НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис) (ініціали, прізвище)
“ ” грудня 2024 р

Магістерська дисертація

на здобуття ступеня магістра
зі спеціальності *125 Кібербезпека та захист інформації*
на тему: «Оцінювання релевантності методу мурашиних колоній у задачах виявлення несанкціонованого доступу»

Виконав:
студент 2 курсу, групи ФБ-з31мп
Ішенко Артем Андрійович _____

Керівник:
доцент кафедри інформаційної безпеки,
доцент, канд. техн. наук,
Гальчинський Леонід Юрійович _____

Рецензент:
доцент кафедри математичного аналізу
та теорії ймовірностей ФМФ,
доцент, к.ф.-м.н.,
Кубайчук Оксана Олексіївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Форма № Н-9.01

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий фізико-технічний інститут
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність 125 Кібербезпека та захист інформації
Освітня програма Системи, технології та математичні методи кібербезпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)
"___"_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Іщенку Артему Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Оцінювання релевантності методу мурашиних колоній у задачах виявлення несанкціонованого доступу.

науковий керівник дисертації Гальчинський Леонід Юрійович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
канд. техн. наук, доцент, доцент кафедри інформаційної безпеки
затверджені наказом по університету від "07" листопада 2024 року № 5002-с
2. Термін подання студентом дисертації 09.12.2024.
3. Об'єкт дослідження релевантность методу мурашиних колоній в системах виявлення вторгнень
4. Вихідні дані технічна документація, теоретичні та статистичні дані
5. Перелік завдань, які потрібно розробити 1. Проаналізувати предметну область. 2. Проаналізувати проблематику задач виявлення несанкціонованого доступу 3. Вивчити можливість використання алгоритму мурашиних колоній в задачах оптимізації вибору ознак в IDS. 4. Розробити проєкт для оцінки різних алгоритмів. 5. Визначити релевантність алгоритму мурашиних колоній за допомогою багатокритеріальної оцінки у порівнянні з іншими алгоритмами.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу): презентація.
7. Орієнтовний перелік публікацій: Іщенко А. А., Гальчинський, Л. Ю. Оцінювання релевантності методу мурашиних колоній (ACO) для вирішення використання в системах виявлення вторгнення. [Електронний ресурс] / Collection of Scientific Papers

«ΛΟΓΟΣ»,(November 15, 2024; Bologna, Italy), 160–169. Режим доступу: https://archive.logos-science.com/index.php/conference-proceedings/issue/view/29/29.

8. Дата видачі завдання 01.09.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз проблематики задач виявлення несанкціонованого доступу	01.09.2024	
2.	Вивчення можливості використання алгоритму мурашиних колоній в задачах оптимізації вибору ознак в IDS	15.09.2024	
3.	Емпіричне дослідження показників алгоритму мурашиних колоній в задачах оптимізації вибору ознак в IDS	01.10.2024	
4.	Реалізація тестового проєкту	18.10.2024	
5.	Отримання багатокритеріальної оцінки релевантності використання алгоритму мурашиних колоній в задачах виявлення несанкціонованого доступу	28.11.2024	
6.	Оформлення пояснювальної записки	01.12.2024	
7.	Підготовка до захисту	08.12.2024	

Студент _____ Артем ІЩЕНКО
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Леонід ГАЛЬЦИНСЬКИЙ
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота обсягом 104 сторінки містить 10 ілюстрацій, 32 таблиці, 31 літературнне джерело та 1 додаток.

Метою даного дослідження є оптимізація вибору ознак для систем виявлення вторгнень за допомогою алгоритму мурашиної колонії, що дозволить підвищити точність і знизити кількість помилок при роботі систем виявлення вторгнень, та визначити релевантність зазначеного методу для оптимізації вибору ознак у системах виявлення вторгнень, порівнюючи результати з іншими методами.

Об’єктом дослідження є алгоритм мурашиних колоній (ACO).

Предметом дослідження є визначення релевантності методу мурашиних колоній у задачах виявлення несанкціонованого доступу.

У процесі виконання роботи було визначено релевантність використання алгоритму мурашиних колоній шляхом детального дослідження роботи самого алгоритму при виконанні задачі оптимізації вибору ознак в системах виявлення вторгнень та порівнянні та оцінюванні отриманих результатів із сімома іншими існуючими методами оптимізації вибору ознак в системах виявлення вторгнень за допомогою методу ELECTRE III.

Ключові слова: системи виявлення вторгнень, багатокритеріальний аналіз, алгоритм мурашиних колоній, багатокритеріальний аналіз, метод ELECTRE.

ABSTRACT

The qualification work of 104 pages contains 10 illustrations, 32 tables, 31 literature source and 1 appendix.

The purpose of this study is to optimize the selection of features for intrusion detection systems using the ant colony algorithm, which will allow to increase the accuracy and reduce the number of errors in the operation of intrusion detection systems, and to determine the relevance of the specified method for optimizing the selection of features in intrusion detection systems, comparing the results with other methods.

The object of the study is the ant colony algorithm (ACO).

The subject of the study is to determine the relevance of the ant colony method in the tasks of detecting unauthorized access.

In the process of performing the work, the relevance of using the ant colony algorithm was determined by a detailed study of the operation of the algorithm itslf when performing the task of optimizing the selection of features in intrusion detection systems and comparing and evaluating the results obtained with seven other existing methods for optimizing the selection of features in intrusion detection systems using the ELECTRE III method.

Keywords: intrusion detection systems, multi-criteria analysis, ant colony algorithm, multi-criteria analysis, ELECTRE method.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDS – системи виявлення вторгнень

NIDS – мережеві системи виявлення вторгнень

GrIDS – графові системи виявлення вторгнень

OIDS – оперативні системи виявлення вторгнень

HIDS – системи виявлення вторгнень на основі хосту

IPS – системи запобігання вторгненням

ACO – алгоритм мурашиної колонії

ПЗ – програмне забезпечення

LAN – локальна мережа

ОС – операційна система

MCDA – метод багатокритеріального аналізу рішень

Random Forest – алгоритм машинного навчання випадковий ліс;

DoS – атака на відмову в обслуговуванні

ELECTRE – метод Elimination Et Choice Traduisant la Realite

ВСТУП

У сучасному світі зростаюча залежність від інформаційних технологій і мережевих систем призводить до збільшення загроз кібербезпеки. Системи виявлення вторгнень є ключовими компонентами в забезпеченні інформаційної безпеки, оскільки вони дозволяють виявляти і запобігати несанкціонованому доступу до інформаційних ресурсів. Однак, ефективність систем виявлення вторгнень значною мірою залежить від якості алгоритму, який використовується для навчання моделей виявлення атак. Невдалий вибір алгоритму може призвести до низької точності та великої кількості помилкових спрацьовувань, що, в свою чергу, впливає на загальну надійність системи, тому життєво важливим є системний та впорядкований підхід до їх відбору. У свою чергу релевантність обраного алгоритму можна оцінити на множині ознак, що застосовуються для тренування моделей виявлення атак. Незважаючи на наявність певного набору алгоритмів, які використовуються для тренування при виявленні атак, питання який з найкращий є відкритим. Тому пошуки нових підходів, зокрема евристичних і відповідне їх оцінювання залишається актуальним.

Актуальність роботи. Тема дослідження обумовлена необхідністю додаткових досліджень з метою виявлення оптимального алгоритму для підвищення ефективності функціонування IDS шляхом оптимізації вибору ознак. Одним із перспективних підходів до розв'язання цієї проблеми є використання алгоритму мурашиної колонії (ACO), що відтворює поведінку мурах під час пошуку припасів для колонії та може бути адаптований для вибору найбільш релевантних ознак для навчання IDS. Як показують існуючі дослідження, використання ACO цілком можливо як для цієї задачі [1], та і для інших складних завдань, таких як криптоаналіз [21].

Мета роботи. Метою даного дослідження є оптимізація вибору ознак для систем виявлення вторгнень за допомогою алгоритму мурашиної колонії, що дозволить підвищити точність і знизити кількість помилок при роботі IDS та визначення релевантності зазначеного методу для оптимізації вибору ознак у системах виявлення вторгнень, у порівнянні з іншими популярними методами.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- дослідити теоретичні джерела за проблематикою дослідження;
- визначити основні підходи до вибору ознак в IDS у системах захисту від несанкціонованого доступу;
- вивчити принципи роботи ACO та його застосування в задачах оптимізації;
- адаптувати алгоритм мурашиної колонії для оптимізації вибору ознак у системах виявлення вторгнень;
- провести експерименти для оцінки ефективності оптимізованого вибору ознак на основі ACO, порівнюючи результати з іншими методами;
- оцінити вплив оптимізації вибору ознак на точність і надійність систем виявлення вторгнень.

Об’єктом дослідження є дослідження проблематики ефективного відбору ознак в IDS у системах захисту від несанкціонованого доступу.

Предметом дослідження є визначення релевантності методу мурашиних колоній у задачах виявлення несанкціонованого доступу.

Результати досліджень, наведених у роботі, оприлюднено та опубліковано у матеріалах VI International Scientific and Practical Conference «RICERCHE SCIENTIFICHE E METODI DELLA LORO REALIZZAZIONE: ESPERIENZA MONDIALE E REALTÀ DOMESTICHE», ст. 160-169.

1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО IDS ТА АТАКИ НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Системи виявлення вторгнень (IDS) є важливою частиною сучасних підходів до забезпечення кібербезпеки, що можуть бути використані для знаходження, аналізу та реагування на потенційні загрози в інформаційних системах, що є програмним чи апаратним засобом. Виявлення вторгнень передбачає спостереження за подіями в комп'ютерних системах чи мережах та аналіз їх для виявлення ознак потенційних інцидентів, які можуть свідчити про порушення політик безпеки, правил використання або стандартних практик захисту інформації. Такі інциденти можуть бути викликані різними факторами: зловмисним програмним забезпеченням (наприклад, вірусами, шпигунськими програмами), хакерами, які отримують несанкціонований доступ через Інтернет, або користувачами системи, що зловживають своїми правами чи намагаються отримати доступ до привілеїв, які їм не належать. При цьому не всі інциденти є умисними: наприклад, користувач може випадково вказати неправильну адресу пристрою й ненавмисно спробувати під'єднатися до іншої системи без дозволу[2].

Підключати та налаштовувати окремі засоби виявлення вторгнень у загальну систему виявлення вторгнень. Основними рекомендаціями з реалізації є послідовне поєднання окремих засобів виявлення вторгнень у систему в систему загального виявлення вторгнень, що забезпечує додаткове охоплення та ефективні можливості виявлення. Інформація, яка міститься в одному засобі виявлення вторгнень, може бути широко розповсюджена в організації, що робить систему загального виявлення більш стійкою та потужною[20].

Загалом, основні рішення у галузі кібербезпеки можна поділити на дві основні категорії: технічні та нетехнічні. До нетехнічних рішень можна віднести фізичні та організаційні. Захист території, фізична безпека

обчислювальних пристроїв, план аварійного відновлення центру обробки даних та створення резервної копії системи можуть зіграти вирішальне значення для збереження фізичної безпеки [27]. Адміністративне управління також є іншими важливими поняттями в нетехнічних рішеннях. Політики, процедури, стандарти, оцінка ризиків, керування постачальниками, розподіл обов'язків за принципом need-to-know та організації навчань особового кладу також є важливими поняттями під час забезпечення захищеності інформації на об'єкті [28]. Навіть якщо фахівцями з кібербезпеки буде побудовано сучасну та багатошарову систему захисту, належної безпеки не буде забезпечено, якщо користувачі цієї системи не будуть належним чином навчені.

Технічні рішення поділяються на три групи: технології та платформи, використовувані інструменти, застосування та науки про дані. Основні технології та платформи наведено на рисунку нижче**.**

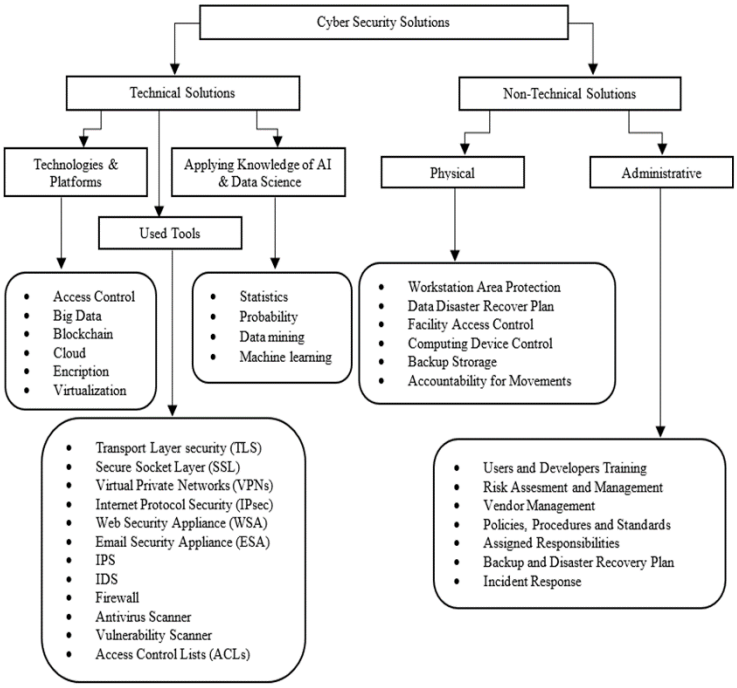

Рис. 1.1. – Представлення технічних і нетехнічних рішень кібербезпеки

Криптографія забезпечує цілісність та конфіденційність даних під час зберігання на диску та під час передачі. Контроль доступу обмежує доступ до даних, що підвищує безпеку, одночасно зменшуючи можливість віддаленої атаки, а також підвищення привілеїв. Великі дані дозволяють аналізувати

велику кількість даних, щоб виявити невідомі шаблони, а також шкідливі характеристики атак.

Нові технології, такі як блокчейн, віртуалізація та великі дані, також починають широко використовуватися в кібербезпеці. Наприклад, технологія блокчейн допомагає перевіряти узгодженість даних, а також виявляти деякі складні атаки. Технології віртуалізації відокремлюють програмні додатки від апаратних компонентів, що підвищує зручність використання програмного забезпечення та знижує вартість, а також скорочує час простою у разі кібератак. Проактивне керування загрозами, розширена безпека даних, масштабованість, висока доступність і ефективне відновлення даних - усі функції платформи хмарних обчислень.

Різноманітні добре відомі інструменти та протоколи допомагають виявляти, запобігати та зменшувати кількість атак. Списки контролю доступу можуть фільтрувати пакети на основі IP-адрес джерела, портів і протоколів. Брандмауери також фільтрують пакети на основі попередньо визначених правил. IDS та IPS використовуються для виявлення вторгнень на основі сигнатур або нетипових подій в інформаційних мережах [29].

1.1 Класифікація IDS.

Типи IDS можуть змінюватися від автоматизованих систем класу «1» до локальних мереж чи мереж з доступом до мережі Інтернет, залежності від поставлених завдань. Найпоширенішими класифікаціями є:

За характером реакції на загрозу:

• Пасивні – системи виявлення, які після виявлення підозрілого трафіку лише інформують користувача або адміністратора про можливу небезпеку;

• Активні – системи запобігання, що вживають заходів проти вторгнення, наприклад, скидаючи з'єднання або змінюючи налаштування брандмауера для блокування підозрілого трафіку;

• Гібридні – поєднують функції виявлення загроз і автономної протидії вторгненням.

За методиками аналізу: ·

- статистичні IDS базуються на аналізі статистичних даних. Після інсталяції адміністратор налаштовує їх, задаючи політику, яка відповідає нормальній мережевій активності. Це включає визначення типів трафіку, з'єднань між вузлами, використовуваних протоколів і портів. Якщо система виявляє відхилення від типової активності чи статистично значущі зміни в трафіку, вона інформує про це адміністратора. Головним викликом такого підходу є складність налаштування, а також ризик виникнення великої кількості хибних тривог через некоректно задані параметри.
- сигнатурні IDS здійснюють аналіз мережевого трафіку або порівнюють пакети з базою даних сигнатур (відомих характеристик атак). Основною проблемою цього підходу є застарівання бази сигнатур.
- гібридна IDS поєднує два і більше підходів для розробки IDS. Дані від агентів на хостах комбінуються з мережевою інформацією для створення найбільш точного представлення про безпеку мережі.

За рівнем виявлення атак:

- Мережеві системи виявлення вторгнень (NIDS). Відстежує вторгнення, перевіряючи мережевий трафік і веде спостереження за декількома хостами. Мережева система виявлення вторгнень отримує доступ до мережевого трафіку, підключаючись до концентратора або комутатору, налаштованому на дублювання портів, або мережевий TAP пристрій. Перевагами NIDS є велике покриття для моніторингу, та, у зв'язку з цим, централізоване управління також NIDS не впливають на продуктивність і топологію мережі. До недоліків цих систем можна віднести: завантаження системи, NIDS потребує додаткового налаштування і функціональності мережевих пристроїв. Системи NIDS не можуть аналізувати зашифровану інформацію і розпізнавати результати атак.
- Графові системи виявлення вторгнень (GrIDS). Ця система являється удосконаленою версією NIDS. У кожний сегмент встановлюється свій снифер.

Інформація від них збирається разом, аналізується і представляється у виді схеми інформаційних потоків. Усі NIDS не залежать від типу використовуваної в мережі ОС. Для роботи їм необхідний виділений вузол у контрольованому сегменті і мережевий адаптер, який уміє приймати усі типи пакетів. Логічним вирішенням буде встановлення захищеного з'єднання між NIDS і консоллю управління. ·

- оперативні системи виявлення вторгнень (OIDS). Система спеціалізується на внутрішніх атаках. Ці системи розробили на випадок, якщо зловмиснику вдалося увійти в систему від імені зареєстрованого користувача. Або, коли атака на мережу відбувається зсередини її самої. Система порівнює дії конкретного користувача у даний момент часу з його звичайними діями, і у разі великих розбіжностей видає повідомлення. Простіше кажучи, оцінюється типовість дій (операцій) кожного з користувачів, в той час, коли NIDS оцінює типовість трафіку.

- системи виявлення вторгнень на основі хосту (HIDS). Ця система працює з інформацією, зібраною всередині одного комп'ютера. Таке розташування дозволяє HIDS аналізувати діяльність з великою вірогідністю і точністю, визначаючи тільки ті процеси і користувачів, які мають відношення до конкретної атаки в ОС. HIDS зазвичай використовують інформаційні джерела двох типів: результати аудиту ОС і системних журналів подій. HIDS здатні відстежувати події безпосередньо на хості та виявляти атаки, які залишаються поза зоною видимості NIDS та можуть функціонувати в системі, в якій мережевий трафік зашифрований, і система не вимагає додаткової функціональності мережевих пристроїв. До недоліків цієї системи відноситься: високе завантаження системи хосту, мале покриття для моніторингу, не мають централізованого управління і вони можуть бути блокованими деякими DoS-атаками або навіть заборонені.

- зовнішні системи виявлення вторгнень ERIDS. Приклад інноваційної та вузькоспеціалізованої системи. Необхідність її створення була продиктована тим фактом, що крім простого і розподіленого способу збору

даних про мережі існують менш тривіальні. Наприклад, зловмисник спочатку здійснює атаку на маршрутизатор, змінює його налаштування так, що він направляє трафік через сегмент, який не контролюється і доступний нападнику [3].

1.2 Структура IDS.

Структура IDS може варіюватися залежно від типу та методу виявлення, але загалом вона включає кілька ключових компонентів. Основні елементи структури IDS:

- Датчики (Sensors): Датчики є основними компонентами IDS, які відповідають за збирання відомостей про трафік у мережі, системні журнали, або інші події в системі. Вони можуть бути розташовані на мережевих пристроях або на самих комп'ютерах, що моніторяться [4].
- Аналізатор (Analyzer): Цей компонент обробляє інформацію, отриману від датчиків та може охоплювати алгоритми для знаходження незвичних подій або підписів (signature-based detection). Аналізатор може бути реалізований у вигляді програмного забезпечення або апаратних рішень [5].
- Консоль управління (Management Console): Консоль надає адміністраторам можливість переглядати результати аналізу, налаштовувати параметри системи та реагувати на інциденти. Вона може також включати функції для генерації звітів та моніторингу стану системи.
- База даних (Database): IDS часто використовує базу даних для логування даних проте, що відбулося, та для зберігання підписів або моделей аномалій. Ця інформація може бути використана для подальшого аналізу та вдосконалення системи [6].
- Модуль реагування (Response Module): Цей модуль відповідає за автоматичну або ручну реакцію на виявлені загрози. Реакція може включати

блокування трафіку, підключення до зовнішніх служб для попередження або вжиття інших заходів.

