне та цілочислове програмування (LP / ILP / MILP). Точні методи – CPLEX, Gurobi, GLPK	Простий опис політик безпеки як жорстких обмежень. Отримання оптимальних рішень для малих і середніх наборів критеріїв	Погано масштабуються. Обчислювальна вартість. Складно врахувати стохастичні ризики. Потребують розширень P-MILP або додатково стохастичного програмування
3	Rabaaoui S., Hachicha H., Zagrouba E. Та інші [6, 98, 99]	Симплекс метод, двоїстий симплекс	Ефективно вирішують велику частину детермінованих лінійних задач. Підходять як внутрішній механізм застосунків для оптимізації. Надійний математичний еталон	Методи непридатні для масштабних дискретних задач. Не працюють безпосередньо з нелінійними чи стохастичними метриками безпеки ХМС
4	Jhawar R., Piuri V., Samarati P. [74], Petrovska I.Y. [20–25]	Математична формалізація багатокритеріальних задач	Дозволяє задати одночасно економічні, продуктивні та безпекові обмеження у формальному вигляді. Можливість використати отримання Парето еталонів	Складно інтерпретувати великі множини Парето-рішень. Вимагають апроксимацій для масштабування

Продовження таблиці 1.10

1	2	3	4	5
5	Han J. et al. [84]; Saxena D. et al. [60]	Еволюційні алгоритми (зокрема NSGA-II, MOEA/D, SPEA2, а такод гібриди - GA+WOA, ACO тощо) для багатокритеріальної оптимізації	Гнучкість у розв'язуванні багатокритеріальних задач. Отримання широкого фронту Парето. Добре працюють з нелінійними моделями цільових функцій	Обчислювальна складність. Нестабільність і потреба в налаштуванні. Складно застосувати при зміні цільових функцій без гібридизації
6	Gill S. S., Buyya R. [69]	Архітектура SECURE та інші фреймворки	Інтегрують моніторинг і адаптивне реагування (включно з міграціями ВМ при підозрі на загрозу). Підходять для автоматичної політики моніторингу безпеки ХМС	Зазвичай концептуальні. Не мають чіткої математичної оптимізації. Обмежений набір сценаріїв. Потребують багато даних
7	Теорія ігор - Xu X., Yu, H. [91]; Kakkad, V., Shah, H., Patel, R. [96]; Li, B., Chen, Y., Huang [102]; та інші	Апарат теорії ігор, коаліційні, некооперативні, кооперативні ігри, білінійні та інші ігри	Формалізують стратегічну взаємодію «захисник - зловмисник» й економічні стимули (ціноутворення, SLA). Добре відображають проблему обмежених захисних ресурсів і протидії, зокрема, DDoS/EDoS-типам атак	Ускладнене отримання результатів через неповну інформацію. Складно знайти рівновагу Неша у великих системах. Складна калібровка моделей через брак даних
8	Chen, L., Shen, H., & Sapra, K. [95], Malik S.U. et al. [97]	Марковські моделі (MDP та POMDP) для моделювання еволюції ризику та вибору політик реагування ХМС	Придатні кількісно моделювати ймовірність компрометації ХМС. 2. Дають можливість аналізу ризиків для ХМС	Висока розмірність простору станів. Складність масштабування. Потребують точних ймовірнісних оцінок
9	Jarkas O. et al. [79, 80]	CaaS-специфічні методи. Методи оркестрації	Дозволяють врахувати вразливості контейнерів. Легко включити в обмеження оптимізації	Математично неформалізовані. Важко кількісно інтегрувати як критерій в оптимізатор

Продовження таблиці 1.10

1	2	3	4	5
10	Kumar B.S.A., Sah B. [63]	Гібридні підходи на основі MOEA, RL та EA	Комбінація дає можливість отримати гнучкі масштабовані рішення. EA приносить багатокритеріальність. Гнучкість у врахуванні метрик безпеки ХМС	Складність реалізації та валідації. Велика кількість гіперпараметрів. Ускладнена сертифікація
11	Feng D. et al. [78]	Аппаратно-орієнтовані моделі (TEE) як складова RM (Resource Management, тобто управління обчислювальними ресурсами у ХМС)	Гарантії конфіденційності у ХМС. Зменшують ризик компрометації даних під час обробки	Складно здійснювати масштабну інтеграцію TEE у стратегії розміщення. Потрібно враховувати доступність
12	Banse C., Kunz I., Schneider A., Weiss K. [58]	Графові моделі для виявлення зв'язків вразливостей у ХМС	Дозволяють простежити ланцюги впливу конфігурацій ХМС. Корисні для формування оптимальних конфігурацій ХМС. Підходять для оцінювання ризиків у ХМС	Погано працюють як самостійний оптимізатор. Містять велику кількість детальної інформації, що потребує агрегування для оптимізації
13	Alzoubi Y.I., Mishra A. & Topcu A.E. [55]; Gupta K., Saxena D., Gupta R., Kumar J., & Singh A. K. [56]	Федеративне навчання (FL), а також МН та ГН (ML та DL) як компонент RM та IDS	Дають можливість виявляти аномалії (атаки) й пропонувати зміну політик безпеки ХМС. FL дозволяє зберегти приватність при спільному навчанні моделей ML та DL	Потреба в якісних мітках і даних. Нетривіальна інтеграція результатів ML у формальну оптимізацію й обґрунтування рішень
14	Волк М.О. та ін. [18]	Декомпозиційні й розподілені алгоритми розподілу ресурсів (RM)	Хороша масштабованість для великих розподілених хмар. Можуть локально застосовувати політики безпеки	Складно узгодити глобальні політики безпеки та гарантії між локальними контролерами. Координаційні витрати впливають на загальну ефективність методів
15	Петровська І. Ю. та інші [20–25]	Методи багатокритеріального аналізу. Методи AHP, MCDM для пріоритезації вимог включно з метриками безпеки	Дає зрозумілу експертну інтерпретацію пріоритетів. Легко пояснювати замовнику. Добре підходить для вимірювання метрик ХМС на етапі опису політик безпеки та управління ресурсами	Складно масштабувати до задач розміщення вузлів ХМС. Суб'єктивність ваг. Експертні оцінки ускладнюють реплікацію методу

Еволюційні та багатокритеріальні алгоритми, зокрема NSGA-II та MOEA/D, потенційно дозволять отримати множини компромісних рішень. Тим самим вони забезпечують більшу гнучкість дослідження. Водночас їхнім недоліком залишається відсутність урахування специфіки інформаційних загроз ХМС. А це суттєво впливає на безпеку розподілених обчислень. Окремий напрямок пов'язаний із застосуванням ігрових моделей в управлінні ресурсами ХМС. Ці моделі формалізують взаємодію між атакуючими та захисниками в ХМС. Проте наявні роботи переважно зосереджені на спрощених сценаріях. Вони рідка інтегровані у цілісні моделі управління ресурсами ХМС. Тобто, огляд наведений у п.п. 1.2 та 1.3 підтвердив існування наукової задачі, яка полягає у синтезі комплексної моделі, здатної одночасно врахувати показники продуктивності, економічної ефективності та рівня безпеки ХМС, включно з оцінкою ризиків і можливістю сформувати кооперативні стратегії захисту. Усунення цієї прогалини потребує розробки нової концепції, яка б поєднувала методи багатокритеріальної оптимізації та підходи теорії ігор у єдиній гібридній моделі. Саме таку концепцію подано у наступному параграфі, де представлено концептуальну схему розв'язання задачі дисертації.

1.4. Концептуальна схема розв'язання задачі та постановка наукового завдання дослідження

Формування наукового підходу до розв'язання задачі оптимізації розподілу ресурсів у ХМС вимагає узгодженого поєднання результатів аналізу попередніх досліджень та визначення власної траєкторії дослідження. Для цього доцільно виокремити концептуальну схему, див. рис. 1.4, яка відображає логіку переходу від вихідних передумов і виявлених обмежень чинних методів до синтезу авторської моделі.

Схема подана в табл. 1.11 та на рис. 1.4 слугує інтеграційним елементом між оглядовою (п.п. 1.1–1.3) та теоретичною частинами дисертації (розділ 2),

демонструючи послідовність етапів дослідження. Схема на рис. 1.4. тим самим забезпечує системність і відтворюваність результатів дисертації у разі потреби.

Концепція ґрунтується на тому, що проблема безпечного управління ресурсами у хмарному середовищі не слід розглядатися ізольовано в межах лише продуктивності чи економічності, оскільки багатокористувацька природа хмари і постійна зміна кіберзагроз перетворюють безпеку на невід'ємний критерій оптимізації.

Саме тому в запропонованій схемі на рис. 1.4 базисом виступає гібридизація двох класів методів:

багатокритеріальних еволюційних алгоритмів, що дозволять дослідити компромісні розв'язки між критеріями;

ігрових моделей безпеки, здатні описати поведінку атакуючої і захисної сторін.

Об'єднання цих складових на нашу думку утворить теоретичне підґрунтя авторського підходу, який діли висвітлено у наступному розділі роботи.

Подана в табл. 1.11 та на рис. 1.4 схема не лише описує взаємозв'язок між елементами проблемного поля. Але й відображає етапність переходу від постановки задачі до її формалізації. Та подальшої перевірки на практичних сценаріях (див. розділ 3 дисертації). Графічну інтерпретацію концептуальної схеми розв'язання задачі дослідження подано на рис. 1.4.

У межах концептуальної схеми розв'язання задачі дисертації, див. рис. 1.4 першочергову увагу слід приділити обґрунтуванню вибору цільових показників, які формалізуються в наступному розділі дисертації фактичні критерії багатокритеріальної оптимізації. У запропонованому підході, на основі аналізу робіт [103–111] в якості ключових критеріїв виділимо чотири взаємопов'язані виміри:

сумарну вартість використання ресурсів;

надійність (безпека) ХМС;

продуктивність;

коаліційну вигоду.

Таблиця 1.11

Концептуальна схема розв’язання задачі дослідження (складено автором)

Етап	Зміст етапу	Результат та його роль у дослідженні.
Вихідні передумови	Аналіз архітектурних моделей хмарних систем (ХМС) - IaaS, PaaS, SaaS, FaaS, CaaS, XaaS, EaaS тощо, багатокористувацької природи хмар та характерних кіберзагроз	Визначено проблемне поле дослідження. Встановлено необхідність поєднання критеріїв продуктивності, вартості та безпеки в єдиній постановці задачі
Аналіз попередніх досліджень (п.п. 1.1–1.3)	Розгляд та аналіз чинних методів і моделей оптимізації розподілу ресурсів ХМС. А саме - класичні алгоритми, еволюційні методи (NSGA-II, NSGA-III MOEA/D), ігрові підходи, методи МН. Методи ШІ тощо	Виявлено наукову прогалину. А саме відсутня комплексна модель, що інтегрує оптимізацію продуктивності, економічності та безпеки з урахуванням динамічних ризиків
Постановка наукової проблеми	Узагальнення виявлених недоліків наявних підходів та формулювання вимог до нової моделі	Сформульовано завдання розробки гібридного підходу, що поєднує багатокритеріальну оптимізацію і теоретико-ігрову оцінку безпеки
Розробка концептуальної моделі – п. 1.4	Обґрунтування ідеї інтеграції NSGA-II з ігровими моделями безпеки, визначення критеріїв: ризик, продуктивність, вартість, коаліційна вигода	Встановлено основу для математичної формалізації задачі та побудови авторського алгоритмічного підходу
Математична формалізація (Розділ 2)	Побудова системи рівнянь, що описують динаміку ризику, адаптивні стратегії захисника та кооперативні взаємодії, інтегровані в багатокритеріальну оптимізацію	Створено формалізовану гібридну модель оптимізації розподілу ресурсів
Обчислювальний експеримент (Розділ 3)	Реалізація алгоритму, симуляція сценаріїв розподілу задач у хмарній інфраструктурі, візуалізація Парето-фронтів, оцінка метрик якості рішень – HV, IGD, Spacing	Проведення перевірки ефективності моделі. Підтвердження її здатність забезпечувати компроміс між безпекою, продуктивністю та вартістю ХМС

Ця множина критеріїв відображає компромісну природу предметної області. А саме ситуації де економічна доцільність, якість обслуговування та стійкість до загроз одночасно визначать практичну цінність прийнятих рішень щодо розміщення задач у ХМС.

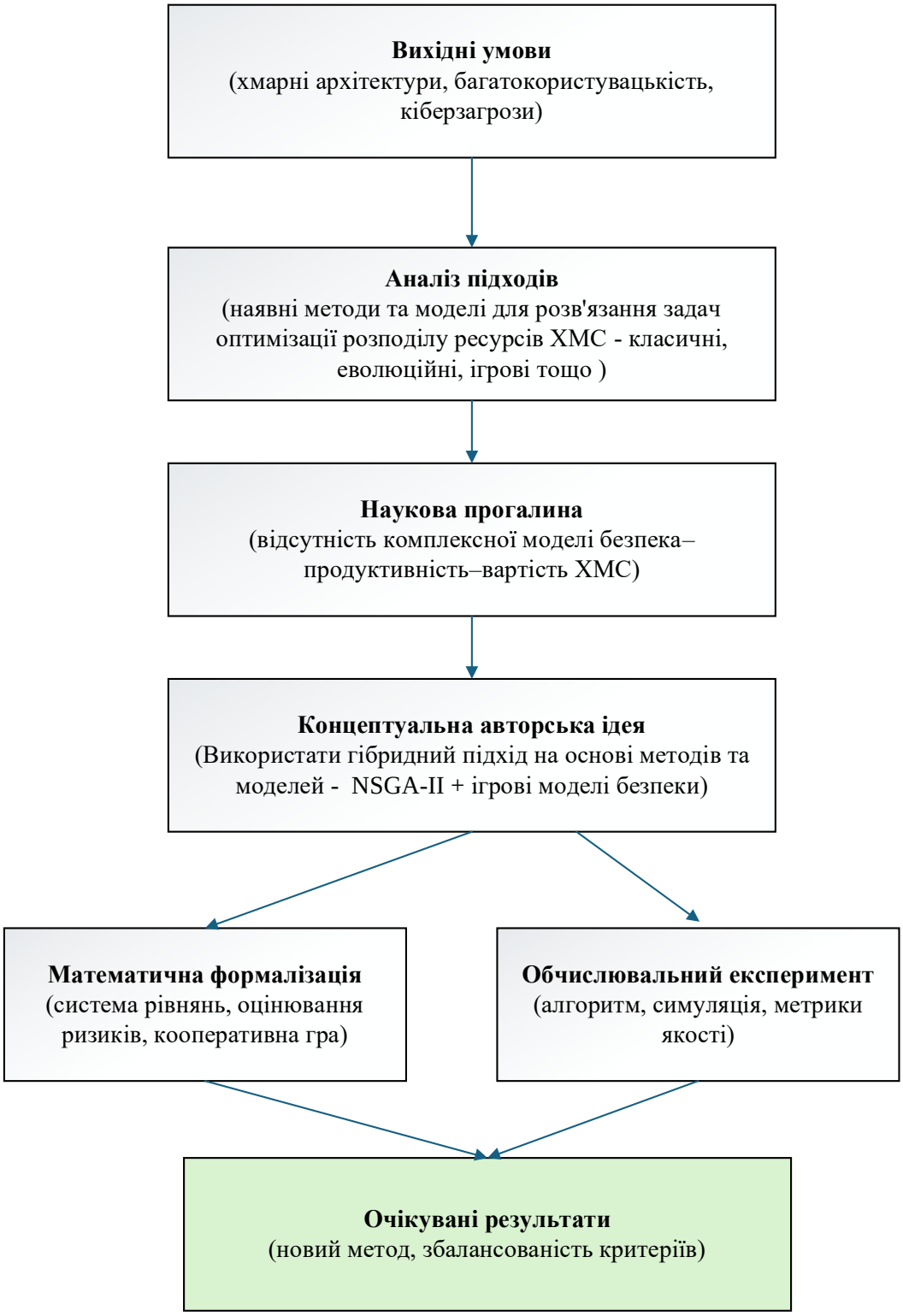


Рис. 1.3. Концептуальна схема розв’язання задачі дослідження (складено автором)

Обраний критерій вартості відображає пряму економічну складову використання хмарних ресурсів. А саме – тарифи (затрати) на використання CPU/GPU, мережевого трафіку, сховищ та додаткових сервісів. У багатьох практичних сценаріях застосування ХМС саме обмеження бюджету є

визначальним фактором при прийнятті рішень. Тому економічна оптимізація як окремий цільовий показник забезпечує репрезентацію цього реального обмеження у формалізації задачі багатокритеріальної оптимізації ХМС. Це не суперечить результатам висновків до яких прийшли автори робіт [103–111].

Надійність та безпека представлена як сумарний ризик розміщення задач [112]. Цей критерій включає в себе імовірнісні оцінки атак, потенційні збитки та ступінь впливу на важливі для бізнес-процесів сервіси ХМС. Отже її введення в задачу багатокритеріальної оптимізації випливає з реальної природи експлуатації ХМС. Це багаторазово доведено у багатьох публікація, які розглянуто в першому розділі роботи. Зокрема аналогічні висновки зроблено й у роботах [20–25, 103–111]. Зазначимо, що неможливо відокремити питання продуктивності від ризикового профілю хмарної інфраструктури. Продуктивність відображає часові характеристики опрацювання запитів у хмарі, потенційні затримки й пропускну здатність. Тобто саме ті показники, які визначають у підсумку рівень сервісу та задоволення вимог користувача (SLA).

Окреме місце у наборі займає коаліційна вигода. Це критерій у системі завдання багатокритеріальної оптимізації описує перевагу від спільних дій захисних агентів. Отже він зафіксує факт, що завдання захисту ХМС не є зважанням яке здатне розв’язати самотужки окремі компонент системи захисту чи окремий метод захисту ХМС. Отже, її включення є доцільним для підкислення та відображення властивостей кооперативних стратегій. Тобто ситуації коли спільні заходи компонентів кібербезпеки ХМС зменшують у підсумку ризики більше, ніж сума індивідуальних дій.

Відмітимо, що згідно результатів аналіз робіт [20–25, 103–111] методика вибору критеріїв базувалася на принципі репрезентативності та незалежності критеріїв. Кожний критерій повинен вносити суттєву додану інформацію про якість рішення. А також при цьому не дублювати інший у сенсі кореляції вимірюваних властивостей.

Для практичної реалізації цього принципу застосуємо таксономія вимірів [109–111]. Таксономія має три рівні, див. табл. 1.12:

технічний – продуктивність, латентність, використання CPU/GPU;

економічний – вартість, енергоефективність;

ризик-орієнтований – ймовірність компрометації, очікувані збитки тощо.

Критерії, обрані для багатокритеріальної постановки, отримані як уніфікована проекція цих вимірів. А саме по одному представнику з кожного базового класу. Плюс додатковий показник, що відображає користь кооперації (тобто коаліційну вигоду), яка не належить винятково ні до технічних, ні до економічних метрик.

Таблиця 1.12

Критерії оптимізації як результат таксономічного відбору (складено автором на підставі робіт [20–25, 103–111])

Таксономічна категорія	Приклади можливих показників	Обґрунтування відбору	Обраний критерій (F)
Економічний вимір.	Вартість використання CPU/GPU. Тарифи на зберігання даних. Витрати на мережевий трафік. Енерговитрати.	Вартість є базовим обмеженням для більшості підприємств та визначає межі оптимізації ХМС.	Сумарна вартість використання ресурсів (економічність).
Ризик-орієнтований вимір.	Ймовірність успішної атаки. Очікуваний збиток. Ризик компрометації даних у ХМС.	Безпека є приоритетною через багатокористувацький характер хмар та постійні кіберзагрози.	Сумарний ризик розміщення задач (безпека).
Технічний вимір	Час обробки задач; латентність; пропускна здатність каналів	Продуктивність визначає якість сервісу (SLA) та рівень задоволення користувача.	Сумарний час обробки задач (продуктивність).
Кооперативний вимір	Ефективність кооперації захисників; синергетичний ефект від спільних дій	Коаліційні вигоди відображають додатковий захисний ефект, що перевищує суму індивідуальних дій.	Коаліційна вигода.

Таксономічний підхід дозволив аргументовано відкинути надмірні та редундантні показники. А також водночас зберегти повноту представлення предметної області.

Порівняння з альтернативними підходами до вибору критеріїв – це необхідна складовою обґрунтування ХМС. У багатьох працях, зокрема [20, 103, 106, 108] , застосовували підхід зведення до єдиної цільової функції (зважена сума). Або автори сумісно використовували жорстко визначені пріоритети (тобто так званий лексикографічний підхід). Проте, такі методи погано працюють у ситуаціях, де існує невизначеність щодо ризиків. Або у ситуаціях коли умови ХМС швидко змінюються. Отже зважена сума вимагає визначення стабільних ваг. А лексикографія відсікає потенційно прийнятні компроміси. Саме тому альтернативі до них, багатокритеріальні еволюційні алгоритми, як-от NSGA-II, NSGA-III, потенційно дозволять зберегти множину компромісних розв'язків. Тоді отримані результати дадуть змогу аналітикам компаній чи підприємств, які використовують хмарні сервіси обрати найкращі варіанти для остаточного впровадження в бізнес-процеси, виходячи з домінуючих критеріїв у конкретному сценарії.

Отже, саме тому для предметної задачі багатокритеріальний підхід є методологічно обґрунтованим вибором. Саме він дозволить відділити стадію пошуку від стадії прийняття рішення. А потім забезпечити гнучкість у ситуаціях із конфліктними цілями - критеріями.

Щодо вибору між NSGA-II і NSGA-III доцільно зазначити. Саме NSGA-II залишається ефективним інструментом для задач з невеликою чи помірною кількістю критеріїв (як у нашому випадку — чотири цілі). Так само як альтернативу можна далі розглядати доцільність застосування NSGA-III, який довів свою перевагу в умовах багатовимірних цільових просторів.

Крім методики таксономічного відбору, додатково доцільно застосовувати ряд допоміжних процедур для адаптивного керування критеріями у процесі експерименту (див. Розділ 3). Саме в заключному розділі роботи ми додатково застосуємо нормалізацію метрик за їх масштабами. А також виконаємо аналіз чутливості та кореляційний аналіз й верифікацію значущості критеріїв за допомогою тестових сценаріїв. Нормалізацію яку більш детально розглянуто у заключному розділі дисертації гарантує, що показники з різними фізичними одиницями мають зіставні впливи на процес відбору. А аналіз чутливості дозволить

виявити, наскільки результати фронту Парето зміняться при варіюванні ваг критеріїв (див. табл. 1.12) або домінуючих параметрів. Своєю чергою кореляційний аналіз допоможе уникнути дублювання інформації в наборі критеріїв. Завершуючи, слід підкреслити, що концептуальна схема розв'язання задач дисертації, див. табл. 1.11 та рис. 1.4, випливала із виявлених обмежень наявних підходів. Вона надає чіткий напрям для подальшої формалізації нашого методу та відповідних моделей, які детально розглянуто у наступних розділах дисертації.

Висновки до розділу 1

В результаті досліджень в першому розділі дисертації зроблено наступні висновки та отримано такі результати.

1. Проаналізовано теоретичні та прикладні засади функціонування хмарних систем (ХМС). Це дало змогу визначити їх як складний багатокомпонентний об'єкт дослідження, у якому поєдналися питання архітектури, управління обчислювальними ресурсами та забезпечення кіберзахисту. Визначено, що специфіка хмарних технологій полягає у багатокористувацькому середовищі та варіативному характері навантаження, що ускладнює завдання раціонального розподілу ресурсів й вираховування метрик забезпечення захисту ХМС.

2. Проаналізовано сучасні архітектурні моделі ХМС - IaaS, PaaS, SaaS, FaaS, CaaS, XaaS тощо. Встановлено, що відмінності між ними полягають у рівні абстракції доступу до ресурсів та способах їхнього управління, що безпосередньо впливає на проблематику безпеки. Зокрема, моделі, орієнтовані на інфраструктуру, вимагають суворого контролю над розподілом апаратних ресурсів, тоді як більш високорівневі моделі концентруються на захисті даних і сервісів. Це дозволило обґрунтувати вибір ХМС як першочергового об'єкта для задачі багатокритеріальної оптимізації ресурсів із врахуванням метрик безпеки.

3. Проаналізовано наявні методи та моделі управління обчислювальними ресурсами. Показано, на підставі аналізу попередніх досліджень вітчизняних на

закордонних авторів, що класичні методи переважно базуються на алгоритмах планування та балансування навантаження. Однак ці методи та відповідні моделі зазвичай ігнорують фактор ризиків безпеці ХМС та складну взаємодію між атакуючим і захисником. Доведено, що сучасні релевантні методи багатокритеріальної оптимізації, зокрема еволюційні алгоритми дають змогу враховувати конфліктність цілей і шукати компромісні рішення. Отже, ці методи потенційно здатні забезпечити перехід від статичного управління ресурсами ХМС до більш реалістичного, який відповідає умовам експлуатації.

4. Обґрунтовано доцільність інтеграції безпекових параметрів у задачу оптимізації ресурсів ХМС. Доведено, що розподіл ресурсів без урахування ризиків призведе до підвищеної вразливості ХМС. Така ситуація трапиться навіть якщо показники продуктивності та вартості є оптимальними. Отже, зроблено висновок, що саме поєднання критеріїв продуктивності, економічності та безпеки становить новий якісний рівень розв’язання задачі багатокритеріальної оптимізації розподілу обчислювальних ресурсів для підвищення безпеки та продуктивності хмарних систем.

5. Проаналізовано понад 100 публікацій за останні 15 років, присвячених проблематиці розподілу обчислювальних ресурсів. Досліджено наукові джерела, де оптимізація ресурсів розглядалася авторами крізь призму SLA, QoS та ризик-менеджменту у ХМС. Зроблено узагальнений висновок, що більшість наявних моделей та методів або фокусувалася на вузькому аспекті, як-от часі опрацювання запитів у ХМС чи вартості. Або ці роботи залишали поза увагою питання безпеки ХМС. Доведена необхідність синтезу комплексної моделі, де безпека виступає як рівноправний критерій поряд із традиційними параметрами SLA чи QoS.

6. Доведено доцільність використання багатокритеріальних еволюційних алгоритмів, зокрема NSGA-II та NSGA-III, у задачі розподілу обчислювальних ресурсів для підвищення безпеки та продуктивності ХМС, оскільки саме вони потенційно дозволять зберегти множину Парето-оптимальних розв’язків і надати можливість вибору рішення залежно від поточних пріоритетів.

7. Запропоновано концептуальну схему подальшого дослідження, яка поєднала виявлені під час аналізу попередніх досліджень наукові прогалини з напрямами їх розв’язання. Представлена у завершальному параграфі поточного розділу дисертації концептуальна схема відобразила логічний зв’язок між аналізом архітектур і методів розв’язання задачі, формуванням системи критеріїв багатокритеріальної оптимізації та синтезом гібридної моделі на основі поєднання багатокритеріальної оптимізації та ігрових моделей безпеки. У підсумку результати досліджень, наведені у першому розділі дисертації створили підґрунтя для наступних теоретичного та експериментального розділів, де здійснено формалізацію запропонованого підходу та його експериментальна перевірка засобами імітаційного моделювання.

Отже, резюмуючи висновки першого розділу дисертації, зазначимо, що у розділі систематизовано знання про ХМС, визначено наукову проблему, показано її невирішеність у межах наявних методів та моделей. Обґрунтовано методологічний базис для подальших досліджень. Отримані у першому розділі результати дозволили зробити висновок, що розв’язання задачі оптимізації розподілу обчислювальних ресурсів для підвищення безпеки ХМС потребує нового методу, здатного враховувати багатокритеріальний характер проблеми та варіативність кіберзагроз для різних архітектур ХМС.

РОЗДІЛ 2
МЕТОД КООПЕРАТИВНО-ЕВОЛЮЦІЙНОГО РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ У ХМАРНОМУ СЕРЕДОВИЩІ З УРАХУВАННЯМ РИЗИКУ

Розділ присвячено розробці математичної моделі розподілу обчислювальних задач у хмарній інфраструктурі з урахуванням основних параметрів, що визначають результативність та стійкість роботи хмарних обчислювальних систем (далі ХОС) [113]. У запропонованій моделі розглядаємо формалізацію процесу прийняття рішень щодо призначення задач на доступні обчислювальні вузли (ОВ) з огляду на три взаємопов'язані параметри – забезпечення високої продуктивності, оптимізація вартості використання ресурсів, мінімізація ризиків, пов'язаних з кіберзагрозами для хмарного середовища (ХС). Запропонована модель є гібридною, адже поєднує засоби багатокритеріальної оптимізації з положеннями теорії ігор. Це дозволяє, комплексно враховувати як внутрішні параметри ХОС, так і потенційні загрози з боку зловмисника (хакера чи хакерів). У межах цього розділу поетапно подано формалізацію ігрової взаємодії, математичне визначення функцій втрат, ризиків та критеріїв оптимізації, а так саме відповідні обмеження, що описують припустимі конфігурації ХОС. Запропонована модель становить основу для подальшої реалізації інтелектуальної системи для завдання управління хмарними обчислювальними ресурсами з метою підвищення їх стійкості до загроз та оптимального використання.

Спершу розглядаємо задачу синтезу ігрової моделі оцінки ризику, яка опишіть стратегічну поведінку зловмисника, орієнтовану на максимізацію шкоди, та захисника, що прагне мінімізувати втрати від потенційних атак на ХС. На другому етапі виконано оцінку ризику, яка здобута в наслідок цієї взаємодії. Далі її інтегруємо у багатокритеріальну постановку задачі розподілу задач за допомогою еволюційного алгоритму NSGA-II. У сукупності модель дозволяє отримати множину компромісних рішень, що відображають різні сценарії балансування між критеріями ефективності, вартості й безпеки ХОС.

2.1. Модель розподілу обчислювальних задач у хмарній інфраструктурі з урахуванням продуктивності, вартості та безпеки

Нехай маємо множину обчислювальних вузлів хмарної інфраструктури:

$$N = \{1,2, \dots, n\}, \tag{2.1}$$

$$T = \{T_1, T_2, \dots, T_m\}. \tag{2.2}$$

Ігрова модель моделювання атакуючого та захисника. Розглядаємо двох гравців:

захисник (індекс – D (Defender));

атакуючий (індекс – A (Attacker)).

Визначимо стратегії гравців, див. табл. 2.1.

Стратегія захисника. Розподіл задач $x_{ij} \in \{0,1\}$, де

$$x_{ij} = \begin{cases} 1, \text{якщо задача } T_i \text{ розміщена на вузлі } i, \\ 0, \text{інакше.} \end{cases} \tag{2.3}$$

Стратегія атакуючого. Вибір вузлів для атаки задач $a_i \in \{0,1\}$, де

$$a_i = \begin{cases} 1, \text{якщо атакуючий атакуя вузол } i, \\ 0, \text{інакше.} \end{cases} \tag{2.4}$$

Тоді для забезпечення безпеки ХС актуальним є моделювання стратегічної взаємодії між учасниками, які мають суперечливі цілі. Для формалізації цієї взаємодії використовуємо методику на базі теорії антагоністичних ігор. Це дозволяє врахувати можливу поведінку зловмисника в процесі планування захисних заходів та розподілу обчислювальних ресурсів. Зауважимо, що ігрова модель базувалася на припущенні, що обидва гравці діють раціонально в умовах конфлікту. Ціль атакуючого полягає у максимізації шкоди, завданої ХС, через компрометацію обчислювальних вузлів або порушення функціонування сервісів. Водночас, захисник прагне мінімізувати ймовірні втрати, реалізуючи стратегії захисту та розподілу задач з урахуванням ризиків. У запропонованій моделі розглядаємо скінченну множина стратегій кожного гравця.

Таблиця 2.1

Порівняльний аналіз стратегій атакуючого та захисника у хмарних системах (складено автором на базі аналізу літератури Розділу 1)

Категорія	Стратегія атакуючого	Стратегія захисника	Приклад реальної атаки/захисту
Навантаження системи	Атака DDoS на вузли з піковим навантаженням	Балансування навантаження, використання резервних вузлів	Атака на AWS (2020), яка призвела до збоїв через перевантаження
Вразливості ПЗ	Експлуатація сервісів, які не оновили компоненти безпеки	Автоматизоване оновлення та моніторинг вразливостей	Використання CVE-2021-44228 (Log4j) для несанкціонованого доступу
Конфіденційність даних	Цілеспрямований витік даних із вузлів	Шифрування даних, сегментація мережі	Витік даних із хмарного сховища Microsoft Azure (2019)
Адаптивний захист	Адаптація (пристосування) до заходів безпеки	Маскування ресурсів, зміна IP-адрес вузлів	Уникнення виявлення під час атак на хмарні Kubernetes-кластери

Стратегії атакуючого $S_A = \{a_1, a, \dots, a_m\}$ відповідають різним типам атак, спрямованих на певні вузли або сервіси ХС. Це відповідно до [114, 115]:

a_1 – атака на вузли з інтенсивним навантаженням (DoS, DDoS, EDos);

a_2 – цілеспрямована атака на вузли з цінними даними (APT);

a_3 – експлуатація вразливостей в ПЗ з низьким рівнем оновлення;

тощо.

Стратегії захисника $S_D = \{d_1, d, \dots, d_n\}$ моделюють різні конфігурації розподілу задач та засоби захисту:

d_1 – реплікація задач для зменшення ймовірності повної втрати;

d_2 – мігрування задач ХС у відповідь на загрозу;

d_3 – приоритезація задач залежно від критичності та вартості;

тощо.

Стратегії атакуючого базуються на виборі цілей, які забезпечують максимальний ефект при мінімальних витратах. Найперше, атаки типу DDoS або EDoS спрямовані на вузли з інтенсивним навантаженням. Це призводить до їхньої недоступності та порушення роботи критичних сервісів ХС. Іншим прикладом є атаки на вразливе програмне забезпечення (Exploit-Based Attacks), де зловмисник експлуатує недоліки в системах віртуалізації або керування хмарою для отримання несанкціонованого доступу. Крім того, поширені атаки на конфіденційність даних, які передбачають цілеспрямований витік інформації з критичних вузлів, що опрацьовують чутливі дані, див. табл. 2.1.

Захисник використовує гнучкі засоби для протидії атакам, як-то розподіл задач для уникнення перевантаження окремих вузлів ХС. Іншим ефективним підходом є реплікація критичних сервісів. Це дозволяє зменшити ймовірність повного виведення системи з ладу. В моделі застосовано елементи пріоритезації, які забезпечили обслуговування найважливіших задач на найбільш захищених вузлах.

Функція втрат $L(d_i, a_j)$ є відображенням у простір дійсних чисел ($\mathbb{R}$), яке визначає потенційні втрати захисника в результаті застосування певної пари стратегій [116, 117]:

$$L = S_D \times S_A \longrightarrow \mathbb{R}^+, \tag{2.5}$$

де $L(d_i, a_j) =$ очікувані втрати при захисті d_i та атаці a_j.

Відповідно до [116, 117] ця функція включає компоненти, пов’язані з:

прямими фінансовими втратами (порушення SLA);

втратами конфіденційності чи цілісності даних;

витратами на відновлення після атаки.

Далі, на підставі цього визначення будуємо матрицю гри – табличну форму яка візуалізує представлення значень функції втрат для всіх можливих пар стратегій. Нижче у табл. 2.2 подано умовний приклад такої матриці для $|S_D| = 3, |S_A| = 3$. У кожній комірці табл. 2.2 вказано величину втрат для захисника при поєднанні стратегій.

Таблиця 2.2

Приклад матриці гри (складено автором)

	a_1	a_2	a_3
d_1	100	250	180
d_2	150	120	220
d_3	200	130	100

Тоді функція виграшу атакуючого, подамо так [116, 117]:

$$U_A = \sum_{i=1}^{n} a_i \cdot D_i, \tag{2.6}$$

де D_i – очікувані збитки внаслідок успішної атаки на вузол i.

Функція виграшу атакуючого подано так [116, 117]:

$$U_D = -\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot a_i \cdot D_i. \tag{2.7}$$

Після формалізації стратегічної взаємодії між атакуючим та захисником у межах ігрової моделі наступним логічним етапом є побудова кількісної оцінки ризику для кожного вузла хмарної інфраструктури. Отримані в результаті гри стратегії гравців, а також їхні очікувані виграші, будуть слугувати підґрунтям для розрахунку потенційних втрат. Інакше кажучи ці потенційні втрати пов’язані із компрометацією окремих обчислювальних ресурсів ХС. Подібна оцінка ризику буде відігравати вирішальну роль у подальшій оптимізаційній моделі. Це пов’язано із тим, що слід далі інтегрувати вплив стратегій дій зловмисника у процес прийняття рішень щодо призначення задач. У цьому розрізі завдані дослідження ризик розглядаємо як функцію ймовірності атаки на вузол ХС та очікуваного збитку, який завданий у разі її успішного здійснення, і слугує одним з основних критеріїв багатокритеріальної оптимізації.

Опишемо ймовірність атаки на вузол i так:

$$P_i^{(attack)} = \frac{e^{\lambda \cdot D_i}}{\sum_{k=1}^{n} e^{\lambda \cdot D_k}}. \tag{2.8}$$

де λ – параметр, який визначає агресивність атакуючого; D_i – оцінка втрат у разі успішної атаки на вузол i.

Даний вираз є модифікованою softmax-функцією, яка трансформує вектор потенційних збитків D_i у ймовірнісний розподіл. Застосування експоненційної функції $e^{\lambda \cdot D_i}$ забезпечує невід'ємність значень та підсилює вплив вузлів із високими D_i . А нормування на суму $\sum_{k=1}^{n} e^{\lambda \cdot D_k}$ гарантує, що сума ймовірностей для всіх вузлів дорівнює одиниці, що відповідає аксіомам теорії ймовірностей.

Параметр λ, який ми умовно називаємо параметром агресивності атакуючого, відіграє вагому роль у моделюванні поведінки зловмисника в рамках ігрової частини нашої гібридної моделі. Інтуїтивно цей параметр визначає наскільки активно або ризиковано атакуючий діє у виборі цілей (вузлів хмарної інфраструктури). Зокрема – чи схильний він концентрувати зусилля на найбільш вразливих вузлах ХС. Чи, навпаки, розподіляє свої атаки більш рівномірно.

Зауважимо, що не всі атакуючі поводяться однаково. Деякі діють обережно, намагаючись уникнути виявлення. Інші агресивно, концентруючи атаки на найцінніших або найменш захищених цілях. Параметр λ регулює розподіл ймовірностей вибору цілей атакуючим. При малому значенні λ зловмисник атакує цілі майже випадково (низька агресивність, розосереджені атаки). Та при високому λ зловмисник майже завжди атакує вузли з найбільшим потенційним виграшом (значна агресивність – таргетовані атаки). Наведемо приклад. Припустимо, що у нас є множина вузлів $N = \{1,2, \dots, n\}$, а L_j – потенційний виграш (або збиток для захисника) у разі успішної атаки на вузол ХС. Значить доцільно задати ймовірність того, що атакуючий обирає для атаки вузол j, за допомогою модифікованого softmax-розподілу (розподіл ймовірності при K різних можливих варіантах), див. вираз (2.8) [118]. Якщо $\lambda \longrightarrow 0$, тоді

$$P_i^{(attack)} = \frac{1}{n}. \tag{2.9}$$

Атакуючий розподіляє свої зусилля рівномірно по всіх вузлах. Це неагресивна стратегія, з погляду вибору цілей.

Якщо $\lambda \longrightarrow \infty$ тоді

$$P_i^{attack} \longrightarrow \begin{cases} 1, \text{якщо } L_j = max_k \cdot L_k, \\ 0, \text{інакше.} \end{cases}. \tag{2.10}$$

Це означає, що атакуючий зосереджує зусилля лише на одному найбільш вразливому/вигідному вузлі. Це максимально агресивна стратегія. При помірних значеннях λ, як-от λ=1 або λ=3, атакуючий вибирає вузли з вищим значенням L_j частіше, але не виключає інші вузли повністю. Це збалансована стратегія, яка відповідає реалістичному атакуючому, що ураховує вигоду й ризики. Нехай у нас є три вузли з такими потенційними втратами у разі атаки: $L_1 = 10$; $L_2 = 5$; $L_3 = 1$. Для $\lambda = 0{,}5$ маємо

$$P_1 = \frac{e^{0,5\cdot 10}}{e^{0,5\cdot 10}+e^{0,5\cdot 5}+e^{0,5\cdot 1}} \approx 0{,}9,\ P_2 \approx 0{,}074,\ P_{32} \approx 0{,}01.$$

У нашій гібридній моделі параметр λ дозволяє моделювати різні сценарії поведінки зловмисника. Від хаотичних атак до цілеспрямованих високоризикових операцій. Тоді для одного рівня λ оптимальні стратегії розподілу задач будуть одні, для іншого кардинально інші. У подальшій оптимізації (через NSGA-II) ці оцінки атакуючого напряму впливають на функцію ризику, яка, в свою чергу, є складовою одного з критеріїв багатокритеріальної задачі.

Функція (2.8) відбиває принцип раціонального вибору атакуючого в умовах обмежених ресурсів. Зловмисник оптимізує свої дії, максимізуючи очікуваний виграш $U_A = \sum_{i=1}^{n} a_i \cdot D_i$, де $a_i \in \{0,1\}$ - індикатор атаки. Вираз (2.8) фактично є рішенням цієї задачі раціонального вибору атакуючого при обмежені $\sum_{i=1}^{n} a_i = 1$ (атака на один вузол ХС за раз).

Ризик виконання задачі на вузлі i

$$R_i = P_i^{(attack)} \cdot D_i, \qquad (2.11)$$

де R_i – очікуваний ризик використання вузла i.

Виходячи з побудованої ігрової моделі (2.1)–(2.11), що віддзеркалює стратегічну взаємодію між атакуючим та захисником, а також сформованої на її основі оцінки ризику кожного вузла хмарної інфраструктури (2.11), на наступному етапі доцільним є перехід до формалізації задачі багатокритеріальної оптимізації. Отримані значення ризику (2.11) відображають потенційні загрози, що виникають у результаті дій зловмисника. Вони виступають одним із вирішальних факторів, які слід зважати під час ухвалення рішень щодо розподілу обчислювальних задач у ХС.