•Керування політиками (Policy Management): Це компонент, що визначає правила та політики для виявлення загроз. Це може включати налаштування чутливості датчиків, визначення типів трафіку, які підлягають моніторингу, і відповідні реакції на виявлені загрози [7].

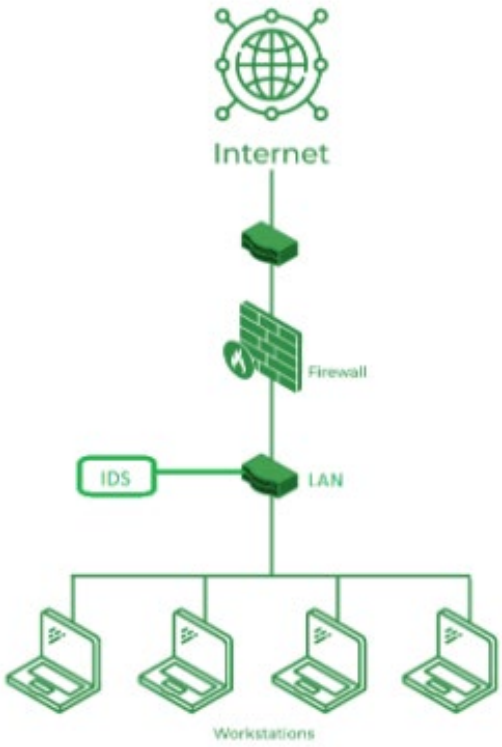

Рис. 1.2. – Структура IDS

1.3 Основні функції IDS

Системи виявлення вторгнень виконують ряд важливих функцій, які дозволяють ідентифікувати потенційні атаки на різних рівнях інформаційної системи. Основні функції IDS можна розподілити на наступні категорії:

- моніторинг та збирання даних: IDS неперервно здійснює моніторинг мережі або окремі її хости, збираючи дані про трафік або події на хостах для

подальшого аналізу. це може включати збір інформації про пакети, підключення, лог-файли та системні події.

- аналіз даних: після збирання даних IDS проводить аналіз для виявлення підозрілих подій. це може бути виконано за допомогою сигнатур, що дозволяє порівнювати зразки трафіку з відомими шаблонами атак, або за допомогою методів машинного навчання для виявлення аномалій у поведінці користувачів і мережі.
- виявлення атак: IDS виявляє атаки, коли вхідний трафік або події на хості відповідають визначеним критеріям аномалії або шаблонам атак. це може включати виявлення DOS-атак, сканування портів, зловмисного програмного забезпечення або інших шкідливих дій.
- повідомлення та реагування: після детекції аномалії IDS може відправляти попередження адміністраторам безпеки для швидкої реакції. у деяких випадках IDS може автоматично блокувати атаки або ізолювати аномальні сеанси зв'язку [8].

1.4 Атаки несанкціонованого доступу: Методи та загрози

Несанкціонований доступ до систем або даних становить одну з найбільш серйозних загроз для безпеки інформаційних систем. Зловмисники мають змогу використовувати багато різних методів з метою незаконного проникнення до систем, серверів або отримання конфіденційних даних, що потенційно може призвести до витоку інформації, порушення роботи систем або навіть повного знищення даних.

Атаки типу Remote to Local (R2L). R2L (Remote to Local) атаки виникають, коли зловмисник з віддаленого місця намагається отримати доступ до локальної системи через вразливості у протоколах мережевої безпеки або неправильні налаштування. Це може бути реалізовано шляхом підбору паролів або експлуатації слабких місць у протоколах автентифікації.

Одним із поширених методів є фішинг, коли зловмисник надсилає користувачеві модифіковані або підмінені електронні листи або використовує фальшиві вебсайти для отримання облікових даних. Після того, як зловмисник отримує облікові дані, він може увійти до системи як легітимний користувач та виконати подальші атаки. Фішинг набув широкого поширення як один із найефективніших методів несанкціонованого доступу до інформаційних систем. Зокрема, він становить значну загрозу для великих організацій, де використання соціальної інженерії дозволяє зловмисникам обманом отримувати конфіденційну інформацію, паролі чи доступ до корпоративних ресурсів. Цей метод часто залишається успішним через людський фактор, адже співробітники, не підозрюючи підступу, можуть відкривати шкідливі посилання чи розголошувати особисті дані [9].

Згідно з дослідженням, R2L атаки є вкрай шкідливими, оскільки вони можуть бути непомітними для звичайних користувачів, особливо якщо атаки реалізуються через соціальну інженерію або приховані у фальшивих повідомленнях [9].

Атаки типу User to Root (U2R). U2R (User to Root) атаки відбуваються, коли зловмисник, маючи обмежений доступ до системи, намагається вдосконалити свої привілеї (права доступу) та отримати доступ на рівні адміністратора або кореневого користувача. Такі атаки зазвичай здійснюються через використання вразливостей в операційних системах або експлуатацію помилок у програмному забезпеченні.

1.5 Вибір ознак в IDS

Вибір ознак – це процес, який вибирає підмножину ознак з усіх початкових ознак, що є цілком важливим через досить велику кількість ймовірних нерелевантних ознак у наборі даних. Також, вибір ознак можна досліджувати як задачу оптимізації для оптимальної підмножини ознак, які краще задовольняють поточні потреби для виконання специфічних завдань.

Якість оптимізації при виборі ознак за своєю сутністю є NP-складною задачею, яку можна виміряти за допомогою певних критеріїв оцінювання, але, станом на сьогодні, не існує оптимального еталонного рішення для пошуку оптимальної підмножини ознак. Типовий процес відбору ознак включає:

1. Створення підмножини ознак.
2. Оцінка обраної підмножини ознак.
3. Визначення критеріїв завершення.
4. Оцінювання результатів роботи алгоритму[25].

Процедури вибору підмножин ознак можуть бути реалізовані за допомогою різних алгоритмів, кожен з яких вибирає підмножини ознак для оцінки на основі певної стратегії пошуку. Найпопулярнішими методами, які можуть використовуватися для даної задачі, є методи: Recursive Feature Elimination, Tree-Based Feature Selection, Gradient Boosting, SelectK Best [26], та методи випадкової або константного вибору ознак.

Random Feature Selection – метод, який випадковим чином вибирає підмножину ознак з усіх доступних ознак для подальшої обробки. Ідея полягає в тому, що випадковий вибір може скоротити кількість обчислень і уникнути використання занадто великого набору ознак, що може призвести до перенавчання.

Переваги:

- Швидкість обробки: оскільки вибір ознак відбувається випадковим чином, метод працює дуже швидко, особливо на великих наборах даних, де точніші методи можуть бути занадто ресурсомісткими.
- Простота реалізації: метод є дуже простим для впровадження і не потребує складних алгоритмів чи великих обчислювальних ресурсів. Він може бути корисним на етапах, коли потрібно швидко скоротити набір ознак.

Недоліки:

- Низька ефективність: через випадковий характер відбору, ризик того, що будуть вибрані нерелевантні або неінформативні ознаки, що може погіршити продуктивність моделі, може стати досить суттєвим.
- Відсутність повторюваності: результати можуть змінюватися при кожному запуску через рандомізацію, що робить метод менш передбачуваним і менш стабільним [15].

Попри недоліки, Random Feature Selection часто використовується як початковий етап у комплексних системах оптимізації ознак. Він може слугувати основою для подальшого застосування більш витончених методів, таких як Recursive Feature Elimination або Gradient Boosting, які аналізують важливість ознак і поступово покращують вибір. Таким чином, випадковий вибір ознак є корисним інструментом для швидкої оцінки набору даних і виявлення можливих напрямків для оптимізації.

Constant Feature Selection – метод, який спрямований на усунення ознак із низькою варіативністю, тобто тих, значення яких залишаються майже незмінними у всьому наборі даних. Такі ознаки часто не несуть значущої інформації для моделі, оскільки їхній вплив на класифікацію або прогнозування є мінімальним. Основна ідея цього методу полягає в тому, щоб зменшити розмірність даних і підвищити ефективність моделі шляхом видалення «шуму» або нерелевантних ознак.

Переваги:

- Легке усунення нерелевантних ознак: Метод дозволяє швидко видалити ознаки, які не вносять жодної корисної інформації через відсутність варіацій. Це скорочує розмірність дата сету та спрощує подальший аналіз.
- Зниження складності моделі: Видаливши неінформативні ознаки, можна уникнути перевантаження моделі зайвими даними, що сприяє побудові простіших і більш інтерпретованих моделей.

Недоліки:

•Не враховує залежності між ознаками: Метод розглядає кожну ознаку окремо, без урахування взаємодії між ними. Іноді навіть постійні ознаки можуть бути досить впливовими в поєднанні з іншими несталими ознаками.

•Можливе видалення корисних ознак: Навіть якщо ознака має постійні значення в наборі даних, вона все ще може знадобитися для моделі в певних сценаріях. Вибір лише за статистичними критеріями може стати результатом втрати важливої інформації.

SelectKBest – метод відбору ознак, який використовує статистичні критерії для оцінки важливості ознак та вибору найбільш релевантних K ознак. Кожна ознака оцінюється окремо за певним критерієм, таким як коефіцієнт кореляції, ANOVA (аналіз дисперсії), χ^2 (хі-квадрат тест) або інші статистичні тести.

Переваги:

•Швидкість і простота: SelectKBest є простим і швидким методом, оскільки кожна ознака оцінюється окремо без потреби в навчанні складних моделей.

•Гнучкість у виборі критеріїв: Можна обирати різні статистичні критерії залежно від типу даних і завдання, що дозволяє адаптувати метод до різних сценаріїв [15].

Недоліки:

•Ігнорує взаємозв'язки серед ознак: SelectKBest оцінює кожну ознаку незалежно, що може бути проблематичним для задач, де важливі взаємозв'язки між ознаками.

•Необхідність власноруч обирати статистичні критерії для тестування: Потрібно правильно обрати статистичний критерій для тестування ознак. Неправильний вибір може знизити ефективність.

Tree-Based Feature Selection – це метод відбору ознак, який використовує модель для ітеративного видалення найменш важливих ознак, поки не буде досягнута бажана кількість ознак.

Переваги:

- Висока точність відбору ознак: RFE використовує модель для оцінки важливості ознак, тому видалення відбувається на основі реальної впливовості ознак на точність моделі. Це робить метод досить точним для відбору найбільш релевантних ознак.
- Скорочення обчислювальної складності: Видалення нерелевантних або малозначущих ознак дозволяє зменшити обсяг даних, що може прискорити навчання моделей і знизити вимоги до обчислювальних ресурсів.

Недоліки:

- Високі обчислювальні витрати: RFE є обчислювально-дорогим методом, оскільки вимагає багаторазового навчання моделі для кожної ітерації відбору ознак. Це може бути цілком проблематичним для об’ємних наборів даних або моделей з високими обчислювальними витратами.
- Не підходить для великих наборів ознак: Через ітеративну природу метод може бути непридатним для наборів даних з великою кількістю ознак, оскільки кількість ітерацій буде дуже великою[16].

Gradient Boosting – метод машинного навчання, який створює сильну модель на основі послідовного додавання слабких моделей (зазвичай дерев рішень). При цьому моделі додаються таким чином, щоб виправляти помилки попередніх.

Переваги:

- Висока точність: Gradient Boosting є одним з найефективніших методів для відбору ознак і побудови моделей, забезпечуючи досить гарну точність за рахунок врахування складних нелінійних залежностей.

•Інформативність ознак: Метод надає індекс важливості для кожної ознаки, що дозволяє оцінити їх значущість і зробити відповідний відбір.

Недоліки:

•Високі обчислювальні витрати: Через складність алгоритму та велику кількість ітерацій, Gradient Boosting може бути досить повільним, особливо для великих наборів даних[14].

Висновки до розділу 1

У першому розділі розглядаються загальні відомості про IDS, а саме: опис, класифікацію та структуру, наведено відомості про атаки несанкціонованого доступу та розглянуто найпоширеніші методи оптимізації ознак в системах виявлення вторгнень.

2 ОПИС ОСНОВНИХ АЛГОРИТМІВ ОПТИМІЗАЦІЇ, АЛГОРИТМ МУРАШИНИХ КОЛОНІЙ

2.1 Евристичні алгоритми оптимізації

За останні десятиліття були винайдені нові методи оптимізації, які концептуально відрізняються від традиційних. Більшість з них засновані на певних характеристиках, властивих біологічним, молекулярним, фізичним, нейробіологічним системам.

Евристичні методи базуються на інтуїтивному мисленні та передбачають несвідомі дії, спрямовані на досягнення чітко визначених цілей. Ці підходи також відомі як методи творчого або винахідливого вирішення завдань в інженерії та інших сферах. Евристичний алгоритм – це алгоритм розв’язування задачі, правильність якого для всіх можливих випадків не доведена, але про який відомо, що він дає досить хороше рішення в більшості випадків. Насправді може бути навіть відомо (тобто доведено) те, що евристичний алгоритм формально невірний. Його все одно можна застосовувати, якщо при цьому він дає невірний результат тільки в окремих, досить рідкісних випадках, які добре виділяються або ж дає неточний, але все ж прийнятний результат.

При всьому цьому, біоінспіровані та евристичні алгоритми не можна назвати повноцінним автономним (штучним) інтелектом, бо для їх успішного користування потрібен чіткий аналіз після кожної ітерації для відстежування подальшого результату. Штучний інтелект взагалі це здатність інженерної системи одержувати, вивчати та застосовувати накопичені знання та навички, не користуючись сторонньою допомогою, що для розглянутих алгоритмів не є можливим.

Простіше кажучи, евристика – це метод, який може бути не ідеально математично точним або формально правильним, але водночас доволі ефективним і практичним.

Важливо зазначити, що на відміну від строго визначеного алгоритму для розв’язання задачі, евристика має свої характерні риси:

- Вона не гарантує знаходження найкращого розв’язку.
- Вона не гарантує знаходження розв’язку, навіть якщо воно явно існує (можливий «пропуск цілі»).
- Вона може дати невірний розв’язок в деяких випадках.

1) Еволюційні методи

Генетичні алгоритми:

- Методи, що імітують імунні системи організмів
- Методи штучних імунних систем.
- Метод розсіювання.
- Еволюційна стратегія перетворення матриці.
- Метод динамічних сіток.
- Метод диференціальної еволюції.
- Метод, що імітує поширення бур'янів.
- Метод, що імітує поведінку зозуль.

2) **Методи ройового інтелекту**

- Метод частинок в зграї.
- **Метод мурашиних колоній.**
- Метод імітації поведінки бактерій.
- Метод бджолиних колоній.
- Метод, що імітує поведінку зграї риб в пошуках корму.
- Метод, що імітує поведінку кажанів.
- Метод, що імітує поведінку світлячків.

- Алгоритм, що імітує поведінку жаб.

3) Методи, що імітують фізичні процеси:

- Метод гравітаційної кінематики.
- Метод імітації відпалу.
- Адаптивний алгоритм імітації відпалу.
- Метод пошуку гармонії.
- Метод, який використовує закон електромагнетизму.

У класичній теорії штучного інтелекту для розв'язання окремої вузькоспрямованої задачі створюється єдина інтелектуальна система, яка володіє всіма необхідними ресурсами. Натомість у теорії полі агентних систем застосовується інший підхід: вважається, що кожний окремий агент має обмежене уявлення про загальну проблему, тому формують групу агентів та наступним кроком забезпечують їхню ефективну взаємодію. У межах концепції «колективного» інтелекту глобальна поведінка системи сприймається як загальний результат взаємодії усіх окремих агентів. [11]. Прихильники напряму «інтелекту рою», зокрема, Р. Брукс, Ж. Денебург, Л. Стилі та інші, вважають, що основними принципами є такі:

• багатоагентна система складається з популяції простих агентів, які залежать один від одного;

• кожен агент самостійно визначає свою реакцію на події в локальному середовищі та взаємодіє з іншими агентами;

• взаємозв’язки серед агентів мають горизонтальний характер, тобто немає центрального агента-супервізора, який би керував їхньою взаємодією;

• відсутні точні правила для визначення глобальної поведінки системи; колективна поведінка, властивості та структура виникають виключно внаслідок локальних взаємодій між агентами.

Станом на сьогодні з успіхом використовуються на практиці безліч алгоритмів, заснованих на інтелекті рою: «мурашині» алгоритми, алгоритми «бджолиних колоній», алгоритми, метод рою частинок, і т.п.

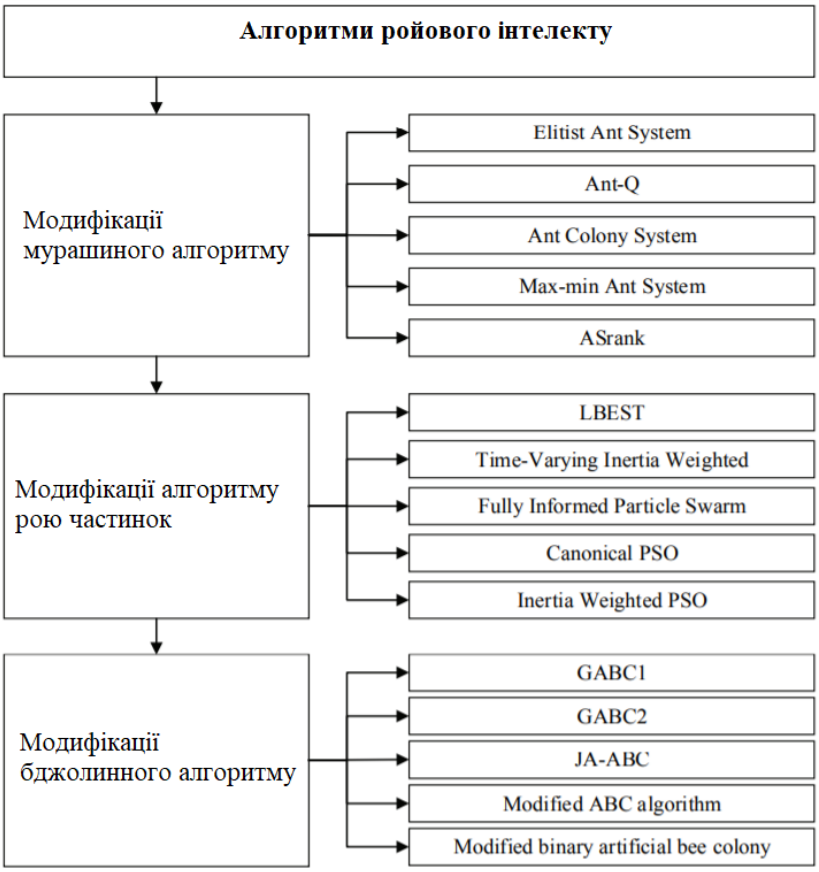

Рисунок 2.1 – Алгоритми ройового інтелекту та їх модифікації

Метод мурашиних колоній є одним з найбільш ефективних методів рішення пошукових задач комбінаторного напряму. Ідея метода складається в рішення задачі оптимізації шляхом запровадження непрямого зв’язку між автономними агентами. Перша версія цього алгоритму була запропонована Марком Доріго у 1992 році. «Мурашині» алгоритми пошуку. Основою для даного методу є математична підробка колективної поведінки мурах. Система колонії мурах заснована на нескладних патернах автономної поведінки кожного мурашки.

Незважаючи на примітивність дій мурах, поведінка всієї колонії є вкрай логічною та закономірною, тобто мурашина колонія, по суті, є природною полі агентною системою.

Опосередкований обмін, відомий як стігмергія (stigmergy), являє собою взаємодію, рознесену в часі, коли одна особина змінює певну ділянку середовища, а інші згодом використовують цю інформацію, коли потрапляють у відповідне місце. Вченими з’ясовано, що такий відстрочений взаємозв'язок відбувається завдяки субстанції – феромону (pheromone), який виділяється певними спеціалізованими залозами та залишається на поверхні під час руху мурахи. Зосередження феромону на шляху формулює пріоритет для руху саме цією траєкторією. Адаптивність поведінки забезпечується випаровуванням феромону, який у природних умовах зберігається та залишається помітним мурахам протягом кількох днів. Розподіл феромону в просторі колонії можна умовно порівняти з «колективним» сховищем пам’яті усіх мурашок, що постійно змінюється.