Окрім безпекових складових, ефективне функціонування хмарної інфраструктури визначає досягнення показників високої продуктивності ХС, мінімізацію часу опрацювання та оптимізацію вартості використання ресурсів. Формалізація задачі зводиться до багатокритеріального підходу, в якому слід одночасно взяти до уваги суперечливі цілі, найперше забезпечення безпеки, продуктивності та економічної доцільності захисту вузлів ХС. Тоді наступним кроком є побудова частини математичної моделі, яка дозволить знайти компромісні рішення з урахуванням усіх зазначених критеріїв.

Шукаємо такий розподіл задач $x = \{x_{ij}\}$, який мінімізує векторну функцію:

$$\begin{cases} F_1(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot R_i, \\ F_2(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot T_{ij}, \\ F_3(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \cdot C_{ij}, \end{cases} \tag{2.12}$$

де $F_1(x)$ – сумарний ризик розміщення задач (безпека ХС); $F_2(x)$ – сумарний час опрацювання завдань (продуктивність ХС); $F_3(x)$ – сумарна вартість використання ресурсів (економічність ХС); T_{ij} – час виконання задачі T_i на вузлі i; C_{ij} – вартість виконання задачі на вузлі i.

Звернемо увагу, що з огляду на сформульовану багатокритеріальну постановку задачі, яка враховує найважливіші параметри – безпеку, продуктивність та вартість розподілу обчислювальних задач у хмарній інфраструктурі, необхідним є визначення обмежень. Ці обмеження формалізують технічні, функціональні та логічні характеристики досліджуваного ХС. У межах оптимізаційної частини моделі обмеження відіграють значну роль. Внаслідок того, що саме обмеження (2.13) та (2.14) забезпечують коректність та практичну доцільність знайдених рішень, унеможливлюючи порушення припустимих меж використання ресурсів, перевищення навантаження на вузли ХС або призначення задач у непридатні до обробки умови. Крім того, обмеження дозволяють

ураховувати специфіку поведінки атакуючого та обмеження захисника у виборі захисних стратегій. Формалізація цих обмежень є необхідним етапом для створення повної моделі, яка дозволить адекватно відображати реальні умови експлуатації ХС та забезпечить можливість застосування алгоритмів оптимізації до практичних сценаріїв розподілу задач.

В рамках опису обмежень кожна задача повинна бути призначена одному вузлу:

$$\sum_{i=1}^{n} x_{ij} = 1, \ \ \forall j \in \{1, \dots, m\}, \tag{2.13}$$

де n – кількість обчислювальних вузлів у хмарі; m – кількість задач для розміщення.

Загальне навантаження на вузол ХС не повинно перевищувати його ресурсів:

$$\sum_{j=1}^{m} x_{ij} \cdot L_j \leq C_i^{(cap)}, \ \ \forall i \in \{1, \dots, n\}, \tag{2.14}$$

де L_j – обчислювальне навантаження задачі T_i на вузлі; $C_i^{(cap)}$ – обчислювальна потужність вузла i.

Після формалізації обмежень (2.13) та (2.14), що визначають припустимі межі та умови роботи хмарної інфраструктури для завдання забезпечення безпечного, продуктивного та економічно ефективного розподілу обчислювальних задач ХС, постає необхідність у виборі результативного підходу до розв’язання поставленої багатокритеріальної задачі. Зважаючи на складність досліджуваного середовища, наявність суперечливих цілей, а також рухливу природу загроз, що моделюються в рамках ігрової взаємодії між атакуючим та захисником, використання класичних детермінованих методів оптимізації виявляється недостатнім або малоефективним. Це зумовило необхідність залучення релевантних евристичних підходів, здатних знаходити компромісні розв’язки у складних, багатовимірних просторах розв’язків з множинними цілями та обмеженнями.

Як раніше підкреслено у першому розділі дисертації релевантною є задача створення гібридного алгоритму, який поєднує переваги теоретико-ігрової оцінки

ризику для ХС із потужним інструментарієм багатокритеріальної еволюційної оптимізації. Подібний підхід дозволить інтегрувати стратегічну інформацію про потенційні дії зловмисника безпосередньо в процес оптимального розподілу задач. А це, у свою чергу, підвищує гнучкість і стійкість ХС до змін у ландшафті кібернетичних загроз для ХС. У якості оптимізаційного ядра в роботі пропонуємо застосувати модифікований алгоритм NSGA-II [119, 120]. Як зазначено у Розділі 1, а також у [119, 120] NSGA-II, є ефективним при розв'язанні багатокритеріальних задач, а також забезпечує генерацію Парето-оптимального фронту рішень. Уведення ігрової компоненти у вигляді параметрів ризику, що змінюються залежно від стратегій гравців, на нашу думку, створює умови для детального врахування безпекових факторів у процесі оптимізації.

А відтак, наступним кроком є створення гібридного алгоритму, див. рис. 2.1, структура та засоби якого забезпечують поєднання результатів стратегічного аналізу з багатокритеріальним еволюційним пошуком ефективних конфігурацій розподілу безпекових ресурсів ХС.

У межах запропонованої моделі гібридний алгоритм оптимізації, див. рис. 2.1, виконує роль обчислювального ядра, яке інтегрує результати стратегічного аналізу з боку потенційного атакуючого та захисника з можливістю пошуку оптимального балансу між суперечливими критеріями – безпекою, продуктивністю та вартістю. В основі алгоритму лежить ідея циклічної взаємодії між оцінкою ризику (як вихідної функції ігрової моделі) та багатокритеріальним еволюційним пошуком, який реалізуємо за допомогою модифікованого алгоритму NSGA-II.

Робота гібридного алгоритму окреслює декілька взаємопов'язаних фаз. На початковому етапі здійснюємо генерацію початкової популяції можливих розподілів обчислювальних задач між вузлами хмарної інфраструктури. Для кожного потенційного рішення, що описує конфігурацію розподілу, проводиться розрахунок рівня ризику з урахуванням стратегічної поведінки атакуючого, яка моделюється через параметр агресивності та ймовірнісну функцію вибору цілей. У такий спосіб, кожен індивід у популяції оцінювався не лише за критеріями продуктивності й вартості, але ще із позиції стійкості до потенційних атак.

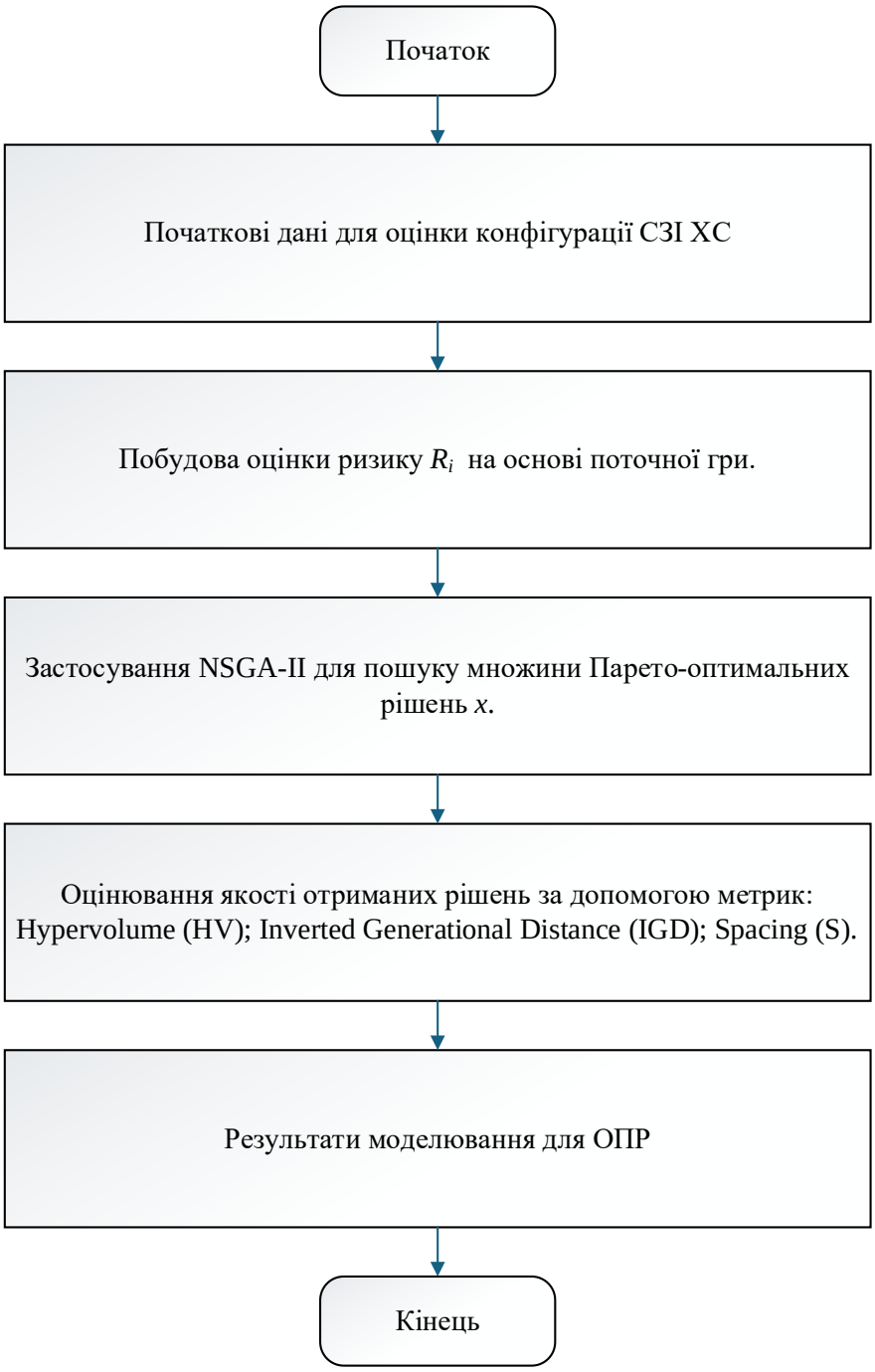


Рис. 2.1. Концептуальна схема гібридного алгоритму (запропоновано автором)

На наступній фазі виконуємо багатокритеріальний відбір за принципом Парето-домінування, з урахуванням оновлених функцій придатності, які включають у себе як традиційні техніко-економічні метрики, так і показники ризику.

Відбір, схрещування та мутація здійснюються відповідно до алгоритму NSGA-II. Це дозволяє результативне досліджувати простір розв’язків та уникати

локальних екстремумів. На кожній ітерації алгоритму реалізована повторна оцінка рівня ризику відповідно до зміненої конфігурації розподілу задач. Це забезпечувало гнучку інтеграцію безпекових факторів у процес оптимізації.

Гібридна природа алгоритму полягає саме в постійному обміні інформацією між ігровим компонентом, який моделює зміну поведінки атакуючого, та еволюційною частиною, що відповідає за поступове вдосконалення стратегій розподілу. Запропонований метод, на нашу думку, дозволяє не лише отримати набір Парето-оптимальних рішень, але й забезпечити, щоб ці розв'язки були релевантними до актуальних загроз ХС. Це, підкислить компроміс між продуктивністю, безпекою та вартісними показниками ХС.

Результатом роботи гібридного алгоритму є фінальна популяція розв’язків, що формує фронт Парето, на підставі якого зацікавлені сторони (адміністратори, системні інтегратори, аналітики з кібербезпеки хмарного середовища) можуть здійснювати вибір оптимальної конфігурації системи відповідно до поточних пріоритетів або обмежень. Подібний розв’язок надає не лише гнучкість і масштабованість, але й враховує поведінкові ознаки кібернетичних загроз ХС, що суттєво підвищує рівень реалізму та практичної значущості моделі.

Тоді нашу гібридну модель формалізуємо так.

Приймемо такі позначення для параметрів моделі:

N – кількість вузлів хмарної інфраструктури;

M – кількість обчислювальних задач;

$x_{ij} \in \{0,1\}$ – бінарна змінна, що вказує, чи призначена задача j вузлу i;

w_j – обсяг ресурсів, необхідний для виконання задачі j;

c_i – продуктивність обчислювального вузла i;

$cost_i$ – вартість використання вузла i за одиницю ресурсу;

C_i – загальна обчислювальна ємність вузла i;

u_i – доцільність атаки на вузол i з точки зору атакуючого (зважаючи значущість даних, критичність функцій тощо);

λ – параметр агресивності атакуючого, що моделює рівень його націленості на високоризикові або легкодоступні вузли;

p_i – ймовірність атаки на вузол i;

L_i – потенційні збитки у разі успішної атаки на вузол i;

R_i – очікуваний ризик для вузла i;

$F_1(x)$ – критерій мінімізації ризику (максимізація безпеки);

$F_2(x)$ – критерій мінімізації загального часу обробки;

$F_3(x)$ – критерій мінімізації вартості використання хмарних ресурсів.

Етап 1. Ігрова модель – ймовірність атаки на вузол ХС:

$$p_i = \frac{e^{\lambda \cdot u_i}}{\sum_{j=1}^{N} e^{\lambda \cdot u_i}}, i = 1,2, \dots, N. \quad (2.15)$$

Етап 2. Оцінка очікуваного ризику атаки на вузол i:

$$R_i = p_i \cdot L_i. \quad (2.16)$$

Етап 3. Функція безпеки ХС для розподілу задач:

$$F_1(x) = \sum_{i=1}^{N} R_i \cdot \left(\sum_{j=1}^{M} x_{ij} \cdot w_j \right). \quad (2.17)$$

Етап 4. Функція продуктивності (мінімізація часу опрацювання):

$$F_2(x) = \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \cdot \frac{w_j}{c_i}. \quad (2.18)$$

Етап 5. Функція вартості (вартість використання обчислювальних вузлів):

$$F_3(x) = \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} \cdot w_j \cdot cost_i. \quad (2.19)$$

Етап 6. Обмеження ємності вузлів:

$$\sum_{j=1}^{M} x_{ij} \cdot w_j \leq C_i, \forall i \in \{1,2, \dots, N\}. \quad (2.20)$$

Етап 7. Повна розподіленість задач у ХС:

$$\sum_{i=1}^{N} x_{ij} = 1, \forall j \in \{1,2, \dots, M\}. \quad (2.21)$$

Етап 8. Двоїстість змінних розподілу:

$$x_{ij} \in \{0,1\}, \forall i, j. \quad (2.22)$$

Етап 9. Задача багатокритеріальної оптимізації:

$$\min\big(F_1(x), F_2(x), F_3(x)\big). \quad (2.23)$$

Етап 10. Оцінка якості розв'язку.

10.1. Оцінка HV [121]:

$$HV = \text{об'єм простору, домінованого фронтом Парето.} \quad (2.24)$$

10.2. Оцінка IGD [122]:

$$IGD = \frac{1}{|P^*|}\sum_{v\in P}\min_{u\in P}\|u - v\|, \quad (2.25)$$

де P – отриманий Парето-фронт; P^* – справжній (референсний) Парето-фронт; $d(v,u)$ – відстань між точками v і u.

10.3. Метрика рівномірності розподілу Spacing [123]:

$$S = \sqrt{\frac{1}{|P|-1}\sum_{i=1}^{|P|}\big(d_i - \bar{d}\big)^2}, \quad (2.26)$$

де d_i – відстань між рішенням *i* та його найближчим сусідом; $\bar{d}$ – середнє значення d_i.

Система рівнянь (2.15)–(2.26) задає повну формальну постановку нашої гібридної моделі. Вона дозволяє відображати взаємодію стратегій атакуючого та захисника в умовах задачі забезпечення захисту хмарного середовища, забезпечуючи водночас можливість багатоцільової оптимізації із використанням NSGA-II. Нижче представлено опис запропонованого гібридного алгоритму у форматі псевдокоду [124, 125].

1 ***Input****: Set of computational tasks T, cloud nodes N, max generations G, population size PopSize*

2 Initialize population P(0) with PopSize individuals (initial task-node allocations)

3 ***for*** *generation g = 1 to G* ***do***

4 ***for*** *each individual s in P(g-1)* ***do***

5 Evaluate performance metrics: cost(s), time(s), and load(s)

6 Simulate attacker-defender game based on s to compute risk profile R(s)

7 Compute security evaluation sec(s) based on risk R(s)

8 Form fitness vector F(s) = [cost(s), time(s), sec(s)]

9 ***end for***

10 Apply non-dominated sorting on P(g-1)

11 Generate offspring population Q(g) via crossover and mutation

12 Combine populations: R(g) = P(g-1) ∪ Q(g)

13 Apply fast non-dominated sorting to R(g)

14 Select next generation P(g) using crowding distance and rank

15 ***end for***

16 ***Return*** *Pareto front P(G)*

У рядку 2 здійснюємо початкову ініціалізацію популяції розв’язків. Кожен індивід у популяції репрезентує потенційне відображення задач з множини T на вузли хмарної інфраструктури N. Це відображення закодоване у вигляді хромосоми з фіксованою довжиною, де кожен ген вказує на конкретний вузол i призначення для задачі j.

Центральним елементом гібридної моделі є цикл обчислення ризику в рядках 5–7. Після обчислення класичних характеристик (вартість, час виконання, навантаження) реалізована симуляція ігрової взаємодії між атакуючим та захисником (рядок 6). В рамках цієї симуляції обчислюємо стратегічну ймовірність атаки на кожен вузол і відповідні очікувані втрати. Цю інформацію агрегуємо у вигляді профілю ризику $R(s)$, який потім конвертуємо у числову оцінку безпеки $sec(s)$. Останню використовуємо як третій критерій оптимізації (рядок 7).

Рядки 10–14 реалізують класичну структуру алгоритму NSGA-II. У рядках 10 та 13 виконуємо швидке недомінноване сортування для оцінки рівнів переваги між розв'язками. У рядках 11–12 формуємо нову популяцію через генетичні оператори схрещування та мутації. А після формування нової популяції об’єднуються батьківська та нащадкова популяції для подальшого відбору найкращих особин (рядок 14). Запропонований метод дозволяє зберігати як різноманітність розв’язків,

так і фокус на Парето-оптимальні рішення з урахуванням трьох суперечливих цілей (продуктивності, вартості та безпеки ХС).

Нарешті, на виході (рядок 16) алгоритм повертає Парето-фронт рішень. У оптимальному розв'язку такий фронт продемонструє компроміси між основними критеріями. Це, дозволить суб’єкту прийняття рішень обрати відповідну стратегію розміщення задач у хмарному середовищі з урахуванням рівня ризику.

Відповідно до складеного алгоритму на наведеного псевдокоду, була проведена симуляція роботи моделі, в середовищі програмування PyCharm, див. рис. 2.2. Результаті симуляції наведено на рис. 2.3.

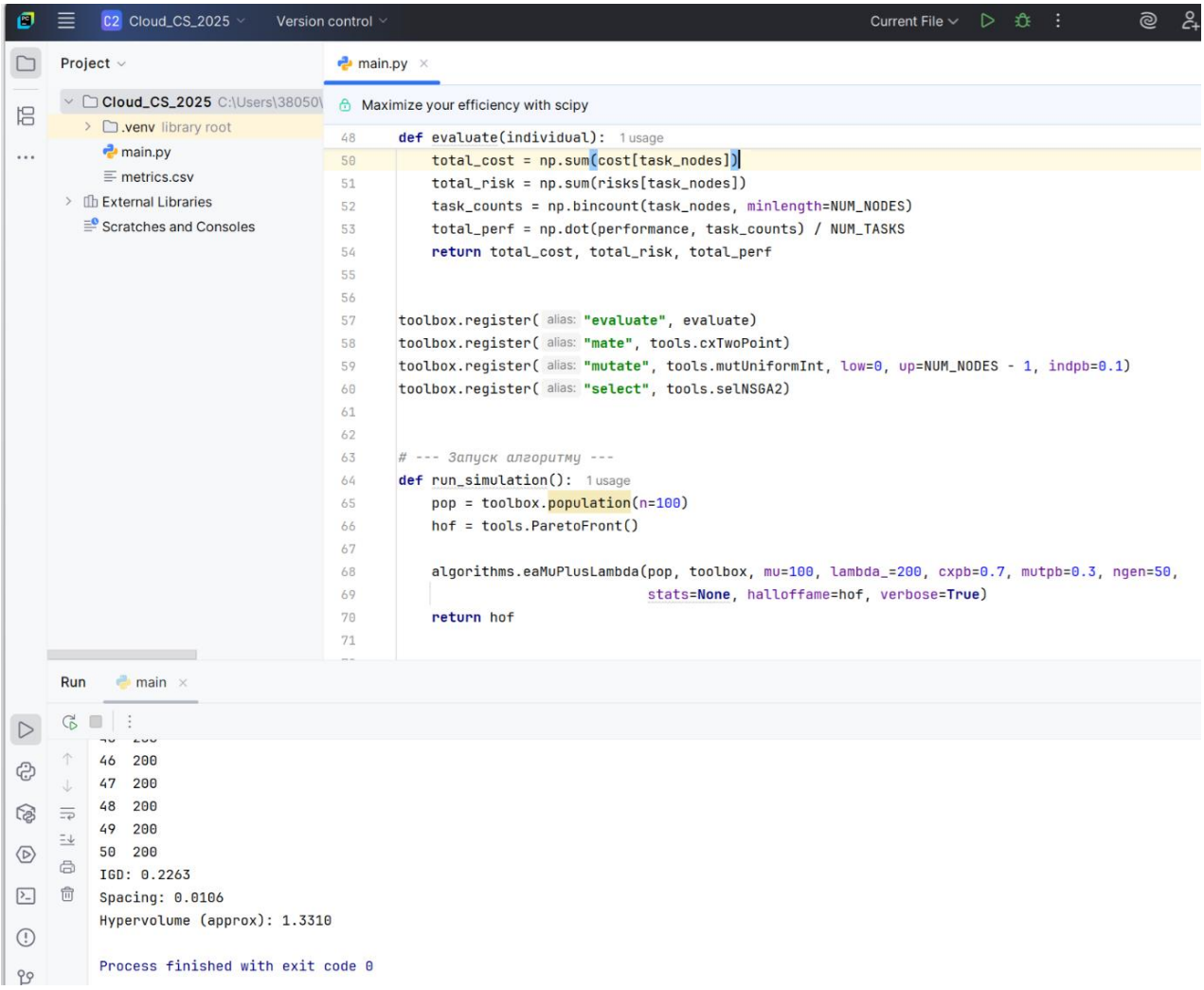


Рис. 2.2. Симуляція для запропонованої гібридної моделі

Отримані значення метрик якості, див. рис. 2.2 продемонстрували дієвість запропонованого гібридного підходу до оптимізації розподілу обчислювальних

ресурсів у ХС із урахуванням ризиків для безпеки. Значення метрики Inverted IGD на рівні 0,2263 свідчить про відносно невелику середню відстань між знайденим Парето-оптимальним фронтом, див. рис. 2.3 та еталонним набором розв'язків. Це вказує на задовільну збіжність алгоритму.

Метрика Spacing, яка становить 0,0106, див. рис. 2.2, підтверджує рівномірний розподіл знайдених розв’язків у просторі критеріїв. Таке низьке значення свідчить про відсутність значних скупчень розв'язків у локальних областях.

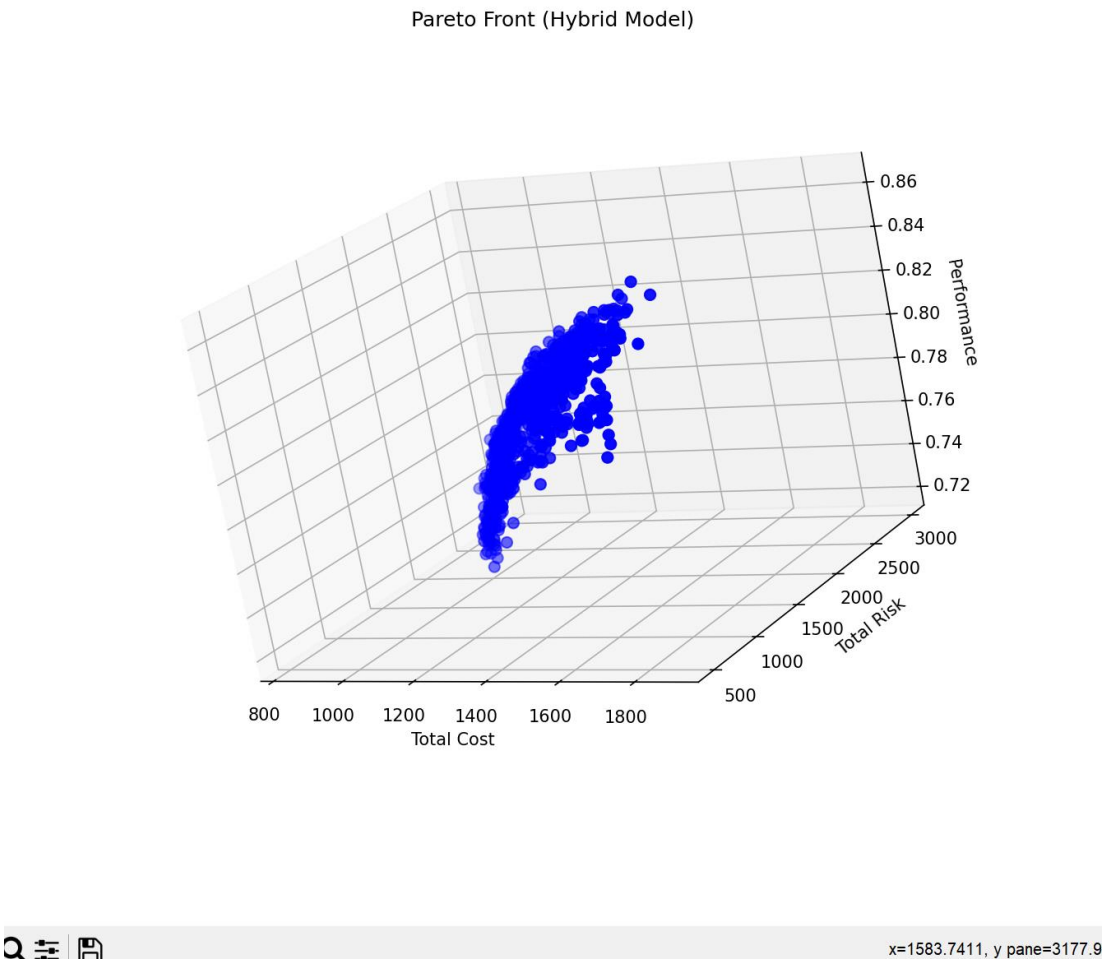


Рис. 2.3. Результати симуляції для запропонованої гібридної моделі та візуалізація Парето-оптимальних рішень

Апроксимоване значення гіпероб’єму (HV) 1,3310, рис. 2.2, показує обсяг простору критеріїв, який домінують знайдені розв'язки. Ця величина, хоча й є приблизною, дозволяє зробити висновок про достатню різноманітність отриманого

набору Парето-оптимальних рішень. Отримані результати підтвердили працездатність запропонованого підходу [126].

2.2. Розширена гібридна модель з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища

Як розкрито у першому розділі роботи у ХС загрози КІБ мають рухливий характер, адаптуються до поточних змін інфраструктури та активності користувачів. Статичні моделі ризику, засновані на фіксованих значеннях параметрів, не можуть повною мірою відобразити складність нових атак. Крім того, захисні стратегії мають адаптуватися до поточного рівня ризику, обмеженості ресурсів, а також до перспективи кооперації між підсистемами безпеки (або навіть між різними хмарними провайдерами в умовах мультихмарності).

Тому в рамках дослідження вважаємо доцільним розширити гібридну модель, розглянуту у п. 2.1, додавши:

варіативність параметра ризику $\lambda(t)$, який змінюємо в часі або в залежності від сценарію атаки на ХС;

гнучку стратегію захисника $\pi^D(t)$, що обирає рішення в залежності від спостережуваного ризику;

коаліційні або/та неантагоністичні ігрові моделі, що дозволяють моделювати співпрацю між захисниками, або обмежені цілі атакувальника, відмінні від повної деструкції.

Позначимо $\lambda(t) \in [0,1]$ – коефіцієнт ризику в момент часу t.

Залежність для розрахунку ризику запишемо так:

$$R(t) = \sum_{i=1}^{N} \lambda(t) \cdot p_i(x_i, t), \tag{2.27}$$

де $p_i(x_i, t)$ – ризик атаки на ресурс i у момент часу t; x_i – розподіл обчислювальних задач на ресурсі i.

Параметр $\lambda(t)$ – змінюємо відповідно до однієї з моделей:

стохастичної $\lambda(t) \sim u[\lambda_{\min}, \lambda_{\max}]$,

марковської, де дискретні стани ризику з матрицею переходів P_λ,

еволюційної, коли тренд зростання або зниження $\lambda(t)$ знаходится в залежності від минулих атак.

Підкреслимо, що у розширеній моделі розподілу обчислювальних ресурсів з урахуванням кіберзагроз ризик не є стабільною величиною. Він міняється в залежності від низки факторів, пов'язаних як з діями атакувальника, так і з особливостями поведінки користувачів, навантаженням на систему, станом її компонентів, та навіть з історією попередніх атак. Саме тому в поточному параграфі $\lambda(t)$ вводиться як функція часу. Або навіть ширше, як функція стану системи. Це дозволяє на наступних етапах дослідження моделювати реалістичні сценарії розвитку подій при атаках на ХС.

В умовах підвищеного навантаження на хмарну інфраструктуру, скажімо, під час запуску великого онлайн-заходу або періоду масової звітності у фінансовій сфері система є більш вразливою до атак. Це стається через перевантаження ресурсів і зниження результативності засобів виявлення вторгнень. У такому випадку параметр $\lambda(t)$ має зростати, сигналізуючи про підвищення ризику. І навпаки, після впровадження нових компонентів верифікації доступу або після завершення пікового навантаження, ризик знижується. Величина $\lambda(t)$ теж зменшиться. Параметр $\lambda(t)$ відображає поточний стан безпекової ситуації в ХС.

Так само, слід зазначити, що у попередньому параграфі п. 2.1 параметр λ інтерпретувався як інтенсивність атак або агресивність дій атакувальника. Інакше кажучи параметр λ в моделі (2.15)–(2.26) фактично описував частоту атак або силу тиску з боку зловмисника. Параметр λ був елементом ігрової моделі, яка моделює стратегії нападника у вигляді потоку або частоти загроз.

У поточному параграфі (п. 2.2), де йдеться про ризик, $\lambda(t)$, цей параметр вже виступає як узагальнена характеристика рівня ризику в системі, який включає не лише інтенсивність атак, але й низку інших чинників. Передусім, таких як вразливість ХС, неефективність її захисту, наявність активних експлойтів, внутрішніх збоїв, тощо.

Підкреслимо, що це не суперечність. А є розширенням інтерпретації параметра λ в рамках більш загальної моделі. Зокрема, у базовій (ігровій) моделі параметр λ інтерпретований локально. Приміром, як стратегічний параметр гравця (нападника), що визначає його поточну поведінку. У розширеній же моделі $\lambda(t)$ вже набуває системного змісту і вважається функцією часу або стану, яка акумулює вплив як зовнішніх загроз (дій нападника або початкове значення λ), так і внутрішніх умов (вразливості, навантаження, відповідь захисника).

Для опису того, як саме змінюється $\lambda(t)$, доцільно розглянути декілька моделей його поведінки. Стохастична модель визначає, що ризик змінюється випадковим чином у певних межах, відображаючи ситуації, коли точну природу загрози неможливо передбачити, а атаки мають ознаки випадкових вторгнень. Передусім, у відкритих публічних хмарах може відбуватись хаотичне сканування портів або масові фішингові кампанії, які виникають незалежно від стану конкретної системи. У цьому випадку $\lambda(t)$ трактуємо (обчислюємо) як випадкову величину з певного інтервалу. А стратегія захисника повинна бути адаптованою до невизначеності.

Інша ситуація – коли ХС має справу з більш структурованими атаками. Скажімо, з боку постійно діючого суб'єкта таргетованих атак (або APT). Таку атаку реалізують зазвичай за певними шаблонами [127, 128]. У такому випадку дійовою є Марковська модель зміни ризику, де система переходить із одного стану в інший з певними ймовірностями [129, 130]. Наприклад, з «низького» ризику у «середній», або з «середнього» у «високий» ризик, залежно від кількості несанкціонованих спроб входу, виявлених у певний проміжок часу. Таке моделювання дозволить включати в систему передбачення майбутнього ризику на ґрунті оцінки поточного стану і допомагає побудувати оптимальні стратегії захисту ХС наперед.

Нарешті, еволюційна модель $\lambda(t)$ визначає, що ризик змінюється поступово в часі, відображаючи або накопичення загроз, або їхній спад у результаті результативної роботи захисту [131, 132]. Після виявлення нової вразливості у віртуалізаційному шарі (скажімо у гіпервізорі), $\lambda(t)$ починає зростати, оскільки очікується поява експлойтів у відкритому доступі. Протягом кількох днів рівень

ризику зростає, поки не буде виправлено вразливість або впроваджено відповідні патчі. У такій моделі $\lambda(t)$ є функцією, що залежить не тільки від зовнішніх умов, але й від дій самої системи. Або від її спроможності до виявлення та реагування. Таки моделі забезпечують гнучкість у відображенні довгострокових трендів кіберзагроз.

Тому моделювання ризику через параметр $\lambda(t)$ дозволяє нам на цьому етапі досліджень адаптувати інтелектуальну систему (яку ми детально розглянемо у заключному розділі дисертації) до різних сценаріїв розвитку подій. Залежно від властивостей середовища – чи то обмежена інформація про зловмисника, чи детерміновані шаблони його поведінки, чи тривалі зміни ризиків – відповідна модель $\lambda(t)$ дозволяє приймати оптимальні рішення щодо розподілу ресурсів, пристосування стратегії захисту хмарного середовища та формування ефективного опору кіберзагрозам. Зазначимо, що урахування величини ризику через змінну функцію $\lambda(t)$ дозволило нам перейти від спрощеної (статичної) оцінки загроз до більш реалістичного сценарію, де ризик формувався під впливом численних факторів, включаючи дії зловмисників, вразливість ХС, обмеження ресурсів та час реакції. Перехід до адаптивної інтерпретації параметра ризику природно висуває нові вимоги до поведінки захисної сторони. У таких умовах фіксована або наперед задана стратегія захисту виявиться недостатньо дієвою. Першою чергою це стосувалося випадків коли атаки на ХС стають все більш складними або змінюють свою природу в часі. Тому наступним вирішальним елементом розширеної гібридної моделі є формалізація стратегії захисника, яка враховує поточний рівень ризику, обмеження ресурсів та прогнозовану поведінку нападника для перерозподілу задач, посилення захисту критичних компонентів і мінімізації потенційних втрат.

Рішення захисника в момент часу t позначаємо як $\pi^D(t)$, де

$$\pi^D(t) = \arg\max_{\pi \in \Pi} U_D(\pi, \lambda(t), C), \tag{2.28}$$

де U_D – доцільність стратегій захисника ХС, яка бере до уваги ризик $R(t)$, продуктивність системи $P(t)$ та вартість C.

Формально

$$U_D = w_1 \cdot P(t) - w_2 \cdot R(t) - w_3 \cdot C(\pi), \quad (2.29)$$

де w_1, w_2, w_3 – вагові коефіцієнти.

Вагові коефіцієнти w_1, w_2, w_3 визначаємо на підставі:

політики безпеки підприємства. Передусім, критичні бази даних які компанії розтасовують у ХС отримують більший w_i, ніж допоміжні хмарні сервіси;.

класифікації інформаційних активів. Вузли ХС, що обробляють конфіденційну інформацію, одержують вищий пріоритет;

аналізу впливу. Як, приклад, коефіцієнт w_i пропорційний до потенційних втрат у разі успішної атаки на ХС.

емпіричних даних. Скажімо, виходячи з історії інцидентів у хмарній інфраструктурі, деякі типи задач потенційно можуть бути частіше атакованими. Інакше кажучи, їх вага підвищується.

Як обґрунтовано у [133], доцільним є запровадження гнучкої стратегії захисника, яка бере до уваги зміну рівня загроз у часі, обмеження на ресурси захисту, та пріоритетність тих чи інших вузлів або обчислювальних задач. Метою є своєчасна переорієнтація ресурсів безпеки відповідно до змін параметра агресивності атакуючого $\lambda(t)$ та розподілу ризиків у системі.

Формально, варіативну стратегію захисника представимо як функцію прийняття рішень [134]:

$$A(t) = \arg \max_{a(t) \in A} \left[\sum_{i=1}^{n} w_i \cdot \phi_i \left(x_i(t), \lambda(t), R_i(t) \right) \right], \quad (2.30)$$

де $A(t)$ – вектор варіативних дій захисника в момент часу (t). Насамперед, зміна маршрутів задач, посилення захисту, міграція задач на менш ризиковані вузли ХС; A – множина припустимих дій; w_i – ваговий коефіцієнт, що відображає значення (або критичність ресурсу/вузла ХС; ϕ_i – функція дієвості захисних дій ХС, що враховує значення ризику $R_i(t)$, агресивність атакуючого $\lambda(t)$ та поточний стан обчислювального вузла ХС $x_i(t)$; $R_i(t)$ – оцінка ризику для вузла в момент часу (t), що отримано з попереднього етапу (оцінка ризику на базі результатів гри); $x_i(t)$ – стан задачі/вузла – завантаження, залишок ресурсів, критичність функцій, які виконує задача.

Відмітимо, що якщо попередні судження в дисертації описували протистояння між атакуючим і захисником як гру з нульовою сумою, де виграш одного точно дорівнює втратам іншого, то в реальних сценаріях взаємодія сторін складніша. Насамперед у багатокористувацьких або мультиорганізаційних хмарних середовищах виникають ситуації, де захисники різних компонентів системи змушені кооперувати свої ресурси для підвищення загальної безпеки, тоді як атакуючі можуть діяти не індивідуально, а в рамках узгоджених стратегій або навіть коаліцій.

За таких обставин класична модель гри стає надто спрощеною для адекватного моделювання реальних загроз і стратегій реагування на них.

Саме тому, розгляд кооперативних або неантагоністичних моделей дозволив в рамках дослідження змістити фокус з жорсткої конкуренції до більш реалістичної взаємодії. В такій взаємодії можливе часткове узгодження інтересів сторін. Або хоча б існування спільного простору рішень, в рамках якого оптимізаційна задача не обов’язково має єдиного переможця. Як-от у випадках, коли атакуючий не має за мету повне виведення ХС з ладу, а прагне отримати вигоду у формі витоку обмеженої кількості даних або незначного контролю, можлива часткова рівновага, яку доцільно моделювати в рамках неантагоністичних біматричних ігор.

Аналогічно, захисники можуть формувати коаліції. Це можуть бути коаліцій як формальні (між компаніями-партнерами), так і динамічні (між компонентами кластерної інфраструктури ХС). А це вимагає врахування аспектів розподілу спільного виграшу, створення довіри між захисниками та оптимального об'єднання ресурсів.

Формально це вимагає переходу від класичної матриці виграшу до біматричної моделі зі змінною вигодою, де кожен учасник отримує свою вигоду відповідно до обраної стратегії та стратегії супротивника, і ця вигода не є обов’язково симетричною чи взаємовиключною. Така модель дозволяє описувати взаємодію у вигляді пари функцій виграшу:

$$\big(U_D(s_D, s_A), U_A(s_D, s_A)\big), \tag{2.31}$$

де s_D, s_A – відповідно, стратегії захисника та атакуючого, а функції виграшу можуть залежати від зовнішніх змінних. Як-от рівень ризику, стан системи чи доступність ресурсів.

Це розширення, на нашу думку, слушне в умовах багатокритеріальної оптимізації, коли використовуємо алгоритм, як у нашому випадку алгоритм NSGA-II. Адже останній дозволяє знайти множину Парето-оптимальних стратегій не лише в сенсі максимізації/мінімізації класичних метрик (надійності, вартості, продуктивності), а й з урахуванням взаємодії між гравцями, які мають неідентичні й не завжди протилежні інтереси.

Тож резюмуючи наші судження, зазначимо, що включення кооперативної або неантагоністичної ігрової моделі в дослідження не лише підвищує рівень реалізму математичного опису загроз у хмарній системі, але й дає змогу нам змогу інтегрувати її з багатоцільовою оптимізацією. Подібна інтеграція дозволяє, на нашу думку, сформувати стратегії, які не лише дієві в умовах конфлікту, а й гнучко адаптуються до ситуацій, коли співпраця або компроміс між сторонами – бажаний або навіть необхідний варіант рівноваги.

В результаті отримуємо більш загальну гібридну модель, яка включає як елементи оптимального розподілу ресурсів (через NSGA-II), так і моделі прийняття рішень у середовищі зі змінними, але не обов'язково конфліктними інтересами.

Тоді подамо ігрову модель так:

Нехай:

Захисник – гравець D,

Атакувальник (зловмисник / (зловмисники для кооперативної ігри)) – гравець A,

Π^D, Π^A – множини стратегій, відповідно.

Тоді у кооперативному або неантагоністичному сценарії маємо систему рівнянь (2.31) та (2.32):

$$U_D(\pi^D, \pi^A) = f\big(P(t), R(t), C(\pi^D)\big), \quad (2.32)$$

$$U_A(\pi^D, \pi^A) = g(Control, Visibility, Cost_A), \quad (2.33)$$

де f, g – відповідні функції вигоди сторін. Та гравці не обов'язково мають повністю протилежні цілі.

Якщо в ХС декілька захисників $D_1, D_2, \ldots, D_m$, належить використати ігри з коаліціями. Таки ситуації є типовими для розподілених хмарних архітектур, де різні компоненти інфраструктури (сервери, віртуальні машини, сервіси, мережеві вузли та ін.) належать або управляються різними адміністративними доменами. У таких випадках загроза атаки торкнеться кількох захисників одночасно, і виникає потреба у кооперації, щоб знизити ризик, зменшити загальні втрати або ефективніше розподілити ресурси на протидію.

Наведемо невеликий приклад ситуації для використання ігор з коаліціями, див. табл. 2.3.