Кожний мурашиний алгоритм, незалежно від модифікацій, може бути представлений в наступному вигляді:

реалізуються наступні кроки алгоритму:

1) «створюємо мурах»;

2) знаходимо рішення;

3) оновлюємо феромон;

4) виробляємо додаткові дії.

Для генерації відповідного мурашиного алгоритму для вирішення будь-якої задачі, необхідно:

1) згенерувати задачу у вигляді набору компонент і переходів або набором неорієнтованих зважених графів, на яких агенти-мурахи можуть будувати рішення;

2) оцінити рівень сліду феромону;

3) визначити евристику дій мурашки, під час побудови або знаходження рішень;

4) за можливості, реалізація дієвого локального пошуку;

5) вибрати певний ACO (Ant Colony Optimization) алгоритм та застосувати його для розв'язання задачі;

6) підібрати параметри цього ACO-алгоритму.

Також найбільш суттєвими є наступні параметри:

1) чисельність мурах;

2) визначення оптимального значення між вивченням простору пошуку та використанням оптимального шляху;

3) поєднання з жадібними евристиками або локальним пошуком;

4) визначити час оновлення феромону.

«Мурашині» алгоритми є ефективним способом вирішення завдань пошуку і оптимізації, що дозволяють відображення шляхом графової системи, що підтверджується експериментальними дослідженнями. До переваг варто віднести можливість застосування до широкого спектру завдань і гарантовану збіжність. З недоліків можна відзначити сильну залежність від початкових настроювальних параметрів алгоритму, які підбираються тільки виходячи з практичного досвіду[12].

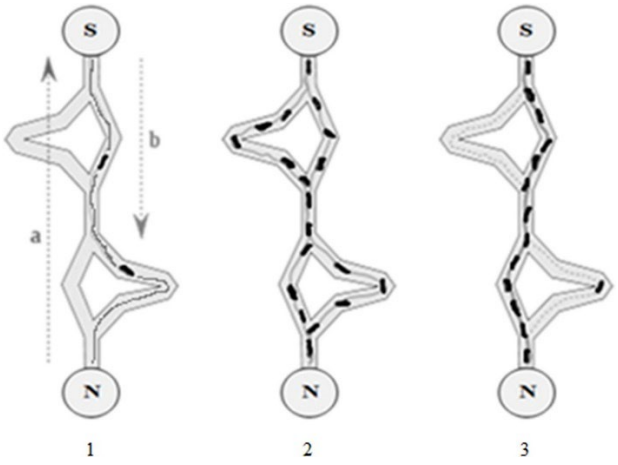

Рисунок 2.2 – Схематичний приклад виконання мурашиного алгоритму

Алгоритм мурашиної колонії представляється у вигляди графа, а мураха у виді програмного агента. Агенту надається базовий набір простих правил, які дозволять йому обирати шлях у графі. Він має список вершин (вузлів), у яких він вже був. Мураха повинна проходити через кожний вузол лише один раз.

Бджолині алгоритми. Засновані на поведінці колонії бджіл у природному середовищі. Існує два основних алгоритму – бджолиний алгоритм (Bee Algorithm) і алгоритм колонії бджіл (Artificial Bee Colony). Бджолиний алгоритм заснований на методі пошуку бджіл елітних ділянок. Основна перевага даного алгоритму - бджоли досліджують також ділянки, що знаходяться в околицях елітних, що дозволяє наблизити рішення до оптимального[10].

Найпростіший алгоритм бджіл можна уявити формально наступним чином:

1) «створення бджіл»;

2) визначення ЦФ (значення цільових функцій) бджіл;

3) вибір ділянок для пошуку;

4) відправка бджіл-розвідників;

5) вибір бджіл з кращими ЦФ;

6) відправка робочих бджіл для випадкового пошуку і визначення їх ЦФ;

7) створення нової популяції бджіл;

8) доки умови закінчення недосягнуті, повторюємо пункти 2-7.

Алгоритм колонії бджіл було представлено у 2005 році Д. Карабога. Основна ідея - імітація діяльності бджіл у вулику при пошуку нектару. Використовується фіксоване розбиття бджіл на групи - робочі бджоли, бджоли-розвідники, бджоли дослідники.

Основні кроки алгоритму колонії бджіл наступні:

1) визначення місця розташування джерел нектару;

2) пошук робочими бджолами нових джерел і дослідження кращого;

3) вибір джерела бджолою-дослідником, в залежності від якості;

4) повтор пунктів 1-3 до того моменту часу, поки рішення не перестане поліпшуватися;

5) запам'ятовуємо краще джерело;

6) заповнюємо решту популяції;

7) повторюємо пункти 2-6, поки не буде досягнута умова виходу.

Методи «бджолиного» рою підтверджують свою ефективність як евристичні полі агентні підходи для випадкового пошуку. Однак їхнім недоліком є значна кількість налаштовуваних параметрів, від яких суттєво залежить результат, при цьому немає обґрунтованих критеріїв для вибору оптимальних значень цих параметрів.

Метод рою часток.

У методі рою частинок (Particle Swarm Optimization) елементами є частинки, кожна з яких на кожному кроці має позицію і швидкість в просторі параметрів задачі. Правила зміни їх положення та швидкості визначаються на основі значення цільової функції для кожної частинки. Канонічний варіант цього методу було представлено в роботі J. Kennedy та R. Eberhart у 1995 році, і ґрунтується на принципі, що на кожній ітерації для визначення нової позиції частинки враховуються як найкращі результати від суміжних частинок, так і відомості про оптимальне значення цільової функції для цієї частинки. Існують також модифікації канонічного методу, які враховують значення цільової функції для всіх частинок рою або групують частки в кілька роїв[11].

У класичному методі рою частинок, що застосовує метричне масштабування, нова позиція частинки “i” визначається за формулою: x_i (t + 1) = x_i (t) + v_i (t + 1), де v_i (t + 1) - швидкість переміщення частинки “i” з позиції x_i (t) в позицію x_i (t + 1).

Первісний стан задається таким чином: $x_i(0)$, $v_i(0)$. Формулу представлено у векторній формі. У цьому методі кращі частинки-агенти, з точки зору цільової функції, визначаються як "центри тяжіння", до яких спрямовуються вектори швидкостей всіх частинок.

Алгоритм рою частинок активно використовується в різних галузях, зокрема в задачах машинного навчання (наприклад, для навчання нейронних мереж та розпізнавання зображень), а також у параметричній та структурній оптимізації (конфігурацій, величин і топологій) у сфері проєктування, сферах, наближених до біомеханіки. Він демонструє ефективність, порівнянну з іншими методами глобальної оптимізації, а його низька алгоритмічна складність спрощує реалізацію[11].

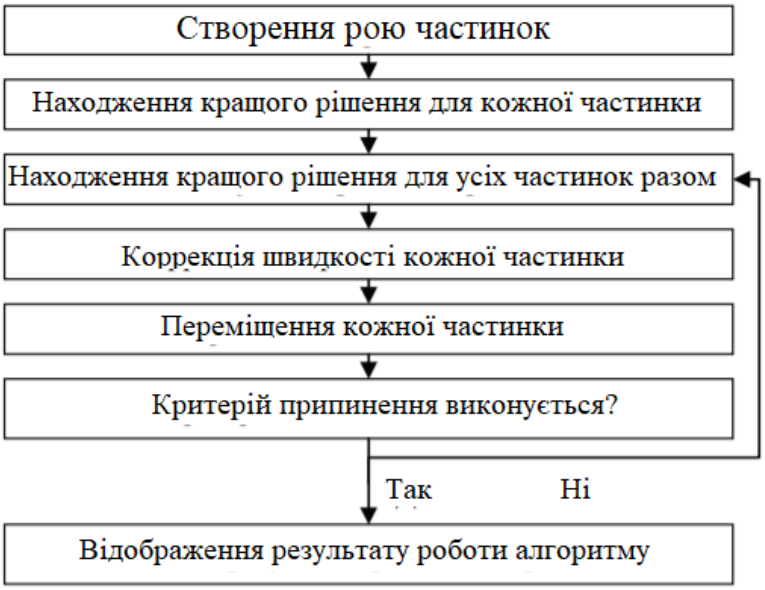

Рисунок 2.3 Схема роботи методу рою часток

Перспективними напрямками майбутніх досліджень у цій галузі є теоретичний аналіз причин збіжності алгоритму рою частинок та пов’язаних з цим питань у рамках ройового інтелекту і теорії хаосу, комбінування різних

модифікацій алгоритму для вирішення складних задач, розгляд алгоритму рою частинок як полі агентної системи, а також вивчення можливості інтеграції аналогів більш складних природних механізмів.

2.2 Метод оптимізації мурашиних колоній

Одним з евристичних методів штучного інтелекту є алгоритм мурашиної колонії. Даний алгоритм був запропонований Марко Дориго в 1992 році і імітує рух колонії мурах в природі.

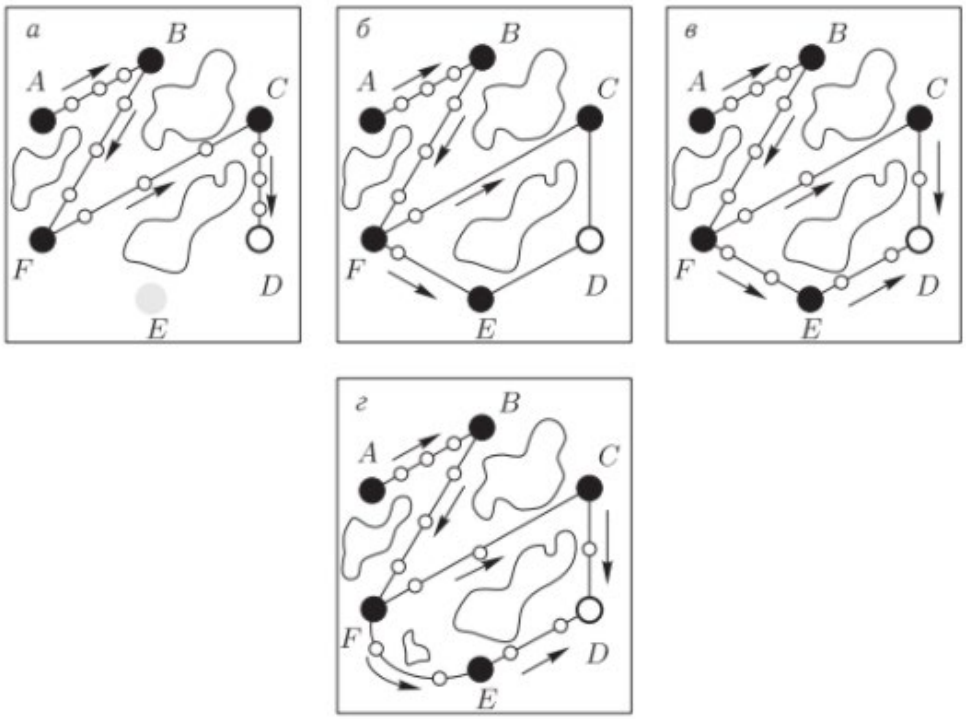

Рисунок 2.4 – Приклад обирання оптимального шляху використовуючи мурашиний алгоритм

Володіючи колективним розумом, мурахи здатні легко адаптувати себе до стану девіантності умов, як показано на малюнку 2.4 і швидко знаходити оптимальний маршрут до заданої точки (наприклад, до джерела їжі)[13].

Ідея алгоритму мурашиної колонії - моделювання сценарію поведінки пошуку маршрутів мурахами в природі.

Будь-який мурашиний алгоритм, незалежно від модифікацій і доповнень, представимо у вигляді малюнка 2.5

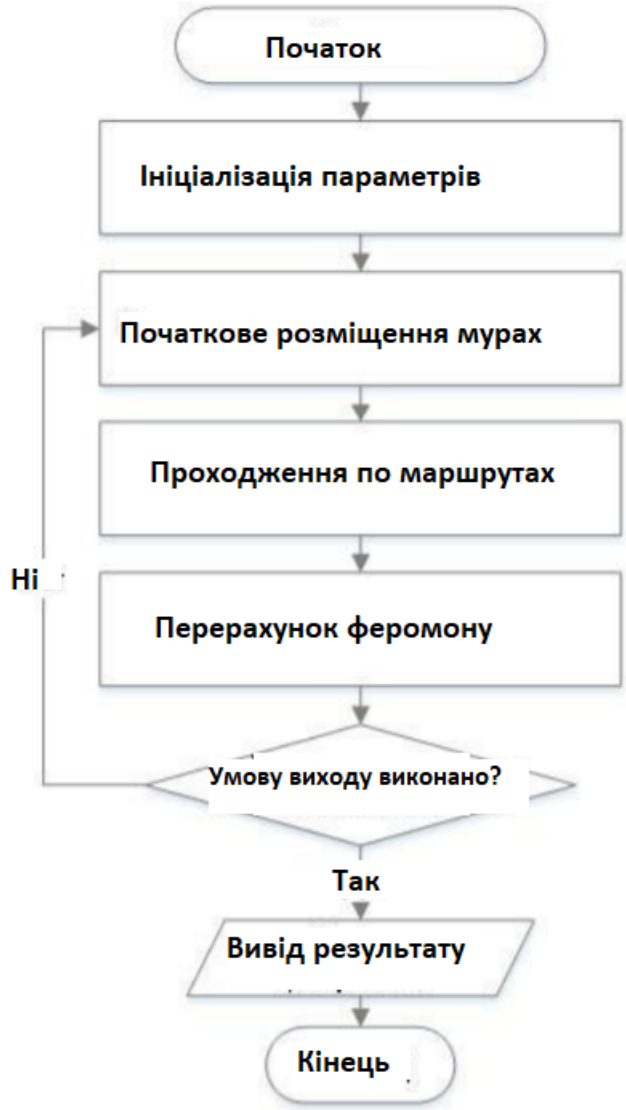

Рисунок 2.5 – Блок-схема виконання мурашиного алгоритму

Детальніше розглянемо кожен крок алгоритму:

1) Ініціалізація початкових параметрів.

Перед початком роботи алгоритму необхідно визначити кількість особин (агентів) в ройовий системі, інтенсивність, з якою будуть випаровуватися феромони, а так само ступінь залежності рішення від значення феромону і таблиці вартостей. Всі параметри визначаються дослідним шляхом.

2) Розташування мурах.

Існує безліч різних способів початкової ініціалізації колонії. Спосіб розташування безпосередньо залежить від чисельності рою і кількості міст в системі.

Для ініціалізації базового циклу, для алгоритмів еволюційних обчислень необхідно сформувати початкову популяцію рішень. Тому початкове розташування мурашиної колонії також впливає на якість рішення і збіжність мурашиного алгоритму. Тут можливі кілька стратегій формування початкової популяції:

- в кожній вершині знаходиться по одному мурашки (трудомісткість алгоритму оцінюється як O (t * n_3), оскільки n = m);
- мурахи випадково розподілені в вершинах графа;
- вся колонія знаходиться в одній вершині;
- в кожен момент часу, тобто на кожній ітерації, вся колонія переміщається в випадково обрану вершину, з якої кожен мураха починає свій маршрут.

3) Проходження по маршрутах

Вибір міста ґрунтується на наступних матрицях:

- матриці відстаней D
- таблиці феромонів T.

Феромон - спеціально хімічна речовина, яка визначає взаємодію між мурахами, які відкладають дана речовина на пройдений шлях. При виборі шляху, мурашки не бере лише факт короткого шляху, але і враховує досвід інших мурах, дану інформацію, мураха отримує з феромонів на кожному шляху. Виходить, що феромони визначають бажання мурашки зробити вибір між тим чи іншим маршрутом. Хоча при цьому підході не можна уникнути досягнення локального розв'язку. Ця виникла проблема вирішується за допомогою випаровування феромонів.

Пам'ять мурашки (список табу) - це пройдені мурахою міст, в які зайти ще раз заборонено. При використанні списку табу мураха абсолютно точно не потрапить в одне місто два рази. Цей список зростає при проходженні шляху і скидається в старті будь ітерації алгоритму. Тоді, - «Список міст, які мурашка «k», що перебуває в місті «i», ще має відвідати, є доповненням до її пам'яті. Видимість - це зворотна відстані = η_i, j = 1 / d_i, j, де D_{ij} – відстань між містами i та j. Видимість є локальною інформацію, що характеризує бажання мурашки переміститись у місто j з міста i - чим вона вища, тим ближче місто і тим сильніше та більше бажання його відвідати.

Мурахи здатні вловлювати слід феромону, в момент часу t на ребрі (i, j) вага феромону становить τ_i, j(t). Напрямок руху мурашки визначає випадкове число, яке відправляє його з позиції i в позицію j з більшою ймовірністю, якщо функція, представлена нижче, приймає більше значення.

$$\begin{cases} P_{ij,k}(t) = \dfrac{\left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum_{l \in J_{i,k}} \left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}, j \in J_{i,k} \\ P_{ij,k}(t) = 0, j \notin J_{i,k}; \end{cases} \quad (1)$$

де τ_i, j - рівень феромону, η_i, j - відстань, α, β - параметри

Залежно прийняття рішення від кількості феромону і значень матриці відстаней[11].

При α = 0 алгоритм вироджується і стає «жадібним», тобто вибір буде зроблений на користь вершини, розташованої на самому мінімальній відстані до поточної. При β = 0 вибір ґрунтується тільки на досвіді інших мурах - буде вибрано напрямок з найбільшим значенням феромону, що може привести до раннього зациклення на локальному мінімумі. Тому, експериментальним шляхом необхідно знайти компроміс між параметричними величинами.

Приклад роботи методу мурашиної колонії з різними варіаціями параметрів для задачі з 50 містами, представлений нижче в рисунку 2.6

Параметри	α = 1 и β = 1		α = 2 и β = 1		α = 5 и β = 1		α = 0 и β = 1		α = 1 и β = 2		α = 1 и β = 5		α = 1 и β = 0	
Кількість ітерацій	10	100	10	100	10	100	10	100	10	100	10	100	10	100
Краще вирішення	6501	4126	3657	3506	4321	4249	10097	9857	4015	3569	**3509**	**3471**	15551	12750
Гірше вирішення	7384	4996	4675	3814	4909	4337	11188	10481	4464	3762	**3582**	**3493**	15578	13402
Середній результат	6943	4561	4166	3660	4615	4293	10643	10169	4240	3666	**3546**	**3482**	15565	13076

Рисунок 2.6 Час роботи алгоритму мурашиної колонії

Зазначимо, що вибір міста відбувається на основі ймовірностей: правило (1) визначає значення ймовірностей переходу з міста і до інших міст. Після обчислення всіх ймовірностей випадковим чином обирається місто, яке додається до маршруту. Хоча правило (1) залишається незмінним протягом роботи алгоритму, ймовірності переходу для різних мурах можуть відрізнятися через те, що кожна мураха має свій унікальний список незаборонених міст.

4) Оновлення феромону.

Нехай T_k (t) - маршрут, який проходить мураха k до моменту часу t, L_k (t) - довжина пройденого маршруту, а Q - параметр, що має значення порядку довжини шляху, який є оптимальним. Тому, кількість феромону, відкладали на даному ребрі, може бути відображено наступною функцією:

$$\Delta\tau_{ij,k} = \begin{cases} \frac{Q}{L_k}, (i,j) \in T_k(t) \\ 0, (i,j) \notin T_k(t) \end{cases} \quad (2)$$

де Q - параметр, що має значення порядку ціни оптимального рішення, тобто Q / L_k - феромон, що відкладається k-им мурахою, що використовують ребро (i, j).

Ваги феромонів, якими позначені ребра графа, змінюються - збільшуються і зменшуються. При частому проходженні мурах з якого-небудь ребра значення ваг збільшується. Якщо ж ребро незадіяне, значення ваги з

часом зменшується - феромон «випаровується». Швидкість процесу зменшення ваги залежить від параметра p і вимагає окремого розгляду. Як представлено у наступній формулі:

$$\tau_{ij}(t+1) = (1-p)\tau_{ij}(t) + \Delta\tau_{ij} \quad (3)$$

Де ρ ∈[0,1] - інтенсивність випаровування, L_k (t) - ціна поточного рішення для k-ого мурашки.

Якщо швидкість буде занадто велика, рішення швидко вироджується.

Критерієм зупинки може бути час життя колонії - кількість скоєних ітерацій або досягнення встановленого значення мінімуму для шуканого рішення[16].

Часова складність цього алгоритму залежить від тривалості існування колонії t (число ітерацій), кількості вершин графа n і числа мурах m, і визначається як O (t * n2 * m).