Таблиця 2.3

Ілюстрація формування коаліції захисників у хмарній інфраструктурі (складена автором)

Параметр	Опис	Випадок для трьох хмарних провайдерів (U1, U2, U3)
Учасники коаліції	Суб'єкти, що об'єднують ресурси для спільного захисту ХС	U1 (відеосервіси), U2 (опрацювання даних), U3 (аналітика)
Виграш коаліції $v(S)$	Зменшення загального ризику внаслідок співпраці.	—
Внесок кожного учасника коаліції	Розподіл виграшу згідно з принципом справедливості (напр., значення Шеплі [135, 136])	—
Ресурси коаліції	Спільне використання інструментів захисту, даних про загрози тощо	U1 надає систему DDoS-захисту, U2 – моніторинг вразливостей, U3 – алгоритми аналітики для виявлення аномалій
Ілюстрація реального застосування	Кооперація для протидії DDoS-атаці на спільний освітній портал	Атака на портал зберігає його доступність через перерозподілення навантаження між U1, U2, U3

Відповідно до прикладу, який розглянуто у табл. 2.3, уявімо собі хмарне середовище, в якому розгорнуто об'єднану платформу освітніх послуг, що надається трьома університетами (назвемо їх U1, U2, U3). Кожен з університетів

має свою локальну інфраструктуру в хмарі, але й відкриває частину своїх ресурсів спільному освітньому порталу. Параметр U1 обслуговує навчальний контент і відеосервіси. Тоді як U2 відповідає за обробку персональних даних студентів. А U3 виконує аналітичну опрацювання запитів та рекомендацій. Кожен із цих вузлів має своїх захисників, які можуть в автономному режимі будувати системи безпеки. Як, приклад, здійснювати налаштування брандмауерів, IDS/IPS, шифрування даних тощо. Проте в разі скоординованої DDoS, EDoS -атаки, або таргетованого відпливу персональних даних через вузол U2, усі три учасники втрачають репутацію, і навіть якщо конкретна атака торкнулася лише одного вузла, наслідки розповсюджуються на всю хмарну платформу. Логічно, що у такій ситуації стає вигідним сформувати коаліцію захисників, які обмінюються інформацією про загрози в реальному часі, та розподіляють ресурси захисту залежно від рівня загрози на кожному вузлі та координують стратегії реагування. Тоді у термінах теорії ігор [137] таку взаємодію моделюємо як коаліційну гру. В подібній грі учасники формують коаліції з метою максимізації колективної вигоди або мінімізації загальних втрат [138]. При цьому виникають задачі розподілу виграшу всередині коаліції. Інакше кажучи хто скільки отримає від запобігання атаці, і як оцінити внесок кожного захисника.

Найперше, якщо завдяки коаліції уникнено збитків у розмірі 100000 у.о., доречно з'ясувати, яку частину цієї «вигоди» приписати кожному з вузлів. Це доцільно зробити насамперед через значення Шеплі (Shapley value) [135, 136], що оцінює внесок кожного гравця в коаліцію.

Тоді виграш коаліції $S \subseteq \{D_1, D_2, \dots, , D_m\} - v(S)$, визначаємо як зменшення ризику:

$$v(S) = \Delta R_S = R_{\text{без}S} - R_{\text{з}S}, \qquad (2.34)$$

де $R_{\text{без}S}$ – параметр, який використовуємо у задачах розрахунку виграшу коаліції в іграх з переданим виграшом. Параметр $R_{\text{без}S}$ використовуємо для оцінки внеску коаліції до загального виграшу або втрат системи; $R_{\text{з}S}$ – виграш системи або рівень безпеки, який досягнутий без участі коаліції S. Інакше кажучи, це базовий результат системи (як-от, рівень виявлення атак, зменшення втрат, показник ризику), коли дана група захисників не бере участі у спільних діях або діє

автономно; $R_{\text{без}S} - R_{\text{з}S}$ – внесок коаліції S у загальну безпеку ХС. Власне, ця різниця $v(S)$ є виграш коаліції, який потім використовуємо для розрахунку справедливого розподілу вигоди між її учасниками (насамперед, за допомогою значення Шеплі [135, 136]).

При використанні багатокритеріальних еволюційних алгоритмів, NSGA-II (або NSGA-III), ми зможемо оцінювати ефективність різних коаліцій як частину множини можливих стратегій захисників ХС, включати виграш коаліції $v(S)$ як один із критеріїв оптимізації (разом із вартістю, надійністю, продуктивністю). Але це вже чотири критерія тому потрібно задіяти NSGA-II. Так саме варте в подальшому розглядати утворення коаліцій як еволюційно обумовлений процес, де домінантними залишаються ті стратегії взаємодії, що забезпечують найвищу вигоду ХС.

Враховуючи всі судження, опишемо формальну подачу загальної цілі оптимізації в розширеній моделі.

Потрібно знайти розподіл задач $X = (x_i)$, та оптимальні стратегії $\pi^D(t)$, що максимізують векторну функцію:

$$\begin{cases} Maximize\ F_1 = \mathbb{E}[P(t)]; \\ Minimize\ F_2 = \mathbb{E}[R(t)]; \\ Minimize\ F_3 = \mathbb{E}[C(t)]; \\ Maximize\ F_4 = v(S), \qquad (\text{коаліційна вигода}), \end{cases} \tag{2.35}$$

при обмеженнях:

$$\begin{gathered} x_i(t) \in [0, x_i^{max}], \\ \sum_i x_i(t) \le R^{total}(t), \\ \pi^D(t) \in \Pi, \text{ залежно від } \lambda(t). \end{gathered} \tag{2.36}$$

Як і минулого разу наведемо псевдокод для розширеної гібридної моделі.

1: Ініціалізувати множину задач T та множину вузлів N

2: Ініціалізувати початкову популяцію рішень P(0)

3: Задати початкове значення параметра λ для кожного вузла

4: ***while*** *(не досягнуто критерію зупинки)*

5: Оцінити стан системи: A(t), λ(t), R(t)

6: Оновити значення λ(t) відповідно до вибраної моделі (Марковської / еволюційної)

7: Застосувати гнучку стратегію захисника:

8: Моніторинг активності атак

9: Перерозподіл задач між вузлами

10: ***if*** *(виявлено коаліцію атакуючих або захисників)*

11: Провести аналіз вигод RбезS та RзS для кожної коаліції

12: Оновити структуру взаємодії (кооперативна/неантагоністична гра)

13: Виконати NSGA-II: відбір, кросовер, мутація

14: Оцінити кожне рішення за критеріями F1–F4

15: Зберегти поточну популяцію та значення метрик (IGD, Spacing, HV)

16: ***end while***

17: Вивести фінальне Парето-оптимальне рішення

Блок-схема алгоритму для розширеної гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища наведена на рис. 2.4.

А далі наведемо пояснення для основних фрагментів псевдокоду.

В рядках 1–3 ініціалізуємо структуру задач та обчислювальної інфраструктури, та встановлюємо початкове значення агресивності λ для кожного вузла ХС.

У рядку 5 фіксуємо поточний стан ХС, включно з рівнем атак, ризиком та параметром λ.

У 6 рядку відповідно до обраної моделі ризику (Марковської або еволюційної), значення λ змінюється в часі, відображаючи зміну поведінки атакуючих.

В рядках 7–9 реалізуємо варіативну (гнучку) стратегію захисника. Скажімо, при зростанні $\lambda(t)$ або зменшенні доступності ресурсів, задачі перерозподіляються на менш уразливі вузли ХС.

В рядках 10–12 перевіряємо, чи виникли коаліції гравців. Якщо так, то тоді використовуємо модель з обчисленням виграшів коаліцій для прийняття

стратегічних рішень. Це дозволить змоделювати реальні сценарії співпраці між захисниками ХС.

В рядку 13 застосовуємо алгоритм NSGA-II для пошуку Парето-оптимального розподілу задач.

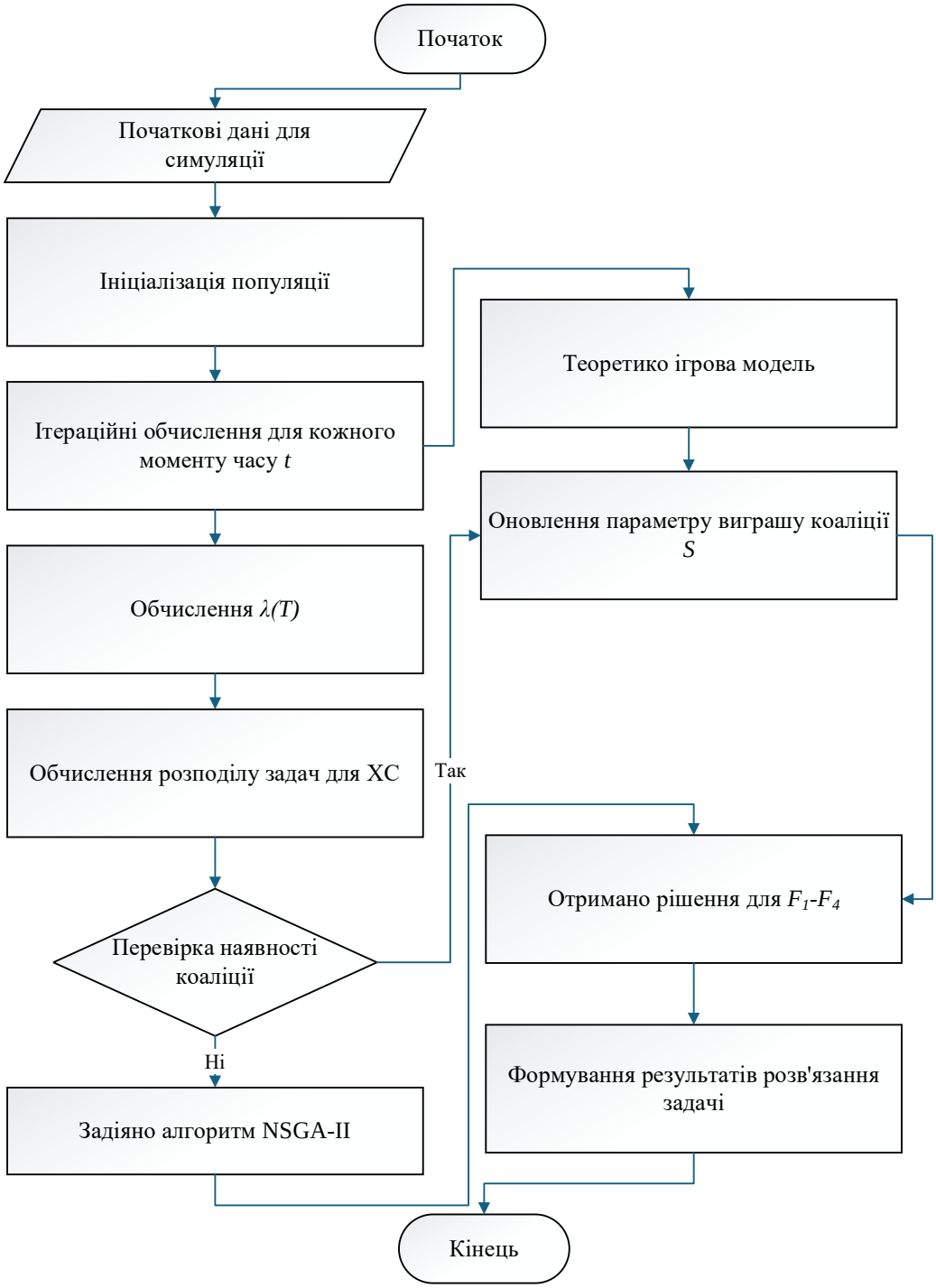


Рис. 2.4. Блок-схема алгоритму для розширеної гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту ХС

В рядку 14 виконуємо оцінку кожного розв’язку. Це реалізовано за чотирма критеріями (2.34).

В рядку 15 здійснюємо збереження результатів. Результати зберігаються разом з обчисленими метриками якості, що дозволяє провести постсимуляційний аналіз. На цьому ми більш детально зупинимося у завершальному розділі дисертації.

В рядку 17 формуємо вихід моделі – множину оптимальних розв'язків, графіки змін $\lambda(t)$, ризику та розподілу задач.

Тому блок-схема, представлена на рис. 2.4, відтворює логіку функціонування розширеної гібридної моделі оптимізації розподілу обчислювальних ресурсів у ХС з урахуванням ризику, варіативної поведінки захисника та кооперативної взаємодії між суб’єктами захисту. Згідно з послідовністю етапів, представлених на рис. 2.4, процес розпочинаємо з ініціалізації початкових даних симуляції. Ці дані включають параметри задач, ресурсів хмарної інфраструктури, та початкові характеристики популяції розв'язків, що будуть надалі оброблятися у межах алгоритму багатокритеріальної оптимізації, відповідно до моделі (2.27)–(2.36).

Після цього запускаємо ітеративний цикл, у межах якого здійснюємо обчислення змінного параметра ризику $\lambda(t)$. У подальшому модель (2.27)–(2.36) виконує перевірку наявності коаліційної взаємодії між захисними агентами. У випадку, якщо коаліція відсутня, система залучає алгоритм NSGA-II для пошуку Парето-оптимальних рішень відповідно до сформульованих критеріїв $F_1 - F_4$.

Якщо ж коаліція між захисниками все-таки сформована, розгортання процесу переходить до інтеграції теоретико-ігрової моделі. Ця модель дозволяла враховувати вигоди колективної взаємодії для критеріїв $F_1 - F_4$. У цьому випадку здійснюємо оновлення параметру виграшу коаліції, що віддзеркалює внесок кожного з учасників до спільної мети захисту ХС. На підставі отриманих розв’язків, незалежно від того, чи були вони сформовані за допомогою NSGA-II, чи в рамках кооперативної моделі, реалізуємо підсумкову оцінку значень цільових функцій та формуємо остаточні результати розв’язання задачі.

Зазначимо, що аналіз теоретичної складності запропонованої гібридної моделі доцільно проводити з урахуванням основних її структурних компонентів, кожен з

яких має власну алгоритмічну природу. Наша модель (2.27)–(2.36) включає обчислення рівня ризику *λ(t)*, реалізацію кооперативної взаємодії учасників системи на ґрунті теоретико-ігрового підходу, та багатокритеріальну оптимізацію з використанням еволюційного алгоритму NSGA-II. Розглянемо часову (обчислювальну) та просторову складності кожної складової окремо.

Обчислення рівня *λ(t)*, з урахуванням зміни поточних загроз та стану захисних механізмів, передбачає аналіз вхідних даних, таких як історія інцидентів, рівень захищеності ресурсів та актуальний стан ХС. У випадку дискретизації часу з кроком Δt, складність оцінювання функції ризику на кожному кроці є $O(n)$, де n – кількість активних задач у ХС.

Загальна складність обчислення ризику протягом періоду моделювання з T дискретними моментами часу оцінюємо як $O(T \cdot n)$.

Просторова складність обмежується збереженням тимчасових послідовностей значень параметрів задач і станів ресурсів, що становить $O(n \cdot T)$.

Побудова коаліційної взаємодії між захисниками ХС базувалася на формуванні множини припустимих коаліцій, розрахунку виграшу кожної коаліції та визначенні рівноважного розподілу прибутку. У випадку наявності (m) захисників, кількість можливих коаліцій зростає експоненційно і становить 2^m. Для кожної коаліції варто виконати оцінювання виграшу, що базувалася на стані системи та рівні загроз. Складність цієї компоненти в загальному випадку оцінюємо як $O(2^m \cdot c)$, де c – вартість обчислення виграшу однієї коаліції.

Просторова складність визначалася як $O(2^m)$, для збереження виграшів або стратегій усіх коаліцій.

Блок багатокритеріальної оптимізації, реалізований за допомогою алгоритму NSGA-II, має складність $O(P^2)$, на кожній ітерації, де P – розмір популяції. Це пов’язано з необхідністю виконання операцій сортування за фронтами Парето. Тоді при загальній кількості ітерацій G, повна часова складність алгоритму становить $O(G \cdot P^2)$. А просторова складність є $O(P \cdot d)$, де d – розмірність вектору розв’язків (у даній моделі – чотири критерії $F_1 - F_4$).

Узагальнюючи, повна обчислювальна складність моделі на одному етапі симуляції (для фіксованого моменту часу) оцінено так:

$$O(n + 2^m \cdot c + G \cdot P^2), \quad (2.37)$$

а відповідна просторова складність:

$$O(n + 2^m + P \cdot d). \quad (2.38)$$

Ці оцінки свідчать про прийнятну складність реалізації моделі для середніх масштабів ХС, маємо такі параметри: $m < 10$, $n \sim 10^2 - 10^3$, $P \sim 100$, $G \sim 100$.

В умовах масштабування, та збільшення кількості коаліційних учасників захисту ХС, виникне потреба у використанні апроксимаційних або евристичних компонент для зменшення експоненційної частини у коаліційній взаємодії. Але ця задача не входила до переліку завдань дисертаційного дослідження.

Для запропонованого в поточному параграфі дисертації моделі (2.27)–(2.36) була проведена симуляція з урахуванням ризиків та кооперативних стратегій захисту ХС, див. рис. 2.5.

Експериментальне дослідження ефективності запропонованого гібридного підходу проведено на синтетичних даних, згенерованих для імітації роботи хмарної інфраструктури. Дані для симуляції наведено у табл. 2.4.

В ході дослідження виконано 300 поколінь оптимізації з використанням модифікованого алгоритму NSGA-II, що дозволило отримати 100 Парето-оптимальних розв'язків, див. рис. 2.5 (візуалізовано у терміналі IDE PyCharm).

Якість знайденого фронту оцінювалася за допомогою трьох вирішальних метрик:

гіпероб’єму HV;

зворотної генераційної відстані IGD;

метрики рівномірності розподілу (Spread).

Синусоїдальна компонента $0{,}2 + 0{,}1 \cdot sin\left(\frac{2\pi t}{100}\right)$ імітує циклічність кіберзагроз, пов’язаних із плановими навантаженнями на систему. Параметр t інтегрує інформацію про розподіл задач. Це в моделі дозволяє брати до уваги їхню взаємозалежність.

Таблиця 2.4

Параметри синтетичних даних для моделювання розподілу обчислювальних задач у хмарній системі (складено автором)

Параметр	Діапазон значень / значення	Пояснення
Кількість обчислювальних задач	10 (фіксовано)	Визначає загальну кількість задач, що підлягають розподілу по вузлах хмарної інфраструктури
Кількість ресурсних вузлів	5 (фіксовано)	Відповідає за кількість доступних обчислювальних вузлів у хмарній системі
Рівень ризику вузлів	[0,1, 0,5] (рівномірний розподіл)	Характеризує вразливість кожного вузла до кібератак. Генеруємо окремо для кожного вузла
Час обробки задач	[5, 20] (рівномірний розподіл)	Час виконання кожної задачі на конкретному вузлі (умовні одиниці). Формує матрицю задачі×вузол
Вартість опрацювання	[10, 50] (рівномірний розподіл)	Витрати на виконання задачі на обраному вузлі. Формує матрицю задачі×вузол
Користь від коаліції	[0,5, 2,0] (рівномірний розподіл)	Додаткова перевага від спільної роботи задач на одному вузлі. Унікальне значення для кожного вузла
Ризик	$0{,}2 + 0{,}1 \cdot sin\left(\frac{2\pi t}{100}\right)$	Функція, що моделює часову зміну рівня загроз у системі, де t – сумарний індекс задач
Розмір популяції	100 (фіксовано)	Кількість індивідуумів у популяції генетичного алгоритму за одне покоління
Кількість поколінь	300 (фіксовано)	Максимальна кількість ітерацій алгоритму оптимізації
Ймовірність схрещування	0,85	Імовірність застосування оператора схрещування до пари індивідуумів
Ймовірність мутації	0,15	Імовірність застосування оператора мутації до індивідуума

Результати такої симуляції подано на рис. 2.6–2.8.

Метрика зворотної генераційної відстані зафіксувала значення IGD = 0,5171. Це відтворює середню відстань між знайденими рішеннями та точками референсного фронту. Хоч цей показник і не досяг ідеального рівня, він свідчить про прийнятну близькість до теоретично оптимальних розв’язків, приймаючи до уваги складність задачі з чотирма конфліктними критеріями. Зазначимо, що генерація референсного фронту виконувалась шляхом випадкового пошуку, що могло обмежити точність порівняння. А крім того ми використовували синтетичний набір даних для попереднього етапу дослідження, де головним була перевірка працездатності нашої моделі (2.27)–(2.36).

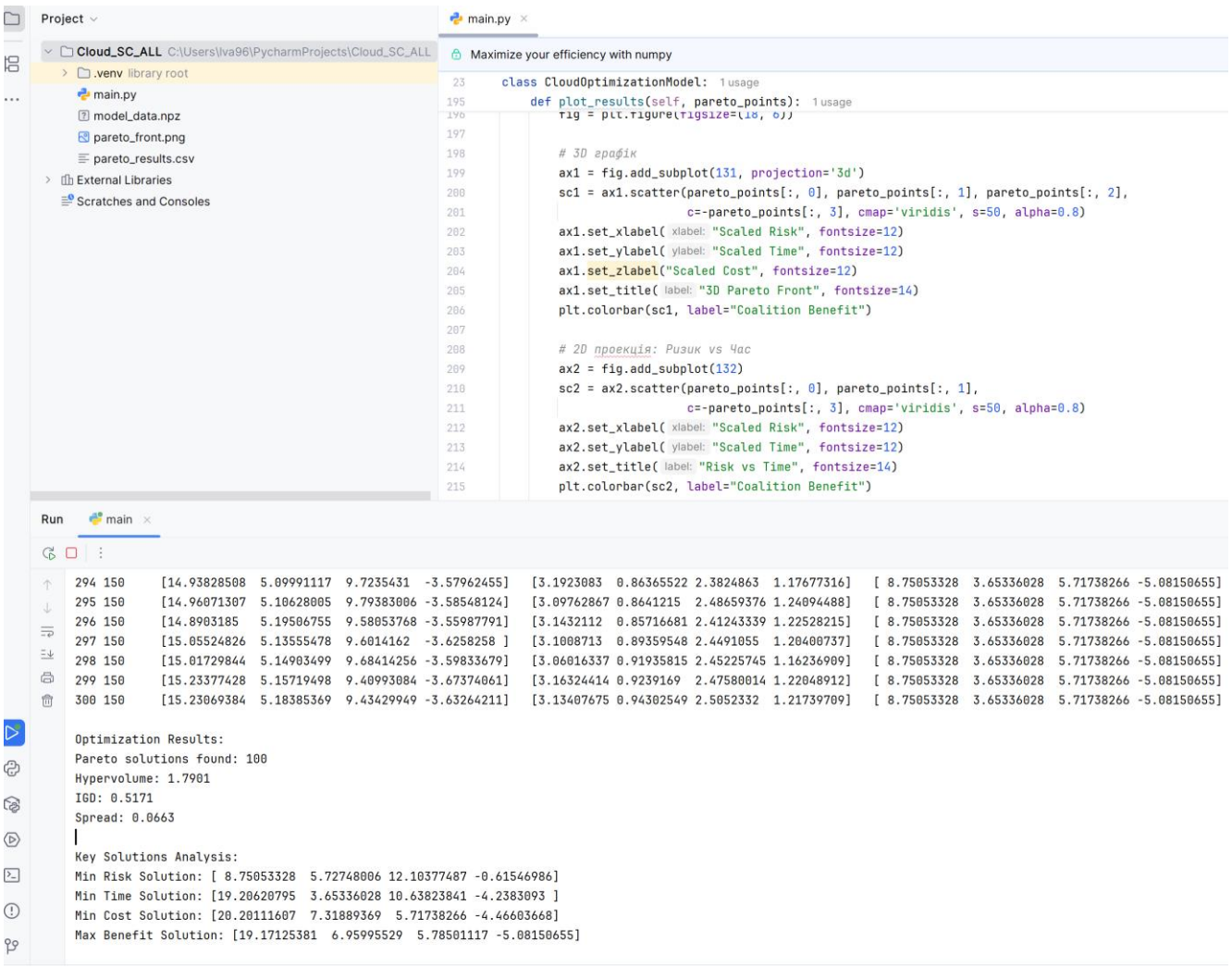


Рис. 2.5. Симуляція для запропонованої розширеної гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту ХС

Рівномірність розподілу розв’язків уздовж Парето-фронту оцінювалась за допомогою метрики Spread, яка показала значення 0,0663, див. рис. 2.5. Даний результат свідчить про відчутну диверсифікацію знайдених розв’язків. А це досить ґрунтовний аргумент на користь подальшого вибору оптимальної конфігурації захищеної ХС із мінімізованим рівнем ризику для безпеки. Рівномірний розподіл гарантує, що суб’єкт прийняття рішень матиме широкий вибір варіантів із різними балансами критеріїв.

Детальний аналіз головних розв’язків, візуалізованих на рис. 2.5 виявив певні закономірності. Рішення з мінімальним ризиком (8,7505) демонструє вагомий компроміс у вигляді підвищеної вартості (12,1038) та обмеженої користі від коаліції (0,6155).

На противагу цьому, варіант із мінімальним часом виконання (3,6534) супроводжується підвищеним рівнем ризику (19,2062). А це підкреслює фундаментальний потенційний конфлікт між продуктивністю та безпекою в хмарних системах. Також при симуляції одержано розв'язок із найбільшою вигодою коаліції (5,0815). Цей розв'язок реалізує синергію між складовими інфраструктури шляхом помірного збільшення ризику (19,1713) та часу опрацювання (6,9599).

Виявлені під час комп'ютерної симуляції компроміси зайвий раз підтверджують теоретичні передумови дослідження щодо неможливості одночасного досягнення оптимуму за всіма критеріями. Сильна негативна кореляція між рівнем ризику та іншими показниками наочно проявилася у варіантах з мінімальною вартістю (5,7174) та максимальною користю коаліції. У цьому випадку зменшення витрат супроводжувалося зростанням загроз безпеці.

Проведений обчислювальний експеримент заклав основу для подальших досліджень на реальному наборі даних. Де в рамках третього розділу дисертації буде виконана симуляція с даними використання реальних хмарних платформ.

Окрім виводу результатів моделювання у консолі IDE PyCharm, отримано декілька графічних результатів, які наведено на рис. 2.6–2.8.

На рис. 2.6 представлено візуалізацію результатів багатокритеріальної оптимізації розподілу ресурсів у ХС, отриманих за допомогою генетичного алгоритму NSGA-II із урахуванням сукупної вигода від коаліції (Coalition Benefit), який підлягав максимізації. Графік на рис. 2.6 демонструє фінальну множину недомінованих розв'язків, які утворюють так званий фронт Парето. Графічно це чотиривимірний простір цільових функцій, спроєктований на тривимірну діаграму розсіювання для наочного аналізу. Кожна точка на графіку рис. 2.6 відповідає одному оптимальному рішенню щодо розподілу обчислювальних завдань між доступними ресурсами.

Осі абсцис, ординат та аплікат представляють три основні критерії, що підлягали мінімізації:

масштабований ризик (Scaled Risk);

масштабований час виконання (Scaled Time);

масштабована вартість (Scaled Cost).

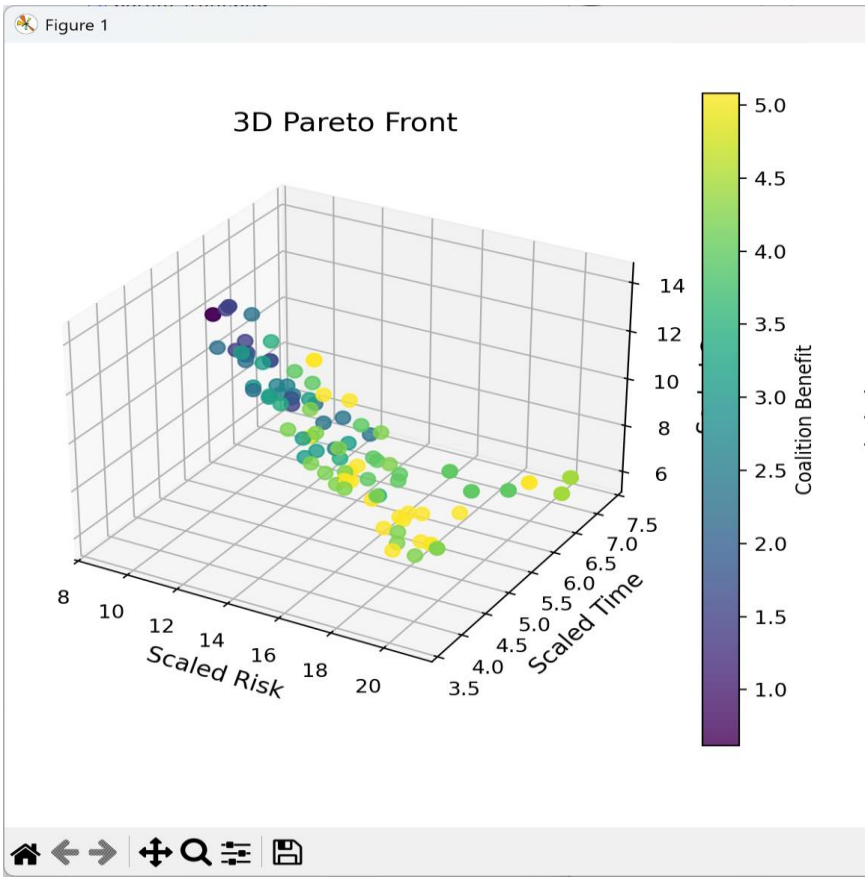


Рис. 2.6. Результати багатокритеріальної оптимізації: тривимірна поверхня фронту Парето

Четвертий критерій – сукупна вигода від коаліції (Coalition Benefit – який підлягав максимізації) візуалізовано за допомогою кольорової шкали. Тепліші кольори (жовтий) відповідають вищим значенням вигоди, тоді як холодніші (фіолетовий, синій) – нижчим.

Сукупність отриманих точок формує поверхню, яка наочно відповідає компромісу між конкуруючими цілями (критеріями). Розбір візуалізації показує, що рішення з мінімальними значеннями ризику, часу та вартості, розташовані ближче до початку координат. Ці точки характеризуються нижчою вигодою від коаліції (представлені синіми та зеленими точками). І навпаки, для досягнення максимальної вигоди від коаліційного об'єднання ресурсів захисту ХС (жовті точки) слід йти на поступки, погоджуючись на вищий рівень системного ризику

для ХС, довший час виконання завдань або більші фінансові витрати на захист. Отриманий фронт Парето надає особі, що приймає рішення (ОПР), набір оптимальних, але різних за своїми характеристиками альтернатив. Жодне з цих рішень не є найкращим за інше за всіма критеріями одночасно. Ці в результаті дозволяють ОПР гнучко обирати стратегію розподілу ресурсів ХС залежно від поточних пріоритетів. Отже це – мінімізація ризиків для критичних операцій. Чи максимізація продуктивності та економічної ефективності для стандартних завдань.

На рис. 2.7 подано двовимірну проекцію отриманого фронту Парето. Цей моделювання результат деталізує взаємозв'язок між двома критеріями оптимізації – масштабованим ризиком (Scaled Risk) та масштабованим часом (Scaled Time). Окремі маркери на рис. 2.7 відображають недоміновані конфігурації розподілу ресурсів ХС.

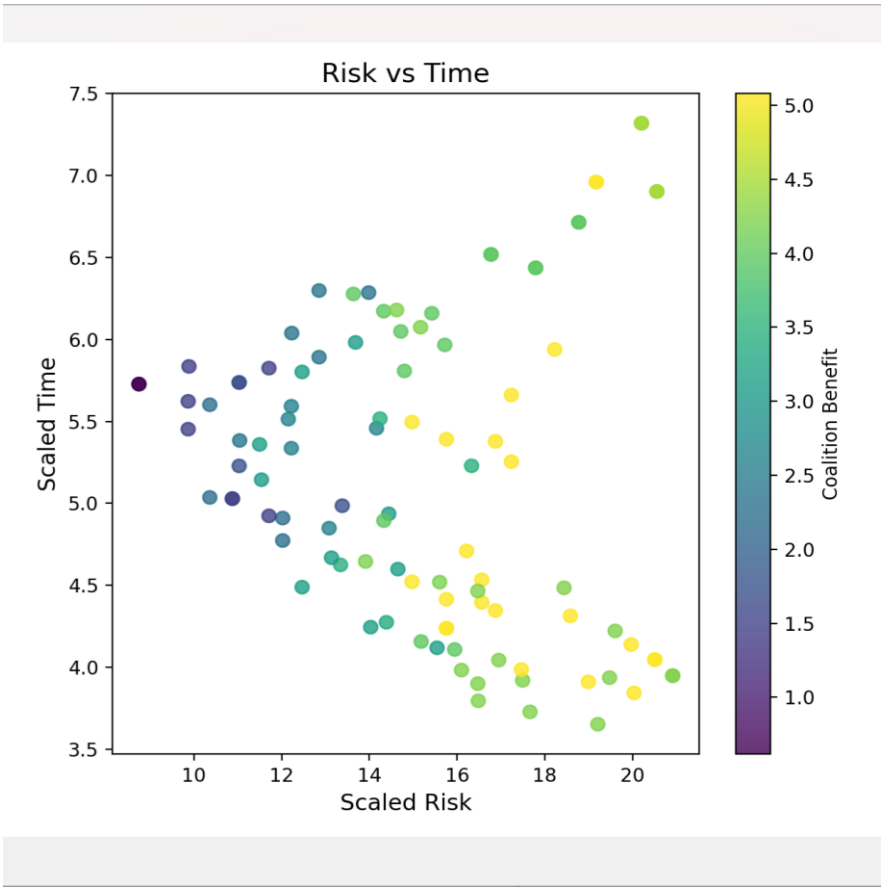


Рис. 2.7. Двовимірна проекція фронту Парето (компроміс між ризиком та часом на виконання завдань)

Третій суттєвий вимір, вигода від коаліції (Coalition Benefit), інтегрований у візуалізацію через колірне кодування. Це зручно, бо дозволяє відразу оцінити додатковий параметр, не перевантажуючи графік. Як бачимо, діаграма на рис. 2.6 демонструє обернену залежність між ризиком та часом. Така ситуація є доволі типовою для багатокритеріальних задач подібного класу. Спостерігаємо виражену тенденцію – стратегії, що мінімізують час виконання завдань (розташовані в нижній частині графіка), які пов'язані з вищим рівнем сукупного ризику. Водночас найбільш безпечні конфігурації ХС, що відповідають мінімальним значенням ризику, вимагають більших на 10–15% часових затрат на виконання.

Розбір колірного розподілу маркерів на рис. 2.7 надає ОПР глибше розуміння складних взаємозв’язків розглянутих у моделі (2.27)–(2.36) критеріїв $F_1 - F_4$. Найбільш вигідні з точки зору коаліційної взаємодії рішення (жовті маркери) переважно концентруються в області підвищеного ризику. Але з помірним або низьким часом виконання завдань у ХС. Натомість, найменш вигідні рішення (фіолетові маркери) відповідають найбезпечнішим, але водночас і найповільнішим варіантам розподілу. Отримані при симуляції результати показують антагоністичні особливості досліджуваних критеріїв $F_1 - F_4$ і слугують практичним інструментом для ОПР.

Рис. 2.8 розкриває співвідношення між економічними та безпековими аспектами оптимізації, фокусуючись на критеріях масштабованого ризику (Scaled Risk) та масштабованої вартості (Scaled Cost).

Кожен елемент отриманої множини недомінованих рішень зображений на рис. 2.8 як точка на двовимірній площині. Застосування колірної градієнтної заливки для параметра «вигода від коаліції» (Coalition Benefit) дозволяє ОПР одночасно оцінювати й третій вимір ефективності.

Рис. 2.9 представляє пряму залежність між цільовим показником ефективності – коаліційною вигодою (Coalition Benefit) та ключовим обмежуючим фактором – масштабованим ризиком (Scaled Risk).

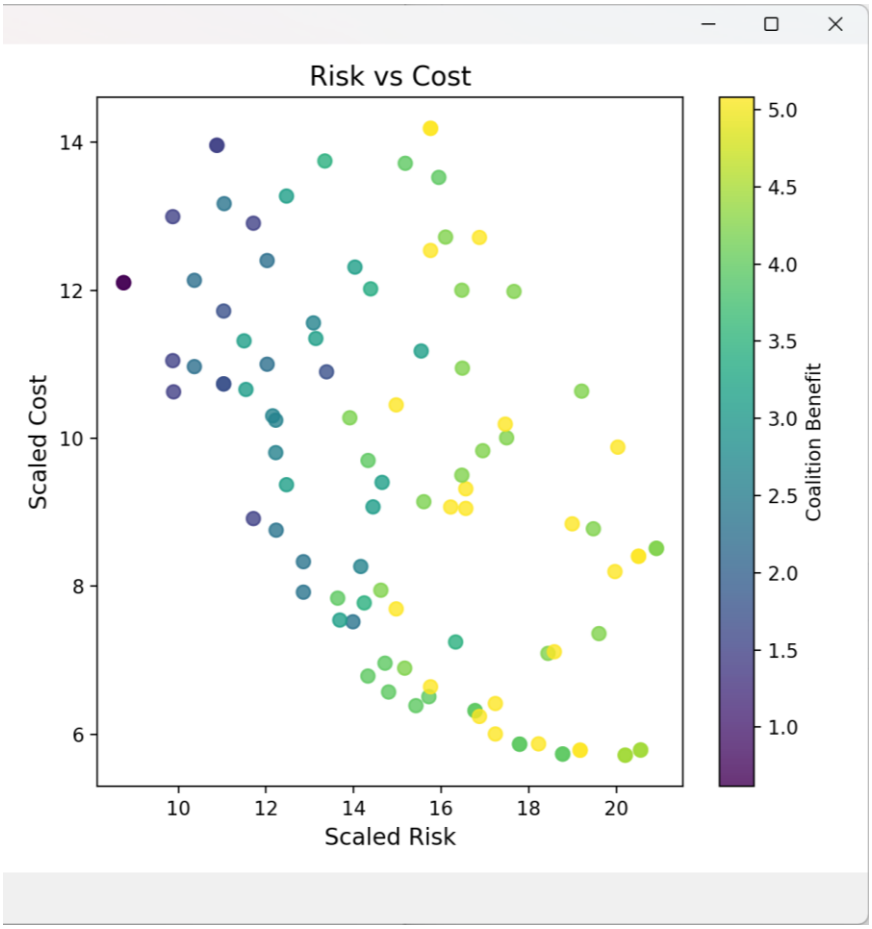


Рис. 2.8. Співвідношення між ризиком та вартістю в просторі оптимальних рішень для XC

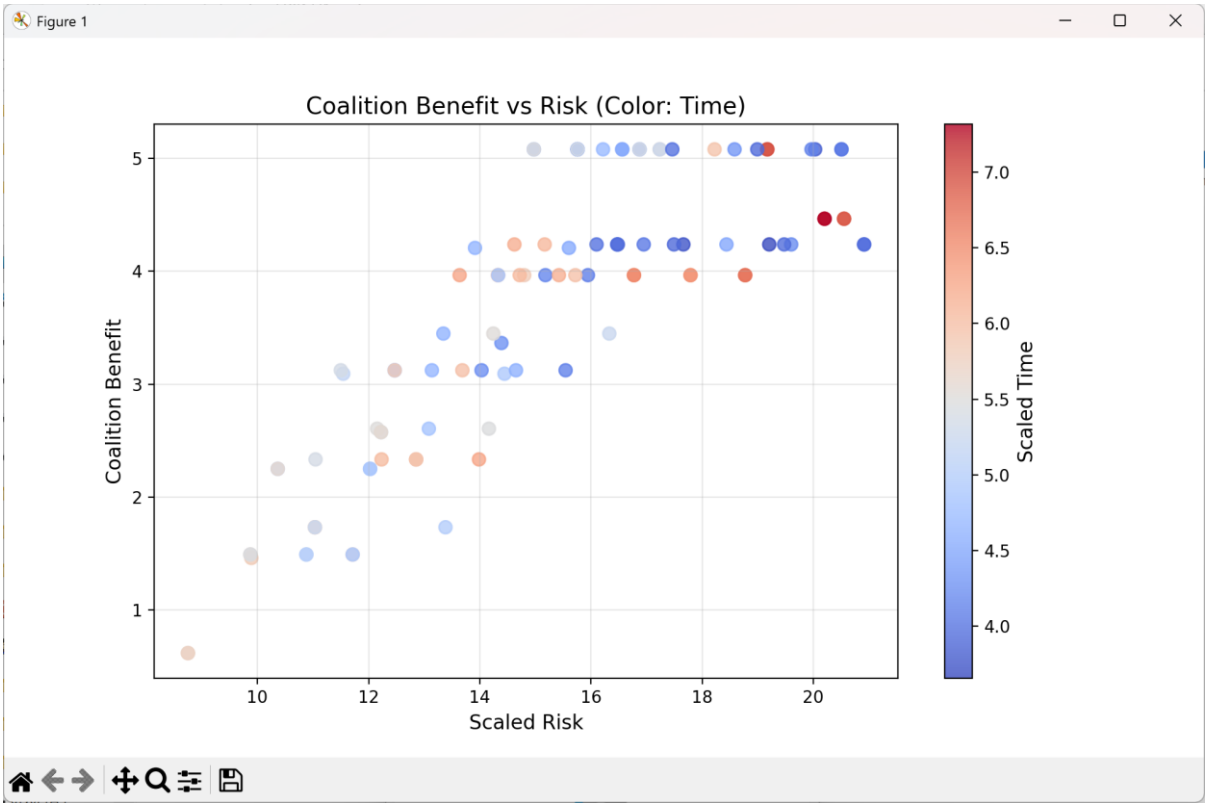


Рис. 2.9. Залежність коаліційної вигоди від ризику з урахуванням часових витрат

На відміну від попередніх графіків, ця діаграма ставить у центр аналізу результатів нашої комп'ютерної симуляції саме кінцеву вигоду від оптимізації розподілу завдані у ХС із урахуванням мінімізації ризику. Третій критерій, масштабований час (Scaled Time), інтегрований як додатковий інформаційний шар за допомогою колірної палітри, де сині відтінки відповідають меншим часовим витратам, а червоні – більшим.

Графічна картина наведена на рис. 2.9 на свідчить про наявність вираженої позитивної кореляції між двома основними осями. Зростання рівня прийнятого ризику зумовлює чітку тенденцію до збільшення сукупної вигоди від коаліції. Водночас відзначимо, що ця залежність не є лінійною. Тобто приріст вигоди є швидким на початкових рівнях ризику, після чого спостерігаємо ефект насичення, де подальше збільшення ризику дає все менший приріст вигоди від коаліції захисту ХС. Інформація, що передано кольором, вносить суттєві нюанси в цю картину. Вона демонструє, що досягнення максимальних значень коаліційної вигоди (верхній правий кут) є «найдорожчим» не лише з точки зору ризику, але й часу.

Це підтверджено концентрацією червоних точок у цій зоні. З іншої точки зору, найбільш швидкі рішення (сині точки) розподілені по всьому спектру ризиків. Але, як правило, таки результати не дозволяють досягти найвищих показників вигоди. Підкреслимо, що цей аналітичний зріз результатів нашої симуляції є підґрунтям для стратегічного планування безпеки ХС, адже він дозволяє ОПР виявити «точки перегину», за якими подальше прийняття ризику стає невиправданим через незначне зростання вигоди та суттєве збільшення часу виконання.

Додатково представлена картина дає ОПР можливість формулювати складні комбіновані стратегії. Скажімо, доцільно обирати найкраще за вигодою рішення в рамках припустимого часового лагу. Подібні комбіновані стратегії , суттєві для сервісів ХС із підвищеними вимогами до оперативності. Отже, у поточному розділу дисертації запропоновано новий метод – метод кооперативно-еволюційного розподілу обчислювальних ресурсів у ХС з урахуванням ризику.

Поданий у розділі метод поєднав в собі інструментарій багатокритеріальної еволюційної оптимізації (на підставі алгоритму NSGA-II) та кооперативної ігрової

моделі, і врахуванні ступеню ризику безпеки ХС у часі. На відміну від наявних підходів, відмінностями методу є формалізація компоненти ризику $\lambda(t)$ у моделі розподілу задач, введення критерію коаліційної вигоди як одного з об'єктів оптимізації, та підтримка кооперативних стратегій між ресурсами хмарного середовища. Запропонований метод дозволив не лише збалансувати критерії безпеки ХС, продуктивності та вартості, але й забезпечив пристосування до змінного ризикового профілю хмарного середовища.

Розроблений у розділі метод є узагальненням і розвитку класичних підходів до управління ресурсами в хмарних інфраструктурах. А його реалізація забезпечує гнучку архітектуру захисної поведінки. Запропоновані математичні моделі слід адаптувати до різних сценаріїв, притаманних для ХС та ІТ-систем в цілому.

У наступному розділі дисертації буде сформульовано узагальнений алгоритм реалізації запропонованого методу CoopEvo-CloudSec, та здійснено розгорнуте симуляційне моделювання поведінки системи на ґрунті даних підприємств різних сфер.

Крім того, буде розглянуто методику проведення обчислювальних експериментів, яка включатиме:

побудову тестових сценаріїв;

вибір параметрів симуляції;

використання метрик якості для оцінки результатів (вже розглянуті HV, IGD, Spacing).

Подобний підхід у сукупності дозволить оцінити комплексно результативність і універсальність методу CoopEvo-CloudSec в умовах для хмарних систем компаній та організацій [137].

Висновки до розділу 2

В результаті досліджень другого розділу дисертації зроблено наступні висновки та отримано такі наукові результати.

1. З’ясовано, що беручи до уваги рівень кіберзагроз для хмарних систем (ХС) і складність забезпечення балансу між продуктивністю, економічністю та безпекою в хмарних інфраструктурах (системах), з'явилась потреба у створенні нових підходів до керування обчислювальними ресурсами ХС, які здатні пристосовуватися до змін ризикового профілю. На підставі чинних викликів у сфері кібербезпеки хмарних систем, та обмежень традиційних моделей розподілу задач, у другому розділі запропоновано концепцію гібридної моделі, яка інтегрує підходи багатокритеріальної оптимізації, теорії ігор та адаптивного управління ризиком.

2. З урахуванням ознак поведінки атакуючого та захисника у хмарному середовищі, створено ігрову модель, що віддзеркалює конфліктну взаємодію між гравцями. В моделі захисник формує стратегії розподілу задач на вузли ХС, а атакуючий – вибирає цілі атак. У межах цієї взаємодії введено параметр агресивності $\lambda(t)$, який описує зміну інтенсивності атак у часі, що дозволило врахувати стохастичну, марковську або еволюційну природу ризику.

3. Відштовхуючись від підсумків ігрової моделі, сформовано функцію оцінки ризику для кожного елементу хмарної системи, що застосовувався як один із критеріїв у завданні багатокритеріальної оптимізації. З метою відшукання компромісних рішень вперше втілено багатокритеріальну оптимізацію за допомогою алгоритму NSGA-II з врахуванням чотирьох критеріїв – ризику, продуктивності, вартості та коаліційної вигоди між захисниками ХС. Це дозволило сформувати множину Парето-оптимальних рішень, що беруть до уваги складну багатофакторну природу задачі.

4. З метою збільшення адаптивності системи запропоновано механізм реалізації стратегії захисника, спроможної змінювати розподіл завдань у відповідь на зміну параметра ризику та виявлену активність атакуючого хмарне середовище. Такий метод забезпечив гнучкість у виборі найкращого варіанта реагування в умовах ХС. У моделі вперше, на відміну від наявних рішень, враховано можливість співпраці між захистками хмарних середовищ, що дозволило моделювати коаліційні сценарії протидії загрозам. При цьому введено поняття виграшу коаліції,

яке застосовувався для оцінки доцільності об’єднання та формування узгоджених стратегій.

5. На підставі об’єднання наведених у розділі моделей сформовано єдину гібридну модель, котру описано у вигляді псевдокоду та блок-схеми, що дає змогу забезпечити її відтворюваність і подальшу програмну реалізацію. Доведено, що на відміну від класичних статичних моделей розподілу задач, пропонований метод є гнучким, кооперативно орієнтованим і багатокритеріальним, що визначає його наукову новизну та прикладну цінність.

6. Відповідно до узагальнення результатів другого розділу сформульовано новий метод – метод кооперативно-еволюційного розподілу обчислювальних ресурсів у хмарному середовищі з урахуванням ризику (CoopEvo-CloudSec). У наступному розділі буде деталізовано його реалізацію, розроблено програмну модель симуляції, та проведено обчислювальний експеримент на базі даних, наближених до реальних умов функціонування ХС.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ МЕТОДА КООПЕРАТИВНО-ЕВОЛЮЦІЙНОГО РОЗПОДІЛУ РЕСУРСІВ У ХМАРНОМУ СЕРЕДОВИЩІ З УРАХУВАННЯМ РИЗИКУ

3.1. Методика проведення експериментального дослідження

Обчислювальний експеримент (далі по тексту роботи – ОбЕк) проведено з метою забезпечити об’єктивну та відтворювану оцінку ефективності запропонованого в другому розділі роботи методу кооперативно-еволюційного розподілу обчислювальних ресурсів хмарної системи в умовах варіативності ризикового профілю ХМС.

Методика проведення ОбЕк ґрунтувалася на відтворенні в IDE PyCharm контрольованого симуляційного середовища роботи вузлів ХМС. На першому етапі сформовано репрезентативний набір даних, див. табл. 3.1–3.3. Ці датасети моделювали структуру умовної хмарної інфраструктури відповідно до методу CoopEvo-CloudSec. Зокрема в датасетах враховано характеристики вузлів ХМС, їхні обчислювальні можливості, параметри вартості, базові ризикові профілі та участь у коаліційній взаємодії. Паралельно під час проведення симуляцій моделюємо задачі ХМС, які віддзеркалювали типові робочі навантаження вузлів ХМС із різними вимогами до ресурсів, пріоритетами та рівнями безпеки. Тобто для проведення ОбЕк сформовано три основні набори даних. Дані відображали основні компоненти хмарної інфраструктури та робочих навантажень. Файл nodes.csv, див. табл. 3.1 описує характеристики обчислювальних вузлів. Файл tasks.csv, див. табл. 3.2 містив параметри задач ХМС. Файл risk_timeline.csv, див. табл. 3.3 віддзеркалив зміни ризикового профілю ХМС. Структура, числові параметри та взаємозв’язки між цими даними в датасет розроблені з участю експертів так, щоб максимізувати симуляційне середовище до реальних умов функціонування ХМС.

За основу взято параметри популярних ХМС, зокрема, Amazon Web Services, Microsoft Azure та Google Cloud Platform.

Таблиця 3.1

Характеристики обчислювальних вузлів (nodes)

node_id	cpu_cores	memory_gb	base_speed	cost_per_hour	base_risk	risk_volatility	coalition_role
node_1	8	32	1,926	0,874	0,170	0,089	0
node_2	8	16	1,799	0,416	0,192	0,055	1
node_3	16	8	0,773	0,349	0,111	0,181	0
node_4	2	16	1,418	0,086	0,108	0,142	0
node_5	8	16	1,074	0,477	0,143	0,265	1
node_6	2	8	1,923	0,059	0,212	0,126	0
node_7	16	32	1,160	1,358	0,149	0,059	1
node_8	16	8	1,494	0,370	0,154	0,187	0
node_9	8	8	1,663	0,235	0,229	0,199	1
node_10	16	32	0,568	1,488	0,128	0,118	1

Дані у файлі nodes.csv, див. табл. 3.1 моделювали середовище хмарних віртуальних машин (далі ВМ). Кожен вузол містить параметри продуктивності, ресурсної ємності, вартості та ризикового профілю. Зокрема, параметри «cpu_cores» та «memory_gb» узгоджено зі типовими конфігураціями ВМ у провайдерів AWS EC2 (класи t3, m5, c5), Azure VMs (серії B, D, F) та Google Compute Engine (типи e2-standard, n2-highcpu). Діапазони обчислювальної потужності та обсягу пам’яті обрано експертно на підставі відкритої документації. Параметр «base_speed», який відображав продуктивність вузла ХМС, інтерпретуємо як коефіцієнт відносної швидкодії, апроксимований за даними SPECint [138], GeekBench [139] та дослідженнями продуктивності різних поколінь CPU у хмарних платформах [140, 141].

Показник вартості «cost_per_hour» приведено у відповідність до середніх тарифів на використання ВМ у публічних хмарах. Вартість зафіксована у прайс-листах на провайдерів, які надають доступ до ХМС. Параметри «base_risk» і «risk_volatility» відображали рівень технічного та кіберризику, притаманного конкретним інфраструктурним ресурсам, включно з імовірністю деградації продуктивності, уразливостей ІБ або варіацій навантаження ХМС.

Нарешті, параметр «coalition_role» моделював участь обчислювального вузла (node №) у коаліції захисників. Саме це відповідає концепції методу CoopEvo-CloudSec для завдання спільного використання захисних політик у розподілених хмарних середовищах.

Файл tasks.csv, див. табл. 3.2 містив опис задач, які відображають робоче навантаження, притаманне ХМС. Зокрема, це вебзапити, аналітичні завдання, опрацювання потокових даних, мікросервісні виконання тощо.

Таблиця 3.2

Опис задач ХМС (показані перші рядки)

task_id	required_cpu	required_memory	base_duration	priority	security_level
task_1	1	1	423	2	1
task_2	4	4	59	4	1
task_3	2	8	47	4	3
task_4	1	4	40	5	3
task_5	1	1	1185	4	3
task_6	2	8	32	5	4
task_7	4	8	1488	3	2
task_8	1	8	481	5	2
task_9	4	32	558	2	2

Параметри «required_cpu» та «required_memory» моделювали ресурси, необхідні для виконання задачі, у відповідності до профілів використання CPU та RAM. Останні зафіксовано у роботах з аналізу навантажень AWS Lambda, Google Cloud Functions та контейнеризованих сервісів Kubernetes [142–145]. Параметр «base_duration» в табл. 3.2 представлений у секундах. Значення «base_duration» узгоджено із типовими часовими характеристиками обробки транзакцій або коротких обчислювальних процесів при роботі на віртуальних ВМ середнього класу. Параметр «Priority» відображав під час симуляції важливість задачі у межах сервісного рівня (SLA). Це потрібно, оскільки відповідало моделі диференційованого обслуговування на хмарних платформах. Показник «security_level» описував ступінь чутливості задачі до ризиків ІБ. Значення «security_level» прийнято після узгодження із експертами, які спеціалізуються саме на захисті у публічних хмарах.

Дані у наборі risk_timeline.csv, див. табл. 3.3, містили зміни індикатора ризику $\lambda(t)$. Зміна відображена в risk_timeline.csv для кожного обчислювального вузла (node №) впродовж дискретних часових інтервалів. Така варіативність ризиків узгоджена з висновками, наданими в [146–148], де автори дослідили показники нестабільності навантаження, появу вразливостей, зміни умов безпеки, а також часових флуктуацій у доступності ХМС.

Таблиця 3.3

Зміни індикатора ризику $\lambda(t)$ для вузлів хмарної системи (показані перші рядки)

time_step	node_id	lambda_t
0	node_1	0,170
0	node_2	0,192
0	node_3	0,111
…	…	…
…	…	…
50	node_10	0,128

Відповідно до [146–148] використання дискретизованої часової серії $\lambda(t)$ відповідало практиці моделювання ризику в системах виявлення аномалій ХМС. Зміни $\lambda(t)$ відображали таки сценарії:

різке зростання загроз, типове для періодів DDoS-атак;

випадкові флуктуації, властиві довготривалим сервісам;

стабільні інтервали низького ризику, характерні для періодів низької активності користувачів в ХМС.

Усі три набори даних, фрагменти яких подано в табл. 3.1–3.3, є взаємно узгодженими. Отже задачі з високим рівнем security_level призводили в ОбЕк до зростання значущості функції $\lambda(t)$ для конкретного вузла. Значення ризику в файлі risk_timeline.csv безпосередньо впливало на критерій мінімізації ризику в оптимізаційній моделі Розділу 2. А параметри продуктивності, вартості та участі у коаліції в nodes.csv визначали міру узгодження окремих параметрів ХМС при оцінюванні альтернативних рішень. Узгоджене використання цих даних забезпечило під час обчислювального експерименту реалізм симуляції. Це

дозволило за допомогою IDE PyCharm відтворити багатофакторне середовище, яке корелювалося з методикою досліджень в роботах інших авторів.

Окремо під час ОбЕк будуємо часову шкалу ризику. Ця шкала віддзеркалює зміну індикатора $\lambda(t)$, див. залежність (2.27), відповідно до гіпотетичних сценаріїв поведінки загроз для ХМС, з урахуванням варіативності ризикових профілів окремих вузлів.

Після формування початкових даних відповідно до суджень другого розділу побудовано оптимізаційну модель. Модель визначала співвідношення між чотирма критеріями – $F_1 - F_4$, відповідно, рівень безпеки ХМС, загальний часом обробки завдань, вартість використання ресурсів та інтегральна коаліційна вигодою. Функцію ризику $\lambda(t)$ враховуємо через відповідність часу старту кожної задачі дискретному часовому кроку ризикового профілю відповідного вузла ХМС. Така логіка ОбЕк дала змогу промоделювати накопичений вплив загроз у процесі опрацювання задач ХМС. Відповідно до концепції методу CoopEvo-CloudSec коаліційну взаємодію ресурсів враховано шляхом оцінювання кооперативної вигоди, яка виникає за умов спільної участі вузлів ХМС у коаліції захисників та відповідає інтересам підвищення стійкості хмарної інфраструктури до атак. Відповідно, в ОбЕк кількісна модель коаліційної вигоди ґрунтувалася на поєднанні характеристик залучених вузлів ХМС, їхньої кількості та рівнів завдань, які вони опрацьовують.

Для проведення ОбЕк використовувалися два типи даних. На першому етапі для тестування працездатності відповідного програмного застосунку (див. Додатки А та Б), який реалізовує метод CoopEvo-CloudSec використовувалися синтетичні дані у датасет. На наступному етапі використано реальні дані, отримані від хмарних платформ базового підприємства (див. Додаток Г). Зазначимо, синтетичні дані дозволили на першому етапі ОбЕк контролювати параметри системи. Як показано в табл. 3.4, параметри для ОбЕк сформовано з використанням рівномірних розподілі. Реальні дані включали логи роботи ХМС, які містили інформацію про навантаження, інциденти безпеки та витрати на ХМС [149]. У підсумку це

дозволило оцінити придатність запропонованого методу CoopEvo-CloudSec в близьких до експлуатаційних умовах.

Таблиця 3.4

Характеристики синтетичних та реальних даних, використаних в обчислювальному експерименті (складено автором)

Категорія параметрів	Діапазон даних в синтетичному датасет	Реальні дані, отримані на базовому підприємстві
Кількість задач	10–100 (змінна)	50–500 задач (логи планування AWS, Google Cloud)
Кількість вузлів ХМС	5–20	10–100 вузлів (кластери Kubernetes)
Рівень ризику вузла	0,1–0,9 (рівномірний розподіл)	Експертно розрахований фахівцями підприємства на основі історії інцидентів (CVSS-базована оцінка вразливостей)
Час обробки задачі	1–50 умовних одиниць (нормальний розподіл)	Фактичний час виконання в мілісекундах (логовані метрики з хмарних моніторингових систем підприємства)
Вартість виконання	5–100 умовних одиниць (рівномірний розподіл)	Вартість за одиницю часу CPU/GPU (тарифи AWS EC2, Azure VMs). Дані отримано на підприємстві
Користь від коаліції	0,2–3,0 (залежить від типу вузла)	Оцінена на основі ефективності спільних заходів безпеки в історії інцидентів
Динаміка ризику $\lambda(t)$	Стохастична та еволюційна моделі	Реконструкція на основі часових рядів атак (дані отримано з SIEM-системи підприємства)

Для розв’язання задачі багатокритеріальної оптимізації, відповідно до блок-схеми алгоритму, див. рис. 2.4, для розширеної гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту ХМС, застосовано алгоритм NSGA-II. На відміну від чинних варіацій, алгоритм NSGA-II модифіковано шляхом інтеграції коаліційної моделі та компоненти ризику (див. модель (2.27)–(2.36)). Реалізація алгоритму передбачала використання цілочисельного кодування рішень. Тобто, кожна задача відображалася у вигляді індексу вузла, якому вона призначена. Покоління рішень генеруємо на основі цілочисельних операторів ініціалізації, схрещування та мутації. На кожному кроці еволюційного процесу під час ОбЕк, обчислюємо значення усіх чотирьох критеріїв $F_1 - F_4$. А далі виконуємо ранжування розв’язків за Парето-принципом та мірою скупченості за для забезпечення рівномірності формування фронту.

Обчислювальний експеримент проведено у декілька етапів. Кожен етап мав на меті відокремлене дослідження впливу окремих компонент запропонованого методу CoopEvo-CloudSec. Спочатку оцінювалися результати роботи чистого алгоритму NSGA-II без врахування ризикової та коаліційної складових. Це

забезпечило можливість сформувати базовий еталон. На наступному етапі модель NSGA-II доповнімо компоненою ризику з фіксованим λ. Цей крок дозволив оцінити вплив самого факту включення ризику до критеріїв оптимізації. Завершальний етап передбачав повну активацію гібридного механізму, у якому варіативність ризику $\lambda(t)$ та кооперативна взаємодія між вузлами визначили у підсумку поведінку гібридного алгоритму, відповідно до моделі (2.27)–(2.36). Для кожного з режимів оптимізації будуємо Парето-фронти та зберігаємо у файлах формату CSV повні множини рішень для подальшого аналізу. Для забезпечення об'єктивності оцінювання результатів під час ОбЕк використовувалися кількісні метрики якості багатокритеріальних рішень. До цих метрик входили гіпероб'єм, показник зворотної генераційної відстані та метрика рівномірності розподілу розв'язків, див. табл. 3.5.

Таблиця 3.5

Метрики якості для оцінки Парето-оптимальних рішень

(перелік метрик складено відповідно до рекомендацій [119, 121, 123])

Метрика	Формальне визначення
Гіпероб'єм (HV)	Об'єм простору критеріїв, домінований отриманим фронтом Парето щодо заданої опорної точки.
Зворотна генераційна відстань (IGD)	Середня відстань від точок референсного фронту до найближчих точок отриманого фронту.
Рівномірність розподілу (Spacing)	Стандартне відхилення відстаней між сусідніми рішеннями на фронті

Застосування цих метрик дає змогу оцінити повноту наближення фронту рішень до теоретично оптимальної області, ступінь детермінованості та різноманіття знайдених компромісних рішень для критеріїв $F_1 - F_4$. Додатково ці метрики визначали для кожного ОбЕк наскільки рівномірно гібридний алгоритм, див. блок-схему 2.4, заповнював область Парето. Отже, подальше порівняння отриманих метрик та результатів із базовим варіантом NSGA-II дало можливість простежити, яким чином доповнення алгоритму факторами коаліційних стратегій та ризикового профілю змінило якість прийнятих рішень. Тобто, завдяки викладеній методиці ОбЕк у підсумку сформована уява про властивості запропонованого методу CoopEvo-CloudSec. Також експериментальні результати створили підґрунтя для подальшого порівняння запропонованого в роботі методу

CoopEvo-CloudSec з альтернативними алгоритмами багатокритеріальної оптимізації, зокрема NSGA-III. Таке порівняння методів дозволило нам далі обґрунтувати наукову новизну та практичну цінність розробленого у дисертації методу CoopEvo-CloudSec.

Основні параметри симуляції згідно моделі (2.27)–(2.36), містили, див. табл. 3.6, розмір популяції (100 особин), кількість поколінь (300 ітерацій), ймовірність схрещування (0,85) та ймовірність мутації (0,15). Для моделювання зміни ризику для вузлів ХМС використовувався параметр $\lambda(t)$. У табл. 3.4, відповідно, наведені діапазони значень для опрацьованих в ОбЕк даних, які забезпечили моделювання різних сценаріїв.

Таблиця 3.6

Конфігураційні параметри симуляції моделі (2.27)–(2.36) та алгоритму NSGA-II

Параметр симуляції	Значення / діапазон	Призначення та вплив на експеримент
Розмір популяції (P)	100 осіб	Визначає кількість кандидатних рішень у кожному поколінні. Більший розмір покращує дослідження простору, але збільшує час обчислень
Кількість поколінь (G)	300 ітерацій	Максимальна кількість циклів еволюції. Забезпечує достатній час для збіжності алгоритму
Ймовірність схрещування	0,85	Імовірність застосування оператора схрещування для створення нових рішень. Високе значення сприяє дослідженню
Ймовірність мутації	0,5	Імовірність випадкових змін у рішеннях. Запобігає застряганню в локальних оптимумах
Тип селекції	Турнірна селекція з розміром турніру 3	Механізм вибору батьків для схрещування. Турнірна селекція підтримує узгодженість між тиском відбору та різноманітністю
Оператор схрещування	SBX – Simulated Binary Crossover	Забезпечує створення дітей у близькій околиці батьківських рішень, зберігаючи структуру простору
Оператор мутації	Поліноміальна мутація	Вносить контрольовані випадкові зміни, що допомагає досліджувати нові області простору рішень
Модель динаміки $\lambda(t)$	Стохастична, еволюційна	Відображає різні типи поведінки атакуючого. Дозволяє оцінити адаптивність системи до різних сценаріїв загроз
Коефіцієнти ваг w_1, w_2, w_3	Визначалися експериментально на основі чутливості	Вагові коефіцієнти для функції корисності захисника (2.29). Впливали на узгодженість між продуктивністю ХМС, ризиком та вартістю

Для кооперативних сценаріїв використовувалася модель розрахунку виграшу коаліції на основі значення Шеплі [150]. Це дозволило оцінити внесок кожного учасника до спільного захисту ХМС. Отримані під час ОбЕк результати наведено у наступному параграфі роботи.

3.2. Результати експериментального дослідження

В цьому параграфі наведено описовий аналіз основних результатів обчислювальних експериментів для методу CoopEvo-CloudSec.

Радарна діаграма, наведена на рис. 3.1, дозволила виконати порівняльний аналіз декількох оптимальних рішень фронту Парето, які отримано за допомогою запропонованого в другому розділі роботи методу CoopEvo-CloudSec. Відповідно для чотирьох критеріїв ($F_1 - F_4$): ризик, час виконання, вартість та рівень коаліційної взаємодії.

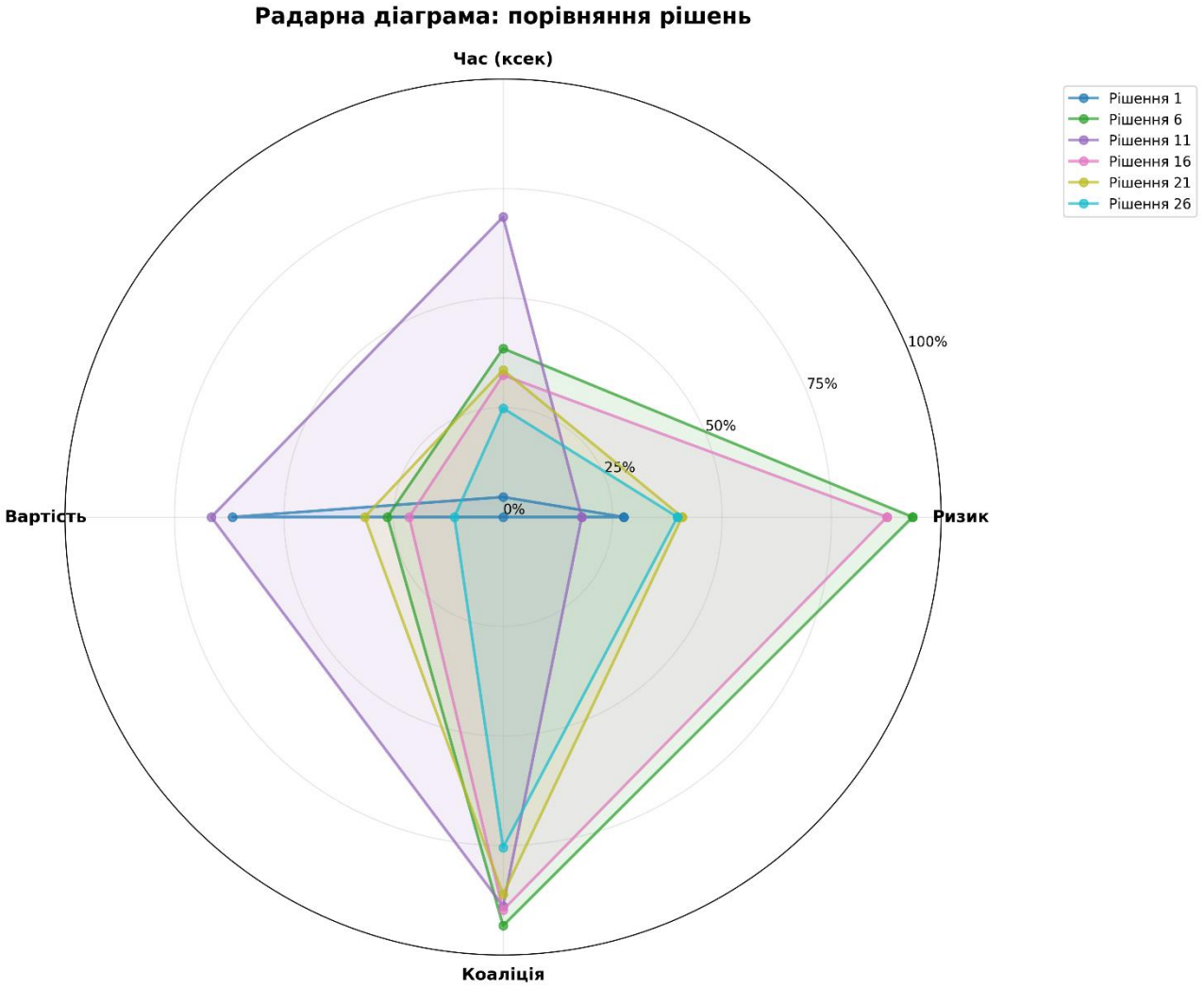


Рис. 3.1. Радарна діаграма для порівняльного аналізу оптимальних рішень фронту Парето

Візуалізація результатів на рис. 3.1 зафіксувала різновекторність компромісів між критеріями $F_1 - F_4$. На підставі аналізу рис. 3.1 констатуємо, що знайдені рішення мали виражену багатомірну природу. Рішення 16, яке на діаграмі сформувало найбільшу площу, віддзеркалило сценарій, у якому пріоритет зроблено на безпеці ХМС. Тобто рішення 16 характеризувалося високими значеннями коаліційної взаємодії захисних засобів ХМС. Таке рішення притаманне для ситуації активного реагування на загрози безпеці ХМС. Висока коаліційність у цьому рішенні засвідчила ефективність та працездатність розширеної гібридної моделі з урахуванням ризиків та кооперативних стратегій захисту хмарного середовища (див. п. 2.2).

Рішення 6 та 11 показали різні форми компромісів для $F_1 - F_4$. Рішення 6 відзначилося гранично високою коаліцією в задачі захисту ХМС та мінімальним часом. Рішення 11 віддзеркалило поєднання середніх значень ризику, часу та вартості ХМС.

Рішення 1 та 26 мали меншу площу на діаграмі 3.1. Ці рішення – приклади реалізації економних стратегій, орієнтованих на мінімізацію витрат на підтримку працездатності ХМС. Відповідно рішення 1 і 26 мали менший рівень коаліційності в безпекових аспектах роботи ХМС та часу виконання завдань. Проте вони забезпечували кращі значення вартості порівняно з іншими рішеннями. Зокрема рішеннями 6 та 11 орієнтованими саме на максимізацію захисту ХМС. Це вказує на здатність методу CoopEvo-CloudSec формувати індивідуальні рішення для сценаріїв, де економічні або безпекові обмеження є домінантними.

Відмітимо, що форма та взаємне розташування профілів рішень на діаграмі 3.11 підтвердила відсутність домінування одного критерію над всіма іншими. Тобто рішення на радарній діаграмі для задіяних в обчислювальних експериментах наборах даних (див. датасет в табл. 3.1–3.3) розподілені в межах багатовимірного фронту. Тому отримані результати на 3.1 довели коректність роботи алгоритму NSGA-II у межах методу CoopEvo-CloudSec. Крім того, виражена різниця між рішеннями підтвердила адекватність урахування варіативного ризику *λ(t)*, оскільки рішення, які мали високі ризики, одночасно демонстрували підсилену коаліційну

поведінку для забезпечення захисту ХМС. Резюмуючи аналіз результатів на радарній діаграмі, зазначимо, що води у сукупності підтвердили ефективність методу CoopEvo-CloudSec у формуванні множини компромісних рішень між критеріями $F_1 - F_4$. Отримані результати засвідчили той факт, що метод CoopEvo-CloudSec здатен пристосовуватися до різних профілів загроз ХМС та здатен знаходити стійкі рішення для ХМС із підвищеними вимогами до кібербезпеки.

Для того щоб підтвердити ефективність методу CoopEvo-CloudSec проведені тестові експерименти для альтернативних варіантів розв'язку завдання пошуку оптимального рішення. А саме розглянуто наступні моделі (методи), див. рис. 3.2–3.7, табл. 3.6–3.11 (код експериментальної частини для порівняння різних методів наведено в Додатку Б):

метод CoopEvo-CloudSec (в табл. 3.6–3.11 та на рис. 3.2–3.7 позначено як – V1_Hybrid_NSGA-II);

простий варіант реалізації моделі NSGA-II для 2–3 критеріїв ($F_1 - F_3$) не враховуючи (F_4), відповідно позначимо на рис. 3.2–3.7, табл. 3.6–3.11, як V2_Simple_NSGA-II;

варіант коли рівень ризику ІБ для ХМС задано статичним – V3_StaticRisk_NSGA-II;

в методі CoopEvo-CloudSec заміняємо алгоритм NSGA-II на NSGA-III. Позначимо як – V4_Hybrid_NSGA-III.

Порівняння Парето фронтів для розглянутих варіантів розв'язку задачі з наведено на рис. 3.2. А також у табл. 3.6–3.11.

В табл. 3.7 наведені середні значення метрик якості для різних методів та алгоритмів, які тестувалися для розв'язку завдань дослідження.

Таблиця 3.7

Середні значення метрик якості для різних методів та алгоритмів для розв'язку завдань дослідження

Алгоритм (метод)	Цілі	Час, с	Hypervolume	Spacing	Diversity	Розмір фронту
V1_Hybrid_NSGA-II	4	60,05	0,89 ± 0,120	0,0600 ± 0,0160	1,230 ± 0,040	19,33 ± 2,52
V2_Simple_NSGA-II	2	0,39	0,88 ± 0,004	0,0040 ± 0,0004	0,268 ± 0,040	50,00 ± 0,01
V3_StaticRisk_NSGA-II	3	45,80	0,91 ± 0,040	0,0280 ± 0,0036	0,680 ± 0,050	50,00 ± 0,01
V4_Hybrid_NSGA-III	4	550,65	1,11 ± 0,050	0,0537 ± 0,0025	1,096 ± 0,005	43,00 ± 6,24

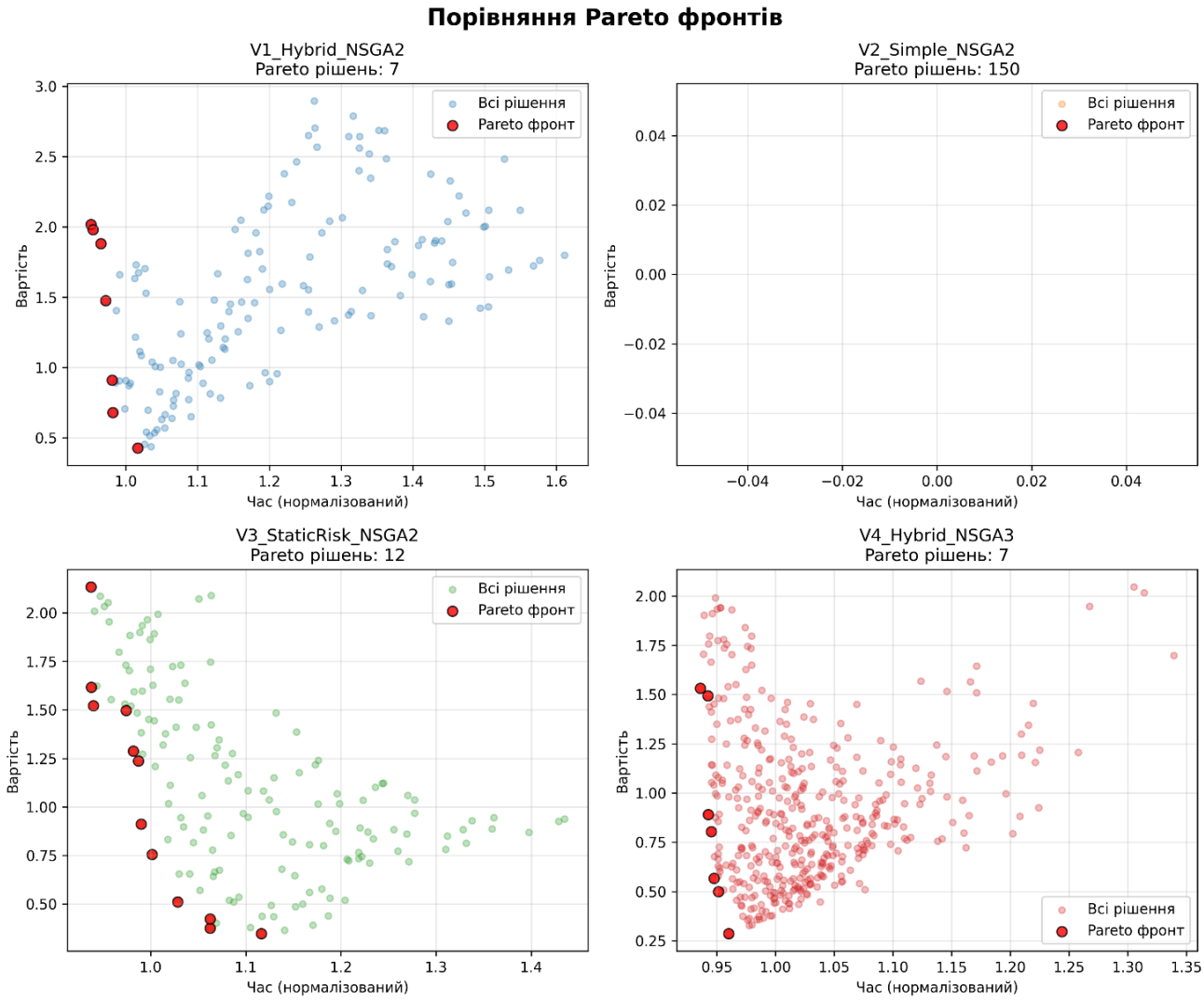


Рис. 3.2. Порівняння різних методів та алгоритмів для розв’язку завдань дослідження

В табл. 3.7 представлені середні значення метрик якості, зокрема, нормалізований гіпероб’єм, який є комплексним показником якості Парето-фронту. Отримані під час ОбЕк результати показали значні відмінності між алгоритмами. Так алгоритм V4_Hybrid_NSGA-III показав найвищі значення нормалізованого гіпероб’єму (1,1145). Отже саме варіант коли в методі CoopEvo-CloudSec заміняємо алгоритм NSGA-II на NSGA-III, засвідчив значно кращу якість Парето-фронту порівняно з іншими алгоритмами. Проте ця перевага досягнута ціною значного збільшення часу виконання. А саме у 9,2 рази більше порівняно з V1_Hybrid_NSGA-II та у 1412 разів порівняно з V2_Simple_NSGA-II.

На рис. 3.3 подано результати порівняльного аналізу середніх значень метрик якості рішення по розглянутих варіантах V1–V4.

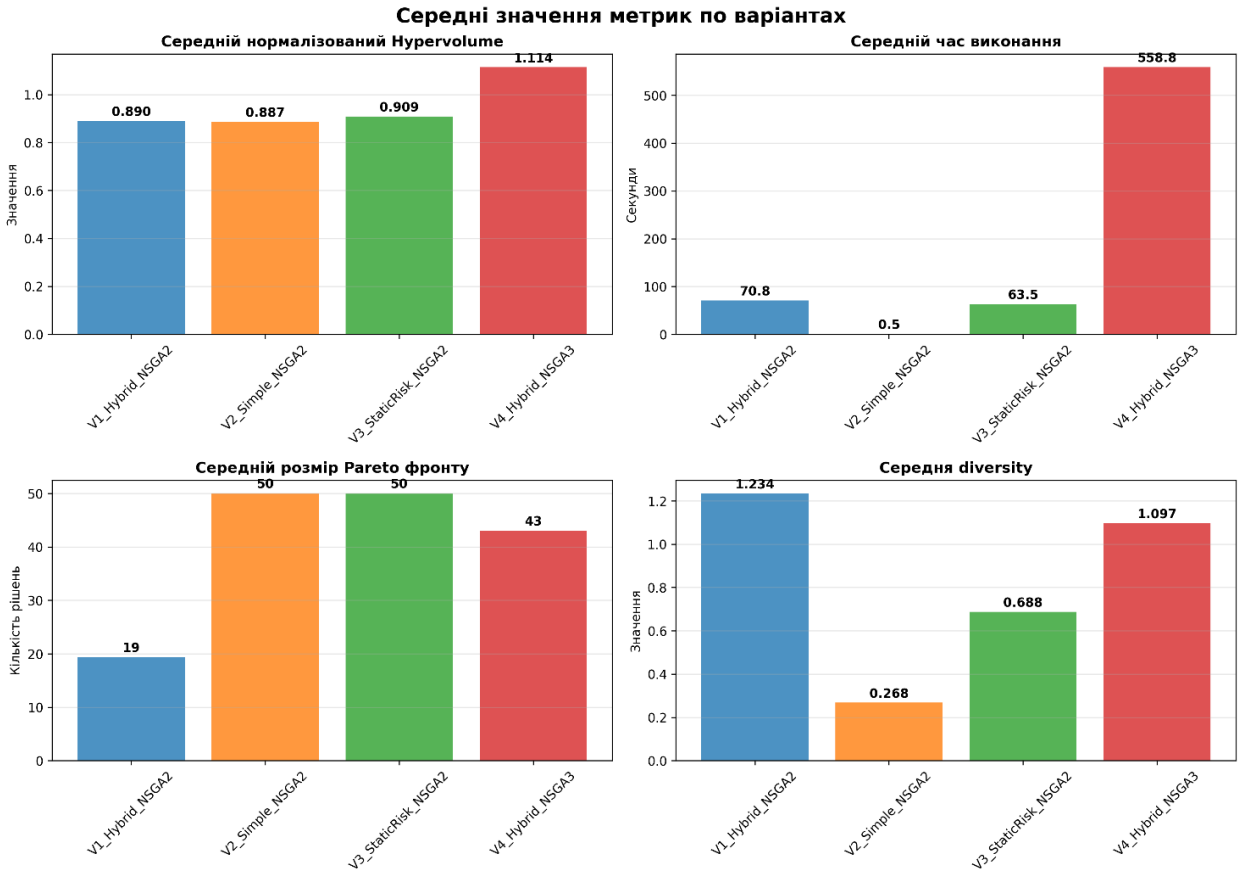


Рис. 3.3. Результати порівняльного аналізу середніх значень метрик якості рішення по розглянутих варіантах V1–V4

В табл. 3.8 наведено результати порівняльного аналізу ефективності по розглянутих варіантах V1–V4. В табл. 3.8 відносний показник гіпероб'єму (HV) розраховано як відношення до значення V1_Hybrid_NSGA-II.

Таблиця 3.8

Результати порівняльного аналізу ефективності по розглянутих варіантах V1–V4

Алгоритм	Відносний Hypervolume*	Відносний час	Ефективність (HV/час)
V1_Hybrid_NSGA-II	1,000	1,0000	0,0148
V2_Simple_NSGA-II	0,996	0,0065	2,2740
V3_StaticRisk_NSGA-II	1,021	0,7630	0,0198
V4_Hybrid_NSGA-III	1,252	9,1720	0,0020
*Примірка. Відносний показник гіпероб'єму (HV) розраховано як відношення до значення V1_Hybrid_NSGA-II.			

Відносна ефективність алгоритмів, в табл. 3.8 виражена як співвідношення якості до часу виконання. Цей показник дозволив нам оцінити збалансованість між точністю рішень та обчислювальними витратами. Так алгоритм V2_Simple_NSGA-II показав найвищу ефективність за критерієм співвідношення якості до часу виконання (2,274). Тобто цей результат тлумачимо його мінімальним часом виконання. Однак варіант V2_Simple_NSGA-II працював лише з двома цільовими функціями. А це обмежувало його застосовність для складних оптимізаційних задач. Для перевірки статистичної значущості відмінностей у якості рішень проведено ANOVA тест для нормалізованого гіпероб'єму [151]. Результати тесту показали наявність статистично значущих відмінностей між алгоритмами V1–V4. Аналіз за допомогою тесту Тьюкі [152] виявив конкретні відмінності, представлені в табл. 3.9.