Варіації мурашиних алгоритмів

Описана вище модель мурашиних обчислень не є єдиною, з моменту її створення були розроблені і інші моделі, засновані на тій же принциповій ідеї. Перша модель отримала назву мурашиної системи (Ant System, AS).

Нижче наведені найпопулярніші варіації мурашиного алгоритму:

• Елітарна мурашина система. Із загальної кількості мурах виділяються так звані «елітні мурахи». За результатами кожної ітерації алгоритму проводиться посилення кращих маршрутів шляхом проходу за даними маршрутами елітних мурах і, таким чином, збільшення кількості феромонів на даних маршрутах. У такій системі кількість елітних мурах є додатковим параметром, що вимагає визначення. Так, для дуже великого числа елітних мурах алгоритм може «застрягти» на локальних екстремумах.

Модель елітної мурашиної системи є розвитком стандартної мурашиної системи. Її головні відмінні риси:

– оновлення феромонами проводиться тільки для одного кращого шляху, знайденого або на останній ітерації, або на всіх попередніх ітераціях;

– концентрація феромону на ребрах графа обмежується і знизу і зверху.

Оновлення феромону в даній моделі проводиться згідно з формулою:

$$\tau_{ij} \leftarrow \left[\rho\tau_{ij} + \Delta\tau_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}} \quad (4)$$

де τ_{min} і τ_{max} - мінімальна і максимальна концентрація феромону, а оператор $[x]_a^b$визначається наступною формулою:

$$[x]_a^b = \begin{cases} a, \ if \ x < a, \\ b, \ if \ x > b, \\ x, \ if \ a \ \leq x \ \leq b. \end{cases} \quad (5)$$

Приріст феромону в цій моделі обчислюється тільки для кращого знайденого шляху:

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L_{best}}, if \ edge \ (i,j) \ enters \ the \ best \ way \\ 0, otherwise \end{cases} \quad (6)$$

де L_{best} - довжина кращого шляху.

• MMAS (Min-Max мурашина система) Додаються граничні умови на кількість феромонів (τ_{min}, τ_{max}). Феромони відкладаються тільки на глобально кращих або кращих в ітерації шляхах. Всі ребра ініціалізуються значенням τ_{max}.

• Рангова мурашина система. Всі рішення ранжуються за ступенем їх придатності. Кількість відкладаємих феромонів для кожного рішення зважено так, що більш релевантні рішення отримують більше феромонів, ніж менш релевантні.

• Тривала ортогональна колонія мурашок (COAC). Механізм відкладення феромонів COAC дозволяє мурашкам шукати рішення спільно і ефективно. Використовуючи ортогональний метод, мурахи можуть досліджувати обрані

області швидко і ефективно, з розширеною здатністю глобального пошуку і точністю [13].

2.3 Відбір ознак для системи виявлення вторгнень за допомогою ACO

Основна ідея ACO полягає у моделюванні процесу відбору ознак як пошуку шляху мінімальної вартості в графі, де кожен вузол представляє окрему ознаку, а ребра – вибір наступної ознаки. Кожен "мураха" у процесі вибору проходить через цей граф, залишаючи феромонний слід, що відповідає корисності певних ознак.

Чим більше мурах обирають конкретну ознаку, тим сильнішим стає її феромонний слід, що підвищує ймовірність її вибору іншими мурахами в наступних ітераціях. Це дозволяє алгоритму поступово зосереджуватися на найбільш релевантних ознаках і виключати зайві або шумові дані.

Також, ACO має здатність одночасно враховувати як феромонний слід, так і поточну інформацію про ефективність ознаки. Це дозволяє алгоритму збалансувати між експлуатацією вже знайдених рішень і дослідженням нових комбінацій ознак, що підвищує шанси на знаходження оптимального рішення.

$$P_{k,i}(t) = \begin{cases} \dfrac{[\tau_i(t)]^{\alpha} * [\eta_i]^{\beta}}{\sum_{u \in J_k} [\tau_u(t)]^{\alpha} * [\eta_u]^{\beta}}, if\ i \in J_k \\ 0 \qquad\qquad\qquad\qquad , otherwise \end{cases} \qquad (7)$$

де J_k - це множина доступних ознак, які мураха k може додати до свого підмножини, τ_i та η_i відповідно є значенням феромону та евристичною інформацією, пов'язаними з ознакою і, параметри α та β визначають відносну вагу феромону та евристичної інформації.

Ймовірність переходу, яку використовує ACO, є балансом між інтенсивністю феромону – τ_i, і евристичною інформацією – η_i. Найкращий баланс між експлуатацією та дослідженням досягається через правильний вибір параметрів α та β.

Якщо α=0, інформація про феромон не використовується, тобто попередній досвід пошуку ігнорується. Пошук тоді перетворюється на стохастичний жадібний пошук. Якщо β=0, привабливість (або потенційна вигода) ходів ігнорується.

Процес відбору ознак за допомогою ACO складається з кількох етапів:

1) Ініціалізація: кожен мураха випадковим чином вибирає одну з ознак.

2) Пошук: мурахи проходять через граф ознак, поступово будуючи підмножини ознак на основі ймовірностей вибору, які залежать від сили сліду, що залишає феромон та евристичної інформації.

3) Оновлення феромонів: після проходження мурах по графу здійснюється оновлення феромонного сліду для підсилення вибору успішних ознак і ослаблення сліду для менш корисних ознак.

4) Оцінка результатів: вибрані підмножини ознак оцінюються на основі продуктивності класифікатора (точність, рівень хибних спрацьовувань), і вибирається найкраща комбінація ознак [17].

Висновки до розділу 2

У другому розділі розглядається класифікація евристичних алгоритмів, їх опис та окремі приклади. Увагу акцентовано саме на алгоритмах ройового інтелекту, їх різновидах та прикладах, а саме, алгоритмі мурашиних колоній (ACO). Описані загальні принципи роботи мурашиного алгоритму, представлені схеми та таблиці, що відображають його роботу. Також представлено алгоритм здійснення відбору ознак для системи виявлення вторгнень за допомогою мурашиного алгоритму.

Згідно з дослідженими даними, ACO може демонструвати високу ефективність у задачах, де рішення можна представити у вигляді графа, прикладами яких є задача комівояжера, задачі маршрутизації або вибору

підмножини ознак. Завдяки використанню феромонного сліду, алгоритм здатний знаходити наближені оптимальні рішення з високою точністю.

Разом з тим, алгоритм здатний адаптуватися до змін у вхідних даних через механізм випаровування феромону. Це дозволяє уникати зациклення на локальних мінімумах і ефективно реагувати на динамічні зміни в задачі.

Але, ефективність ACO значною мірою залежить від вибору параметрів, таких як кількість агентів (мурах), інтенсивність випаровування феромону та значення параметрів α і β, що являють у собі баланс між евристикою та феромонним слідом. Неправильний вибір вищенаведених параметрів з великою вірогідністю призводить до пониження ефективності роботи алгоритму, що підкреслює його потребу в попередньому налаштуванні.

Також, можна зазначити, що виходячи з принципів роботи алгоритму, складність якого є поліноміальною, ACO є більш ефективним для задач середнього розміру, а його обчислювальна складність може бути високою для задач великої розмірності.

3 ОЦІНЮВАННЯ РЕЛЕВАНТНОСТІ МЕТОДУ МУРАШИНИХ КОЛОНІЙ У ЗАДАЧАХ ВИЯВЛЕННЯ НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Вибір оптимальних ознак для навчання моделей є ключовим завданням у розробці IDS, оскільки він не лише впливає на точність класифікації, але й дозволяє знизити обчислювальні витрати. Серед сучасних методів оптимізації особливо виділяються евристичні алгоритми, такі як алгоритм мурашиних колоній (ACO), які демонструють високий потенціал у вирішенні задач оптимізації.

У цьому розділі буде розглянуто підхід до оцінювання релевантності ACO у задачах вибору ознак для IDS. Спочатку буде описано методику оцінювання алгоритму, далі представлено дані та їх підготовку, а також програмну реалізацію. Особливу увагу приділено багатокритеріальному аналізу, що дозволяє здійснити порівняння ACO з іншими методами, такими як Recursive Feature Elimination, Gradient Boosting, Tree-Based Feature Selection та SelectKBest.

Проведене оцінювання спрямоване на виявлення сильних сторін та недоліків ACO у контексті його застосування у задачі вибору ознак в системах виявлень вторгнень. Дослідження включає аналіз емпірично зібраних даних, оцінювання релевантності алгоритмів і програмну реалізацію.

Для оцінки ефективності ACO використовуються такі критерії, як точність (Accuracy), повнота (Recall), точність передбачення (Precision), F1-міра, кількість відібраних ознак та час виконання алгоритму. Ці показники дозволяють отримати комплексну оцінку ефективності алгоритму. Результати ACO будуть порівнюватися з іншими методами вибору ознак, такими як Recursive Feature Elimination, Tree-Based Feature Selection, Gradient Boosting, SelectKBest, Random Feature Selection, Constant Feature Selection, а також комплексно оцінюватись багатокритеріальним методом ELECTRE III.

Дослідження базуються на дата сеті KDD Cup 99, який є типовим для вивчення проблематики виявлення вторгнень. Цей набір містить 41 характеристику (ознаку), що описують мережеві пакети, і мітки, які класифікують активність як нормальну чи шкідливу. Підготовка даних передбачає нормалізацію числових характеристик, перетворення категоріальних змінних (наприклад, протоколу або типу сервісу) у зручний для аналізу формат, а також вирівнювання класів, що є критично важливим через значну нерівномірність між нормальними та шкідливими запитами.

3.1. Методика оцінювання алгоритму.

Для оцінювання релевантності алгоритму мурашиних колоній у задачах виявлення неправомірного доступу застосовували багатокритеріальний підхід, який дозволяє врахувати різні аспекти роботи алгоритму. Основними метриками оцінювання є:

- Точність класифікації (Accuracy): визначає частку правильно класифікованих прикладів у загальному обсязі даних.
- F1-міра: середнє гармонійне поміж точністю (Precision) та повнотою (Recall). Використовується для збалансованої оцінки, особливо у випадках із незбалансованими даними.
- Кількість обраних ознак: метрика, яка показує здатність алгоритму зменшити розмірність даних, залишаючи лише найбільш інформативні ознаки.
- Час виконання: визначає швидкість роботи алгоритму, що є критичним у реальних сценаріях із великими обсягами даних.
- Стабільність результатів: оцінює повторюваність результатів алгоритму за різних початкових умов.

Крім зазначених метрик, для багатокритеріального аналізу використовується метод ELECTRE III, який дозволяє врахувати значущість

кожного критерію та побудувати ранжування алгоритмів за сукупністю параметрів.

Процедура оцінювання включала такі етапи:

1) Попередня підготовка даних: забезпечення коректності вхідних даних через нормалізацію, кодування та видалення аномалій.

2) Оптимізація вибору ознак: застосування ACO для відбору релевантної підмножини ознак.

3) Тренування і тестування алгоритмів: побудова класифікаційних моделей із використанням оптимізованого набору ознак.

4) Збір метрик: обчислення точності, F1-міри, кількості обраних ознак та часу виконання.

5) Порівняльний аналіз: оцінювання результатів ACO у порівнянні з іншими методами відбору ознак.

3.2 Багатокритеріальний аналіз рішень та його методи

Станом на сьогодні, складні завдання ухвалення рішень розв’язуються за допомогою математичних моделей, статистичних методів, економічних теорій та інформаційно-крмунікаційних технологій, що дозволяють автоматизувати розрахунки та оцінювання ймовірних розв’язків.

Одним із варіантів вирішення є використання багатокритеріального аналізу (далі – MCDA) що є одним із найефективніших засобів вирішення задач у цій сфері та містить різні методи, кожен з яких має свої особливості. Цей підхід враховує як якісні, так і кількісні критерії, необхідні для формулювання найбільш оптимального розв’язку. Але, з іншого боку, його використання викликає додаткові труднощі саме на етапі аналізу, в частині необхідності правильного підбору критеріїв, від яких залежить правильна та ефективна оцінка або порівняння [22].

MCDA містить різні елементи та концепції, засновані на характері проблеми прийняття рішень, такі як :

- Альтернативи, або можливі варіанти дій;
- Атрибут, що визначається як «вимірна характеристика альтернативи»;
- Агрегація стосується «розгляду ефективності альтернативи за конкретними критеріями для прийняття рішення щодо альтернативи»;
- Змінні рішення визначаються як «компоненти вектора альтернатив»;
- Простір прийняття рішень представлено як «можливі альтернативи»;
- Заходи визначаються як «елементи, які застосовуються для кількісного оцінювання альтернативи його атрибуту шляхом присвоєння атрибутам чисел або символів»;
- Критерії визначаються як «інструменти для оцінки та порівняння альтернатив з точки зору наслідків їх вибору»;
- Уподобання визначаються як «як альтернатива задовольняє потреби особи, яка приймає рішення, щодо даного атрибута»;
- Рішення відрізняються залежно від типу проблеми, яка може включати проблеми вибору, ранжирування та сортування.

3.3 Використання методу ELECTRE III у задачі визначення релевантності методів оптимізації ознак у системах виявлення вторгнень

Враховуючи існуючі методи MCDA та з урахуванням особливостей задачі оцінювання релевантності алгоритмів для оптимізації ознак в системах виявлення вторгнень, було обрано метод Elimination Et Choice Traduisant la Realite (ELECTRE).

Метод ELECTRE належить до неконкурентних підходів багатокритеріального прийняття рішень і базується на порівнянні альтернатив з урахуванням окремих критеріїв. Основна відмінність ELECTRE від компенсаторних методів, полягає в тому, що ваги критеріїв тут відображають їхню важливість, а не слугують коефіцієнтами заміщення. Крім того, метод не дозволяє компенсувати низьке значення одного критерію високими показниками інших. Деякі дослідники розглядають ELECTRE як частково

компенсаторний підхід, проте зазвичай його відносять до неконкурентних методів

Метод ELECTRE був представлений у 1966 році Роєм Бенайуном та його колегами для вирішення конкретної практичної задачі в консалтинговій компанії SEMA. Перший детальний опис методу опублікував Рой у 1968 році, назвавши його ELECTRE I. У подальші роки різні автори вдосконалювали метод, розширюючи його теоретичну базу для встановлення відносин переваги та домінування, а також для допомоги у прийнятті рішень через систематичний аналіз.

Згодом було запропоновано кілька версій методу, найпоширенішими версіями якого є: ELECTRE I, II та III.

Хронологія розробки та вдосконалення методу ELECTRE:

- ELECTRE I став першою реалізацією методу та був представлений у 1968 році;
- У 1971 році Рой та Берт'є розробили ELECTRE II як вдосконалений варіант ELECTRE I;
- У 1978 році було представлено модифікацію ELECTRE III, яка, базуючись на принципах ELECTRE II, використовує псевдокритерії замість класичних істинних критеріїв, що дозволяє описувати відносини домінування у нечіткій формі.

Ці методи спрямовані на вибір оптимальної альтернативи, дотримуючись двох основних умов:

- Узгодження переваг для більшості критеріїв оцінки;
- Відсутність переваг, якщо хоча б один критерій перевищує допустимі межі.

Метод ELECTRE I базується на трьох ключових концепціях: пороговому значенні, індексах узгодження (concordance) і розбіжності (discordance). У методі ELECTRE II індекси узгодження і розбіжності враховують як сильні, так і слабкі зв'язки, які є повними протилежностями. Це призводить до створення двох фінальних рейтингів: сильного і слабкого.

У свою чергу, ELECTRE III перевершує попередні версії тим, що дозволяє взаємодіяти з невизначеними, неточними та нечіткими даними завдяки використанню концепції нечіткої логіки. Загалом, версії методу ELECTRE мають відмінності, наприклад, деякі підходять для вирішення задач вибору (як ELECTRE I), інші орієнтовані на ранжування (ELECTRE II та III), а окремі, менш поширені версії, такі як ELECTRE TRI, використовуються для класифікації [30].

Метод ELECTRE має численні переваги, проте йому притаманні й певні недоліки. Цей алгоритм багатьома у світі вважається одним із найкращих методів багатокритеріального прийняття рішень (MCDM) завдяки таким перевагам:

- Простота логіки;
- Повне використання інформації, що міститься в матриці рішень;
- Точний і вдосконалений обчислювальний процес;
- Застосування підходу домінування;
- Моделювання неповного знання через урахування порогів байдужості та переваги;
- Уникнення компенсації низьких значень одного критерію високими значеннями інших;
- Не компенсаторний характер, який є менш жорстким порівняно з іншими методами.

Водночас у літературі виділено низку недоліків методу ELECTRE:

- Використання порогових значень, які можуть бути довільними, але суттєво впливають на кінцевий результат;
- Трудомісткість процесу обчислень;
- Невизначеність у точності ранжування, отриманого через метод ELECTRE I;
- Неможливість роботи з чисто порядковими шкалами, оскільки для обчислення індексу розбіжності необхідна метрична шкала. Таким чином,

ELECTRE I слід використовувати лише тоді, коли критерії закодовані у числовому вигляді;

• Можливість появи феномену зміни ранжування (rank reversal) у методах ELECTRE II та III, коли додавання або видалення альтернативи може змінити порядок між іншими альтернативами.

Для дослідження було обрано модифікацію ELECTRE III у зв'язку з тим, що цей метод використовує нескладну та зрозумілу логіку, та має можливість порівняння альтернатив навіть при сильній розбіжності критеріїв, але при цьому також має інструментарій для обмеження порівнювальної між різними варіантами, тобто має нечіткий характер прийняття рішень [23]-[24].

Алгоритм роботи методу ELECTRE III є наступним:

Для кожного критерію «j» визначаються три порогові значення, Поріг байдужості (**Indifference**);Поріг переваги (**Preference**); Поріг вето (**Veto**), Разом з тим, при виборі цих параметрів повинно виконуватись наступне:

$$\text{Veto} > \text{Preference} > \text{Indifference} \qquad (8), \text{ де}$$

Поріг байдужості (i) – це найбільша дозволена різниця між значеннями критеріїв для двох альтернатив, за якої вони вважаються практично однаковими.

Поріг переваги (p): мінімальна різниця між значеннями критеріїв, за якої одна альтернатива вважається значно кращою за іншу.

Поріг вето (v): значення, за якого альтернатива повністю втрачає право домінувати над іншою через занадто великий розрив за певним критерієм.

Також, як і в інших методах, визначаються ваги важливості w_j для кожного критерію. На основі цих порогів ідентифікуються суворі, байдужі та слабкі відносини між альтернативами.

Далі обчислюється індекс відповідності та будується матриця відповідності. Індекс відповідності визначає, наскільки дві альтернативи відповідають умовам переваги за певним критерієм. Для кожної пари

альтернатив A_k і A_l та для кожного критерію «j» обчислюється індекс відповідності, який враховує значення критеріїв та пороги.

• Розрахунок індексу відповідності проводиться за нечіткою формулою, що враховує різницю між значеннями критеріїв для кожної пари альтернатив.

• Результати для кожного критерію заносяться в матрицю відповідності. Кожен елемент цієї матриці відображає відсоток критеріїв, за якими одна альтернатива принаймні не гірша за іншу. Наприклад, якщо C_{kl} = 0.6, це означає, що $\frac{6}{10}$ критеріїв підтверджує, що A_k принаймні не гірша за A_l. Загальна формула для обчислення індексу відповідності:

$$C_{kl} = \frac{\Sigma_{j \in F} * w_j}{\Sigma_{j \in J} * w_j} \quad \text{, де} \tag{9}$$

де F - це набір критеріїв, для яких A_k принаймні так само хороша, як A_l, а J є набором всіх критеріїв.

Після цього, обчислюється індекс розбіжності, який визначає, наскільки одна альтернатива гірша за іншу за певним критерієм, коли різниця між альтернативами перевищує певний поріг вето.

• Якщо для певного критерію $X_j(A_k)$ - це значення критерію для альтернативи A_k, а $X_j(A_l)$ - для альтернативи A_l, то індекс розбіжності показує, чи можна вважати різницю між ними суттєвою.

• Індекс розбіжності розраховується за допомогою нечіткої логіки, а результат записується в матрицю розбіжності.

Формула для обчислення індексу розбіжності:

$$D_{k,l} = \begin{cases} 1, & \text{якщо } |X_j(A_k) - X_j(A_l) - p_j| \geq v_j \\ |X_j(A_k) - X_j(A_l) - p_j| & \text{якщо } p_j \leq |X_j(A_k) - X_j(A_l) - p_j| < v_j \\ 0, & \text{якщо } |X_j(A_k) - X_j(A_l) - p_j| < p_j \end{cases} \tag{10}$$

Потім, обчислюються ступені випередження для кожної пари альтернатив. Ступінь випередження визначає, наскільки можна довіряти тому, що одна альтернатива є кращою за іншу.