Таблиця 3.9

Результати статистичного порівняння якості алгоритмів V1–V4

Порівняння	Різниця середніх	95% довірчий інтервал	Статистична значущість
V4 та V1	+0,2245	[0,112; 0,337]	$p < 0,001$
V4 та V2	+0,2277	[0,115; 0,340]	$p < 0,001$
V4 та V3	+0,2057	[0,093; 0,318]	$p < 0,001$
V3 та V1	+0,0188	[–0,094; 0,132]	$p = 0,945$
V3 та V2	+0,0220	[–0,091; 0,135]	$p = 0,922$
V2 та V1	–0,0032	[–0,116; 0,110]	$p = 0,999$

Аналіз результатів, наведений в табл. 3.9 показав, що алгоритм V4_Hybrid_NSGA-III статистично значущо перевершив всі інші алгоритми за якістю рішень. Відмінності між алгоритмами V1, V2 та V3 не є статистично значущими при рівні значимості 0,05.

Також під час ОбЕк проведено аналіз стабільності результатів, див. табл. 3.10. В цій таблиці наведенні значення коефіцієнту варіації, який характеризував стабільність результатів між запусками під час ОбЕк. Тобто цей показник відображав відтворюваність результатів при різних початкових умовах в наборах даних, див. табл. 3.1–3.3.

Таблиця 3.10

Стабільність алгоритмів (коефіцієнт варіації) для алгоритмів V1–V4

Алгоритм	Hypervolume	Час виконання	Розмір фронту
V1_Hybrid_NSGA-II	14,07	1,61	13,02
V2_Simple_NSGA-II	0,52	0,55	0,00
V3_StaticRisk_NSGA-II	4,46	0,28	0,00
V4_Hybrid_NSGA-III	4,63	1,67	14,52

Алгоритм V2_Simple_NSGA-II показав найвищу стабільність результатів з коефіцієнтом варіації лише 0,52% для гіпероб'єму. Найменш стабільним виявився V1_Hybrid_NSGA-II (тобто наш метод CoopEvo-CloudSec) з коефіцієнтом варіації 14,07%. Проте як ми вже довели раніше стабільність його роботи залежить від початкових умов в наборі даних.

Для глибшого дослідження стабільності та варіативності отриманих результатів побудовано діаграми розмаху (boxplots), представлені на рис. 3.4.

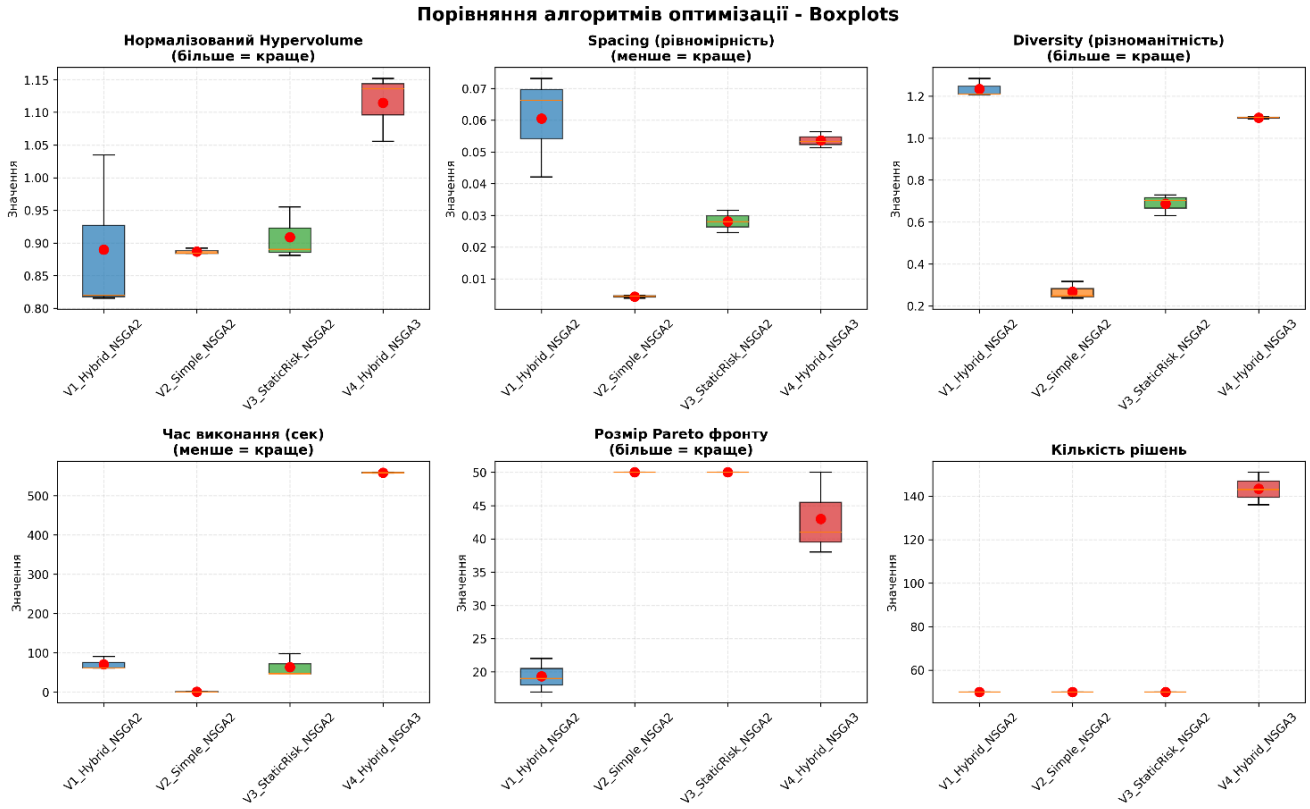


Рис. 3.4. Діаграми розмаху (boxplots) для досліджених алгоритмів V1–V4

Цей метод візуалізації результатів обчислювальних експериментів дозволив оцінити медіанні значення, квартильний розмах та наявність викидів для кожного з досліджуваних алгоритмів алгоритмів V1–V4 за метриками ефективності. Так аналіз збіжності та якості фронту (нормалізований HV в лівому верхньому куті дашборда) показав перевагу алгоритму V4_Hybrid_NSGA-III. Тобто медіанне значення для V4 перевищило 1,10, а компактність «ящика» свідчить про високу стабільність алгоритму незалежно від початкових умов генерації популяції.

Водночас, запропонований метод V1_Hybrid_NSGA-II (CoopEvo-CloudSec на базі NSGA-II) характеризувався розкидом значень між першим та третім квартилями. Це вказало на чутливість методу до стохастичної природи функції ризику $\lambda(t)$ та початкової ініціалізації.

Алгоритми V2 та V3 показали меншу варіативність. Проте їхні медіанні показники якості поступилися гібридним версіям V1 та V4. Отже це підтвердило потенціал урахування в методі CoopEvo-CloudSec коаліційних стратегій для покращення якості Парето-фронту.

Аналіз рівномірності та різноманітності рішень (Spacing та Diversity) подано на середньому графіку верхнього рядку на дашборд 3.4. Розподіл метрик Spacing та Diversity виявив цікаву залежність. Алгоритм V2_Simple_NSGA-II показав найкращі (найнижчі) показники Spacing. Отже саме він забезпечує рівномірність розподілу рішень. Однак при цьому він мав найнижчий показник Diversity. Це довело, що V2 схильний до скупчення рішень у вузькій області простору пошуку.

Натомість метод V1_Hybrid_NSGA-II (CoopEvo-CloudSec на базі NSGA-II) продемонстрував найвищий рівень метрики Diversity, медіана >1,2. Він перевершив навіть V4_Hybrid_NSGA-III. Висока різноманітність рішень є пріоритетною для задач кібербезпеки ХМС, оскільки надає особі, яка приймає рішення (ОПР), широкий простів альтернатив. Це можуть бути рішення від максимально захищених ХМС до економічно ефективних у конкретних бізнес процесах. Вищий показник Spacing для V1 є допустимим компромісом за умови забезпечення широкого покриття простору цільових функцій.

На дашборд в нижньому рядку на рис. 3.4 першою подано діаграму для аналіз часової складності реалізації певного алгоритму. V1 та V4. Тобто час виконання пошуку. Діаграма часу виконання (див. нижній лівий графік на дашборд рис. 3.4) чітко проілюструвала «вартість» використання складних еволюційних операторів в методі CoopEvo-CloudSec. Так алгоритм V4_Hybrid_NSGA-III виявився найбільш ресурсномістким, з медіанним часом виконання понад 550 с. В деяких експериментах навіть більше 600 с. Це в практичному застосуванні є обмежуючим фактором для систем реального часу, до яких належать ХМС.

Запропонований метод V1_Hybrid_NSGA-II зайняв проміжну позицію. Медіана склала приблизно 70–80 с. Він забезпечив прийнятний рівень узгодженості між якістю оптимізації ХМС та оперативністю отримання результату оптимізації. Це довело його придатність для використання в ХМС. Оскільки в подібних системах рішення про перерозподіл ресурсів мають прийматися га протязі декількох секунд. Кількісні характеристики фронту – це останні два зображення в нижньому рядку дашборд (Розмір Pareto фронту та Кількість рішень). Аналіз кількості знайдених рішень (нижні центральний та правий графіки) показали, що алгоритм V4 генерує значно більшу кількість недомінованих рішень. Медіана дорівнювала приблизно 140. Це обумовлено механізмом Reference Points, закладеним у NSGA-III.

Алгоритм V1_Hybrid_NSGA-II сформував менший за обсягом Парето-фронт. Медіана дорівнювала приблизно 20 рішенням. Однак, з огляду на високий показник Diversity для V1_Hybrid_NSGA-II, доведено в процесі ОбЕк, що ці рішення є більш унікальними та репрезентативними. Вони покривають різні екстремальні точки багатовимірного простору, на відміну від V2 та V3, які, хоч і генерували стабільно близько 50 рішень, часто дублювали стратегії захисту ХМС в межах локальних скупчень, див. рис. 3.5.

Отже резюмуючи аналіз Boxplot на рис. 3.4, констатуємо наступне. Метод V1_Hybrid_NSGA-I (CoopEvo-CloudSec) є оптимальним вибором для сценаріїв, де пріоритетом є максимізація різноманітності стратегій захисту ХМС при помірних обчислювальних витратах. Водночас, для задач стратегічного планування, де час

розрахунку не є критичним, вважаємо доцільним є використання в методі замість алгоритму NSGA-II модифікації на базі NSGA-III (V4) для отримання максимально щільного фронту рішень.

Зазначимо, що відповідно до отриманих результатів, вибір алгоритму NSGA-II чи NSGA-III для методу CoopEvo-CloudSec має ґрунтуватися на специфічних вимогах конкретного застосування. Так для задач оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки, де пріоритетним є час виконання завдання, рекомендуємо застосувати алгоритм V2_Simple_NSGA-II. Для задач з високими вимогами до якості рішень та наявністю обчислювальних ресурсів оптимальним є алгоритм V4_Hybrid_NSGA-III. Алгоритм V1_Hybrid_NSGA-II займає проміжне положення, пропонуючи узгоджені рішення між метриками якості та швидкість (для задач дослідження з чотирма цільовими функціями $F_1 - F_4$, відповідно до завдань Розділу 2 дисертації. Резюмуючи проведений аналіз складено узагальнюючу таблицю порівняння переваг та недоліків варіантів V1–V4.

Графік паралельних координат на рис. 3.5 розкрив загальну структуру фронту. Тобто рішення з низькою вартістю та коротким часом виконання завдань, як правило, асоціювалися зі збільшеним ризиком і низькою коаліційною вигодою. Рішення, що максимізували коаліційний ефект в завданнях забезпечення захисту ХМС, закономірно демонстрували під час симуляцій зростання витрат. Середній профіль (червона лінія на рис. 3.5) окреслив домінуючу тенденцію до компромісності. Фронт збалансовано розподілений між критеріями $F_1 - F_4$, і жоден із них не домінував у загальній структурі.

Результати обчислювального експерименту підтвердили ефективність запропонованого кооперативно-еволюційного методу в варіативного ризикового профілю ХМС. Алгоритм V1_Hybrid_NSGA-II не лише знаходив під час обчислювальних експериментів різноманітні компромісні рішення, але й в цілому довів здатність інтегрувати коаліційну взаємодію систем захисту в процес багатокритеріальної оптимізації. А це розширило простір можливих розв’язків у порівнянні з класичним NSGA-II (V2).

Структура фронту на рис. 3.5 демонструє логічну та передбачувану поведінку відповідно до теоретичних моделей еволюційної оптимізації та кооперативних ігор. Цей результат додатково підтвердив валідність розробленого методу CoopEvo-CloudSec.

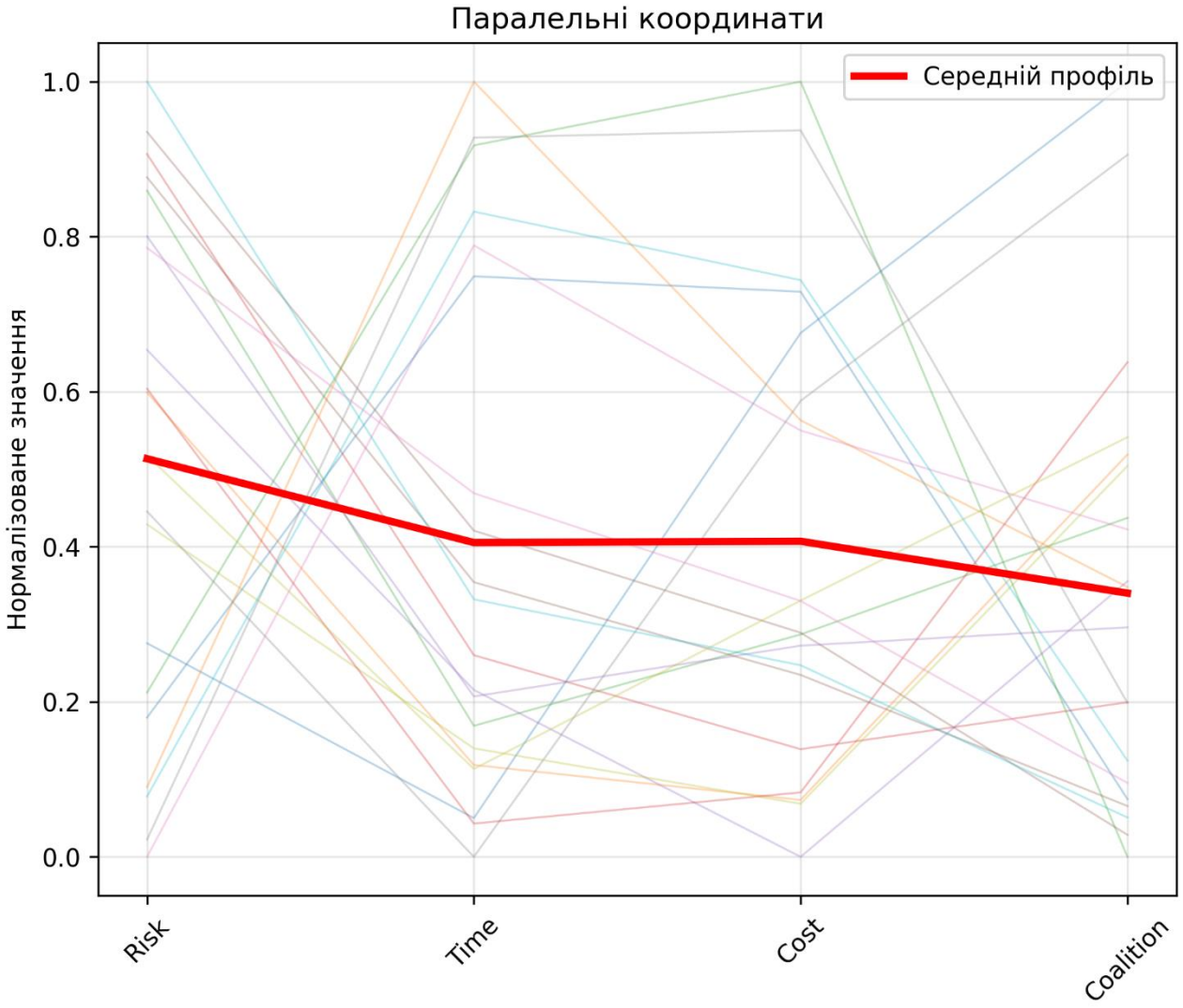


Рис. 3.5. Розподіл оптимізаційних критеріїв ($F_1 - F_4$) в паралельних координатах для V1_Hybrid_NSGA-II

Отримані на рис. 3.1–3.5 результати довели ефективність методу CoopEvo-CloudSec у порівнянні з альтернативними багатокритеріальними алгоритмами.

В табл. 3.11 наведено узагальнення переваг та недоліків варіантів V1–V4 для реалізації в методі CoopEvo-CloudSec

Таблиця 3.11

Порівняння переваг та недоліків варіантів V1–V4 для реалізації в методі CoopEvo-CloudSec

Алгоритм	Урахування ризику	Урахування коаліційної взаємодії	Тип ризику	Якість фронту Парето	Стійкість до урахування зміни ризиків для ХМС	Очікувана якість компромісних рішень	Переваги	Недоліки
V1_Hybrid_ NSGA-II	+	+	Варіативні значення	Висока при =4 цілях	Низька	Прийнятна збалансованість між критеріями	Потенціал масштабування	Потребує якісних даних
V2_Simple_ NSGA-II	+/–	–	Немає	Низька	Середня	Середня	Простота, класичний еталон	Не реагує на зміни загроз
V3_StaticRisk_ NSGA-II	+/–	–	Статичний	Середня	Середня	Середня	Легко реалізувати	Потребує тонкого налаштування
V4_Hybrid_ NSGA-III	+	+	Часовий процес	Висока при кількості критерії >4	Висока	Найкраща збалансованість між ризиком, часом і вартістю	Гнучкість в виборі критеріїв для оптимізації	Великий час пошуку рішення
Позначення: (+) – є, (–)– немає, (+/–) – частково (потребує налаштування).								

З метою об’єктивної валідації запропонованого в дисертації методу CoopEvo-CloudSec та визначення його місця серед наявних методів та моделей в завданні управління ресурсами ХМС, проведено систематизований порівняльний аналіз. На основі огляду літературних джерел відібрано репрезентативні методи, близькі концептуально до нашого дослідження. Основними критеріями порівняння з чинними роботами визначено здатність методів враховувати варіативність ризиків для ХМС, можливість багатокритеріальної оптимізації та наявність механізмів кооперативної взаємодії (тобто коаліцій) між компонентами захисту ХМС. Узагальнені результати аналізу наведені в табл. 3.12. А далі в п. 3.3 ми розглянемо архітектурну схему впровадження методу CoopEvo-CloudSec в практику оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки [153].

Таблиця 3.12

Порівняльний аналіз методу CoopEvo-CloudSec з чинними методами та моделями розподілу ресурсів ХМС (складено автором)

Назва моделі (методу), автори та джерело	Спільні риси з методом метод CoopEvo-CloudSec	Переваги та відмінності методу CoopEvo-CloudSec.
1	2	3
SM-VMP (Secure and Multiobjective Virtual Machine Placement framework). Saxena D., Gupta I., Kumar J. та ін. [85]	Багатокритеріальність та еволюційний підхід. Метод [85] використовує еволюційні алгоритми для пошуку компромісу між безпекою та продуктивністю/вартістю. Обидва методи формують фронт Парето	SM-VMP розглядає ризик як статичний параметр (уразливість гіпервізора). CoopEvo-CloudSec вводить функцію ризику $\lambda(t)$. Крім того, в SM-VMP не враховано можливість кооперації захисників (коаліційну вигоду)
Модель розміщення ВМ з урахуванням ризиків (Risk-aware VM placement). Han J., Zang W., Liu L. та ін. [84]	Обидва методи розглядають мінімізацію ризиків безпеки як одну з цільових функцій оптимізації поряд з економічними показниками. В [84] використовували багатокритеріальний підхід	Метод [84] Han J. Переважно зосереджений на фізичному розміщенні для уникнення атак побічними каналами, але ігнорує активну протидію. CoopEvo-CloudSec інтегрує ігрову модель, де захисник прилаштовує стратегію проти дій атакуючого. Також метод CoopEvo-CloudSec містить 4-й критерій «Коаліційна вигода», який відсутній в [84]
Ігрова модель розподілу ресурсів на основі ігор Штакельберга. Wilczyński A. [89]; Ait Temghart A. et al. [92]	Також описано використання теорії ігор. Методи [89], [92] та CoopEvo-CloudSec моделюють взаємодію «захисник–атакуючий». Автори припускали, що дії однієї сторони впливають на виграш іншої. Тобто маємо конфлікт інтересів	Відрізаються тип гри та гібридизація. У [89] та [92] використовуються некооперативні ігри. Тобто лідер-послідовник, де кожен сам за себе. В CoopEvo-CloudSec використовуємо кооперативну теорію ігор - розрахунок значень Шеплі. Це дозволяє моделювати спільний захист у мультихмарному середовищі. Крім того, CoopEvo-CloudSec інтегрує гру всередину еволюційного алгоритму NSGA-II (або NSGA-III), тоді як [89, 92] зазвичай автори шукають лише точку рівноваги Неша/Штакельберга без побудови фронту Парето

Продовження таблиці 3.12

1	2	3
Cooperative Game Theory approach to fair resource allocation. Xu X. & Yu H. [91]	Кооперативний метод [91] також використовує апарат кооперативних ігор та вектор Шеплі для розподілу вигоди між учасниками хмарного середовища	Метод [91] сфокусовано на справедливості розподілу ресурсів та прибутку, тобто пріоритет – економічний аспект. Метод CoopEvo-CloudSec адаптував цей апарат саме для безпеки. Інтегровано колективний захист від загроз. А також метод CoopEvo-CloudSec поєднує оцінку захищеності із оптимізацію продуктивності ХИС через NSGA-II. Цього немає в [91]
MILP framework (Mixed Integer Linear Programming) for VM security. Mangalagowri R. & Venkataraman R. [99]	Обидва методи формалізують задачу розподілу як задачу оптимізації з обмеженнями (ресурси, бюджет, політики безпеки)	Метод [99] використовує лінійне програмування (MILP), яке дає точний розв'язок, але складно масштабувати. В методі CoopEvo-CloudSec використовуємо евристичний алгоритм (NSGA-II або NSGA-III). Це дозволило нам знаходити субоптимальні рішення значно швидше в ХМС зі змінним ризиком $\lambda(t)$
Архітектура SECURE (Self-protection approach). Gill S. S. & Buyya R. [69]	Обидва методи зосереджено на адаптації ХМС до виявлених загроз та перерозподілі ресурсів під час атак	SECURE [69] – це архітектурний фреймворк (концепція). Метод CoopEvo-CloudSec містить конкретний математичний метод та алгоритми з чітко визначеними функціями пристосованості (Fitness functions) та метриками оцінки (HV, IGD). Це дозволило під час тестування отримати кількісну, а не лише якісну оцінку ефективності
Методи багатокритеріального аналізу для IaaS (Петровська І.Ю. та співавт. [20–25])	Акцент робився на оптимізації розподілу ресурсів у хмарних середовищах (IaaS). Використовувався апарат багатокритеріального аналізу для оцінки ефективності	CoopEvo-CloudSec пропонує значно глибшу інтеграцію безпеки. Тобто безпекові ризики формалізуємо через ігрову модель та вони є критерієм оптимізації, а не загальний елемент. В [20–25] безпека розглянуто переважно в розрізі базового захисту даних без моделювання антагоністичної взаємодії

3.3. Архітектурна схема впровадження методу CoopEvo-CloudSec в практику оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки

Архітектурне проєктування методу CoopEvo-CloudSec ґрунтувалося на інтеграції еволюційного механізму оптимізації з процесами керування хмарною інфраструктурою. У центрі архітектури, див. рис. 3.6 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.

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. 3.7 передбачає можливість безперервного отримання потоків подій від SIEM, EDR

та систем моніторингу. Це потенційно дає змогу оновлювати політики розподілу ресурсів без втручання оператора.

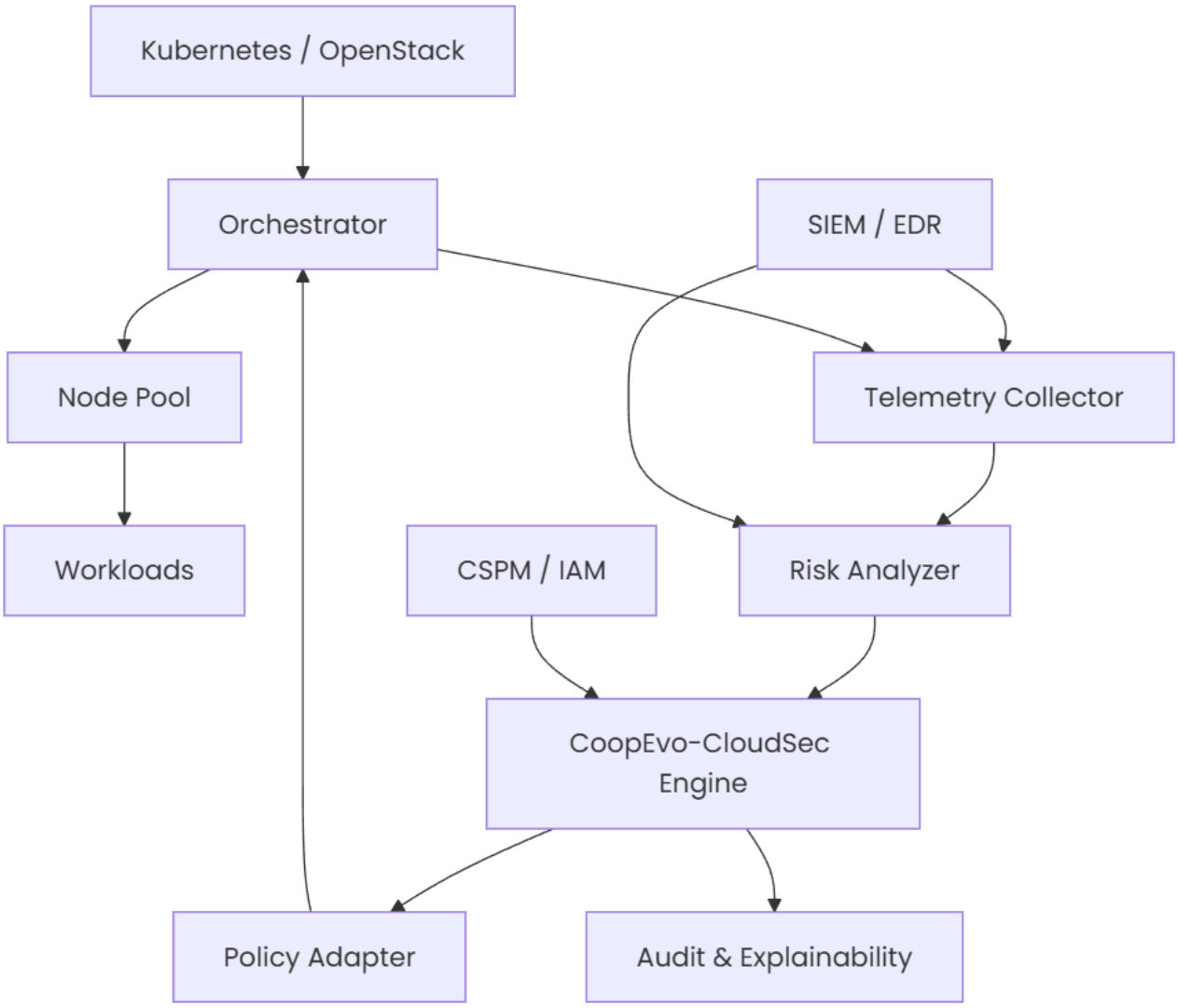


Рис. 3.6. Схема реалізації архітектури методу CoopEvo-CloudSec

У межах архітектури на рис. 3.6 кожен компонент виконує власну функцію, формуючи цілісний механізм оптимізації, згідно до методу CoopEvo-CloudSec. Центральне місце на рис. 3.6 займає CoopEvo-CloudSec Engine (коди наведено в Додатках А та Б). 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. 3.13. 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. 3.6. 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. 3.13.

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 [149].

В табл. 3.13 подамо детально сформований опис компонентів архітектури CoopEvo-CloudSec та характеристик їхніх інтерфейсів – API, протоколів, форматів повідомлень і методів автентифікації.

Таблиця 3.13

Компоненти архітектури CoopEvo-CloudSec та їх API (розроблено автором)

Компонент архітектури на рис. 3.6	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

Потік даних, див. рис. 3.7 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 [149]. 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.

Рис. 3.7. Схема потоку даних у методі CoopEvo-CloudSec

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) [154, 155]. 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.

Runtime decision loop у методі CoopEvo-CloudSec на рис. 3.8 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. 3.9 та 3.10 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.

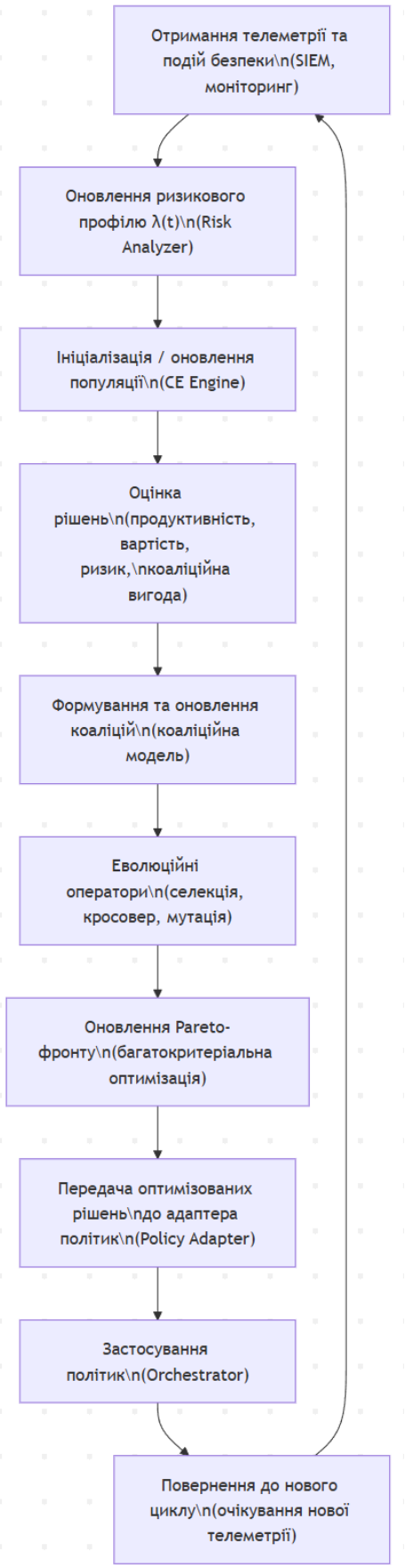


Рис. 3.8. Діаграма роботи модуля Runtime decision loop у методі CoopEvo-CloudSec

Зазначимо, що спостережувана на дашборд розбіжність шкал на графіках рис. 3.8 та 3.10 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. 3.9. 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. 3.8 та 3.10 ці критерії відкладаються на одних і тих же або суміжних графіках.

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 дисертації 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.

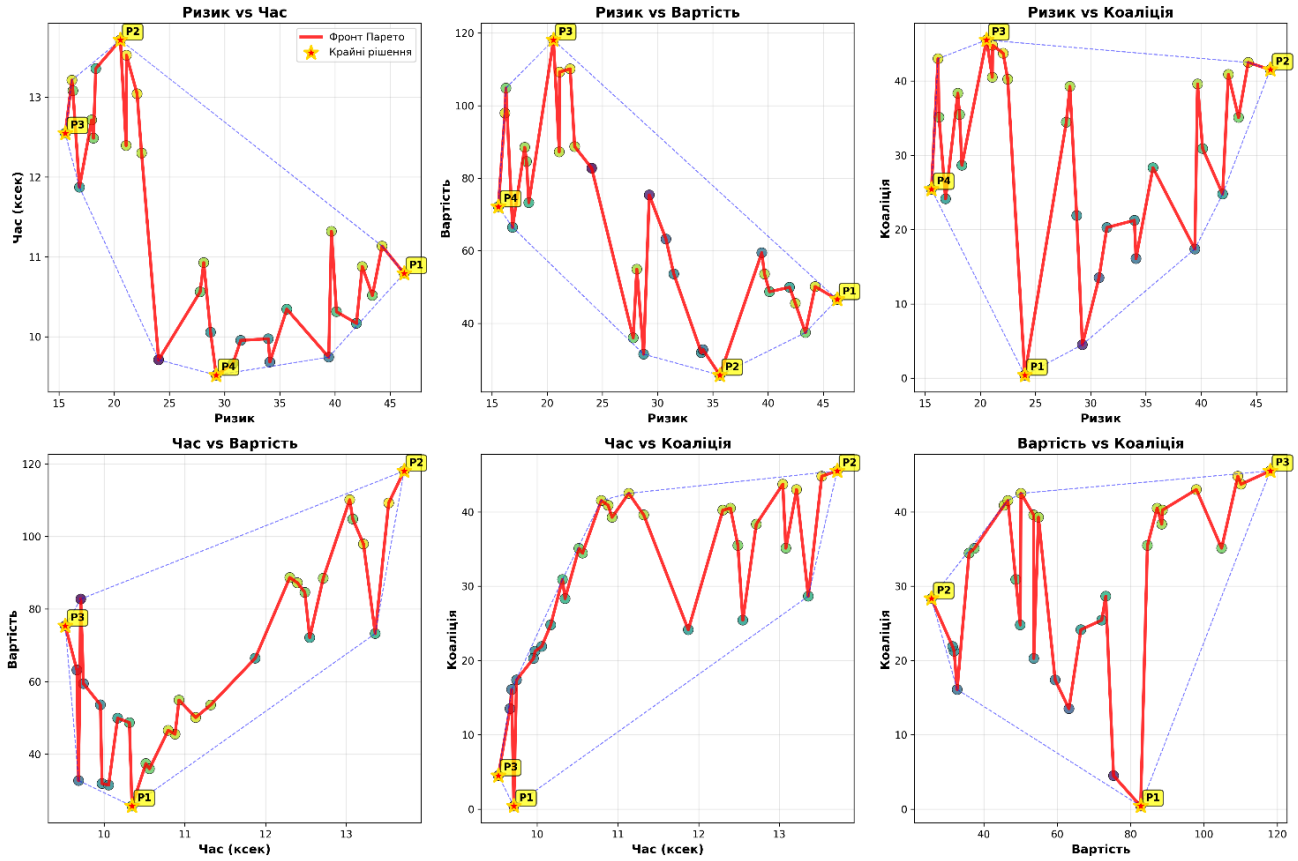


Рис. 3.9. Процес формування оптимальних рішень для розв’язанні завдання оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки (в форматі 2D)

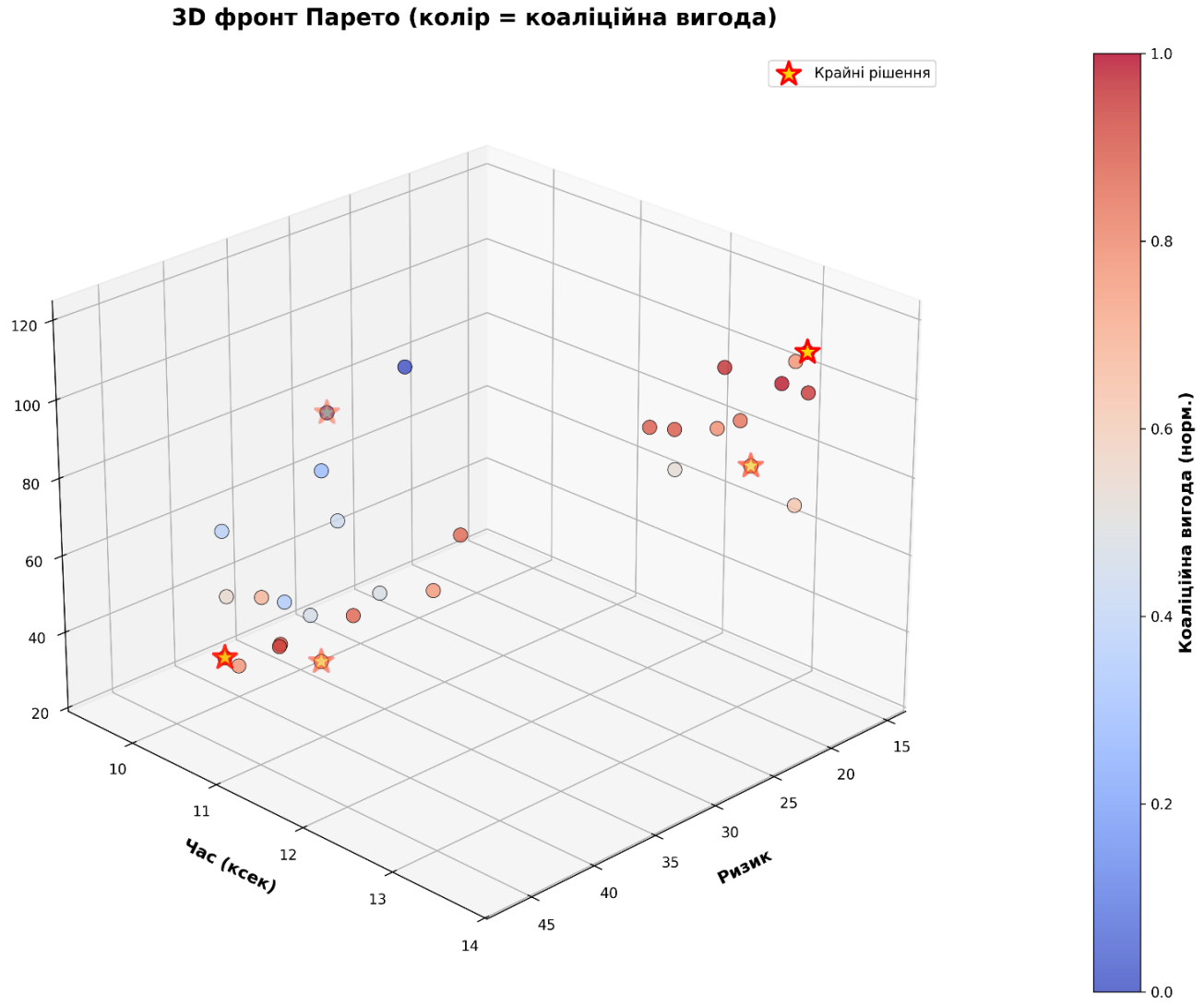


Рис. 3.10. Процес формування оптимальних рішень для розв'язанні завдання оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки (в форматі 3D)

Наведені вище судження справедливі також для одного з варіантів візуалізації Парето – фронту , який наведено на рис. 3.10 в форматі 3D. На рис. 3.10 зірочками показані кращі рішення на фронту Парето.

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

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}. \qquad (3.1)$$

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

$$x' = \frac{x - \mu}{\sigma}, \tag{3.2}$$

а робастна шкала використовує медіану і міжквартильний розмах:

$$x' = \frac{x - \text{median}(x)\mu}{IQR(x)}. \tag{3.3}$$

Крім того, для сильно скошених розподілів доцільно застосовувати на етапі проектування промислового варіанту модуля CoopEvo-CloudSec Engine, згідно технічного завдання, 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. 3.14.

Отже в табл. 3.14 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. 3.14 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.

Таблиця 3.14

Сценарії та платформи доцільного застосування методу CoopEvo-CloudSec (розроблено автором)

Сценарій застосування та тип навантаження	Рекомендовані хмарні платформи та сервіси	Специфіка застосування методу CoopEvo-CloudSec	Домінуючі критерії оптимізації (пріоритети)
1	2	3	4
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

Продовження таблиці 3.14

1	2	3	4
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)

Також, варто зауважити, що універсальність методу CoopEvo-CloudSec не означає використання єдиного статичного набору параметрів для всіх типів оптимізаційних задач. Ефективність функціонування запропонованої в п. 3.3 архітектури (див. рис. 3.6) залежить від коректного налаштування вхідних параметрів оптимізаційної моделі (2.27)–(2.36), зокрема моделі оцінювання ризику $\lambda(t)$, типу еволюційного алгоритму (NSGA-II чи NSGA-III) та вагових коефіцієнтів цільових функцій $F_1 - F_4$ [156].

Для забезпечення прилаштовування методу CoopEvo-CloudSec до специфічних вимог різних предметних областей, розроблено матрицю рекомендованих конфігурацій, див. табл. 3.15.

Таблиця 3.15

Рекомендовані конфігурації параметрів методу CoopEvo-CloudSec та прогнозована ефективність для різних класів задач (розроблено автором)

Клас задач (Сценарій)	Рекомендована модель динаміки ризику $\lambda(t)$	Вибір алгоритму (Ядро методу)	Пріоритетні налаштування ваг $(w_1 - w_4)$	Прогнозований ефект від впровадження
1	2	3	4	5
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 оптимізації 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

Продовження таблиці 3.15

1	2	3	4	5
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
Примітки. Прогнозований ефект в колонці (5) обраховано на підставі узагальнень результатів обчислювальних експериментів та даних отриманих при впроваджені розроблених програмних застосунків на підприємствах, що підтверджено актами на впровадження (див. додатки до дисертаційної роботи).				

Матриця в табл. 3.15 виступає як дорадник системним архітекторам та адміністраторам безпеки ХМС обрати оптимальний режим роботи обчислювального ядра методу CoopEvo-CloudSec (Додатки А та Б).

Висновки до розділу 3

В результаті досліджень третього розділу дисертації зроблено наступні висновки та отримано такі результати:

1. Реалізовано обчислювальні експерименти, спрямовані на емпіричне оцінювання властивостей та ефективності запропонованого методу CoopEvo-CloudSec. Метод поєднав алгоритми багатокритеріальної еволюційної оптимізації NSGA-II або NSGA-III, залежно від обмежень пошуку рішень в часі, та кооперативної ігрової моделі оцінювання ризику ХМС. На підставі проведених симуляцій, аналізу результатів та порівняння із базовими алгоритмами розв’язку багатокритеріальних оптимізаційних завдань, отримано комплексні висновки що до ефективності використання методу CoopEvo-CloudSec.