- Ступінь випередження обчислюється за допомогою комбінування індексів узгодження та розбіжності, враховуючи, які критерії показують, що одна альтернатива принаймні не гірша за іншу.
- Якщо для пари альтернатив A_k та A_l достовірність $S_{k,l}$перевищує певний поріг, це підтверджує, що A_k переважає A_l.
- Результати заносяться в матрицю випередження, яка буде використана на наступних етапах для ранжування альтернатив.

Формула для обчислення ступеня випередження:

$$S_{k,l} = \frac{\Sigma_{j \in F^* D_{k,l}}}{|F|} \qquad (11)$$

де F – це набір критеріїв, де домінування A_k над A_l визнано достовірним.

Після вищенаведених дій, виконується попереднє сортування результатів за зростанням і за спаданням. Перше попереднє сортування виконується методом низхідної дистиляції: спочатку обираються найкращі альтернативи, поступово переходячи до найгірших. Друге попереднє сортування здійснюється за допомогою висхідної дистиляції, починаючи з вибору найгірших альтернатив і завершуючи найкращими.

За результатами порівняння результатів оптимізації ознак ACO у порівнянні з іншими поширеними алгоритмами було підготовлено дані для оцінювання (Таблиця 3.5).

3.4. Вибір інструментів

3.4.1 Вибір мови програмування:

Програмну реалізацію алгоритму виконано на мові Python, яка була обрана завдяки своїй популярності, універсальності, широкому спектру інструментів для здійснення аналізу відомостей, даних та активній підтримці спільноти. У процесі реалізації використовувались такі бібліотеки:

- Pandas – для обробки даних, їх фільтрації та маніпуляції;

- NumPy – для виконання математичних операцій і роботи з багатовимірними масивами;
- Scikit-learn – для класифікації, поділу даних на тренувальну та тестову вибірки, а також оцінки моделей;
- Matplotlib і Seaborn – для візуалізації результатів дослідження;
- RandomForestClassifier – як базова модель класифікації для оцінки релевантності вибраних ознак.

Реалізація ACO включає ініціалізацію початкових феромонів для кожної ознаки, виконання ітеративного процесу оптимізації, використання феромонного сліду для вибору найбільш релевантних ознак, а також оновлення феромонів із застосуванням механізму випаровування. Оцінка релевантності ознак здійснюється через інтеграцію ACO із класифікаційними моделями.

Цей підхід дозволяє порівняти ефективність різних алгоритмів вибору ознак для систем виявлення вторгнень та отримати дані для заповнення відповідної таблиці результатів.

3.4.2 Вибір дата сету: KDD Cup 99

Дата сет KDD Cup 99 обраний для виконання задачі виявлення вторгнень у мережі та інформаційно-комунікаційних системах. Він є найбільш використовуваним у дослідженнях систем виявлення вторгнень (IDS). Основною причиною вибору цього набору даних є його велика кількість записів та різноманіття типів вторгнень, які дозволяють ефективно тестувати методи класифікації та алгоритми оптимізації ознак.

Дата сет містить такі основні типи атак:

- DOS (Denial of Service): атаки на відмову у обслуговуванні.
- R2L (Remote to Local): спроби отримати незареєстрований доступ до віддаленої машини.

- U2R (User to Root): спроби підвищити привілеї звичайного користувача до рівня адміністратора.
- Probe: сканування мережі на наявність слабких місць.

Для аналізу обрано скорочену версію цього набору даних – 10% від повного розміру (обсягу), що все ще дозволяє отримати досить вивірені результати при скороченому часі обробки та аналізу.

3.4.3 Вибір середовища розробки

Jupyter Notebook: Для реалізації дослідження було обрано Jupyter Notebook, що є потужним інструментом для наукових обчислень і аналізу даних. Основні переваги використання цього середовища:

- Інтерактивність: Jupyter Notebook дає змогу запускати код у клітинках, що забезпечує можливість поетапного тестування та налагодження. Це має особливе значення при обробці великих наборів даних, оскільки дозволяє ідентифікувати помилки ще на початкових етапах.
- Документування експериментів: Можливість додавати коментарі та описувати кожен етап роботи безпосередньо в коді дозволяє створювати зрозумілі та добре структуровані звіти. Це полегшує подальшу презентацію роботи та обмін досвідом з колегами.
- Візуалізація даних: Jupyter підтримує вбудовану візуалізацію за допомогою бібліотек Matplotlib і Seaborn, що дозволяє легко генерувати графіки та діаграми для аналізу даних.
- Гнучкість: Середовище дозволяє використовувати різноманітні мови програмування, але особливо його оптимізовано для Python. Це дозволяє легко інтегрувати інші технології та фреймворки, якщо це необхідно.

3.4.4 Підготовка та попередня обробка даних

Підготовка та підготовча обробка даних є вкрай необхідними етапами у роботі з датасетом KDD Cup 99. Цей процес включає кілька наступних ключових кроків:

1. Завантаження даних: Спочатку дані завантажуються з CSV-файлу, що містить інформацію про мережевий трафік. Для цього використовується бібліотека Pandas, яка дозволяє легко працювати з табличними даними.

2. Визначення назв стовпців: Для полегшення подальшої роботи з даними визначаються назви стовпців, які відображають типи даних та їх значення.

3. Оцінка даних: Проводиться первинний аналіз даних, включаючи виведення загальної інформації про набір даних (типи стовпців, наявність пропущених значень, кількість записів) і статистичного опису.

4. Кодування категоріальних ознак: Оскільки алгоритми машинного навчання зазвичай працюють з числовими даними, категоріальні ознаки (наприклад, "protocol_type", "service" та "flag") потрібно закодувати. Використовується метод one-hot encoding, який створює нові бінарні стовпці для кожної категорії, що дозволяє моделі краще розуміти структуру даних.

5. Замінювання міток класів: Мітки класів у колонці "label" замінюються на числові значення: нормальний трафік (normal) отримує значення 1, тоді як всі інші типи атак (R2L, DoS, U2R) позначаються як 0. Це спрощує подальший аналіз і навчання моделі.

6. Розділення на навчальну та тестову вибірки: Дата сет ділиться на навчальну (70%) та тестову (30%) вибірки. Це дозволяє навчати модель на одній частині даних і перевіряти її ефективність на іншій, що є важливим для оцінки реальної продуктивності моделі.

3.4.5 Використання ACO для оптимізації ознак

Алгоритм мурашиної колонії реалізується для вибору оптимальних ознак, що мають вкрай високий вплив на результати класифікації. ACO є метаевристичним алгоритмом, що імітує поведінку колонії мурах, які шукають оптимальний шлях до джерела їжі. Основними кроками роботи ACO є:

1. Ініціалізація: Визначаються кількість мурах, кількість ітерацій, початковий рівень феромонів і значення параметрів, що контролюють процес. Феромони використовуються для моделювання ймовірності вибору тієї чи іншої ознаки.

2. Вибір ознак: Кожна мураха випадково обирає підмножину ознак, використовуючи ймовірності, що мають залежність до значення позначки феромонів. Чим значення феромонів більше на конкретній ознаці, тим вища ймовірність її вибору. Це допомагає виявити найбільш значущі ознаки.

3. Оцінка моделі: Для кожної підмножини ознак створюється модель класифікації (Random Forest), і її продуктивність оцінюється за метриками, такими як точність, точність класифікації, повнота та F1-метрика. Це дозволяє зрозуміти, наскільки добре працює обрана комбінація ознак.

4. Оновлення феромонів: Після оцінки кожної підмножини ознак оновлюються рівні феромонів. Успішні комбінації ознак отримують більше феромонів, збільшуючи ймовірність їх вибору в наступних ітераціях. Це створює механізм навчання, де модель адаптується до нових даних.

5. Завершення: Процес повторюється до досягнення певної кількості ітерацій або поки не буде досягнуто задовільного рівня продуктивності. У кінцевому підсумку ACO забезпечує підмножину ознак, які забезпечують найкращі результати класифікації.

3.4.6 Навчання та оцінка моделі класифікації

Після вибору оптимальних ознак проводиться навчання моделі класифікації. У роботі використовується Random Forest Classifier, що є алгоритмом ансамблевого навчання, який поєднує кілька дерев рішень для підвищення точності класифікації. Random Forest Classifier - це простий у використанні та гнучкий алгоритм машинного навчання, який зазвичай демонструє хороші результати без необхідності складного підбору гіперпараметрів. Цей алгоритм популярний через свою невисоку складність та універсальність, адже він підходить як для завдань класифікації, так й щоб виконати регресію. Випадковий ліс є методом навчання з учителем, який формує деяку множину дерев рішень, зазвичай навчених методом «мішкування». Найбільш головна задумка цього методу полягає в тому, що комбінація кількох моделей покращує загальний результат.

Однією з ключових переваг Random Forest є можливість його використання для обох типів завдань – регресії і класифікації, які становлять більшість сучасних систем машинного навчання. Алгоритм випадкового лісу має три основні гіперпараметри, які потрібно встановити перед навчанням: розмір вузла, кількість дерев і кількість вибраних об'єктів. Це дозволяє класифікатору випадкового лісу ефективно вирішувати завдання як регресії, так і класифікації.

Алгоритм випадкового лісу складається з кількох дерев рішень, кожне з яких формує вибірку даних з навчального набору з поверненням, відому як вибірка «з завантаженням». При цьому одна третина даних відкладена як тестова вибірка, або «вибірка поза мішком» (oob), до якої ми ще повернемося. Інший аспект випадковості додається через вибірку ознак, що підвищує різноманітність відомостей і оптимізує кореляцію між деревами. В залежності від типу задачі, метод прогнозування може варіюватися: для регресії прогнози отримуються шляхом усереднення результатів окремих дерев, тоді як для

класифікації - за допомогою голосування, тобто вибору найчастішої категорії як прогнозованого класу. Вибірка oob, в свою чергу, використовується для перехресної перевірки, завершуючи процес прогнозування.

Алгоритм випадкового лісу має сукупність як плюсів, так і недоліків при використанні для класифікаційних або регресійних задач. Ось деякі з них:

- Знижений ризик переобладнання: Дерева рішень є потенціально схильними до переобладнання, оскільки вони зазвичай адаптуються до вибірок навчальних даних. Проте, завдяки надійній кількості дерев у випадковому лісі, класифікатор менш схильний до переповнення моделі, оскільки усереднення некорельованих дерев знижує загальну дисперсію та помилку прогнозування.

- Гнучкість: Випадковий ліс може ефективно справлятися як з завданнями регресії, так і з класифікацією з високим рівнем точності, що робить його впізнаваним серед дослідників даних. Цей метод також дозволяє ефективно оцінювати відсутні значення, зберігаючи точність, навіть коли частина даних недоступна.

- Визначення важливості ознак: Алгоритм дозволяє легко оцінити важливість змінних у моделі. Існує декілька шляхів оцінки важливості: наприклад, важливість Джіні та середнє зменшення домішки (MDI), які вимірюють, наскільки зменшується точність моделі при виключенні певної змінної. Інший показник, важливість перестановки (MDA), визначає середнє зниження точності через випадкову перестановку значень ознак у вибірках oob.

Ключові недоліки:

- Тривалий процес: Оскільки алгоритми випадкового лісу можуть працювати з великими наборами даних, вони здатні надавати точні прогнози,

але їх обробка може бути повільною, оскільки потрібно обчислювати дані для кожного окремого дерева рішень.

- Високі вимоги до ресурсів: Метод випадкових лісів вимагає значних ресурсів для зберігання та обробки великих обсягів даних.

- Складність інтерпретації: Прогноз одного дерева рішень легше зрозуміти, ніж прогноз, отриманий від ансамблю дерев.

Основні етапи:

1. Тренування моделі: Модель Random Forest навчається на вибраних ознаках з навчальної вибірки. На цьому етапі модель створює кілька дерев рішень, кожне з яких робить прогнози на основі різних випадкових підмножин даних та ознак.

2. Тестування моделі: Після навчання модель тестується на тестовій вибірці, що дозволяє оцінити її ефективність у класифікації вторгнень. На цьому етапі моделі здійснюють прогнозування, використовуючи тестові дані, що не були залучені під час процесу навчання.

3. Оцінка моделі: Оцінювання моделі здійснюється за допомогою метрик:

- Точність (accuracy): Показує загальну частку записів, класифікованих коректно.
- Точність класифікації (precision): Вимірює частку коректно передбачених позитивних результатів серед усіх передбачених позитивних.
- Повнота (recall): Вимірює частку коректно передбачених позитивних результатів серед усіх фактичних позитивних.
- F1 Score: Гармонійне середнє між precision та recall, що забезпечує збалансовану оцінку моделі, особливо в умовах нерівномірного розподілу класів.

3.4.7 Порівняльне оцінювання алгоритмів

Оцінка моделі базується на результатах, отриманих за допомогою вказаних метрик. Залежно від отриманих значень є можливість зробити висновки про можливість використання ACO для оптимізації ознак у системах виявлення вторгнень.

3.5 Дослідження та апробація результатів використання алгоритму мурашиних колоній для оптимізації ознак в IDS:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

Рисунок 3.1 – Імпортовані бібліотеки

Завантаження нашого дата сету:

data = pd.read_csv('kddcup.data_10_percent_corrected', header=None)

Дата сет завантажується за допомогою pd.read_csv, при цьому зчитуються всі дані, а заголовки стовпців не вказуються (задано параметр header=None).

Кодування категоріальних ознак:

Використовується one-hot кодування для трансформації категоріальних змінних на числові. Параметр drop_first=True дозволяє уникнути проблеми мультиколінеарності, видаляючи перший стовпець для кожної категорії.

data_encoded['label'] = data_encoded['label'].map(lambda x: 1 if x == 'normal.' else 0)

Перетворення міток:

Цільова змінна `label` перетворюється з категоріальної в числову (1 для "normal", 0 для "attack"), що необхідно для навчання моделі.

sns.countplot(x='label', data=data_encoded, palette='Set2') plt.title('Розподіл міток вторгнень у дата сеті KDD99') plt.xticks(rotation=90) plt.show()

Візуалізація розподілу міток: Створюється стовпчиковий графік для візуалізації кількості нормальних та атакуючих міток у дата сеті. Це дозволяє побачити дисбаланс класів.

plt.figure(figsize=(12, 10)) sns.heatmap(data_encoded.corr(), annot=False, cmap='coolwarm') plt.title('Матриця кореляції') plt.show()

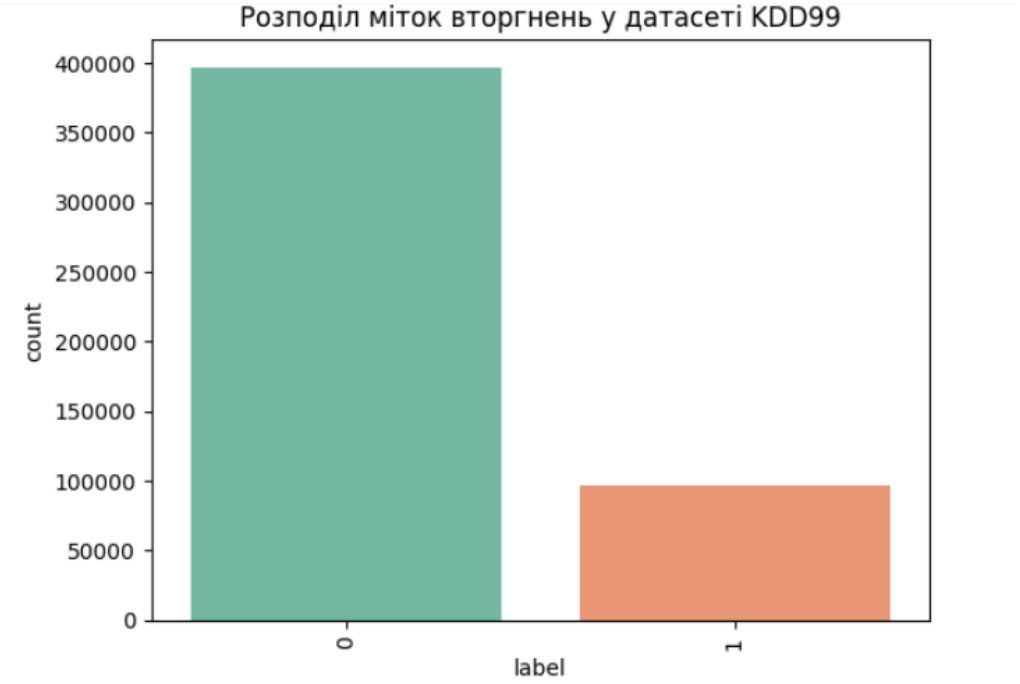

Рисунок 3.2 – Розподіл міток вторгнень у дата сеті KDD99, де 0 – нормальний трафік, 1 -атаки.

Матриця кореляції:

Матриця кореляції є важливим статистичним інструментом, що використовується для виявлення та аналізу зв'язків між кількома змінними. Вона представляє собою таблицю, в якій елементи відображають кореляційні коефіцієнти між парами змінних, надаючи можливість дослідникам оцінити, як зміна однієї змінної впливає на іншу [18]. Кореляційний коефіцієнт, зазвичай, обчислюється за формулою Пірсона, що коливається в діапазоні від -1 до 1, значення, наближені до одиниці, свідчать про сильну позитивну

кореляцію, тоді як значення, близькі до -1, вказують на сильну негативну кореляцію. Значення, близьке до 0, означає відсутність лінійної кореляції між змінними.

Матриця кореляції може бути візуалізована у вигляді теплової карти, що дозволяє дослідникам швидко інтерпретувати дані та виявляти закономірності, які можуть бути неочевидними у текстовій формі [18].