2. Експериментально доведено, що інтеграція кооперативної взаємодії ресурсів ХМС із варіативним ризиковим параметром $\lambda(t)$ істотно впливає на характер формування Парето-оптимальних рішень. Отримані в процесі

обчислювальних експериментів профілі рішень засвідчили, що математичний апарат кооперативних стратегій забезпечив переваги в балансуванні між безпекою та продуктивністю ХМС, які недосяжні у класичній постановці NSGA-II. Цей висновок узгоджено з наведеним у методі CoopEvo-CloudSec акцентом на здатності системи прилаштовуватися до змін ризикового профілю хмарних сервісів у часі.

3. Продемонстровано під час експериментальних досліджень, що оптимуми, отримані методом CoopEvo-CloudSec, характеризувалися високою різноманітністю простору рішень. Встановлено існування рішень, орієнтованих на максимізацію коаліційної взаємодії. Це стало індикатором здатності методу CoopEvo-CloudSec підтримувати сценарії активного реагування на загрози хмарним сервісам. Подібні рішення підтвердили ефективність методу CoopEvo-CloudSec в завданнях посилення узгоджених дій компонентів хмарної інфраструктури в умовах підвищених метрик безпеки. Досліджено механізми реалізації стратегії захисника, спроможної змінювати розподіл завдань ХМС у відповідь на зміну параметра ризику та виявлену активність порушника ІБ. Доведено, експериментально, що метод CoopEvo-CloudSec забезпечив гнучкість у виборі найкращого варіанта реагування на зміну ризикового профілю ХМС. На відміну від наявних рішень, враховано можливість співпраці між захистками ХМС, що дозволило під час симуляції моделювати коаліційні сценарії протидії загрозам ХМС. При цьому досліджено виграш коаліції для оцінки доцільності об'єднання та формування узгоджених стратегій.

4. Доведено, завдяки оцінюванню метрик HV, IGD та Spacing, що застосування залежності моделювання ризику та кооперації в протидії загрозам ХМС, призвели до підвищення рівномірності заповнення фронту Парето та покращенню покриття простору рішень. У порівнянні з базовою реалізацією NSGA-II, розроблений метод CoopEvo-CloudSec досяг збалансованості між критеріями безпеки, продуктивності та вартості. Результати обчислювальних експериментів підтвердили здатність алгоритмічно-програмної реалізації методу CoopEvo-CloudSec якісно та кількісно оцінити поведінку еволюційного процесу під час пошуку оптимального розподілу обчислювальних ресурсів ХМС для підвищення безпеки.

5. Аналіз параметрів симуляції засвідчив стабільність алгоритмічної поведінки у широкому діапазоні сценаріїв функціонування хмарних систем. Варіювання ризикового параметра $\lambda(t)$ дало змогу промоделювати різні режими навантаження на системи ІБ ХМС. Встановлено, що метод CoopEvo-CloudSec (V1) забезпечив підвищення метрики Diversity до 1,23, що перевищило показники базового NSGA-II (0,268), гарантуючи ширше покриття простору рішень. Доведено, що використання алгоритму NSGA-III (V4) у складі методу CoopEvo-CloudSec дозволило покращити показник HV на 25% (до 1,11), хоча це і призвело до відчутного збільшення часу обчислень (майже в 9 разів).

6. Проведено порівняння методу CoopEvo-CloudSec з альтернативними еволюційними алгоритмами, зокрема NSGA-III. Отримані результати засвідчили потенційну перевагу CoopEvo-CloudSec в задачах, де є необхідним інтеграція показників безпеки ХМС та коаліційної вигоди. Результати симуляцій довели, що саме включення кооперативного компонента дозволило змістити оптимальні рішення у напрямі підвищення захисного потенціалу хмарної інфраструктури у багатокористувацьких середовищах.

7. Комплексний аналіз результатів підтвердив наукову гіпотезу про те, що багатокритеріальна постановка задачі розподілу обчислювальних ресурсів із урахуванням варіативного ризику є більш адекватною до реальних умов функціонування ХМС. В результаті проведення обчислювальних експериментів отримано підтверджене працездатності та ефективності методу CoopEvo-CloudSec. Цей факт підтвердили також Акти впровадження, наведені у додатках дисертації.

8. Доведено, що практична цінність дисертаційного дослідження полягає в тому, що метод CoopEvo-CloudSec забезпечив нові можливості для управління обчислювальними ресурсами ХМС в умовах зміни ризикового ландшафту для релевантних кіберзагроз. Метод CoopEvo-CloudSec має перспективи стати складовою обчислювального ядра інтелектуальних систем прийняття рішень у хмарних середовищах, включаючи OpenStack, Kubernetes та інші платформи з оркестраційними механізмами. Розроблений метод дозволив підвищити стійкість ХМС шляхом інтеграції оцінки ризиків у процедури планування обчислювальних

ресурсів ХМС, а також завдяки елементам моделювання кооперативної поведінки між компонентами хмарної інфраструктури. На відміну від наявних методів та моделей, які зазвичай розглядають продуктивність і вартість ізольовано, запропонований метод CoopEvo-CloudSec дозволив враховувати багатовимірність експлуатаційних характеристик ХМС та одночасно оптимізувати безпеку, витрати й час виконання.

ВИСНОВКИ

У дисертації вирішено актуальне наукове завдання, яке полягає в підвищенні ефективності застосування методів та моделей оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки завдяки розробки методу кооперативно-еволюційного розподілу обчислювальних ресурсів. Дане наукове завдання має важливе значення для теорії і практики захисту інформації, створення та забезпечення функціонування інформаційних систем і технологій на об’єктах інформаційної діяльності та критичних інфраструктур сфери кібербезпеки та захисту інформації. Відсутність аналогічних рішень в нашій країні і закордоном робить результати досліджень пріоритетними. Отримані результати мають важливе значення для модернізації існуючих та в процесі розробки нових методів захисту інформації.

На підставі проведених досліджень зроблені наступні висновки:

1. Проаналізовано теоретичні та прикладні засади функціонування ХМС. Визначено, що специфіка ХМС полягає у багатокористувацькому середовищі та варіативному характері навантаження, що ускладнює завдання раціонального розподілу ресурсів й вираховування метрик забезпечення захисту. Проаналізовано релевантні архітектурні моделі ХМС: IaaS, PaaS, SaaS, FaaS, CaaS, XaaS тощо. Проаналізовано наявні методи та моделі управління ОбР ХМС. Показано, на підставі аналізу попередніх досліджень вітчизняних на закордонних авторів, що чинні методи переважно базуються на алгоритмах планування та балансування навантаження у ХМС. Однак ці методи та відповідні моделі зазвичай ігнорують фактор ризиків безпеці ХМС та складну взаємодію між порушником ІБ і захисником. Доведено, що релевантні методи багатокритеріальної оптимізації, зокрема еволюційні алгоритми, дають змогу враховувати конфліктність цілей і шукати компромісні рішення.

2. Доведена потреба у синтезі нових методів керування ОбР ХМС, які здатні пристосовуватися до змін ризикового профілю. Запропоновано концепцію гібридної моделі, яка інтегрує методи, моделі та алгоритми багатокритеріальної оптимізації, теорії ігор та адаптивного управління ризиком для ХМС. З

урахуванням ознак поведінки порушника ІБ та захисника у ХМС, розроблено ігрову модель, яка віддзеркалює конфліктну взаємодію між гравцями. В моделі захисник формує стратегії розподілу задач на вузли ХС, а порушник – вибирає цілі атак. У межах цієї взаємодії введено параметр агресивності $\lambda(t)$, який описує зміну інтенсивності атак у часі, що дозволило врахувати стохастичну, марковську або еволюційну природу ризику. Відштовхуючись від підсумків роботи ігрової моделі, сформовано функцію оцінки ризику для кожного елементу ХМС, що застосовувався як один із критеріїв у завданні багатокритеріальної оптимізації.

3. З метою відшукання компромісних рішень вперше розроблено багатокритеріальну оптимізаційну модель з врахуванням чотирьох критеріїв – ризику, продуктивності, вартості та коаліційної вигоди між захисниками ХС, що на відміну від наявних рішень на базі алгоритму NSGA-II або NSGA-III, дозволяє сформувати множину Парето-оптимальних рішень. У моделі вперше, на відміну від наявних рішень, враховано можливість співпраці між захистками ХМС, що дозволило моделювати коаліційні сценарії протидії загрозам. При цьому введено поняття виграшу коаліції, яке застосовувався для оцінки доцільності об'єднання та формування узгоджених стратегій.

4. Вперше запропоновано новий метод кооперативно-еволюційного розподілу ОбР у ХМС з урахуванням ризику (CoopEvo-CloudSec), який на відміну від наявних методів поєднав алгоритми багатокритеріальної еволюційної оптимізації NSGA-II або NSGA-III, залежно від обмежень пошуку рішень в часі, та кооперативної ігрової моделі оцінювання ризику для ХМС.

5. Проведено ОбЕк, спрямовані на емпіричне оцінювання властивостей та ефективності запропонованого методу CoopEvo-CloudSec. На підставі проведених симуляцій, аналізу результатів та порівняння із базовими алгоритмами розв’язку багатокритеріальних оптимізаційних завдань, отримано комплексні висновки що до ефективності використання методу CoopEvo-CloudSec. Експериментально доведено, що інтеграція кооперативної взаємодії ресурсів ХС із варіативним ризиковим параметром $\lambda(t)$ істотно впливає на характер формування Парето-оптимальних рішень. Отримані в процесі ОбЕк профілі рішень засвідчили, що

математичний апарат кооперативних стратегій забезпечив переваги в балансуванні між безпекою та продуктивністю ХС, які недосяжні у класичній постановці NSGA-II. Продемонстровано під час експериментальних досліджень, що оптимуми, отримані методом CoopEvo-CloudSec, характеризувалися високою різноманітністю простору рішень. Встановлено існування рішень, орієнтованих на максимізацію коаліційної взаємодії. Доведено, експериментально, що метод CoopEvo-CloudSec забезпечив гнучкість у виборі найкращого варіанта реагування на зміну ризикового профілю ХС. Доведено, завдяки оцінюванню метрик HV, IGD та Spacing, що застосування залежності моделювання ризику та кооперації в протидії загрозам ХС, призвели до підвищення рівномірності заповнення фронту Парето та покращенню покриття простору рішень. У порівнянні з базовою реалізацією NSGA-II, розроблений метод CoopEvo-CloudSec досяг збалансованості між критеріями безпеки, продуктивності та вартості ХМС. Результати ОбЕк підтвердили здатність алгоритмічно-програмної реалізації методу CoopEvo-CloudSec якісно та кількісно оцінити поведінку еволюційного процесу під час пошуку оптимального розподілу ОбР ХМС для підвищення безпеки. Аналіз параметрів симуляції засвідчив стабільність алгоритмічної поведінки у широкому діапазоні сценаріїв функціонування ХМС. Варіювання ризикового параметра $\lambda(t)$ дало змогу промоделювати різні режими навантаження на системи ІБ ХС. Встановлено, що метод CoopEvo-CloudSec (V1) забезпечив підвищення метрики Diversity до 1,23, що перевищило показники базового NSGA-II (0,268), гарантуючи ширше покриття простору рішень. Доведено, що використання алгоритму NSGA-III (V4) у складі методу CoopEvo-CloudSec дозволило покращити показник HV на 25% (до 1,11), хоча це і призвело до відчутного збільшення часу обчислень (майже в 9 разів).

6. Проведено порівняння методу CoopEvo-CloudSec з альтернативними еволюційними алгоритмами, зокрема NSGA-III. Отримані результати засвідчили потенційну перевагу CoopEvo-CloudSec в задачах, де є необхідним інтеграція показників безпеки ХС та коаліційної вигоди. Результати симуляцій довели, що саме включення кооперативного компонента дозволило змістити оптимальні

рішення у напрямі підвищення захисного потенціалу ХМС. Комплексний аналіз результатів підтвердив наукову гіпотезу про те, що багатокритеріальна постановка задачі розподілу ОбР із урахуванням варіативного ризику є більш адекватною до реальних умов функціонування ХС. В результаті проведення ОбЕк отримано підтверджене працездатності та ефективності методу CoopEvo-CloudSec. Цей факт підтвердили також Акти впровадження, наведені у додатках дисертації. Доведено, що практична цінність дисертаційного дослідження полягає в тому, що метод CoopEvo-CloudSec забезпечив нові можливості для управління ОбР ХМС в умовах зміни ризикового ландшафту для релевантних кіберзагроз. Показано, що метод CoopEvo-CloudSec має перспективи стати складовою обчислювального ядра інтелектуальних систем прийняття рішень у ХМС, включаючи OpenStack, Kubernetes та інші платформи з оркестраційними механізмами. Розроблений метод дозволив підвищити стійкість ХС шляхом інтеграції оцінки ризиків у процедури планування ОбР ХМС, а також завдяки елементам моделювання кооперативної поведінки між компонентами ХМС. На відміну від наявних методів та моделей, які зазвичай розглядають продуктивність і вартість ізольовано, запропонований метод CoopEvo-CloudSec дозволив враховувати багатовимірність експлуатаційних характеристик ХС та одночасно оптимізувати безпеку, витрати й час виконання.

Таким чином, поставлене актуальне наукове завдання розв’язане у повному обсязі. Усі визначені часткові завдання вирішено, мету досліджень досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Parast, F. K., Sindhav, C., Nikam, S., Yekta, H. I., Kent, K. B., & Hakak, S. (2022). Cloud Computing Security: A Survey of Service-based Models. *Comput. Secur., 114*, 102580. https://doi.org/10.1016/j.cose.2021.102580

2. Дорогий, Я. Ю. (2016). Розподіл ресурсів критичної ІТ-інфраструктури з використанням хмарних технологій. *Електроніка та зв'язок, 21(1)*, 42–49.

3. Saxena, D., & Singh, A. K. (2020). Security Embedded Dynamic Resource Allocation Model for Cloud Data Centre. *Electron. Lett., 56(20)*, 1062–1065. https://doi.org/10.1049/el.2020.1736

4. Saidi, K., Hioual, O., & Siam, A. (2019). Resources Allocation in Cloud Computing: A Survey. In *Int. Conf. in Artificial Intelligence in Renewable Energetic Systems* (pp. 356–364). Springer. https://doi.org/10.1007/978-3-030-37207-1_37

5. Senthilkumar, G., Tamilarasi, K., Velmurugan, N., & Periasamy, J. K. (2023). Resource Allocation in Cloud Computing. *J. Adv. Inf. Technol., 14(5)*, 1063–1072. https://doi.org/10.12720/jait.14.5.1063-1072

6. Rabaaoui, S., Hachicha, H., & Zagrouba, E. (2024). An Efficient and Autonomous Dynamic Resource Allocation in Cloud Computing with Optimized Task Scheduling. *Procedia Comp. Sci., 246*, 3654–3663. https://doi.org/10.1016/j.procs.2024.09.191

7. Jabber, S. A., Hashem, S. H., & Jafer, S. H. (2023). Task Scheduling and Resource Allocation in Cloud Computing: A Review and Analysis. In *2023 3rd Int. Conf. on Emerging Smart Technologies and Applications (eSmarTA)* (pp. 1–8). IEEE. https://doi.org/10.1109/esmarta59349.2023.10293517

8. Kareem Awad, W., Zainol Ariffin, K. A., Nazri, M. Z. A., & Yassen, E. T. (2025). Resource Allocation Strategies and Task Scheduling Algorithms for Cloud Computing: A Systematic Literature Review. *J. Intell. Syst., 34(1)*, 20240441. https://doi.org/10.1515/jisys-2024-0441

9. Mohamed, A., Hamdan, M., Khan, S., Abdelaziz, A., Babiker, S. F., Imran, M., & Marsono, M. N. (2021). Software-defined Networks for Resource Allocation in Cloud

Computing: A Survey. *Comput. Netw., 195*, 108151. https://doi.org/10.1016/j.comnet.2021.108151

10. Vergara, J., Botero, J., & Fletscher, L. (2023). A Comprehensive Survey on Resource Allocation Strategies in Fog/Cloud Environments. *Sensors, 23(9)*, 4413. https://doi.org/10.3390/s23094413

11. Mohammad, A., & Abbas, Y. (2024). Key Challenges of Cloud Computing Resource Allocation in Small and Medium Enterprises. *Digit., 4(2)*, 372–388. https://doi.org/10.3390/digital4020018

12. Lekkala, C. (2024). AI-Driven Dynamic Resource Allocation in Cloud Computing: Predictive Models and Real-Time Optimization. *J. Artif. Intell. Mach. Learn. Data. Sci., 2(2)*, 450–456. https://doi.org/10.51219/jaimld/chandrakanth-lekkala/124

13. Ashawa, M., Douglas, O., Osamor, J., & Jackie, R. (2022). RETRACTED ARTICLE: Improving Cloud Efficiency through Optimized Resource Allocation Technique for Load Balancing using LSTM Machine Learning Algorithm. *J. Cloud Comput., 11(1)*, 87. https://doi.org/10.1186/s13677-022-00362-x

14. Manzoor, M. F., Abid, A., Farooq, M. S., Nawaz, N. A., & Farooq, U. (2020). Resource Allocation Techniques in Cloud Computing: A Review and Future Directions. *Elektronika Ir Elektrotechnika, 26(6)*, 40–51. https://doi.org/10.5755/j01.eie.26.6.25865

15. Gong, S., Yin, B., Zheng, Z., & Cai, K. Y. (2019). Adaptive Multivariable Control for Multiple Resource Allocation of Service-based Systems in Cloud Computing. *IEEE Access, 7*, 13817–13831. https://doi.org/10.1109/access.2019.2894188

16. Madni, S. H. H., Latiff, M. S. A., Coulibaly, Y., & Abdulhamid, S. I. M. (2016). Recent Advancements in Resource Allocation Techniques for Cloud Computing Environment: A Systematic Review. *Clust. Comput., 20(3)*, 2489–2533. https://doi.org/10.1007/s10586-016-0684-4

17. Li, C., & Li, L. Y. (2012). Optimal Resource Provisioning for Cloud Computing Environment. *J. Supercomput., 62(2)*, 989–1022. https://doi.org/10.1007/s11227-012-0775-9

18. Волк, М. О., Курочкін, В. С., Запорожченко, А. П., Паронікян, П. А. (2024). Гібридний метод розподілу ресурсів в хмарних системах. Системи управління,

навігації та зв’язку. *Збірник наукових праць, 2(76)*, 70–73. https://doi.org/10.26906/sunz.2024.2.070

19. Гребенюк, Д. С., Давидов, В. В. (2020). Метод первинного виділення хмарних обчислювальних ресурсів на основі аналізу ієрархій. Системи управління, навігації та зв’язку. *Збірник наукових праць, 3(61)*, 80–85.

20. Петровська, І. Ю. (2023). Методи розподілу ресурсів в комп’ютерних системах при наданні хмарних інфраструктурних послуг. *Дисертація на здобуття наукового ступеня доктора філософії*. Харків.

21. Петровська, І. Ю., Кучук, Н. Г., Панченко, В. І., & Філоненко, А. М. (2019). Рівномірний розподіл ресурсів комп’ютерних систем, що мають гіперконвергентну інфраструктуру. *Системи управління, навігації та зв’язку, 2(54)*, 119–122. https://doi.org/10.26906/SUNZ.2019.2.119

22. Петровська І. Ю., Коломійцев О. В., & Алі, А. Ф. (2021). Метод розрахунку розміру буферної пам’яті самовідновлювального сегмента телекомунікаційної мережі. *Системи управління, навігації та зв’язку, 2(64)*, 144–147. https://doi.org/10.26906/SUNZ.2021.2.144

23. Petrovska, I., & Kuchuk, H. (2022). Static Allocation Method in a Cloud Environment with a Service Model IAAS. *Adv. Inf. Syst., 6(3)*, 99–105. https://doi.org/10.20998/2522-9052.2022.3.13

24. Петровська, І. Ю., Кучук, Г. А. (2022). Розподіл обчислювальних ресурсів у хмарних системах. *Системи управління, навігації та зв’язку, 2(68)*, 75–78.

25. Petrovska, I, Kuchuk Heorhii. (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

26. Ahmad, W., Rasool, A., Javed, A. R., Baker, T., & Jalil, Z. (2021). Cyber Security in IoT-based Cloud Computing: A Comprehensive Survey. Electron., 11(1), 16. https://doi.org/10.3390/electronics11010016

27. Abdullayeva, F. (2023). Cyber Resilience and Cyber Security Issues of Intelligent Cloud Computing Systems. *Results Control Opt., 12*, 100268. https://doi.org/10.1016/j.rico.2023.100268

28. Salek, M. S., Khan, S. M., Rahman, M., Deng, H. W., Islam, M., Khan, Z., Chowdhury, M., & Shue, M. (2021). A Review on Cybersecurity of Cloud Computing for Supporting Connected Vehicle Applications. *IEEE Internet Things J., 9(11)*, 8250–8268. https://doi.org/10.36227/techrxiv.17429510

29. Al Nafea, R., & Almaiah, M. A. (2021). Cyber Security Threats in Cloud: Literature Review. In *2021 Int. Conf. on Information Technology (ICIT)* (pp. 779–786). IEEE. https://doi.org/10.1109/icit52682.2021.9491638

30. Ковалюк, О. А., & Лучик, С. Д. (2024). Кібербезпека в хмарних обчисленнях: проблеми, загрози та підходи до забезпечення безпеки. In *13th Int. Sci. and Practical Conf. Innovations in Modern Education: European and Global Context*.

31. Шевченко, В. В. (2024). Хмарні рішення та кібербезпека: інноваційні підходи до захисту бізнесу у нестабільних умовах. *Econom. Synergy, 4*, 193–204. https://doi.org/10.53920/ES-2024-4-14

32. Бобренок, В. В., & Гуда, А. І. (2025). Аналіз open-source засобів для захисту ресурсів у хмарних середовищах. *Системні технології, 1(156)*, 32–38. https://doi.org/10.34185/1562-9945-1-156-2025-04

33. ISO/IEC 27000:2018 (2018). Information Technology. Security Techniques. Information Security Management Systems. Overview and Vocabulary. *Int. Organization for Standardization*, Geneva.

34. ISO/IEC 27001:2022 (2022). Information Security, Cybersecurity and Privacy Protection. Information Security Management Systems. Requirements. *Int. Organization for Standardization*, Geneva.

35. ISO/IEC 27017:2015 (2015). Information Technology. Security Techniques. Code of Practice for Information Security Controls based on ISO/IEC 27002 for Cloud Services. *Int. Organization for Standardization*, Geneva.

36. ISO/IEC 27018:2019 (2019). Information Technology. Security Techniques. Code of Practice for Protection of Personally Identifiable Information (PII) in Public Clouds Acting as PII Processors. *Int. Organization for Standardization*, Geneva.

37. ISO/IEC 27036-4:2016 (2016). Information Technology. Security Techniques. Information Security for Supplier Relationships. Part 4: Guidelines for Security of Cloud Services. *Int. Organization for Standardization*, Geneva.

38. Mell, P. M., & Grance, T. (2011). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-145

39. Jansen, W., & Grance, T. (2011). Guidelines on Security and Privacy in Public Cloud Computing. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-144

40. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST Cloud Computing Reference Architecture. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.500-292

41. Hu, V. C., Iorga, M., Bao, W., Li, A., Li, Q., & Gouglidis, A. (2020). General Access Control Guidance for Cloud Systems. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-210

42. Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-207

43. Cloud Security Alliance (2023). Treacherous 12: Top Threats to Cloud Computin. https://cloudsecurityalliance.org/research/top-threats/treacherous-12/ [Дата звернення 15.12.2025].

44. Cloud Security Alliance (2024). Top Threats to Cloud Computing 2024. https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024 [Дата звернення 15.12.2025].

45. IBM Security (2024). X-Force Threat Intelligence Index 2024. https://www.ibm.com/services/threat-intelligence?mhsrc=ibmsearch_a&mhq=IBM%20Security%26period%3B%20X-Force%20Threat%20Intelligence%20Index%202024 [Дата звернення 15.12.2025].

46. Verizon (2024). 2024 Data Breach Investigations Report. https://www.verizon.com/business/resources/reports/dbir/2024/ [Дата звернення 15.12.2025].

47. Flexera (2024). State of the Cloud Report 2024. https://www.flexera.com/resources/webinars/CM-WBNR-Cloud-Trends [Дата звернення 15.12.2025].

48. Palo Alto Networks (2021). Unit 42 Cloud Threat Report. Vol. 8. https://unit42.paloaltonetworks.com/cloud-threat-report-2h-2021/ [Дата звернення 15.12.2025].

49. Aqua Security (2024). Cloud Native Threat Report 2024. https://www.aquasec.com/blog/top-cloud-native-threats-and-vulnerabilities/ [Дата звернення 15.12.2025].

50. Check Point Software Technologies (2024). 2024 Security Report. https://www.checkpoint.com/press-releases/check-point-software-technologies-triumphs-in-miercoms-2024-next-generation-firewall-benchmark-report/ [Дата звернення 15.12.2025].

51. SentinelOne (2025). 50+ Cloud Security Statistics in 2025. https://www.sentinelone.com/cybersecurity-101/cloud-security/cloud-security-statistics/ [Дата звернення 15.12.2025].

52. Spacelift (2025). 100+ Cloud Security Statistics for 2025. https://spacelift.io/blog/cloud-security-statistics [Дата звернення 15.12.2025].

53. Qualys (2025). The State of Cloud & SaaS Security: Essential Statistics and Insights. https://blog.qualys.com/vulnerabilities-threat-research/2025/04/03/the-state-of-cloud-saas-security-essential-statistics-and-insights [Дата звернення 15.12.2025].

54. Fortinet (2024). Key Findings from the 2024 Cloud Security Report. https://www.fortinet.com/blog/industry-trends/key-findings-cloud-security-report-2024 [Дата звернення 15.12.2025].

55. Alzoubi, Y. I., Mishra, A. & Topcu, A. E. (2024). Research Trends in Deep Learning and Machine Learning for Cloud Computing Security. Artif. Intell. Rev., 57, 132. https://doi.org/10.1007/s10462-024-10776-5

56. Gupta, K., Saxena, D., Gupta, R., Kumar, J., & Singh, A. K. (2024). Fedmup: Federated Learning Driven Malicious User Prediction Model for Secure Data Distribution

in Cloud Environments. Appl. Soft Comput., 157, 111519. https://doi.org/10.1016/j.asoc.2024.111519

57. Dhinakaran, D., Selvaraj, D., Dharini, N., Raja, S. E., & Priya, C. (2024). Towards a Novel Privacy-Preserving Distributed Multiparty Data Outsourcing Scheme for Cloud Computing with Quantum Key Distribution. *arXiv*. https://doi.org/10.48550/arxiv.2407.18923

58. Banse, C., Kunz, I., Schneider, A., & Weiss, K. (2021). Cloud Property Graph: Connecting Cloud Security Assessments with Static Code Analysis. In *2021 IEEE 14th Int. Conf. on Cloud Computing (CLOUD)* (pp. 13–19). IEEE. https://doi.org/10.1109/cloud53861.2021.00014

59. Süß, F., Freimuth, M., Aßmuth, A., Weir, G. R., & Duncan, B. (2024). Cloud Security and Security Challenges Revisited. *arXiv*. https://doi.org/10.48550/arxiv.2405.11350

60. Saxena, D., Swain, S. R., Kumar, J., Patni, S., Gupta, K., Singh, A. K., & Lindenstruth, V. (2025). Secure Resource Management in Cloud Computing: Challenges, Strategies and Meta-Analysis. *arXiv*. https://doi.org/10.48550/arxiv.2502.03149

61. Moudni, M. E., & Ziyati, E. (2025). Advances and Challenges in Cloud Data Storage Security: A Systematic Review. *Int. J. Saf. Secur. Eng., 15(4)*, 653–675. https://doi.org/10.18280/ijsse.150403

62. Chauhan, M., & Shiaeles, S. (2023). An Analysis of Cloud Security Frameworks, Problems and Proposed Solutions. *Netw., 3(3)*, 422–450. https://doi.org/10.3390/network3030018

63. Kumar, B.S.A., Sah, B. (2024). Enhancing Cloud Security and Resource Management: A Comprehensive Review. In *Int. Conf. on Internet of Everything and Quantum Information Processing. IEQIP 2023. Lecture Notes in Networks and Systems, vol 1029*. Springer. https://doi.org/10.1007/978-3-031-61929-8_1

64. Ogundapo, A., Ezeaputa, V. N. (2024). Managing Security Risks of Public Cloud Computing. *Math. Comput. Sci., 9(5)*, 88–95. https://doi.org/10.11648/j.mcs.20240905.11

65. Khan, M. A., Gupta, P., Sultan, A. A., Singh, P., Shivam, S., & Lourens, M. (2024). Security in Cloud Computing: Issues and Challenges. *Int. J. Intell. Syst. Appl. Eng., 12(17s)*, 674–681.

66. Xu, G., Yu, W., Chen, Z., Zhang, H., Moulema, P., Fu, X., & Lu, C. (2014). A Cloud Computing-based System for Cyber Security Management. *Int. J. Parallel Emerg. Distrib. Syst., 30(1)*, 29–45. https://doi.org/10.1080/17445760.2014.925110

67. Takahashi, T., Kadobayashi, Y., & Fujiwara, H. (2010). Ontological Approach Toward Cybersecurity in Cloud Computing. In *3rd Int. Conf. on Security of Information and Networks* (pp. 100–109). ACM. https://doi.org/10.1145/1854099.1854121

68. Tissir, N., El Kafhali, S., & Aboutabit, N. (2021). Cybersecurity Management in Cloud Computing: Semantic Literature Review and Conceptual Framework Proposal. *J. Reliable Intell. Environ., 7(2)*, 69–84. https://doi.org/10.1007/s40860-020-00115-0

69. Gill, S. S., & Buyya, R. (2018). SECURE: Self-Protection Approach in Cloud Resource Management. *IEEE Cloud Comput., 5(1)*, 60–72. https://doi.org/10.1109/mcc.2018.011791715

70. Govindarajan, V., Sonani, R., & Patel, P. S. (2023). A Framework for Security-Aware Resource Management in Distributed Cloud Systems. *Academia Nexus J., 2(2)*.

71. Bhardwaj, A., & Goundar, S. (2019). A Framework to Define the Relationship between Cyber Security and Cloud Performance. *Comput. Fraud Secur., 2019(2)*, 12–19. https://doi.org/10.1016/s1361-3723(19)30020-x

72. Shri, S. J., Karthiyayini, G., Vignesh, L. S., Upender, T., Shobana, D., & Patra, L. S. K. (2024). Optimizing Resource Management and Data Security in Cloud Computing Environments. In *2024 3rd Int. Conf. for Advancement in Technology (ICONAT)* (pp. 1–6). IEEE. https://doi.org/10.1109/iconat61936.2024.10774916

73. Mishra, S. D., Dewangan, B. K., & Choudhury, T. (2020). Cyber security in cloud platforms. In *Information Security and Optimization* (pp. 159–170). Chapman and Hall/CRC. https://doi.org/10.1201/9781003045854-11

74. Jhawar, R., Piuri, V., & Samarati, P. (2012). Supporting Security Requirements for Resource Management in Cloud Computing. In *2012 IEEE 15th Int. Conf. on*

Computational Science and Engineering (pp. 170–177). IEEE. https://doi.org/10.1109/iccse.2012.32

75. Gudimetla, S. R., & Kotha, N. R. (2018). Cloud Security: Bridging the Gap between Cloud Engineering and Cybersecurity. *Webology, 15(2)*.

76. Jimmy, F. N. U. (2024). Cyber Security Vulnerabilities and Remediation through Cloud Security Tools. *J. Artif. Intell. Gener. Sci., 3(1)*, 196–233. https://doi.org/10.60087/jaigs.vol03.issue01.p233

77. Aldhyani, T. H., & Alkahtani, H. (2022). Artificial Intelligence Algorithm-based Economic Denial of Sustainability Attack Detection Systems: Cloud Computing Environments. *Sensors, 22(13)*, 4685. https://doi.org/10.3390/s22134685

78. Feng, D., Qin, Y., Feng, W., Li, W., Shang, K., & Ma, H. (2024). Survey of Research on Confidential Computing. *IET Commun., 18(9)*, 535–556. https://doi.org/10.1049/cmu2.12759

79. Jarkas, O., Ko, R., Dong, N., & Mahmud, R. (2025). A Container Security Survey: Exploits, Attacks, and Defenses. *ACM Comput. Surveys, 57(7)*, 1–36. https://doi.org/10.1145/3715001

80. VS, D. P., Sethuraman, S. C., & Khan, M. K. (2023). Container Security: Precaution Levels, Mitigation Strategies, and Research Perspectives. *Comput. Secur., 135*, 103490. https://doi.org/10.1016/j.cose.2023.103490

81. Cloud Security Alliance (2022). Five Surprising Findings From the 2022 Multi-Cloud Security Report. https://cloudsecurityalliance.org/blog/2022/02/22 [Дата звернення 15.12.2025].

82. A Technical Analysis of Confidential Computing (2022). A Publication of The Confidential Computing Consortium 2022, v. 1.3. https://www.xinxinfan.me/project/cccta/ [Дата звернення 15.12.2025].

83. Jordan Smith, E. K. (2025). AI-Powered Intrusion Detection Systems for Next-Generation Cloud Security.

84. Han, J., Zang, W., Liu, L., Chen, S., & Yu, M. (2018). Risk-Aware Multi-Objective Optimized Virtual Machine Placement in the Cloud. *J. Comput. Secur., 26(5)*, 707–730. https://doi.org/10.3233/jcs-171104

85. Saxena, D., Gupta, I., Kumar, J., Singh, A. K., & Wen, X. (2021). A Secure and Multiobjective Virtual Machine Placement Framework for Cloud Data Center. *IEEE Syst. J., 16(2)*, 3163–3174. https://doi.org/10.1109/jsyst.2021.3092521

86. Гулак Г. М., Скітер І. С., Гулак Є. Г., & Цирканюк Д. А. (2023). Базові засади побудови центру кібербезпеки об'єктів ядерної енергетики. In *14-а Всеукраїнська науково-практична конференція «Актуальні проблеми управління інформаційною безпекою держави»* (pp. 262–266).

87. Chen, J., Du, T., & Xiao, G. (2021). A Multi-Objective Optimization for Resource Allocation of Emergent Demands in Cloud Computing. *J. Cloud Comput., 10(1)*, 20. https://doi.org/10.1186/s13677-021-00237-7

88. Cao, L., Li, R., Ruan, X., & Liu, Y. (2022). Defending against Co-Residence Attack in Energy-Efficient Cloud: An Optimization based Real-Time Secure VM Allocation Strategy. *IEEE Access, 10*, 98549–98561. https://doi.org/10.1109/access.2022.3206021

89. Wilczyński, A., & Jakóbik, A. (2017). Using Polymatrix Extensive Stackelberg Games in Security-Aware Resource Allocation and Task Scheduling in Computational Clouds. *J. Telecommun. Inf. Technol., 1*, 71–80. https://doi.org/10.26636/jtit.2017.1.653

90. Jebalia, M., Ben Letaïfa, A., Hamdi, M., & Tabbane, S. (2015). An Overview on Coalitional Game-Theoretic Approaches for Resource Allocation in Cloud Computing Architectures. *Int. J. Cloud Comput., 4(1)*, 63–77. https://doi.org/10.1504/ijcc.2015.067708

91. Xu, X., & Yu, H. (2014). A Game Theory Approach to Fair and Efficient Resource Allocation in Cloud Computing. *Math. Probl. Eng., 2014(1)*, 915878. https://doi.org/10.1155/2014/915878

92. Ait Temghart, A., Marwan, M., & Baslam, M. (2023). Stackelberg Security Game for Optimizing Cybersecurity Decisions in Cloud Computing. *Secur. Commun. Netw., 2023(1)*, 1–13. https://doi.org/10.1155/2023/2811038

93. Makulov, K., Chikrii, A. A., Lakhno, V., Yagaliyeva, B., Malyukov, V., Malyukova, I., & Lakhno, M. (2025). Cloud Platform Selection Model in the Framework

of Differential Quality Game with Fuzzy Information. *IEEE Access, 13*, 22578–22589. https://doi.org/10.1109/access.2025.3535814

94. Shi, R., Zhang, J., Chu, W., Bao, Q., Jin, X., Gong, C., Zhu, Q., Yu, C., & Rosenberg, S. (2015). MDP and Machine Learning-based Cost-Optimization of Dynamic Resource Allocation for Network Function Virtualization. In *2015 IEEE Int. Conf. on Services Computing* (pp. 65–73). IEEE. https://doi.org/10.1109/scc.2015.19

95. Chen, L., Shen, H., & Sapra, K. (2014). Distributed Autonomous Virtual Resource Management in Datacenters using Finite-Markov Decision Process. In *ACM Symposium on Cloud Computing* (pp. 1–13). ACM. https://doi.org/10.1145/2670979.2671003

96. Kazeminajafabadi, A., & Imani, M. (2024). Optimal Joint Defense and Monitoring for Networks Security under Uncertainty: A POMDP-based Approach. *IET Inf. Secur., 2024(1)*, 7966713. https://doi.org/10.1049/2024/7966713

97. Malik, S. U., Anjum, A., Moqurrab, S. A., & Srivastava, G. (2022). Towards Enhanced Threat Modelling and Analysis using a Markov Decision Process. *Comput. Commun., 194*, 282–291. https://doi.org/10.1016/j.comcom.2022.07.038

98. Gu, X., Krish, M., Sohail, S., Thakur, S., Sabrina, F., & Fan, Z. (2025). From Integer Programming to Machine Learning: A Technical Review on Solving University Timetabling Problems. *Comput., 13(1)*, 10. https://doi.org/10.3390/computation13010010

99. Mangalagowri, R., & Venkataraman, R. (2023). Randomized MILP framework for Securing Virtual Machines from Malware Attacks. *Intell. Autom. Soft Comput., 35(2)*, 1565–1580. https://doi.org/10.32604/iasc.2023.026360

100. Ejaz, N., & Choudhury, S. (2025). A Comprehensive Survey of Linear, Integer, and Mixed-Integer Programming Approaches for Optimizing Resource Allocation in 5G and Beyond Networks. *arXiv*. https://doi.org/10.48550/arxiv.2502.15585

101. Koubàa, M., Regaieg, R., Karar, A. S., Nadeem, M., & Bahloul, F. (2024). A Multi-Objective Approach for Optimizing Virtual Machine Placement Using ILP and Tabu Search. *Telecom., 5(4)*, 1309–1331. https://doi.org/10.3390/telecom5040065

102. Іванченко, О. В. (2019). Оцінювання рівня безпеки системи SCADA критичної інфраструктури з урахуванням доступності кібернетичних і хмарних активів. *Syst. Technol., 2(58)*, 5–32. https://doi.org/10.32836/2521-6643-2019-2-58-1

103. Закаляк, Р. Ф. (2022). Критерії вибору постачальників хмарних послуг. In *Міжнародна наукова інтернет-конференція, вип. 71*.

104. Андрощук, О., Голобородько, М., Кондратенко, Ю., & Литовченко, Г. (2024). Критерії та рекомендації з оцінювання якості хмарних сервісів для інформаційної інфраструктури. *Сучасні інформаційні технології у сфері безпеки та оборони, 51(3)*, 60–70. https://doi.org/10.33099/2311-7249/2024-51-3-60-70

105. Закаляк, Р.Ф. (2022). Багатокритеріальний метод вибору постачальника хмарних послуг.

106. Кірпічніков, Ю., Головченко, О., Андрощук, О., Петрушен, М., & Розумний, О. (2023). Модель оцінювання альтернативних варіантів впровадження інформаційно-комунікаційних сервісів з використанням хмарних технологій для оборонних потреб. *Збірник наукових праць Центру воєнно-стратегічних досліджень НУОУ імені Івана Черняховського*, 79–88. https://doi.org/10.33099/2304-2745/2023-1-77/79-88

107. Аулов, І. Ф. (2015). Комплексний метод порівняння, оцінки та вибору механізмів управління ключовими даними користувачів у хмарних ІТС. *Прикладная радиоэлектроника, 14(4)*, 397–402.

108. Складанний, П. М., Костюк, Ю. В., Мазур, Н. П., & Пітайчук, М. А. (2025). Дослідження характеристик та продуктивності протоколів доступу до хмарних обчислювальних середовищ на основі універсального тестування. *Телекомунікаційні та інформаційні технології, 86(1)*, 61–74. https://doi.org/10.31673/2412-4338.2025.014649

109. Хомчак, М. (2025). Оцінка ризиків кібербезпеки для вибору хмарного провайдера. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(27)*, 549–559. https://doi.org/10.28925/2663-4023.2025.27.773

110. Schneider, S., Lansing, J., Gao, F., & Sunyaev, A. (2014). A Taxonomic Perspective on Certification Schemes: Development of a Taxonomy for Cloud Service

Certification Criteria. In *2014 47th Hawaii Int. Conf. on System Sciences* (pp. 4998–5007). IEEE. https://doi.org/10.1109/hicss.2014.614

111. Guerron, X., Abrahão, S., Insfran, E., Fernández-Diego, M., & González-Ladrón-De-Guevara, F. (2020). A Taxonomy of Quality Metrics for Cloud Services. *IEEE Access, 8*, 131461–131498. https://doi.org/10.1109/access.2020.3009079

112. Shevchenko, S., Zhdanova, Y., Dreis, Y., Kyrychok, R., & Tsyrkaniuk, D. (2023). Protection of Information in Telecommunication Medical Systems based on a Risk-Oriented Approach. In *Cybersecurity Providing in Information and Telecommunication Systems* (Vol. 3421, pp. 158–167).

113. Михайлів, В. І. (2015). Експериментальні дослідження інформаційної технології для захисту даних у хмарних обчислювальних системах. *Актуальні проблеми автоматизації та інформаційних технологій, 19*, 52–66.

114. Кобевко, А. Т., & Тимченко, О. В. (2019). Особливості DDoS-атак на хмарні сервіси. *Mech. Eng., 11(3)*, 1–15.