У нашому прикладі будується тепловий графік, що показує кореляції між різними ознаками. Це допомагає виявити сильні кореляції між змінними, що може бути корисним для вибору ознак:

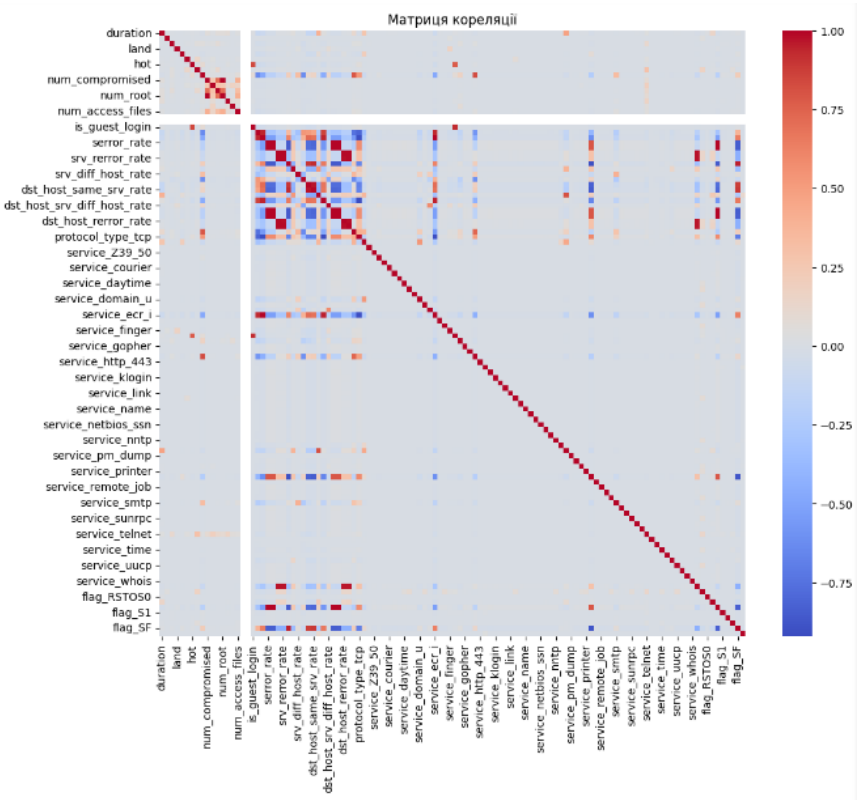

Рисунок 3.3 – Матриця кореляції

Функція оцінки моделі:

Описує функцію, яка обчислює основні метрики ефективності моделі (accuracy, precision, recall, F1-метрику) на основі фактичних та прогнозованих міток:

def evaluate_model(y_test, y_pred):

accuracy = accuracy_scr(y_test, y_pred) precision = precision_scr(y_test, y_pred, average='weighted', zero_division=0) recall = recall_scr(y_test, y_pred, average='weighted', zero_division=0) f1 = f1_scr(y_test, y_pred, average='weighted', zero_division=0) return accuracy, precision, recall, f1

ACO:

Ініціалізація класу для алгоритму мурашиної колонії (ACO). Параметри включають кількість мурах, кількість ознак, кількість ітерацій, значення alpha і beta, а також рівень випаровування феромонів. Перший масив феромонів ініціалізується одиницями.

class ACO:

def __init__(slf, n_murah, n_features, n_itratsyi, alpha, beta, evapt_rate): slf.n_murah = n_murah slf.n_features = n_features slf.n_itratsyi = n_itratsyi slf.alpha = alpha slf.beta = beta slf.evapt_rate = evapt_rate slf.pheromone = np.ones(n_features)

Метод вибору ознак: Метод генерує випадковий вибір ознак для кожної мурашки. Визначається випадкова кількість вибраних ознак (від 1 до 41), після чого створюється масив булевих значень, де True вказує на вибрану ознаку.

def select_features(slf): feature_selection = [] for _ in range(slf.n_murah):

n_selected = np.random.randint(1, 42) selected_indices =

np.random.choice(range(slf.n_features), n_selected, replace=False)

selected_features = np.zeros(slf.n_features, dtype=bool)

selected_features[selected_indices] = True

feature_selection.append(selected_features) return np.array(feature_selection)

Метод оновлення феромонів оновлює рівні феромонів на основі точності вибраних ознак. Феромони зменшуються внаслідок випаровування, і додається точність, отримана в поточній ітерації.

def update_pheromone(slf, accuracy): slf.pheromone = (1 - slf.evapt_rate) * slf.pheromone + accuracy

Метод оптимізації: Основний метод для оптимізації вибору ознак. Створюються словники для зберігання найкращих метрик та найкращих

вибраних ознак. Відбувається цикл по ітераціях, в якому генеруються вибори ознак для мурашок.

```
def optimize(slf, X, y):
best_metr = {'accuracy': 0, 'precision': 0, 'recall': 0, 'f1': 0}
best_features = None all_metr = []
for iteration in range(slf.n_itratsyi): feature_selection = slf.select_features() metr = []
```

Для кожного набору вибраних ознак здійснюється підбір тренувальних і тестових даних. Модель Random Forest навчається на тренувальних даних, а потім прогнозує мітки на тестових даних. Результати оцінюються за допомогою функції evaluate_model.

```
for features in feature_selection:
 X_selected = X.loc[:, features]
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)
model = RandomForestClassifier() model.fit(X_train, y_train)
y_pred = model.predict(X_test) metr.append(evaluate_model(y_test, y_pred))
```

Обчислення та оновлення метрик:

Середні метрики обчислюються для всіх мурашок, а найкращі метрики та вибрані ознаки оновлюються, якщо нові результати перевищують попередні. Також оновлюються феромони на основі середньої точності.

```
avg_metr = np.mean(metr, axis=0)

all_metr.append(avg_metr)

if avg_metr[0] > best_metr['accuracy']: best_metr['accuracy'] = avg_metr[0]

best_metr['precision'] = avg_metr[1] best_metr['recall'] = avg_metr[2]

best_metr['f1'] = avg_metr[3]

best_features = feature_selection[np.argmax(metr[:, 0])]

slf.update_pheromone(avg_metr[0])
```

Повернення результатів: Повертаються найкращі вибрані ознаки та відповідні метрики, які були отримано під час проведення оптимізації.

return best_features, best_metr

Підготовка матриці ознак, цільової змінної та запуск класу ACO: Змінна X містить всі стовпці, крім цільового, а y містить лише цільову змінну. Створюється екземпляр класу ACO з визначеними параметрами, і виконується оптимізація вибору ознак за допомогою методу optimize.

X = data_encoded.drop('label', axis=1)

y = data_encoded['label']

aco = ACO(n_murah=10, n_oznak=X.shape[1], n_itratsyi=50, alph=1, bet=1, evap_rate=0.4) best_features, best_metr = aco.optimize(X, y)

Запуск ACO, виведення результатів та їх візуалізація:

print("Найкращі метрики:", best_metr)

print("Вибрані ознаки:", np.where(best_features)[0])

metr_df = pd.DataFrame(all_metr, columns=['Accuracy', 'Precision', 'Recall', 'F1-Score']) metr_df.plot(kind='bar', figsize=(10, 6))

plt.title('Метрики ACO за ітераціями') plt.xlabel('Ітерація') plt.ylabel('Метрики') plt.xticks(range(len(all_metr)), range(1, len(all_metr) + 1), rotation=0) plt.show()

Аналіз результатів:

У нашому випадку використовувалися наступні параметри для мурашиного алгоритму:

```
aco = ACO(n_ants=10, n_features=X.shape[1], n_iterations=50, alpha=1.0, beta=1.0, evaporation_rate=0.5)
```

Рисунок 3.4 – значення параметрів, що використовувались для виконання в ACO

де n_murah – кількість мурах, n_features - повертає кількість ознак у вашому наборі даних X, n_itratsyi – кількість ітерацій, alpha – коефіцієнт важливості феромону у процесі відбору ознак, beta – коефіцієнт важливості «евристична інформація» у процесі відбору ознак, evapt_rate – коефіцієнт випаровування.

За результатами проведених емпіричних досліджень було отримано результати, які приведено в таблиці 1.2.

Таблиця 3.1 – Результати оцінки за усіма ознаками без застосування ACO

Результати	RandomForest
Accuracy	0.9997705
Precision	0.9997706
Recall	0.9997705
F1 Score	0.9997706

За результатами використання ACO було отримано результати, що наведено в Таблиці 3.2.

Таблиця 3.2 – Результати оцінки із застосуванням ACO для оптимізації ознак

Кількість ітерацій	**Accuracy**		**Precision**		**Recall**		**F1 Score**		**Кількість вибраних ознак**	
	max	**average**	**max**	**average**	**max**	**average**	**max**	**average**	**min**	**average**
50	0.999694	0,984624	0,999694	0,984319	0,999694	0,984624	0,999694	0,981283	24	35
Найкраща ітерація:										
№ 26	0.999694		0.999694		0.999694		0.999694		27	

При виконанні програми спостерігалась тенденція до поступового зменшення середньої кількості обраних ознак з кожною ітерацією при

відносному збереженні точності оцінювання, що свідчить про поступову адаптацію алгоритму до виконання задачі через зміни коефіцієнта феромону на кожній з ознак [31].

У фінальному найкращому варіанті після проходження 50 ітерацій, з 41 ознаки було обрано 27 ознак, які наведено в таблиці 3.3.

Таблиця 3.3 – Ознаки, обрані у найкращій ітерації

Обрані ознаки	**Важливість ознаки**
dst_bytes	0.222379
logged_in	0.128552
dst_host_srv_diff_host_rate	0.090148
src_bytes	0.087216
srv_count	0.081350
protocol_type	0.057033
diff_srv_rate	0.055224
service	0.052592
flag	0.048621
same_srv_rate	0.037743
dst_host_same_srv_rate	0.033775
dst_host_srv_count	0.026399
dst_host_diff_srv_rate	0.023303
dst_host_srv_rerror_rate	0.011033
duration	0.010026
hot	0.008849
dst_host_serror_rate	0.007583
srv_serror_rate	0.007121
dst_host_rerror_rate	0.004568
serror_rate	0.003014
srv_rerror_rate	0.002440
is_guest_login	0.000685
num_root	0.000223

Кінець Таблиці 3.3.

Обрані ознаки	Важливість ознаки
num_file_creations	0.000087
num_access_files	0.000021
urgent	0.000014
is_host_login	0.000000

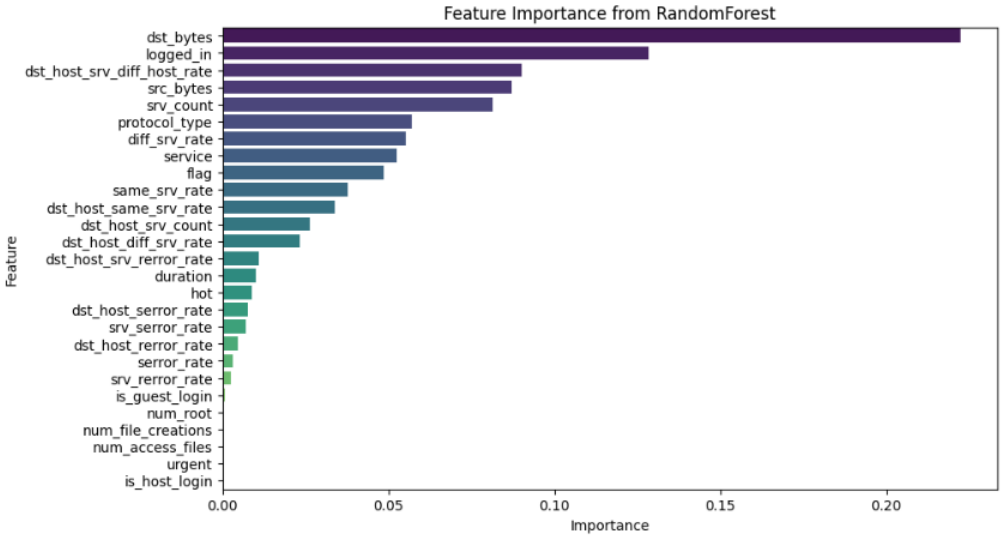

Рис. 3.5 – Важливість ознак за результатами аналізу

Значення результуючого рівня феромону наведено у Таблиці 3.4.

Таблиця 3.4 – Значення результуючого рівня феромону

№	Ознака	Коефіцієнт результуючого феромону
1	duration	0.88
2	protocol_type	0.88
3	service	0.54
4	flag	0.47
5	src_bytes	0.88
6	dst_bytes	0.88
7	land	0.79
8	wrong_fragment	0.63
9	urgent	0.76

Продовження Таблиці 3.4

№	Ознака	Коефіцієнт результуючого феромону
10	hot	0.88
11	num_failed_logins	0.76
12	logged_in	0.87
13	num_compromised	0.59
14	root_shell	0.55
15	su_attempted	0.79
16	num_root	0.66
17	num_file_creations	0.84
18	num_shells	0.41
19	num_access_files	0.87
20	num_outbound_cmds	0.78
21	is_host_login	0.88
22	is_guest_login	0.87
23	count	0.40
24	srv_count	0.52
25	serror_rate	0.87
26	srv_serror_rate	0.33
27	rerror_rate	0.41
28	srv_rerror_rate	0.88
29	same_srv_rate,	0.47
30	diff_srv_rate,	0.63
31	srv_diff_host_rate	0.76
32	dst_host_count	0.76
33	dst_host_srv_count	0.14
34	dst_host_same_srv_rate	0.53

Кінець Таблиці 3.4

№	Ознака	Коефіцієнт результуючого феромону
35	dst_host_diff_srv_rate	0.88
36	dst_host_same_src_port_rate	0.42
37	dst_host_srv_diff_host_rate	0.88
38	dst_host_serror_rate	0.88
39	dst_host_srv_serror_rate	0.81
40	dst_host_rerror_rate	0.87
41	dst_host_srv_rerror_rate	0.15

Таблиця 3.5 - Підготовлені за результатами дослідження дані для оцінювання

	ACO	Recursive Feature Eliminati-on	Tree-Based Feature Selec-tion	Gradient Boosting	Select K Best	Random Feature Selec-tion	Constant Feature Selection
Time	0,65	0,77	0,97	0,96	0,98	0,99	0,98
Accuracy	0,99	0,99	0,99	0,99	0,99	0,90	0,80
Precision	0,99	0,99	0,99	0,99	0,99	0,92	0,64
Recall	0,99	0,99	0,99	0,99	0,99	0,90	0,80
F1Score	0,99	0,99	0,99	0,99	0,99	0,92	0,72
Effectiveness	0,63	0,65	0,85	0,99	0,23	0,23	0,10
Number of signs	0,90	0,90	0,95	0,99	0,90	0,90	0,85

Для обчислення ефективності виконання алгоритму спочатку було нормалізовано дані за допомогою Min-Max нормалізації, перетворивши кожне значення критеріїв «Час», «Кількість обраних ознак» та «Точність» в діапазон від 0 до 1.

Формула для нормалізації:

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (12)$$

Після цього для обчислення ефективності моделі використано формулу:

$$\text{Ефективність} = a * Accuracy_n + b * (1 - Time_n) + c * K_n \quad (13)$$

де *a, b* та *c* - вагові коефіцієнти, що відображають важливість кожного нормалізованого значення критерію (0.6, 0.1 та 0.3 відповідно).

Час та кількість обраних ознак було перетворено, корелюючи значення від найкращого до найгіршого, використовуючи проміжок від 0 до 1, де значення кращого результату відповідно знаходиться ближче до 1 [31].

Зазначені дані для оцінок алгоритмів було оброблено за допомогою програмного забезпечення XLSSTAT на базі MS Excel.

За результатами опитування експертів було визначено вагу критеріїв та значення порогів Indifference, Preference та Veto для оцінювання, що наведено у таблицях 3.6 та 3.7 відповідно.

Таблиця 3.6 – Вага критеріїв для оцінювання, визначена експертами

Time	0,2
Accuracy	0,9
Precision	0,45
Recall	0,45
F1Score	0,5
Effectiveness	0,62
Number of signs	0,41

Таблиця 3.7 – Значення порогів Indifference, Preference та Veto

Indifference	**Preference**	**Veto**
0,20	0,25	0,35
0,01	0,02	0,10
0,02	0,04	0,10
0,02	0,04	0,10
0,01	0,02	0,10
0,09	0,11	0,20
0,04	0,06	0,20

За результатами оцінювання за допомогою алгоритму ELECTRE III складено матрицю відповідності (Таблиця 3.8), матрицю розбіжності (Таблиця 3.9) та матрицю випередження (Таблиця 3.10).

Таблиця 3.8 – Матриця відповідності

a/b	ACO	Recursive Feature Elimina-tion	Tree-Based Feature Selection	Gradient Boosting	Select K Best	Random Feature Selection	Constant Feature Selection
ACO	1,000	1,000	0,710	0,652	0,943	0,943	0,943
Recursive Feature Elimina-tion	1,000	1,000	0,766	0,708	0,989	0,977	0,989
Tree-Based Feature Selection	1,000	1,000	1,000	0,766	1,000	1,000	1,000
Gradient Boosting	1,000	1,000	1,000	1,000	1,000	1,000	1,000
SelectK Best	0,824	0,824	0,766	0,708	1,000	1,000	1,000
Random Feature Selection	0,173	0,173	0,115	0,057	0,348	1,000	1,000
Constant Feature Selection	0,115	0,115	0,057	0,057	0,115	0,115	1,000

Таблиця 3.9 – Матриця розбіжності

a/b	ACO	Recursive Feature Elimina-tion	Tree-Based Feature Selection	Gradient Boosting	SelectK Best	Random Feature Selection	Constant Feature Selection
ACO	1,000	1,000	1,000	1,000	0,000	0,000	0,000
Recursive Feature Elimina-tion	1,000	1,000	1,000	1,000	0,000	0,000	0,000

Кінець Таблиці 3.9

a/b	ACO	Recursive Feature Elimina-tion	Tree-Based Feature Selection	Gradient Boosting	SelectK Best	Random Feature Selection	Constant Feature Selection
Tree-Based Feature Selection	0,000	0,000	1,000	1,000	0,000	0,000	0,000
Gradient Boosting	0,000	0,000	0,766	1,000	0,000	0,000	0,000
SelectK Best	0,943	0,989	1,000	1,000	1,000	0,004	0,000
Random Feature Selection	0,943	0,977	1,000	1,000	1,000	1,000	0,000
Constant Feature Selection	0,943	0,989	1,000	1,000	1,000	1,000	1,000

Таблиця 3.10 – Матриця випередження

a/b	ACO	Recursive Feature Elimina-tion	Tree-Based Feature Selec-tion	Gradient Boosting	SelectK Best	Random Feature Selection	Constant Feature Selection
ACO	I	P	P	P	P	P	P
Recursive Feature Elimina-tion	NP	I	P	P	P	P	P
Tree-Based Feature Selection	NP	NP	I	I	P	P	P
Gradient Boosting	NP	NP	I	I	P	P	P
SelectK Best	NP	NP	NP	NP	I	P	P

Кінець Таблиці 3.10

a/b	ACO	Recursive Feature Elimina-tion	Tree-Based Feature Selec-tion	Gradient Boosting	SelectK Best	Random Feature Selection	Constant Feature Selection
Random Feature Selection	NP	NP	NP	NP	NP	I	P
Constant Feature Selection	NP	NP	NP	NP	NP	NP	I

Де:

1. a P b означає, що дія a є кращою перед дією b.
2. a NP b означає, що дія a не є кращою перед дією b.
3. a R b означає, що дія a не порівнянна з дією b.
4. a I b означає, що дія a байдужа до дії b.

За результатами дослідження можемо зробити наступні висновки:

- ✓ GradientBoosting є найкращим алгоритмом;
- ✓ Tree-BasedFeatureSelection зайняв друге місце;
- ✓ ACO та RecursiveFeatureElimination є однаковими по ефективності та разом займають 3 місце;
- ✓ SelectKBest – займає 4 місце;
- ✓ RandomFeatureSelection – передостанній за ефективністю;
- ✓ ConstantFeatureSelection є найгіршим[31].

Висновки до розділу 3.

У третьому розділі було представлено практичну частину в частині застосування ACO для оптимізації кількості ознак та наведено порівняння результатів роботи цього алгоритму з іншими поширеними методами оптимізації ознак в системах виявлення вторгнень. За результатами аналізу було висвітлено результати дослідження, а саме: порівняно результати

виконання без оптимізації ознак за допомогою ACO та з оптимізацією і наведено огляд результатів оптимізації ознак декількох інших алгоритмів.

Також, було визначено релевантність використання алгоритму мурашиних колоній шляхом детального дослідження роботи самого алгоритму при виконанні задачі оптимізації вибору ознак в системах виявлення вторгнень та порівнянні та оцінюванні отриманих результатів із сімома іншими існуючими методами оптимізації вибору ознак в системах виявлення вторгнень завдяки методу ELECTRE III.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

4.1 Опис ідеї стартап-проєкту

Ідея стартапу полягає у створенні інноваційного методу для виявлення неправомірного доступу до інформаційних систем, заснованого на методі мурашиних колоній та продаж ліцензії на його використання. В основі методу лежить новаторський підхід до виявлення аномалій у мережевих запитах, що застосовує алгоритми мурашиних колоній для покращення процесу пошуку та оптимізації маршруту виявлення загроз, що дозволяє знизити ймовірність помилкових спрацьовувань і підвищити точність виявлення потенційних атак.

Таблиця 4.1 – Опис ідеї, напряму та вигоди стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Метод виявлення неправомірного доступу на основі мурашиних колоній	1. Інтеграція в антивірусні програми	Підвищення точності виявлення загроз, зниження ймовірності помилкових спрацьовувань, виявлення нових типів атак
	2. Використання у відділах кібербезпеки великих компаній	Підвищення рівня захисту ІТ-систем та даних, адаптація до нових загроз у реальному часі
	3. Підвищення ефективності існуючих систем виявлення загроз	Оптимізація процесу виявлення з використанням алгоритмів мурашиних колоній, що адаптуються до змін у мережах
	4. Рішення для малих та середніх підприємств	Доступність ефективних методів кібербезпеки для малих організацій з обмеженими ресурсами

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	Конкурент			
1.	Використання методу для виявлення аномалій	Застосування алгоритму мурашиних колоній	Використання класичних алгоритмів	Високі вимоги до обчислювальних ресурсів	Не всі алгоритми підходять для різних атак	Зменшення помилкових спрацьовувань і підвищення точності

Кінець Таблиці 4.2

2.	Швидкодія методу	Оптимізація маршруту виявлення загроз	Використання класичних методів дерева рішень	Складність оптимізації в умовах великих даних	У випадку середніх обсягів даних швидкодія схожа	Підвищення швидкодії за рахунок оптимізації процесу
3.	Точність виявлення атак	Використання аномалій у мережевих запитах	Не оцінює точність для складних атак	Залежність від якості вхідних даних	Точність схожа при простих сценаріях	Покращення точності навіть для складних мережевих атак

4.2 Технологічний аудит ідеї проєкту

У межах цього підрозділу було проведено аудит технологій, необхідних для реалізації ідеї стартапу, спрямованого на створення інноваційного методу виявлення неправомірного доступу до інформаційних систем із застосуванням алгоритмів мурашиних колоній. Для цього було оцінено існуючі технології, їх наявність і доступність. Результати аналізу зібрані у таблиці 4.3.