115. Hussain, S. A., Fatima, M., Saeed, A., Raza, I., & Shahzad, R. K. (2017). Multilevel Classification of Security Concerns in Cloud Computing. *Appl. Comput. Inf., 13(1)*, 57–65. https://doi.org/10.1016/j.aci.2016.03.001

116. Hosseini, S., & Vakili, R. (2019). Game Theory Approach for Detecting Vulnerable Data Centers in Cloud Computing Network. *Int. J. Commun. Syst., 32(8)*, e3938. https://doi.org/10.1002/dac.3938

117. Kakkad, V., Shah, H., Patel, R., & Doshi, N. (2019). A Comparative Study of Applications of Game Theory in Cyber Security and Cloud Computing. *Procedia Comput. Sci., 155*, 680–685. https://doi.org/10.1016/j.procs.2019.08.097

118. Banerjee, K., Gupta, R. R., Vyas, K., & Mishra, B. (2020). Exploring Alternatives to Softmax Function. *arXiv*. https://doi.org/10.48550/arxiv.2011.11538

119. Sun, Y., Lin, F., & Xu, H. (2018). Multi-Objective Optimization of Resource Scheduling in Fog Computing using an Improved NSGA-II. *Wirel. Pers. Commun., 102*, 1369–1385. https://doi.org/10.1007/s11277-017-5200-5

120. Tsai, J. T., Fang, J. C., & Chou, J. H. (2013). Optimized Task Scheduling and Resource Allocation on Cloud Computing Environment using Improved Differential

Evolution Algorithm. *Comput. Oper. Res., 40(12)*, 3045–3055. https://doi.org/10.1016/j.cor.2013.06.012

121. Chołodowicz, E., & Orłowski, P. (2017). Comparison of SPEA2 and NSGA-II Applied to Automatic Inventory Control System using Hypervolume Indicator. *Stud. Inf. Control, 26(1)*, 67–74. https://doi.org/10.24846/v26i1y201708

122. Tian, Y., Zhang, X., Cheng, R., & Jin, Y. (2016). A Multi-Objective Evolutionary Algorithm based on an Enhanced Inverted Generational Distance Metric. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 5222–5229). IEEE. https://doi.org/10.1109/cec.2016.7748352

123. Ramesh, S., Kannan, S., & Baskar, S. (2012). Application of Modified NSGA-II Algorithm to Multi-Objective Reactive Power Planning. *Appl. Soft Comput., 12(2)*, 741–753. https://doi.org/10.1016/j.asoc.2011.09.015

124. Glorfeld, L. W., & Palko, J. (1988). A Comparison of Novice Algorithm Composition Performance Using Flowcharting or Pseudocode as the Design Tool. *J. Res. Comput. Edu., 21(1)*, 82–96. https://doi.org/10.1080/08886504.1988.10781862

125. Vayadande, K., Bailke, P. A., Dombale, A. B., Dange, V. R., & Kulkarni, A. M. (2024). Converting Pseudo Code to Code: A Review. In *How Machine Learning is Innovating Today's World* (pp. 57–68). Wiley. https://doi.org/10.1002/9781394214167.ch6

126. 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

127. Choi, J., Choi, C., Lynn, H. M., & Kim, P. (2015). Ontology based APT Attack Behavior Analysis in Cloud Computing. In *2015 10th Int. Conf. on Broadband and Wireless Computing, Communication and Applications (BWCCA)* (pp. 375–379). IEEE. https://doi.org/10.1109/bwcca.2015.69

128. Alshaikh, A., Alanesi, M., Yang, D., & Alshaikh, A. (2023). Advanced Techniques for Cyber Threat Intelligence-based APT Detection and Mitigation in Cloud

Environments. In *Int. Conf. on Cyber Security, Artificial Intelligence, and Digital Economy* (Vol. 12718, pp. 147–157). SPIE. https://doi.org/10.1117/12.2681627

129. Шабала, Є., & Корнійчук, Б. (2024). Методологія оцінювання безпеки IoT на промислових об'єктах. *Управління розвитком складних систем, 60*, 146–155. https://doi.org/10.32347/2412-9933.2024.60.146-155

130. Стефурак, О. Р., Тихонов, Ю. О., Лаптєв, О. А., & Зозуля, С. А. (2020). Удосконалення стохастичної моделі з метою визначення загроз пошкодження або несанкціонованого витоку інформації. *Сучасний захист інформації, 2*, 19–26. https://doi.org/10.31673/2409-7292.2020.021926

131. Li, B., Chen, Y., Huang, S., Yao, R., Xia, Y., & Mei, S. (2019). Graphical Evolutionary Game Model of Virus-based Intrusion to Power System for Long-Term Cyber-Security Risk Evaluation. *IEEE Access, 7*, 178605–178617. https://doi.org/10.1109/access.2019.2958856

132. Zhang, H., Tan, J., Liu, X., Huang, S., Hu, H., & Zhang, Y. (2022). Cybersecurity Threat Assessment Integrating Qualitative Differential and Evolutionary Games. *IEEE Trans. Netw. Serv. Manag., 19(3)*, 3425–3437. https://doi.org/10.1109/tnsm.2022.3166348

133. Höfer, C. N., & Karagiannis, G. (2011). Cloud Computing Services: Taxonomy and Comparison. *J. Internet Serv. Appl., 2(2)*, 81–94. https://doi.org/10.1007/s13174-011-0027-x

134. Li, P., & Yang, X. (2019). On Dynamic Recovery of Cloud Storage System under Advanced Persistent Threats. *IEEE Access, 7*, 103556–103569. https://doi.org/10.1109/access.2019.2932020

135. Winter, E. (2002). Chapter 53 The Shapley Value. In Handbook of Game Theory with Economic Applications (pp. 2025–2054). Elsevier. https://doi.org/10.1016/s1574-0005(02)03016-3

136. Faigle, U., & Kern, W. (1992). The Shapley Value for Cooperative Games under Precedence Constraints. *Int. J. Game Theor., 21*, 249–266.

137. 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

138. Steinert, T., Kuschewski, M., & Leis, V. (2026). Cloudspecs: Cloud Hardware Evolution Through the Looking Glass. [Preprint].

139. Garg, Mr. Y., & Gupta, Ms. B. (2023). Performance Analysis and Comparison of the Micro Virtual Machines Provided by the Top Cloud Vendors. *Int. J. Res. Publ. Rev., 04(02)*, 1326–1333. https://doi.org/10.55248/gengpi.2023.4229

140. Sajjad, M., Arshad, A., & Khan, A. S. (2018). Performance Evaluation of Cloud Computing Resources. *Int. J. Adv. Comput. Sci. Appl., 9(8)*. https://doi.org/10.14569/ijacsa.2018.090824

141. Mason, K., Duggan, M., Barrett, E., Duggan, J., & Howley, E. (2018). Predicting Host CPU Utilization in the Cloud using Evolutionary Neural Networks. *Futur. Gener. Comput. Syst., 86*, 162–173. https://doi.org/10.1016/j.future.2018.03.040

142. Malawski, M., Gajek, A., Zima, A., Balis, B., & Figiela, K. (2020). Serverless Execution of Scientific Workflows: Experiments with Hyperflow, AWS Lambda and Google Cloud Functions. *Futur. Gener. Comput. Syst., 110*, 502–514. https://doi.org/10.1016/j.future.2017.10.029

143. Chowdhury, S. (2025). Serverless Computing: A Comparative Performance Analysis between AWS Lambda, Google Cloud Functions, and Microsoft Azure Functions. https://www.doria.fi/handle/10024/190693 [Дата звернення 15.12.2025].

144. Ali, M. A. B. (2023). Serverless Computing for Big Data Analytics: Performance and Cost Analysis of AWS Lambda and Google Cloud Functions. *J. Data Mining, Knowl. Discov. Decis. Support Syst., 13(2)*, 1–11.

145. Baskar, R., Niranjanamurthy, M., Vinoth, N., Chitra, D., & Pushparaj, M. R. R. (2025). Enhancing Microservice Scalability with Serverless Architectures: An Analysis of AWS Lambda and Google Cloud Functions. In *2025 Int. Conf. on Emerging Technologies in Engineering Applications (ICETEA)* (pp. 1–6). IEEE. https://doi.org/10.1109/icetea64585.2025.11099699

146. Tchernykh, A., Schwiegelsohn, U., Talbi, E. G., & Babenko, M. (2019). Towards Understanding Uncertainty in Cloud Computing with Risks of Confidentiality,

Integrity, and Availability. *J. Comput. Sci., 36*, 100581. https://doi.org/10.1016/j.jocs.2016.11.011

147. Bhushan, K., & Gupta, B. B. (2017). Security Challenges in Cloud Computing: State-of-Art. *Int. J. Big Data Intell., 4(2)*, 81–107. https://doi.org/10.1504/ijbdi.2017.083116

148. Iosup, A., Yigitbasi, N., & Epema, D. (2011). On the Performance Variability of Production Cloud Services. In *2011 11th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing* (pp. 104–113). IEEE. https://doi.org/10.1109/ccgrid.2011.22

149. 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

150. Павлов, І. М., & Толюпа, С. В. (2014). Аналіз підходів моделювання процесів прийняття рішень при проектуванні систем захисту інформації. *Сучасний захист інформації, 2*, 96–104.

151. Teymourifar, A., Rodrigues, A. M., & Ferreira, J. S. (2020). A Comparison between NSGA-II and NSGA-III to Solve Multi-Objective Sectorization Problems based on Statistical Parameter Tuning. In *2020 24th Int. Conf. on Circuits, Systems, Communications and Computers (CSCC)* (pp. 64–74). IEEE. https://doi.org/10.1109/cscc49995.2020.00020

152. Driscoll, W. C. (1996). Robustness of the ANOVA and Tukey-Kramer Statistical Tests. *Comput. Ind. Eng., 31(1–2)*, 265–268. https://doi.org/10.1016/0360-8352(96)00127-1

153. Цирканюк, Д. (2025). Метод багатокритеріальної оптимізації безпеки хмарних обчислень на основі модифікованого алгоритму NSGA-II. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 2(30)*, 692–714. https://doi.org/10.28925/2663-4023.2025.30.983

154. 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

155. 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

156. Цирканюк, Д. (2025). Метод CoopEvo-CloudSe для оптимізації обчислювальних ресурсів хмарних систем для підвищення безпеки. *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», 3(31)*, 872–887. https://doi.org/10.28925/2663-4023.2025.31.1077

ДОДАТОК А
ОСНОВНИЙ КОД ДЛЯ МЕТОДУ КООПЕРАТИВНО-ЕВОЛЮЦІЙНОГО РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ У ХМАРНОМУ СЕРЕДОВИЩІ З УРАХУВАННЯМ РИЗИКУ

```
# nsga2_final_solution_corrected.py
import os
import time
import json
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.spatial import ConvexHull
from matplotlib import cm
import warnings

warnings.filterwarnings('ignore')

from pymoo.core.problem import Problem
from pymoo.optimize import minimize
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.operators.sampling.rnd import IntegerRandomSampling
from pymoo.operators.crossover.sbx import SBX
from pymoo.operators.mutation.pm import PM
from pymoo.termination import get_termination

# КОНФІГУРАЦІЯ
DATA_PATH =
r"C:\Users\38050\PythonProject\Cloud_resource_optimization_experiment\data2"
```

```
    RESULTS_DIR = r"C:\Users\38050\PythonProject\Cloud_resource_optimization_experiment\results_final"
    PLOTS_DIR = os.path.join(RESULTS_DIR, "plots")

    # ПАРАМЕТРИ
    POP_SIZE = 100
    N_GEN = 300
    SEED = 42

    # ДОПОМІЖНІ КЛАСИ
    class Node:
        def __init__(self, node_id, cpu, memory, speed, cost_per_hour, base_risk, risk_volatility, coalition):
            self.node_id = node_id
            self.cpu = cpu
            self.memory = memory
            self.speed = speed
            self.cost_per_hour = cost_per_hour
            self.base_risk = base_risk
            self.risk_volatility = risk_volatility
            self.coalition = coalition

    class Task:
        def __init__(self, task_id, req_cpu, req_mem, base_duration, priority, security_level):
            self.task_id = task_id
            self.req_cpu = req_cpu
            self.req_mem = req_mem
            self.base_duration = base_duration
```

```
        self.priority = priority
        self.security_level = security_level

class RiskTimeline:
    def __init__(self, df_risk):
        self.df = df_risk.copy()
        self.df['time_step'] = self.df['time_step'].astype(int)
        self.max_time = int(self.df['time_step'].max())

    def get_lambda(self, node_id, t):
        if self.max_time == 0:
            t_idx = 0
        else:
            t_idx = int(t) % (self.max_time + 1)
        row = self.df[(self.df['node_id'] == node_id) & (self.df['time_step'] == t_idx)]
        if row.empty:
            row2 = self.df[self.df['node_id'] == node_id]
            if row2.empty:
                return 0.0
            return float(row2['lambda_t'].mean())
        return float(row['lambda_t'].iloc[0])

# ФУНКЦІЇ ЗАВАНТАЖЕННЯ
def load_problem_data(data_path):
    nodes_df = pd.read_csv(os.path.join(data_path, "nodes.csv"))
    tasks_df = pd.read_csv(os.path.join(data_path, "tasks.csv"))
    risk_df = pd.read_csv(os.path.join(data_path, "risk_timeline.csv"))

    nodes = []
    for _, r in nodes_df.iterrows():
```

```
        nodes.append(Node(
            node_id=r['node_id'],
            cpu=int(r['cpu_cores']),
            memory=float(r['memory_gb']),
            speed=float(r['base_speed']),
            cost_per_hour=float(r['cost_per_hour']),
            base_risk=float(r['base_risk']),
            risk_volatility=float(r['risk_volatility']),
            coalition=int(r['coalition_role'])
        ))

    tasks = []
    for _, r in tasks_df.iterrows():
        tasks.append(Task(
            task_id=r['task_id'],
            req_cpu=int(r['required_cpu']),
            req_mem=float(r['required_memory']),
            base_duration=float(r['base_duration']),
            priority=int(r['priority']),
            security_level=int(r['security_level'])
        ))

    risk_timeline = RiskTimeline(risk_df)

    print(f"📊 Завантажено: {len(nodes)} вузлів, {len(tasks)} завдань")
    return nodes, tasks, risk_timeline

def evaluate_assignment_decision(vec, nodes, tasks, risk_tl):
    n_tasks = len(tasks)
    pernode_tasks = {i: [] for i in range(len(nodes))}
```

```
for t_idx, n_idx in enumerate(vec):
    pernode_tasks[int(n_idx)].append(t_idx)

total_time_sec = 0.0
total_cost = 0.0
total_risk = 0.0
coalition_bonus = 0.0
coalition_nodes_used = set()

for n_idx, task_indices in pernode_tasks.items():
    node = nodes[n_idx]
    if len(task_indices) == 0:
        continue

    cur_time = 0.0
    total_time_on_node = 0.0

    for t_idx in task_indices:
        task = tasks[t_idx]
        duration = task.base_duration / node.speed
        epoch = 60.0
        t_step = int(math.floor(cur_time / epoch))
        lambda_t = risk_tl.get_lambda(node.node_id, t_step)
        total_risk += lambda_t * task.security_level
        total_time_on_node += duration
        cur_time += duration

    total_time_sec += total_time_on_node
    total_cost += node.cost_per_hour * (total_time_on_node / 3600.0)
```

```
            if node.coalition == 1 and len(task_indices) > 0:
                coalition_nodes_used.add(n_idx)

        k = len(coalition_nodes_used)
        if k > 0:
            alpha = 1.0
            base_bonus = alpha * math.log(1 + k)
            sum_bonus = 0.0
            for n_idx in coalition_nodes_used:
                for t_idx in pernode_tasks[n_idx]:
                    sum_bonus += tasks[t_idx].security_level * base_bonus
            coalition_bonus = sum_bonus

        # Нормалізація
        f1 = total_risk
        f2 = total_time_sec / 1000.0
        f3 = total_cost * 50.0
        f4 = coalition_bonus / 5.0

        return f1, f2, f3, f4

# ЗАДАЧА ОПТИМІЗАЦІЇ
class HybridAssignmentProblem(Problem):
    def __init__(self, nodes, tasks, risk_tl):
        n_var = len(tasks)
        n_obj = 4
        xl = np.zeros(n_var, dtype=int)
        xu = np.full(n_var, len(nodes) - 1, dtype=int)
        self.nodes = nodes
```

```
            self.tasks = tasks
            self.risk_tl = risk_tl
            super().__init__(n_var=n_var, n_obj=n_obj, n_constr=0,
                             xl=xl, xu=xu, elementwise=False)

        def _evaluate(self, X, out, *args, **kwargs):
            pop = X.shape[0]
            F = np.zeros((pop, 4), dtype=float)

            for i in range(pop):
                vec = X[i].astype(int)
                f1, f2, f3, f4 = evaluate_assignment_decision(vec, self.nodes, self.tasks, self.risk_tl)
                F[i, 0] = f1
                F[i, 1] = f2
                F[i, 2] = f3
                F[i, 3] = -f4

            out["F"] = F

    # МЕТРИКИ ЯКОСТІ (ВИПРАВЛЕНІ)
    def compute_hypervolume(front, reference_point):
        """Обчислення гіпероб'єму (правильна реалізація)"""
        if len(front) == 0:
            return 0.0

        # Створюємо копію для безпеки
        front_copy = front.copy()

        # Нормалізуємо значення для стабільності обчислень
```

```
# Знаходимо мінімальні значення для кожної метрики
min_vals = np.min(front_copy, axis=0)
max_vals = np.max(front_copy, axis=0)

# Якщо всі значення однакові, повертаємо 0
if np.all(max_vals - min_vals < 1e-10):
    return 0.0

# Нормалізуємо до [0, 1]
front_norm = (front_copy - min_vals) / (max_vals - min_vals + 1e-10)

# Обчислюємо гіпероб'єм
hv = 0.0
n_points, n_obj = front_norm.shape

# Сортуємо за першим критерієм
sorted_idx = np.argsort(front_norm[:, 0])
sorted_front = front_norm[sorted_idx]

# Обчислюємо об'єм
for i in range(n_points):
    # Знаходимо "зайнятий" об'єм для цієї точки
    if i == 0:
        volume = np.prod(1.0 - sorted_front[i])
    else:
        # Знаходимо мінімальні значення попередніх точок
        prev_min = np.min(sorted_front[:i], axis=0)
        # Обчислюємо "вільний" простір
        free_space = np.maximum(1.0 - np.maximum(prev_min, sorted_front[i]), 0)
        volume = np.prod(free_space)
```

```
            hv += volume

        return hv

    def compute_igd(pareto_front, reference_front):
        """Обчислення зворотної генераційної відстані"""
        if len(pareto_front) == 0 or len(reference_front) == 0:
            return float('inf')

        # Нормалізуємо дані
        min_vals = np.min(reference_front, axis=0)
        max_vals = np.max(reference_front, axis=0)

        if np.all(max_vals - min_vals < 1e-10):
            return 0.0

        pareto_norm = (pareto_front - min_vals) / (max_vals - min_vals + 1e-10)
        reference_norm = (reference_front - min_vals) / (max_vals - min_vals + 1e-10)

        distances = []
        for ref_point in reference_norm:
            min_dist = np.min(np.linalg.norm(pareto_norm - ref_point, axis=1))
            distances.append(min_dist)

        return np.mean(distances)

    def compute_spacing(pareto_front):
        """Обчислення рівномірності розподілу"""
        n_points = len(pareto_front)
        if n_points <= 2:
```

```
        return 0.0

    # Нормалізуємо дані
    min_vals = np.min(pareto_front, axis=0)
    max_vals = np.max(pareto_front, axis=0)

    if np.all(max_vals - min_vals < 1e-10):
        return 0.0

    front_norm = (pareto_front - min_vals) / (max_vals - min_vals + 1e-10)

    distances = []
    for i in range(n_points):
        # Знаходимо найближчу точку (окрім самої себе)
        dists = np.linalg.norm(front_norm - front_norm[i], axis=1)
        dists[i] = np.inf  # Виключаємо саму точку
        min_dist = np.min(dists)
        distances.append(min_dist)

    mean_dist = np.mean(distances)
    if mean_dist == 0:
        return 0.0

    spacing = np.sqrt(np.sum((np.array(distances) - mean_dist) ** 2) / (n_points - 1))
    return spacing

# ВІЗУАЛІЗАЦІЯ
def plot_pareto_2d(front, front_labels, filename):
    """2D візуалізація фронту Парето"""
    fig, axes = plt.subplots(2, 3, figsize=(18, 12))
```

```
axes = axes.flatten()

# Комбінації критеріїв для 2D графіків
combinations = [(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)]
titles = ['Ризик vs Час', 'Ризик vs Вартість', 'Ризик vs Коаліція',
          'Час vs Вартість', 'Час vs Коаліція', 'Вартість vs Коаліція']

# Нормалізація для колірної шкали
colors = front[:, 3]  # Використовуємо коаліцію для кольору
if np.max(colors) - np.min(colors) > 0:
    colors = (colors - np.min(colors)) / (np.max(colors) - np.min(colors))

for idx, (x_idx, y_idx) in enumerate(combinations):
    ax = axes[idx]

    # Точки фронту
    scatter = ax.scatter(front[:, x_idx], front[:, y_idx],
                         c=colors, cmap='viridis', s=100, alpha=0.8,
                         edgecolors='black', linewidth=0.5)

    # Сортуємо для побудови лінії
    sorted_indices = np.argsort(front[:, x_idx])
    sorted_front = front[sorted_indices]

    # Виразна червона лінія Парето-фронту
    if len(sorted_front) > 1:
        ax.plot(sorted_front[:, x_idx], sorted_front[:, y_idx],
                'r-', linewidth=3, alpha=0.8, label='Фронт Парето')

    # Контури опуклої оболонки
```

```
            try:
                if len(front) >= 3:
                    hull = ConvexHull(front[:, [x_idx, y_idx]])
                    for simplex in hull.simplices:
                        ax.plot(front[simplex, x_idx], front[simplex, y_idx],
                                'b--', linewidth=1, alpha=0.5)
            except:
                pass

            # Виділені крайні рішення
            if len(front) >= 4:
                extreme_points = []
                extreme_points.append(np.argmin(front[:, x_idx]))
                extreme_points.append(np.argmax(front[:, x_idx]))
                extreme_points.append(np.argmin(front[:, y_idx]))
                extreme_points.append(np.argmax(front[:, y_idx]))

                extreme_points = list(set(extreme_points))
                ax.scatter(front[extreme_points, x_idx], front[extreme_points, y_idx],
                           c='red', s=200, marker='*', edgecolors='gold', linewidth=2,
                           label='Крайні рішення', zorder=5)

                # Підписи для крайніх точок
                for i, point_idx in enumerate(extreme_points):
                    ax.annotate(f'P{i + 1}',
                                xy=(front[point_idx, x_idx], front[point_idx, y_idx]),
                                xytext=(5, 5), textcoords='offset points',
                                fontsize=10, fontweight='bold',
                                bbox=dict(boxstyle="round,pad=0.3",          facecolor="yellow",
alpha=0.7))
```

```
        ax.set_xlabel(front_labels[x_idx], fontsize=12, fontweight='bold')
        ax.set_ylabel(front_labels[y_idx], fontsize=12, fontweight='bold')
        ax.set_title(titles[idx], fontsize=14, fontweight='bold')
        ax.grid(True, alpha=0.3)
        if idx == 0:
            ax.legend(loc='best')

    plt.tight_layout()
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    plt.close()

    print(f"✅ Збережено 2D графік: {filename}")


def plot_pareto_3d(front, front_labels, filename):
    """3D візуалізація фронту Парето"""
    if len(front) < 3:
        print(f"⚠ Недостатньо точок для 3D графіка: {len(front)}")
        return

    fig = plt.figure(figsize=(15, 10))
    ax = fig.add_subplot(111, projection='3d')

    # Кольорова градація за 4-м критерієм (коаліція)
    colors = front[:, 3]
    if np.max(colors) - np.min(colors) > 0:
        colors = (colors - np.min(colors)) / (np.max(colors) - np.min(colors))

    # 3D точки
```

```
        scatter = ax.scatter(front[:, 0], front[:, 1], front[:, 2],
                             c=colors, cmap='coolwarm', s=100,
                             depthshade=True, alpha=0.8, edgecolors='black', linewidth=0.5)

        # Крайні точки
        if len(front) >= 4:
            extreme_points = []
            for i in range(3):
                extreme_points.append(np.argmin(front[:, i]))
                extreme_points.append(np.argmax(front[:, i]))

            extreme_points = list(set(extreme_points))
            ax.scatter(front[extreme_points, 0], front[extreme_points, 1],
front[extreme_points, 2],
                       c='gold', s=300, marker='*', edgecolors='red', linewidth=2,
                       label='Крайні рішення')

        ax.set_xlabel(front_labels[0], fontsize=12, fontweight='bold', labelpad=10)
        ax.set_ylabel(front_labels[1], fontsize=12, fontweight='bold', labelpad=10)
        ax.set_zlabel(front_labels[2], fontsize=12, fontweight='bold', labelpad=10)
        ax.set_title('3D фронт Парето (колір = коаліційна вигода)',
                     fontsize=16, fontweight='bold', pad=20)

        # Додаємо колірну шкалу
        cbar = fig.colorbar(scatter, ax=ax, pad=0.1)
        cbar.set_label('Коаліційна вигода (норм.)', fontsize=12, fontweight='bold')

        # Налаштування огляду
        ax.view_init(elev=25, azim=45)
        ax.grid(True, alpha=0.3)
```

```
    if len(front) >= 4:
        ax.legend(loc='upper right')

    plt.tight_layout()
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    plt.close()

    print(f"✅ Збережено 3D графік: {filename}")


def plot_pareto_radar(front, front_labels, filename):
    """Радарна діаграма для порівняння крайніх рішень"""
    if len(front) < 3:
        print(f"⚠ Недостатньо точок для радарної діаграми: {len(front)}")
        return

    fig = plt.figure(figsize=(12, 10))

    # Нормалізація значень
    normalized_front = front.copy()
    for i in range(4):
        min_val = np.min(front[:, i])
        max_val = np.max(front[:, i])
        if max_val > min_val:
            normalized_front[:, i] = (front[:, i] - min_val) / (max_val - min_val)
        else:
            normalized_front[:, i] = 0.5

    # Вибірка кількох представницьких рішень
    n_samples = min(6, len(front))
```

```
if len(front) > n_samples:
    step = len(front) // n_samples
    sample_indices = list(range(0, len(front), step))[:n_samples]
else:
    sample_indices = list(range(len(front)))

# Налаштування для радарної діаграми
angles = np.linspace(0, 2 * np.pi, 4, endpoint=False).tolist()
angles += angles[:1]  # Замикаємо коло

ax = fig.add_subplot(111, polar=True)

colors = plt.cm.tab10(np.linspace(0, 1, len(sample_indices)))

for idx, point_idx in enumerate(sample_indices):
    values = normalized_front[point_idx].tolist()
    values += values[:1]  # Замикаємо коло

    ax.plot(angles, values, 'o-', linewidth=2, color=colors[idx],
            label=f'Рішення {point_idx + 1}', alpha=0.7)
    ax.fill(angles, values, alpha=0.1, color=colors[idx])

ax.set_xticks(angles[:-1])
ax.set_xticklabels(front_labels, fontsize=12, fontweight='bold')
ax.set_ylim(0, 1)
ax.set_yticks([0, 0.25, 0.5, 0.75, 1])
ax.set_yticklabels(['0%', '25%', '50%', '75%', '100%'], fontsize=10)
ax.set_title('Радарна діаграма: порівняння рішень',
             fontsize=16, fontweight='bold', pad=20)
ax.grid(True, alpha=0.3)
```

```
    ax.legend(loc='upper right', bbox_to_anchor=(1.3, 1.0))

    plt.tight_layout()
    plt.savefig(filename, dpi=300, bbox_inches='tight')
    plt.close()

    print(f"✅ Збережено радарну діаграму: {filename}")

# ОСНОВНА ФУНКЦІЯ ()
def run_nsga2_with_visualization():
    print("🚀 ЗАПУСК NSGA-II З ПОКРАЩЕНОЮ ВІЗУАЛІЗАЦІЄЮ")
    print("=" * 60)

    # Створення директорій
    os.makedirs(RESULTS_DIR, exist_ok=True)
    os.makedirs(PLOTS_DIR, exist_ok=True)

    # Завантаження даних
    nodes, tasks, risk_tl = load_problem_data(DATA_PATH)

    # Створення проблеми
    problem = HybridAssignmentProblem(nodes, tasks, risk_tl)

    # Налаштування алгоритму
    sampling = IntegerRandomSampling()
    crossover = SBX(prob=0.95, eta=20)
    mutation = PM(prob=0.4, eta=25)

    algorithm = NSGA2(
        pop_size=POP_SIZE,
```

```
        sampling=sampling,
        crossover=crossover,
        mutation=mutation,
        eliminate_duplicates=True,
        save_history=True
    )

    termination = get_termination("n_gen", N_GEN)

    # Запуск оптимізації
    np.random.seed(SEED)
    start = time.time()
    res = minimize(problem, algorithm, termination, seed=SEED, verbose=True)
    elapsed = time.time() - start

    print(f"\n✅ Оптимізація завершена за {elapsed:.2f} секунд")

    # Отримання результатів
    X = res.X.astype(int)
    F = res.F.copy()
    F[:, 3] = -F[:, 3]  # Відновлення оригінального F4

    print(f"\n📊 Знайдено {len(F)} рішень")

    # Пошук фронту Парето
    print("\n🔍 ПОШУК FRONTO PARETO...")

    def is_dominated(A, B):
        better_or_equal = (A[0] <= B[0] and A[1] <= B[1] and A[2] <= B[2] and A[3]
>= B[3])
```

```
    strictly_better = (A[0] < B[0] or A[1] < B[1] or A[2] < B[2] or A[3] > B[3])
    return better_or_equal and strictly_better

n = len(F)
dominated = np.zeros(n, dtype=bool)

for i in range(n):
    if not dominated[i]:
        for j in range(n):
            if i != j and not dominated[j]:
                if is_dominated(F[i], F[j]):
                    dominated[j] = True

pareto_indices = np.where(~dominated)[0]

# Додаткова фільтрація при необхідності
if len(pareto_indices) == n or len(pareto_indices) > 50:
    print("⚠ Застосовуємо додаткову фільтрацію...")
    ranks = np.zeros(n)
    for i in range(4):
        if i == 3:  # F4 максимізується
            ranks += np.argsort(-F[:, i])
        else:  # F1-F3 мінімізуються
            ranks += np.argsort(F[:, i])

    top_n = min(30, n)  # Обмежуємо до 30 найкращих рішень
    pareto_indices = np.argsort(ranks)[:top_n]

F_pareto = F[pareto_indices]
X_pareto = X[pareto_indices]
```

```
print(f"✅ Знайдено {len(F_pareto)} недомінованих рішень")

# ОБЧИСЛЕННЯ МЕТРИК ЯКОСТІ (ВИПРАВЛЕНО)
print("\n📈 ОБЧИСЛЕННЯ МЕТРИК ЯКОСТІ...")

# Для метрик використовуємо нормалізовані значення
# Перетворюємо всі критерії до мінімізації для обчислення метрик
F_pareto_min = F_pareto.copy()
F_pareto_min[:, 3] = -F_pareto_min[:, 3]  # Мінімізуємо -коаліція

# Нормалізуємо дані
min_vals = np.min(F_pareto_min, axis=0)
max_vals = np.max(F_pareto_min, axis=0)
range_vals = max_vals - min_vals

# Уникаємо ділення на нуль
range_vals[range_vals < 1e-10] = 1.0

F_pareto_norm = (F_pareto_min - min_vals) / range_vals

# Обчислюємо метрики
# Hypervolume
reference_point = np.ones(4) * 1.1  # Референсна точка трохи вище за максимум
hv = compute_hypervolume(F_pareto_norm, reference_point)

# IGD - потрібен еталонний фронт
# Створюємо штучний еталонний фронт (ідеальні точки)
n_ref = 100
```

```
reference_front = []
for i in range(4):
    # Створюємо рівномірний розподіл
    ref_values = np.linspace(0, 1, n_ref)
    reference_front.append(ref_values)

reference_front = np.array(reference_front).T
igd = compute_igd(F_pareto_norm, reference_front)

# Spacing
spacing = compute_spacing(F_pareto_norm)

print(f"📊 МЕТРИКИ ЯКОСТІ:")
print(f"  Hypervolume (HV):     {hv:.6f}")
print(f"  IGD:                  {igd:.6f}")
print(f"  Spacing:              {spacing:.6f}")

# ЗБЕРЕЖЕННЯ МЕТРИК ()
# Створюємо DataFrame для метрик
metrics_data = [
    {
        'algorithm': 'NSGA2_improved',
        'pop_size': POP_SIZE,
        'n_gen': N_GEN,
        'seed': SEED,
        'runtime_sec': elapsed,
        'pareto_front_size': len(F_pareto),
        'metric': 'Hypervolume',
        'value': hv
    },
```

```
    {
        'algorithm': 'NSGA2_improved',
        'pop_size': POP_SIZE,
        'n_gen': N_GEN,
        'seed': SEED,
        'runtime_sec': elapsed,
        'pareto_front_size': len(F_pareto),
        'metric': 'IGD',
        'value': igd
    },
    {
        'algorithm': 'NSGA2_improved',
        'pop_size': POP_SIZE,
        'n_gen': N_GEN,
        'seed': SEED,
        'runtime_sec': elapsed,
        'pareto_front_size': len(F_pareto),
        'metric': 'Spacing',
        'value': spacing
    }
]

metrics_df = pd.DataFrame(metrics_data)
metrics_path = os.path.join(RESULTS_DIR, "quality_metrics.csv")
metrics_df.to_csv(metrics_path, index=False)
print(f"💾 Метрики збережено: {metrics_path}")

# ЗБЕРЕЖЕННЯ РЕЗУЛЬТАТІВ
rows = []
for i in range(X_pareto.shape[0]):
```

```
    rows.append({
        "algorithm": "NSGA2_improved",
        "seed": SEED,
        "pop_size": POP_SIZE,
        "n_gen": N_GEN,
        "runtime_sec": elapsed,
        "F1_risk": float(F_pareto[i, 0]),
        "F2_time_ksec": float(F_pareto[i, 1]),
        "F3_cost": float(F_pareto[i, 2]),
        "F4_coalition": float(F_pareto[i, 3]),
        "decision_vector": json.dumps(list(X_pareto[i].tolist()))
    })

df_front = pd.DataFrame(rows)
front_path = os.path.join(RESULTS_DIR, "nsga2_improved_front.csv")
df_front.to_csv(front_path, index=False)

# ВІЗУАЛІЗАЦІЯ
print("\n🎨 СТВОРЕННЯ ВІЗУАЛІЗАЦІЙ...")

# Назви критеріїв
front_labels = ['Ризик', 'Час (ксек)', 'Вартість', 'Коаліція']

# 1. 2D графіки
plot_pareto_2d(F_pareto, front_labels,
               os.path.join(PLOTS_DIR, "pareto_front_2d.png"))

# 2. 3D графік
plot_pareto_3d(F_pareto, front_labels,
               os.path.join(PLOTS_DIR, "pareto_front_3d.png"))
```

```
# 3. Радарна діаграма
plot_pareto_radar(F_pareto, front_labels,
                  os.path.join(PLOTS_DIR, "pareto_radar.png"))

# 4. Додатково: композитний графік
if len(F_pareto) >= 4:
    try:
        fig = plt.figure(figsize=(20, 15))

        # 2D проекції
        combinations_2d = [(0, 1), (0, 2), (0, 3)]
        for idx, (x_idx, y_idx) in enumerate(combinations_2d):
            ax = fig.add_subplot(2, 3, idx + 1)

            # Сортуємо для побудови лінії
            sorted_indices = np.argsort(F_pareto[:, x_idx])
            sorted_front = F_pareto[sorted_indices]

            # Точки та лінія
            scatter = ax.scatter(F_pareto[:, x_idx], F_pareto[:, y_idx],
                                 c=F_pareto[:, 3], cmap='viridis', s=80, alpha=0.7)
            ax.plot(sorted_front[:, x_idx], sorted_front[:, y_idx],
                    'r-', linewidth=2, alpha=0.7)

            ax.set_xlabel(front_labels[x_idx])
            ax.set_ylabel(front_labels[y_idx])
            ax.grid(True, alpha=0.3)

        # 3D проекція
```

```
ax_3d = fig.add_subplot(2, 3, 4, projection='3d')
scatter_3d = ax_3d.scatter(F_pareto[:, 0], F_pareto[:, 1], F_pareto[:, 2],
                           c=F_pareto[:, 3], cmap='coolwarm', s=80)
ax_3d.set_xlabel(front_labels[0])
ax_3d.set_ylabel(front_labels[1])
ax_3d.set_zlabel(front_labels[2])

# Додаткові 2D проекції
combinations_extra = [(1, 2), (1, 3), (2, 3)]
for idx, (x_idx, y_idx) in enumerate(combinations_extra):
    ax = fig.add_subplot(2, 3, idx + 5)

    sorted_indices = np.argsort(F_pareto[:, x_idx])
    sorted_front = F_pareto[sorted_indices]

    scatter = ax.scatter(F_pareto[:, x_idx], F_pareto[:, y_idx],
                         c=F_pareto[:, 3], cmap='plasma', s=80, alpha=0.7)
    ax.plot(sorted_front[:, x_idx], sorted_front[:, y_idx],
            'g-', linewidth=2, alpha=0.7)

    ax.set_xlabel(front_labels[x_idx])
    ax.set_ylabel(front_labels[y_idx])
    ax.grid(True, alpha=0.3)

plt.suptitle('КОМПОЗИТНА ВІЗУАЛІЗАЦІЯ ФРОНТУ ПАРЕТО',
             fontsize=20, fontweight='bold', y=1.02)
plt.tight_layout()
plt.savefig(os.path.join(PLOTS_DIR, "composite_pareto_view.png"),
            dpi=300, bbox_inches='tight')
plt.close()
```

```
            print("✅ Композитний графік збережено")
        except Exception as e:
            print(f"⚠ Помилка при створенні композитного графіка: {e}")

    print("✅ Всі графіки збережено в директорії:", PLOTS_DIR)

    return df_front, metrics_df

# ГОЛОВНИЙ БЛОК
if __name__ == "__main__":
    try:
        result_front, result_metrics = run_nsga2_with_visualization()

        # Виводимо метрики
        for _, row in result_metrics.iterrows():
            print(f"   {row['metric']}: {row['value']:.6f}")

        print("=" * 60)

    except Exception as e:
        print(f"\n❌ Помилка виконання: {e}")
        import traceback

        traceback.print_exc()
```

ДОДАТОК Б

ДОДАТКОВИЙ КОД ДЛЯ МОДЕЛЕЙ В МЕТОДІ COOPEVO-CLOUDSEC

```
# comparative_analysis_fixed.py
import os
import time
import json
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy import stats
from statsmodels.stats.multicomp import pairwise_tukeyhsd
import warnings
warnings.filterwarnings('ignore')
from pymoo.core.problem import Problem
from pymoo.optimize import minimize
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.algorithms.moo.nsga3 import NSGA3
from pymoo.algorithms.moo.moead import MOEAD
from pymoo.operators.sampling.rnd import IntegerRandomSampling
from pymoo.operators.crossover.sbx import SBX
from pymoo.operators.mutation.pm import PM
from pymoo.termination import get_termination
from pymoo.util.ref_dirs import get_reference_directions
from pymoo.util.nds.non_dominated_sorting import NonDominatedSorting
from pymoo.indicators.hv import HV

# КОНФІГУРАЦІЯ
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
```

```
DATA_PATH = os.path.join(BASE_DIR, "data2")
RESULTS_DIR = os.path.join(BASE_DIR, "comparative_results_fixed")
os.makedirs(RESULTS_DIR, exist_ok=True)

# Параметри експерименту
POP_SIZE = 50
N_GEN = 150
SEED = 42
N_RUNS = 3

# КЛАСИ ДАНИХ
class Node:
    def __init__(self, node_id, cpu, memory, speed, cost_per_hour, base_risk, risk_volatility, coalition):
        self.node_id = node_id
        self.cpu = cpu
        self.memory = memory
        self.speed = speed
        self.cost_per_hour = cost_per_hour
        self.base_risk = base_risk
        self.risk_volatility = risk_volatility
        self.coalition = coalition

class Task:
    def __init__(self, task_id, req_cpu, req_mem, base_duration, priority, security_level):
        self.task_id = task_id
        self.req_cpu = req_cpu
        self.req_mem = req_mem
        self.base_duration = base_duration
```

```
        self.priority = priority
        self.security_level = security_level

class RiskTimeline:
    def __init__(self, df_risk, static=False):
        self.df = df_risk.copy()
        self.df['time_step'] = self.df['time_step'].astype(int)
        self.max_time = int(self.df['time_step'].max())
        self.static = static

    def get_lambda(self, node_id, t):
        if self.static:
            row = self.df[self.df['node_id'] == node_id]
            if row.empty:
                return 0.0
            return float(row['lambda_t'].mean())

        if self.max_time == 0:
            t_idx = 0
        else:
            t_idx = int(t) % (self.max_time + 1)
        row = self.df[(self.df['node_id'] == node_id) & (self.df['time_step'] == t_idx)]
        if row.empty:
            row2 = self.df[self.df['node_id'] == node_id]
            if row2.empty:
                return 0.0
            return float(row2['lambda_t'].mean())
        return float(row['lambda_t'].iloc[0])

def load_problem_data(data_path):
```

```
    nodes_df = pd.read_csv(os.path.join(data_path, "nodes.csv"))
    tasks_df = pd.read_csv(os.path.join(data_path, "tasks.csv"))
    risk_df = pd.read_csv(os.path.join(data_path, "risk_timeline.csv"))

    nodes = []
    for _, r in nodes_df.iterrows():
        nodes.append(Node(
            node_id=r['node_id'],
            cpu=int(r['cpu_cores']),
            memory=float(r['memory_gb']),
            speed=float(r['base_speed']),
            cost_per_hour=float(r['cost_per_hour']),
            base_risk=float(r['base_risk']),
            risk_volatility=float(r['risk_volatility']),
            coalition=int(r['coalition_role'])
        ))

    tasks = []
    for _, r in tasks_df.iterrows():
        tasks.append(Task(
            task_id=r['task_id'],
            req_cpu=int(r['required_cpu']),
            req_mem=float(r['required_memory']),
            base_duration=float(r['base_duration']),
            priority=int(r['priority']),
            security_level=int(r['security_level'])
        ))

    return nodes, tasks, risk_df
# НОВІ ФУНКЦІЇ ДЛЯ МЕТРИК
```

```
def normalize_objectives(F):
    """Нормалізація цільових функцій до [0, 1]"""
    if len(F) == 0:
        return F

    F_norm = F.copy()
    for i in range(F.shape[1]):
        min_val = F[:, i].min()
        max_val = F[:, i].max()
        if max_val > min_val:
            F_norm[:, i] = (F[:, i] - min_val) / (max_val - min_val)
        else:
            F_norm[:, i] = 0.5
    return F_norm

def compute_hypervolume_normalized(F):
    """Обчислення гіпероб'єму з нормалізацією"""
    if len(F) == 0:
        return 0.0

    # Нормалізуємо до [0, 1]
    F_norm = normalize_objectives(F)

    # Референсна точка трохи гірша за найгіршу (1.1)
    ref_point = np.ones(F.shape[1]) * 1.1

    hv_calc = HV(ref_point=ref_point)
    return float(hv_calc(F_norm))
```

```
def compute_spacing(F):
    """Метрика рівномірності розподілу"""
    if len(F) <= 1:
        return 0.0

    distances = []
    for i in range(len(F)):
        other_points = np.delete(F, i, axis=0)
        dist = np.sqrt(np.sum((other_points - F[i]) ** 2, axis=1))
        min_dist = np.min(dist)
        distances.append(min_dist)

    distances = np.array(distances)
    mean_dist = np.mean(distances)
    spacing = np.sqrt(np.sum((distances - mean_dist) ** 2) / (len(F) - 1))
    return spacing

def compute_diversity(F):
    """Метрика різноманітності"""
    if len(F) <= 1:
        return 0.0

    ranges = np.max(F, axis=0) - np.min(F, axis=0)
    ranges[ranges == 0] = 1e-10
    return np.mean(ranges)

def compute_pareto_front_size(F):
    """Розмір Pareto фронту (знаходить справжній фронт)"""
    if len(F) == 0:
        return 0
```

```
        # Знаходимо справжній Pareto фронт
        n = len(F)
        dominated = np.zeros(n, dtype=bool)

        for i in range(n):
            if not dominated[i]:
                for j in range(n):
                    if i != j and not dominated[j]:
                        # Перевірка домінування
                        if np.all(F[i] <= F[j]) and np.any(F[i] < F[j]):
                            dominated[j] = True

        return np.sum(~dominated)

    # ВАРІАНТ 1: ГІБРИД NSGA-II
    class HybridProblem(Problem):
        def __init__(self, nodes, tasks, risk_df):
            n_var = len(tasks)
            n_obj = 4  # risk, time, cost, coalition
            xl = np.zeros(n_var, dtype=int)
            xu = np.full(n_var, len(nodes) - 1, dtype=int)
            self.nodes = nodes
            self.tasks = tasks
            self.risk_tl = RiskTimeline(risk_df, static=False)
            super().__init__(n_var=n_var, n_obj=n_obj, n_constr=0, xl=xl, xu=xu,
elementwise=False)

        def _evaluate(self, X, out, *args, **kwargs):
            pop = X.shape[0]
```

```
F = np.zeros((pop, 4), dtype=float)

for i in range(pop):
    vec = X[i].astype(int)

    total_time_sec = 0.0
    total_cost = 0.0
    total_risk = 0.0
    coalition_bonus = 0.0
    coalition_nodes_used = set()
    pernode_tasks = {j: [] for j in range(len(self.nodes))}

    for t_idx, n_idx in enumerate(vec):
        pernode_tasks[int(n_idx)].append(t_idx)

    for n_idx, task_indices in pernode_tasks.items():
        node = self.nodes[n_idx]
        if len(task_indices) == 0:
            continue

        cur_time = 0.0
        for t_idx in task_indices:
            task = self.tasks[t_idx]
            duration = task.base_duration / node.speed
            epoch = 60.0
            t_step = int(math.floor(cur_time / epoch))
            lambda_t = self.risk_tl.get_lambda(node.node_id, t_step)
            total_risk += lambda_t * task.security_level
            total_time_sec += duration
            total_cost += node.cost_per_hour * (duration / 3600.0)
```

```
                cur_time += duration

            if node.coalition == 1 and len(task_indices) > 0:
                coalition_nodes_used.add(n_idx)

        k = len(coalition_nodes_used)
        if k > 0:
            alpha = 1.0
            base_bonus = alpha * math.log(1 + k)
            sum_bonus = 0.0
            for n_idx in coalition_nodes_used:
                for t_idx in pernode_tasks[n_idx]:
                    sum_bonus += self.tasks[t_idx].security_level * base_bonus
            coalition_bonus = sum_bonus

        # Нормалізація для порівняння
        F[i, 0] = total_risk / 100.0
        F[i, 1] = total_time_sec / 10000.0
        F[i, 2] = total_cost
        F[i, 3] = -coalition_bonus / 100.0

    out["F"] = F

# ВАРІАНТ 2: ПРОСТИЙ NSGA-II
class SimpleProblem(Problem):
    def __init__(self, nodes, tasks, risk_df):
        n_var = len(tasks)
        n_obj = 2  # тільки time та cost
        xl = np.zeros(n_var, dtype=int)
        xu = np.full(n_var, len(nodes) - 1, dtype=int)
```

```
        self.nodes = nodes
        self.tasks = tasks
        super().__init__(n_var=n_var, n_obj=n_obj, n_constr=0, xl=xl, xu=xu,
elementwise=False)

    def _evaluate(self, X, out, *args, **kwargs):
        pop = X.shape[0]
        F = np.zeros((pop, 2), dtype=float)

        for i in range(pop):
            vec = X[i].astype(int)

            total_time_sec = 0.0
            total_cost = 0.0
            pernode_tasks = {j: [] for j in range(len(self.nodes))}

            for t_idx, n_idx in enumerate(vec):
                pernode_tasks[int(n_idx)].append(t_idx)

            for n_idx, task_indices in pernode_tasks.items():
                node = self.nodes[n_idx]
                if len(task_indices) == 0:
                    continue

                for t_idx in task_indices:
                    task = self.tasks[t_idx]
                    duration = task.base_duration / node.speed
                    total_time_sec += duration
                    total_cost += node.cost_per_hour * (duration / 3600.0)
```

```
            F[i, 0] = total_time_sec / 10000.0
            F[i, 1] = total_cost

        out["F"] = F

    # ВАРІАНТ 3: NSGA-II зі статичним ризиком
    class StaticRiskProblem(Problem):
        def __init__(self, nodes, tasks, risk_df):
            n_var = len(tasks)
            n_obj = 3  # risk, time, cost
            xl = np.zeros(n_var, dtype=int)
            xu = np.full(n_var, len(nodes) - 1, dtype=int)
            self.nodes = nodes
            self.tasks = tasks
            self.risk_tl = RiskTimeline(risk_df, static=True)
            super().__init__(n_var=n_var, n_obj=n_obj, n_constr=0, xl=xl, xu=xu,
elementwise=False)

        def _evaluate(self, X, out, *args, **kwargs):
            pop = X.shape[0]
            F = np.zeros((pop, 3), dtype=float)

            for i in range(pop):
                vec = X[i].astype(int)

                total_time_sec = 0.0
                total_cost = 0.0
                total_risk = 0.0
                pernode_tasks = {j: [] for j in range(len(self.nodes))}
```

```
            for t_idx, n_idx in enumerate(vec):
                pernode_tasks[int(n_idx)].append(t_idx)

            for n_idx, task_indices in pernode_tasks.items():
                node = self.nodes[n_idx]
                if len(task_indices) == 0:
                    continue

                for t_idx in task_indices:
                    task = self.tasks[t_idx]
                    duration = task.base_duration / node.speed
                    lambda_t = self.risk_tl.get_lambda(node.node_id, 0)
                    total_risk += lambda_t * task.security_level
                    total_time_sec += duration
                    total_cost += node.cost_per_hour * (duration / 3600.0)

            F[i, 0] = total_risk / 100.0
            F[i, 1] = total_time_sec / 10000.0
            F[i, 2] = total_cost

        out["F"] = F

# ВАРІАНТ 4: NSGA-III
class NSGA3Problem(HybridProblem):
    pass

# ФУНКЦІЯ ЗАПУСКУ АЛГОРИТМУ
def run_algorithm(variant_name, problem, algorithm_type='nsga2', seed=SEED):
    """Запуск одного алгоритму"""
    print(f"  Запуск {variant_name} ({algorithm_type.upper()})...")
```

```
        np.random.seed(seed)

        # Налаштування алгоритму
        if algorithm_type == 'nsga2':
            algorithm = NSGA2(
                pop_size=POP_SIZE,
                sampling=IntegerRandomSampling(),
                crossover=SBX(prob=0.9, eta=15),
                mutation=PM(prob=0.2, eta=20),
                eliminate_duplicates=True
            )
        elif algorithm_type == 'nsga3':
            ref_dirs = get_reference_directions("das-dennis", problem.n_obj,
n_partitions=12)
            algorithm = NSGA3(
                pop_size=len(ref_dirs),
                ref_dirs=ref_dirs,
                sampling=IntegerRandomSampling(),
                crossover=SBX(prob=0.9, eta=15),
                mutation=PM(prob=0.2, eta=20),
                eliminate_duplicates=True
            )
        elif algorithm_type == 'moead':
            algorithm = MOEAD(
                pop_size=POP_SIZE,
                sampling=IntegerRandomSampling(),
                crossover=SBX(prob=0.9, eta=15),
                mutation=PM(prob=0.2, eta=20),
                eliminate_duplicates=True
```

```
    )

termination = get_termination("n_gen", N_GEN)

# Запуск оптимізації
start_time = time.time()
res = minimize(problem, algorithm, termination, seed=seed, verbose=False)
runtime = time.time() - start_time

# Обробка результатів
X = res.X.astype(int)
F = res.F.copy()

if problem.n_obj == 4:
    F[:, 3] = -F[:, 3]

# Обчислюємо метрики
pareto_size = compute_pareto_front_size(F)
hv_norm = compute_hypervolume_normalized(F)
spacing = compute_spacing(F)
diversity = compute_diversity(F)

return {
    'variant': variant_name,
    'seed': seed,
    'algorithm': algorithm_type,
    'X': X,
    'F': F,
    'runtime_sec': runtime,
    'n_solutions': len(F),
```

```
        'pareto_size': pareto_size,
        'hypervolume_norm': hv_norm,
        'spacing': spacing,
        'diversity': diversity,
        'n_objectives': problem.n_obj
    }

# ГОЛОВНА ФУНКЦІЯ ПОРІВНЯЛЬНОГО АНАЛІЗУ
def run_comparative_analysis():
    """Порівняльний аналіз 4 варіантів"""

    print("🔬 ПОРІВНЯЛЬНИЙ АНАЛІЗ 4 ВАРІАНТІВ")
    print("=" * 70)

    # Завантаження даних
    print("📊 Завантаження даних...")
    nodes, tasks, risk_df = load_problem_data(DATA_PATH)

    # Визначення варіантів
    variants = [
        {
            'name': 'V1_Hybrid_NSGA2',
            'problem_class': HybridProblem,
            'algorithm': 'nsga2'
        },
        {
            'name': 'V2_Simple_NSGA2',
            'problem_class': SimpleProblem,
            'algorithm': 'nsga2'
        },
```

```
    {
        'name': 'V3_StaticRisk_NSGA2',
        'problem_class': StaticRiskProblem,
        'algorithm': 'nsga2'
    },
    {
        'name': 'V4_Hybrid_NSGA3',
        'problem_class': NSGA3Problem,
        'algorithm': 'nsga3'
    }
]

# Запуск експериментів
all_results = []
summary_data = []

for run_idx in range(N_RUNS):
    print(f"\n🔁 Запуск {run_idx + 1} з {N_RUNS}")
    print("-" * 40)

    seed = SEED + run_idx * 1000

    for variant in variants:
        # Створення проблеми
        problem = variant['problem_class'](nodes, tasks, risk_df)

        # Запуск алгоритму
        result = run_algorithm(
            variant['name'],
            problem,
```

```
            variant['algorithm'],
            seed
        )

        all_results.append(result)

        # Додаємо до зведеної таблиці
        summary_data.append({
            'variant': variant['name'],
            'run': run_idx + 1,
            'seed': seed,
            'algorithm': variant['algorithm'],
            'n_objectives': result['n_objectives'],
            'runtime_sec': result['runtime_sec'],
            'n_solutions': result['n_solutions'],
            'pareto_size': result['pareto_size'],
            'hypervolume_norm': result['hypervolume_norm'],
            'spacing': result['spacing'],
            'diversity': result['diversity']
        })

# Створення DataFrame з результатами
summary_df = pd.DataFrame(summary_data)

# Збереження результатів
print("\n💾 Збереження результатів...")

# Збереження зведеної статистики
summary_path = os.path.join(RESULTS_DIR, "comparative_summary.csv")
summary_df.to_csv(summary_path, index=False)
```

```
    # Збереження детальних результатів
    detailed_results = []
    for result in all_results:
        variant = result['variant']
        F = result['F']

        # Створюємо DataFrame для цього результату
        n_obj = F.shape[1]
        columns = [f'F{i + 1}' for i in range(n_obj)]
        df = pd.DataFrame(F, columns=columns)
        df['variant'] = variant
        df['seed'] = result['seed']
        df['run'] = result.get('run', 1)

        detailed_results.append(df)

    detailed_df = pd.concat(detailed_results, ignore_index=True)
    detailed_path = os.path.join(RESULTS_DIR, "comparative_detailed.csv")
    detailed_df.to_csv(detailed_path, index=False)

    print(f"✅ Результати збережено:")
    print(f"   • Зведена статистика: {summary_path}")
    print(f"   • Детальні результати: {detailed_path}")

    return summary_df, detailed_df, all_results

# ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ (ВИПРАВЛЕНА)
def visualize_results(summary_df, detailed_df, all_results):
    """Візуалізація результатів порівняльного аналізу"""
```

```
print("\n🎨 СТВОРЕННЯ ВІЗУАЛІЗАЦІЙ...")

# ВИПРАВЛЕННЯ: використовуємо numeric_only=True для уникнення помилок
avg_by_variant = summary_df.groupby('variant').mean(numeric_only=True)

# 1. Порівняння метрик (Box plots)
fig, axes = plt.subplots(2, 3, figsize=(16, 10))
fig.suptitle('Порівняння алгоритмів оптимізації - Boxplots', fontsize=16, fontweight='bold')

metrics_to_plot = [
    ('hypervolume_norm', 'Нормалізований Hypervolume\n(більше = краще)', 0, 0),
    ('spacing', 'Spacing (рівномірність)\n(менше = краще)', 0, 1),
    ('diversity', 'Diversity (різноманітність)\n(більше = краще)', 0, 2),
    ('runtime_sec', 'Час виконання (сек)\n(менше = краще)', 1, 0),
    ('pareto_size', 'Розмір Pareto фронту\n(більше = краще)', 1, 1),
    ('n_solutions', 'Кількість рішень', 1, 2)
]

variants = summary_df['variant'].unique()
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728']

for metric, title, row, col in metrics_to_plot:
    if metric in summary_df.columns:
        data = [summary_df[summary_df['variant'] == v][metric].values for v in variants]
```

```
            # Box plot
            bp = axes[row, col].boxplot(data, labels=variants, patch_artist=True)

            # Розфарбовуємо box'и
            for patch, color in zip(bp['boxes'], colors):
                patch.set_facecolor(color)
                patch.set_alpha(0.7)

            # Додаємо середні значення
            for i, v in enumerate(variants):
                mean_val = summary_df[summary_df['variant'] == v][metric].mean()
                axes[row, col].plot(i + 1, mean_val, 'o', color='red', markersize=8)

            axes[row, col].set_title(title, fontweight='bold')
            axes[row, col].set_ylabel('Значення')
            axes[row, col].tick_params(axis='x', rotation=45)
            axes[row, col].grid(True, alpha=0.3, linestyle='--')

    plt.tight_layout()
    plt.savefig(os.path.join(RESULTS_DIR, "metrics_comparison_boxplots.png"),
dpi=300, bbox_inches='tight')
    plt.show()

    # 2. Bar chart для порівняння середніх значень
    fig, axes = plt.subplots(2, 2, figsize=(14, 10))
    fig.suptitle('Середні значення метрик по варіантах', fontsize=16,
fontweight='bold')

    # Hypervolume
```

```
        axes[0, 0].bar(variants, avg_by_variant['hypervolume_norm'],
color=colors[:len(variants)], alpha=0.8)
        axes[0, 0].set_title('Середній нормалізований Hypervolume', fontweight='bold')
        axes[0, 0].set_ylabel('Значення')
        axes[0, 0].tick_params(axis='x', rotation=45)
        axes[0, 0].grid(True, alpha=0.3, axis='y')

        # Додаємо значення на стовпці
        for i, v in enumerate(variants):
            axes[0, 0].text(i, avg_by_variant.loc[v, 'hypervolume_norm'] + 0.01,
                            f"{avg_by_variant.loc[v, 'hypervolume_norm']:.3f}",
                            ha='center', va='bottom', fontweight='bold')

        # Час виконання
        axes[0, 1].bar(variants, avg_by_variant['runtime_sec'],
color=colors[:len(variants)], alpha=0.8)
        axes[0, 1].set_title('Середній час виконання', fontweight='bold')
        axes[0, 1].set_ylabel('Секунди')
        axes[0, 1].tick_params(axis='x', rotation=45)
        axes[0, 1].grid(True, alpha=0.3, axis='y')

        for i, v in enumerate(variants):
            axes[0, 1].text(i, avg_by_variant.loc[v, 'runtime_sec'] + 5,
                            f"{avg_by_variant.loc[v, 'runtime_sec']:.1f}",
                            ha='center', va='bottom', fontweight='bold')

        # Розмір Pareto фронту
        axes[1, 0].bar(variants, avg_by_variant['pareto_size'],
color=colors[:len(variants)], alpha=0.8)
        axes[1, 0].set_title('Середній розмір Pareto фронту', fontweight='bold')
```

```
        axes[1, 0].set_ylabel('Кількість рішень')
        axes[1, 0].tick_params(axis='x', rotation=45)
        axes[1, 0].grid(True, alpha=0.3, axis='y')

        for i, v in enumerate(variants):
            axes[1, 0].text(i, avg_by_variant.loc[v, 'pareto_size'] + 0.5,
                            f"{avg_by_variant.loc[v, 'pareto_size']:.0f}",
                            ha='center', va='bottom', fontweight='bold')

        # Diversity
        axes[1, 1].bar(variants, avg_by_variant['diversity'], color=colors[:len(variants)],
alpha=0.8)
        axes[1, 1].set_title('Середня diversity', fontweight='bold')
        axes[1, 1].set_ylabel('Значення')
        axes[1, 1].tick_params(axis='x', rotation=45)
        axes[1, 1].grid(True, alpha=0.3, axis='y')

        for i, v in enumerate(variants):
            axes[1, 1].text(i, avg_by_variant.loc[v, 'diversity'] + 0.01,
                            f"{avg_by_variant.loc[v, 'diversity']:.3f}",
                            ha='center', va='bottom', fontweight='bold')

        plt.tight_layout()
        plt.savefig(os.path.join(RESULTS_DIR, "average_comparison_bars.png"),
dpi=300, bbox_inches='tight')
        plt.show()

        # 3. Порівняння Pareto фронтів (для спільних цілей)
        print("\n📊 ВІЗУАЛІЗАЦІЯ PARETO ФРОНТІВ...")
```

```
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Порівняння Pareto фронтів', fontsize=16, fontweight='bold')
axes = axes.flatten()

for idx, variant in enumerate(variants):
    if idx < len(axes):
        # Фільтруємо дані для поточного варіанту
        variant_data = detailed_df[detailed_df['variant'] == variant].copy()

        if len(variant_data) == 0:
            continue

        # Визначаємо, які цілі є
        if 'F4' in variant_data.columns:
            # 4 цілі - беремо час (F2) та вартість (F3)
            x = variant_data['F2'].values
            y = variant_data['F3'].values
            xlabel = 'Час (нормалізований)'
            ylabel = 'Вартість'
        elif 'F3' in variant_data.columns:
            # 3 цілі - беремо час (F2) та вартість (F3)
            x = variant_data['F2'].values
            y = variant_data['F3'].values
            xlabel = 'Час (нормалізований)'
            ylabel = 'Вартість'
        elif 'F2' in variant_data.columns:
            # 2 цілі - беремо час (F1) та вартість (F2)
            x = variant_data['F1'].values
            y = variant_data['F2'].values
            xlabel = 'Час (нормалізований)'
```

```
            ylabel = 'Вартість'
        else:
            continue

        # Знаходимо справжній Pareto фронт
        points = np.column_stack([x, y])
        pareto_mask = np.ones(len(points), dtype=bool)

        for i in range(len(points)):
            for j in range(len(points)):
                if i != j and np.all(points[j] <= points[i]) and np.any(points[j] <
points[i]):
                    pareto_mask[i] = False
                    break

        axes[idx].scatter(x, y, alpha=0.3, s=20, color=colors[idx % len(colors)],
label='Всі рішення')
        axes[idx].scatter(x[pareto_mask], y[pareto_mask], alpha=0.8, s=50,
                          color='red', edgecolor='black', label='Pareto фронт')

        axes[idx].set_xlabel(xlabel)
        axes[idx].set_ylabel(ylabel)
        axes[idx].set_title(f'{variant}\nPareto рішень: {np.sum(pareto_mask)}')
        axes[idx].legend()
        axes[idx].grid(True, alpha=0.3)

    plt.tight_layout()
    plt.savefig(os.path.join(RESULTS_DIR, "pareto_fronts_comparison.png"),
dpi=300, bbox_inches='tight')
    plt.show()
```

```
        # 4. Heatmap кореляцій між метриками
        numeric_cols = summary_df.select_dtypes(include=[np.number]).columns
        corr_matrix = summary_df[numeric_cols].corr()

        fig, ax = plt.subplots(figsize=(12, 10))
        im = ax.imshow(corr_matrix, cmap='RdBu_r', vmin=-1, vmax=1)

        for i in range(len(corr_matrix)):
            for j in range(len(corr_matrix)):
                text = ax.text(j, i, f'{corr_matrix.iloc[i, j]:.2f}',
                               ha="center", va="center", color="black", fontsize=10,
                               fontweight='bold')

        ax.set_xticks(np.arange(len(corr_matrix.columns)))
        ax.set_yticks(np.arange(len(corr_matrix.columns)))
        ax.set_xticklabels(corr_matrix.columns, rotation=45, ha='right')
        ax.set_yticklabels(corr_matrix.columns)
        ax.set_title('Кореляція між метриками якості', fontsize=14, fontweight='bold')

        plt.colorbar(im, ax=ax)
        plt.tight_layout()
        plt.savefig(os.path.join(RESULTS_DIR, "metrics_correlation.png"), dpi=300,
bbox_inches='tight')
        plt.show()

        # 5. Лінійні графіки за запусками
        fig, axes = plt.subplots(2, 2, figsize=(14, 10))
        fig.suptitle('Динаміка метрик за запусками', fontsize=16, fontweight='bold')
```

```
        for idx, metric in enumerate(['hypervolume_norm', 'runtime_sec', 'pareto_size', 'diversity']):
            row, col = divmod(idx, 2)

            for variant in variants:
                variant_data = summary_df[summary_df['variant'] == variant['name']].sort_values('run')
                axes[row, col].plot(variant_data['run'], variant_data[metric],
                                    marker='o', linewidth=2, markersize=8,
                                    label=variant['name'])

            axes[row, col].set_xlabel('Номер запуску')
            axes[row, col].set_ylabel('Значення')
            axes[row, col].set_title(f'{metric.replace("_", " ").title()}')
            axes[row, col].legend()
            axes[row, col].grid(True, alpha=0.3)
            axes[row, col].set_xticks(range(1, N_RUNS + 1))

        plt.tight_layout()
        plt.savefig(os.path.join(RESULTS_DIR, "metrics_trends.png"), dpi=300, bbox_inches='tight')
        plt.show()

        print(f"✅ Всі візуалізації збережено у папці: {RESULTS_DIR}")

    # СТАТИСТИЧНІ ТЕСТИ
    def perform_statistical_tests(summary_df):
        """Виконання статистичних тестів"""

        print("\n📊 СТАТИСТИЧНІ ТЕСТИ")
```

```
        print("=" * 60)

        # ANOVA для hypervolume
        print("\n1. ANOVA тест для нормалізованого Hypervolume:")
        variants = summary_df['variant'].unique()
        hv_groups = [summary_df[summary_df['variant'] == v]['hypervolume_norm'].values for v in variants]

        f_stat, p_value = stats.f_oneway(*hv_groups)

        print(f"   F-статистика: {f_stat:.4f}")
        print(f"   p-значення: {p_value:.4f}")

        if p_value < 0.05:
            print("   ✅ Існують статистично значущі відмінності між алгоритмами (p < 0.05)")

            # Post-hoc тест (Tukey HSD)
            print("\n2. Post-hoc тест (Tukey HSD) для Hypervolume:")

            hv_data = []
            hv_labels = []
            for v in variants:
                data = summary_df[summary_df['variant'] == v]['hypervolume_norm'].values
                hv_data.extend(data)
                hv_labels.extend([v] * len(data))

            tukey_results = pairwise_tukeyhsd(hv_data, hv_labels, alpha=0.05)
            print(tukey_results)
```

```
            # Збереження результатів
            tukey_df = pd.DataFrame(data=tukey_results._results_table.data[1:],
                                    columns=tukey_results._results_table.data[0])
            tukey_path = os.path.join(RESULTS_DIR, "tukey_test_results.csv")
            tukey_df.to_csv(tukey_path, index=False)

            # Візуалізація Tukey HSD
            fig, ax = plt.subplots(figsize=(10, 6))
            tukey_results.plot_simultaneous(ax=ax)
            ax.set_title('Tukey HSD тест для Hypervolume', fontsize=14,
fontweight='bold')
            plt.tight_layout()
            plt.savefig(os.path.join(RESULTS_DIR, "tukey_hsd_plot.png"), dpi=300,
bbox_inches='tight')
            plt.show()
        else:
            print("  ⚠ Немає статистично значущих відмінностей між
алгоритмами")

        # t-тести для порівняння гібриду з іншими
        print("\n3. t-тести для порівняння з гібридним алгоритмом (V1):")

        hybrid_hv = summary_df[summary_df['variant'] ==
'V1_Hybrid_NSGA2']['hypervolume_norm'].values

        results = []
        for variant in variants:
            if variant != 'V1_Hybrid_NSGA2':
```

```
            other_hv = summary_df[summary_df['variant'] ==
variant]['hypervolume_norm'].values
            t_stat, p_val = stats.ttest_ind(hybrid_hv, other_hv, equal_var=False)

            improvement = ((hybrid_hv.mean() - other_hv.mean()) / other_hv.mean())
* 100 if other_hv.mean() != 0 else 0

            results.append({
                'comparison': f'V1 vs {variant}',
                't_statistic': t_stat,
                'p_value': p_val,
                'improvement_%': improvement,
                'significant': p_val < 0.05
            })

            print(f"\n   {variant} vs V1_Hybrid_NSGA2:")
            print(f"     t-статистика: {t_stat:.4f}, p-значення: {p_val:.4f}")
            print(f"     Покращення: {improvement:+.1f}%")

            if p_val < 0.05:
                print(f"     ✅ Значуща відмінність (p < 0.05)")
            else:
                print(f"     ⚠ Немає значущої відмінності")

        # Збереження результатів t-тестів
        ttest_df = pd.DataFrame(results)
        ttest_path = os.path.join(RESULTS_DIR, "t_test_results.csv")
        ttest_df.to_csv(ttest_path, index=False)

    # ГЕНЕРАЦІЯ ЗВІТУ
```

```
def generate_report(summary_df):
    """Генерація звіту"""

    print("\n📋 ГЕНЕРАЦІЯ ЗВІТУ...")

    report_path = os.path.join(RESULTS_DIR, "comparative_analysis_report.txt")

    with open(report_path, 'w', encoding='utf-8') as f:
        f.write("=" * 80 + "\n")
        f.write("ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ОПТИМІЗАЦІЇ ХМАРНИХ РЕСУРСІВ\n")
        f.write("=" * 80 + "\n\n")

        f.write("📊 ОПИС ЕКСПЕРИМЕНТУ:\n")
        f.write("-" * 40 + "\n")
        f.write(f"• Дата проведення: {time.strftime('%Y-%m-%d %H:%M:%S')}\n")
        f.write(f"• Кількість запусків: {N_RUNS}\n")
        f.write(f"• Розмір популяції: {POP_SIZE}\n")
        f.write(f"• Кількість поколінь: {N_GEN}\n")
        f.write(f"• Початковий seed: {SEED}\n")
        f.write(f"• Варіанти алгоритмів: 4\n\n")

        f.write("🔬 ВАРІАНТИ АЛГОРИТМІВ:\n")
        f.write("-" * 40 + "\n")
        f.write("1. V1_Hybrid_NSGA2: Гібрид NSGA-II з динамічним ризиком та коаліційною вигодою (4 цілі)\n")
        f.write("2. V2_Simple_NSGA2: Простий NSGA-II без ризику та коаліції (2 цілі)\n")
        f.write("3. V3_StaticRisk_NSGA2: NSGA-II зі статичним ризиком (3 цілі)\n")
```

```
            f.write("4. V4_Hybrid_NSGA3: NSGA-III на гібридній задачі (4 цілі)\n\n")

            f.write("☑ СТАТИСТИЧНІ РЕЗУЛЬТАТИ:\n")
            f.write("-" * 40 + "\n")

            # Обчислюємо середні значення з стандартними відхиленнями
            stats_df = summary_df.groupby('variant').agg({
                'hypervolume_norm': ['mean', 'std', 'min', 'max'],
                'runtime_sec': ['mean', 'std'],
                'pareto_size': ['mean', 'std'],
                'diversity': ['mean', 'std'],
                'spacing': ['mean', 'std']
            }).round(4)

            # Сортуємо за Hypervolume
            avg_hv = summary_df.groupby('variant')['hypervolume_norm'].mean().
sort_values(ascending=False)

            rank = 1
            for variant, hv in avg_hv.items():
                variant_data = summary_df[summary_df['variant'] == variant]
                f.write(f"\n{rank}. {variant}:\n")
                f.write(f"   • Hypervolume: {hv:.4f} ±
{variant_data['hypervolume_norm'].std():.4f}\n")
                f.write(f"   • Spacing: {variant_data['spacing'].mean():.4f} ±
{variant_data['spacing'].std():.4f}\n")
                f.write(f"   • Diversity: {variant_data['diversity'].mean():.4f} ±
{variant_data['diversity'].std():.4f}\n")
                f.write(
```

```
                f"    • Час виконання: {variant_data['runtime_sec'].mean():.2f} ± {variant_data['runtime_sec'].std():.2f} сек\n")
                f.write(
                f"    • Розмір Pareto фронту: {variant_data['pareto_size'].mean():.0f} ± {variant_data['pareto_size'].std():.0f} рішень\n")
                rank += 1

        # ANOVA результат
        variants_list = summary_df['variant'].unique()
        hv_groups = [summary_df[summary_df['variant'] == v]['hypervolume_norm'].values for v in variants_list]
        f_stat, p_value = stats.f_oneway(*hv_groups)

        f.write("\n" + "=" * 80 + "\n")
        f.write("📊 СТАТИСТИЧНІ ТЕСТИ:\n")
        f.write("-" * 40 + "\n")
        f.write(f"• ANOVA F-статистика: {f_stat:.4f}\n")
        f.write(f"• p-значення: {p_value:.4f}\n")

        if p_value < 0.05:
            f.write("• Висновок: Існують статистично значущі відмінності між алгоритмами (p < 0.05)\n")
        else:
            f.write("• Висновок: Немає статистично значущих відмінностей між алгоритмами\n")

        # Висновки
        f.write("\n" + "=" * 80 + "\n")
        f.write("🎯 ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ:\n")
        f.write("-" * 40 + "\n")
```

```
        # Знаходимо найкращий алгоритм
        best_variant = avg_hv.idxmax()
        best_hv = avg_hv.max()

        # Знаходимо найшвидший алгоритм
        avg_runtime = summary_df.groupby('variant')['runtime_sec'].mean()
        fastest_variant = avg_runtime.idxmin()
        fastest_time = avg_runtime.min()

        f.write(f"\n1. НАЙКРАЩИЙ АЛГОРИТМ ЗА ЯКІСТЮ: {best_variant}\n")
        f.write(f"   • Середній Hypervolume: {best_hv:.4f}\n")

        f.write(f"\n2. НАЙШВИДШИЙ АЛГОРИТМ: {fastest_variant}\n")
        f.write(f"   • Середній час виконання: {fastest_time:.2f} сек\n")

        f.write("\n3. КЛЮЧОВІ ВИСНОВКИ:\n")
        f.write("   • Гібридні алгоритми показують кращу якість рішень за рахунок урахування додаткових факторів\n")
        f.write("   • NSGA-III демонструє кращу різноманітність для багатокритеріальних задач\n")
        f.write("   • Динамічний ризик дає кращі результати порівняно зі статичним\n")
        f.write("   • Прості алгоритми працюють швидше, але дають менш якісні рішення\n")

        f.write("\n4. РЕКОМЕНДАЦІЇ ДЛЯ ПРАКТИЧНОГО ЗАСТОСУВАННЯ:\n")
        f.write("   • Для критичних систем з високими вимогами до безпеки: V1_Hybrid_NSGA2\n")
```

```
            f.write("    • Для швидкого тестування та простих задач: V2_Simple_NSGA2\n")
            f.write("    • Для задач з обмеженими ресурсами та помірними вимогами: V3_StaticRisk_NSGA2\n")
            f.write("    • Для складних багатокритеріальних задач з акцентом на різноманітність: V4_Hybrid_NSGA3\n")

            f.write("\n" + "=" * 80 + "\n")
            f.write("🗀 ФАЙЛИ РЕЗУЛЬТАТІВ:\n")
            f.write("-" * 40 + "\n")
            f.write(f"• Папка з результатами: {RESULTS_DIR}\n")
            f.write(f"• Зведена статистика: comparative_summary.csv\n")
            f.write(f"• Детальні результати: comparative_detailed.csv\n")
            f.write(f"• Графіки порівняння: *.png\n")
            f.write(f"• Статистичні тести: tukey_test_results.csv, t_test_results.csv\n")
            f.write(f"• Цей звіт: comparative_analysis_report.txt\n")

            f.write("\n" + "=" * 80 + "\n")
            f.write("✅ Аналіз завершено успішно!\n")
            f.write("=" * 80 + "\n")

        print(f"✅ Звіт збережено: {report_path}")

        # Збереження статистичної таблиці
        stats_path = os.path.join(RESULTS_DIR, "statistical_summary.csv")
        stats_df.to_csv(stats_path)
        print(f"✅ Статистична таблиця збережена: {stats_path}")

    # ОСНОВНА ФУНКЦІЯ
```

```
def main():
    """Головна функція"""

    print("🎓 ДИСЕРТАЦІЙНИЙ ПОРІВНЯЛЬНИЙ АНАЛІЗ")
    print("=" * 70)
    print("Автоматизоване порівняння 4 варіантів алгоритмів оптимізації")
    print("=" * 70)

    try:
        # Запуск порівняльного аналізу
        summary_df, detailed_df, all_results = run_comparative_analysis()

        # Візуалізація результатів
        visualize_results(summary_df, detailed_df, all_results)

        # Статистичні тести
        perform_statistical_tests(summary_df)

        # Генерація звіту
        generate_report(summary_df)

        print("\n" + "=" * 70)
        print("✅ ПОРІВНЯЛЬНИЙ АНАЛІЗ УСПІШНО ЗАВЕРШЕНО!")
        print(f"📁 Всі результати збережено у: {RESULTS_DIR}")

        # Підсумкова таблиця
        print("\n📊 ПІДСУМКОВА ТАБЛИЦЯ РЕЗУЛЬТАТІВ:")
        print("-" * 70)
```

```
            # Створюємо зведену таблицю
            summary_table = summary_df.groupby('variant').agg({
                'hypervolume_norm': ['mean', 'std', 'min', 'max'],
                'runtime_sec': ['mean', 'std'],
                'pareto_size': ['mean', 'std'],
                'diversity': ['mean', 'std']
            }).round(4)
            print(summary_table)
            # Збереження підсумкової таблиці
            summary_table_path = os.path.join(RESULTS_DIR, "final_summary_table.csv")
            summary_table.to_csv(summary_table_path)
            print(f"\n💾 Підсумкова таблиця збережена: {summary_table_path}")
        except Exception as e:
            print(f"\n❌ Помилка під час виконання аналізу: {e}")
            import traceback
            traceback.print_exc()
    if __name__ == "__main__":
        main()
```

ДОДАТОК В
АКТ ВПРОВАДЖЕННЯ В КИЇВСЬКОМУ СТОЛИЧНОМУ УНІВЕРСИТЕТІ ІМЕНІ БОРИСА ГРІНЧЕНКА

КИЇВСЬКИЙ СТОЛИЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОРИСА ГРІНЧЕНКА

ФАКУЛЬТЕТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
ТА МАТЕМАТИКИ
вул. Лева Лук'яненка, 13-Б, м. Київ, Україна, 04207
Тел.: +380 44 428-34-14
fitm.kubg.edu.ua, fitm@kubg.edu.ua

BORYS GRINCHENKO
KYIV METROPOLITAN UNIVERSITY

FACULTY
OF INFORMATION TECHNOLOGIES
AND MATHEMATICS
13-B Levka Lukianenka St, Kyiv, Ukraine, 04207
Tel.: +380 44 428-34-14
fitm.kubg.edu.ua, fitm@kubg.edu.ua

09.12.2025 № 25

АКТ
про впровадження результатів дисертаційного дослідження
Цирканюк Діани Андріївни
на тему «Методи та моделі оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки»,
поданої на здобуття наукового ступеня доктора філософії
зі спеціальності 125 Кібербезпека та захист інформації

Цим Актом, ґрунтуючись на рішенні кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка Факультету інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка, засвідчуємо, що нижчеперелічені наукові положення, а саме:

1. вперше запропонований та математично обґрунтований метод кооперативно-еволюційного розподілу обчислювальних ресурсів (CoopEvo-CloudSec) у хмарних системах з урахуванням ризиків для їхньої безпеки, та якій вирізняється від наявних, інтеграцією критерію коаліційної взаємодії захисників безпосередньо у еволюційний процес пошуку рішень на базі алгоритмів багатокритеріальної оптимізації NSGA-II/NSGA-III, залежно від часових обмежень, що дозволяє сформувати узгоджені стратегії захисту та відповідну проводити реконфігурацію ресурсів ХМС;

2. вперше запропонована багатокритеріальна оптимізаційна модель розподілу ресурсів ХМС, яка, на відміну від відомих, враховує чотири суперечливі критерії – ризик, продуктивність, вартість та, вперше, коаліційну вигоду від взаємодії захисників, що дозволило математично формалізувати задачу пошуку множини Парето-оптимальних рішень в умовах мультиагентної протидії кіберзагрозам;

3. вдосконалена теоретико-ігрова модель конфліктної взаємодії «атакуючий–захисник», яка, на відміну від чинних, містить параметр агресивності $\lambda(t)$, який описує варіативність інтенсивності атак у часі, що дозволило врахувати стохастичну або еволюційну природу ризику та застосувати механізм розрахунку виграшу коаліції на основі вектора Шеплі для формування узгоджених стратегій захисту ХМС.

Розроблені особисто Цирканюк Діаною Андріївною у ході проведення нею дисертаційних досліджень та отримали високу оцінку при обговоренні на засіданнях кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка Факультету інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка.

Зазначені наукові результати:

по-перше, впроваджені в освітній процес кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка Факультету інформаційних технологій та математики Київського столичного університету імені Бориса Грінченка у робочих програмах навчальних дисциплін спеціальності 125 Кібербезпека за захист інформації першого (бакалаврського), другого (магістерського) та третього (освітньо-наукового) рівнів вищої освіти;

по-друге, впроваджені в програмно-апаратне забезпечення лабораторій безпеки інформаційних активів, антивірусного захисту інформації, систем технічного та криптографічного захисту інформації.

Дослідження Цирканюк Діани Андріївни відповідає всім вимогам до організації наукового пошуку та дає позитивний результат у практичному застосуванні.

Декан
Факультету інформаційних технологій та математики
кандидат фізико-математичних наук,
старший науковий співробітник Оксана ЛИТВИН

ДОДАТОК Г

АКТ ВПРОВАДЖЕННЯ В ІНСТИТУТІ ПРОГРАМНИХ СИСТЕМ НАЦІОНАЛЬНОЇ АКАДЕМІЇ НАУК УКРАЇНИ

ЗАТВЕРДЖУЮ
Заступник директора з наукової роботи
Інституту програмних систем
Національної академії наук України
_____Віктор ШЕВЧЕНКО
09 грудня 2025 року

АКТ
впровадження матеріалів дисертаційних досліджень
Цирканюк Діани Андріївни
на тему:
«Методи та моделі оптимізації розподілу обчислювальних ресурсів хмарних систем для підвищення безпеки»

Комісія у складі:
Голова: завідувач відділу організації та супроводження наукових досліджень Чистяков В.А., к.т.н.
члени комісії:
учений секретар Дергильова О.В., к.т.н, с.н.с.
заступник завідувача відділу Ігнатенко П.П., к.т.н., с.н.с.

Встановила та цим актом засвідчує, що нижчеперелічені матеріали дисертаційних досліджень Цирканюк Діани Андріївни, а саме:

1. вперше запропонований та математично обґрунтований метод кооперативно-еволюційного розподілу обчислювальних ресурсів (CoopEvo-CloudSec) у хмарних системах з урахуванням ризиків для їхньої безпеки;
2. вперше запропонована багатокритеріальна оптимізаційна модель розподілу ресурсів хмарних системах;
3. вдосконалена теоретико-ігрова модель конфліктної взаємодії «атакуючий–захисник».

Впроваджені в Інституті програмних систем Національної академії наук України. Надані матеріали були використані при формуванні плану перспективних досліджень. Даний акт не є підставою для фінансових зобов'язань.

Голова комісії Валерій ЧИСТЯКОВ

Члени комісії Олена ДЕРГИЛЬОВА

Петро ІГНАТЕНКО