Таблиця 4.3 – Технологічні можливості здійснення ідеї

№ п/п	Ідея проєкту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення методу виявлення неправомірного доступу	Алгоритми мурашиних колоній (Ant Colony Optimization)	Технологія наявна, але потребує адаптації до завдань кібербезпеки	Доступна, потребує знань алгоритмів оптимізації
2	Виявлення вторгнень	Дата сет KDD Cup 99	Дата сет доступний у відкритому доступі	Доступний, використовується для навчання та тестування
3	Підвищення точності та зниження помилкових спрацьовувань	Адаптивне навчання, класифікація Random Forest Classifier	Наявна технологія, активно використовується у задачах класифікації	Доступна, реалізується за допомогою бібліотек Python
4	Оптимізація маршруту виявлення загроз	Створюються словники для зберігання найкращих метрик та найкращих вибраних ознак	Технологія можлива, але потребує розробки відповідної архітектури	Доступна, за умови програмування відповідного алгоритму

За результатами аналізу таблиці, технологічна реалізація проєкту можлива. Обрані технології, такі як алгоритми мурашиних колоній, класифікатор Random Forest та використання відкритого дата сету KDD Cup 99, є доступними. Крім того, вони активно використовуються на ринку для вирішення задач оптимізації, класифікації та виявлення нестандартних подій.

4.3 Аналіз можливостей запуску стартап-проєкту на ринок

У цьому підрозділі було проведено аналіз попиту, включаючи його наявність, динаміку розвитку ринку та обсяг попиту. Також були визначені ринкові можливості, які можуть бути використані під час імплементації проєкту, а також ринкові загрози, що можуть перешкодити успішній реалізації проєкту. Результати цього аналізу зібрані в таблиці 4.4.

Таблиця 4.4 – Характеристика потенційного ринку для проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	Невелика кількість великих компаній та стартапів, які займаються виявленням вторгнень
2	Загальний обсяг продаж, грн/ум.од	Потенційний ринок оцінюється в кілька мільярдів гривень завдяки зростанню кіберзагроз
3	Динаміка ринку (якісна оцінка)	Зростає, особливо у зв’язку з розвитком IT-інфраструктури та збільшенням кіберзагроз
4	Наявність обмежень для входу (вказати характер обмежень)	Високі початкові витрати на розробку та сертифікацію. Потрібен доступ до великих наборів даних
5	Специфічні вимоги до стандартизації та сертифікації	Вимоги до відповідності міжнародним стандартам інформаційної безпеки (ISO/IEC 27001)
6	Середня норма рентабельності в галузі (або по ринку), %	Висока (до 20–25%), особливо для компаній, які пропонують інноваційні рішення

За попереднім оцінюванням, ринок є привабливим для імплементації стартапу, оскільки:

- Динаміка ринку зростає. Постійне збільшення кіберзагроз стимулює попит на системи виявлення несанкціонованого доступу.

- Висока рентабельність. Показники прибутковості галузі перевищують банківський відсоток, що робить інвестиції перспективними.
- Рівень конкуренції помірний. Незважаючи на наявність великих гравців, є можливість зайняти нішу за рахунок інноваційних технологій (мурашині колонії).

Інформацію про потенційних клієнтів та їх групи було зібрано у Таблиці 4.5. Після визначення цих груп було проведено аналіз ринкового середовища, що включає таблиці факторів, які сприяють ринковому впровадженню проєкту, а також факторів, що можуть перешкоджати його реалізації (Таблиці 4.6 та 4.7).

Таблиця 4.5 – Потенційні клієнти продукту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Забезпечення безпеки інформаційних систем, виявлення вторгнень	Великі підприємства, державні органи, фінансові установи, IT-компанії	Орієнтація на високу надійність, відповідність стандартам безпеки, складність у впровадженні та потреба в підтримці	Висока точність, здатність до масштабування, відповідність стандартам безпеки (ISO, GDPR). Професійний підхід, технічна підтримка, досвід роботи з великими клієнтами
2	Автоматизація виявлення атак, зниження людського фактора	Малі та середні підприємства, стартапи у галузі кібербезпеки	Більш чутливі до ціни та часу впровадження, схильні до пошуку простих та економічно ефективних рішень	Простота інтеграції, знижена вартість, ефективність у реальному часі. Гнучкість у налаштуванні, зручність у використанні та підтримці
3	Оптимізація безпеки, зниження кількості помилкових спрацьовувань	Організації, що займаються критичними даними, медіакомпанії	Більша увага до точності та відсутності помилок у виявленні загроз, технічні вимоги до систем	Висока точність, адаптивність до нових загроз. Високі стандарти обслуговування, наявність підтримки 24/7

Кінець Таблиці 4.5

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
4	Протидія складним загрозам, виявлення нових типів атак	Великі корпорації, технологічні компанії	Орієнтація на інноваційні рішення та найсучасніші методи виявлення загроз	Потужність обробки, здатність до масштабування для досить об’ємних обсягів даних. Забезпечення високого рівня підтримки, технологічна експертиза

Таблиця 4.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зміни в нормативно-правовому середовищі	Нові закони або регулювання, що можуть вимагати змін у продуктах чи послугах	Моніторинг законодавства, своєчасна адаптація продукту до нових вимог
2	Технічні ризики	Невизначеність щодо стабільності та надійності використовуваних технологій	Інвестиції в тестування, оновлення програмного забезпечення, пошук нових технологій
3	Зміни в попиті на ринку	Зменшення попиту на специфічні технології в результаті зміни ринкових умов	Гнучкість у зміні продукту, введення нових функцій для підтримки попиту
4	Низька швидкість обробки великої кількості інформації	Прорахунок всіх варіантів та обробка великої кількості даних може спричинити затримки в роботі системи	Оптимізація алгоритмів обробки даних, використання потужніших обчислювальних ресурсів або розподілених систем

Таблиця 4.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Розвиток ринку кібербезпеки	Зростання попиту на рішення для захисту інформаційних систем через збільшення кількості кіберзагроз	Розширення пропозицій, маркетинг та вдосконалення продуктів для задоволення потреб

Кінець Таблиці 4.7

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
2	Інновації в технологіях	Поява нових інноваційних технологій для швидшого виявлення загроз	Інвестування в нові технології, інтеграція нових рішень до продукту
3	Вирощення усвідомлення безпеки	Підвищення обізнаності серед підприємств і урядів про важливість кібербезпеки	Залучення нових клієнтів через освітні кампанії та інформаційні ресурси
4	Адаптація до нових ринків	Розширення на нові ринки або географічні регіони, де попит на кібербезпеку зростає	Розробка локалізованих версій продукту, виведення на нові ринки

Таблиця 4.8 – Ступеневий аналіз ринкової конкуренції

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції (монополістична)	Ринок представлений багатьма гравцями, які пропонують схожі продукти з інноваційними елементами	Фокус на розробці унікальних функцій продукту, таких як адаптивні алгоритми та висока точність
2. За рівнем конкурентної боротьби (міжнародний)	Конкуренція відбувається між компаніями не тільки на національному, але й на міжнародному рівні, де є багато провідних гравців у сфері кібербезпеки	Орієнтація на відповідність міжнародним стандартам, адаптація продукту до різних ринків, вихід на нові географічні регіони
3. За галузевою ознакою (внутрішньогалузева)	Конкуренція між компаніями, що спеціалізуються на розробці методів виявленні загроз у сфері кібербезпеки	Розробка продукту, який перевершує конкурентів за точністю, швидкістю обробки та зручністю
4. Конкуренція за видами товарів (товарно-видова)	Конкуренція між продуктами, що пропонують схожі методи виявлення загроз	Розширення функціоналу продукту: інтеграція з іншими системами, підвищення гнучкості у налаштуваннях

Кінець Таблиці 4.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5. За характером конкурентних переваг (нецінова)	Ставка робиться на технологічні рішення, репутацію та рівень технічної підтримки, а не на ціну	Інвестування в інновації, розвиток технічної підтримки 24/7, побудова довгострокової довіри клієнтів
6. За інтенсивністю (марочна)	Велика увага приділяється тим методам, що показали себе в галузі та надійності продукту	Створення сильної маркетингової стратегії, PR-кампаній, участь у профільних виставках і конференціях

Для аналізу конкурентного середовища за моделлю 5 сил М. Портера заповнено таблицю 4.9. Розглядається стартап у контексті галузі кібербезпеки, використовуючи специфіку проєкту.

Таблиця 4.9 - Аналіз галузевої конкуренції за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Компанії, що пропонують рішення у сфері кібербезпеки, зокрема: Symantec, McAfee, Palo Alto Networks	Значні початкові інвестиції у розробку	Постачальники обчислювальних ресурсів та хмарних платформ	Організації різних рівнів, включаючи державні установи, фінансові компанії	Існуючі IDS/IPS-рішення, які використовують традиційні підходи до виявлення загроз
Висновки	Інтенсивність боротьби за вихід на ринок середня, зважаючи на наявність сильних гравців	Можливість виходу на ринок висока, бар'єри є подоланими	Вплив постачальників помірний	Вплив клієнтів високий через високі вимоги до продуктивності та інтеграції	Обмеження з боку товарів-замінників низькі, оскільки товари-замінники не можуть забезпечити таку ж ефективність

На основі проведеного аналізу конкурентного середовища та умов конкуренції в галузі можна зробити висновок, що проєкт має реальні можливості для успішного виходу та розвитку на ринку. Основними сильними сторонами проєкту є його інноваційний підхід до вирішення завдань

кібербезпеки, зокрема використання алгоритмів мурашиних колоній, а також орієнтація на потреби сучасних компаній із високими вимогами до захисту даних.

Відсутність прямих конкурентів на ринку вказує на перспективність проєкту, тоді як наявність потенційних конкурентів та товарів-замінників визначає необхідність постійного вдосконалення технологій. Розширення ринку кібербезпеки, особливо з урахуванням зростання кількості загроз у сфері IoT, сприяє збільшенню кількості потенційних клієнтів.

Для забезпечення конкурентоспроможності проєкт повинен пропонувати високу ефективність рішень, зручність інтеграції та відповідати очікуванням споживачів, як зазначено у таблиці 4.9. Подальший аналіз конкурентоспроможності, сильних та слабких сторін проєкту проведено у відповідних таблицях 4.10 та 4.11.

Таблиця 4.10 - Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)**
1	Інноваційність технології	Використання унікального алгоритму аналізу загроз, зокрема алгоритму мурашиних колоній, дозволяє швидше і точніше виявляти кібератаки, що надає продукту перевагу над існуючими рішеннями конкурентів
2	Гнучкість у налаштуванні	Можливість адаптації продукту під різні потреби клієнтів (малий, середній і великий бізнес), забезпечуючи індивідуальний підхід та інтеграцію з наявними системами безпеки
3	Ефективність роботи	Проєкт забезпечує пониження кількості хибних спрацювань та покращує виявлення реальних загроз, що підвищує загальну надійність системи кіберзахисту
4	Захист від замінників	Відсутність на ринку аналогів, які використовують аналогічну технологію виявлення загроз, забезпечує унікальність проєкту та ускладнює конкуренцію з боку замінників

Таблиця 4.11 – Порівняльний аналіз сторін проєкту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з CrowdStrike					
			–3	–2	–1	0	+1	+2
1	Інноваційність технологій (алгоритми мурашиних колоній)	20			+			
2	Точність виявлення загроз	18						+
3	Можливість адаптації до різних галузей	17						
4	Швидкість обробки даних	15					+	
5	Вартість впровадження	14						+

Фінальним етапом аналізу ринкових можливостей для реалізації стартап-проєкту став SWOT-аналіз, побудований на основі ідентифікованих ринкових небезпек та можливостей, а також виявлених сильних і слабких сторін. SWOT-аналіз передбачає визначення сильних і слабких сторін, а також потенційних можливостей і ризиків. Результати аналізу представлені у таблиці 4.12.

Таблиця 4.12 – Порівняльний аналіз сторін проєкту

Сильні сторони	Слабкі сторони
Унікальність алгоритмів (мурашині колонії). Низька конкуренція в сегменті. Потенційно нижча ціна порівняно з лідерами ринку.	Обмежена інтеграція з популярними платформами. Високі обчислювальні вимоги до алгоритмів. Відсутність досвіду на ринку кібербезпеки.
Можливості	**Загрози**
Розширення в нові галузі завдяки адаптивності технологій. Зростання попиту на ефективні засоби кіберзахисту. Можливість отримання інвестицій у сфері кібербезпеки.	Конкуренція зі сторони великих компаній. Ризик недовіри до нових гравців ринку. Швидкі зміни в технологіях і ринку.

Розроблено альтернативні рішення моделювання ринку, на основі проведеного SWOT-аналізу. Результати подано у таблиці 4.13.

Таблиця 4.13 – Альтернативи для виходу на ринок

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Швидке тестування MVP (мінімально життєздатного продукту)	Висока (80-90%)	3–6 місяців
2	Стратегія партнерства з великими компаніями (інтегратори рішень)	Середня (60-70%)	6–12 місяців
3	Акцент на інноваційності технологій (PR, конференції)	Середня (50-60%)	3–9 місяців
4	Цінова стратегія для залучення середнього сегмента ринку	Висока (70-80%)	6 місяців після запуску MVP
5	Локалізація рішень для різних галузей	Середня (60%)	9–18 місяців
6	Інвестиції в розвиток клієнтської підтримки	Середня (50-70%)	12–24 місяці
7	Моніторинг ринку та адаптація стратегії	Низька (30-40%)	Неперервно

4.4 Розроблення ринкової стратегії проєкту

Розробка ринкової стратегії для стартапу передбачає насамперед, визначення стратегії охоплення ринку, що включає опис цільових груп потенційних споживачів. Цей етап був реалізований у Таблиці 4.14.

Таблиця 4.14 – опис цільових груп клієнтів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах груп	Інтенсивність конкуренції в секторі	Простота входу
1	Малі та середні підприємства (МСП) у сфері кібербезпеки	Висока (потреба в доступних рішеннях для кібербезпеки)	Середній, зростаючий попит	Середня (є конкуренти, але мало специфічних рішень)	Середня (високий інтерес, але потрібна інфраструктура)
2	Великі компанії в фінансовому секторі	Середня (потрібні високоякісні та перевірені рішення)	Високий попит на інновації в кібербезпеці	Висока (багато великих гравців, таких як CrowdStrike)	Низька (високі вимоги до продукту та інвестицій)

Кінець таблиці 4.14

3	Стартапи та інноватори в технологічній сфері	Висока (готові інвестувати в нові рішення)	Середній попит, стабільний	Низька (мало гравців у цьому сегменті)	Середня (необхідні стратегічні партнерства)
4	Клієнти в медичному секторі (охорона здоров’я)	Середня (потрібна спеціалізована безпека для медичних даних)	Середній, зростаючий попит	Середня (великі гравці та спеціалізовані рішення)	Низька (високі бар’єри для входу)
5	Освітні установи та університети	Висока (потреба в захисті від кіберзагроз)	Низький попит, але можливий ріст	Низька (менше конкурентів у цьому секторі)	Середня (потрібна адаптація до специфіки)

Таблиця 4.15 - Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Диференціація	Охоплення специфічних ринкових сегментів, зокрема великі компанії в сфері кібербезпеки, МСП та технологічні стартапи	Унікальність технології (алгоритми мурашиних колоній), інноваційний підхід до виявлення загроз, висока точність	Стратегія диференціації

Оскільки стартап пов'язаний із використанням інноваційних алгоритмів мурашиних колоній для виявлення неправомірного доступу до систем, стратегія диференціації здається найбільш релевантною для розвитку проєкту. Вона дозволяє виділитися серед конкурентів за рахунок унікальних властивостей продукту, а також створити міцний бренд.

Таблиця 4.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проєкт «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так (використовує унікальні алгоритми мурашиних колоній для виявлення загроз)	Шукати нових споживачів, орієнтуючись на малий та середній бізнес, стартапи та галузі з високим попитом на кібербезпеку	Не буде копіювати характеристики товарів конкурентів, а пропонуватиме унікальну технологію	Стратегія «першопрохідця» з фокусом на інноваційну диференціацію

Стратегія заняття конкурентної ніші (Нішер) - ця стратегія передбачає фокусування на конкретних сегментах ринку, які потребують специфічних рішень в сфері кібербезпеки, таких як малий та середній бізнес, державні установи, або навіть певні індустрії, які ще не мають достатньо захищених IDS.

Таблиця 4.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
1	Висока точність детекції загроз, мінімізація помилкових спрацьовувань	Стратегія диференціації	Інноваційні алгоритми мурашиних колоній для точного виявлення загроз; спеціалізація на кібербезпеці для малих і середніх підприємств	1. Інноваційність (передова технологія, що вирізняється серед конкурентів) 2. Надійність (висока точність виявлення загроз) 3. Доступність (рішення для малих і середніх підприємств)
2	Гнучкість у налаштуваннях і інтеграції з існуючими системами	Стратегія диференціації	Легкість інтеграції в наявні IT-системи; гнучкість і адаптивність до потреб різних галузей	1. Адаптивність (можливість налаштування під різні потреби клієнтів) 2. Сумісність (легкість інтеграції з існуючими системами) 3. Індивідуальний підхід

Кінець таблиці 4.17

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проєкту	Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)
3	Доступність і підтримка для малого і середнього бізнесу	Стратегія диференціації	Продукти, доступні для малого та середнього бізнесу, зручні у використанні, з доступною технічною підтримкою	1. Доступність (ціна та ефективність для малих і середніх підприємств) 2. Простота (зручність у використанні) 3. Підтримка (технічна підтримка та навчання користувачів)

4.5 Розроблення маркетингової програми проєкту

У цьому підрозділі за допомогою даних, наведених у таблицях, було описано процес розробки та формування концепції кінцевого продукту (товару), що базується на проаналізованих аспектах його конкурентоспроможності.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Висока точність виявлення загроз	Покращена безпека завдяки мінімізації помилкових спрацьовувань	Унікальність технології (алгоритми мурашиних колоній), яка забезпечує цілком прийнятну точність виявлення навіть складних загроз
2	Адаптивність до різних ІТ-систем	Легкість інтеграції в існуючі ІТ-структури клієнтів	Гнучкість налаштувань і інтеграції з різними системами безпеки, що відрізняє продукт від конкурентів з обмеженою сумісністю
3	Забезпечення постійного вдосконалення	Регулярні оновлення та вдосконалення алгоритмів	Постійний розвиток технології, зокрема адаптація до нових загроз та автоматичне оновлення, що забезпечує стабільний рівень безпеки

Таблиця 4.19 - Опис 3 рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Ліцензія на використання продукту для забезпечення ефективної кібербезпеки для бізнесу через виявлення та блокування несанкціонованого доступу до інформаційних систем.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Алгоритм виявлення загроз на основі мурашиних колоній. 2. Інтуїтивна модель для адміністраторів системи.		
	Якість: відповідає міжнародним стандартам кібербезпеки (ISO/IEC 27001, GDPR).		
	Пакування: електронний продукт у вигляді програмного забезпечення.		
	Марка: Назва стартапу ("Ант Секьюріті").		
III. Товар із підкріпленням	До продажу: - Безкоштовна консультація щодо впровадження продукту у систему клієнта. - Демоверсія з обмеженим функціоналом для тестування.		
	Після продажу: - Технічна підтримка. - Регулярні оновлення для підвищення ефективності системи та адаптації до сучасних викликів.		
Захист від копіювання через патентування алгоритму, ліцензійну угоду, використання унікальних шифрувальних ключів.			

Таблиця 4.20 - Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	Прості сигнатурні методи для визначення вторгнень, надані open-source платформами (0–5 тис. $/рік за інтеграцію).	Ліцензії на сучасні методи виявлення вторгнень (машинне навчання) - від компаній, як-от Kaspersky, CrowdStrike (10–50 тис. $/рік).	Антивірусні компанії малого та середнього рівня з доходами від 500 тис. до 5 млн $/рік.	Нижня межа: 1 тис. $/рік; Верхня межа: 3 тис. $/рік

Кінець таблиці 4.20

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
2	Open-source компоненти для інтеграції з обмеженим функціоналом (до 10 тис. $ одноразово)	Комерційні ліцензії для корпоративного рівня на основі патентованих технологій - від Symantec, Palo Alto (50–200 тис. $/рік).	Лідери ринку антивірусів із доходами понад 50 млн $/рік.	Нижня межа: 5 тис. $ одноразово; Верхня межа: 15 тис. $ одноразово

Таблиця 4.21 - Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Клієнти (антивірусні компанії) обирають продукти, які легко інтегруються у їхні існуючі системи, з наданням технічної документації та супровідного обслуговування.	Надання ліцензії, технічної підтримки, інтеграційних консультацій, оновлення.	Прямий канал (без посередників).	Власна система збуту через прямий контакт із клієнтами (B2B продажі).
2	Великі компанії можуть використовувати посередників (партнерські інтегратори), які допомагають інтегрувати технології до їх рішень.	Ліцензування через партнерів, навчання інтеграторів, технічна підтримка партнерів.	Дворівневий канал (через партнерів).	Залучена система збуту через партнерські мережі (інтегратори).

Таблиця 4.22 - Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Антивірусні компанії шукають технології, які підвищують точність і швидкість виявлення вторгнень. Високо цінують інноваційність, надійність та легкість інтеграції.	Галузеві конференції (Black Hat, RSA), спеціалізовані блоги, аналітичні огляди, вебінари, професійні форуми, LinkedIn.	Високоточна інноваційна технологія з простою інтеграцією у вже існуючі системи.	Підкреслити ефективність, інноваційність і конкурентні переваги методу.	"Ваша конкурентна перевага - найкращий алгоритм виявлення вторгнень! "
2	Великі антивірусні компанії співпрацюють із партнерами для впровадження нових технологій. Рішення приймаються після ознайомлення з технічними кейсами.	Партнерські зустрічі, персональні демо-презентації, технічні статті, професійні книги, виставки.	Гнучкість використання та інтеграції; економія ресурсів на розробку власного алгоритму.	Переконати у доцільності ліцензування технології замість розробки власної.	"Розробіть більше, витрачаючи менше: довірте виявлення загроз нашій технології! "

Висновки до розділу 4

На основі проведеного аналізу проєкту, який передбачає створення ліцензії на використання антивірусними компаніями методу визначення вторгнень на основі алгоритмів мурашиних колоній, зроблено такі узагальнення:

Проведений аналіз показує, що попит на інноваційні технології кібербезпеки є високим, особливо серед антивірусних компаній, які прагнуть підвищити ефективність і точність своїх систем. Ринок рішень у сфері кібербезпеки демонструє стабільне зростання, що вказує на рентабельність роботи на цьому ринку.

Перспективи впровадження:

- Потенційні групи клієнтів - це антивірусні компанії, які потребують нових технологій для поліпшення своїх продуктів, та інтегратори, які впроваджують складні рішення для корпоративного ринку.
- Основні бар’єри входження: висока конкуренція та необхідність створення довіри до нової технології.
- Стан конкуренції: ринок містить визнаних лідерів, однак інноваційні методи, які можуть забезпечити перевагу в точності і швидкості роботи, мають шанси на успіх за умови ефективного позиціонування.
- Конкурентоспроможність проєкту забезпечується за рахунок використання унікального алгоритму, його простоти інтеграції та економічної ефективності.

Доцільно обрати модель прямого ліцензування для середніх компаній із подальшою співпрацею через партнерські мережі для великих клієнтів. Такий підхід дозволить ефективно охопити різні сегменти ринку. Проєкт має перспективи успішної ринкової реалізації за умови подальшої оптимізації маркетингової програми, залучення професійних каналів комунікації для просування та проведення детального фінансово-економічного аналізу. Технологія є релевантною в умовах сучасного ринку, що є підставою для подальшої імплементації та масштабування.

Таким чином, проєкт рекомендовано до подальшого впровадження та комерціалізації.

ВИСНОВКИ

Поставлені завдання виконані у повному обсязі, було проаналізовані евристичні алгоритми та досліджено можливість застосування мурашиного алгоритму для задач виявлення несанкціонованого доступу. Актуальність роботи обумовлена відсутністю еталонного вирішення задачі побудови IDS, а тому, необхідності додаткового дослідження шляхів покращення роботи, в тому числі у шляхом оптимізації вибору ознак.

У першому розділі розглядаються загальні відомості про IDS, а саме: опис, класифікацію та структуру, наведено відомості про атаки несанкціонованого доступу та розглянуто найпоширеніші методи оптимізації ознак в системах виявлення вторгнень.

Результатом виконання другого розділу був розгляд класифікації евристичних алгоритмів, їх опису та окремих прикладів. Увагу акцентовано саме на алгоритмах ройового інтелекту, їх різновидах та прикладах, а саме, алгоритмі мурашиних колоній (ACO). Описані загальні принципи роботи мурашиного алгоритму, представлені схеми та таблиці, що відображають його роботу. Також представлено алгоритм здійснення відбору ознак для системи виявлення вторгнень за допомогою мурашиного алгоритму.

У третьому розділі було представлено практичну частину в частині застосування ACO для оптимізації кількості ознак та наведено порівняння результатів роботи цього алгоритму з іншими поширеними методами оптимізації ознак в системах виявлення вторгнень. За результатами аналізу було висвітлено результати дослідження, а саме: порівняно результати виконання без оптимізації ознак за допомогою ACO та з оптимізацією і наведено огляд результатів оптимізації ознак декількох інших алгоритмів.

Також, було визначено релевантність використання алгоритму мурашиних колоній шляхом детального дослідження роботи самого алгоритму при виконанні задачі оптимізації вибору ознак в системах виявлення вторгнень та порівнянні та оцінюванні отриманих результатів із

сімома іншими існуючими методами оптимізації вибору ознак в системах виявлення вторгнень завдяки методу ELECTRE III.

У четвертому розділі було проведено дослідження можливості реалізації стартап-проєкту на основі запропонованої ідеї. Виконано аналіз за різними критеріями, що дозволило оцінити доцільність подальшої розробки. На основі отриманих результатів зроблено висновок про перспективність створення програмного продукту, який використовуватиме алгоритм мурашиних колоній для оптимізації ознак з метою його інтеграції в системи виявлення вторгнень.

Отже, підводячи підсумки, можемо сказати, що алгоритм мурашиних колоній має потенціал для використання у задачах виявлення несанкціонованого доступу, але результати емпіричних досліджень показують, що при обробці великих об’ємів даних він поступається еталонним алгоритмам, заснованих на основі дерев рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Mehdi Hosseinzadeh Aghdam. Feature Selection for Intrusion Detection System Using Ant Colony Optimization [Електронний ресурс] / Mehdi Hosseinzadeh Aghdam, Peyman Kabiri // International Journal of Network Security. – 2016. – Режим доступу: http://ijns.jalaxy.com.tw/contents/ijns-v18-n3/ijns-2016-v18-n3-p420-432.pdf..

2 Scarfone K. A. Guide to Intrusion Detection and Prevention Systems (IDPS) [Електронний ресурс] / K. A. Scarfone, P. M. Mell. – Gaithersburg, MD : National Institute of Standards and Technology, – 2007. – Режим доступу: https://doi.org/10.6028/nist.sp.800-94.

3 Мєшков, В. І. Аналіз сучасних систем виявлення та запобігання вторгнень в інформаційно-телекомунікаційних системах [Електронний ресурс] / В. І. Мєшков, В. О. Віролайнен // Матеріали III Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» – 2015. – Режим доступу: https://ela.kpi.ua/handle/123456789/17609.

4 P W. A Survey of Intrusion Detection System [Електронний ресурс] / Wanda P // International Journal of Informatics and Computation. – 2020. – Vol. 1, no. 1. – P. 1. – Режим доступа: https://doi.org/10.35842/ijicom.v1i1.7.

5 STALLINGS W. Network Security Essentials: Applications and Standards, Global Edition / WILLIAM STALLINGS. – [S. l.] : PEARSON, 2016.

6 Scarfone K. Intrusion Detection and Prevention Systems [Електронний ресурс] / Karen Scarfone, Peter Mell // Handbook of Information and Communication Security. – Berlin, Heidelberg, 2010. – P. 177–192. – Режим доступа: https://doi.org/10.1007/978-3-642-04117-4_9.

7 Yao-Min Chen. Policy management for network-based intrusion detection and prevention [Електронний ресурс] / Yao-Min Chen, Yanyan Yang // 2004 IEEE/IFIP Network Operations and Management Symposium, Seoul, South Korea. – [S. l.]. – Режим доступа: https://doi.org/10.1109/noms.2004.1317855.

8 Carl Endorf. Intrusion Detection & Prevention / Carl Endorf, Jim Mellander, Eugene Schultz., 2004. – 386 с. – (McGraw Hill Professional).

9 Mitrokotsa, A., Dimitrakakis, C., & Douligeris, C. (2011). Intrusion detection with cost-sensitive classification. International Journal of Computer Applications, 19(4), 12-18.

10 Іщенко А.А., Висоцький І.С. «Погляд на правові особливості і проблеми застосування штучного інтелекту в сфері» – К.: КПІ ім. Ігоря Сікорського, 2021. – 192 с. – С. 116.

11 Самигулина Г. А., Масимканова Ж. А. Обзор современных методов роевого интеллекта для компьютерного молекулярного дизайна лекарственных препаратов. Алма-Ата: Институт информационных и вычислительных технологий, 2016. – 51-57 с.

12 Schneier B. Applied cryptography. 2nd ed., / Schneier B.1., 1996. – 784 с. – (International Journal of Network Security).

13 Colorni, Alberto. Distributed Optimization by Ant Colonies. [Електронний ресурс] / Colorni, Alberto, Dorigo, Marco, Maniezzo, Vittorio. // Proceedings of the First European Conference on Artificial Life. – 1991. – Режим доступу: https://www.researchgate.net/publication/216300484_Distributed_Optimization_by_Ant_Colonies.

14 The Elements of Statistical Learning: Data Mining, Inference, and Prediction / ed. by H. Trevor, T. Robert, S. (. service). – New York, NY : Springer-Verlag New York, 2009. – 745 p.

15 Guyon, I., & Elisseeff, A. (2003). "An introduction to variable and feature selection." *Journal of Machine Learning Research*, 3, 1157-1182

16 Breiman, L. (2001). "Random Forests." *Machine Learning*, 45(1), 5-32.

17 Vishwakarma S. An Intrusion Detection System using KNN-ACO Algorithm [Електронний ресурс] / Satyendra Vishwakarma, Vivek Sharma, Ankita Tiwari // International Journal of Computer Applications. – 2017. – Vol. 171, no. 10. – P. 18–23. – Режим доступу: https://doi.org/10.5120/ijca2017914079

18 Tabachnick, Barbar. Fidell Using Multivariate Statistics / Tabachnick, Barbar, Linda, S.F. // Pearson. – 2001.

19 Field A. BUNDLE : Field : Discovering Statistics Using IBM SPSS Statistics + Cunningham: Using SPSS / Andy Field. – [S. l.] : SAGE Publications, Incorporated – 2013.

20 Порядок вибору заходів захисту інформації, вимога щодо захисту якої встановлена законом та не становить державної таємниці, для інформаційних систем НД ТЗІ 3.6-006-24, Київ, Адміністрація Державної служби спеціального зв'язку та захисту інформації України 2024 – 51-57 с. 429

21 Іщенко А. А. The relevance of the ant algorithm as a tool for cryptanalysis / Іщенко А. А., Кубайчук О. О.. // Науково-практична конференція студентів, курсантів, аспірантів, докторантів та молодих вчених «Актуальні питання застосування інформаційно-телекомунікаційних систем». – 2021. – №1. – с. 50.

22 Taherdoost H. Multi-Criteria Decision Making (MCDM) Methods and Concepts [Електронний ресурс] / Hamed Taherdoost, Mitra Madanchian // Encyclopedia. – 2023. – Vol. 3, no. 1. – P. 77–87. – Режим доступу: https://doi.org/10.3390/encyclopedia3010006.

23 Galchynsky, L., Graivoronskyi, M., & Dmytrenko, O. (2021). Evaluation of Machine Learning Methods to Detect DoS / DDoS Attacks on IoT.CEUR Workshop Proceedings, 3241, 225–236.

24 Тостоган, Є. Г., Гальчинський, Л. Ю. Вибір інструментів оцінювання кіберризиків для організацій на основі багатокритеріального аналізу // XXII Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених теоретичні і прикладні проблеми фізики, математики та інформатики, (22), 176–178. – 2021. – Режим доступу до ресурсу: https://ela.kpi.ua/handle/123456789/69948.

25 Huan Liu. Toward integrating feature selection algorithms for classification and clustering [Електронний ресурс] / Huan Liu, Lei Yu // IEEE Transactions on

Knowledge and Data Engineering. – 2005. – Vol. 17, no. 4. – P. 491–502. – Режим доступу: https://doi.org/10.1109/tkde.2005.66.

26 Enhanced Feature Selection Using Genetic Algorithm for Machine-Learning-Based Phishing URL Detection [Електронний ресурс] / Emre Kocyigit [et al.] // Applied Sciences. – 2024. – Vol. 14, no. 14. – P. 6081. – Режим доступу: https://doi.org/10.3390/app14146081.

27 AC/35-D/2001-REV3. Directive on Physical Security. NATO Unclassified

28 AC/35-D/2000-REV8. Directive on Personel Security. NATO Unclassified

29 A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions [Електронний ресурс] / Ömer Aslan [et al.] // Electronics. – 2023. – Vol. 12, no. 6. – P. 1333. – Режим доступу: https://doi.org/10.3390/electronics12061333.

30 Taherdoost H. A Comprehensive Overview of the ELECTRE Method in Multi Criteria Decision-Making [Електронний ресурс]/ Hamed Taherdoost, Mitra Madanchian // Journal of Management Science & Engineering Research. – 2023. – Vol. 6, no. 2. – Режим доступу: https://doi.org/10.30564/jmser.v6i2.5637.

31 Іщенко А. А., Гальчинський, Л. Ю. Оцінювання релевантності методу мурашиних колоній (ACO) для вирішення використання в системах виявлення вторгнення. [Електронний ресурс] / Collection of Scientific Papers «ΛΌГОΣ», (November 15, 2024; Bologna, Italy), 160–169. Режим доступу: https://archive.logos-science.com/index.php/conference-proceedings/issue/view/29/29.

Додаток А

Текст програми виконання оптимізації ознак в системі виявлення вторгнень за допомогою алгоритму мурашиних колоній на основі дата сету KDD99

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metr import accuracy_scr, precision_scr, recall_scr, f1_scr


data = pd.read_csv('kddcup.data, header=None)
column_names = [
"duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes",
"land", "wrong_fragment", "urgent", "hot", "num_failed_logins", "logged_in",
"num_compromised", "root_shell", "su_attempted", "num_root",
"num_file_creations",
"num_shells", "num_access_files", "num_outbound_cmds", "is_host_login",
"is_guest_login", "count", "srv_count", "serror_rate", "srv_serror_rate",
"rerror_rate", "srv_rerror_rate", "same_srv_rate", "diff_srv_rate",
"srv_diff_host_rate", "dst_host_count", "dst_host_srv_count",
"dst_host_same_srv_rate",
"dst_host_diff_srv_rate", "dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate",
"dst_host_serror_rate", "dst_host_srv_serror_rate", "dst_host_rerror_rate",
"dst_host_srv_rerror_rate", "label" ]
data.columns = column_names
print(data.info())
print(data.describe())
```

```
def custom_encode(df):
    for col in ['protocol_type', 'service', 'flag']:
        most_frequent = df[col].value_counts().idxmax()
        df[col] = (df[col] == most_frequent).astype(int)
    return df


data_encoded = custom_encode(data)


data_encoded['label'] = data_encoded['label'].map(lambda x: 1 if x == 'normal.'
else 0)
sns.countplot(x='label', data=data_encoded, palette='Set2')
plt.title('Distribution of Intrusion Labels in KDD99 Dataset')
plt.xticks(rotation=90)
plt.show()
def evaluate_model(y_test, y_pred):
    accuracy = accuracy_scr(y_test, y_pred)
    precision = precision_scr(y_test, y_pred, average='weighted', zero_division=0)
    recall = recall_scr(y_test, y_pred, average='weighted', zero_division=0)
    f1 = f1_scr(y_test, y_pred, average='weighted', zero_division=0)
    return accuracy, precision, recall, f1
class ACO:
    def __init__(slf, n_murah, n_features, n_itratsyi, alpha, beta, evapt_rate):
        slf.n_murah = n_murah
        slf.n_features = n_features
        slf.n_itratsyi = n_itratsyi
        slf.alpha = alpha
        slf.beta = beta
        slf.evapt_rate = evapt_rate
        slf.pheromone = np.full(n_features, 0.1)
```

```
    def select_features(slf):
        feature_selection = []
        for _ in range(slf.n_murah):
            probabilities = slf.pheromone ** slf.alpha
            probabilities /= np.sum(probabilities)
            n_selected = np.random.randint(1, slf.n_features + 1)
            selected_indices = np.random.choice(range(slf.n_features), n_selected,
replace=False, p=probabilities)
            selected_features = np.zeros(slf.n_features, dtype=bool)
            selected_features[selected_indices] = True
            feature_selection.append(selected_features)
        return np.array(feature_selection)
    def update_pheromone(slf, accuracy, selected_features):
        slf.pheromone[selected_features] += accuracy * (1 - slf.evapt_rate)
        slf.pheromone *= (1 - slf.evapt_rate)
        slf.pheromone = np.clip(slf.pheromone, 0, None)
    def optimize(slf, X, y):
        best_metr = {'accuracy': 0, 'precision': 0, 'recall': 0, 'f1': 0}
        best_features = None
        all_metr = []
        for iteration in range(slf.n_itratsyi):
            feature_selection = slf.select_features()
            metr = []
            for features in feature_selection:
                X_selected = X.loc[:, features]
                X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
test_size=0.3, random_state=42)
                classifier = RandomForestClassifier()
                classifier.fit(X_train, y_train)
```

```
            y_pred = classifier.predict(X_test)
            metr.append(evaluate_model(y_test, y_pred))
        metr = np.array(metr)
        avg_metr = np.mean(metr, axis=0)
        slf.update_pheromone(avg_metr[0], feature_selection[np.argmax(metr[:,
0])])
        print(f'Iteration {iteration + 1}:')
        print(f'Accuracy: {avg_metr[0]}')
        print(f'Precision: {avg_metr[1]}')
        print(f'Recall: {avg_metr[2]}')
        print(f'F1 Score: {avg_metr[3]}')
        print(f'K obranyh oznak: {np.sum(feature_selection[np.argmax(metr[:,
0])])}\n')
        all_metr.append(avg_metr)
        if avg_metr[0] > best_metr['accuracy']:
            best_metr['accuracy'] = avg_metr[0]
            best_metr['precision'] = avg_metr[1]
            best_metr['recall'] = avg_metr[2]
            best_metr['f1'] = avg_metr[3]
            best_features = feature_selection[np.argmax(metr[:, 0])]
    return best_features, best_metr, np.array(all_metr)
def plot_pheromone(slf, feature_names):
    plt.figure(figsize=(10, 6))
    ax = sns.barplot(x=feature_names, y=slf.pheromone, palette='viridis')
    plt.title('Pheromone Concentration Matrix')
    plt.xlabel('Feature')
    plt.ylabel('Pheromone Level')
    plt.xticks(rotation=90)
    for p in ax.patches:
```

```
ax.annotate(f'{p.get_height():.2f}', (p.get_x() + p.get_width() / 2., p.get_height()),
ha='center', va='bottom')
plt.show()
X = data_encoded.drop('label', axis=1)
y = data_encoded['label']
aco = ACO(n_murah=10, n_oznak=X.shape[1], n_itratsyi=50, alph=1.0, bet=1.0,
evap_rate=0.4)
best_features, best_metr, all_metr = aco.optimize(X, y)
print(f'Best metr:')
print(f'Accuracy: {best_metr["accuracy"]}')
print(f'Precision: {best_metr["precision"]}')
print(f'Recall: {best_metr["recall"]}')
print(f'F1 Score: {best_metr["f1"]}')
print(f'K obranyh oznak:  {np.sum(best_features)}')
classifier = RandomForestClassifier()
classifier.fit(X.loc[:, best_features], y)
importances = classifier
classifier = RandomForestClassifier()
classifier.fit(X.loc[:, best_features], y)
importances = classifier.feature_importances_
feature_importances_df = pd.DataFrame({
    'Feature': X.columns[best_features],
    'Importance': importances
}).sort_values(by='Importance', ascending=False)
print(feature_importances_df)


plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importances_df,
palette='viridis')
```

```
plt.title('Feature Importance from RandomForest')
plt.show()
aco.plot_pheromone(X.columns)
```